



City Research Online

City St George's, University of London

Citation: Comley, R.A. (1978). Portable computers for real-time signal processing: EEG analysis as a case study. (Submitted Doctoral thesis, City University, London)

This is the published version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: <https://city-ref-test.eprints-hosting.org/id/eprint/18587/>

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to [City Research Online policy](#).

PORTABLE COMPUTERS FOR REAL-TIME
SIGNAL PROCESSING: EEG ANALYSIS
AS A CASE STUDY
(VOLUME I)

R.A.COMLEY

Thesis presented for the degree of
Doctor of Philosophy
of the City University, London

1978

ABSTRACT

Recent advances in both digital hardware and digital signal theory have led to a rapid expansion in the importance and application of computer-aided measurement (CAM) techniques. Of these advances, the emergence of cheap microprocessor technology of sufficient processing power and speed to support some of the real-time signal-processing tasks encountered in CAM, is probably the single most important factor.

The Roving Slave Processor (RSP) represents a novel extension to the field of CAM. The RSP is a basic hardware unit comprising, in its simplest form, a central processor, a memory system and a means of input-output. By the use of a microprocessor, it is possible to reduce the size of the complete system to very small dimensions, i.e. to construct a portable computer.

The unit is wholly dependent upon a master computer for the provision of all fundamental peripherals (e.g. teletype, reader-punch, etc.) and for all program preparation. To provide these facilities, a special purpose interface has been constructed. The RSP is, however, capable of disconnected operation and this is shown to lead to a very efficient and economical means by which to perform CAM operations.

The design and development of two prototypes is described with particular attention being given to the choice of processor, the storage system and the link to the master computer. Some consideration has also been given to the problem of how the RSPs should be programmed and a scheme based on a high-level calling system is detailed. Problems of reliability, both hardware and software, are also discussed.

An application of the RSP technique in the very demanding field of real-time EEG analysis is described,

with particular attention given to the development of an automatic spike detector algorithm. The occurrence of spikes in the EEG signal is of particular clinical significance as it is indicative of the onset of an epileptic attack. Sharp-waves, slow-waves and all other abnormal behaviour have been omitted from this study.

A system, based on a filtered, first-order difference of the EEG signal has been developed and is described. Very encouraging results have been obtained, with a 96% success rate for the abnormal spikes occurring in a series of test records.

Finally, techniques for the production of a miniature version of the RSP, which may be attached to and conveniently carried by a patient, are discussed.

CONTENTS

	<u>Page</u>
Abstract	i
Contents	iii
1. Introduction	1
2. Review	4
3. The Roving Slave Processor Concept	11
4. Practical Realization of the Roving Slave Processor	19
4.1 Introduction	20
4.2 Choice of Processor	21
4.3 Interface to Master Computer	35
4.4 Memory System	39
4.5 Input - Output Considerations	49
4.6 Power Supply Considerations	52
4.7 The Software System	57
4.8 Reliability	67
5. Nature of the Problem	70
6. Signal Processing Considerations	98
6.1 Mathematical Representation	101
6.2 Input - Output Considerations	104
6.2.1 The Sampling Process	105
6.2.1.1 Time Domain Sampling	106
6.2.1.2 Frequency Domain Sampling	107
6.2.2 The z-transform	108
6.2.3 Aliasing	110
6.2.4 Quantization	111
6.3 Effect of Errors on Proposed Signal- Processing Operations	114
6.3.1 Register Length	118
6.3.2 Limit-Cycle Behaviour	120
6.3.3 Parameter Quantization	122

	<u>Page</u>
6.4 Recursion Noise	128
7. Experimental Method and Results	130
7.1 Differentiation	131
7.2 Choice of Filter	137
7.3 Filter Design	141
7.4 Practical Implementation	146
7.5 EEG Results	154
7.5.1 Artifact Rejection	165
7.6 Concluding Remarks	169
8. Discussion and Suggestions for Further Work	170
8.1 Discussion	171
8.2 Suggestions for Further Work	176
9. Conclusions	182
10. Acknowledgements	185
11. Appendices	187
A. Publications	188
i) The Roving Slave Processor	189
ii) Relieving the Real-Time Signal Processing Load	194
iii) Super-tool: the microprocessor is revolutionising industry	199
iv) More bits, more power, more precision	203
v) Digital Filter Implementation by means of Slave Processors	206
B. Memory Technology Review	212
B.1 Introduction	213
B.2 Semiconductor Random Access Memories	213
B.3 Semiconductor Read Only Memories	220
B.4 Semiconductor, Non-Volatile Random Access Memory	223
B.5 Core Memory	225
B.6 Charge-Coupled Devices	228
B.7 Magnetic Bubble Memory	232

	<u>Page</u>
B.8 Cassette, Cartridge and Disc Stores	238
C. Memory Reliability	244
D. Program Listings	250
i) Autocorrelation Program	251
ii) Spike Detector Program	255
E. Error Calculations for the Second-Order	
Butterworth Filter	261
E.1 Introduction	262
E.2 Cut-Off Frequency Error	262
E.3 Amplitude Error	263
12. References	269

Chapter 1

Introduction

The purpose of the work described in this thesis is two-fold. The initial phases are concerned with the theoretical constraints, design philosophy and development of a portable digital computer system, suitable for real-time signal-processing applications. The latter phases are concerned with the use of these portable computers in the very demanding field of real-time analysis of abnormal electroencephalogram (EEG) signals.

The portable computers, known as Roving Slave Processors (RSPs), are very basic hardware units that depend upon a master computer for the provision of all basic peripherals. In their simplest form they comprise a central processor, program and data store and a means of input-output. The RSPs receive their programs from the master computer and, once loaded, they may be disconnected and transported to some required site for operation. In this manner they can be made to fill a whole variety of roles by the provision of a suitable program.

At the commencement of this project, microprocessors of sufficient speed and processing power, suitable for the RSP, were just becoming available. Two prototype devices, based on different microprocessors, have been developed and are described. One of these was eventually chosen as the vehicle for a study of the feasibility of providing the real-time EEG analysis, chosen as a target processing task. The random, non-stationary nature of the EEG signal make it a very demanding test case for the portable systems and provides a useful guideline as to their signal-processing potential and power.

The EEG case is of particular interest, in that ever since Hans Berger [1] showed that the electrical activity of the brain could be monitored by electrodes placed on the scalp, attempts have been made to link this activity with the activity of the underlying brain. To this day, there is

still much conjecture as to the usefulness of this gross, averaged and distorted signal, as to whether it conveys any meaningful information or is merely an interesting phenomenon and nothing more [2].

Berger, however, was not only the first to demonstrate the EEG phenomenon in man but was also the first to discover that it was abnormal in epilepsy. This, combined with the work of Walter [3] who established that slowly varying voltages arose near the tissue surrounding brain tumours which could thus be detected via the EEG, gave electroencephalography an air of respectability and led to its acceptance as a tool for clinical diagnosis.

In order to test the RSP principle in this field it was desirable to choose one demanding application, so it is in the detection of the abnormal activity occurring as a precursor to an epileptic attack that the main use of the RSPs has been directed. This abnormality usually manifests itself as high-frequency activity in the EEG and the eventual aim is to provide a means of detecting this activity as it occurs (i.e. in real-time).

The two main themes of the research are closely related, though this may not at first sight be evident. A major consideration when the design of the portable computer was embarked upon was the hope that technology would eventually permit the construction of a pocket-size, portable computer, which may be attached to, and conveniently carried by, a patient. This would permit the continuous monitoring of that patient and provide a means of automatically detecting, for example, any abnormal activity of the brain under different environmental conditions.

The thesis is divided into two volumes, with this volume, volume I, containing the main body of the research and volume II, technical details of the hardware and software developed.

Chapter 2

Review

During the past decade computer-aided measurement (CAM) has become a recognizable and well defined branch of applied science. As the name suggests, it involves the use of a computer to assist in the solution of a measurement problem. It often involves the computer in controlling and performing the whole measurement task.

There are two main reasons for this rapid development to the present level of importance and applicability, one economic and the other technical necessity. The basic underlying economic factor is that the computer, by confining speciality to software, enables extensive use to be made of its capital intensive hardware; i.e. the hardware can be effectively reconfigured to perform a whole range of different operations by the provision of a suitable program. This is not generally the case with specialized equipment, which tends to be designed for one specific application.

The technical necessity behind the rapid developments is simply that many of today's measurement problems are not soluble without the processing power offered by a computer.

Major advances made in integrated circuit technology have had a dramatic impact on signal processing techniques and associated hardware, not only in the design, construction and processing power of digital computers but also on such devices as digital-to-analogue (D-A) and analogue-to-digital (A-D) converters. These items are indispensable in interfacing with the real (time-continuous) world. A recent development which, it appears likely, will have a revolutionary effect on the field of digital signal-processing, and with it CAM in general, is the introduction of the first single-chip microprocessors, in particular the more powerful sixteen-bit devices.

A microprocessor incorporates all of the essential features of the central processing unit (CPU) of a conventional computer, integrated onto a single silicon die and housed

in a single package. These new devices offer a sufficient operating speed, digital wordlength and computational power to make them suitable for many signal-processing applications.

The rapid advances made in integrated circuit technology have led to a considerable reduction in the cost and physical dimensions of digital processors (Fig. 2.1) thus making them a viable proposition for many applications and has, further, brought them into range of many potential users. This cost versus power consideration adds considerable weight to the economic factors already discussed.

Accompanying the recent hardware developments, and of equal importance, are rapid advances being made in digital signal-processing techniques and algorithms [4][5]. The most important of these is probably the rediscovery and subsequent development of the fast Fourier transform (FFT) algorithms [6]. Advances made in digital filter theory and techniques have also been responsible for the increasing use of digital rather than analogue equipment for signal processing.

The development of digital techniques has not only brought about the simulation and replacement of analogue techniques, but has also led to entirely new theories and methods which exploit the discrete nature of the data and have no counterpart in the older theories. The combination of these theories and techniques with the processing power of modern computing devices is providing the experimenter with unprecedented signal-processing capabilities.

A brief review of the application of this new found capability to the problem of real-time analyses of the electrical activity of the brain will now be presented.

An enormous amount of effort has been expended in random signal analysis techniques for application to EEG resulting in many papers on computerized pattern recognition (or feature extraction) schemes [7][8]. Of these, most are concerned with non-real-time analyses of pre-recorded data,

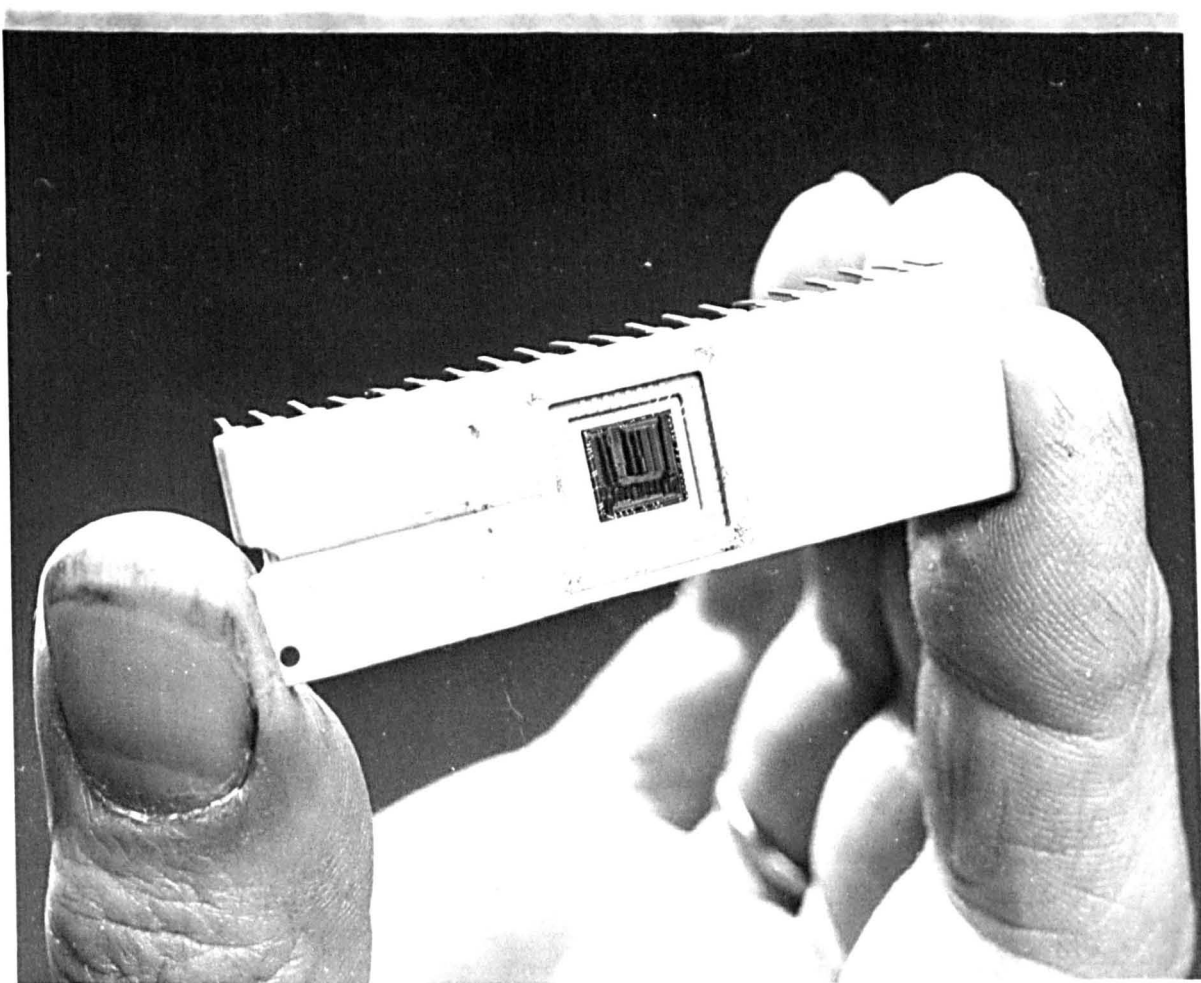


Figure 2.1 The F100L - a complete central processing unit integrated onto a single silicon chip.

and are thus of little direct relevance to the current project, except where the algorithms may be modified to produce a real-time solution. From the papers concerned with real-time EEG analysis, none could be found which utilized a microprocessor as the basic processing element, and so it has been assumed that none exists. Indeed there is very little published work in any branch of signal-processing concerning the use of microprocessors. The papers to be reviewed have been selected with an emphasis placed on schemes which offer the potential for possible modification and implementation on a microprocessor based system.

Special purpose hardware systems have been built for automatic spike detection (the high-frequency abnormality referred to in Chapter 1) that employ the second derivative and spike width as the criteria for detection [9][10]. Carrie used a hybrid computer for spike detection working on both the original EEG signal and its second derivative [11]. A quantitative comparison was made between values from consecutive EEG waves and a moving average. An output was generated when the ratio of the input sample to the moving average exceeded a preset limit. Carrie reported that use of the second derivative gave better results than the first derivative.

Smith [12] used a technique of modelling the waveform to be recognized. The model adopted was a triangle to classify the three essential features of an abnormal spike:-

- i) a relatively large and smooth slope followed by a similar slope of opposite polarity
- ii) a sharp apex
- iii) a definable base of 20 - 80ms.

These parameters were all adjusted to give the optimum correlation between the processed results and those of trained EEG readers. The detection criteria thus defined were applied to the filtered first derivative of the EEG waveform. A success rate of 85% was reported for the records

analysed with this technique, the predominant cause of false alarms being muscle artifact (which satisfied the spike model).

Various methods of performing a direct filtering operation on the original data have been implemented. One such method was that of matched filtering [13]. For this, a non-recursive filter whose pulse response is the required signal in reverse form is used. Reasonable results can be obtained by the use of this method but it suffers from a high false alarm and 'miss' rate because many different spike waveforms may occur, each requiring a different matched filter.

Another approach is the use of a bandpass filter centred around the spike frequency. Stevens et al [14] describe such a system which utilizes a 6 - 10Hz bandpass filter, a slope recognition unit and a voltage trigger. The parameters were preset for each subject on an empirical basis, which appears to be its major disadvantage. The method was used to carry out an automatic analysis, on recorded data, of spikes occurring in 24hr. epochs. The performance of the system, to quote the authors, '...suffered from arbitrary errors... but is likely to be more consistent than intuitive recognition by the electroencephalographer' (i.e. the system does not appear to be a very sound means of automatic recognition).

A long term analyser has also been developed by Gergely and Paul [15] which enables a short length of record preceeding and following a spike-and-wave discharge to be recorded. This leads to a considerable condensation of the data collected during a long term analysis, by removing the substantial lengths of record between the features of interest. The device operates on all sixteen channels of a conventional EEG recording simultaneously, but trigger levels can be set separately for each channel. The system is currently undergoing evaluation and no results are as yet available.

Various studies have been conducted involving the fast Fourier transform (FFT) since the original work of Grass and Gibbs [16]. These are concerned more with an analysis of the frequency components of the complete waveform rather than for the detection of specific features. As a result they were considered to be an unsuitable method for the purposes of detecting spike activity. A further consideration against the Fourier methods is that of computational requirements and the effect this would have on the real-time bandwidth of any system using them as a means of analysis.

An evaluation of many of the methods discussed in this chapter is given in Chapter 5, where the nature of the EEG signal in general and the abnormal activity under investigation in particular, are considered in far greater detail.

Chapter 3

The Roving

Slave Processor

Concept

Any computer installation engaged in computer-aided measurement will almost certainly experience periods of inefficient usage. The extent of this inefficiency will be directly related to the proportion of time during which the computer performs real-time signal-processing measurements.

This appears to be a contradictory state of affairs, after all if the purpose of the computer is to assist in measurement tasks any increase in this demand should lead, automatically, to an increase in the efficiency of use. However, a closer inspection of a number of typical CAM problems indicates that this assumption may not always be correct. In general, three distinct phases may be identified:-

- i) preparation
- ii) measurement
- iii) post mortem.

A consideration of the computational requirements of these phases reveals that phase (ii) is very different in nature from phases (i) and (iii).

The preparation phase is the period during which the measurement problem is analysed and the programs, necessary for its solution, written and tested. This will normally require the use of many of the system resources, both peripherals and system software.

The measurement phase consists of using the programs prepared in phase (i) to carry out the prescribed measurement. This will not, in general, require any of the system resources other than the central processing unit (CPU) itself and a few signal-processing peripherals (e.g. analogue-to-digital and digital-to-analogue converters, real-time clock, etc.).

The final phase, post mortem, is an optional phase and the usual activity here is further analysis of the results obtained in phase (ii), for display in some suitable format, for example. This, again, will require many of the system resources used for phase (i).

As can be seen, phase (ii) requires none of the system resources required by the other two and it is as a consequence of this difference that the computer system can be put to very inefficient use. The situation is further aggravated by the fact that much, or all, of the measurement phase may involve real-time processing, usually with full occupancy of the CPU.

It is worthwhile defining the term real-time, at this stage, as it has been the author's experience that the term suffers from a 'Humpty Dumpty' type of definition in that it means whatever the user wants it to mean! By real-time, the author is implying that the interaction between input and output of the computer is such that a minimum processing speed exists, below which there is an essential breakdown in the intended sequence of operation.

This should not be confused with on-line for which there is usually no such lower limit, other than the artifice of time-out circuits, which may be included on some peripherals, and the patience of the user. Real-time is a special class of on-line operation. All three phases of the typical CAM exercise discussed involve on-line processing but only phase (ii) will involve any real-time demands. Phases (i) and (iii) may be conducted quite conveniently under multi-user, time-sharing conditions, but the processes involved in phase (ii) may be so time dependent that the entire processing power of the CPU must be devoted to the execution of the measurement program. As a result, all of the expensive peripherals will stand idle until the measurement is completed. Obviously, phase (ii) makes very inefficient use of the computer installation and as the real-time load is increased so the efficiency of use will decrease.

One possible solution to this problem is the inclusion of a front-end or pre-processor. These usually comprise a minicomputer, special hardware or, more recently, a micro-

processor system. The normal relationship between the pre-processor and the main processor is some form of hierarchy with the main processor acting as master and the pre-processor its slave. The connection of the on-line experiment is now modified from the simple layout of Figure 3.1 to that shown in Figure 3.2. The slave processor can now be delegated various operations concerned with the real-time measurement task only requesting attention from the main computer when absolutely necessary.

This can ease the load on the main processor considerably, leaving it free to perform other tasks between interruptions from the experiment. The number of interruptions received by the main processor during the course of a measurement will depend upon the nature of the measurement and the computational power of the slave processor. If a minicomputer is used it should be able to act in almost total autonomy, once loaded with its program from the main processor. An interrupt need only be requested upon completion of the assigned task or if, for example, a block of data is to be transferred to the main backing store (e.g. disc). In contrast, if the slave falls in the special hardware category it may be very basic and will demand attention from the main processor for all but very simple operations; e.g. the slave may be a simple level detector which causes an interrupt to the main processor every time the input rises above a pre-defined threshold level. The main computer would then take over and perform the required analysis on the input waveform returning control back to the slave upon completion, to await the next interrupt. The microprocessor based solution falls between these two extremes and its power is a direct function of the power and complexity of the microprocessor and associated system.

Various pre-processors have been interfaced to the Department's computer by past research students, the most

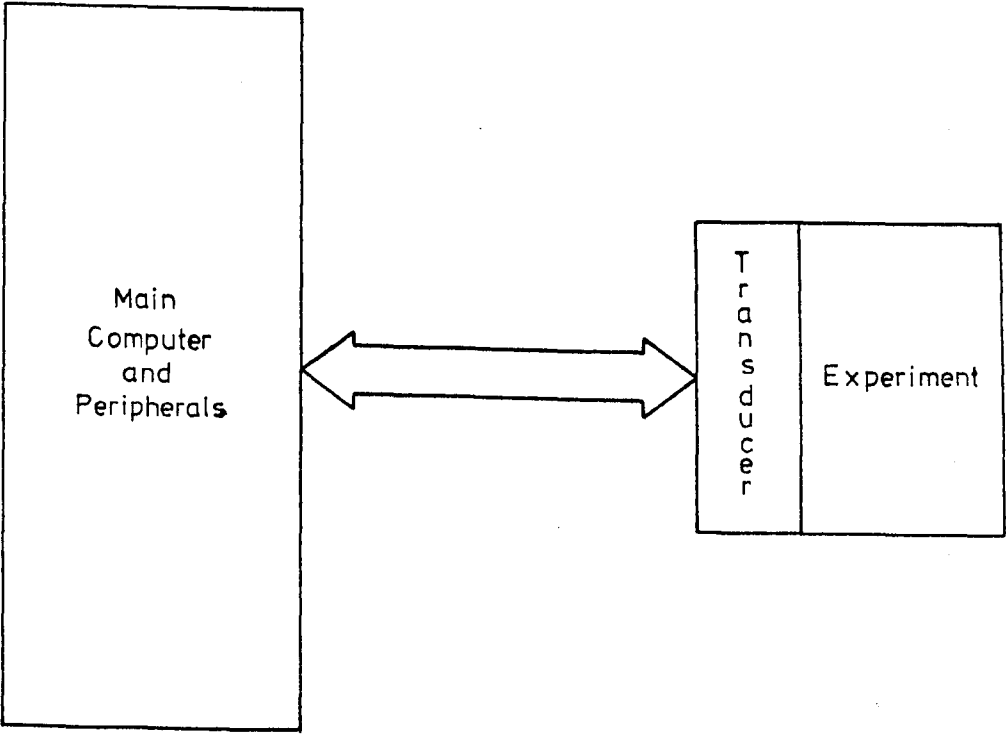


Figure 3.1 Connection of an on-line experiment.

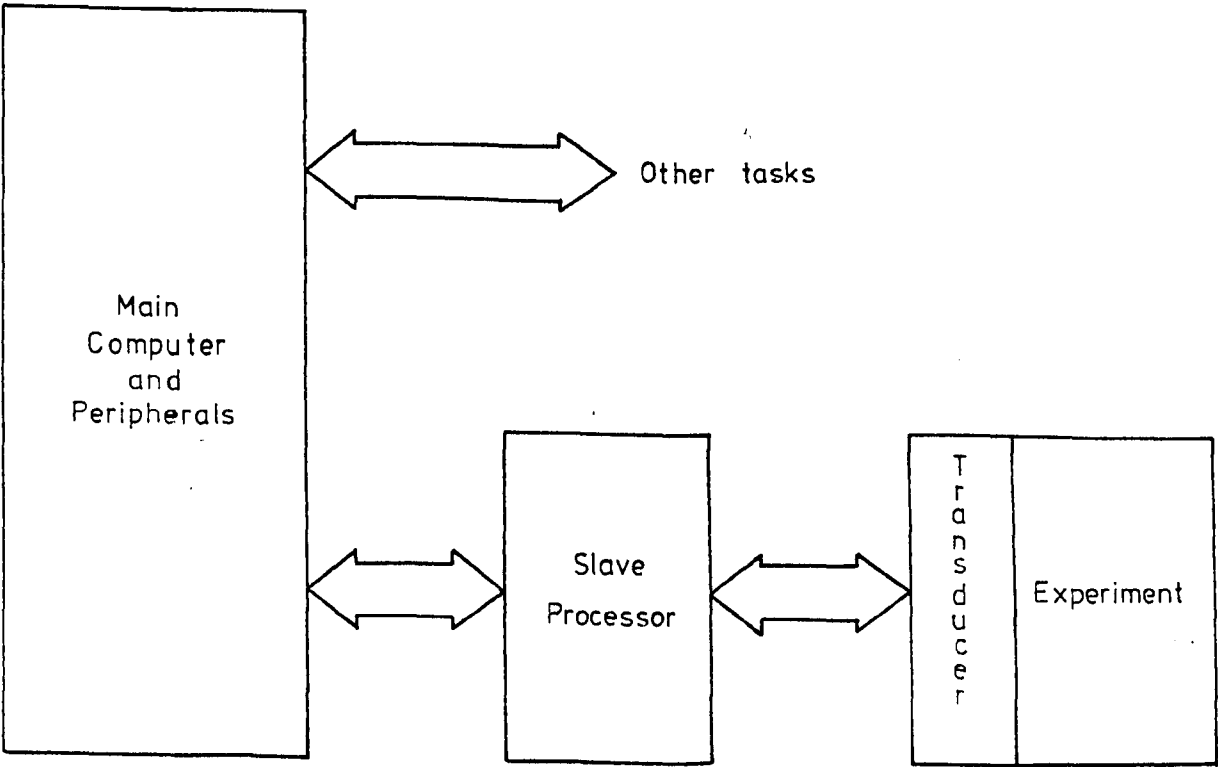


Figure 3.2 Hierarchical method of on-line connection.

important of these being the Pulse Height Analyser [17] and Transient Recorder [18]. It was during this work that another problem was becoming more prominent, and that was the physical immobility of the computer installation. There was a real need to be able to transport the power of the main computer to remote sites, and it was from this need that the idea of the Roving Slave Processor arose [19][20][21][22]. In this, the hierarchical system already outlined is taken a stage further and the slave processor is made detachable. It is then free to be taken to the object under test rather than vice-versa, as is the normal case (see Figs. 3.3.a and 3.3.b).

The RSP is dependent upon the main computer for all program preparation and provision of peripherals, as is any pre-processing device, but unlike the others once it has been loaded with a suitable program it is capable of disconnected operation.

Several important advantages accrue from an approach of this type, besides those already mentioned; a few of the more important will be considered briefly.

Firstly, numerous RSPs may be serviced via a single link to the master, instead of each pre-processor requiring its own individual link which can involve important economic consequences.

Individual users may have exclusive use of an RSP for considerable periods which can be of great importance as some measurements can involve many hours, or even days, of continuous monitoring.

Certain important peripherals can be given a roving commission, e.g. the Transient Recorder [23].

Finally, and probably most important of all, is the fact that the RSP is, in essence, a portable, software definable instrument. By the provision of a suitable program, the user may define the RSP as a whole range of different

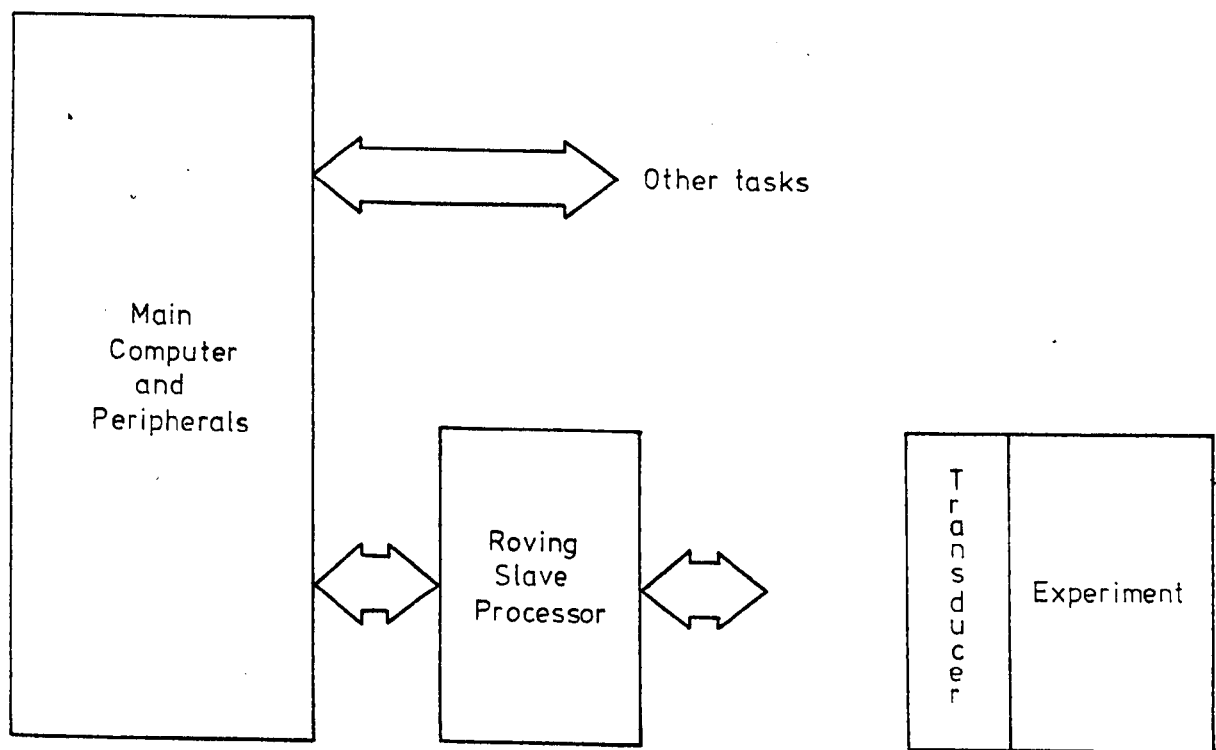


Figure 3.3.a The RSP in slave mode.

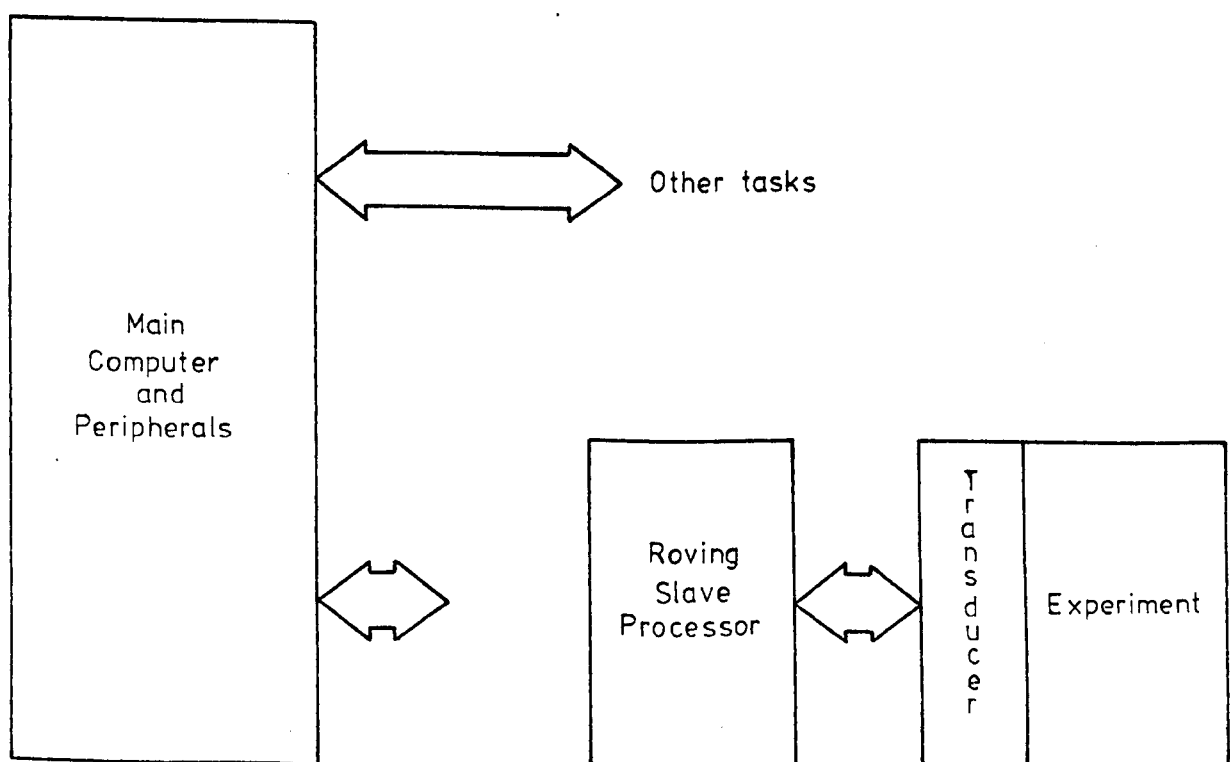


Figure 3.3.b The RSP in on-line experiment control mode.

instruments to perform a variety of measurement and control operations.

Clearly, the hardware unit devised to fulfil the role of the RSP must be of sufficient computational power to facilitate its useful application to a variety of measurement problems and also be of suitable dimensions to permit easy transportation, i.e. it must be portable rather than transportable. Ideally, the final version will be a pocket-sized device.

Of equal importance to the hardware considerations is the software system necessary to exploit the RSPs. With the availability of these basic hardware units it is possible to describe an instrument as a concatenation of standard software blocks [24]. Ultimately, the usefulness and applicability of the RSP technique will depend upon the organization of the software system and the ease with which it may be employed by operators from differing disciplines, not just computing experts.

Chapter 4

Practical Realization of the Roving Slave Processor

4.1 Introduction

The design constraints laid down for the RSP in the previous chapter were that it must be of sufficient computational power to permit its useful application to many signal-processing and control problems and that the complete system must be portable, easy to use (interface) and to program. The practical implications of these constraints will be considered in this chapter.

It is proposed that a very basic hardware unit comprising, in its simplest form, a central processor (microprocessor), a program and data store and a means of input-output, form the basis of the RSP (Fig. 4.1). A configuration of this type offers the potential of producing a miniature device. Special hardware units may be added to the basic processing unit, as required, for specific applications (e.g. real-time clock, hardware multiply-divide, etc.). The basic unit, however, should be capable of performing many of its processing tasks alone, without the aid of these special add-on hardware units.

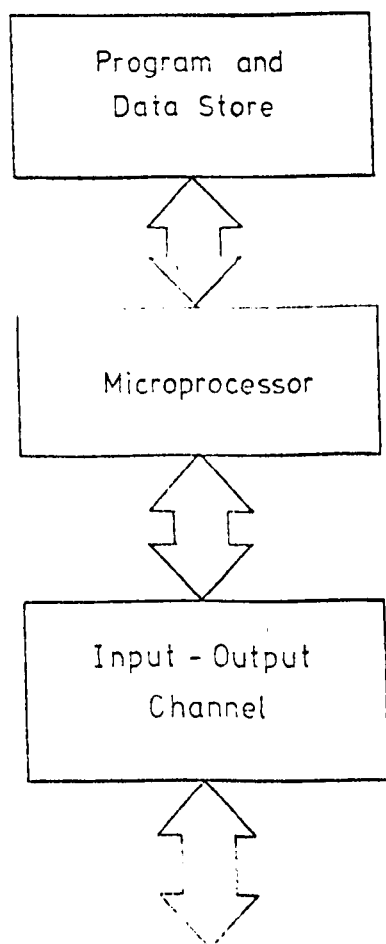


Figure 4.1 The basic RSP system.

Three prototype stages were planned, the first to be a large rack-mounted system housed in a mobile unit. This system is the one used in the present research to evaluate the applicability and potential power of the RSP concept. The second prototype is to be a portable device of similar proportions to a modern oscilloscope and a final stage involves the construction of a miniature version, of about the size of a pocket calculator.

4.2 Choice of Processor

This is the most fundamental decision to be made, an incorrect decision at this stage could invalidate the whole project. The two major considerations are the choice between multichip (i.e. bit-slices) or single board microprocessors and single chip devices, and the wordlength required for the envisaged applications.

The bit-slice approach appears attractive on first inspection since the wordlength can be tailored to suit the application as can the instruction set via microprogramming. Bit-slice and single board systems must, however, be rejected on size and weight grounds. Not only are the circuits involved bulky, but large supplies are also required to power them. Also, microprogramming is very expensive in man-hours, which would have posed serious problems.

These constraints effectively limit the choice of microprocessor to a single chip device. Further, the processor eventually chosen must be of sufficient computational power to fulfil the processing requirements outlined earlier (Chap. 2), on its own. If it becomes necessary to resort to a multiprocessor design to achieve the desired goals then a bit-slice or single board system may as well have been opted for from the outset, since the two systems will be of similar dimensions and will both involve similar power supply requirements.

As an example, a dual processor design based on the F100L microprocessor is compared with the single board 'Miproc' system, in Figure 4.2. The elements shown in the dual F100 system correspond to the processing unit represented by the single board Miproc. A hardware multiply-divide chip has been included since Miproc has this facility and the memory and interface for the second processor must also be included since they are additions, necessarily introduced by the second processor. In both cases, the main memory and other system peripherals have been omitted and so, as far as possible, two like systems are being compared.

As can be seen, the dual processor system does not appear very attractive when considered as a complete processing unit with all of its additional circuitry. With other single chip processors the situation would be somewhat worse, since the F100 is geared to multiprocessor configurations. Not only is the multiprocessor system slower than a comparable single board system, of approximately the same board area and with similar power supply requirements, but it is also far more complicated to program. One of the design criteria for the RSP is that it must be easy to use and understand so that persons whose expertise does not lie in the computing field may utilize them (see Chap.3). The concept of a multiprocessor approach was favoured at the commencement of the project [22] and is still the subject of much research, the main aim being to relieve possible input-output bottlenecks.

The original intention was to produce a dual processor system with the secondary processor responsible for all input-output operations. As has been shown by Figure 4.2, this is not such an attractive system as was first anticipated. The dual processor approach represents an elegant conceptual system but it is now the author's opinion that it is unsuitable for the RSP, particularly for the planned miniature

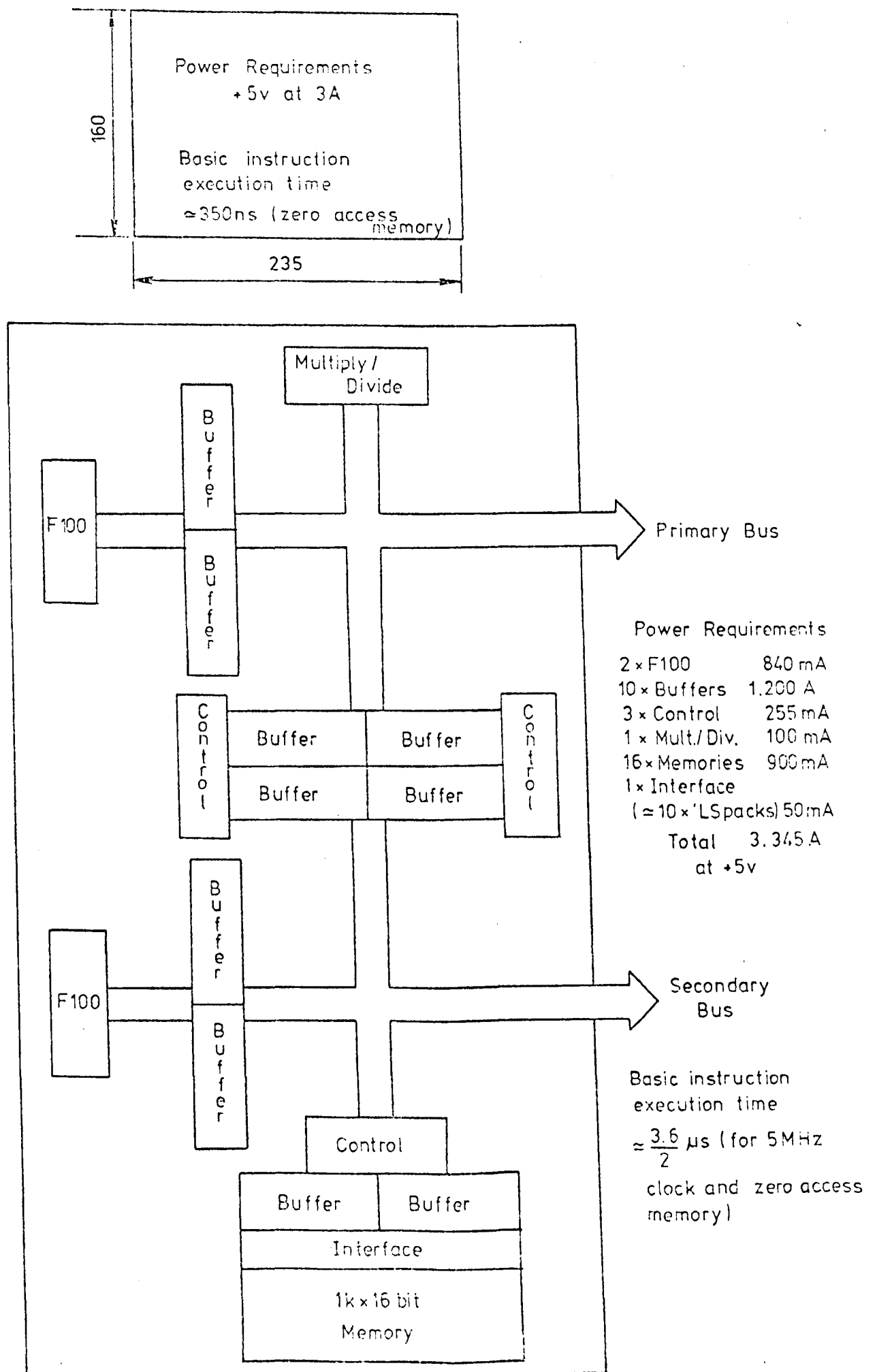


Figure 4.2 Comparison of sixteen-bit single-board and dual single-chip microprocessor systems.

version.

It is possible to reduce the total power requirement of the dual processor system given in Figure 4.2 by the substitution of specially constructed circuits to replace some of the interface sets (e.g. the memory interface). This, suprisingly, does not involve any appreciable increase in the physical dimensions of the interface and can reduce power consumption quite appreciably. The major problem of this approach, however, is the considerable amount of valuable research time which must be spent in development.

These comments do not, however, rule out the use of secondary microprocessors as part of special processing units (SPUs), or peripheral control units (PCUs), the optional add-on hardware units discussed at the beginning of this chapter. Indeed, a microprocessor based solution will often represent the optimum choice for an SPU or PCU application. It should be re-emphasized, though, that these are optional add-on units, extra to the basic RSP.

Before any comparisons are made between the available single chip microprocessors, it is necessary to determine the wordlength requirement. This will be specified by the most demanding application envisaged for the RSP. These will arise in real-time signal-processing applications. Of the real-time processing tasks, digital filtering is likely to present one of the most demanding computational problems, and was therefore used as a 'benchmark' to help determine wordlength requirements. The theoretical constraints imposed during the design phase have been subsequently borne-out by practical assessments on prototype devices [25].

For most signal-processing applications an interface to the real world will be required, i.e. an analogue-to-digital (A-D) converter. This will produce a quantized representation of the time-continuous input signal, the accuracy of which is determined by the number of different

discrete levels used to describe the original waveform (see Chap.6). The difference, in amplitude, between adjacent quantization levels, and hence the number of points used to describe a given input level, is related to the number of binary digits available to describe each level. The number of bits used has a direct effect on the accuracy to which a given input level can be defined:-

number of bits	number of levels	% error
4	16	6.25
8	256	0.39
10	1024	0.098
12	4096	0.024
14	16384	0.006
16	65536	0.0015

Clearly, ten bits would appear to be the optimum number of bits to use as it provides an accuracy of 1 part in 10^3 , which should be suitable for most instrumentational requirements. However, eight-bit devices (A-D converters) are more readily available and these offer an accuracy of 4 parts in 10^3 , which was considered adequate for the present applications (see Chap.6).

At this stage eight or sixteen-bit microprocessors appear equally acceptable. Problems start to occur for eight-bit devices as soon as any processing is performed on the input data, however. For any input above half full-scale (i.e. bit 7 set) ambiguities arise over the sign of the data and hence the sign of any calculated results. A reduction in the A-D conversion accuracy to seven bits giving an accuracy of 8 parts in 10^3 (approx. 1%) was considered unacceptable. As a result, the user will be immediately forced to double-length working with its associated increase in computational load and programming effort. Neither of these overheads are compatible with the RSP concept. The effect of the increased program length will be to more than halve the operating speed of the

system, thereby reducing the real-time bandwidth.

It can be argued that some eight-bit microprocessors operate at more than twice the speed of current sixteen-bit devices and so double-length working would impose no bandwidth penalty. The faster eight-bit devices, however, will impose a greater load on the power supply than the sixteen-bit machine they would replace. This need not necessarily be only as a function of the processor itself but due to such factors as the requirement for faster memory systems, with an associated increase in power consumption.

From the point of view of increased programming effort, double-length working demands a comprehensive understanding and programming knowledge on the part of the user, which has the effect of making the RSP a difficult device to use.

The programming area is where the sixteen-bit microprocessors have an undisputable advantage over the eight-bit machines. The range and flexibility of instructions facilitated by a sixteen-bit wide instruction field is vastly superior to anything which can be supported by an eight-bit field.

Finally, some operations involve the use of fixed constants which are used in calculations, e.g. the coefficients required in a digital filter calculation. The range these coefficients may take is again governed by the number of binary digits available for their quantized representation. For this purpose even sixteen-bits is none too generous, offering only $\pm 32,768$ levels (i.e. 0.003%). Further, when these constants, or indeed any quantities, are used in calculations, the number of bits required to represent the results will, in general, be more than was required to describe the original values (e.g. intermediate results in filter calculations). If insufficient bits are available, overflow or underflow will result and again, sixteen-bits

are often inadequate. The effects of finite wordlengths are considered further in Chapter 6.

Following the choice of a sixteen-bit processor a review of those available was made. Three main contenders were identified, each offering a specific advantage over the other two [26]. The three chosen were the Ferranti F100L, General Instrument Corporation CPl600 and Texas Instruments TMS9900.

The F100L has already been mentioned and is attractive as it has been designed to a military specification and has come from a computer manufacturer unlike the other two, which come from semiconductor manufacturers. This background manifests itself in subtle ways in the architecture, input-output structure and instruction set.

The CPl600 is attractive from the point of view that it was designed primarily for real-time applications. It is not only a very powerful single chip microprocessor but is also easy to understand and use. In addition to the normal mnemonic assembly language, a 'Super Assembly Language' is also available (see sect. 4.7) which provides a very powerful, yet easy to use, real-time programming language. Also worthy of special note is the inclusion of two interrupt inputs, one maskable in software and the other not, and a group of four outputs and one input which may be demultiplexed to provide a powerful polling system comprising sixteen possible inputs.

The TMS9900 is probably the most powerful of the three single chip microprocessors considered. The inclusion of a hardware multiply-divide facility in the circuit which, even if only for unsigned operations, gives the device a great advantage in terms of processing power. The major disadvantage of the TMS9900 is the large number of additional packages it requires for a minimal system configuration. Any advantage offered to the RSP by the hardware multiply-

divide facility is more than offset by this latter requirement.

The processor should not, however, be considered in isolation, special purpose interface chips must also be considered. These are not necessarily for incorporation into the RSP but may prove very useful for peripherals to be attached to the RSP. Special purpose interface chips make the task of interfacing user peripherals to the system very simple and hence make the RSP easier to use.

The F100 is well supported from this point of view, having a three-chip 'interface-set' which can be made to operate in one of five modes:-

- i) peripheral mode
- ii) store mode
- iii) special processor unit (SPU) mode
- iv) bus extension mode
- v) buffer mode

Not all of these modes are usable in a real-time system, for example when one is employed as a store interface a minimum delay of 775ns for a read and 1 μ s for a write operation is introduced for current 5MHz devices. This timing overhead, added to the access time of the memory devices makes the memory cycle prohibitively long for any real-time system. Further, the size of the full interface-set, three forty-pin packages, and power requirement of 325mA virtually exclude their use in any mode within the RSP. However, in their place, the interface-sets can simplify considerably the task of connecting peripherals to the F100 bus.

The CP1600 is also well supported in terms of special interface chips having two main support chips, the input-output buffer (IOB) and peripheral interface controller (PIC). These are designed specifically as peripheral interface devices, both housed in forty-pin packages. The IOB is basically an eight-bit buffer offering two bi-directional,

eight-bit ports to the user. Also included are registers to hold the interrupt vector for the attached peripheral, an error input and associated interrupt vector and timer circuit which may be used to count external events or be controlled by an oscillator.

The PIC device is virtually a microcomputer in its own right and may be delegated tasks from the CP1600. Unfortunately this requires that the PIC has an internal program store, which is of the mask-programmable type. This rules out their application to experimental work since, to be economic, they must be manufactured in large quantities (>1000) to cover the initial masking charges.

The TMS9900, at present, has no special interface devices and it is left to the user to construct any that may be required. This makes the TMS9900 inconvenient when peripheral interfacing is involved.

As well as the hardware required for interface, the method of data transfer must also be considered. The F100 operates on an asynchronous transfer basis (i.e. transfers are not tied to the system clock) whereas the other two use synchronous systems. The asynchronous method offers a faster overall transfer rate, which is in keeping with real-time considerations, but in general requires a more complex interface involving more hardware than an equivalent synchronous system. Further, many peripherals are inherently synchronous in operation and so little speed advantage can be offered by the asynchronous system. Take for example, the commonest peripheral, a memory. The normal sequence for a memory access, shown in Figure 4.3, has strict timing requirements imposed on the clocking waveforms. With a synchronous system, the system clock can generally be the source of these waveforms. For the asynchronous system, either monostables or some form of clock and dividing system must be used. The use of monostables is not favoured

for high-integrity systems, which the RSP must be, due to problems of false triggering and aging of external timing components which will alter any original delay settings. The second method represents a better solution but involves a greater hardware overhead (see Vol.II.3).

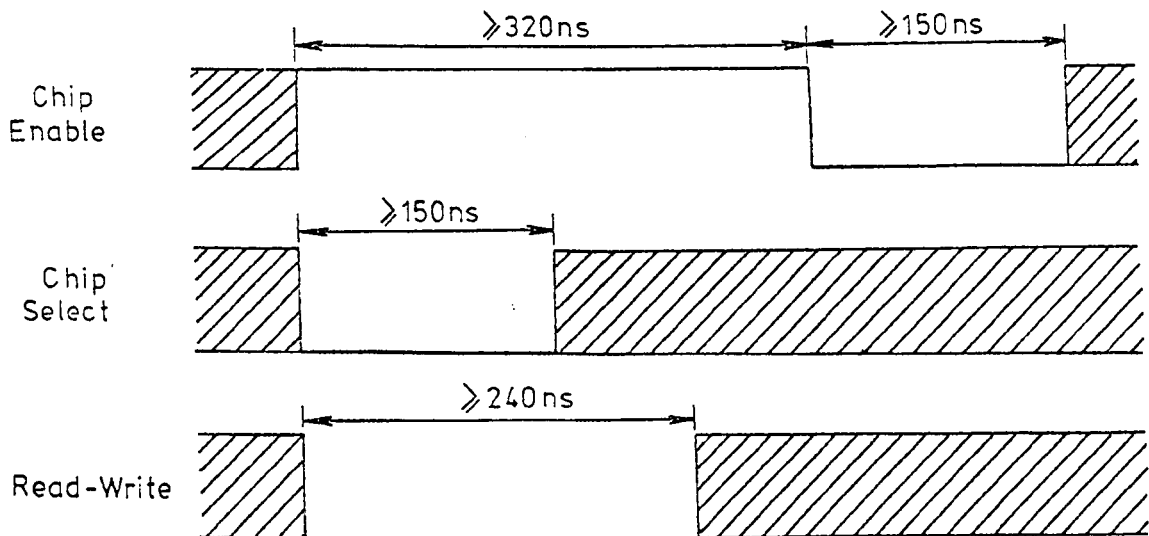


Figure 4.3 Typical timing diagrams for a semiconductor random access memory (RAM) - the TMS4030, $4\text{k} \times 1$ dynamic RAM.

Another point requiring special consideration is that of interrupts. These generally inform the central processor that a peripheral requires servicing. Interrupts do, however, place additional hardware requirements on the system in that an interrupt priority system must be established (e.g. 'daisy chain') and any interrupting device must supply the start address of its service routine to the processor. Wherever possible, a polling scheme should be used, which will often suffice for many applications, although for some, the interrupt facility will still be required. The polling and interrupt system offered by the CP1600 make it very attractive from this point of view.

A final point in the choice of processor, is the provision of internal registers in which intermediate results may be stored as opposed to main memory locations. The CP1600

has eight internal registers, the F100L only one and the TMS9900 relies entirely on external memory. The argument in favour of the use of memory locations instead of internal registers is that interrupts may be serviced very quickly, since few registers need be saved; the TMS9900 merely switches to a new group of memory registers in a single operation. The argument in favour of internal registers is that they make the device easier to program.

Another important consideration, in favour of the internal register approach, for the RSP, is one of power consumption. The memory system will draw most power when it is accessed, due to the fact that current must be switched into capacitive loads (see sect.4.4). The situation is emphasized for many memory devices which may be placed in a low-power standby mode between access operations. For certain devices the standby current can be reduced to a negligible value (pA) e.g. CMOS devices (see sect.4.4 and Appendix B). Hence, it can be seen that any method of minimizing the number of memory access operations, involved in a given program, will minimize the supply drain imposed by the system. This is an important consideration since the miniature RSPs are to be battery operated and so the average supply drain is very significant.

An example to demonstrate the effect of internal registers on the average power drain imposed by the memory, is given in Table 4.1, where the same programming operation performed on the CPL600 and F100L are compared. The example illustrates not only the considerable reduction in the number of memory accesses but also the program simplification offered by the multiple internal registers. Hence, on both counts, the provision of internal registers must be considered as a very desirable feature of the processor to be used in the RSP.

The fundamental characteristics of all three processors

are summarized in Table 4.2. From the foregoing section and a study of Table 4.2, it would appear that the CP1600 represents the optimum choice for the microprocessor most suited to the RSP application (presently available).

Two prototype systems have, in fact, been constructed, one based on the CP1600 and the other on the F100M, which is a five board processor, equivalent to the single chip F100L in all but size and power requirements (Fig.4.4).

Table 4.1 Effect of internal registers on number of memory access operations.

Function:-
$$y_n = \frac{x_n}{2} + y_{n-1} + \frac{x_{n-1}}{2}$$

F100L	CP1600	Comments
LDA x	*MVI x,R0	Load x_n value
SRL 1	SLR R0,1	Divide by two
STO m1	-	Save result (in location m1)
ADD m3	ADDR R0,R1	Add y_{n-1} term
ADD m2	ADDR R2,R1	Add $\frac{x_{n-1}}{2}$ term
STO y	*MVO R1,y	Output result
STO m3	-	Store next y_{n-1} value
LDA m1	-	Fetch next $\frac{x_{n-1}}{2}$ value
STO m2	MOVR R0,R2	Store new $\frac{x_{n-1}}{2}$ value
-	-	Continue

* - Double-word instruction

F100L	CP1600
9 instr. fetch cycles	8 instr. fetch cycles
+ 6 memory cycles	+ 0 memory cycles
= 15 memory accesses	= 8 memory accesses

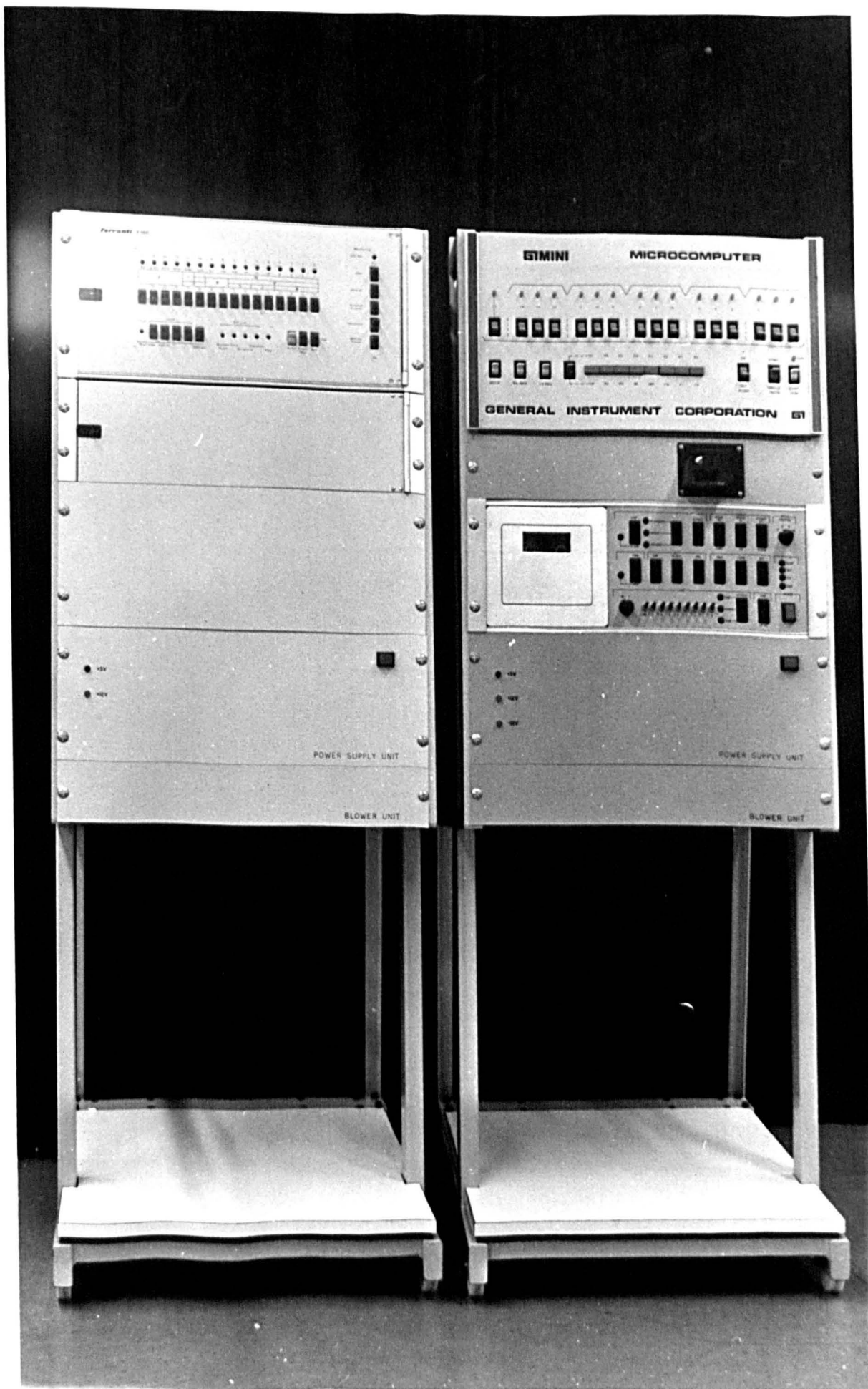


Figure 4.4 The two prototype Roving Slave Processors mounted in their mobile rack units.

Device	No. of bits	Basic instrct.	Basic instrct. execute time	Intrnl regs.	No. of clocks	Package	Power suppl. (typ)	Total power dissip. (typ)	Document.	Interface support	Special features
F100L	16	109	1.9 - 8.8 μ s (350ns memory - 5MHz CPU)	1	1	40 pin dil	+5V @420mA	375mW in CPU + 1.725W in pass transis.	Poor	Good	Special Processor facility
CP1600	16	87	2.4 - 7.2 μ s basic CPU 1.6 - 4.8 μ s 'A' versn.	8	2	40 pin dil	+12V @ 70mA +5v @ 12mA -3V @ 0.2mA	900mW	Very good	Good	Branch external polling system. Super assmblr.
TMS 9900	16	72	4.7 - 41.3 μ s (assuming no wait cycles for slow memory)	0	4	64 pin dil	+12V @ 30mA +5V @ 125mA -5V @ 1mA	990mW	Good	Poor	Hardware multi. - divide facility

Table 4.2 Comparison of three, sixteen-bit, single-chip microprocessors.

4.3 Interface to Master Computer

The interface, or link, between the main computer and the RSP is a fundamental component of the overall RSP philosophy. A great deal of design consideration and constructional effort has been expended on this part of the project. The first task was to define a design specification for the link and the functions it was to perform. Basically, the function of the link is to transfer data between the master computer and the RSP, and vice-versa. This data transfer may take the form of block or single word transfers. Since the final RSPs will have no front-panels a means of controlling start-stop operations from the master computer via the link will also be required.

Three basic approaches were identified, each offering different merits and drawbacks. The three methods are:-

- i) a special parallel highway, connected directly into the main computer's bi-directional data highway
- ii) modification of an existing channel already available on the main computer (e.g. high-speed teletype channel to give a high-speed serial link)
- iii) use of a general purpose input-output channel, available as an option on most minicomputers.

The option based on method (iii) above, has been developed and is, at present, the only link between the master and slave processors, although both of the other methods are being evaluated [27][28]. The basic features of all three approaches are summarized in Table 4.3. Method (i), the Fast Transfer Unit, offers the best performance of the three systems but involves a considerable hardware development effort to produce the direct interface to the main computer bus. The serial link makes use of a conventional teletype channel, at its highest speed setting. Again, a considerable hardware development effort is required; indeed

an eight-bit microprocessor has been used as the basis of the system.

General Input-Output (Gipop)	Sixteen-bit, bi-directional Slow - 7k bits/sec.
Serial	Send-receive (teletype) pair Medium - 31k bits/sec.
Fast Transfer Unit (FTU)	Twentyfour-bit, bi-directional Fast - 12M bits/sec.

Table 4.3 Characteristics of Link Methods.

The general input-output (Gipop) channel, although the poorest choice from the performance point of view, represents the simplest approach both in terms of design and construction time and effort. Since the link forms an essential, but not a major section of the overall RSP development program, it was decided that, for the purposes of this research, the Gipop method offered the best solution.

The link system developed is basically a sixteen-bit parallel, bi-directional data highway to the RSP, constituted from two sixteen-bit uni-directional highways from the master computer (Fig.4.5). Data are transferred as a series of direct memory access operations controlled by an asynchronous handshake pair plus an additional control line. The control input serves to latch a four-bit control word into the link which specifies the type of operation to be performed. The control word is stable at the link output before any data transfer request signals are issued.

At the time of writing, only the prototype based on the F100M is interfaced to the link; the CP1600 being, at present, a more self-contained system is less dependent upon

the master computer for program preparation and associated activities. A separate buffer board was constructed to form the interface between the F100M data bus and the Gipop link. The design of the buffer was complicated by problems encountered with time-out circuits built into the F100M system, which did not allow sufficient time for data transfers from the link. As a result, the buffer circuitry became a little more complex than was originally intended. From Table 4.3, it can be seen that with the link, transfer rates of up to 70Hz are possible. Hence, for a thousand-word program, a loading time of 15sec. will be required. This may appear excessive, but in practice proves more than usable.

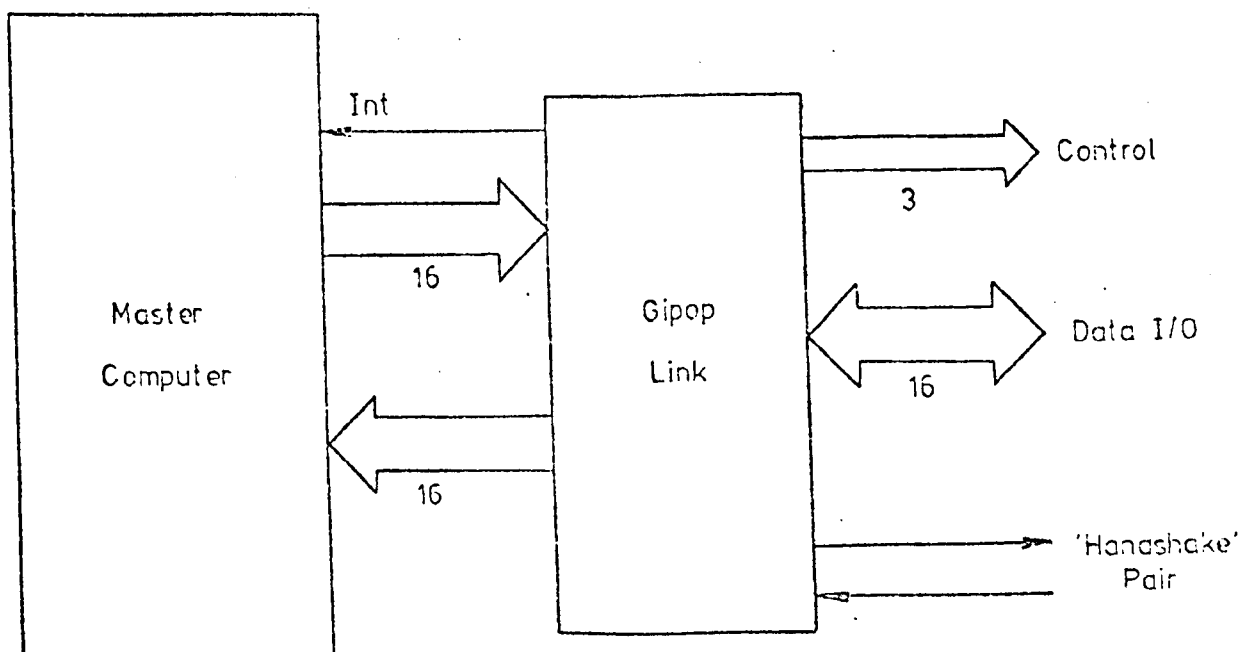


Figure 4.5 General input-output link to the master computer.

Full design and constructional details of both the Gipop link and buffer circuits are given in Volume II (Chaps. 1 and 2), together with listings of the control programs required to operate the system. At present the remote start-stop facility is not incorporated, but provision has been made for its later inclusion.

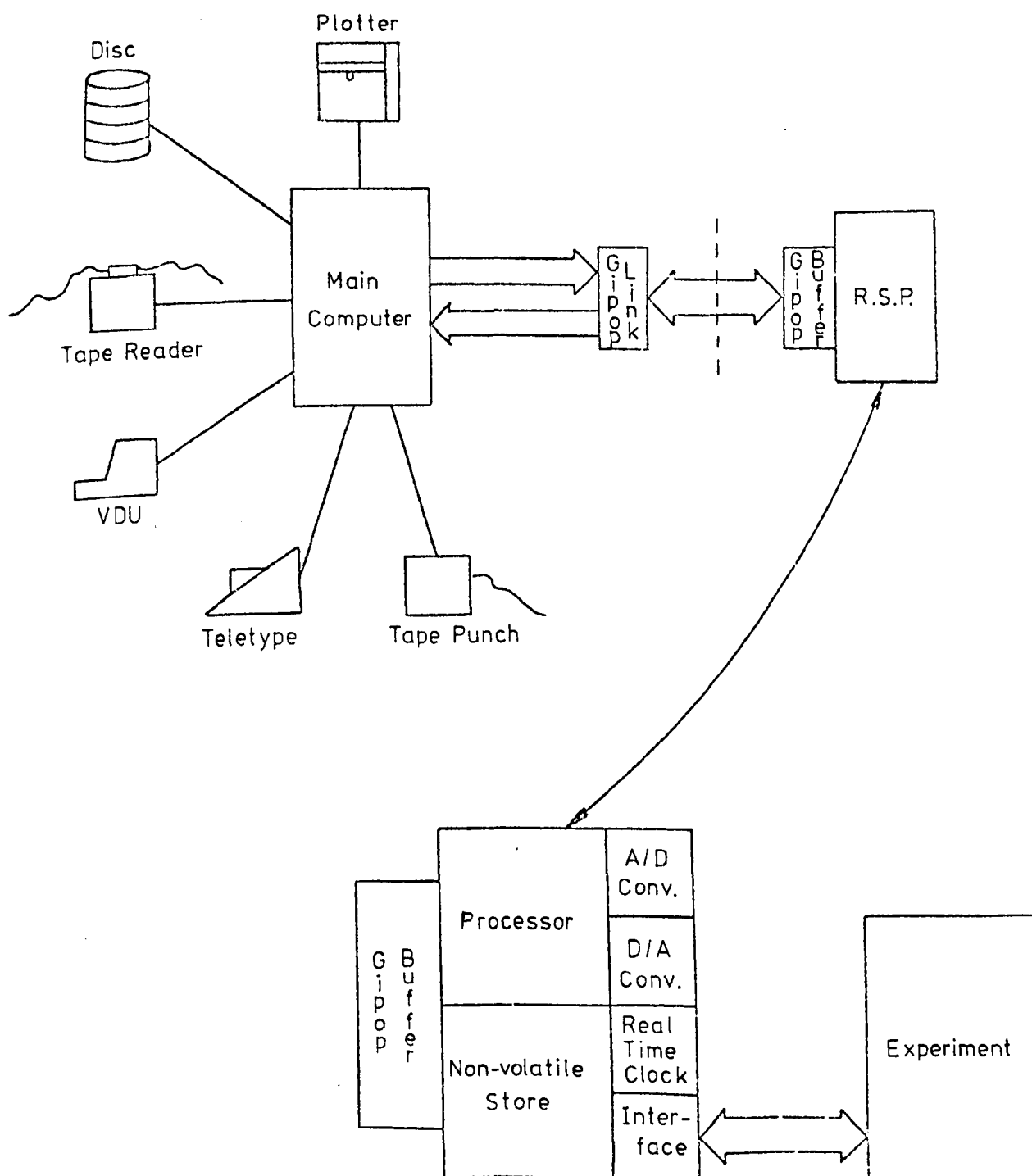


Figure 4.6 Diagrammatic representation of the Gipop system in use.

A diagrammatic representation of the Gipop system in use is given in Figure 4.6. For data transfers between the master and slave computers, the RSP is plugged into the Gipop link, as shown, and when ready, may be disconnected and taken to the site of the experiment. A further consideration arises at this stage, again due to the fact that the final RSP versions are to have no front-panels.

It is essential during program loading to check that the program has been transferred to the RSP's store correctly. A simple method of performing this check is to read the data, just written, back from the RSP's store, one word at a time, and compare it with the corresponding word in the master computer's store. The comparison process is merely an exclusive-OR operation; any result other than zero indicates an error and the bit position(s) indicate the location(s) of the error(s).

The error checking process is, in fact, performing an integrity check on the Gipop link, buffer and RSP memory, as well as checking the transferred data.

It may also be desirable to perform a check on data read from the RSP to the master computer (e.g. for phase (iii) operations - see Chap.3). In this mode, the initial read and checking operations would be performed in the normal manner, except that now the check word in dispute (i.e. second read from an RSP store location) must also be stored. When an error is found, a third read must be performed to establish which of the previous two readings was in error. It should be noted that the read check cannot identify errors in the Gipop link, buffer or RSP memory; a write-read operation is the only way in which this check may be performed.

4.4 Memory System

The choice of memory system for the RSP is possibly more complicated than the choice of processor, discussed in

section 4.2. A vast range of memory devices is available from which to fabricate the system, all with different technical specifications.

The basic requirements of the memory are that it is sixteen-bits wide, since a sixteen-bit processor is to be used, and 4k (4096) words in length. For many applications, a 1k (1024) word memory would suffice but it is considered that to make the RSP completely general purpose, a minimum of 4k words is desirable.

It is proposed that the entire memory be of the read-write type, with no predefined distinction made between which portions are to be used for program or data storage. It is left to the user to define the partitioning of the memory, which means it may be tailored to suit individual applications. A restriction will normally be imposed on the start address of the program area which is defined by the reset address of the microprocessor employed. Any reset address may be provided for the CPL600 and this will normally be location zero, but the F100L is confined to reset to location 2048 or 16384.

The use of read-write memory makes the task of programming and re-programming fairly easy, which is an important consideration for a device like the RSP, which will require frequent program changes. A further requirement of the memory is that it must be fast in order to maintain the real-time potential of the RSP system. This last requirement imposes many conflicting design problems since, in general, as the speed of the memory is increased so the power consumption rises.

The fact that the RSP is to be a portable device places further special requirements on the memory system and power supply. In addition to the size and weight considerations, common to all elements of the RSP, the store must be non-volatile; i.e. it must retain its stored information while

mains power is removed.

From the non-volatility point of view, core storage would appear attractive, but unfortunately this must be rejected on the grounds of size and weight. A 4kx16 core memory would not only be bulky, but the power supplies required to drive it would also have to be large.

A recent development in this area, is the introduction of bubble memories. Like core store, these devices depend upon a magnetic form of data storage. At present, bubble memories are better suited to mass storage systems (see later), being organized in a 64k or 92kx1 configuration with average access times in the order of 4ms. Further, they impose a fairly heavy load on the power supply during use. Hence, although bubble memories offer a non-volatile storage medium with a high packing density, they are not suitable for the read-write store of the RSP, in their present form.

Two other devices, recently introduced, are also of some interest. The first of these is the electrically alterable read-only memory (EAROM), which is a programmable read-only memory (PROM) that may be programmed and re-programmed with the use of an overvoltage input (compared with TTL levels). These devices are far more convenient in use than their ultra-violet erasable counterparts in that they do not need to be cleared before re-programming and further, the re-programming may be performed with the chips in circuit (provided any TTL circuits are protected). Unfortunately, present devices are fairly slow in operation, with typical access times of 1 - 4 μ s. Their main disadvantage, however, is that they are only suitable for program storage; it is not a practical proposition to program the EAROMs via the RSP during program execution. Hence, if EAROMs were used it would be necessary to separate the program and data stores, an idea which has already been rejected for the RSP's storage system.

A better solution is just becoming available in the form of a non-volatile random access memory (RAM). This is a very new device and still requires some technological improvements before it will be useful for the RSP. Basically, the device comprises two distinct storage areas, one a RAM and the other an EAROM, into which the data held in RAM are copied upon power down. When power is restored the contents of the EAROM are returned to their appropriate RAM locations and the system is again ready for use. Present devices are organized in a 256X4 bit configuration and although rather slow in operation (typical access time 1.5 μ s) they hold great promise for the future.

All of the devices discussed so far are inherently non-volatile. A considerable advantage may be gained by the adoption of an alternative approach which makes use of volatile storage elements provided with a means of back-up supply (i.e. battery) which takes over automatically if the main supply is interrupted.

The use of this technique brings the potential of a whole range of semiconductor memory devices into reach of the RSP. The first, and most obvious choice would be RAMs produced using the CMOS technology. This family of devices is very attractive from the power supply point of view since they may be operated over a wide range of supply voltages (3 -11v typical) and will maintain their data storage down to even lower voltages (2.2v typical) with very little supply current drain. A further attraction of these devices is that the surface leakage currents are so small that they may be considered as only drawing power when actually accessed; during all other times they are effectively in the standby mode.

Unfortunately, the price paid for this very attractive form of memory device is a low packing density. Until very recently only 1kx1 bit devices were available, involving

some sixty-four packages in the RSP's store, although now 4kx1 bit chips are becoming available.

For the miniature RSP, CMOS devices are the only practical solution since the only power source will be a battery. The portable version, however, is to be powered from the mains during operation and so alternative avenues of approach may prove profitable. One such alternative approach involves the use of devices fabricated in the n-channel MOS (NMOS) technology. Much higher packing densities are possible with this technology and at the commencement of this project, 4kx1 bit chips were available, although still very new. These devices, when operated from the same supply and at the same speed as CMOS memories, consume a comparable amount of energy, but require considerably more power during the standby phase. It was felt, however, that a 4kx16 bit memory could be constructed from NMOS devices and a moderately sized battery (130x35x60mm approx.) with a standby period of at least one day (24hrs.). A standby period of this duration was considered as adequate for most RSP applications.

There are two main classes of NMOS device, one based on a flip-flop type of storage element and the other on the parasitic capacitance of an FET. The flip-flop element may be set to either state and will remain in this state indefinitely, provided power is maintained, and is hence termed a 'static' memory. In contrast, the other device, which relies on the charge stored in a capacitive element, needs to be continually refreshed (i.e. the charge on the capacitor must be replenished) if the stored data are to be retained. The rate at which these refresh operations must be performed is governed by the surface leakage currents of the material used in the fabrication of the storage cell and is typically of the order of 2ms.

The refresh operation requires that each row of the memory matrix be accessed which automatically restores all

of the storage elements in that row. Obviously, some additional circuitry will be required to perform the refresh operation, typically ten to fifteen packages. The increase in the total memory system size as a consequence of the inclusion of the refresh circuits is offset by the much higher packing density offered by these 'dynamic' memories over static devices and the lower average standby power supply requirements (see Vol. II.3).

A great deal of the early design and development phase of the RSP system was spent on the design and construction of a non-volatile store based on the above principle, using 4kx1 bit dynamic NMOS memory elements. A full account of this design work together with many of the more technical considerations involved can be found in Volume II-3.

The main objectives throughout the design was to produce a memory system for which the standby power requirements have been minimized without impairing the operating performance when accessed. Most effort has been expended in the design of the refresh circuit in order to minimize both the size and power consumption of this particular functional unit. The power requirements can be reduced to a minimum between refresh cycles by the use of CMOS devices in the refresh circuits and by the implementation of special techniques.

Most important of these special techniques is that employed to select the address sequence during refresh. The peak currents involved in switching signals into capacitive loads in a short period can be very considerable. For the devices used, the maximum input capacitance for each address input is 7pF. Sixteen of these loads are placed in parallel on each address line driver, six in all, involving a worst case load of 112pF on each. When these loads are driven, the charge involved is given by:-

$$\frac{\delta q}{\delta t} = \delta i \quad \text{and} \quad \delta q = C \delta v$$

Assuming the load is to be driven in 20ns and a supply voltage of 5v is used, the peak current is:-

$$\frac{\delta q}{\delta t} = \frac{112.10^{-12} \times 5}{20.10^{-9}}$$
$$= 28\text{mA}$$

Clearly, the number of times each address is switched must be reduced to a minimum during the standby mode if the power supply drain is to be minimized. A total of sixty-four cycles is required in order to refresh the complete memory. If a simple binary counter is used a total of sixty-four switches is involved. An obvious way of reducing the number of switches is to implement an alternative form of count, and a Grey code will obviously represent a minimum, since only one input will change during each cycle. The total number of switches is now reduced to thirty-two for a complete refresh (see Table 4.4).

During standby, the entire sixty-four cycle refresh operation is performed once every 2ms, with the memory placed in the low-power standby mode between refreshes. In the operating mode, the refresh cycles are dispersed throughout the 2ms refresh period so that one row is refreshed every 30 μ s approximately. This is an important consideration from the real-time operating point of view since, if the entire refresh operation were performed as one operation with the memory in use, any access request from the processor could be delayed by up to sixty-four read cycle times. If the refresh operation is performed one row at a time, the maximum delay is one read cycle time, which is far less serious for real-time operations. Having two different refresh modes does however require that additional circuitry be used in the store's implementation.

The problem of memory reliability is also of the utmost importance. This can be a particular problem with semiconductor types of memory element[29]. A means of detecting and

automatically correcting any run-time memory failures is the ideal solution, but this can involve a considerable amount of additional circuitry. The whole question of memory and program reliability is discussed further in section 4.8.

In addition to the main read-write store, just discussed, some applications will require the use of large amounts of memory for the storage of data collected during tests. For this purpose, it is proposed that a bulk backing store be provided as an add-on hardware module, to be included when required. Just as for the RSP store, a great variety of memory types are available for this purpose.

Solid-state devices are, of course, to be favoured for this application, and two devices are of particular interest, bubble memories and charge-coupled devices (CCDs). Both offer very high packing densities, 64kX1 bits in a single package. Of the two, bubble memories have the advantage in that they are inherently non-volatile, although special precautions are necessary when power is removed. CCDs tend to be more compact and exhibit a lower operating power requirement and are generally much easier to use. The interface for a CCD memory will be smaller than that of an equivalent bubble system.

Both devices are very new and so mechanically based systems have been considered, as they are more readily available. Of the devices available, the most useful types for this application were considered to be the cassette, cartridge and floppy disc storage systems. Of these, the cassette and cartridge solutions are favoured since micro-versions are available which may prove useful for the miniature system. A cassette storage unit has been developed for this purpose and is interfaced to the CP1600 (Vol. II.7). This has proved a very important and useful peripheral.

A technical review of all of the memory devices, discussed in this section, is given in Appendix B.

0 0 0 0 0 0	1 0 0 0 0 1	
0 0 0 0 0 1	1 0 0 0 1 0	
0 0 0 0 1 0	1 0 0 0 1 1	
0 0 0 0 1 1	1 0 0 1 0 0	
0 0 0 1 0 0	1 0 0 1 0 1	
0 0 0 1 0 1	1 0 0 1 1 0	
0 0 0 1 1 0	1 0 0 1 1 1	
0 0 0 1 1 1	1 0 1 0 0 0	
0 0 1 0 0 0	1 0 1 0 0 1	
0 0 1 0 0 1	1 0 1 0 1 0	
0 0 1 0 1 0	1 0 1 0 1 1	
0 0 1 0 1 1	1 0 1 1 0 0	
0 0 1 1 0 0	1 0 1 1 0 1	
0 0 1 1 0 1	1 0 1 1 1 0	
0 0 1 1 1 0	1 0 1 1 1 1	
0 0 1 1 1 1	1 1 0 0 0 0	
0 1 0 0 0 0	1 1 0 0 0 1	
0 1 0 0 0 1	1 1 0 0 1 0	
0 1 0 0 1 0	1 1 0 0 1 1	
0 1 0 0 1 1	1 1 0 1 0 0	
0 1 0 1 0 0	1 1 0 1 0 1	
0 1 0 1 0 1	1 1 0 1 1 0	
0 1 0 1 1 0	1 1 0 1 1 1	
0 1 0 1 1 1	1 1 1 0 0 0	
0 1 1 0 0 0	1 1 1 0 0 1	
0 1 1 0 0 1	1 1 1 0 1 0	
0 1 1 0 1 0	1 1 1 0 1 1	
0 1 1 0 1 1	1 1 1 1 0 0	
0 1 1 1 0 0	1 1 1 1 0 1	
0 1 1 1 0 1	1 1 1 1 1 0	
0 1 1 1 1 0	1 1 1 1 1 1	
0 1 1 1 1 1	0 0 0 0 0 0	Total number
1 0 0 0 0 0	1 2 4 8 16 32	of 0 - 1 switches
		= 64

Table 4.4.a Number of 0 - 1 switches for a binary code.

0 0 0 0 0 0	1 1 0 0 0 1	
0 0 0 0 0 1	1 1 0 0 1 1	
0 0 0 0 1 1	1 1 0 0 1 0	
0 0 0 0 1 0	1 1 0 1 1 0	
0 0 0 1 1 0	1 1 0 1 1 1	
0 0 0 1 1 1	1 1 0 1 0 1	
0 0 0 1 0 1	1 1 0 1 0 0	
0 0 0 1 0 0	1 1 1 1 0 0	
0 0 1 1 0 0	1 1 1 1 0 1	
0 0 1 1 0 1	1 1 1 1 1 1	
0 0 1 1 1 1	1 1 1 1 1 0	
0 0 1 1 1 0	1 1 1 0 1 0	
0 0 1 0 1 0	1 1 1 0 1 1	
0 0 1 0 1 1	1 1 1 0 0 1	
0 0 1 0 0 1	1 1 1 0 0 0	
0 0 1 0 0 0	1 0 1 0 0 0	
0 1 1 0 0 0	1 0 1 0 0 1	
0 1 1 0 0 1	1 0 1 0 1 1	
0 1 1 0 1 1	1 0 1 0 1 0	
0 1 1 0 1 0	1 0 1 1 1 0	
0 1 1 1 1 0	1 0 1 1 1 1	
0 1 1 1 1 1	1 0 1 1 0 1	
0 1 1 1 0 1	1 0 1 1 0 0	
0 1 1 1 0 0	1 0 0 1 0 0	
0 1 0 1 0 0	1 0 0 1 0 1	
0 1 0 1 0 1	1 0 0 1 1 1	
0 1 0 1 1 1	1 0 0 1 1 0	
0 1 0 1 1 0	1 0 0 0 1 0	
0 1 0 0 1 0	1 0 0 0 1 1	
0 1 0 0 1 1	1 0 0 0 0 1	
0 1 0 0 0 1	1 0 0 0 0 0	Total number
0 1 0 0 0 0	<u>0 0 0 0 0 0</u>	of 0 - 1 switches
1 1 0 0 0 0	<u>1 1 2 4 8 16</u>	= 32

Table 4.4.b Number of 0 - 1 switches for a Grey code.

4.5 Input-Output Considerations

The final functional unit of the RSP to be considered is the means of input-output. It is felt that both digital and analogue input-output channels should be provided as standard.

The digital input-output channel presents no real problems since it would be sufficient just to provide any peripheral with access to the RSP's data bus. To simplify the use of this channel, data output latches followed by line-drivers should be provided. This addition will simplify the task of interfacing user peripherals to the digital input-output channel considerably.

The analogue input-output channel, although more complicated than the digital channel, is also fairly straightforward. Again, the size-weight-power versus operating speed problem arises, but the choices available in this instance are somewhat limited, which reduces the design and development problems.

The digital-to-analogue (D-A) channel is very simple and can be constructed using very few packages (Vol. II.6). The main decision to be made is the number of bits required in the digital word to be converted to analogue form. As discussed in section 4.2, an eight-bit wordlength gives an accuracy of 4 parts in 10^3 and it was decided that this is a suitable accuracy for the D-A converter. An eight-bit D-A converter with an output in the range 0 - 2.55v was used, comprising $255 \times 10\text{mv}$ steps.

The main choice for the analogue-to-digital (A-D) channel, besides the number of bits, was whether to use a commercially available converter or to construct a converter from a D-A device. Most A-D converters are fairly bulky and expensive devices, with sizable power requirements. One device, however, was identified as being the ideal choice for the RSP system, a single chip D-A or

A-D converter housed in a single standard fourteen-pin package (the Ferranti ZN425E). This device may be used as either an eight-bit D-A converter or, by the use of an internal counter, an eight-bit, counter type A-D converter. When it is used in the A-D mode, being a counter type converter, the operating speed is low, with a worst case conversion time of 0.5ms at the maximum clock frequency. Hence, if the converter is operated in this mode an external sample and hold circuit is essential. This will obviously involve additional circuitry and an alternative approach, only involving the addition of two extra packages, yields a better solution. The alternative system employs a successive approximation register together with the converter in its D-A mode and a comparator, to make a successive approximation type A-D converter. This is the form of A-D converter used in the prototype RSPs (Vol. II.6) and will probably be retained for both the portable and miniature versions. The conversion time for the successive approximation converter can be as low as $16\mu\text{s}$, but for the CPL600 version the clock rate used increases the conversion time to $\approx 26\mu\text{s}$. Hence, the need for a sample and hold circuit is effectively removed.

The interrupt and direct memory access (DMA) signals must also be considered as part of the input-output system of the RSP. It is via these lines that peripherals may demand attention of the RSP at any stage during the processing task. The DMA facility is required for data transfers between the master computer and RSP (see sect. 4.3) and interrupts may or may not be required for a particular application. The interrupt and DMA request and accept lines can be taken directly to and from the microprocessor with any priority encoding (e.g. 'daisy-chain') being provided by the user as part of the peripheral interface. For the CPL600, two levels of interrupt are available, one maskable and the

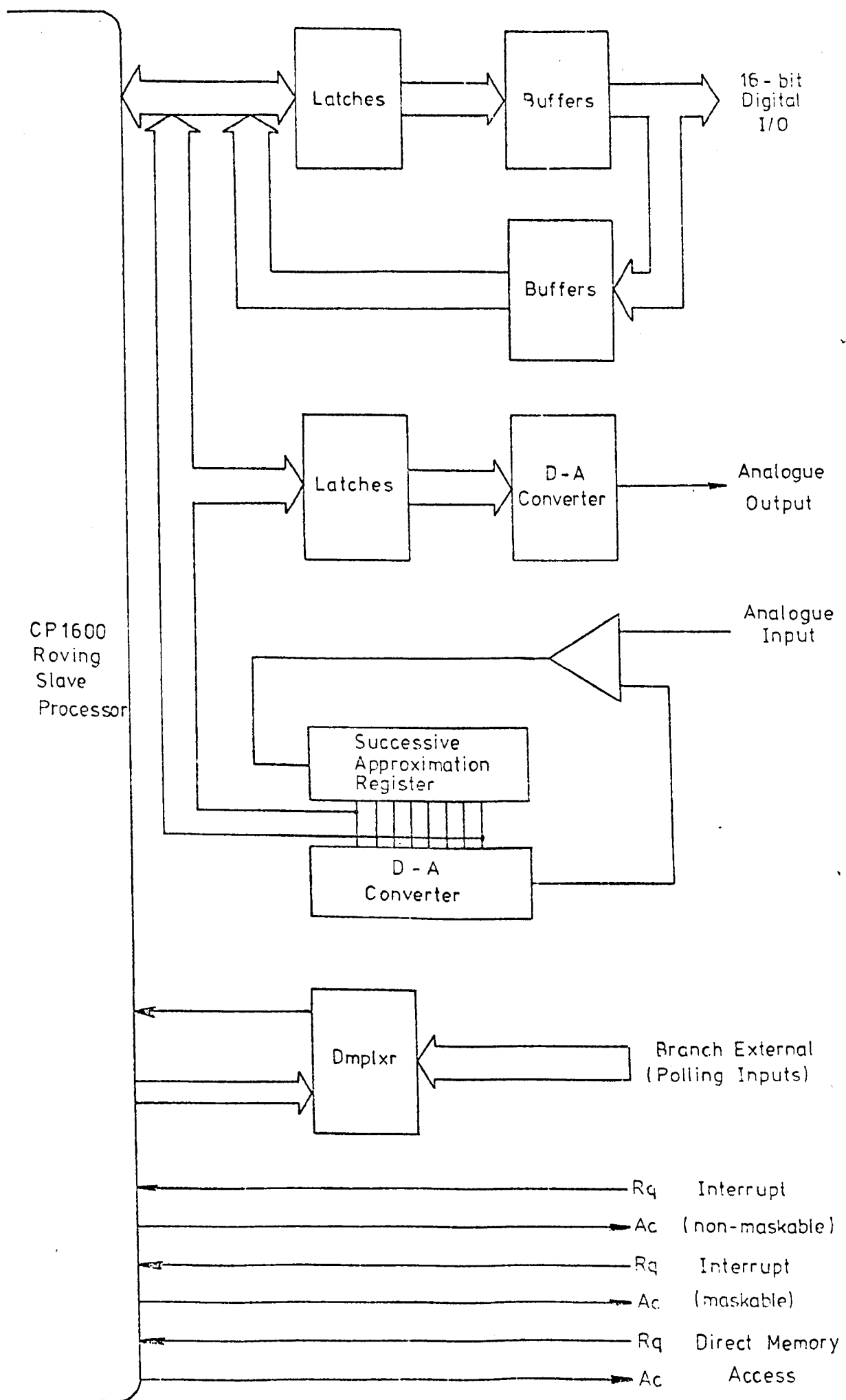


Figure 4.7 The range of input-output facilities available for the CP1600 version of the RSP.

other not. The choice of which should be used for a given application is the responsibility of the user.

As previously mentioned (sect. 4.2), the CP1600 also has the capability of providing a very powerful polling system, of up to sixteen channels. These sixteen inputs form a very important input-output control channel. With the large number of lines available no multiplexing should be required but should it become necessary the priority encoding is again the responsibility of the user.

The complete range of input-output facilities available for the CP1600 version of the RSP are shown in Figure 4.7.

4.6 Power Supply Considerations

It is probable that the power supply will be the most bulky unit of any RSP system and it is therefore of the utmost importance that every effort is made to reduce all unnecessary power consumption. Various techniques aimed at reducing the necessary power consumption of the different elements of the system have already been discussed. The main area of concern has been the standby state, where consumption must be reduced to a minimum. To achieve this goal, power should only be supplied to essential circuits during standby, i.e. the memory and its refresh circuits. Some sections of the memory may be powered-down (e.g. memory address range circuits) together with the rest of the system. Hence power is always supplied to essential sections of the memory but is switched off to all other circuits during the standby period (see Vol. II.3).

Ironically, one of the major sources of wasted power is the power supply itself. A great deal of power can be dissipated in the regulator circuits used to provide the necessary stable voltage output to the logic circuits. Methods aimed at reducing the physical dimensions and improving the efficiency of the power supply for use with

the RSP are to be discussed in this section.

Ideally, a battery supply should be used but this is not practical at present; the large power requirement of the current RSPs when operated would demand a very bulky battery. Further advances in both semiconductor and battery technology must be awaited, to provide integrated circuits with lower power dissipations and batteries with improved power:weight ratios.

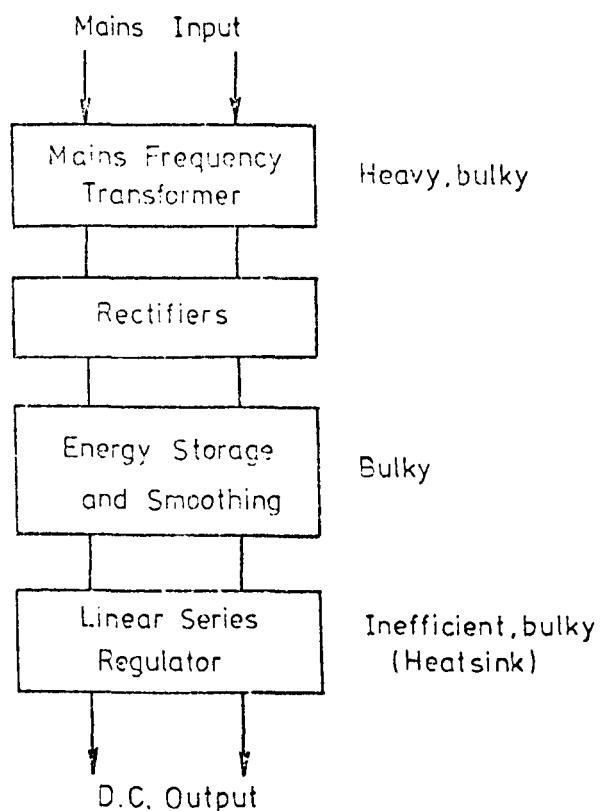
As a result, a mains supply must be used to power the system. Standard linear supplies are used in the prototype RSPs but these are fairly bulky arrangements, weighing some 27kgs.; not exactly conducive with the concept of a portable device!

The portable version is to be powered with both mains and battery supplies and the final miniature version is to be powered entirely from a battery supply. Hence, both mains and battery supplies are of relevance.

The physical dimensions of mains power supplies can be reduced considerably by the use of switching techniques. These operate on the principle of oscillating the mains input to a much higher frequency, typically 20 - 30kHz, and then using this to provide the regulated output. Operating at higher frequencies means that the size of the transformer (inversely proportional to the operating frequency) and smoothing capacitors can be greatly reduced. The resulting supply can be up to 40% more efficient than standard linear supplies (efficiencies of 80% are possible) and can be as little as one-fifth of the size and weight. The principles of operation of the two types of supply are shown in Figure 4.8.

There are several drawbacks associated with switch-mode supplies, however, which partially offset the gains. Summarizing these disadvantages:-

i) Stabilization. The degree of stabilization that any



b) Direct offline closed loop (switching) regulator.

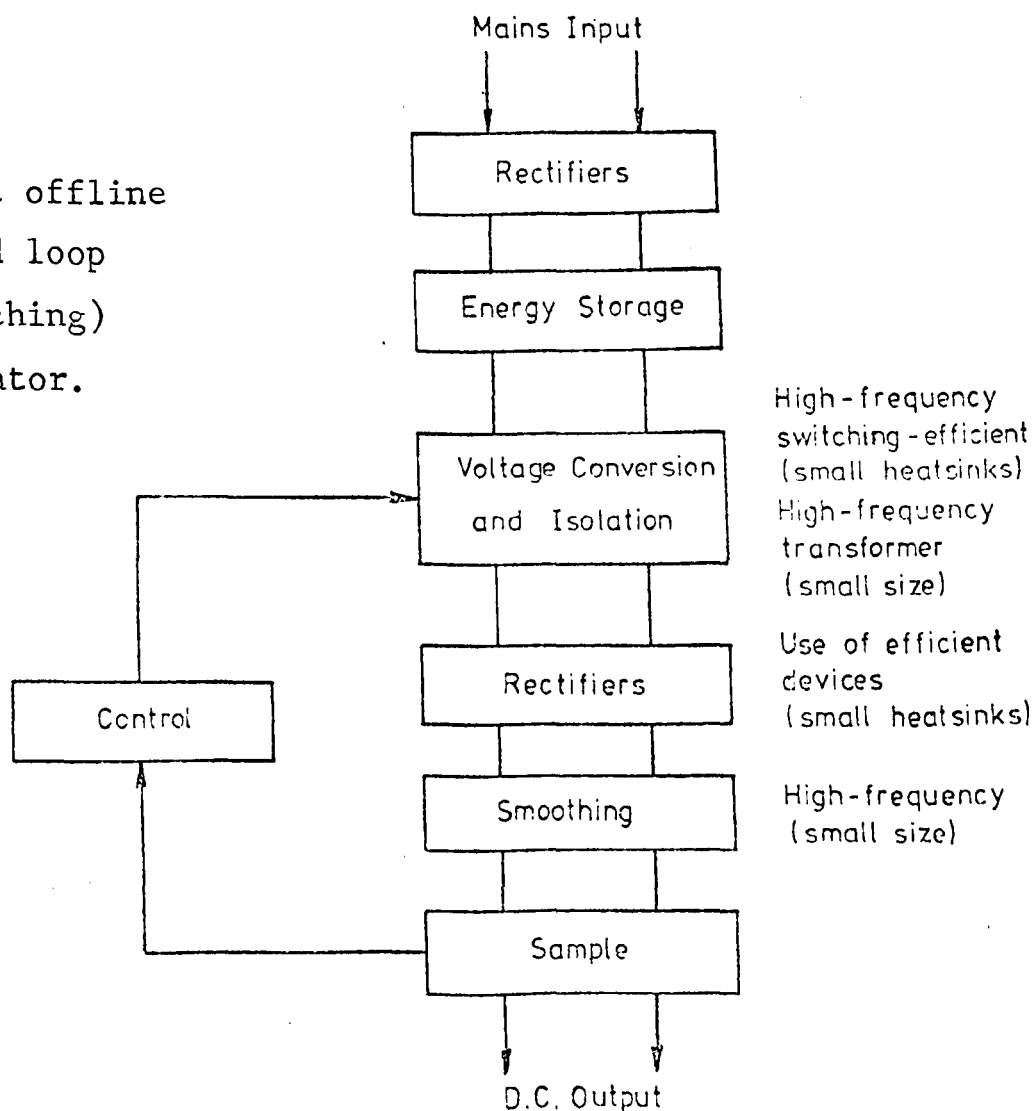


Figure 4.8 Comparison of linear and switched-mode power supplies.

regulator can produce is related to the amount of loop gain in the feedback circuit. In a switching regulator various problems restrict the gain to relatively modest values. This leads to a stabilization performance several orders of magnitude poorer than is available from a linear design.

ii) Noise. This is one of the major problems associated with switching supply designs. The switching process generates a great deal of radio frequency (rf) noise which may 'leak' out through the dc output leads or the mains cable or by direct radiation.

iii) Transient response. Switching supplies typically take milliseconds to respond to external transient disturbances as compared to microseconds expected from linear stabilizers.

iv) Complexity and Reliability. A glance at the schematic of even a simple switching supply gives an indication of their complexity. Since reliability is generally regarded as being inversely related to complexity, it is obvious that switching supplies offer a lower degree of reliability than an equivalent linear system.

v) Cost. Despite the savings made in bulk material requirements, switching supplies are generally more expensive than linear supplies, owing to the cost of the components used in the switching circuits.

All of these disadvantages are more than compensated by the considerable savings in size, weight and heat dissipation of the complete supply. When the dimensions of the supplies used in the prototype RSPs are considered it is evident that switched-mode supplies are the only possible choice for the portable versions.

As discussed earlier, the memory is to be provided with a battery back-up supply and the miniature version powered entirely from a battery. Before a choice of battery can be made the load to be placed upon it must be considered.

The total power consumption of the miniature RSP is difficult to assess theoretically and will be best determined when a prototype version has been constructed. For the 4k dynamic memory, transient demands of several amps can be involved during refresh operations and a typical quiescent current drain of 8 - 10mA is involved (see Vol. II.3) during the standby state. Obviously a battery of fairly large capacity is required if a standby period of any reasonable duration is to be obtained, a 24hr. standby period being desirable. It must also be able to cope with the large transient demands.

Another important requirement is that the battery must be of the re-chargeable type. This essentially limits the choice to either a nickel-cadmium or lead-acid type of cell. The former has the advantage of a good power:weight ratio whereas the latter has a larger capacity capability.

The large capacity requirement rules out the use of nickel-cadmium batteries at present. As a result, sealed lead-acid batteries were decided upon. The use of sealed cells ensures that corrosive fumes are not generated during use, as is the case with conventional lead-acid accumulators. These can have a disastrous effect on integrated circuits printed circuit boards, p.v.c. coverings, etc.. Sealed cells also have the advantage of being more compact and rugged than the 'wet' types. Lead-acid batteries, however, will be too bulky for the miniature RSP, so the whole problem of battery type will need re-evaluation when this version becomes a practical proposition.

It is normal practice to include a dc-to-dc converter in cascade with a battery supply when it is to be used with logic. Standard logic circuits impose stringent demands upon the voltage levels, which must be regulated to close tolerances. Dc-to-dc converters, however, operate on a similar principle to the switching regulators described earlier,

i.e. the dc input is switched at high-frequency and then transformed and rectified to produce the appropriate dc output. Hence, they suffer from similar disadvantages to the switching regulators, the most serious of which, for this application, is the poor response to input voltage and load changes. The latter of these renders them virtually useless for the dynamic memory application without the inclusion of a fairly bulky smoothing capacitor to cope with the large transient demands. A series type regulator could be used if the associated loss of efficiency and wasteful power drain from the battery could be tolerated.

An important consideration for any power supply to be used with logic circuits is that of protection, particularly against over-voltage transients. In order to gain some experience with power supply and protection circuits a protected 12v-0-12v linear supply was constructed for the CP1600 prototype (Vol. II.8).

One final consideration, on the subject of power supplies, is the trade-off between the use of separate supplies for each required voltage or the use of only one supply and dc-to-dc converters to generate any others. During development, separate supplies have been favoured as the total demands on each voltage level are ill-defined and changeable. The situation should be investigated more thoroughly during the production of the portable and miniature versions.

4.7 The Software System

The potential power and applicability of the hardware system described, so far, is wholly dependent upon the software system employed and its organization. Two conflicting requirements exist, the most important being that any programs produced must be as efficient as possible if the real-time capabilities of any instrument, thus defined, are to be maintained. The other major consideration is that the system

must be easy to use. The people requiring to use the RSPs will not normally be software specialists, so the method employed must enable them to generate efficient machine code programs starting with only a very basic programming knowledge.

The first requirement necessitates that a comparatively low-level assembly language be used whereas the second implies some form of high-level language. High-level languages are not suitable for real-time programming, however, since they tend to produce inefficient machine code, and so will not be considered further.

At the low-level, an algebraic type of assembly language appears to offer major advantages, both in terms of ease of use and scrutibility of programs so produced [30][31]. If some means of structuring the programs is included (e.g. if-then-else type constructions) then a reasonably high-powered and efficient programming language results.

An assembly language for use with the CP1600, 'Super Assembly Language', incorporates many of the above features and is, to my knowledge, the only suitable language to be commercially available. The Super Assembly Language enables the user to program with Fortran-like statements rather than at the machine level of conventional assembly languages, while maintaining a high-level of efficiency in the generated machine code. It includes LET, IF, CALL, DO and GOTO statements as well as all the instruction mnemonics and assembly directives of the more basic CP1600 assembly language. The basic assembly language statements can be intermixed freely with high-level statements to yield a very powerful and useful real-time programming language [32].

The 'ease of use' problem is a little more complicated. As already mentioned, high-level languages are not suitable for the generation of real-time programs. Equally, the use of a low-level language, even of the form discussed, is not, on its own, a satisfactory method of program preparation for

inexpert users. The language itself may be capable of conversion to efficient machine code but it is only as good as the original program. For users not accustomed to programming at the assembly level, major problems will still exist.

As for the RSP, a consideration of some typical CAM exercises, this time from a programming point of view, suggests a possible solution.

Many applications will involve the use of standard signal-processing techniques such as signal averaging or digital filtering. The programs required to perform these operations would normally be written as subroutines and made available to other users as required. If enough of these subroutines were available, then, by the use of a suitable programming system, the possibility of defining an instrument as a concatenation of 'standard software blocks' exists.

A subroutine, however, must be general purpose to be of any use. This will obviously impose software overheads which will extend the length of code, requiring a larger program store and, more seriously, may increase the execution time of the routine. These two drawbacks make the straightforward subroutine approach an unsuitable basis for the proposed programming scheme.

The system to be described in the following paper was designed to overcome these problems and has been further developed to include other important features. To make the system easy to use, routines, other than the 'High-Level Definer', described in the paper, have been written, the most important of these being the 'Library Update Routine' (see Vol. II.4).

The combination of this programming system with the Super Assembly Language provides a very powerful, and easy to use method of programming the RSPs.

THE SOFTWARE REALISATION OF INSTRUMENTS

R.A. Comley and T.R. Hewish

Electrical and Electronic Engineering Department, The City University, London

This paper deals with the technique and implications of defining instruments by means of a concatenation of 'standard software packages'. A host computer holds all the standard packages which are called by high-level commands and linked to form a complete program. The program is then transferred to a subordinate processor, realised in the form of a 16-bit microprocessor system. Comprehensive simulator facilities are used to debug programs prior to running where bottlenecks may be identified and, if required, software modules replaced by plug-in hardware modules. Examples are given, particular attention being paid to medical applications where 'personalised' instruments are often required.

INTRODUCTION

This paper deals with the technique and implications of defining instruments by means of a concatenation of 'standard software packages'. The instruments thus defined are intended for use in the demanding environment of real-time signal processing measurements.

Digital computers are playing an ever increasing role in the measurement field, mainly because many of today's most urgent problems are only soluble by the application of modern computing and mathematical techniques, which often also offer a more economical solution.

Most measurement problems require the use of some specialised piece of equipment which tends to be expensive, initially, and further, experiences a very low degree of utilisation once the task for which it was purchased has been completed. The next measurement problem will probably require some other new piece of equipment. The digital computer, by confining speciality to software, relieves this situation and it may be quickly redefined (reprogrammed) to behave as a different instrument.

Hence, we have the basis of a software defined instrument. This is not, however, a very satisfactory solution as it stands; to be useful the instrument must be reasonably small and portable as it is not always convenient, or possible, to bring the test object to the measuring instrument.

THE HARDWARE SYSTEM

To overcome this problem two prototype devices have been constructed, based on 16-bit microprocessors, the Ferranti F100L and General Instrument CP1600 devices. These are particularly suitable for this application by virtue of their processing power, cheapness, compact size and ease of interfacing.

They also are provided with support software which offers attractive features. The CP1600 has an algebraic assembler with efficient high-level features, while F100L

offers a very comprehensive set of support software including macros, link editing and simulation.

The systems are very basic hardware units comprising, in their simplest form, a central processor, program and data store and an input-output channel, i.e. a basic microcomputer. They are dependent upon a host computer for provision of all fundamental peripherals (e.g. teletype, reader-punch, front panel etc.) and for all program preparation. Special signal processing type peripherals may be added to the basic system (e.g. analogue-digital channel) as required for the various applications. The systems are connected to the host computer via a special link which allows data to be transferred to or from the subordinate processor. The program, when ready, is dumped to the subordinate processor over this link via a direct memory access (DMA) operation. Once loaded the instrument may be detached from the host computer and transported to the required site for operation.

The link also allows the system to be run in a hierarchical mode with the master computer controlling operations. This can be very useful at the debug stage.

The basic hardware unit is known as the Roving Slave Processor (RSP), refs. 1 and 2. Fig. 1 gives an overall view of the master-slave relationship.

THE SOFTWARE SYSTEM

A method of programming these hardware units is now required so that they may be quickly and easily reconfigured as different instruments. The most important feature of any programming system designed for this application is that the programs produced must be efficient if the real-time capabilities of the instruments are to be maintained. Another important consideration is that the system must be easy to use. The people requiring to use the RSP's will normally be engineers who are not software specialists, so the method employed must enable them to generate efficient machine code starting with only a very basic programming knowledge.

These are conflicting requirements, the first necessitates that a comparatively low level assembly language be used, whereas the second implies some form of high-level language. The approach adopted by the authors was to use three well established concepts in computer programming, subroutines, macros and modular programming, to develop a programming system which allows high-level type commands to be used to generate blocks of efficient assembly code.

Instead of a single general subroutine being written to perform some signal processing task, a suite of routines is written. The suite consists of various segments written to perform specific operations within the processing task. The required subroutine is built up from a number of these segments, called by a high-level command and linked together to form a single block of code.

The suite is written in the assembly language of the RSP and is stored on the master computer's backing store (disc). A control segment is included, written for the master computer, to organise the calling and linking of the other segments of the suite. The control segment uses parameters specified in the high-level call to determine which of the segments are required and also perform some checking on the calling command (syntax) and the compiled code. Diagnostics relating to signal processing considerations and constraints are also provided via the master segment, e.g. warnings about bandwidth, accuracy, input range etc. may be provided where relevant.

For linking purposes the segments must be considered as individual modules and the constraints imposed by modular programming techniques must be observed [3].

For most applications several of these subroutine suites will be required, and these will all be called from a master program at compilation time. High-level commands and assembly language instructions may be mixed freely in the master program.

During compilation all assembly language statements are passed directly to the output file together with any comments. High-level commands are decoded and the appropriate block of assembly language code is passed to the output file in place of the command.

As for the segments, the blocks of code generated by a master program must be considered as modules and linked accordingly. A further refinement to the system enables these modules to be inserted as either a subroutine or macro in the output file, the distinction being resolved by the compiler upon the number of calls made to a given suite from one master program. In this way the additional code required for subroutine calls is eliminated where not required, although some care is needed to ensure that registers from one part of the program are not inadvertently corrupted.

A further stage which is necessary to resolve the conflict between efficiency and ease of use is one of 'pre-computation' on the master computer. For example, a digital filter routine will require the provision of certain coefficients which are derived by an indirect, but well defined, procedure from a primary specification. It is too much to expect the inexperienced user to perform this calculation, so the system must be capable of working from primitive data (such as type of filter and cut-off frequency) and produce an appropriate data table ready for insertion into the efficient code at the assembly stage.

Another important consideration is the definition of data areas. These will normally be required to be accessible from several different modules generated for the program. As a result they too must be considered as modules and specified as such in the master program. In this manner the other modules may transfer data to and from the various data areas. The programming system is outlined in Figs. 2 and 3.

USING THE SYSTEM

It is now possible with such a system to generate efficient programs quickly and easily with a fairly limited programming knowledge. It is also possible to describe and define a given instrument as a concatenation of 'standard software blocks', i.e. to realise an instrument in the form of a program. The problem has now become one of modularising the programming task, and this should be approached as follows:-

- (i) analyse the problem to be solved and draw-up a scheme of basic requirements
- (ii) identify where standard modules may be used
- (iii) try to identify any modules which should be written as standard modules in the remaining code
- (iv) write the program accordingly.

The third stage of the process is probably the most difficult of the four. It is no longer sufficient to think solely in terms of subroutines, the code must be written as a subroutine suite together with a control segment. Further, the suite must be segmented carefully if it is to produce efficient code and be released for general use. It is very advantageous to have a software specialist available for this stage who can produce the necessary programs.

TEST PROCEDURES

Once the program has been produced it must be assembled and then tested as thoroughly as possible before use. It is always a difficult problem to test real-

time programs but a great deal can be achieved if comprehensive simulator facilities are available.

Simulators are available for both processors used in the RSP's, the one for the FLOOL being installed on the host computer. This makes testing very convenient since the programs may be prepared, tested and finally transferred to the FLOOL based RSP all on the same master computer. This simulator has timing traces provided, which make it possible to identify potential real-time bottlenecks that could cause the system to fail at run time. One such bottleneck may arise as a result of the execution speed of a software module (e.g. multiply) and if this is the case the user may replace this routine with an optional plug-in hardware module and associated control program. Several of these hardware modules have been developed and others are under consideration to relieve currently identified problem areas, e.g. multiply, divide, FFT.

A further aid to testing is to keep the RSP connected to the host computer after the program has been transferred, and run the systems in a hierarchical mode, with the host computer in overall control. This enables the program to be checked while it is actually residing and running in the RSP. For this purpose an on-line editor/controller is being developed and is at a fairly advanced stage. This enables the host computer to edit directly into the store of the RSP and to control its operation, e.g. to single step the RSP through the program. We feel this will prove almost indispensable during the fault finding phase.

APPLICATIONS

The applications envisaged for these software defined instruments are many and diverse. The devices are being developed primarily for use in computer-aided measurement problems in the research field, but they could also prove extremely useful for industrial applications. Two examples will be given to illustrate the power of the technique described; both are in the medical field.

The first of these arises in post-operative intensive care situations where various parameters must be kept under constant surveillance, e.g. heart-rate, blood pressure, respiration rate etc. Not all of these will require monitoring for any given patient and to produce the required instrument the user would write a master program with the necessary high-level calls and associated parameters (e.g. upper and lower safety limits). A suitable warning routine should also be selected which may be called by any of the other modules should their inputs go out of range (i.e. specified limits).

The machine code produced would then be loaded into the basic hardware unit and the various peripherals (transducers) and associated control units required, plugged in. Hence the instrument may be reconfigured very quickly and easily which is essential for this type of application as it must be ready for use as soon as the patient leaves the operating theatre. When the patient has no further need of the device it may be removed and quickly reconfigured, perhaps for a completely different type of task. Hence, these instruments can provide a very efficient and cost effective solution to this demanding problem.

The second application to be considered is the use of this technique in the detection of the onset of epileptic attacks.

Epileptic attacks are characterised by a distinctive precursor in the electrical activity of the brain. The precursor is usually referred to as a spike-and-wave complex as this is the appearance it has on an electroencephalograph (e.e.g.) of an epileptic patient.

The spike-and-wave complexes are all similar in nature but differ greatly from patient to patient; this necessitates the use of a 'personalised' instrument. It is possible to classify the features into various groups and produce standard modules

to detect the specified feature for each. The length of each module (and hence its execution speed) depends upon the breadth of the classification for each group. The more specific the definition applied to the detection decision process the shorter the code becomes. Care is needed, however, as a point will be reached where one patient may fall into several of these groups, requiring that a number of these smaller modules be compiled for a complete classification. This can lead to less efficient code being produced than that arising from less rigorously defined groups.

The original classification could be made by a visual inspection of the e.e.g. record or by coupling the patient to some larger computer which runs a complete pattern recognition and classification routine.

This is a very complex problem and will require considerable programming effort to provide a solution of the type outlined above. One factor, which may justify the effort, is our hope that the basic hardware unit can be reduced to a sufficiently small size that it may be attached, conveniently, to a patient who may then be monitored in an environment which is as near normal as possible.

DRAWBACKS

It is worthwhile, at this stage, to consider the potential drawbacks of this technique. The major obstacle to its successful implementation is the difficulty of persuading users to spend the extra time during the initial stages, necessary to segment their modules and write them in the form of a general subroutine suite with an associated control segment; it is much quicker to write the module as a subroutine for a given application. Another problem is the additional time required to test fully and document the subroutine suite for other users.

CONCLUSIONS

If these obstacles can be overcome, however, the technique can be a very powerful tool not only in the instrumentational field but in any field where a subroutine library proves useful and the system to be designed is constrained by timing restrictions.

The main power of the system lies in the fact that it enables engineers, signal processing experts, medics, etc. (i.e. individuals whose expertise does not lie in the software field) to program these devices effectively and efficiently, starting with only a limited programming knowledge. It does, however, require a knowledge of the capabilities of the modules and the various constraints imposed by the system, e.g. the maximum sampling rate (bandwidth) obtainable and any aliasing effects which may arise as a result of this limitation. Hence, these devices and the programming system are not intended for use by the ill-informed, but rather to provide a powerful tool for those who have a comprehensive grasp of their own subject and a working knowledge of other allied fields.

ACKNOWLEDGEMENTS

One of the authors, R.A. Comley, would like to acknowledge the support of the Science Research Council by the provision of a maintenance grant.

REFERENCES

- 1 Young R, Brignell J.E., Microcomputer architectures for software defined instruments, (this conference).
- 2 Brignell J.E., Comley R.A., Young R., The Roving Slave Processor, Microprocessors, Vol. 1, No. 2, p. 79-84, December 1976.

- 3 Maynard, J., Modular Programming, Butterworths, London, 1972.
- 4 Brown, P.J., Macro Processors, John Wiley and Sons, 1974.
- 5 Sohrabji, N., Macro Processor Simplifies Microcomputer Programming, Computer Design, August, 1976.

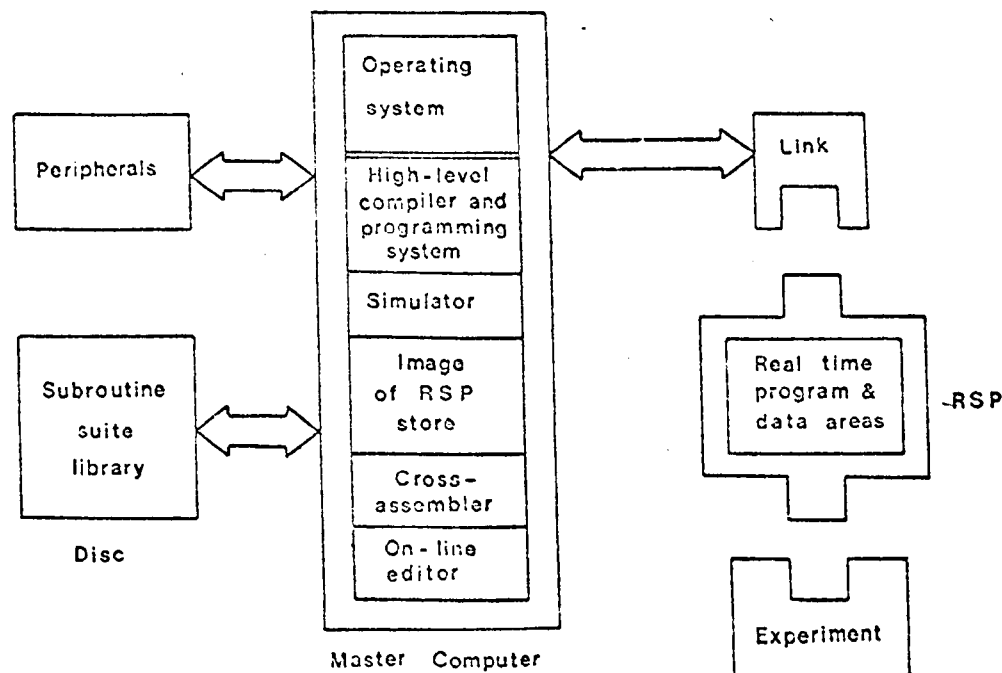


Fig.1 Overall view of the software - hardware system

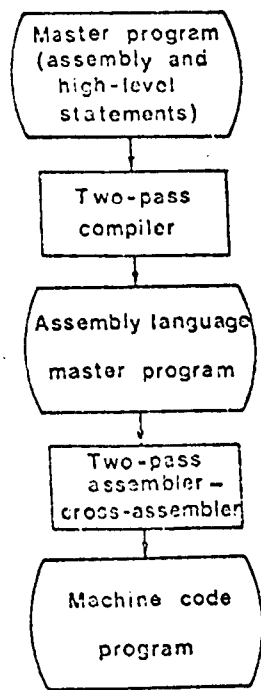


Fig.2 Sequence for conversion of master program to machine code

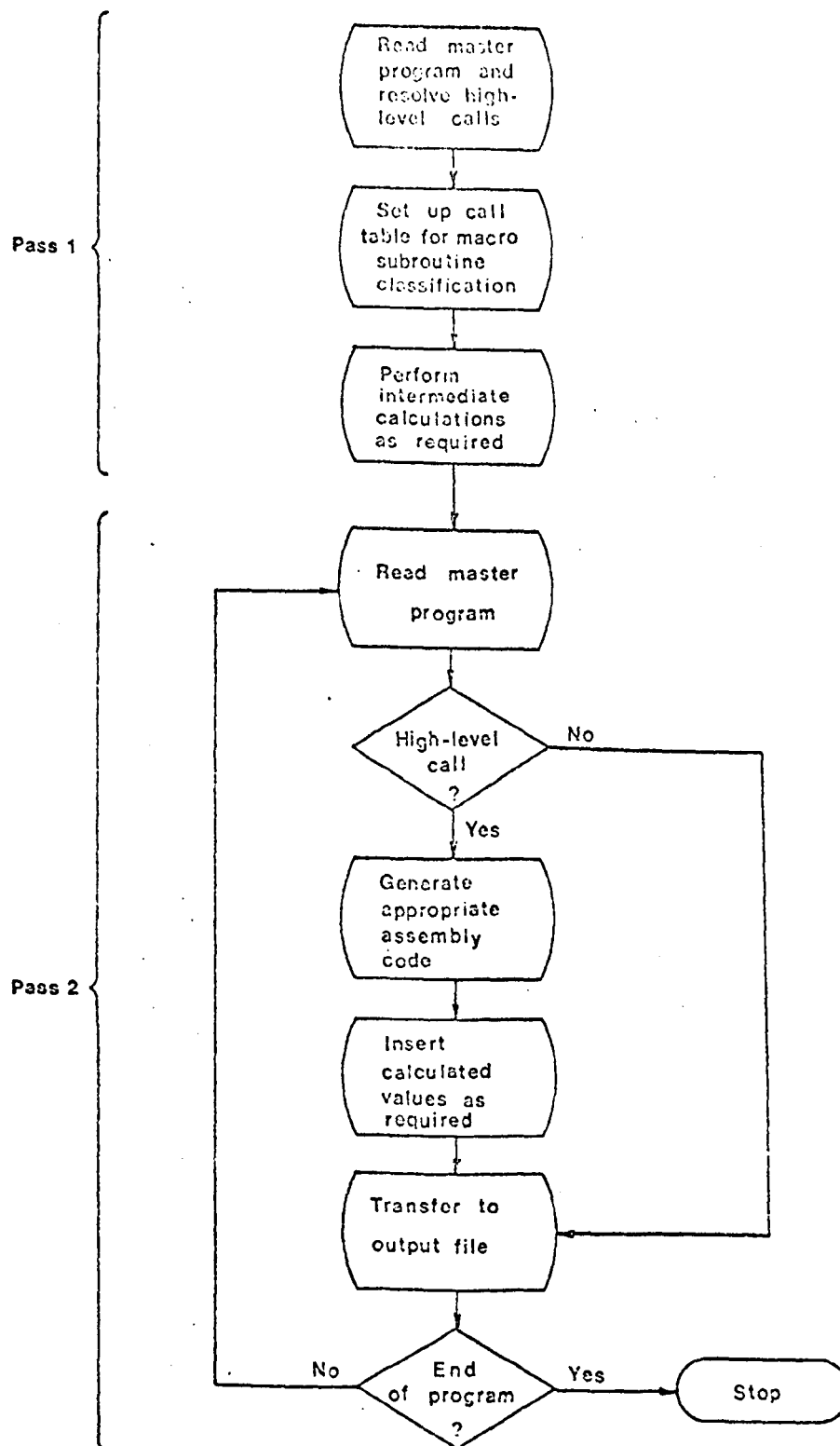


Fig.3 Operation of the two-pass compiler

4.8 Reliability

Reliability is of particular importance to the RSP as the portable and miniature versions will have no means of displaying or correcting errors when they are disconnected from the master computer, having no front panels of their own. As a result, some consideration must be given to the reliability of the complete system and of the individual components which go to make up the whole. The potential of possible future developments must also be reviewed in the light of such reliability considerations.

As the complexity of individual integrated circuits increases so their corresponding dependability both in production (yield) and use (reliability) will decrease, although a single integrated circuit should be more reliable than an equivalent multiple collection. In the limit, a point will be reached at which the failure cost, financial and otherwise, will outweigh the utility of the working system. Before this unfortunate state is reached, however, there will be many devices which are both cost effective and of sufficient reliability to be of use.

The overall reliability of a system may be improved by the use of 'redundancy techniques' in which some degree of duplication is generally involved. Since perfect reliability is a practical impossibility the use of such techniques becomes increasingly important.

The problem now becomes one of how and where to provide the redundant elements in order to achieve an optimum solution. The overheads involved in the implementation of these techniques are, to name a few, extra material, extra time to perform functions, extra power and cooling, more complex and expensive design and, very important in this case, added weight and volume.

Another major consideration before any scheme is decided upon is the function or functions required of the redundant

elements. It must be decided whether they are intended to prevent or postpone the occurrence of an error, or to circumvent, detect, diagnose or correct an error after it has occurred.

After a careful consideration of all the factors affecting the choice of a system for the RSP application it was decided that it would be sufficient to detect and, where possible, to correct errors after their occurrence. The main proviso placed on any design intended to perform this function is that it must be simple and interfere as little as possible with any constraints imposed by the overall philosophy of the RSP.

There are two main approaches available for the addition of 'redundant hardware', one is the straight duplication of circuits and the other involves the construction of error checking circuits. For the duplication technique a number of functionally identical units are placed in parallel in such a way that if one fails another may assume full control. This involves a considerable increase in the hardware required for the complete system, which is an unacceptable overhead. Further, it is aimed more at prevention and postponement of errors rather than facilitating their detection. As a result, this method was rejected.

A better solution is to construct error checking circuits to monitor activities within the system and to generate an 'error interrupt' should an error be detected. The interrupt routine may then effect an ordered shut-down of the system and finally stop the processor. One obvious circuit is a power-fail detector. Another area in which such techniques may be gainfully employed is in the detection of errors arising within the memory system. By the provision of a relatively simple parity-checking circuit it is possible to detect any single-bit errors which may occur as the result of a memory failure or noise in the system. A slightly more

complex design allows all single-bit errors to be corrected automatically and also enables multiple errors to be detected.

Details of both systems are given in Appendix C, where it is shown that a 70 - 80% improvement in reliability is possible. Neither design has, as yet, been implemented but it is likely that the simple parity check scheme will be adopted as being the most suitable for the RSP.

Software reliability can be of equal importance to the hardware reliability of a system. This is a subject in its own right and numerous publications appear each year on various aspects [33][34]. Two useful definitions arising from one such publication [33], which together serve to specify software reliability on a broad basis, are:-

Correctness - a program is correct if it performs properly the functions that were intended and has no unwanted side effects.

Robustness - a program is robust if it will continue to do something reasonable in the presence of environmental changes (such as hardware failure) and demands (such as incorrect data) that were not foreseen. The terms fault-tolerant and error-resistant are often used to describe this property.

Although no attempt has been made to incorporate any software reliability scheme, the overall concept was borne in mind when programs were prepared. Such simple precautions as performing branch operations on positive or negative results instead of zero conditions are included whenever possible as this can considerably increase the program reliability.

Chapter 5

Nature of the Problem

Before any method of detection can be decided upon it is first necessary to set up a suitable mathematical model to specify the required features of the signal. Obviously, the model will be an approximation, the quality of the approximation determining the accuracy of the detection process. Various models have been proposed and used as a basis for detection, with varying degrees of success.

Epilepsy is generally characterized by the appearance of sharp spikes in the EEG record (Fig. 5.1). Of particular clinical interest is the spike-and-wave discharge in which the spike is followed by a low-frequency (slow) wave of fairly large amplitude (Fig. 5.1). Most epileptic discharges are of this form although individual sharp waves are possible. The spike-and-wave model was used by Binnie and Lloyd [13] as the basis for a matched filter detection scheme (see Chap. 2).

Unfortunately, the matched filter approach met with very little success. A possible explanation for this failure lies in the fact that for the simple matched filter, any noise component is assumed to be random and completely uncorrelated with the sought signal. For the EEG case, this means that the whole of the record between the spike-and-wave features must be considered as white noise, which is definitely not the case. Although much of the signal appears random and uncorrelated, certain periodicities are evident (e.g. alpha-waves), particularly during periods of abnormal activity (e.g. slow-waves) which are the periods of interest! This fact can be demonstrated quite conclusively by a study of the autocorrelation functions of typical sections of EEG record.

To produce an autocorrelogram of a waveform the input signal is multiplied by a time-shifted version of itself, over the length of a record, the period of which must be greater than the autocorrelation lag. If stationarity is assumed (see later) then it can be shown that [35] for a

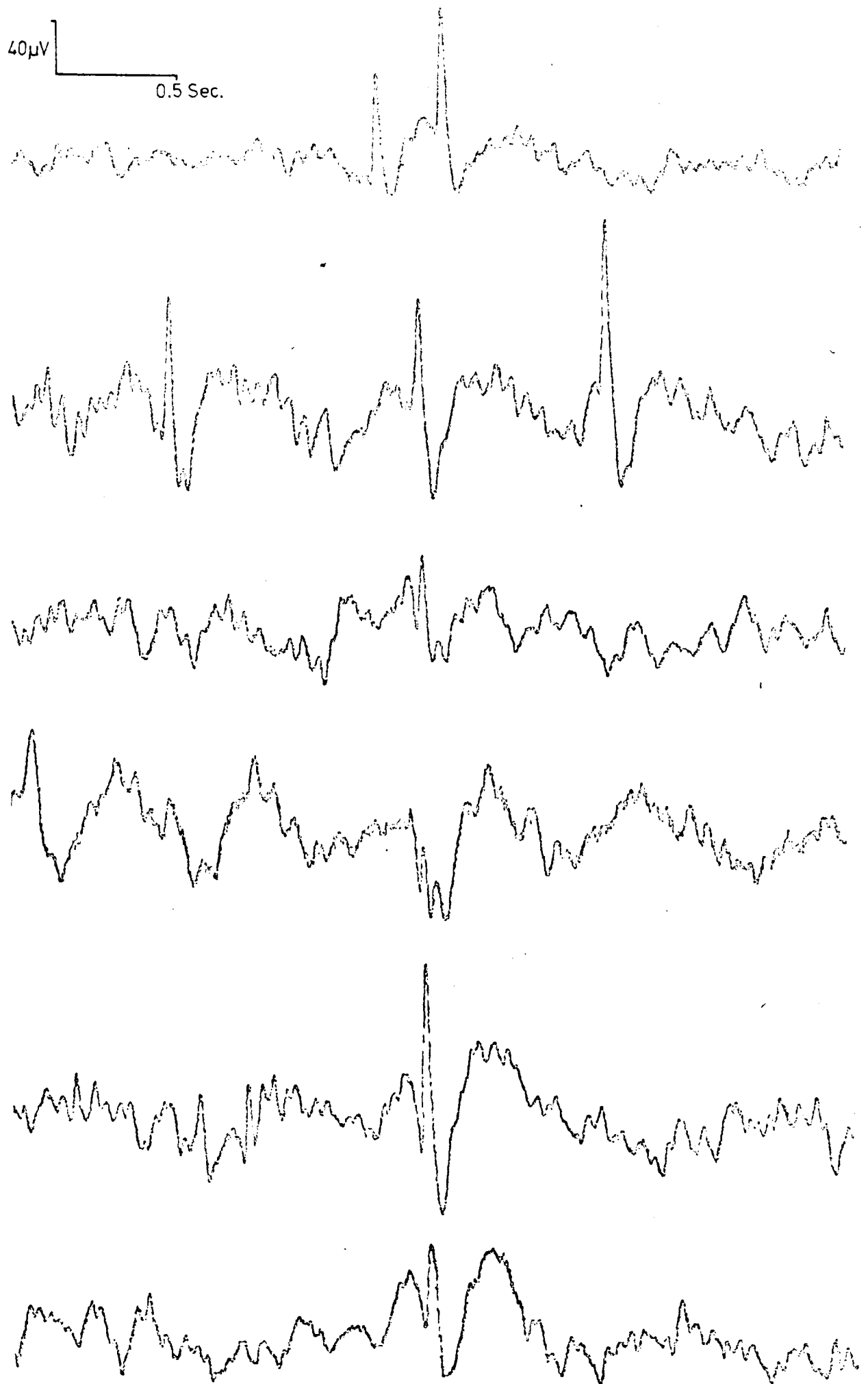


Figure 5.1 Typical spikes occurring in the EEG record.

periodic waveform $x(t)$, the autocorrelation function can be expressed as:-

$$R_{xx}(\tau) = \frac{1}{2T} \int_{-T}^T x(t).x(t + \tau).dt$$

where T is the delay period and $2T$ the period of the complete record. This can be expressed in discrete form as:-

$$R_{xx}(\tau) = \frac{1}{N} \sum_{i=1}^N x(i).x(i + \tau)$$

where N is the number of samples in the record.

A correction to the basic equation is required for finite length records. The normal method employed for digital implementations is to limit the range of values for τ by truncation. The autocorrelation function is then expressed as:-

$$R_{xx}(\tau) = \frac{1}{N - \tau} \sum_{i=1}^{N-\tau} x(i).x(i + \tau)$$

Alternative, more complicated, methods may be employed such as the use of window functions (see later), $W(\tau)$, designed to taper the signals thereby reducing their contribution to the average as τ approaches N .

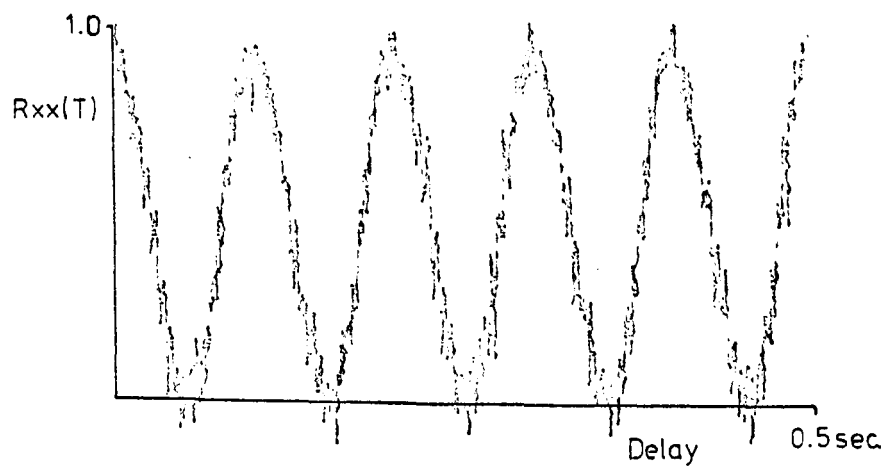
The above equation assumes a zero mean value, which is not the case for EEG records (see Chaps. 6 and 7) and so a correction is required:-

$$R_{xx}(\tau)_r = R_{xx}(\tau) - (\bar{x})^2$$

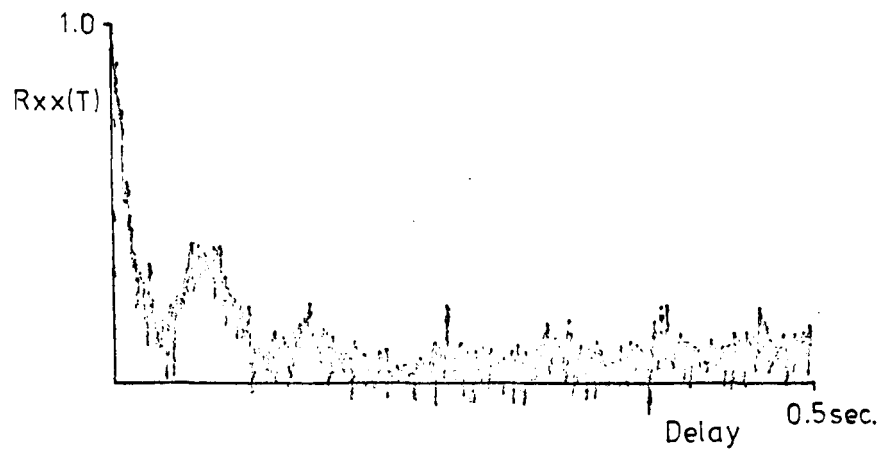
where the square of the mean input value is subtracted from the mean value autocorrelation function.

The above algorithms were implemented on the Gemini prototype system and the results obtained are as shown in Figure 5.2. The 10Hz sinusoidal input was used as a test for the autocorrelation routine which appears to be functioning correctly. The noisy appearance of the waveform is due to 'processor noise', introduced by the Gemini microcomputer while it is dumping the results.

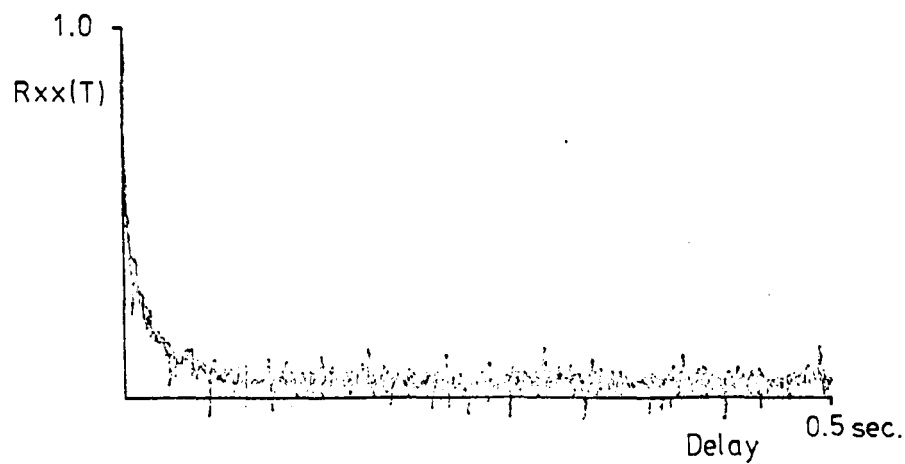
The autocorrelograms show that there is a certain amount



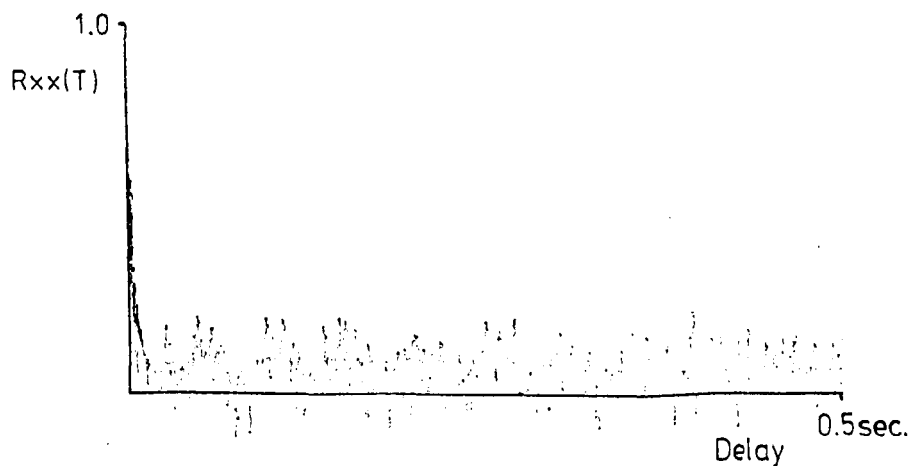
a) 10Hz sinusoidal
(test) input.



b) Abnormal EEG
record.



c) Grossly
abnormal EEG
record.



d) Normal EEG
record.

Figure 5.2 Autocorrelograms of various waveforms.

of correlated behaviour for the abnormal record analysed in Figure 5.2.b but this disappears in the grossly abnormal record of Figure 5.2.c. This result is completely consistent with the findings of Binnie and Lloyd [13] who's matched filter performed best with grossly abnormal records. The final autocorrelogram is that of a normal EEG record (eyes closed) in which the effects of alpha activity can be clearly seen. A program listing for the autocorrelation routine is given in Appendix D.

Whalen [36] gives an analysis of the matched filter for use in non-white noise, yielding:-

$$\int_0^T h_o(z) \cdot R_n(T-z) \cdot dz = x(T-T) \quad 0 \leq T \leq T$$

for a filter $h_o(z)$ in noise having an autocorrelation function $R_n(T)$. It is felt, however, that a great deal of time and effort would be required in order to realize a filter of this type which, when complete, may still not be capable of dealing with the obvious correlated behaviour. Further, the implementation of these techniques is likely to place a very heavy computational load upon the processing system, which could rule out its use for a real-time analysis.

The matched filter also suffers from other major disadvantages, which tend to make it unsuitable for the present application. The most serious of these disadvantages is the difficulty experienced in the formulation of a suitable model with which to represent the abnormal epileptic activity. For the matched filter to operate correctly a concise and rigorously defined model is required which will, in general, necessitate the formulation of a new filter for every patient, since epileptic precursors and background activity can differ greatly from patient to patient and may, in some cases, vary considerably within the record from a single patient. Great care is also required in the interpretation of results from a matched filter, since the filter tends to reinforce any assumptions made about the signal in order to formulate the

matched filter function.

In view of all of the problems associated with the matched filter approach, it was considered to be an unsuitable vehicle on which to base the EEG analysis system.

The basic problem prohibiting the establishment of a suitable mathematical model is one of definition. The official definition of an abnormal epileptic discharge is '.... a wave distinguished from background activity and having a duration of 1/12 sec. or less'[37]. As observed by Smith[12] the clause 'distinguished from background activity' in the definition hinders the design of an abnormal spike detection system. A trained electroencephalographer frequently finds it impossible to conclude whether or not a spike-like waveform is abnormal if only one or two minutes of record are available, on which to base the decision.

Discussions with trained personnel at St. Bartholomew's Hospital (London), have borne out the above statement; indeed the whole detection criterion used by human observers appears to be based more on an acquired, intuitive decision making process, rather than a formalized one.

It was during these discussions, and the initial phases of the development of the detection system that further complications began to occur. The first of these problems is the distinction made between a spike and a sharp wave. Both are abnormal, but the sharp wave is defined as '... a wave distinguished from background activity with a duration of more than 1/12 sec. and less than 1/5 sec.'[37]. Typical examples of sharp waves are given in Figure 5.3 together with a spike for comparison. It is evident that the sides of the sharp wave are as steep as those of the spike, the only distinguishing feature being the sharpness of the apex, i.e. the time (or number of samples) taken to change from a positive-going to a negative-going slope.

For the purposes of the present research, it was decided

that the analysis should be confined to a spike detection scheme and that sharp waves should, at present, be rejected by the system.

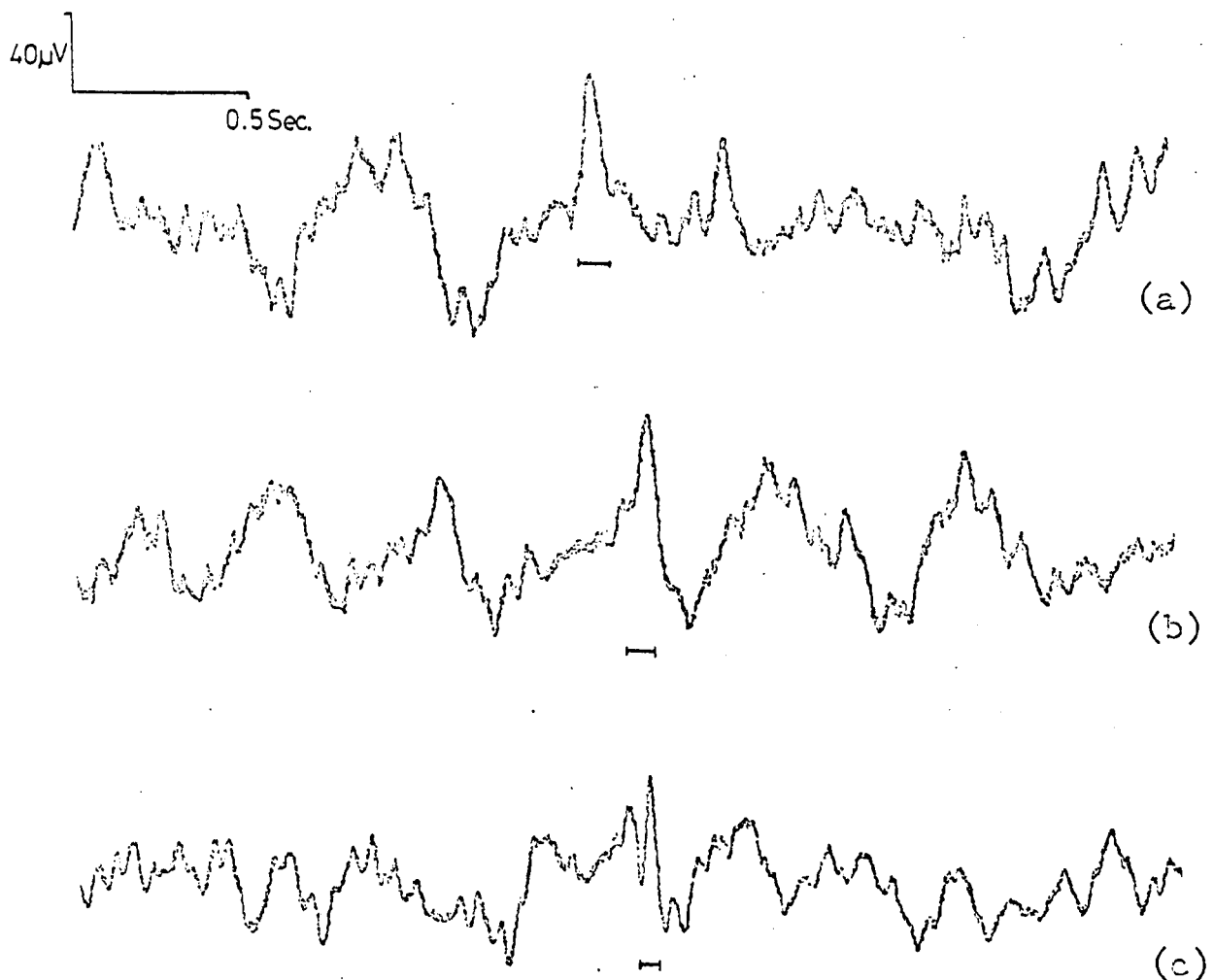


Figure 5.3 Typical sharp waves (with a spike (c) for comparison).

A second problem arises as a consequence of the manner in which the EEG signals are derived and referenced. Consider the example of two signals, given in Figure 5.4, where two methods of derivation and reference are shown. In Figure 5.4.a a common reference electrode is used (common reference method) but in Figure 5.4.b, each electrode is referenced with its adjacent neighbour (bipolar method). The most commonly employed method is that given in Figure 5.4.b, and the records to be analysed were derived in this manner. The use of this method can give rise to some apparently abnormal features in the EEG record (Fig. 5.5). Such behaviour is not , however,

abnormal and arises as a result of the reference method used. In fact, it merely shows that there has been a large positive (surface negative) shift in the signal at the reference electrode.

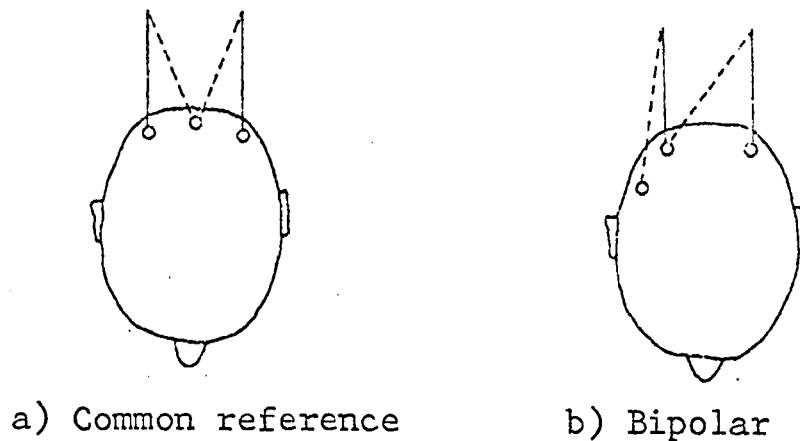


Figure 5.4 Methods of signal derivation.

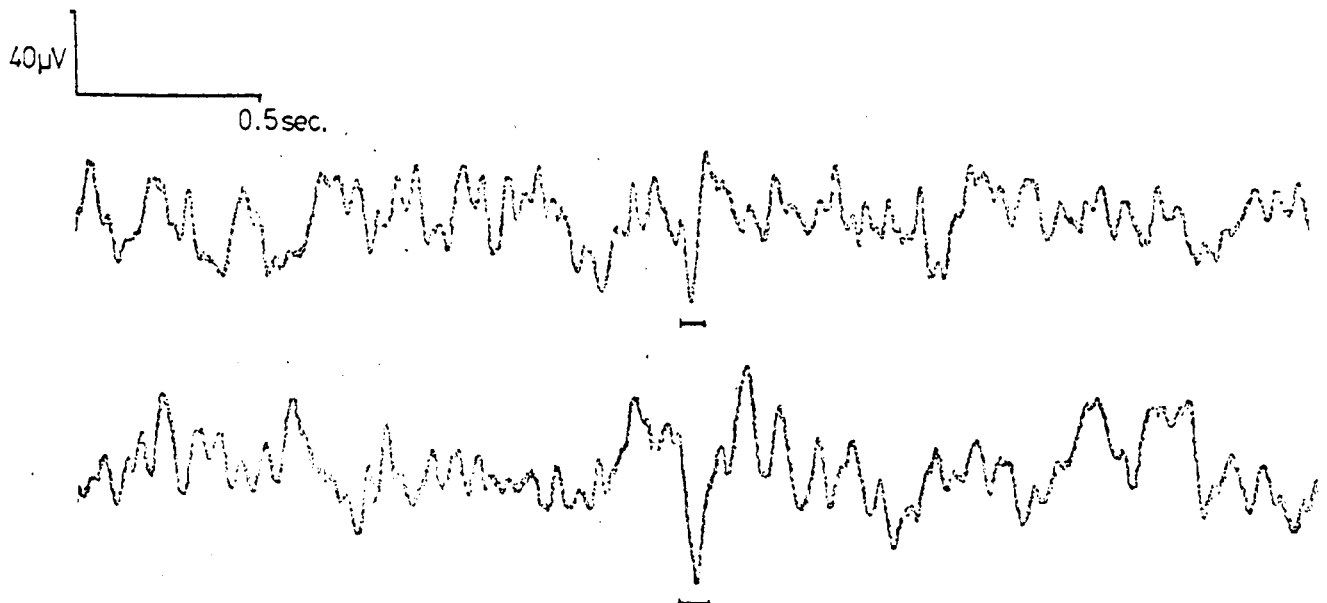


Figure 5.5 Apparently abnormal EEG activity, arising with the bipolar method of data acquisition.

The final problem is that of electrode placement and associated artifact problems. An internationally agreed standard defines the placement of scalp electrodes, as shown in Figure 5.6. Muscle artifact may be present on all inputs, but the frontal electrodes, in general, present the major problems, due mainly to eye movements. The normal form of artifact is a burst of very sharp features in the record (Fig. 5.7) which it should be possible to reject with a

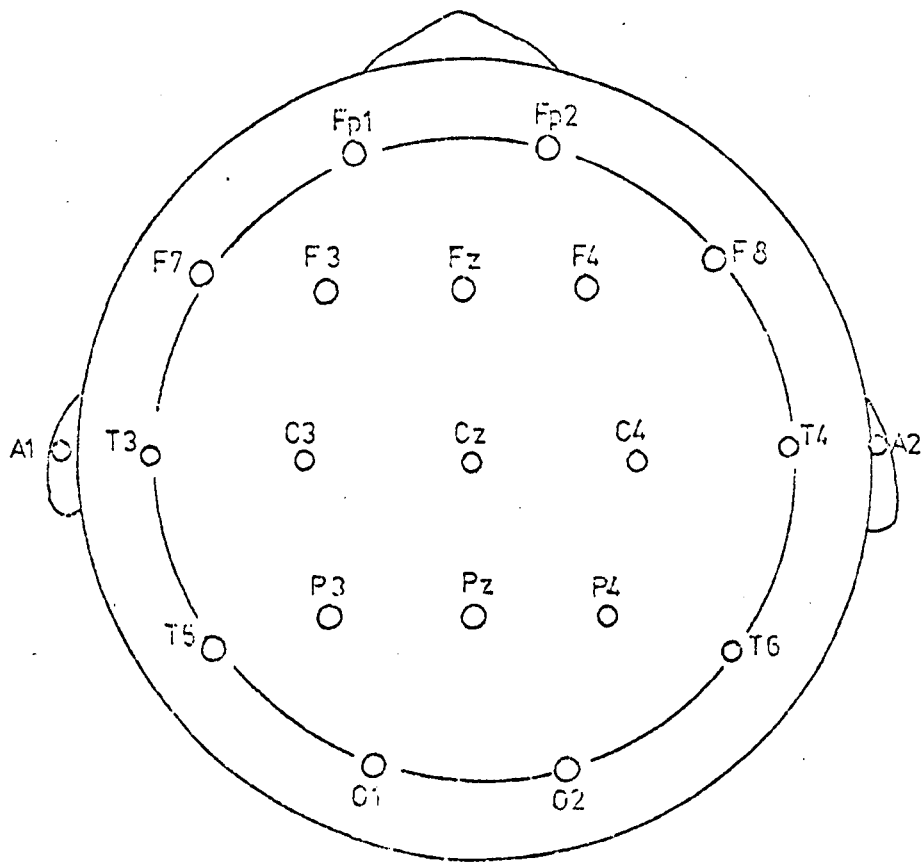


Figure 5.6 Electrode montage for EEG recordings.

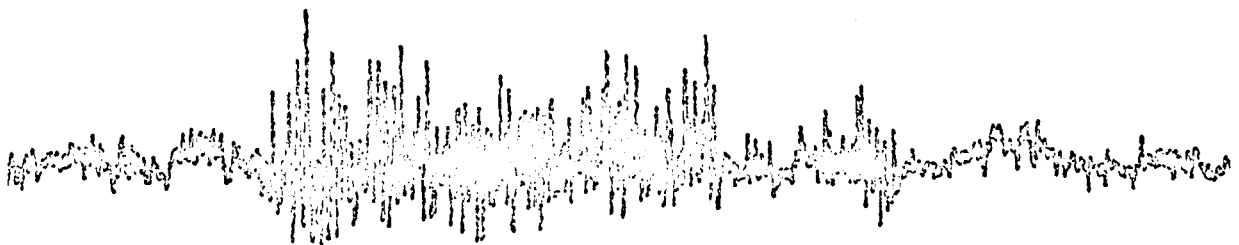
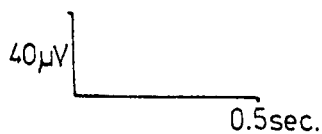


Figure 5.7 A typical burst of artifact behaviour.

suitable algorithm. A more troublesome form of artifact behaviour is described by Mulsby [38] and is specifically concerned with artifact introduced by eye movement (Fig. 5.8). It is felt that little can be done to reject behaviour of this type without degrading the overall performance of the spike detection system. Mulsby [38] states that '... spikes and sharp waves of cerebral origin always occupy a definable electrical field on the scalp and should always be seen in

two or more nearby electrode sites'. Hence, one possible solution to this problem may be to use the signals from two or more electrode sites and base the detection criteria on their joint behaviour.

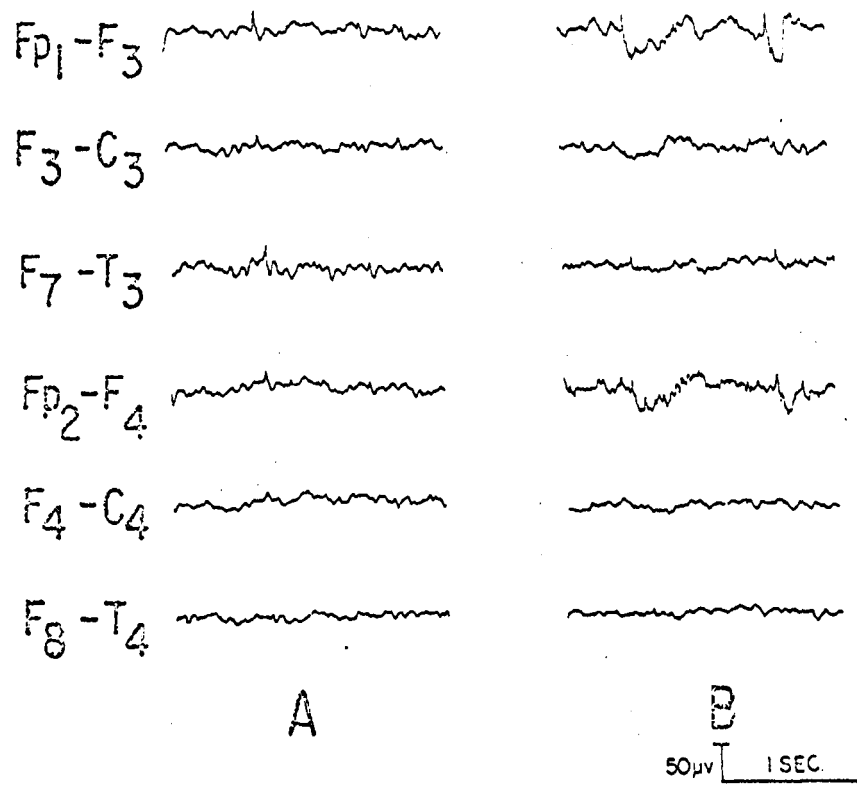


FIGURE 5.8

Samples of artifactual spikes produced by extrinsic eye muscles. A: Tracing from a 35 year old woman with a history of mild head trauma; normal neurological examination and skull X-rays. A neurologist interpreted this record as abnormal, showing bi-frontal spike discharges. High frequency filters were used. B: The myogenic origin of these frontal spikes is more obvious because the high frequency filters were off, and because the spikes were always followed by an obvious eye movement potential. This 27 year old woman was referred for headaches; neurological examination was normal.

Before a model is finalized, another observation of Smith et al[39] concerning EEG analysis was noted; '... Almost all of the effort is concerned with a frequency decomposition of the EEG (i.e. an analysis of the inherent periodicities). Although this has provided considerable information, many of the significant patterns are aperiodic To detect these waveforms, different data-processing techniques must be employed'.

This comment adds weight to a personal conviction that a time rather than a frequency domain analysis of the EEG is

likely to yield more profitable results.

The reasoning behind this opinion lies in the fact that the EEG signal is non-stationary, or time-varying, and this presents certain problems, particularly for frequency analysis schemes. In the real world, time invariant, or stationary, processes are seldom found but many vary so slowly with regard to time that they may be considered as stationary in the short term.

A stationary process can be defined as one for which the statistical properties of the signal do not change with time[35]. This is the strict definition of stationarity, a more general definition is that the mean and variance estimates of $x(t)$ do not vary with time. Signals of this type are generally referred to as 'weakly stationary processes' but for most practical purposes can be considered as if they were completely stationary.

Wide-sense stationarity can be defined by saying that a process is stationary if the expected value is a constant and the autocorrelation:-

$$R_{xx}(t_1, t_2) = E\{x(t_1) \cdot x(t_2)\}$$

can be expressed as a function of one variable, the time difference $\tau = t_1 - t_2$ [21]:-

$$R_{xx}(\tau) = E\{x(t + \tau) \cdot x(t)\}$$

where $E\{x(t + \tau) \cdot x(t)\}$ represents the ensemble average. Note, the expected value, $E\{x(t)\}$ equals a constant.

It is known, however, that the statistical characteristics of the EEG may change appreciably with time, depending upon the state of the subject (e.g. eyes open - eyes closed) (Fig. 5.9).

Campbell et al[40] concluded that the EEG record was non-stationary over the 60 sec. epochs they used and Elul[41], using 2 sec. epochs, found that the resting EEG followed a Gaussian distribution for only 66 percent of the time and this fell to 32 percent during the performance of an arith-

metric task. Huber et al[42] suggests that even for 1 sec. epochs, significant non-stationarities can be revealed in EEG data by the use of bispectral analysis.

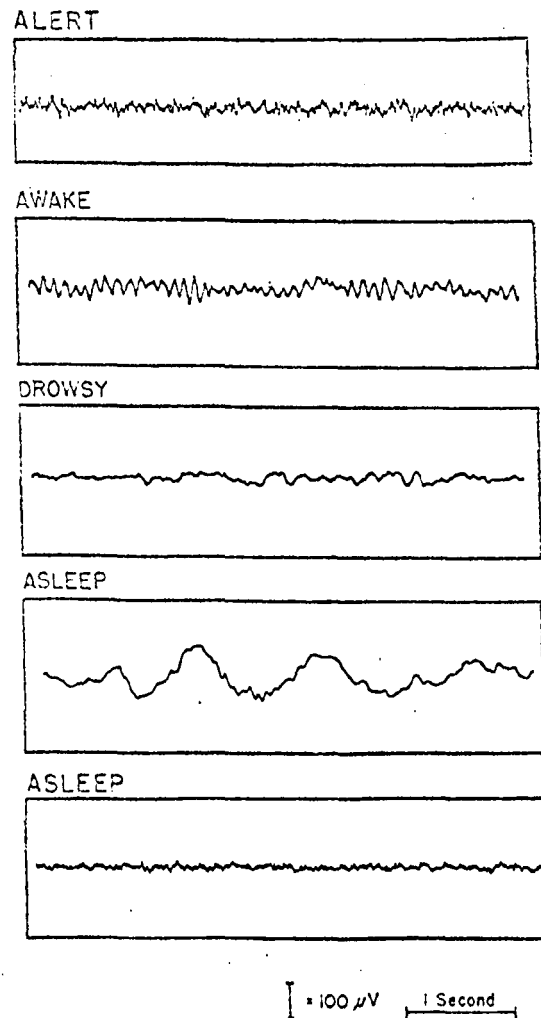


FIG. 5.9 CHANGES IN THE NORMAL ELECTROENCEPHALOGRAM DURING ALERTNESS, RELAXATION, DROWSINESS AND NOCTURNAL SLEEP

The delta activity is usually seen in the early part of the night. The 'spindling' superimposed on the slow waves is characteristic of this stage. As the duration of sleep lengthens these features tend to drop out as shown in the lowest strip.

Kawabata[43] recognizes the problem of non-stationarity but then proceeds with a frequency domain analysis and a discussion of how the inherent difficulties may be overcome.

Various techniques may be employed to deal with non-stationary processes which range from the assumption of a piecewise stationarity to the application of advanced mathematical methods[35]. The mathematical methods provide for a more rigorous analysis of the signal but unfortunately tend to generate physically meaningless results(e.g. a two-

dimensional power spectrum)[35].

The piecewise approximation makes use of the assumption that if a continuous record is divided into a number of sufficiently short sections (in time) each section may be considered as a stationary signal[35] (Fig. 5.10). Obviously, the choice of value for ΔT is of great importance and will be fixed by the rate of change of the signal parameters with time.

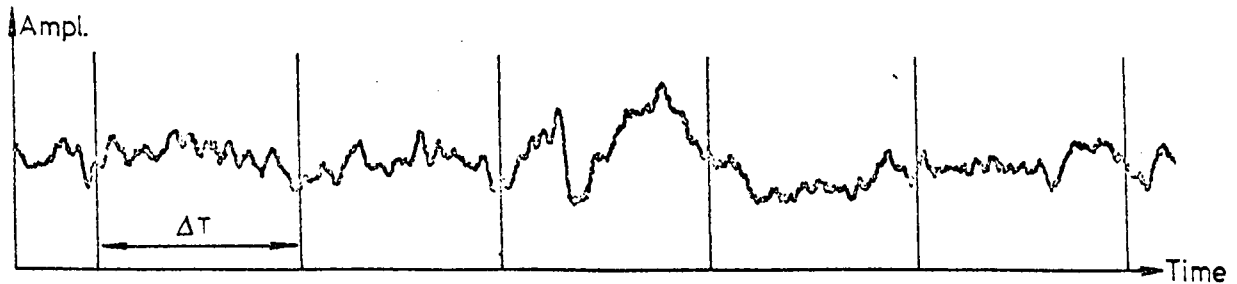


Figure 5.10 Application of the piecewise approximation.

The uncertainty principle (see later) precludes the formation of an instantaneous spectrum, although the idea of a changing, short-term spectrum is often used to give a three-dimensional representation in the ω - t plane[35]. For all but very slowly varying processes, though, this representation is likely to be fairly crude[21].

Heisenberg's uncertainty principle states that uncertainties involved in the specification of two conjugate characteristics can never be less than a constant. By the use of the Schwarz inequality[44], this can be expressed:-

$$\Delta E \times \Delta t \geq \frac{h}{2\pi}$$

where E and t represent the energy and time respectively and h is Plank's constant. From the above relationship it can be seen that as one of the conjugate variables is specified more precisely, e.g. if $\Delta t \rightarrow 0$, then the uncertainty of the value of the other, ΔE , becomes correspondingly larger.

Frequency and time form such a conjugate pair in signal

analysis, a fact which may be illustrated by a consideration of a typical signal processing operation, e.g. a filter. If a direct filter implementation is assumed (see Chap. 7) and that a measurement has been performed at time δt before t and a further measurement, taken at time t , is to be statistically independent of the previous measurement at $t - \delta t$, a zero memory system is required. To achieve this, the filter must not contain any information from the previous measurement, i.e. its output must fall to zero during time δt .

As the time between samples, δt , is decreased (i.e. $\delta t \rightarrow 0$) then the Q-factor of the filter must also tend to zero, in which case the filter will be of infinite bandwidth and it will not be possible to discriminate a particular frequency, f . Similarly, if the filter is required to resolve a particular frequency, f , an infinitely narrow filter and hence infinite Q-factor is required which implies an infinite memory filter[35].

The interdependence between the time and frequency scales can be expressed in a more rigorous form, from a consideration of the basic Fourier transform. The Fourier transform of a signal $x(t)$ is defined by:-

$$\mathcal{F}[x(t)] = X(\omega) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j\omega t} \cdot dt$$

If the variable in the time function is multiplied by a constant this will have the effect of expanding or contracting the time scale, depending upon whether the magnitude of the constant is greater or less than unity. The Fourier transform of the scaled time function is:-

$$\mathcal{F}[x(at)] = \int_{-\infty}^{\infty} x(at) \cdot e^{-j\omega t} \cdot dt$$

and changing the variable of integration to $\lambda = at$:-

$$\mathcal{F}[x(at)] = \frac{1}{a} \int_{-\infty}^{\infty} x(\lambda) \cdot e^{-j\omega \frac{\lambda}{a}} \cdot d\lambda$$

$$= \frac{1}{a} X\left(\frac{\omega}{a}\right) \quad a > 0$$

If a is negative, which would correspond to a reversal of the time scale, the limits of integration will be reversed when the variable is changed:-

$$\mathcal{F}[x(at)] = -\frac{1}{a} X\left(\frac{\omega}{a}\right) \quad a < 0$$

These two results can be combined in one expression:-

$$\mathcal{F}[x(at)] = \frac{1}{|a|} X\left(\frac{\omega}{a}\right)$$

Thus, it can be seen that as the time scale of a function is contracted, so the frequency spectrum is expanded and vice-versa. The $1/|a|$ term is necessary to maintain the energy balance between the two domains.

There are, however, certain problems involved when strict limits are imposed on a signal. The formation of a finite record, $x(t)$, in the time domain, for example, can be considered as equivalent to the result of multiplying an infinitely long random signal, $y(t)$, with a finite rectangular window (Fig. 5.11). The finite record inevitably begins and ends abruptly and this introduces errors into the signal, i.e. additional frequencies (side-lobes) will be added to the original data.

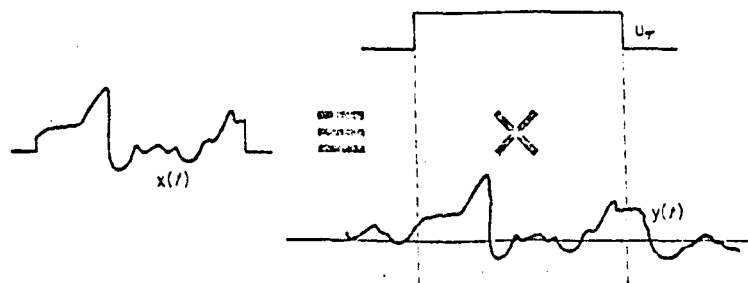


Fig. 5.11 Equivalent finite length record.

These effects can best be explained by a consideration of the very important Fourier transform pair, the block function:-

$$b(x) = U(x + 1/2) - U(x - 1/2)$$

and sinc function $(\sin x)/x$, which are Fourier transforms of each other (Fig. 5.12)[45].

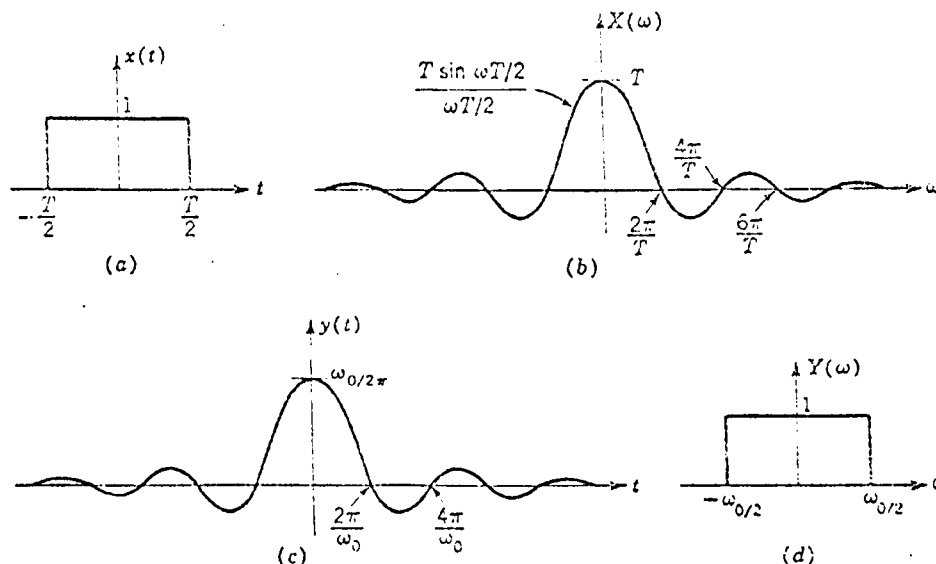


Fig.5.12 Fourier transform pairs which illustrate symmetry of Fourier transform and inverse Fourier transform.

If a sharp band limit is imposed upon the signal (i.e. multiply signal by $b(\omega)$) the effect is to convolve with the oscillating sinc function in the time domain: Gibbs phenomenon [46]. Equally, if a sharp time limit is imposed, the effect is to convolve the sinc function in the frequency domain (smearing) [35].

The principal effect of this behaviour is manifest as a 'leakage' of energy from individual frequencies into other adjacent frequencies. Various techniques are employed to help minimize the effects of this leakage, many of which make use of window functions to modify the signal. The purpose of these window functions is either to smooth the response in the frequency domain or round the corners of the block function in the time domain.

As an example, consider the low-pass filter:-

$$H(z) = \frac{1}{4} z + \frac{1}{2} + \frac{1}{4} z^{-1}$$

Its effect in the frequency domain is:-

$$H(\omega) = \frac{1}{2} (1 + \cos 2\pi\omega/\omega_s)$$

The Hann window, as an example, operates on a reversal of this arrangement; the low-pass filtering operation is performed on the ordinates of the spectral estimates as though they constituted a time series. This is exactly equivalent to multiplying the time data block function by the cosine bell function, expressed earlier for the frequency domain [21] .

Numerous types of window function are available, e.g. Hann, Chebychev, Parzen [35] [49] all offering different advantages and disadvantages, the choice being determined by the result required. It is proposed that no window functions shall be employed for the present analysis and so the subject will not be pursued further, at this stage. It is possible, however, that the problem will require re-evaluation at a later date, after the preliminary studies have been completed.

If the features of the EEG signal to be detected are now considered, the choice of a time domain analysis becomes obvious. To summarize, the sought features are sharp spikes in the EEG record which are almost invariably surface negative in polarity [38]. A spike may be defined as having a definable base of 1/12 sec. or less. Many spike, and sharp wave, discharges are followed by a slow wave or a series of slow deflections. This, however, is not always the case and so will be omitted from the present model.

Clearly, if the spike detection process were considered as a frequency domain problem, a fairly precise frequency 'window' must be specified which would necessitate the use of long periods of EEG record. As discussed earlier, it is desirable to be able to consider the EEG signal as stationary and, for this assumption to hold, fairly short records (1 sec. or less) must be used. The non-stationarity problem places certain predefined restrictions on the value of Δt and it would seem logical to consider the whole spike detection

problem as a time domain problem. This will mean that Δt must be defined more precisely, at the expense of frequency information, and it becomes possible to perform the analysis over very short time periods (see later) which will further improve the stationarity assumption.

Carrie [11] used a method based on a running mean of the input EEG signal (see Chap. 2). The running mean performs an averaging, or low-pass filtering, operation on the input data by the formation of an output sequence from an unweighted average of several consecutive input samples:-

$$y_k = \frac{1}{n} \sum_{r=0}^{n-1} x_{k-r} \quad (i)$$

which, via the use of the z-transform (see Chap. 6) may be expressed as:-

$$H(z) = \frac{1}{n} \sum_{r=0}^{n-1} z^{-r} \quad (ii)$$

This represents a non-recursive filter and may be computed more efficiently if expressed in a recursive form [21]. The geometric progression of equation (i) may also be expressed as:-

$$H(z) = \frac{1}{n} \cdot \frac{1 - z^{-n}}{1 - z^{-1}} \quad (iii)$$

for which the inverse transform is:-

$$y_k = y_{k-1} + \frac{1}{n} (x_k - x_{k-n}) \quad (iv)$$

The choice of value for n does not appear to have any formalized basis and is more a trial and error decision. Obviously, as the number of samples, n , is increased, so the amount of filtering will be increased. If n is made too large, problems will arise as a consequence of the non-stationary character of the EEG waveform and, further, in extreme cases the degree of filtering may be so large that the spike features may be removed altogether! Equally, a reasonable value must be

chosen for n if the running mean is to perform any useful filtering, or averaging, operation.

Other problems associated with the running mean type of filter become apparent when the effects of the operation are considered in the frequency domain. Substituting $z = e^{j\omega T}$ into equation (iii) yields:-

$$\begin{aligned} H(\omega) &= \frac{1}{n} \cdot \frac{1 - e^{-nj\omega T}}{1 - e^{-j\omega T}} \\ &= e^{-(n-1)j\omega \frac{T}{2}} \cdot \frac{\sin(n\omega T/2)}{n\sin(\omega T/2)} \end{aligned}$$

The exponential term is the phase term and since it has unit modulus, the effect on the amplitude is given by:-

$$\begin{aligned} |H(\omega)| &= \left| \frac{\sin(n\omega T/2)}{n\sin(\omega T/2)} \right| \\ &= \left| \frac{\sin(\pi n\omega/\omega_s)}{n\sin(\pi\omega/\omega_s)} \right| \end{aligned}$$

An examination of this function shows that $|H(\omega)| \rightarrow 1$ as $\omega \rightarrow 0$, so zero and low order frequencies are unaffected. The behaviour over the rest of the spectrum is determined by the zero values of ω/ω_s at $1/n, 2/n, \dots (n-1)/2n$, for n odd, and $1/n, 2/n, \dots 1/2$ for n even. The numerator is periodic but the denominator steadily increases as $\omega/\omega_s \rightarrow 1/2$ and the resultant effect is the formation of side-lobes of decreasing magnitude (c.f. sinc function). Hence, although the running mean provides a form of low-pass filter, the elimination of frequencies beyond the cut-off is not absolute. Some frequencies will be removed completely whereas others will be only partially suppressed. Examples of $H(\omega)$ for two values of n are given in Figure 5.13.

The idea of a running mean, or moving average, is an attractive one when the phrase '... a wave distinguished from background activity ...' in the official definition, is

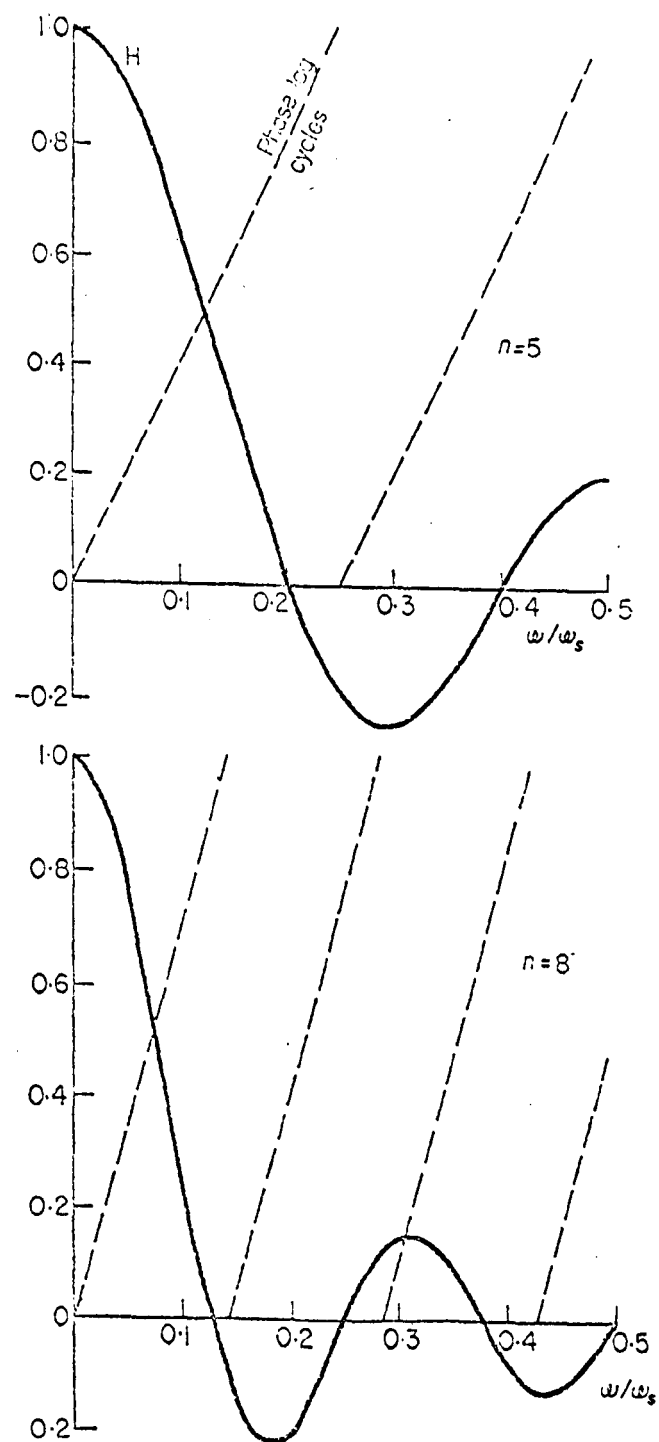
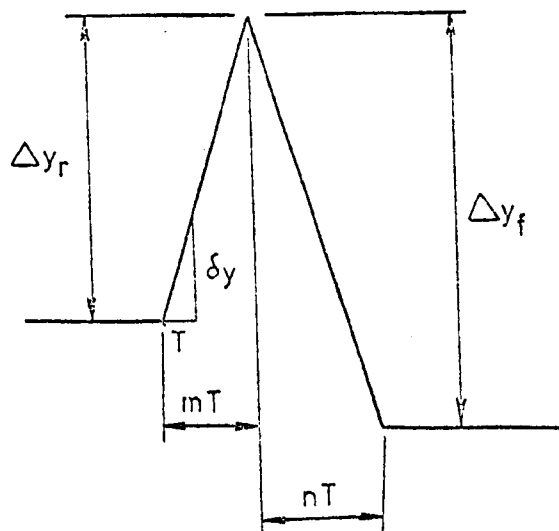


Figure 5.13 Amplitude and phase characteristics of the Running Mean smoothing formula for $n = 5$ and $n = 8$.

considered. Carrie reports that good results were obtained (i.e. a close agreement was obtained between the computed results and those from a human observer), but it is considered that the running mean represents a very poor form of low-pass filter and that the method is unsuitable for the present analysis. Assuming a filtering operation is required, it is the author's opinion that a much more precise and well defined digital filter will be required if the results are to have any significance.

The mathematical model to be used to represent the spike feature is similar to that used by Smith [12] and is basically a triangle (Fig. 5.14). It comprises a smooth positive-going slope, over m samples (equal time divisions) followed by a smooth negative-going slope of n samples (n.b. this corresponds to a surface negative spike). If the rate of change of slope of these two edges with respect to time (δy) exceeds a preset level and the total number of samples ($m + n$) is less than a second preset value, then the feature may be classified as a spike.



- Δy_r - height of rising edge of spike
- Δy_f - height of falling edge of spike
- δy - change in magnitude per unit time
- T - one time division (i.e. $1/w_s$)
- m - number of samples on rising edge
- n - number of samples on falling edge

Figure 5.14 Model of spike waveform.

Considering one edge, from Figure 5.14 :-

$$\Delta y = m \cdot \delta y$$

and from the definition given earlier, the period $0 - m$ must be $1/12 \times 1/2$ sec. or less (the $1/12$ sec. period refers to the complete spike, so for one edge this must be halved). It is proposed that a sampling rate of 500Hz be used, so each time division corresponds to $1/500$ sec.. If, for convenience, the period $0 - m$ is taken as $1/25$ sec., then:-

$$m \leq \frac{500}{25} \quad \text{or} \quad m \leq 20$$

Hence:-

$$\delta y \geq \frac{\Delta y}{20}$$

which sets two criteria on which to base the detection algorithm. Both are dependent upon the value chosen to represent the peak amplitude of the spike (Δy), since the minimum value of Δy will set a lower limit for m .

The only suitable method for the selection of the value of Δy is by a visual inspection of a number of abnormal records to give an estimate and then by a series of tests in which a spread of values around the estimated value are tried.

For the purposes of testing, an FM tape recording of actual data was used, in which the signal had been amplified to $0 - 2.0V$ in amplitude. From an inspection of six such abnormal records, sharp spikes varying in amplitude from $0.3 - 2.0V$ were observed. The lower input value sets a minimum value of:-

$$y = \frac{0.3}{20} = 0.015V$$

for the difference between successive samples.

If the output is not to be noise dominated, a ten-bit analogue-to-digital converter will be required which would quantize the input in $1mV$ steps. The system to be used is

equipped with an eight-bit analogue-to-digital converter (see Chap. 6), which gives a quantization step of 10mV, and so the basic algorithm will require some modification. Basically, what is required is a means of increasing the value of δy . A simple method of achieving this aim is to decrease the sampling frequency. A sampling frequency of 250Hz is regarded as a minimum for an accurate representation of the input signal (which is band-limited to 70Hz). This gives a value of 30mV (i.e. three input levels) for δy .

This is not, however, considered a very satisfactory solution and it is probable that this simple modification will prove inadequate and that the algorithm will remain very noise sensitive. As a consequence, an alternative method was sought. One such method of increasing the magnitude of the spike activity, in relation to the ongoing background activity, is to perform a differentiation operation on the input data, which will yield a different model for the detection process.

Unfortunately, past analysis of the differentiation process suggests that the resulting output will contain a considerable amount of high-frequency noise (see Chap. 6). This arises, again, as a consequence of the input quantization errors. It is therefore usually necessary to follow the differentiation process with a low-pass filtering operation to remove this noise, before any spike detection algorithm is applied (see Chap. 7).

The differentiation process will produce a spike for any high-frequency edge, the polarity of the spike being determined by the polarity of the slope. A spike in the original signal will be represented as a double spike in the differentiated output, of opposite polarities (Fig. 5.15).

The use of this technique suggests a possible method by which the detection process may be simplified considerably. A simple level detection process applied to the differentiated

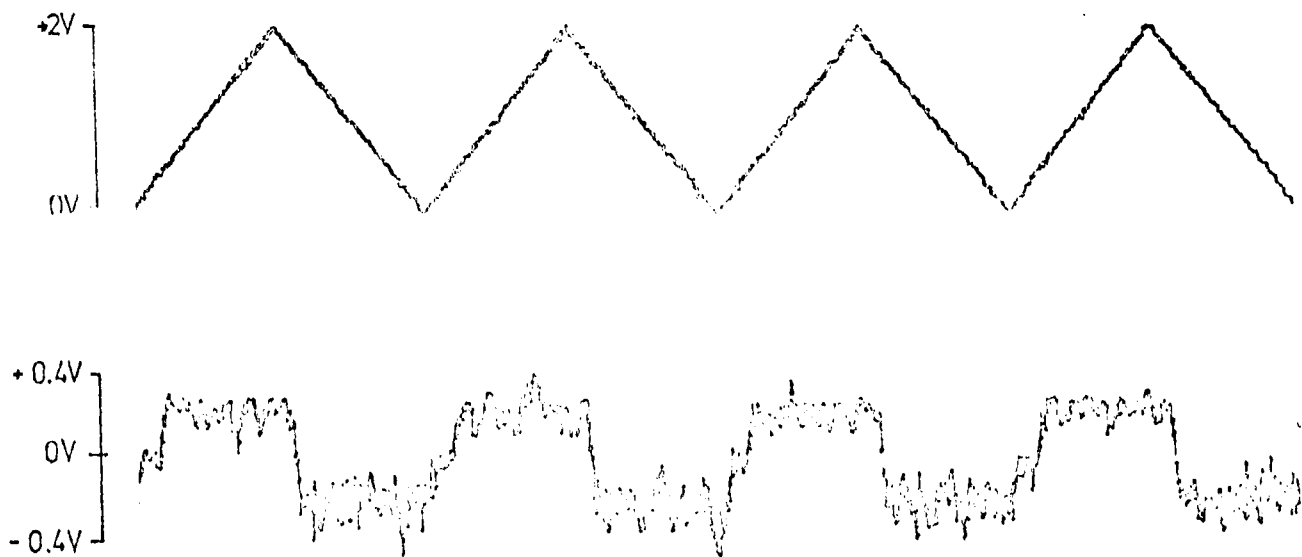


Figure 5.15 Effect of differentiation process on spike (triangular) input.

and filtered waveform may be useful in isolating the edges in the original record from the ongoing activity. The use of this technique would give an output of the form shown in Figure 5.16, for a typical spike.

If the level is set correctly, it should be possible to reject any edges which are not sufficiently sharp to be classified as a spike edge. The success of the method will depend upon the amount of emphasis given to the spike activity over the ongoing EEG activity. As for the determination of the value for Δy , an inspection of experimental data is the only suitable method of determining the effectiveness of the level detection algorithm (see Chap. 7). If the method proves satisfactory, the complete recognition algorithm can be reduced to the simple task of ensuring that a positive-going pulse is followed by a negative-going pulse. The only additional requirement is that the cross-over region is of a sufficiently short duration, which determines the sharpness of the apex, and this may be implemented as a simple count of the number of samples between the two pulses output from the level detector. The upper bound set on this number will set the discrimination level between spikes and sharp waves.

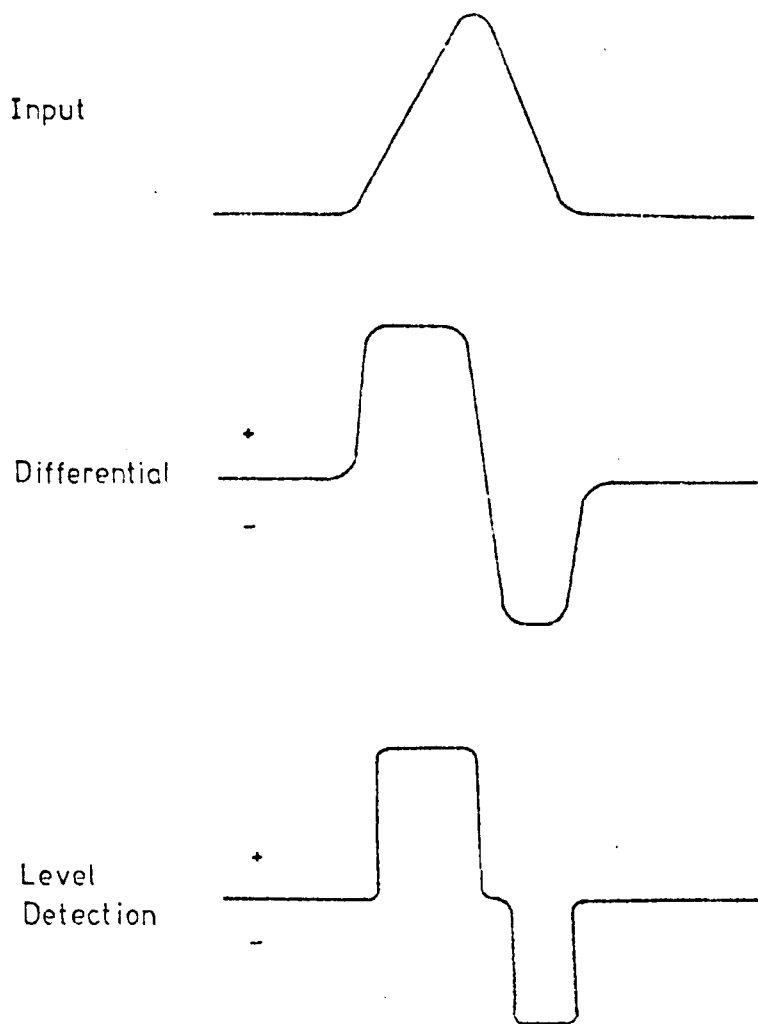


Figure 5.16 Effect of level detection process on output waveform.

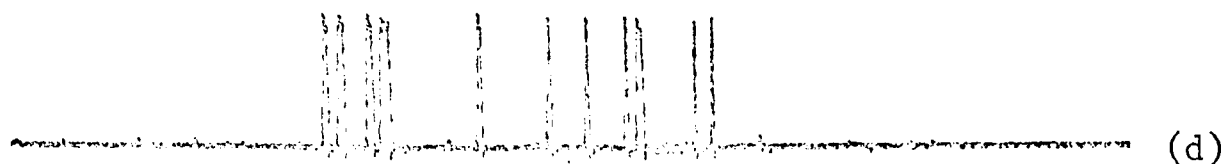
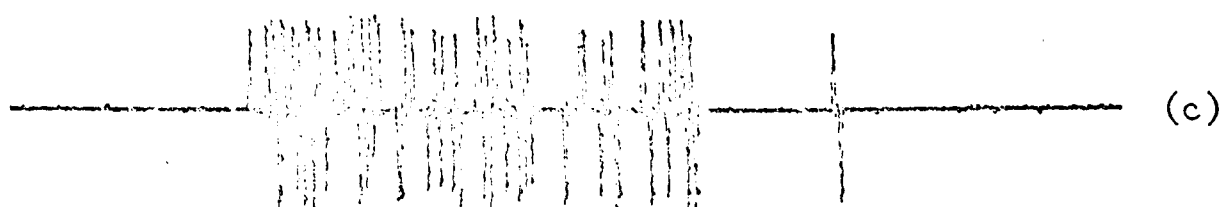
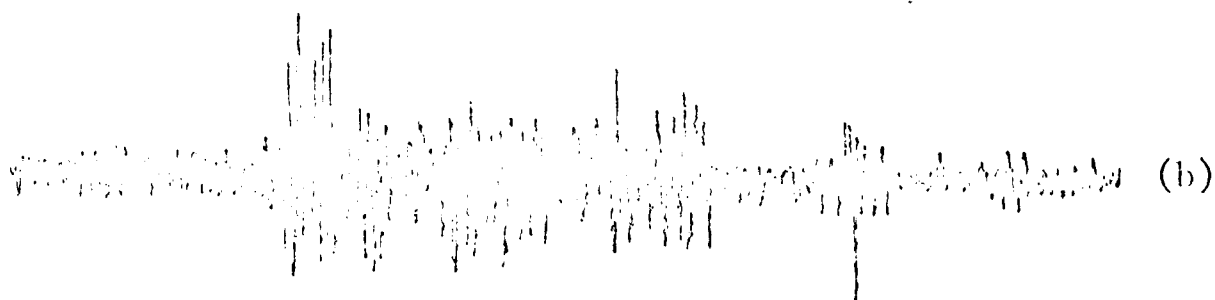
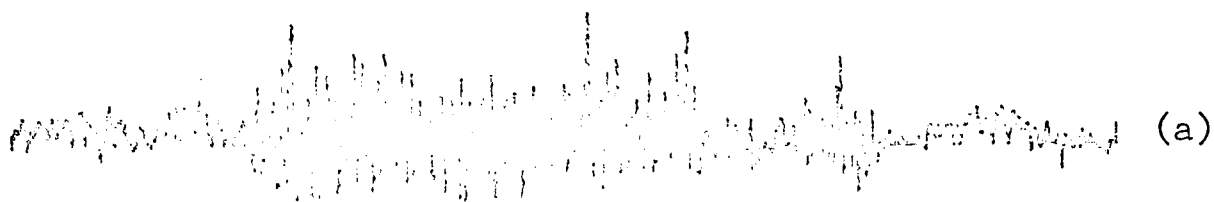
It is considered that this method of determination of the sharpness of the apex is vastly superior to the more normal second differential approach. The effect of a second differential process would be to provide further emphasis to the high-frequency noise, already amplified by the first differentiation. This would result in an extremely noisy output, and make the task of accurate spike detection considerably more difficult.

If the method described proves suitable, the only remaining problem is that of artifact rejection. As discussed earlier, it is a virtually impossible task to reject all artifact behaviour, but an attempt must be made to minimize the number of false alarms due to this cause. By far the commonest form of artifact is a short burst of high-frequency

activity (see Fig. 5.7). One possible method of providing some degree of protection against this behaviour is to perform a running count of the number of zero-crossings which occur in the differentiated signal. When artifact is present, this should be increased considerably from the value obtained from an artifact-free record.

It is proposed that the zero-crossing count be performed on the differentiated and filtered output rather than after the level detection process, since it is probable that the difference between the artifact and artifact-free counts will be greater at this stage of the operation. The artifact behaviour will, in general, be of a much higher frequency than the spike activity and the filter will hence tend to reduce its amplitude, so that only the larger peaks cross the threshold level. As a result, it would not be possible to reject the first artifact peak if the count is performed on the level detector output, since the detection process is to work in real-time. There is at least a finite probability that this first peak may be rejected using the method described, since there will normally be a period of increased zero-crossing activity before the first level detector output, on which to base the rejection decision (Fig. 5.17).

The next section will deal with the constraints imposed on the processes, discussed in this chapter, by the use of the RSP for their implementation.



- a) Input waveform.
- b) First order difference.
- c) Level detector output.
- d) Spike detector output.

Figure 5.17 Zero-crossing count method of artifact rejection.

Chapter 6

Signal

Processing

Considerations

A consideration of the restrictions imposed by the use of the RSP must be made before any of the techniques discussed in the previous chapter are implemented. Particular attention will be given to the proposed differentiation and filtering operations, as these present the most demanding problems. The major constraints will arise as a consequence of the instruction execution speed and sixteen-bit wordlength.

Obviously, the operating speed of the microprocessor will impose restrictions on the bandwidth capabilities of the system. The actual instruction execution times of the microprocessors considered, are comparable with those offered by current minicomputers. Minicomputers, however, generally offer a higher operating speed by virtue of their longer instruction word. The power of individual instructions is a direct function of the number of bits available for their description. Hence, a minicomputer with a wordlength of greater than sixteen-bits will be able to specify more complex operations as individual instructions than are possible with a sixteen-bit microprocessor. As a result, the minicomputer will in general be able to execute a given program more rapidly than the microcomputer.

Part of the overall design philosophy of the RSP is the provision of an 'add-on' hardware facility (see Chapter 4) so that any process that imposes a heavy (time consuming) load on the central processing unit (CPU) may be delegated to a special hardware unit (e.g. multiplication). However, for this particular application the eventual aim is to be able to perform the necessary computations in a miniature version of the RSP. Therefore, every effort must be made to ensure that all the necessary processing may be performed by the CPU so as to facilitate the minimization of the hardware at a later date.

To achieve this goal the real-time processes involved in the analysis should be kept as simple as possible and the code for their implementation as short as possible (the usual state

of affairs for real-time programs).

The available wordlength also has a direct bearing on the accuracy obtainable with the system. The inaccuracies will manifest themselves as quantization errors in the representation of system coefficients and as rounding or truncation errors incurred in mathematical operations as a consequence of the need to prevent overflow.

The use of a sixteen-bit wordlength permits coefficients to be specified to one of 65,536 levels (i.e. 1.5 parts in 10^5 or 0.0015%) or 32,768 levels (i.e. 3 parts in 10^5 or 0.003%) if the most significant digit is reserved as the sign bit, as is normal. Clearly, it will be possible to describe many coefficients to a suitable accuracy within this range but the results of any processing will naturally require additional bits for their representation. As an example, an n -bit data sample multiplied by an n -bit coefficient will produce a product of $2n$ bits. The effects of quantization errors in this context will depend upon such factors as whether fixed or floating-point arithmetic is used and whether, for fixed-point work, a fractional or integer representation is employed.

It is normal practice in signal-processing applications to use a fixed-point fractional representation whenever possible so that the product of two numbers can be contained within the limited register length by truncation or rounding of the least significant digits. Note that with this representation there is no need to truncate or round the results of an addition operation, although care is needed to ensure that overflow does not occur (i.e. the result cannot be represented as a fraction). For floating-point arithmetic, dynamic range can generally be ignored owing to the large range of representable numbers, but now quantization errors will be introduced for both multiplication and addition operations.

6.1 Mathematical Representation

If a binary number is considered to be described by $n+1$ bits, with the extra bit reserved for sign representation, then considering the fixed-point case, if E_T denotes the error due to truncation (i.e. value after truncation minus value before) then, for positive numbers:-

$$\text{magnitude before truncation } M_1 = \sum_{i=1}^{n_1} a_i 2^{-i}$$

$$\text{and magnitude after truncation } M_2 = \sum_{i=1}^{n_2} a_i 2^{-i}$$

where $n_2 < n_1$.

The magnitude of the error E_T is then:-

$$\begin{aligned} E_T &= M_2 - M_1 \\ &= \sum_{i=1}^{n_2} a_i 2^{-i} - \sum_{i=1}^{n_1} a_i 2^{-i} \\ &= - \sum_{i=n_2+1}^{n_1} a_i 2^{-i} \end{aligned}$$

$$\text{i.e. } E_T \leq 0$$

that is the error tends to decrease the value of the quantity to be represented.

The position for negative numbers is complicated by the lack of standardization for their representation. Three methods are in common use:-

- i) Sign-magnitude in which the most significant digit is used as the sign bit (0 for positive and 1 for negative) with the rest of the word used to represent the magnitude.
- ii) One's complement in which the negative number is represented by subtracting the magnitude from the largest number representable in the register.
- iii) Two's complement in which the negative number is represented by 2.0 minus its magnitude.

The effects of these different representations on the truncation error are given in Table 6.1.

The rounding process is independent of the method used to represent negative numbers. The error is always in the

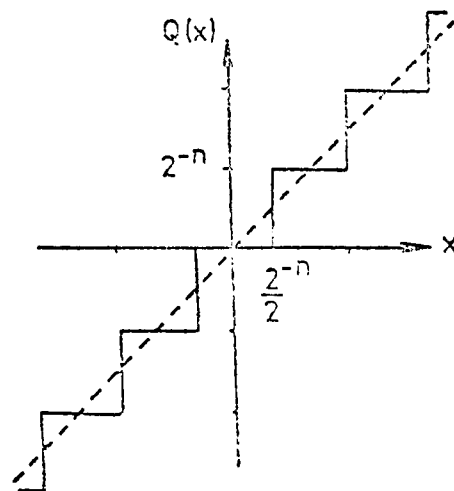
range:-

$$-\frac{1}{2} 2^{-n} \leq E_T \leq +\frac{1}{2} 2^{-n}$$

i.e. plus or minus half the least significant bit.

A graphical representation of the effects of the various errors is given in Figure 6.1.

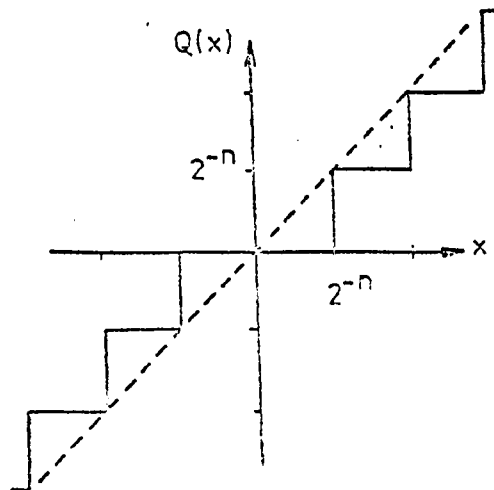
It should be noted that, for floating-point numbers only the mantissa will be affected by truncation or rounding errors.



$$E_T = Q(x) - x$$

$$-\frac{1}{2} \cdot 2^{-n} \leq E_T \leq +\frac{1}{2} \cdot 2^{-n}$$

a) Rounding

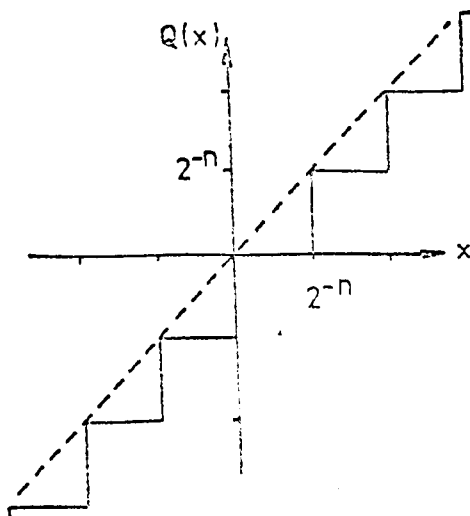


$$0 \leq E_T < 2^{-n}; x > 0$$

$$0 \geq E_T > -2^{-n}; x < 0$$

b) Truncation:

1's Complement and
Sign Magnitude



$$0 \leq E_T < 2^{-n}$$

c) Truncation:

2's Complement

Figure 6.1 Graphical representation of quantization errors given in Table 6.1.

i) Sign-magnitude.

$$\begin{aligned} E_T &= (2^0 + \sum_{i=1}^{n_2} a_i 2^{-i}) - (2^0 + \sum_{i=1}^{n_1} a_i 2^{-i}) \\ &= - \sum_{i=n_2+1}^{n_1} a_i 2^{-i} \end{aligned}$$

$$\text{i.e. } E_T \leq 0$$

i.e. the magnitude of the negative value is decreased, that is the quantity is made less negative.

ii) One's complement.

$$\begin{aligned} E_T &= (2^0 + \sum_{i=1}^{n_2} 2^{-i} - \sum_{i=1}^{n_2} a_i 2^{-i}) - (2^0 + \sum_{i=1}^{n_1} 2^{-i} - \sum_{i=1}^{n_1} a_i 2^{-i}) \\ &= \sum_{i=n_2+1}^{n_1} a_i 2^{-i} - \sum_{i=n_2+1}^{n_1} 2^{-i} \end{aligned}$$

and since $\sum_{i=n_2+1}^{n_1} 2^{-i} > \sum_{i=n_2+1}^{n_1} a_i 2^{-i}$

$$E_T \leq 0$$

i.e. the magnitude of the negative value is decreased, that is the quantity is made less negative.

iii) Two's complement.

$$\begin{aligned} E_T &= (2^1 - \sum_{i=1}^{n_2} a_i 2^{-i}) - (2^1 - \sum_{i=1}^{n_1} a_i 2^{-i}) \\ &= \sum_{i=1}^{n_1} a_i 2^{-i} - \sum_{i=1}^{n_2} a_i 2^{-i} \\ &= \sum_{i=n_2+1}^{n_1} a_i 2^{-i} \end{aligned}$$

$$E_T \geq 0$$

i.e. the magnitude of the negative value is increased, that is the quantity is made more negative.

Table 6.1 Comparison of quantization errors introduced by different number representation schemes.

6.2 Input-Output Considerations

Before a digital processor can deal with an input from the real (time continuous) world it is necessary to perform a sampling operation on the input so as to represent the signal as a series of discrete samples. The process involves the sampling of the analogue input at equal time intervals, determined by the sampling frequency, and then representing the analogue amplitude by its nearest digital equivalent (analogue-to-digital conversion) (Fig. 6.2).

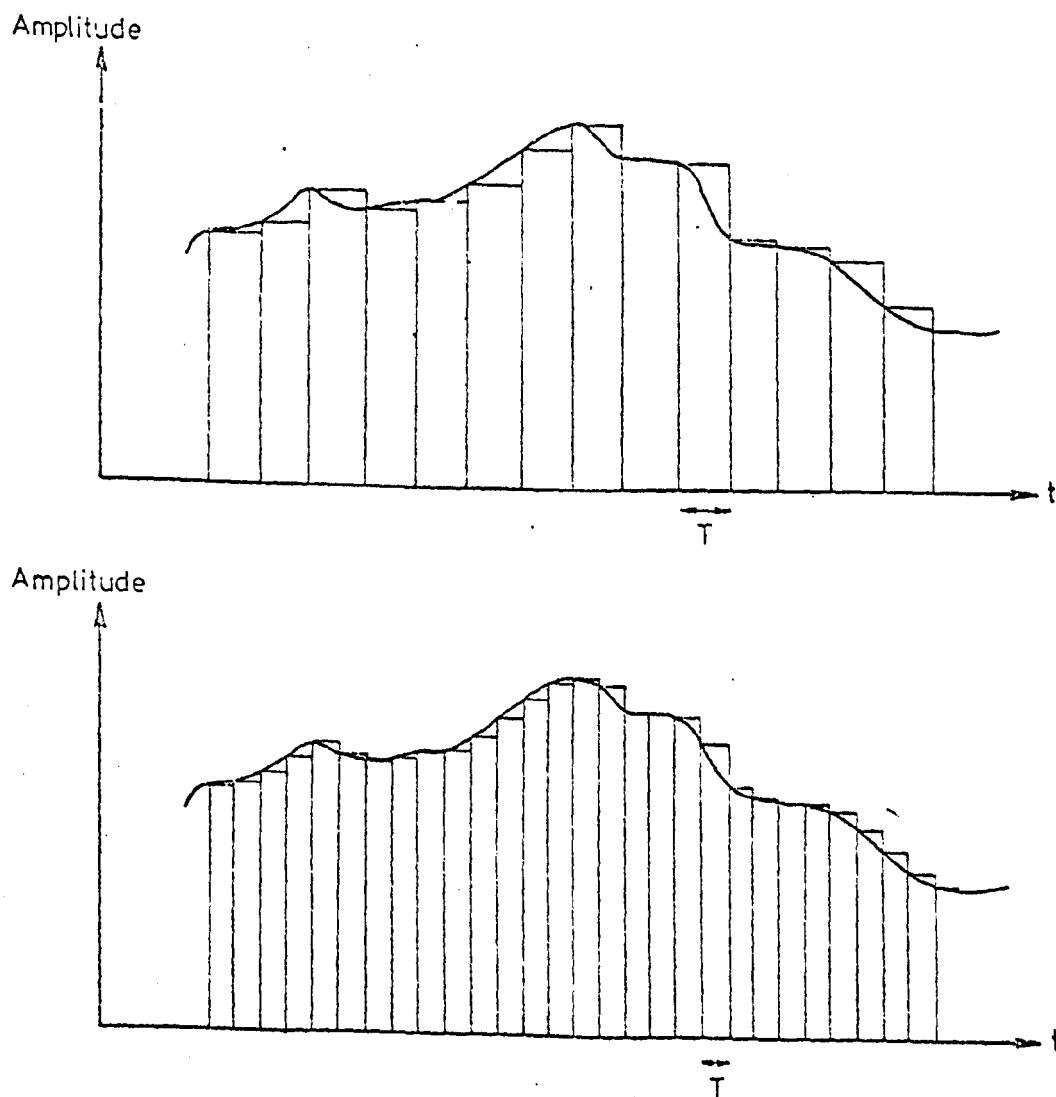


Figure 6.2 The analogue-to-digital conversion process and its accuracy dependence on the sampling frequency.

6.2.1 The Sampling Process

It is convenient to regard the sampling process as an impulse modulation of the continuous input signal [47]. This is shown diagrammatically in Figure 6.3, where an input signal $x(t)$ is sampled every T secs., to give an output signal of $y(t)$.

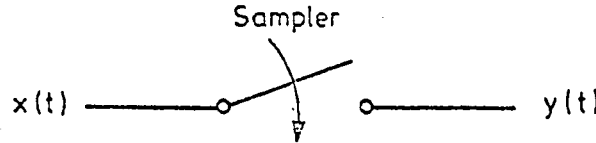


Figure 6.3 Sampler representation.

The ideal sampler can be represented by use of the Dirac delta function to express a unit impulse train $\sum \delta(t - nT)$. This corresponds to a series of equally spaced pulses at $t = nT$ secs. (n positive) and so, for a continuous input, the ideal sample unit impulse train may be represented as:-

$$\delta_T(t) = \sum_{n=0}^{\infty} \delta(t - nT)$$

Hence, the sampled output signal may be written:-

$$y(t) = x(t) \cdot \sum_{n=0}^{\infty} \delta(t - nT)$$

or, rewriting to include the input signal as a function of time, when $t = nT$:-

$$y(t) = \sum_{n=0}^{\infty} x(nT) \cdot \delta(t - nT) \quad (i)$$

i.e. the sampler output is an impulse train with an amplitude which is equal to the amplitude of the continuous input signal at the time of sampling (c.f. Fig. 6.2).

The Laplace transform of the ideal sampler is given by:-

$$\begin{aligned} \mathcal{L}[\delta_T(t)] &= \mathcal{L}\left[\sum_{n=0}^{\infty} \delta(t - nT)\right] \\ &= \sum_{n=0}^{\infty} e^{-nTs} \end{aligned} \quad (ii)$$

From this result, the Laplace transform of the sampler

output (i) can be expressed as:-

$$y(s) = \sum_{n=0}^{\infty} x(nT) \cdot e^{-nTs} \quad (\text{iii})$$

6.2.1.1 Time Domain Sampling

If a sinusoidal input signal is considered:-

$$x(t) = A \sin(\omega t)$$

then, from (i), the sampler output is:-

$$y(t) = \sum_{n=0}^{\infty} A \sin(\omega nT) \cdot \delta(t - nT)$$

From equation (iii) the Laplace transform of the sampler output for a sinusoidal input is:-

$$y(s) = \sum_{n=0}^{\infty} A \sin(\omega nT) \cdot e^{-nTs}$$

The input, sampler and sampler output signals, described by the above equations, are shown in Figure 6.4.

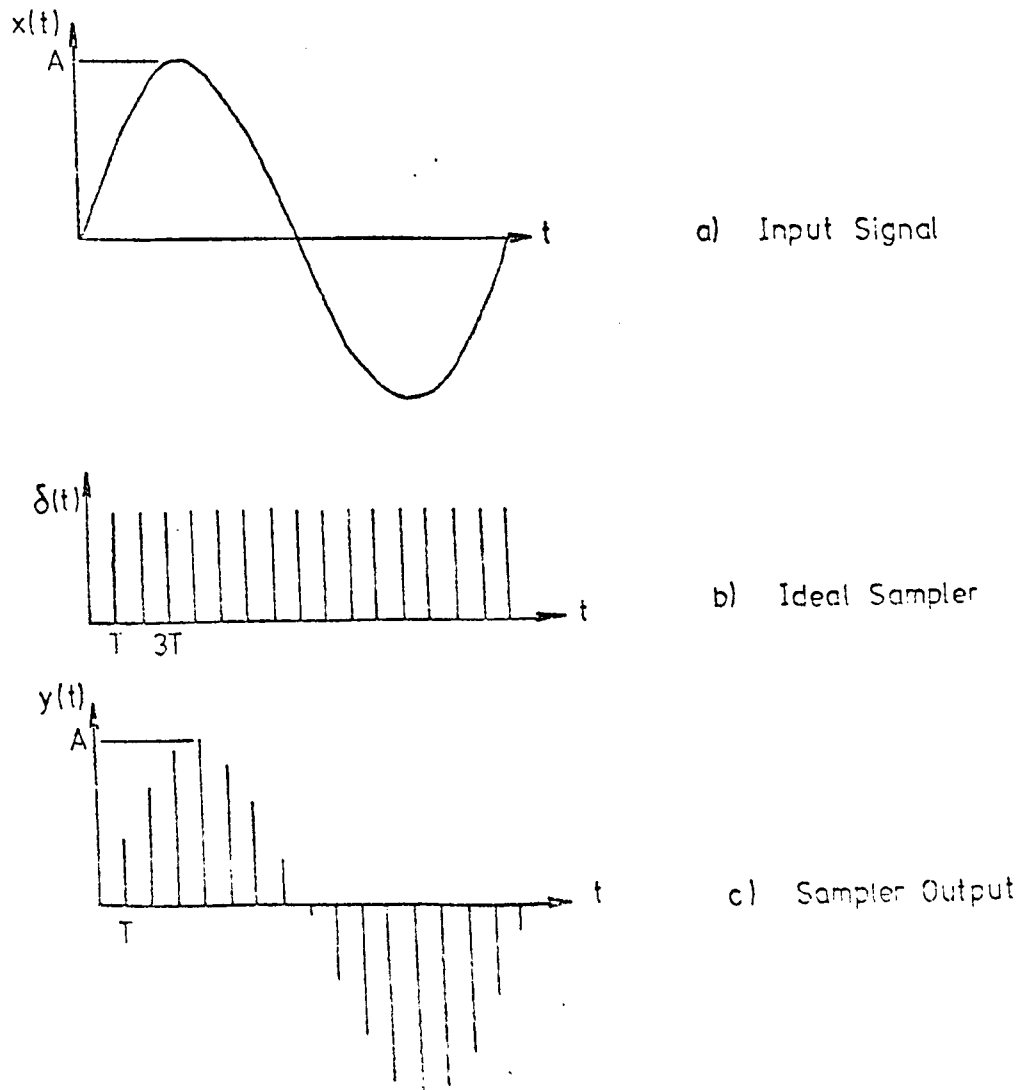


Figure 6.4 Time domain sampling.

As can be seen, a time domain analysis gives a useful interpretation of the characteristics of the sampled signal, but for a complete interpretation the frequency domain must also be considered.

6.2.1.2 Frequency Domain Sampling

For a frequency domain analysis, the Laplace transform of the ideal sampler (equ. (ii)) is expressed as:-

$$\sum_{n=0}^{\infty} e^{-nTs} = 1 + e^{-sT} + e^{-2sT} + e^{-3sT} + \dots$$

which, for $e^{-sT} < 0$, converges to:-

$$\frac{1}{1 - e^{-sT}}$$

The Laplace transform of the sampler output is given by the Laplace transform of the product of the input signal and the ideal sampler:-

$$\begin{aligned} y(s) &= \mathcal{L}[x(t) \cdot \delta_T(t)] \\ &= \sum_{n=0}^{\infty} x(nT) \cdot e^{-nTs} \end{aligned}$$

or may be rewritten as:-

$$\begin{aligned} y(s) &= \mathcal{L}[x(t)] * \mathcal{L}[\delta_T(t)] \\ &= X(s) * \frac{1}{1 - e^{-sT}} \end{aligned}$$

which, via contour integration, can be shown to be [47] :-

$$Y(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(s + jn\omega_s)$$

where ω_s is the sampling frequency.

It is important to note that the sampler and its output are periodic, with a period of $j\omega_s$. This means that $y(s) = y(s + j\omega_s)$ and is represented in the s-plane by a series of periodic strips along the $j\omega$ -axis. As a result, the sampling operation generates a series of single line spectra in the

frequency domain, each occurring at an integer multiple of the sampling frequency.

If the sinusoidal input signal is again considered:-

$$x(t) = A \sin(\omega_a t)$$

then, for a sampling frequency of ω_s rads/sec., the sampler output spectrum will be periodic with spurious sidebands at all multiples of ω_s . The input spectrum is centred around each of the spurious sidebands (Fig. 6.5).

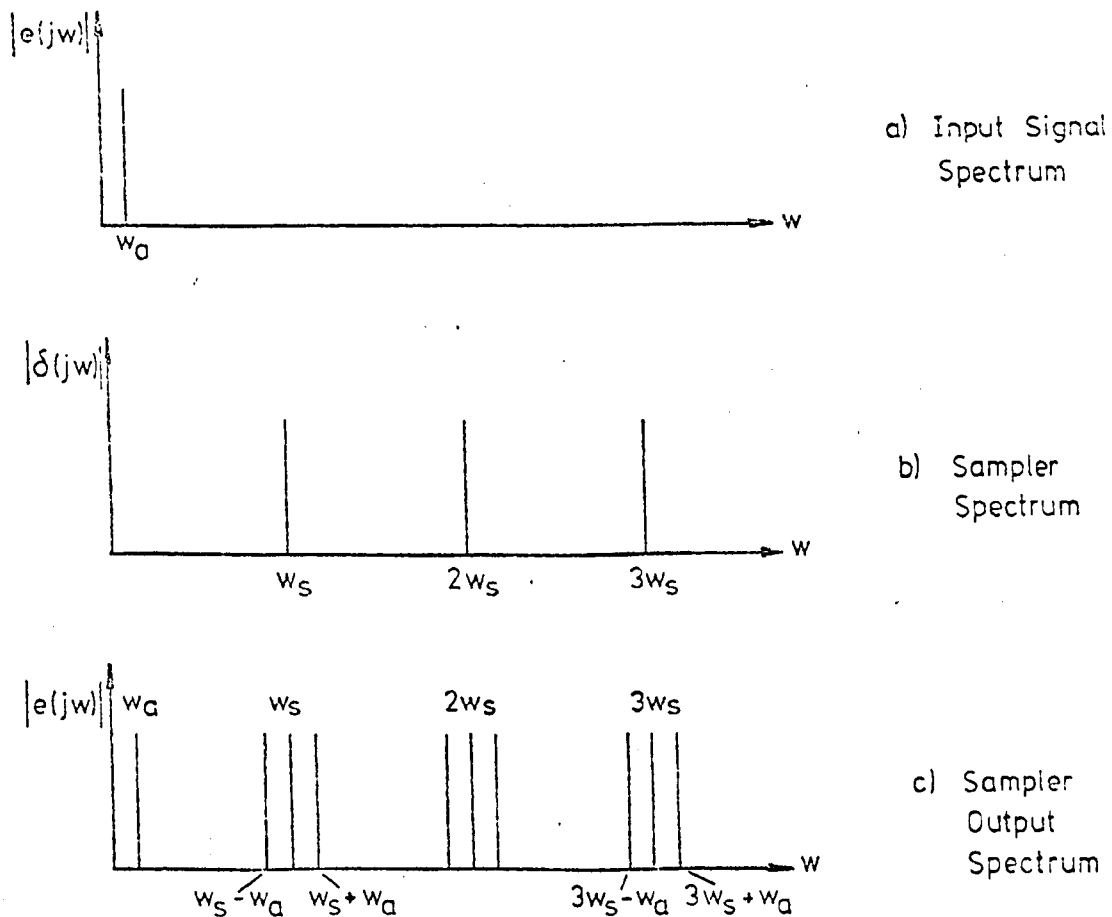


Figure 6.5 Frequency domain sampling.

6.2.2 The z-transform

Although the Laplace transform is very useful for the description of time continuous systems there are more convenient and meaningful transforms available for the description of sampled data systems; one such transform is the z-transform. The z-transform describes a signal at the

sampling instants and so only contains information about the corresponding time function at those instants. The z-transform may be obtained, quite simply, by the substitution:-

$$z = e^{sT}$$

or $s = \frac{1}{T} \ln(z)$

where z represents a complex transform variable. Thus, every continuous signal that has a Laplace transform also has a z-transform.

The above substitution causes the periodic strip from $-\omega_s/2$ to $+\omega_s/2$ on the $j\omega$ -axis of the s-plane to map into a unit circle in the z-plane, where ω_s is the sampling frequency. The portion of the strip lying in the left-hand half of the s-plane is mapped inside this unit circle, and that in the right-hand half is mapped on the outside (Fig. 6.6). Successive $\omega_s = 2\pi/T$ strips on the left-hand half of the s-plane will all map into the same unit circle and likewise all corresponding right-hand half strips will be mapped outside.

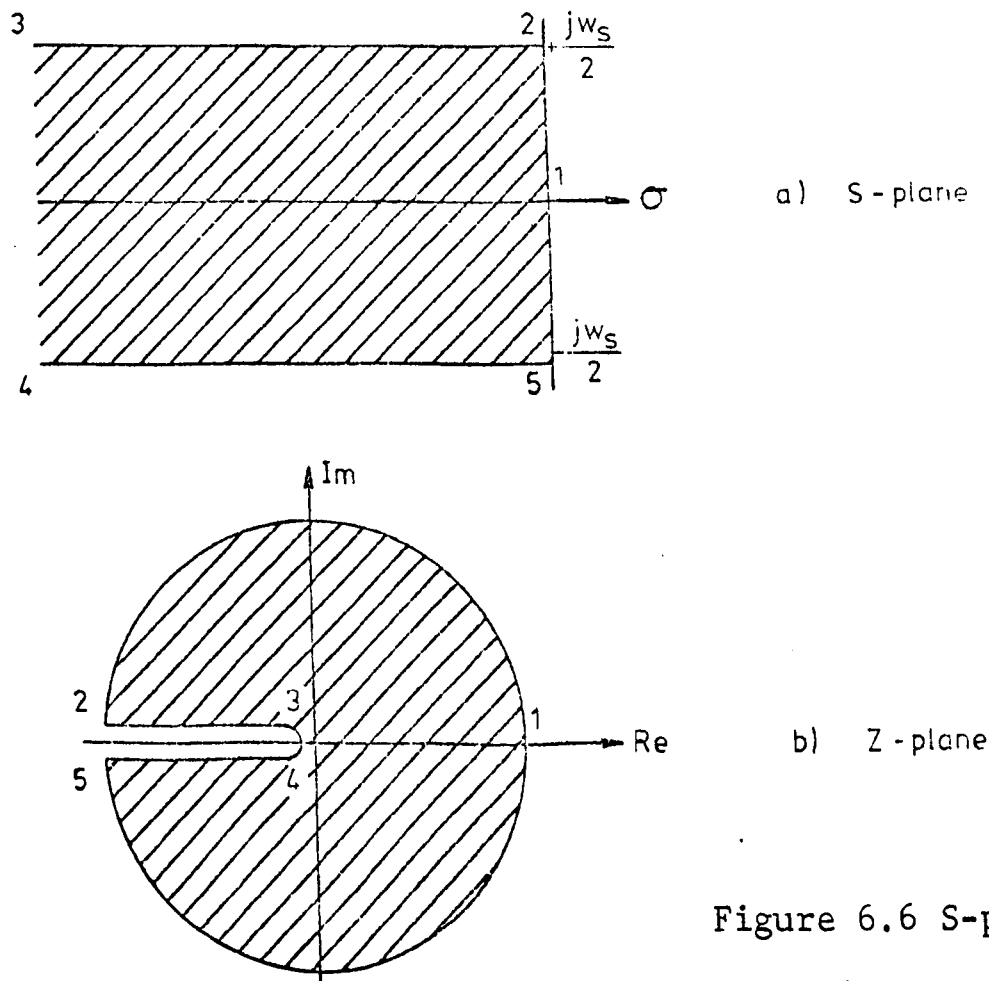


Figure 6.6 S-plane to z-plane mapping.

If a transfer function is to be stable, its poles must lie in the left-hand half of the s-plane and so, correspondingly, must also lie within the unit circle of the z-plane.

If $z = e^{sT}$ is substituted in equation (i), the z-transform output for a sampled input signal is found to be:-

$$y(z) = \sum_{n=0}^{\infty} x(nT).z^{-n}$$

where z^{-n} represents a delay operator and n a number of integral unit delays (c.f. equ. (iii)).

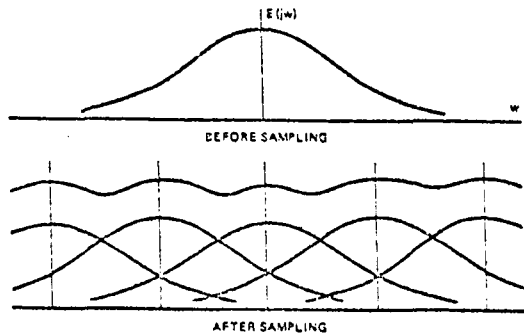
Clearly, this is a very convenient form of representation for use with a digital system as unit delays are very easy to realize in the form of buffers or shift-registers. The real power of the z-transform method, however, lies in the fact that it enables the polynomials of the discrete sampled system to be manipulated in a simple algebraic manner, just as the Laplace transform does for polynomials in the continuous system.

Because a digital processor must always perform a sampling operation on any time continuous input signal and must therefore always work with sampled data, the significance of the z-transform to the digital processing world is immediately apparent. A great deal of use is made of the z-transform in the next chapter to describe the digital processing techniques involved in the real-time spike detection system.

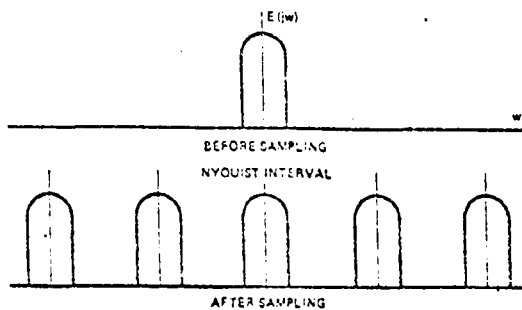
6.2.3 Aliasing

One final problem to be considered, which arises as a consequence of the sampling process, is that of aliasing. As discussed for the frequency domain (sect. 6.2.1.2), a sampled output will contain a series of spurious sidebands at multiples of ω_s , the sampling frequency (see Fig. 6.5). Now, the interval between $-\omega_s/2$ to $+\omega_s/2$, known as the Nyquist interval, places a bound on the bandwidth of the

sampler input signal. If the input signal is not bandlimited to below $\omega_s/2$ rads/sec. then the spurious components in the sampled output will overlap thus making it impossible to recover the original signal (Fig. 6.7.a). If, however, the input is bandlimited, it is possible to reject these spurious multiples and thereby recover the bandlimited input signal from the sampled output (Fig. 6.7.b).



(a) Input Signal Not Bandwidth Limited



(b) Input Signal Bandwidth Limited

Figure 6.7 Aliasing effects for non-limited and bandlimited input signals.

6.2.4 Quantization

Obviously, a discrete representation of a time continuous signal will involve quantization errors, the magnitudes of which are of great importance as the whole system will be effected either directly or indirectly by the inaccuracies present at the input. Indeed, this is true for any system, whether analogue or digital.

The error between the true input and its quantized representation will be evenly distributed over the range $\pm q/2$ where q represents the difference between adjacent quantization levels (i.e. $2^n - 2^{n-1}$).

The A-D conversion process is essentially non-linear [21] and, as a result of difficulties encountered in non-linear analysis, it is usual to treat the A-D problem as a noise problem. Consider the simple system shown in Figure 6.8.a in which a sequence $e(nt)$ is added to the input, to represent the noise introduced by the A-D conversion process.

The error of $\pm q/2$ is evenly distributed about zero (Fig. 6.8.b). To estimate the value of this error, consider the following:-

the probability density function $P(x)$ is defined by:-

$$\begin{aligned} P(x) &= 1 && \text{for } -0.5 \geq x \geq +0.5 \\ &= 0 && \text{otherwise.} \end{aligned}$$

The variance is then:-

$$\sigma_x^2 = q^2 \int_{-\infty}^{\infty} (x - \bar{x})^2 \cdot P(x) \cdot dx$$

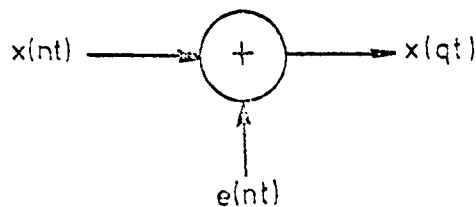
and since the mean value ($\bar{x}$) is zero:-

$$= q^2 \int_{-0.5}^{0.5} x^2 \cdot dx$$

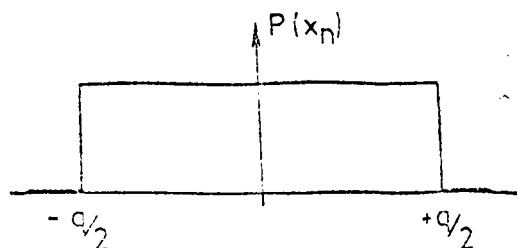
$$= q^2 \left[\frac{x^3}{3} \right]_{-0.5}^{0.5}$$

$$= \frac{q^2}{12}$$

which gives a standard deviation of $\sigma_x = 1/\sqrt{12} = 0.29$ for a single quantization level.



a) A-D conversion noise as an additive noise source



b) Probability Density Function of error in output

Figure 6.8 Representation of the A-D conversion error.

For an eight-bit A-D converter, the signal is quantized to 256 levels which gives a signal:noise ratio at the output of:-

$$\frac{256}{0.29} \approx 1000 \quad \text{or} \quad 60\text{dB}$$

and an accuracy of 0.39% (or 4 parts in 10^3).

For ten-bits, these figures become:-

$$S/N = \frac{1024}{0.29} \approx 3500 \quad \text{or} \quad 70\text{dB}$$

and an accuracy of 0.098% (or 1 part in 10^3).

The problems encountered with the D-A process are far less serious than for the A-D case, just considered. The D-A process is essentially linear and since it occurs at the output any errors will not affect the rest of the system. Inaccuracies due to truncation or rounding of results will be introduced if the value to be output requires 'scaling' (i.e. multiplication or division), but there will be no quantization errors since the value to be output is already in a quantized form.

There is, however, one problem associated with the D-A conversion process and that is the question of 'hold'; i.e. the interpolation necessary to restore or produce a continuous signal from a discrete one. The simplest form is a zero-order hold, in which the output is maintained at the previous output setting until the next sample arrives. This can be represented by a mixture of discrete and continuous operators [21] :-

$$(1 - z^{-1}) \cdot \frac{1}{s}$$

where s is the normal Laplace operator and z^{-1} the discrete operator meaning delay by one sample. Note that although $z = e^{sT}$, the above expression does not have precisely the same implication as $(1 - e^{-sT}) \cdot 1/s$ which is a continuous operation representing the transform of the block function $b(t) = U(t) - U(t - T)$ (see Chap. 5). Both parts of the above expression

are, however, linear.

6.3 Effect of errors on proposed signal-processing operations

Of the processes to be implemented for the spike detection system, digital filtering will impose the greatest computational demands and so will be given special consideration. Some consideration must, however, also be given to the proposed differentiation operation and its associated problems. Differentiation in the t-domain is equivalent to a multiplication by s in the frequency domain:-

$$\frac{d}{dt} f(t) \leftrightarrow sF(s)$$

$$\text{i.e.} \quad \frac{d}{dt} \leftrightarrow s$$

If $j\omega$ is substituted for s , a plot of the magnitude of the operation against frequency may be obtained (Fig. 6.9), from which the amplification of high-frequencies, caused by the differentiation process, becomes obvious (see Chap. 5).

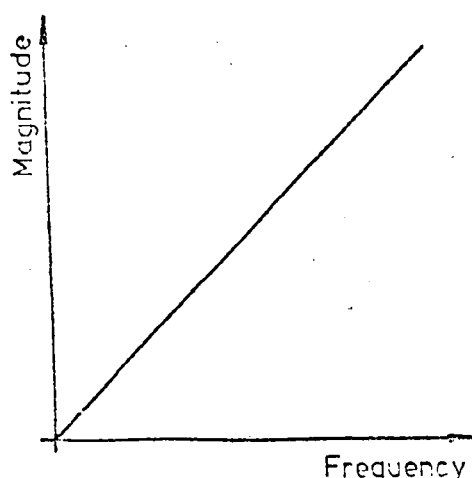


Figure 6.9 Magnitude of the differentiation operation as a function of frequency.

The high-frequency amplification can be a very serious problem for discrete systems since any quantization errors may be considered as an addition of a high-frequency noise component to the signal. Obviously, the differentiation operation will tend to amplify the errors introduced as a

result of quantization. This problem may be alleviated, though, by the inclusion of a low-pass filter to tailor the differentiated output. A far more serious problem arises when an attempt is made to realize the differentiation operation in a discrete system.

As stated earlier, d/dt becomes s in the complex frequency domain and for the z -domain, inverting the defining relationship $z = e^{sT}$, becomes:-

$$\frac{d}{dt} \leftrightarrow s \leftrightarrow \frac{1}{T} \ln(z)$$

Unfortunately, the logarithm of a complex variable is infinitely valued [48] ; a direct manifestation of aliasing:-

$$\ln(z) = \sigma + j\omega T + j2\pi k$$

where $k = 0, 1, 2, 3, \dots$. Even if only the principal value is taken, i.e. k is set to zero thereby forbidding aliasing, there is still another problem in that only real powers of z may be used for the realization of the process (i.e. only delays or advances by multiples of T in the time domain). Thus, the best approximation which may be obtained is a series, which to be practical, must be truncated.

The simplest form of series, suitable for this application is the Taylor series, which may be expressed as:-

$$f(z) = \sum_{n=0}^{\infty} \frac{f^{(n)}(z_0)}{n!} \cdot (z - z_0)^n$$

in the region of a point z_0 [48] . Expanding $\ln(z)$ in the region of $z = 1 + j0$ gives:-

$$\ln(z) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \cdot (z - 1)^n$$

so one form for the differentiation operation is:-

$$\frac{d}{dt} \rightarrow \frac{1}{T} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} \cdot (z - 1)^n$$

the Newton-Gregory forward difference formula. A crude approximation to the differentiation process may be obtained by taking the first term of this series:-

$$\frac{d}{dt} \rightarrow \frac{1}{T} \cdot (z - 1)$$

or in terms of samples, x_i :-

$$\left. \frac{dx}{dt} \right|_i \approx \frac{1}{T} \cdot (x_{i+1} - x_i)$$

A second approximation would be:-

$$\left. \frac{dx}{dt} \right|_i \approx \frac{1}{T} \cdot \left(-\frac{1}{2}x_{i+2} + 2x_{i+1} - \frac{3}{2}x_i \right)$$

Note that the series has been derived for the region on the unit circle near $z = 1 + j0$, which corresponds to $\omega = 0$. As a result, the approximation will only be valid for low frequencies.

If the behaviour of the differentiation approximation formulae is considered in the frequency domain, then the accuracy of the various approximations to the ideal response, may be compared on a magnitude versus frequency plot (Fig. 6.10) [21] .

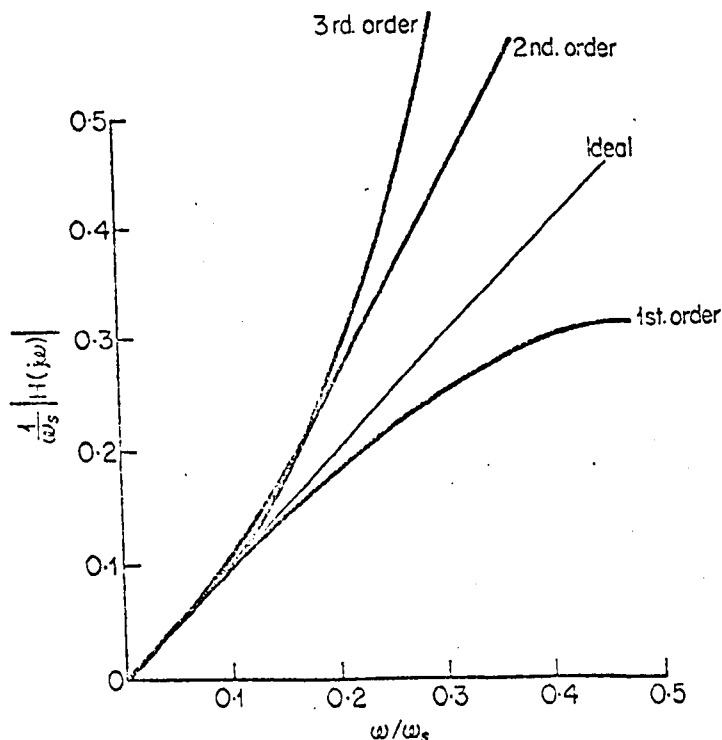


Figure 6.10
Frequency domain
behaviour of the
Newton-Gregory
forward difference
formula for
differentiation.
Note the restriction
of validity to low
frequencies.

Considering the first approximation:-

$$H(z) = \frac{1}{T} (z - 1)$$

this may be written as:-

$$H(\omega) = \frac{1}{T} (e^{j\omega T} - 1)$$

since $z = e^{j\omega T}$. The above equation may be rewritten as:-

$$\begin{aligned} H(\omega) &= \frac{1}{T} e^{j\frac{\omega T}{2}} (e^{j\frac{\omega T}{2}} - e^{-j\frac{\omega T}{2}}) \\ &= \frac{1}{T} e^{j\frac{\omega T}{2}} 2j \sin(\omega T/2) \end{aligned}$$

and substituting $T = 2\pi/\omega_s$ gives:-

$$\begin{aligned} H(\omega) &= \frac{\omega_s}{2\pi} e^{j\pi\frac{\omega}{\omega_s}} 2j \sin(\pi\omega/\omega_s) \\ &= \underbrace{e^{j\pi\frac{\omega}{\omega_s}}}_{\text{phase}} \underbrace{j \frac{\omega_s}{\pi} \sin(\pi\omega/\omega_s)}_{\text{amplitude}} \end{aligned}$$

From the above, it can be seen that for low frequencies (i.e. $\omega \rightarrow 0$) then:-

$$\sin(\pi\omega/\omega_s) \rightarrow \pi\omega/\omega_s$$

and since the phase term has unit modulus, the amplitude of the function becomes:-

$$\begin{aligned} \text{Ampl.} &\simeq j \frac{\omega_s}{\pi} \cdot \pi \frac{\omega}{\omega_s} \\ &= j\omega \end{aligned}$$

i.e. it approximates closely to the desired differentiation operation:-

$$\frac{d}{dt} \leftrightarrow s$$

As the frequency increases, so the approximation breaks down and the expression tends to a sine function, as can be seen in Figure 6.10.

The response of the higher order approximations may be calculated in a similar manner, the first three being shown in Figure 6.10.

It has been proposed that a sampling rate of 500Hz is used (see Chap. 5) and, since the EEG records to be analysed are bandlimited to 70Hz, the ratio of ω to ω_s will be:-

$$\frac{\omega}{\omega_s} = \frac{70}{500} = 0.14$$

From an inspection of Figure 6.10, it can be seen that any of the approximations to the differentiation process should yield equally good results, with only a small deviation from the ideal. In fact, it would appear that the first approximation may offer the best solution up to this particular ω/ω_s ratio.

The theory of design and implementation of the various types of digital filter is well documented elsewhere (e.g. [4][5][49]) and will not be reproduced here. Instead, the constraints imposed by the operating speed and wordlength restrictions, already considered in general terms, will be investigated.

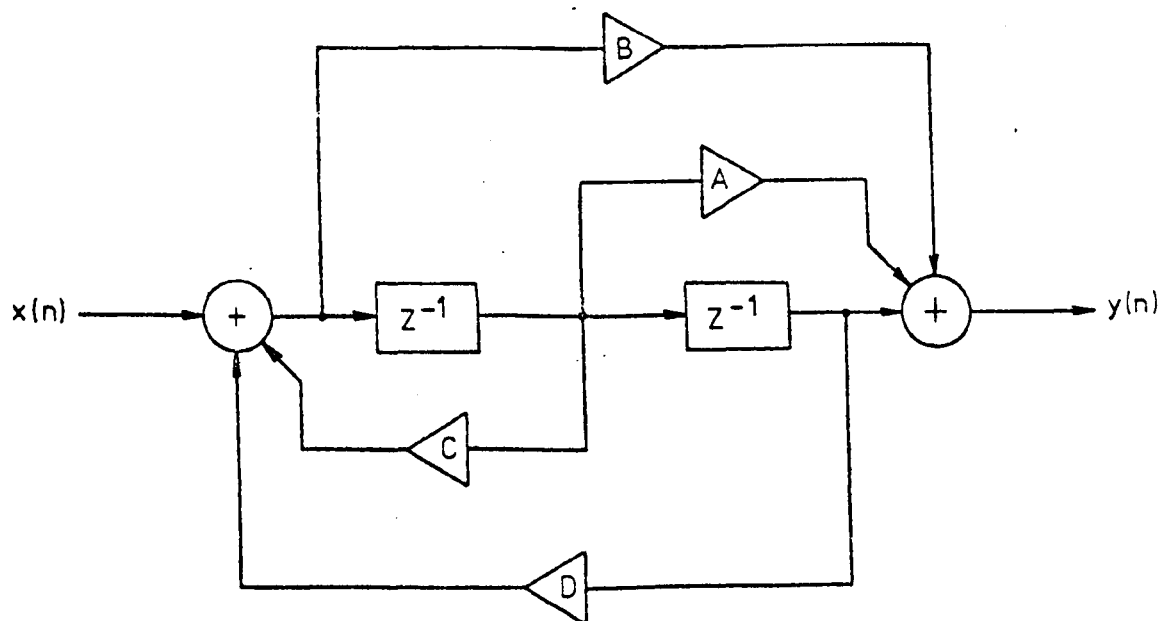
The maximum operating speed of the CPU places an upper limit on the sampling rate (ω_s) available for the filter. This in turn places restrictions on the upper value for the cut-off frequency (ω_c). In order to maintain a usable bandwidth the filtering operation must be kept as simple as possible, for any given application. As a result, it is proposed that fixed-point fractional arithmetic be used wherever possible and that truncation instead of rounding be used to maintain the dynamic range of intermediate results. Further, every effort should be made to contain the mathematical calculations within the range of single length working.

6.3.1 Register length

The limited register length available with microprocessors

is the fundamental factor governing the type and complexity of digital filter which may be supported. Errors will arise as a consequence of the truncation of results and the quantization of the input and coefficient values. The contribution of all of the errors emanating from different points in the filter can be calculated, but an estimation of their combined effect on the output can be complicated, particularly for recursive filters (see sect. 6.4).

The major limitation, however, is in the accommodation of the inherent gain experienced by the data as it passes through the filter. In general, as the cut-off frequency is decreased for a constant sampling frequency, i.e. the ratio ω_c/ω_s is decreased, so the peak gain through the filter increases. Hence, as the cut-off frequency is taken closer to zero so the extra number of bits required to hold intermediate results will increase. A typical example is given in Figures 6.11 and 6.12 [47] from which it can be seen that peak gains of two-hundred or more may be required (i.e. eight-bits more than the input value).



$$H(z) = G \left[\frac{1 + Az^{-1} + Bz^{-2}}{1 - Cz^{-1} + Dz^{-2}} \right]$$

$$M_A = \frac{1}{\sqrt{\left(1 - \frac{C^2}{4D}\right)(1 - D)^2}}$$

Figure 6.11 Typical Canonical two-pole digital filter.

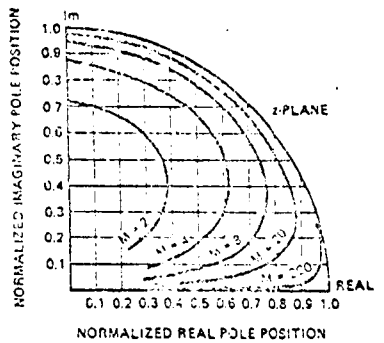


Figure 6.12 Peak gain of the filter given in Figure 6.11, for various pole positions in the z-plane.

Clearly, the problem of gain through the filter will impose severe restrictions on the type and complexity of filter to be realized and on the choice of values for ω_s and ω_c . The effects of this internal magnification can be reduced by the use of different configurations for the realization of a given filter [25][47].

6.3.2 Limit-cycle behaviour

The peak internal gain will set a limit on the number of bits available to represent the input sample, to ensure that overflow does not occur. However, as the number of bits used for the input is decreased so the effects of truncation or rounding of results will increase. This leads to a group of effects known as limit-cycle behaviour, in which periodic sequences appear in the output as a result of the successive approximations. The phenomenon is non-linear and is dependent upon the filter configuration, the type of arithmetic, the method of producing finite results and the actual filter coefficients.

The most serious limit cycles occur as a result of overflow in filters implemented using one's or two's complement arithmetic.

Another type of limit-cycle behaviour, first investigated by Blackman [50], is the appearance of 'deadbands' in which the output stops changing even though it has not reached its theoretically correct value. As an example, consider a first-

order filter represented by the equation:-

$$y(nT) = Gy(nT - T) + (1 - G)x(nT)$$

If the output $y(nT) = 10$ and the input $x(nT)$ is suddenly reduced to zero, then for $G = 0.6$:-

n	y(nT)	$\hat{y}(nT)$	$\tilde{y}(nT)$
0	10.00	10	10
1	6.00	6	6
2	3.60	4	3
3	2.16	2	2
4	1.29	1	1
5	0.78	1	0
6	0.47	1	0
7	0.28	1	0

Table 6.2 Example of the 'deadband' effect.

The $\hat{y}(nT)$ column represents rounded values and the $\tilde{y}(nT)$ column, truncated values. As can be seen, for $n \geq 4$ the rounded value has stopped changing, even though it has not reached the correct value. If the output had been increasing instead of decreasing, then a similar effect would have been observed for the truncated result.

Clearly, if an extra digit had been preserved the deadband would not be so pronounced. The deadband will now exist for $n \geq 6$ and the magnitude of the deadband output is reduced to 0.5.

n	y(nT)	$\hat{y}(nT)$	$\tilde{y}(nT)$
.	.	.	.
.	.	.	.
4	1.29	1.5	1.0
5	0.78	1.0	0.5
6	0.47	0.5	0.0
7	0.28	0.5	0.0

Table 6.3 Effect of increased wordlength on deadband behaviour.

The significance of this result is that the register length should be sufficient to ensure that the deadband is a negligible fraction of the total wordlength. This will help to minimize the possibility of its occurrence and also reduce the error to a small value.

The effects of zero input limit-cycle behaviour, just considered, can be effectively removed, as far as the output is concerned, by the use of a longer wordlength to represent the results of calculations than is required for the output. If, for example, the value of $y(nT)$ in the previous example is calculated to ten-bits and only eight-bits are required for the output, by the truncation of the two least significant digits the limit-cycle effects would be removed.

6.3.3 Parameter Quantization

Quantization of the coefficients in a digital filter leads to slight changes in their value. As a result, the poles and zeros are displaced slightly from their correct locations thereby altering the actual filter. The effects of this error can be reasonably well predicted, since they occur only once. Clearly, the limited number of bits available to represent the coefficients places a restriction on the values which may be assigned to the coefficients.

The best that can be achieved with a sixteen-bit wordlength is an accuracy of five decimal places for any given coefficient value. From a study of various texts concerned with digital filter implementation (e.g. [4][5][49]), it is apparent that a minimum of four decimal places (i.e. fourteen-bits) will be required for any practical purposes; many filters will demand a much greater accuracy. No problems should be encountered in this range for simple filters, provided the effects of quantization (and inherent truncation or rounding) is to reduce the magnitude of the coefficients, since this will cause the poles to be moved towards the centre

of the unit circle in the z-domain, i.e. in a direction which will tend to improve the stability of the filter, but at the expense of a reduced Q-factor.

Some coefficients have a negative sign associated with them and if the coefficient is to be represented as a negative number, then care is needed in the choice of representation. As was shown earlier, the one's complement and sign-magnitude representations can lead to an increase in magnitude of the value which may be sufficient to cause the pole to stray over the stability boundary (i.e. move outside the unit circle).

The safest method, and the one to be used to realize the filter for this application, is to define all of the coefficients as positive quantities and to deal with the signs during the final summation process.

As an example of the calculation of the errors involved due to coefficient quantization, consider the second-order system specified by:-

$$H(z) = \frac{1}{1 - Az^{-1} + Bz^{-2}}$$

The poles are located at $z = r.e^{\pm j\theta}$

$$\text{where } r = \sqrt{B}$$

$$\text{and } \theta = \cos^{-1} \frac{A}{2\sqrt{B}}$$

To determine how the poles vary with changes in A and B, the total differentials must be taken:-

$$\Delta r = \frac{\partial r}{\partial A} \Delta A + \frac{\partial r}{\partial B} \Delta B$$

$$= \frac{1}{2r} \Delta B$$

$$\Delta \theta = \frac{\partial \theta}{\partial A} \Delta A + \frac{\partial \theta}{\partial B} \Delta B$$

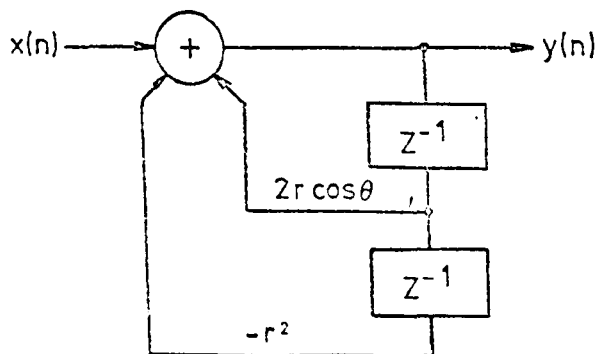
$$= - \frac{\Delta A}{2r \sin \theta} + \frac{\Delta B}{2r^2 \tan \theta}$$

Thus, the changes introduced by the inaccuracies due to quantization of the coefficient values may be calculated.

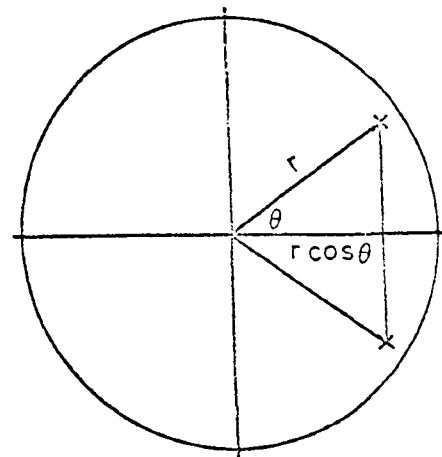
A graphical interpretation of the quantization effects can be given from a consideration of the z-plane. As an example, consider the second-order filter shown in Figure 6.13.a for which:-

$$A = 2r \cos \theta$$

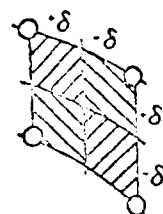
$$B = r^2$$



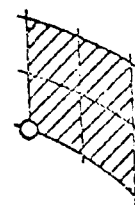
(a)



(b)



Rounding



Truncation

O - Realizable pole positions

$$\delta = \pi/2$$

(c)

Figure 6.13 Effects of quantization on pole locations for typical filter.

Variations in the value of A will map as a series of vertical lines and variations in B will map as a set of concentric circles, in the z -plane. As can be seen, in Figure 6.13.c, a given continuous region maps into a finite number of representable points. The shaded area represents a 'quantization region' and any pole to be placed in this region can only be specified to one of the points shown.

As for other parameters, the coefficients may be expressed to their nearest rounded or truncated quantization level. Note that for rounding, the quantization region is sub-divided into four smaller regions. The actual positioning of the pole will depend upon which of the four sub-regions the unquantized pole would fall. If a truncated representation of the pole coefficients is used, then the whole of the quantization region will map into one realizable position (Fig. 6.13.c). Hence the use of a rounded value facilitates a more accurate placement of the pole in relation to its true position but for poles close to the unit circle there is a possibility that they may be moved outside the stability boundary. Truncation, although inherently less accurate, does not involve any such instability problems.

The area enveloped by the quantization region is governed by the number of quantization levels available to represent the coefficients; the coarser the quantization the larger the area.

For small values of θ , problems will be encountered as the mapping for the r terms will be tending to vertical lines. As can be seen in Figure 6.14.a, if the quantization process produces an increase in the value of A and a decrease in the value of B then it will not be possible to define the pole. Note, this problem will not arise if the values of A and B are truncated; truncation will always tend to reduce both values. For very small values of θ (Fig. 6.14.b), the realized value of θ may be reduced to zero!

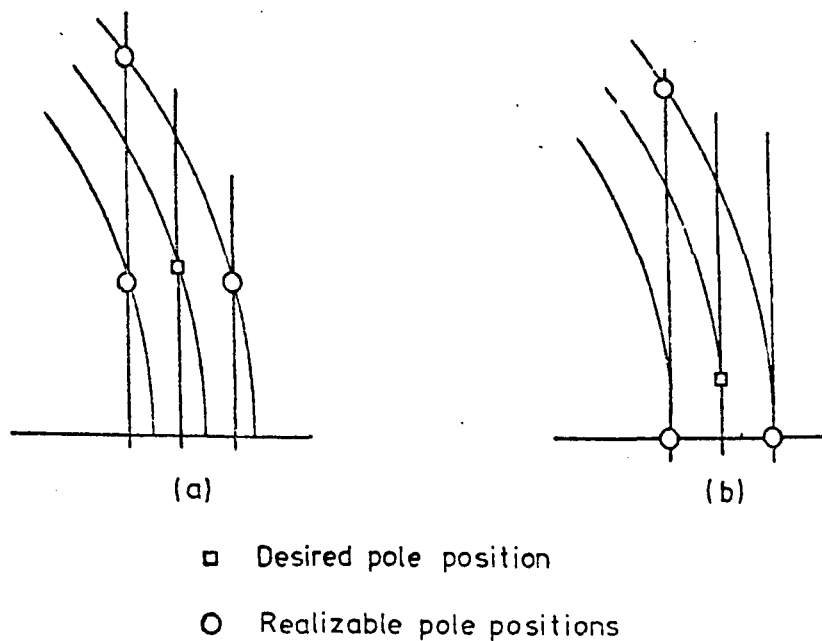
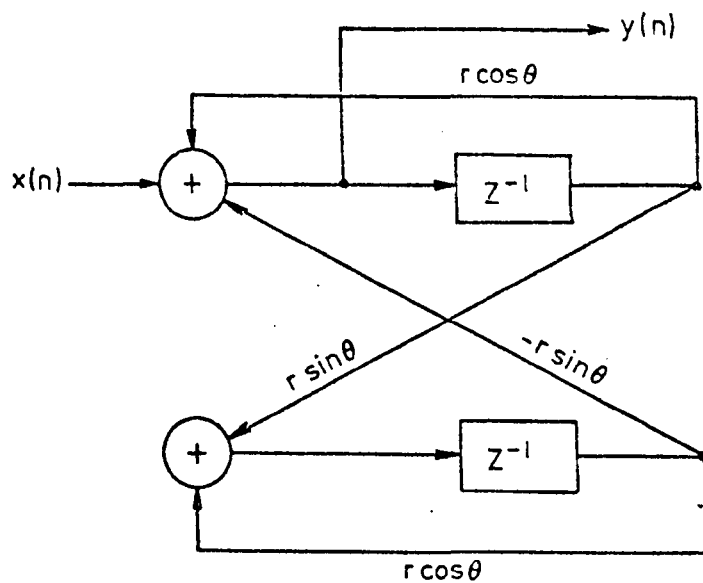


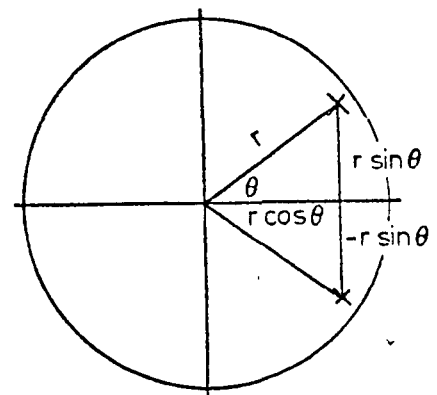
Figure 6.14 Problems encountered with small values of θ .

For cases, such as these, it will be necessary to adopt some alternative means of filter implementation. One such alternative is the coupled form proposed by Rader and Gold [51] (Fig. 6.15.a). In this, the coefficients are defined by $r\cos\theta$ and $r\sin\theta$ terms and consequently the quantization region is defined by a rectangular grid (Figs. 6.15.c). Care is again needed when rounded values are used to specify poles which are close to the unit circle.

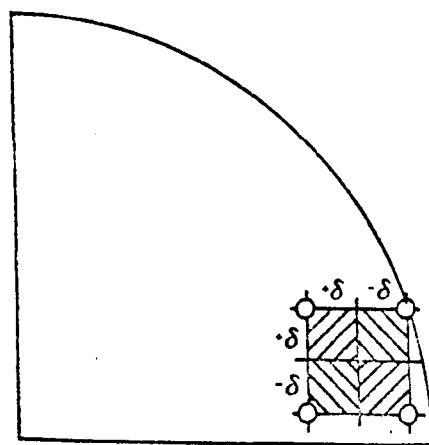
There are, in theory, many different realizations available for the implementation of a given second-order filter, but the two given are the most common. Different structures will, of course, give rise to different grids in the z -plane and hence different quantization regions. In general, a structure should be chosen for which the grid is dense in the region of the z -plane where the poles are to be located.



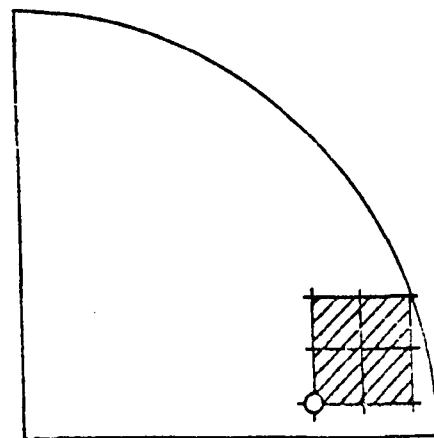
(a)



(b)



Rounding



Truncation

○ - Realizable pole positions

$$\delta = q/2$$

(c)

Figure 6.15 Quantization regions for coupled form of filter realization.

6.4 Recursion Noise

An assessment of the total effect of the various noise sources on the final output signal of a recursive filter is complicated. The noise processes can be modelled by the addition of noise sources at appropriate points throughout the filter, each representing the contribution of a particular calculation. As these noise sources are added at points inside the filter, they may not all pass through the same portions of the filter with the result that their contributions to the output signal noise may be quite different.

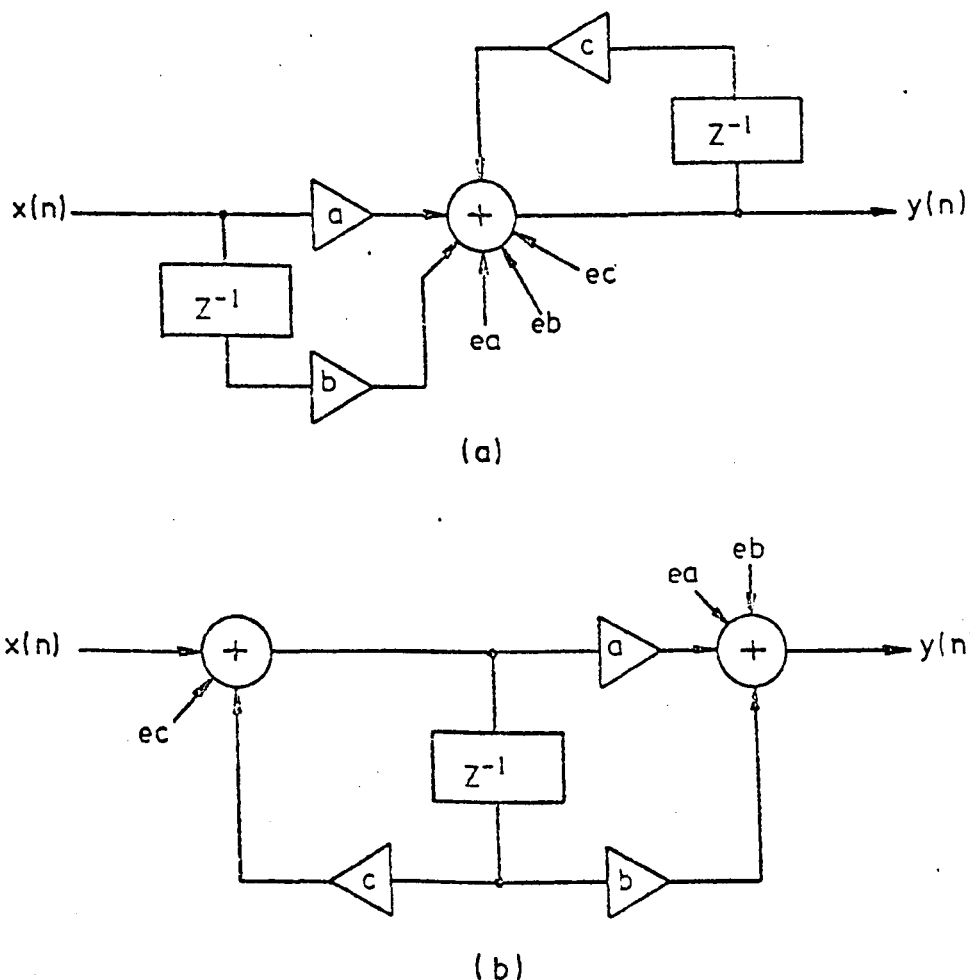


Figure 6.16 Internal noise sources in digital filters.

As an example, consider the effects of two different implementations for a second-order filter, as shown in Figures 6.16.a and b. For the implementation shown in Figure 6.16.a, the noise inputs for each stage are all

operated on by the (c) coefficient, whereas for Figure 6.16.b the noise due to the (c) coefficient is operated on by the entire filter but those due to (a) and (b) merely add to the output and are not operated on by any part of the filter.

As a result of this dependence of the overall noise output upon the filter configuration, any assessment must be deferred to a later chapter when the actual filter design has been decided upon.

Chapter 7

Experimental Method and Results

The prototype based on the CPT800 microprocessor has been selected as the vehicle on which to implement the spike detection scheme described in Chapter 5. In addition to the basic hardware unit, an eight-bit analogue-to-digital converter is required for input and an eight-bit, bipolar digital-to-analogue converter for the output (see Vol. II-6).

Four abnormal and three normal records were analysed, each of approximately three minutes duration. The features of each record and, where possible, the origin are summarized in Table 7.1. The records are stored in the form of four-track, 6.4mm ($\frac{1}{4}$ in.) FM tape recordings which were replayed at normal speed to simulate real-time operation. The recorded signals are in the range 0 - 2.0V and have been band-limited to 70Hz..

The fundamental criterion set out for the spike detection process was that it must be kept as simple as possible in order to maximize the bandwidth of the system. Further, it had to be capable of detecting all abnormal spikes in the records to be analysed, as indicated by a trained observer, with a minimal false alarm rate.

7.1 Differentiation

As discussed in Chapter 5, it is considered that a straightforward analysis of the original input data would suffer from the disadvantage that, in order to reduce the false alarm rate, the detection threshold (number of points on slope) would have to be raised, with the consequent problem that small amplitude spikes may be missed. A further problem is that, owing to the heavy dependence of the results of this straightforward approach upon the spike amplitude, the algorithm would almost certainly require modification when used with different EEG records.

One method of reducing this amplitude sensitivity is to apply a preferential amplification to the high-frequency

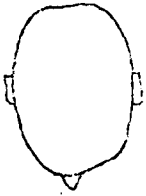
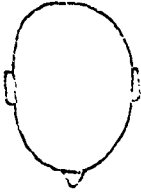
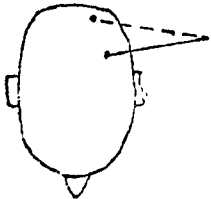
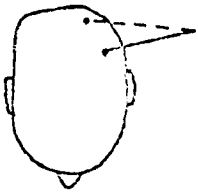
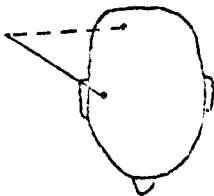
	Electrode Placement	Features
1		6 Abnormal Spikes
2		22 Abnormal Spikes
3		66 Abnormal Spikes
4		23 Abnormal Spikes
5		Normal (+Artifact)
6		Normal (+Artifact)
7		Normal

Table 7.1 Origin and features of records to be analysed.

(sharp) edges occurring in the record. An obvious method of realizing this operation is to differentiate the input signal. Since the differential represents the rate of change of input with respect to time, the sharp edges will be amplified whereas the slower activity will not.

The problems encountered when an attempt is made to realize the differentiation operation in a discrete system were discussed in Chapter 6, where two possible algorithms were established. One represents a simple first-order difference and is described by:-

$$\frac{dx}{dT} \approx \frac{1}{T} (x_n - x_{n-1})$$

$$\text{or} \quad \frac{dx}{dT} \approx \frac{1}{T} (1 - z^{-1})$$

and the other, a second-order approximation described by:-

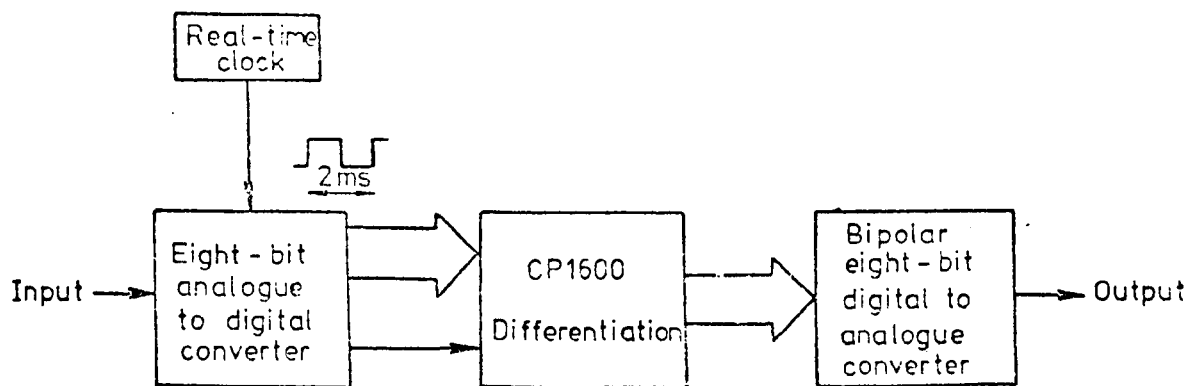
$$\frac{dx}{dT} \approx \frac{1}{T} \left(-\frac{1}{2}x_n + 2x_{n-1} - \frac{3}{2}x_{n-2} \right)$$

$$\text{or} \quad \frac{dx}{dT} \approx \frac{1}{T} \left(-\frac{1}{2} + 2z^{-1} - \frac{3}{2}z^{-2} \right)$$

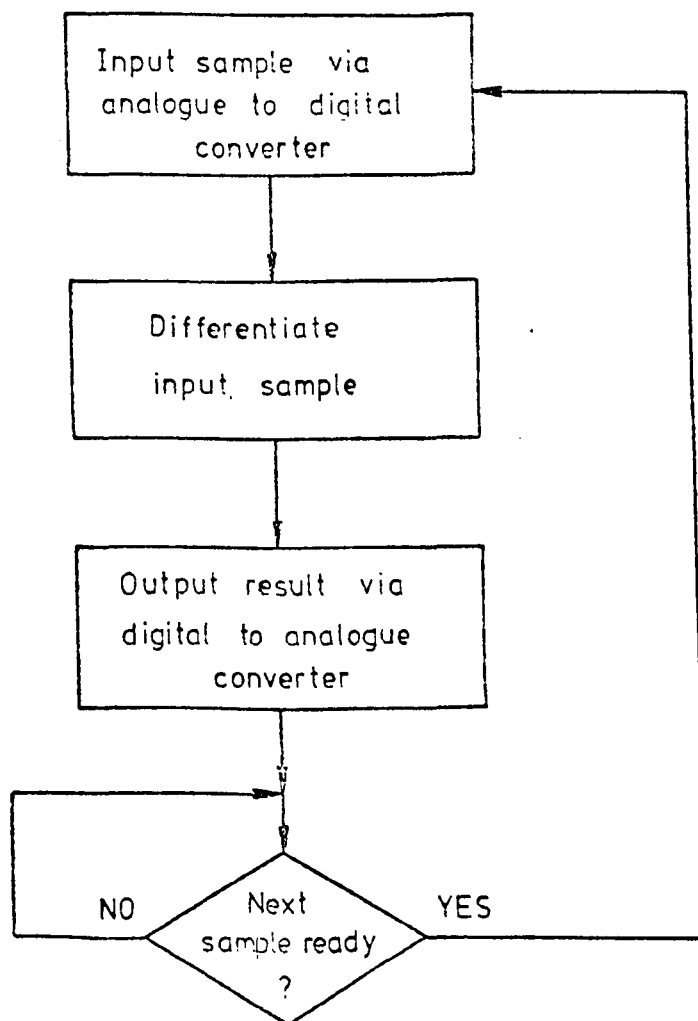
Both algorithms were implemented on the CP1600, the basic system being as shown in Figure 7.1. An external oscillator, set to 500Hz., was used as a real-time clock to set the sampling frequency.

For the purposes of testing, a signal generator was used to provide the input signal. The results for square and triangular-wave inputs are as shown in Figures 7.2 and 7.3. The outputs were captured and plotted via a Biomation 8100 transient recorder and an X-Y plotter.

As can be seen from the results, there is little difference between the outputs produced by either algorithm. Indeed, it could be argued that the first approximation is better in that it responds more quickly to sudden changes at the input, since only one past value is used in the calculation. The square-wave test demonstrates this point,



a) System configuration for differentiation process.



b) Sequence of operations for the differentiation process.

Figure 7.1 Differentiation operation.

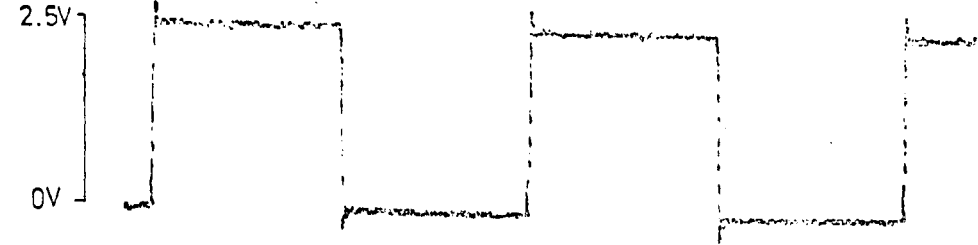
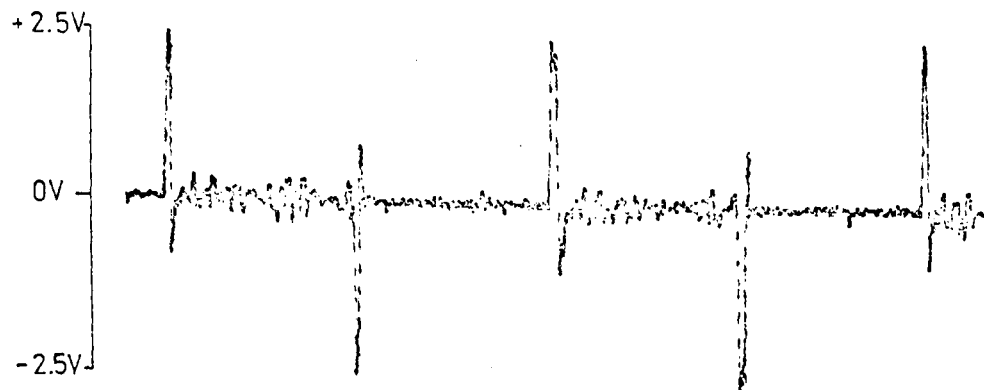


Figure 7.2
Square-wave
test input.



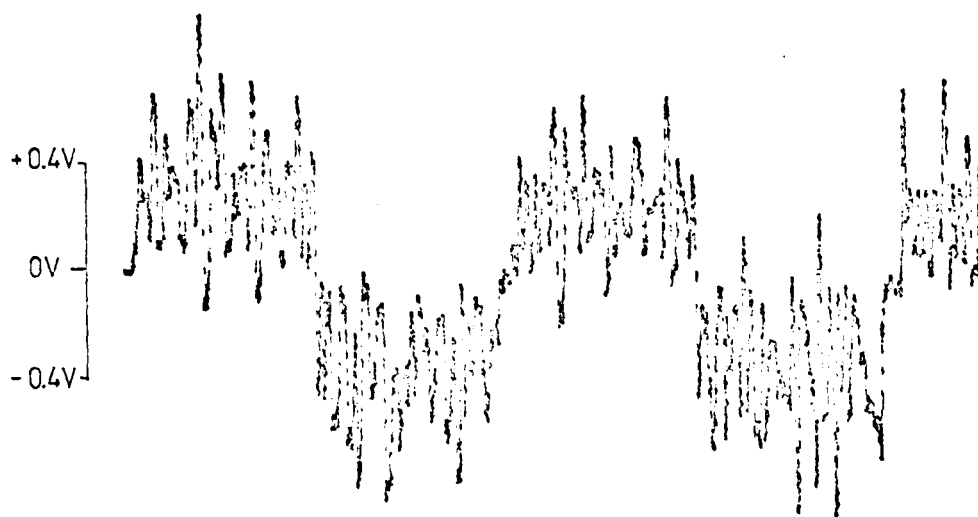
First
approximation



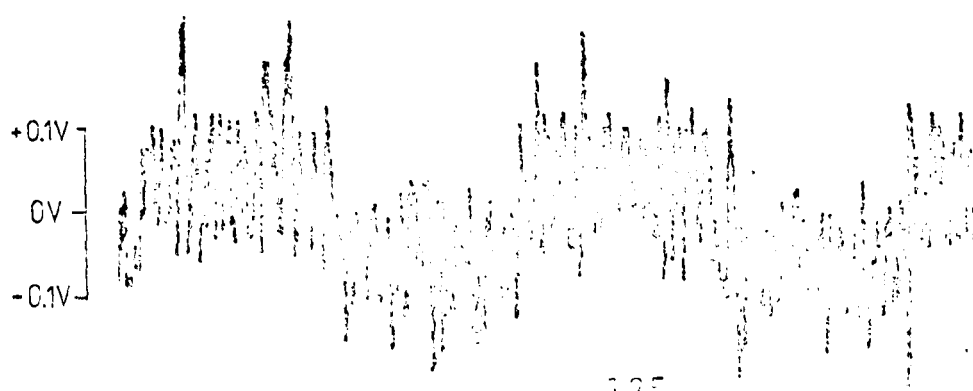
Second
approximation



Figure 7.3
Triangular-
wave test
input.



First
approximation



Second
approximation

where the extra terms are shown to introduce an overshoot into the output signal. The possibility that the first approximation may yield a better result was noted in Chapter 6, when the frequency domain behaviour of the differentiation process was considered.

The square-wave is not, of course, a valid test since the high-frequencies present in the edges disobey the sampling theorem; it serves more as an illustration. The triangular-wave is more representative of the features to be classified.

As there appears to be little difference in the output for either algorithm when tested with the triangular-wave, it was decided that the simpler method should be adopted, since it offers a superior real-time capability. It should be noted, however, that the first-order approximation represents only a differencing operation and is not a true differentiation. The frequency response of the first-order approximation is given in Chapter 6 (Fig. 6.10).

The output for a section of one EEG record, when operated on by the first-order differencing process, is shown in Figure 7.4. As predicted in Chapter 5, all of the

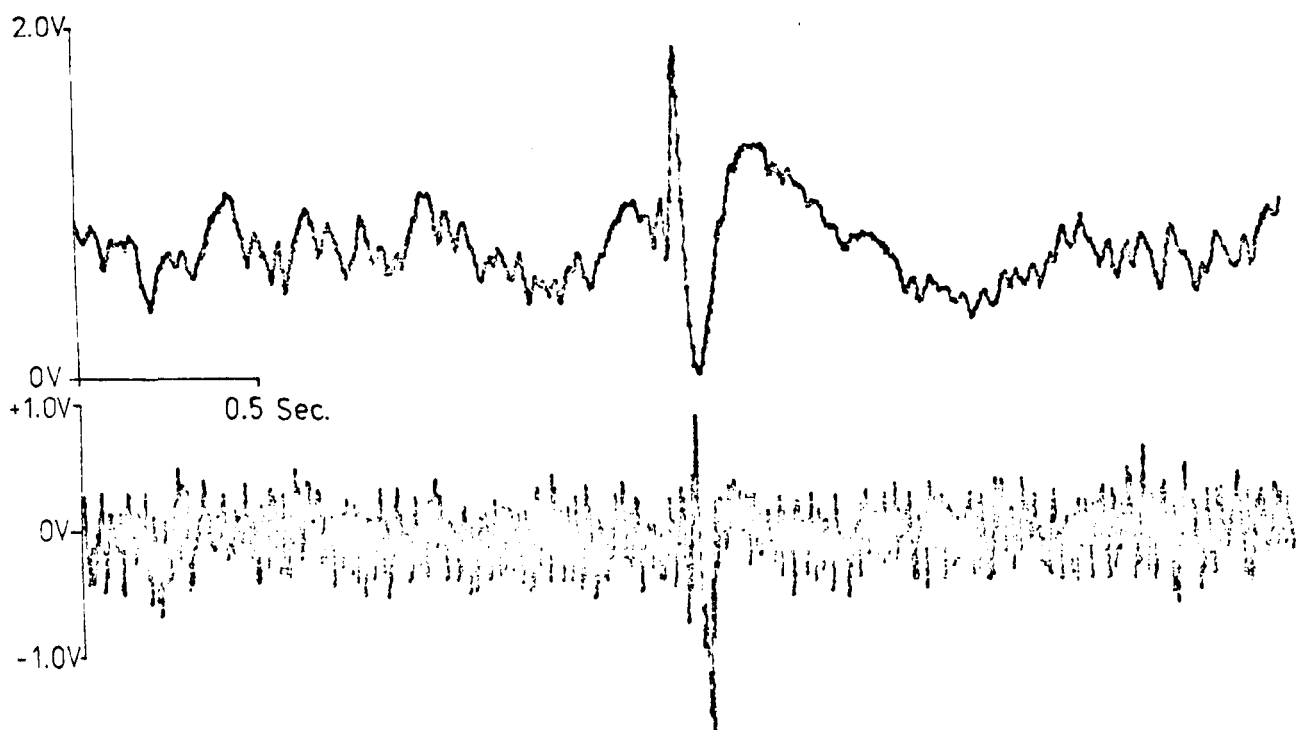


Figure 7.4 Effect of first-order differencing operation on a section of an EEG record.

high-frequencies have been amplified which results in a very 'noisy' output. The actual output was somewhat worse than that shown in Figure 7.4; the mechanical compliance of the plotter introduced a smoothing effect on the signal, analogous to a single-pole RC filter.

Clearly, the differentiated output must be filtered before any attempt can be made to identify the spike activity from the background noise, as proposed in Chapter 5.

7.2 Choice of Filter

The decision as to which type of filter to use is a complicated one. All offer an approximation to the ideal rectangular amplitude versus frequency response, by trading one characteristic against another. The main features of concern are:-

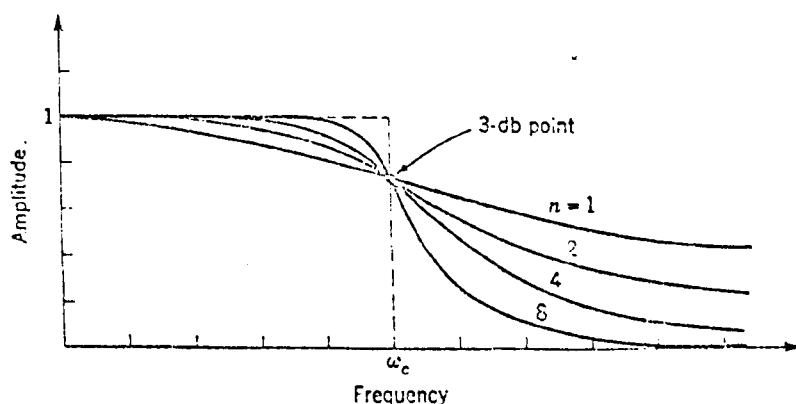
- i) the sampling frequency (ω_s)
- ii) the cut-off frequency (ω_c)
- iii) rate of attenuation beyond cut-off
- iv) phase response
- v) pass-band ripple
- iv) stop-band ripple.

The filter for this application is to be of a low-pass design with a sampling frequency of 500Hz..

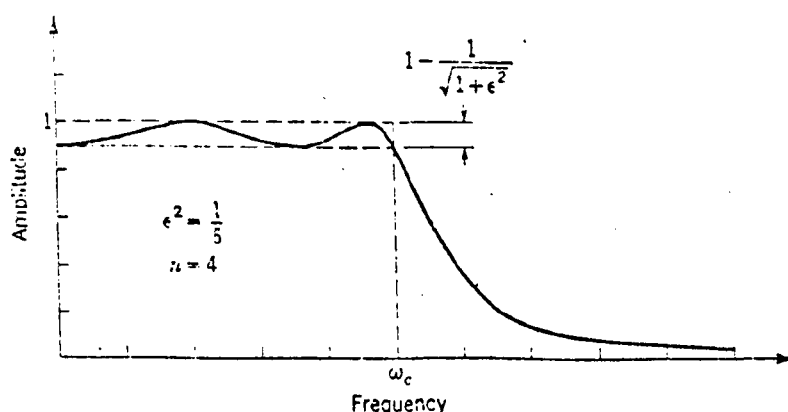
The simple running mean type of low-pass filter was rejected in Chapter 5 on grounds of its poor performance. It was decided that a proper digital filter configuration should be used for any required filtering operations.

Three filter types were considered, Butterworth, Chebychev and elliptic, and their characteristics compared. The frequency responses for each type of filter are given in Figure 7.5. Of the three, the Butterworth filter offers the best phase response and the elliptic the worst, although none of them has a linear characteristic. As can be seen

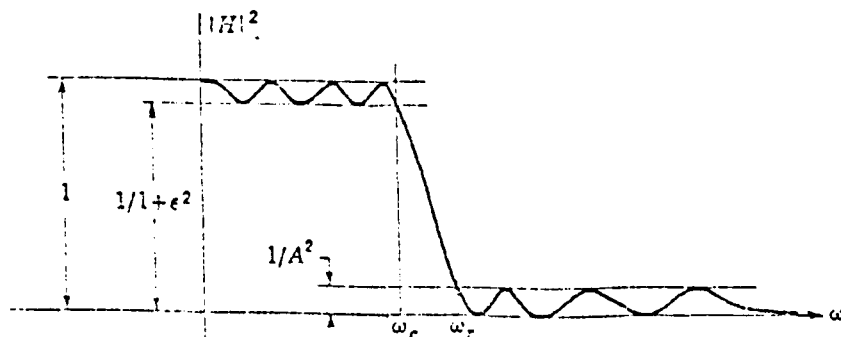
from Figure 7.5, the Butterworth filter gives the flattest pass and stop-band response. The Chebyshev filter, by allowing ripple in the pass-band, achieves a sharper cut-off for the same number of poles and zeros. The elliptic filter achieves an even sharper cut-off than the Chebyshev for the same filter complexity but at the expense of ripple in both the pass and stop-bands. The elliptic filter also exhibits a shorter settling time than either the Butterworth or Chebyshev filters, which may take several periods to settle at the cut-off frequency [49].



a) Butterworth frequency responses.



b) Chebyshev frequency response.



c) Elliptic-filter frequency response.

Figure 7.5
Amplitude versus
frequency
characteristics
for various low-
pass filters.

It was considered, for this application, that the amplitude response of the filter was of more importance than the rate of attenuation beyond the cut-off frequency. The features to be extracted are in the range 12 - 70Hz. (the input signal being band-limited to 70Hz.), whereas the noise arises as a result of errors introduced into the values of successive samples and so is of a much higher frequency. Hence, the difference between the frequencies to be passed and those to be stopped is fairly large and so a slow rate of attenuation beyond cut-off is acceptable. Furthermore, since the proposed detection system is to be based on a level-detection scheme (see Chap. 5), a flat amplitude response in the pass-band is of great importance.

As a result, it was decided that a second-order Butterworth filter would be implemented and its performance assessed when used with the output from the differentiation process. If this filter had proved unsatisfactory, then it would have been necessary to consider either a higher-order configuration or a different type of filter.

It was considered that a first-order filter would be of little use since, in analogue terms, this would be equivalent to cascading a differentiator with an integrator. All that could be reasonably expected from such an arrangement is a distorted version of the input signal (due to phase shifts). Nevertheless, a single-pole filter was implemented and the system tested. The difference between the time constants for the two operations were such that the performance, when tested with a triangular-wave input, was much better than anticipated, the output being as shown in Figure 7.6.

Despite the success of this simple filter with the low-frequency triangular input, it was not considered to be satisfactory for the present application owing to the relatively poor performance of single-pole filters in dealing with higher frequency signals, which will be present in the

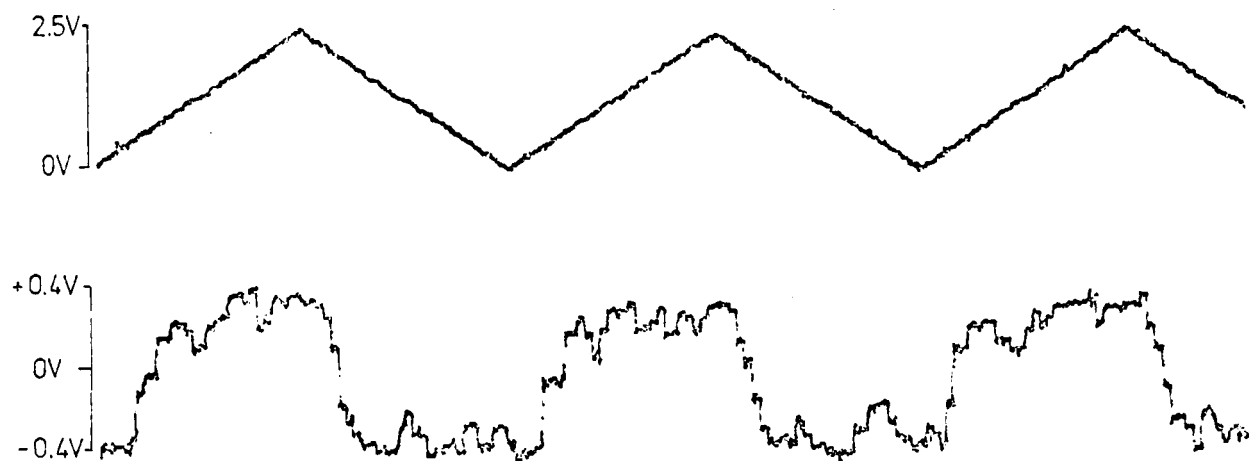


Figure 7.6 Response of single-pole, low-pass filter for a 10Hz. triangular-wave input.

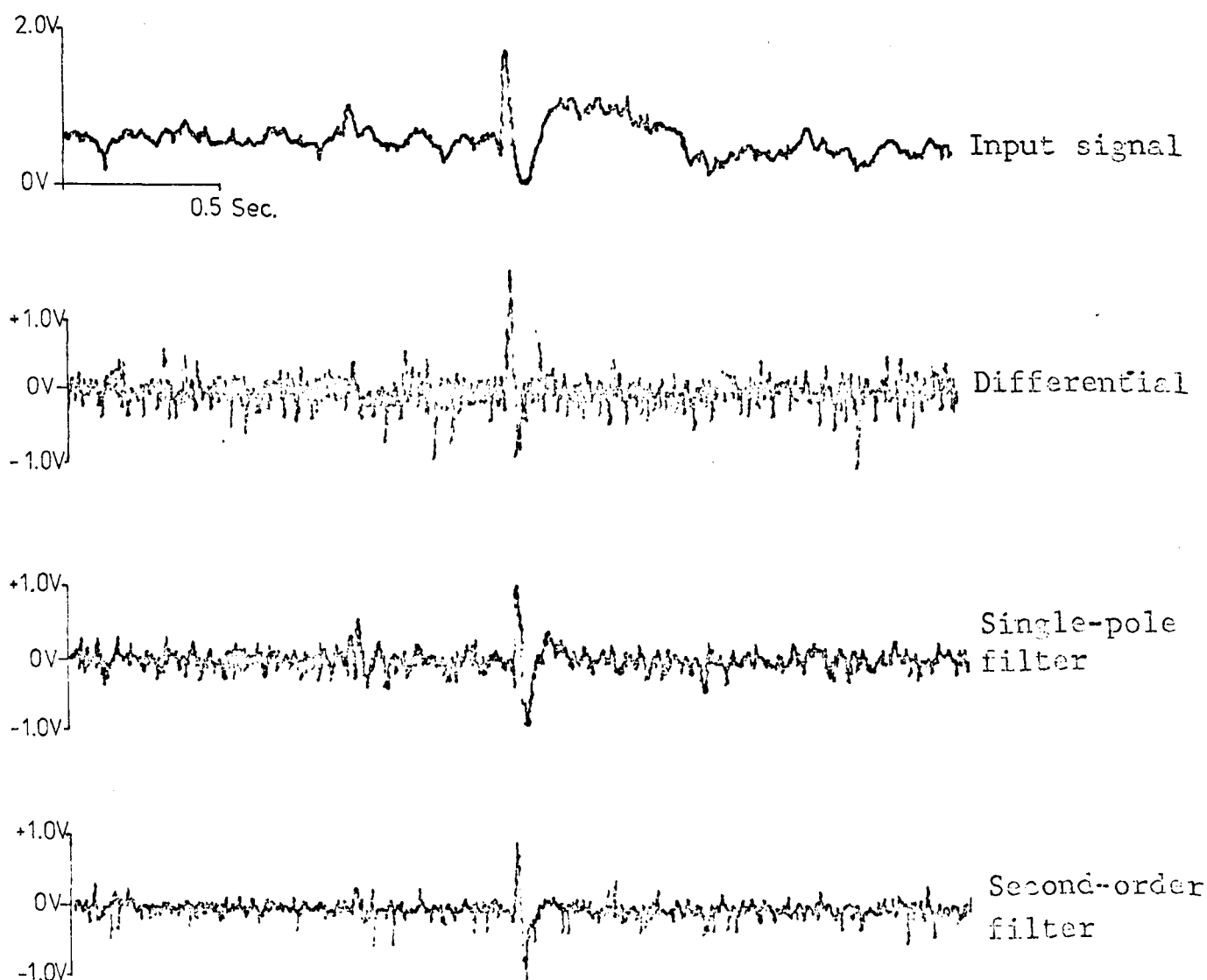


Figure 7.7 Comparison of the single-pole and second-order, low-pass digital filters.

differentiated output. The effect of the single-pole filter on a section of differentiated EEG signal is shown in Figure 7.7. The output from a second-order Butterworth filter (see later), for the same input, is also included, from which the greatly improved performance is evident.

7.3 Filter Design

Once the type and order of filter had been decided, further decisions about the method of implementation had to be made. Numerous realizations are possible for a given filter. The first fundamental consideration is whether a non-recursive or recursive form of filter should be used. For the non-recursive filter, the output is the weighted sum of present and previous input samples. For the recursive filter, the output is the weighted sum of present and previous input and output samples.

Several distinct properties are possessed by recursive and non-recursive filters. The non-recursive filter has a finite memory due to the practical limitations imposed on the number of realizable delays. The phase response is generally very good, although a large number of elements is required to obtain a sharp cut-off. Recursive filters have an infinite memory, because of the utilization of previous output values, and generally require significantly fewer elements to achieve a given cut-off characteristic. The phase response of recursive filters, however, is generally poorer than that of a non-recursive design.

Non-recursive filters are not generally favoured for digital implementation schemes because of the large number of elements which can be involved for fairly simple filters. Obviously, as the number of elements increases so does the computational load, which will impose considerable real-time constraints. A further advantage offered by the recursive approach is that recursive filters are discrete counterparts

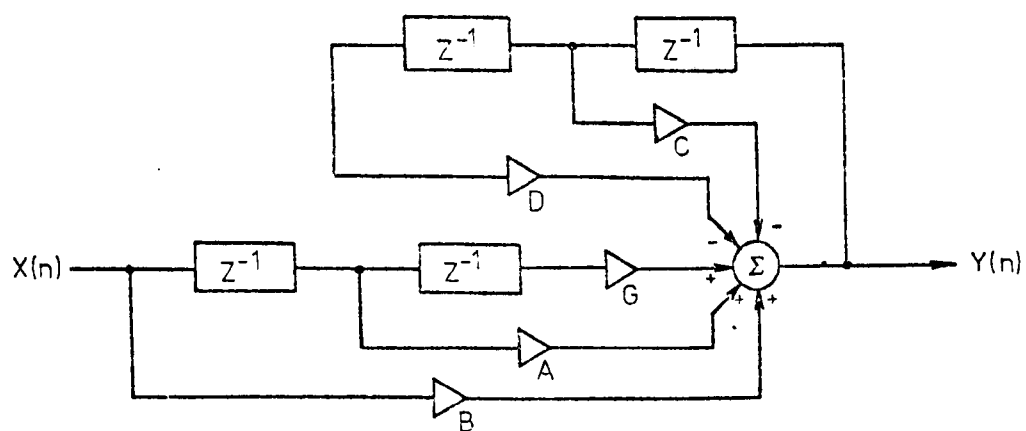
of linear, lumped-parameter, time-continuous systems. This can be an important conceptual consideration since the normal mode of design is aimed at producing a digital equivalent of an analogue filter. The inclusion of weighted values of previous outputs in the calculation of the present output is equivalent to the provision of feedback in an analogue filter.

For design purposes, the normal method employed is to derive the required frequency domain transfer function, $H(s)$, and then to convert this to the time domain via the z -transform (see Chap. 6) to give $H(z)$.

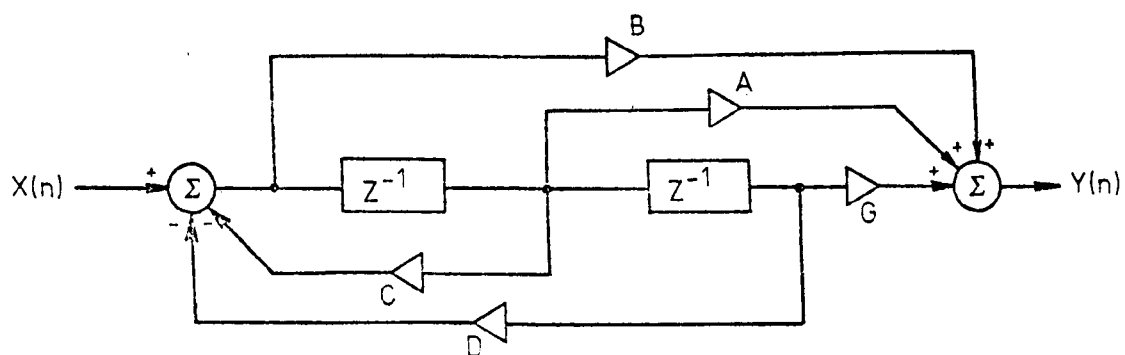
The next problem is the choice of realization of the filter. Four possible realizations for a second-order filter are given in Figure 7.8. The direct form, given in Figure 7.8.a, is probably the most straightforward, although it is not favoured for complex filters with a sharp cut-off requirement. The choice between the parallel and cascade forms is not well defined and is generally decided from a consideration of the initial form of the continuous filter transfer function.

Since the filter to be implemented is a fairly basic second-order system it was decided that the direct approach should be used. This decision is in keeping with the requirement that the complete detection system must be kept as simple and as short, in terms of execution speed, as possible.

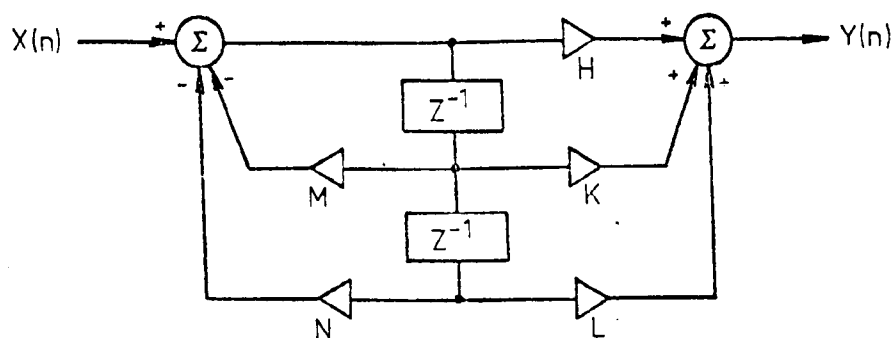
Instead of a direct mapping between the s and z -planes, it is more normal practice to use a transform known as the bilinear z -transform. This maps the imaginary axis of the s -plane into a unit circle in the z -plane in such a way that the entire left-hand side of the s -plane maps into the interior of the unit circle (see Chap. 5). The entire right-hand half of the s -plane is mapped outside the unit circle (Fig. 7.9).



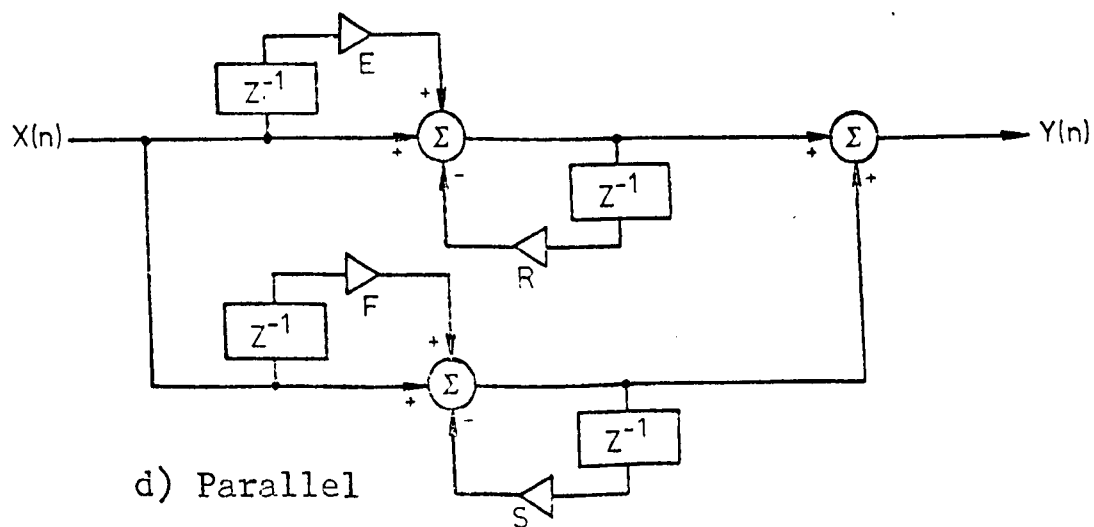
a) Direct



b) Direct



c) Cascade



d) Parallel

Figure 7.8 Realizations of a second-order recursive filter.

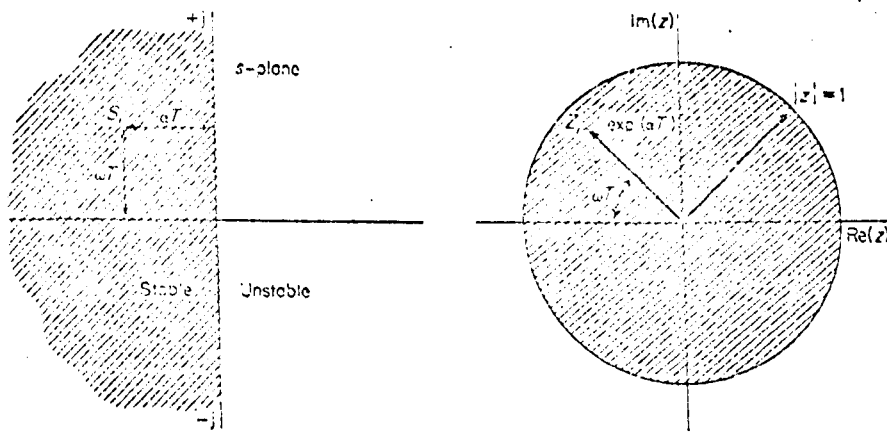


Figure 7.9 Mapping between the s and z-planes.

The technique involves the transformation of the continuous transfer function $H(s)$ in the s-plane into a new transfer function $H(s_1)$ in the s_1 -plane that is periodic in ω , with period $\omega_s = 2\pi/T$. The transform used to give this result is:-

$$s = \frac{2}{T} \tanh \frac{s_1 T}{2}$$

which, upon substitution of $z = e^{s_1 T}$, yields:-

$$s = \frac{2}{T} \left[\frac{z - 1}{z + 1} \right]$$

Several important advantages accrue from the use of the bilinear transformation, the most important being that aliasing errors, possible with the direct z-transform method, are removed. Repeated poles of a discrete transfer function resulting from aliasing effects are all mapped into a single unique position within the unit circle (assuming a stable filter).

The major disadvantage of this method is that it produces a non-linear conversion of frequencies from the continuous system (ω_a) to frequencies in the discrete system (ω_d). This is usually referred to as a 'warping' of the frequency scale. The two are related by:-

$$\omega_a = \frac{2}{T} \tan \frac{\omega_d T}{2}$$

This has the effect of compressing the complete continuous frequency characteristic into a limited digital filter frequency range of $0 < \omega T < \pi$, as shown in Figure 7.10.

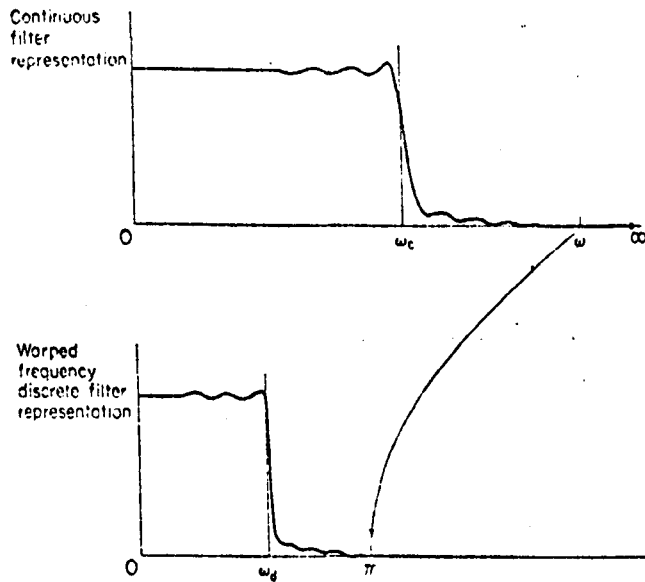


Figure 7.10 Warping of the frequency scale.

All that now remains is to define the cut-off frequency, ω_c , of the filter. As explained earlier, the features of interest will be in the 12 - 70Hz. band of frequencies. From an inspection of the abnormal spikes in the test records, it would appear that the great majority will be confined to the lower half of this range. As a result of this observation and owing to the relatively slow rate of attenuation beyond cut-off offered by a second-order Butterworth filter (see Fig. 7.5.a) it was decided that ω_c should be set at 50Hz..

This represents a ratio of $\omega_c/\omega_s = 0.1$, so:-

$$\omega_d = 0.1\omega_s = \frac{2\pi \times 0.1}{T}$$

i.e. $\omega_d T = 2\pi \times 0.1$

Sustituting:-

$$\omega_a = \frac{2}{T} \tan \frac{2\pi \times 0.1}{2}$$

$$\frac{\omega_a}{\omega_s} = \frac{1}{\pi} \tan \frac{\pi}{10} = 0.1034$$

For a Butterworth filter, the poles will be at:-

$$s = 0.1034 \times (-0.707 \pm j0.707)$$

and the zeros lie on the real axis (Fig. 7.11).

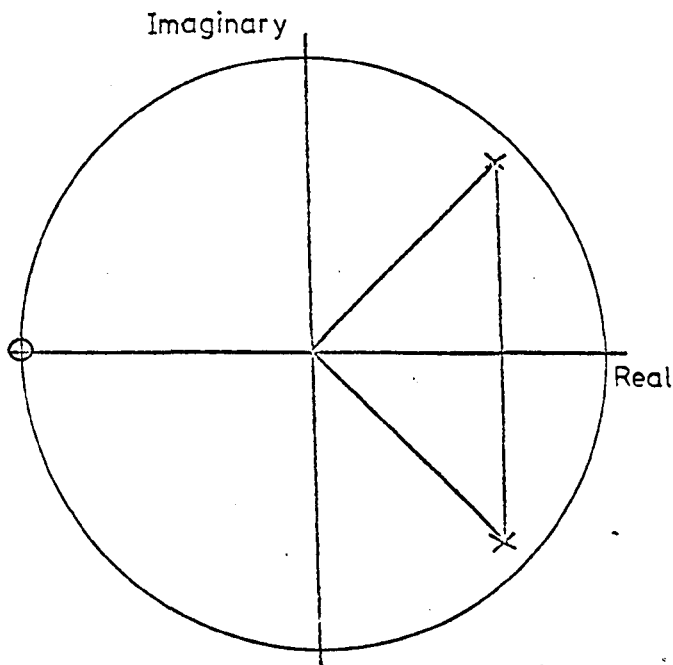


Figure 7.11

Pole-zero locations
for the second-order
Butterworth filter.

The continuous-filter transfer function, having unity gain is:-

$$\begin{aligned} H(s) &= \frac{ab}{(s-a)(s-b)} \\ &= \frac{0.0107}{s^2 + 0.1462s + 0.0107} \end{aligned}$$

Substituting $s = \frac{1}{\pi} \left[\frac{z-1}{z+1} \right]$ gives:-

$$\begin{aligned} H(z) &= \frac{0.0107}{\left[\frac{1}{\pi} \right]^2 \left[\frac{z-1}{z+1} \right]^2 + \frac{0.1462}{\pi} \left[\frac{z-1}{z+1} \right] + 0.0107} \\ &= 0.0675 \frac{1 + 2z^{-1} + z^{-2}}{1 + 1.144z^{-1} - 0.414z^{-2}} \end{aligned}$$

The realization of this filter is given in Figure 7.12.

7.4 Practical Implementation

The filter given in Figure 7.12 was implemented on the

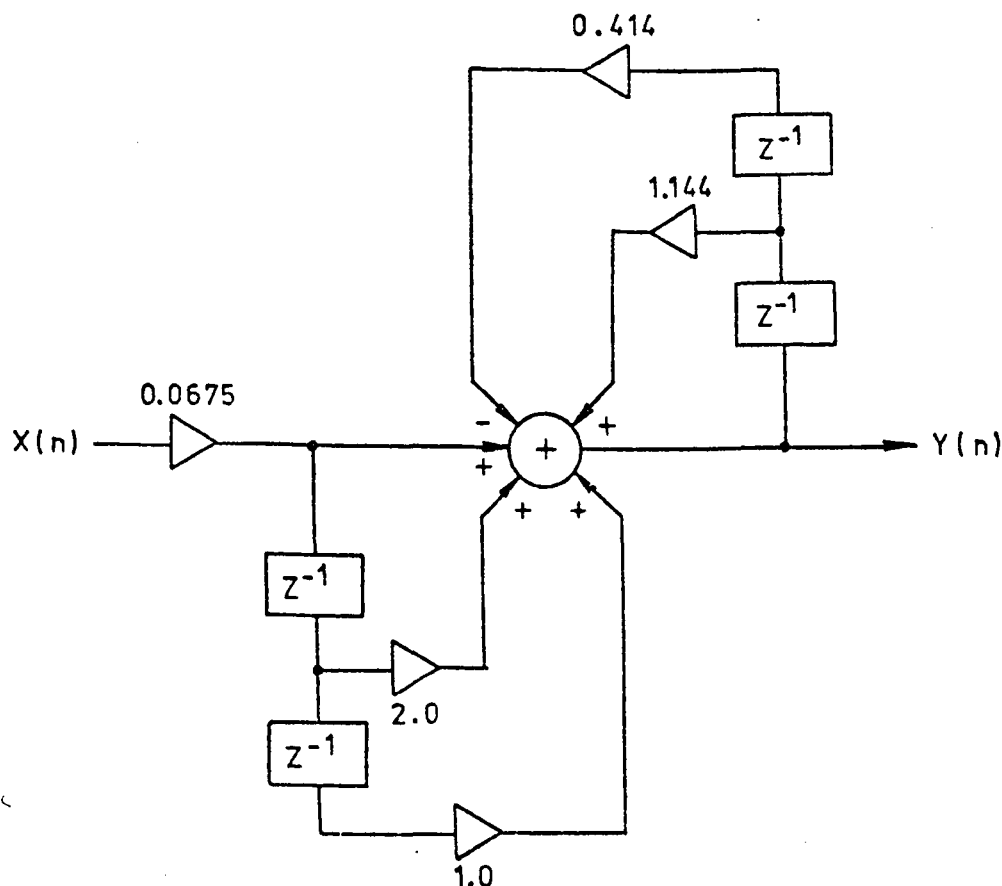


Figure 7.12 Realization of second-order Butterworth filter.

CP1600 in the form:-

$$y(n) = 0.0675 [x(n) + 2x(n - 1) + x(n - 2)] + 1.144y(n - 1) - 0.414y(n - 2)$$

where the $x(n)$ terms refer to input samples and the $y(n)$ terms, to the output samples.

From the equation given in Figure 6.11, it can be shown that the peak gain through this filter will be 2.853. Hence, two bits must be allowed to accommodate intermediate results if overflow is to be avoided. This leaves thirteen-bits with which to represent the input sample, if the most significant bit is reserved as the sign-bit. To make full use of the available register length, the eight-bit input was shifted left five places (equivalent to a multiplication by thirty-two). This permits the intermediate results to be expressed to the full fifteen-bits accuracy but unfortunately also

amplifies the inaccuracies present at the input. Since the multiplication process is a linear one, however, the overall signal-to-noise ratio is not degraded.

From an inspection of the digital filter equation, it can be seen that five multiplications are required. To alleviate the computational load imposed by the multiplication process a slightly unorthodox approach was adopted.

An approximation to the required coefficient is made by a series of shift and add operations on the appropriate sample. For this operation the digital word was considered as being divided into decimal fields, as shown in Figure 7.13, with the decimal point in the position indicated. The input was considered as a wholly fractional number.

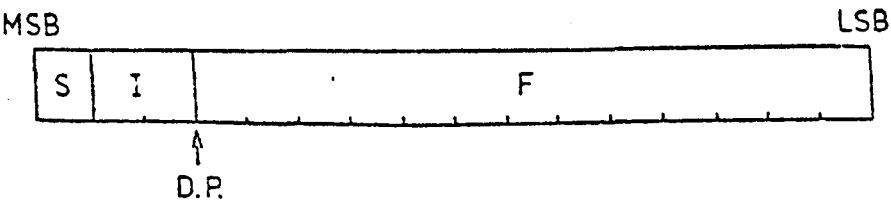


Figure 7.13 Division of computer word for shift and add operations.

As an example of the technique, consider the 0.0675 coefficient which can be realized as shown in Table 7.2.

0.0675	
<u>0.0625</u>	1/16 (4 shifts)
0.0050000	
<u>0.0039062</u>	1/256 (8 shifts)
0.00109380	
<u>0.00097656</u>	1/1024 (10 shifts)
0.000117240	
<u>0.000061035</u>	1/16384 (14 shifts)
<u>0.000056205</u>	

Table 7.2 Calculation of the 0.0675 coefficient as a series of shift and add operations.

The use of this method can reduce the time required by a full software multiply by an order of magnitude. The shifted values are all truncated before addition to the running total. Further, the difference between the realized and actual coefficients at the end of the calculation, is also truncated; i.e. the value of the final shift and add is not adjusted so that a smaller negative error rather than the truncated positive error is generated, which would correspond to rounding. The stability problems which can arise as a result of rounding were discussed in Chapter 6.

The overall accuracy of the filter can be greatly improved by a small modification to the configuration shown in Figure 7.12. It involves a shift of the 0.0675 scaling factor so that the x values component is calculated using unscaled input values and then the result scaled before

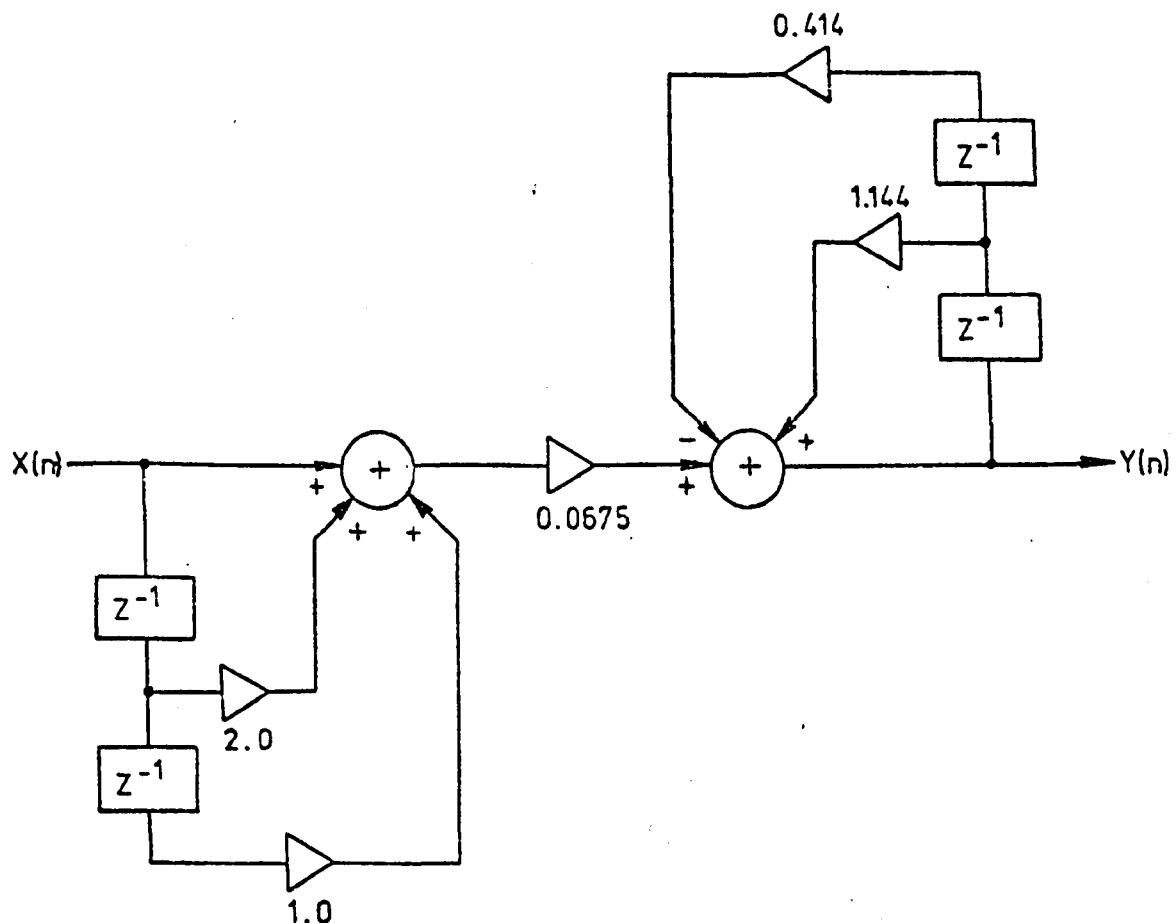


Figure 7.14 Modified filter realization.

proceeding with the y component. This means that the error introduced by the scaling coefficient is included only once. The modified filter is as shown in Figure 7.14.

No overflow problems were encountered with this modified filter since the maximum value of the x values summation will be $4 \times x(n)$ and since $x(n)$ is always less than one, the summation is always less than four. Two-bits have already been allowed to accommodate the internal magnification of the original configuration and these are sufficient to cope with the peak x values sum.

The filter had to be capable of dealing with a bipolar input since the output from the differencing process is in this form. This was achieved by a conversion of the negative input values to positive numbers before the shift and add multiplication was performed and then followed by a sign correction at the end.

The final $y(n)$ value had to be rescaled before it was output via the eight-bit D-A converter. This was performed quite easily by a five-place shift to the right, which has the added advantage that it removes any zero-order deadband effects (see Chap. 6).

The filter was tested with sinusoidal, square and triangular-wave inputs from a signal generator. The A-D converter can only deal with positive input signals and so to test the behaviour of the filter for both positive and negative signals, the converter output had to be inverted under program control. The square-wave response is shown in Figure 7.15 and that of an over-range triangular-wave is given in Figure 7.16. The triangular input demonstrates the ability of the filter to deal with full-scale inputs without overflow. The results shown were recorded via a Biomation 8100 transient recorder and an X-Y plotter.

As specified in Chapter 6, the filter is implemented entirely in software, without the need for any additional

external hardware (e.g. multiply) and involves only single-length working. The result is a filter which fulfils its design specifications and has an execution time of $\approx 500\mu\text{s}$.

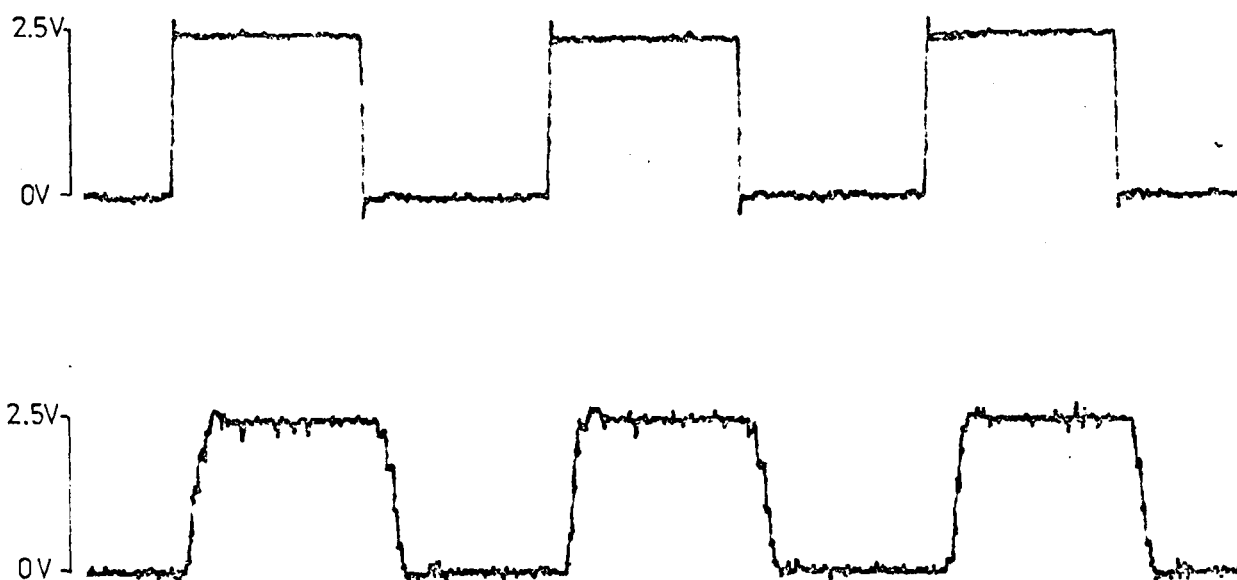


Figure 7.15 Response of filter to 10Hz. square-wave input.

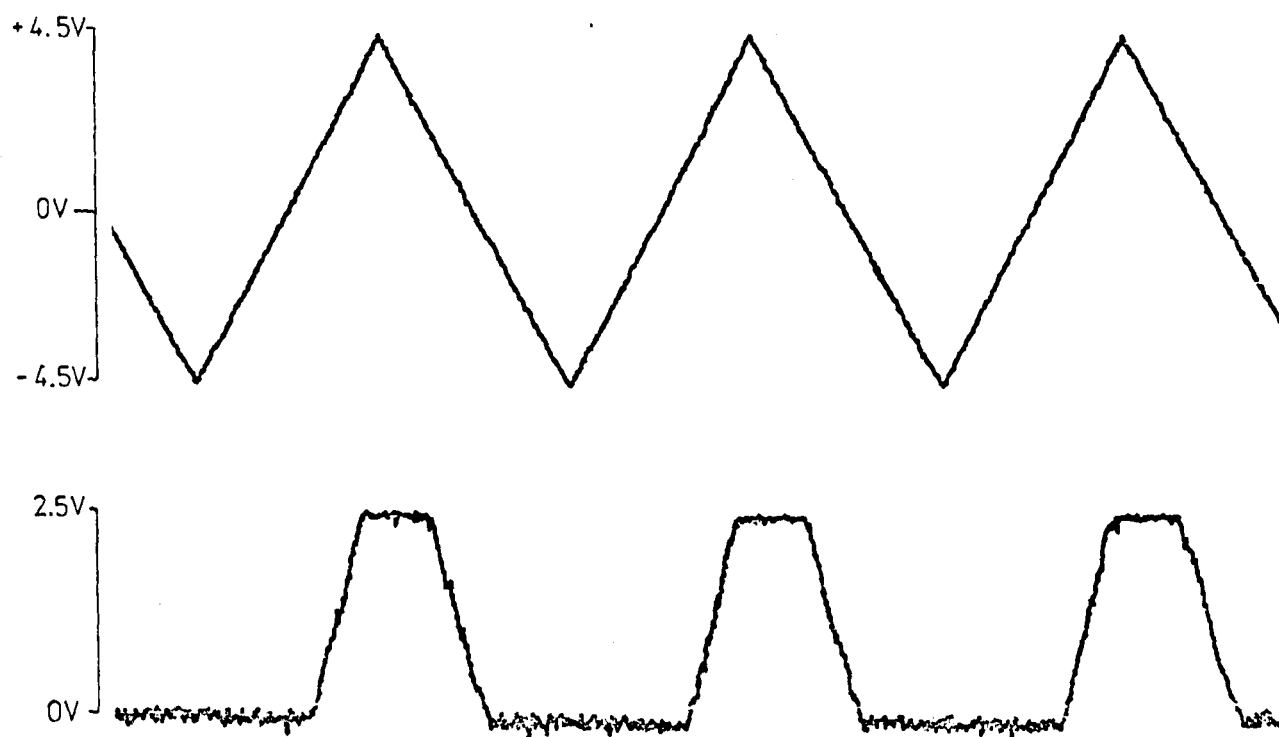


Figure 7.16 Response of filter to over-range, 10Hz. triangular-wave input.

To measure the execution speed of the filter, the routine was modified so that it ran in a continuous loop and a pulse was output via the D-A converter (i.e. the output was set to full-scale for one instruction period) for every complete pass of the program. The distance between successive output pulses was then measured on an oscilloscope.

The magnitude-frequency and phase-frequency characteristics of the filter were measured using the method shown in Figure 7.17. The results obtained are as shown in Figure 7.18.

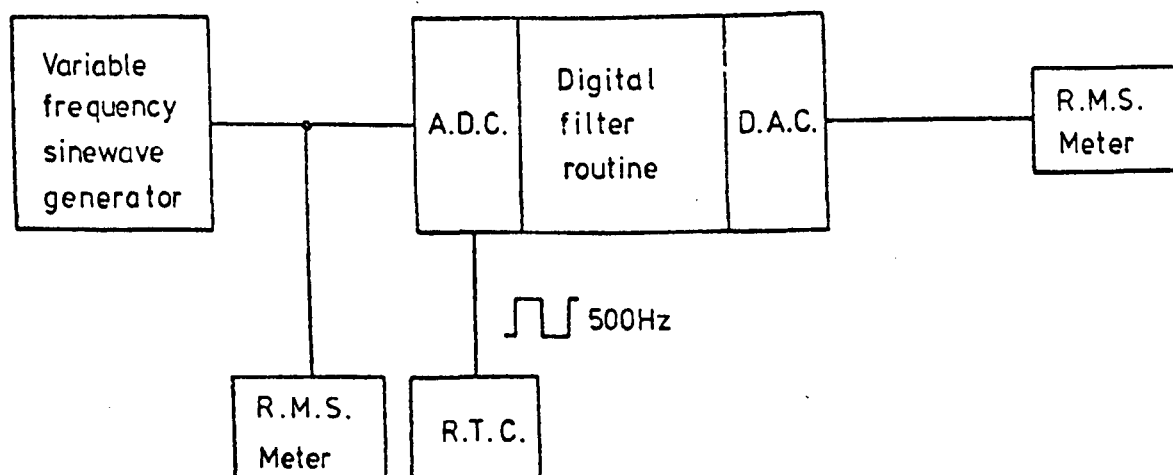
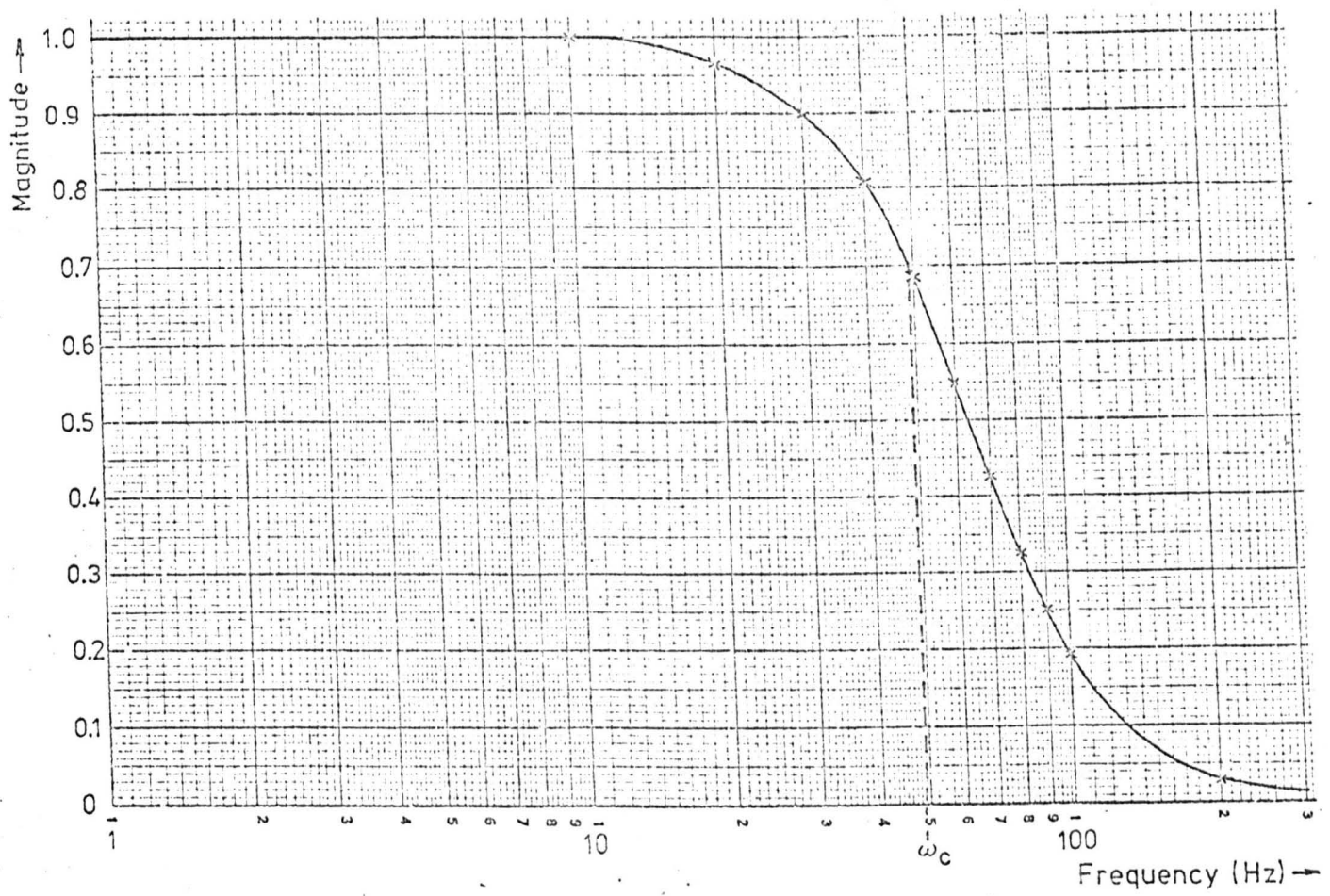
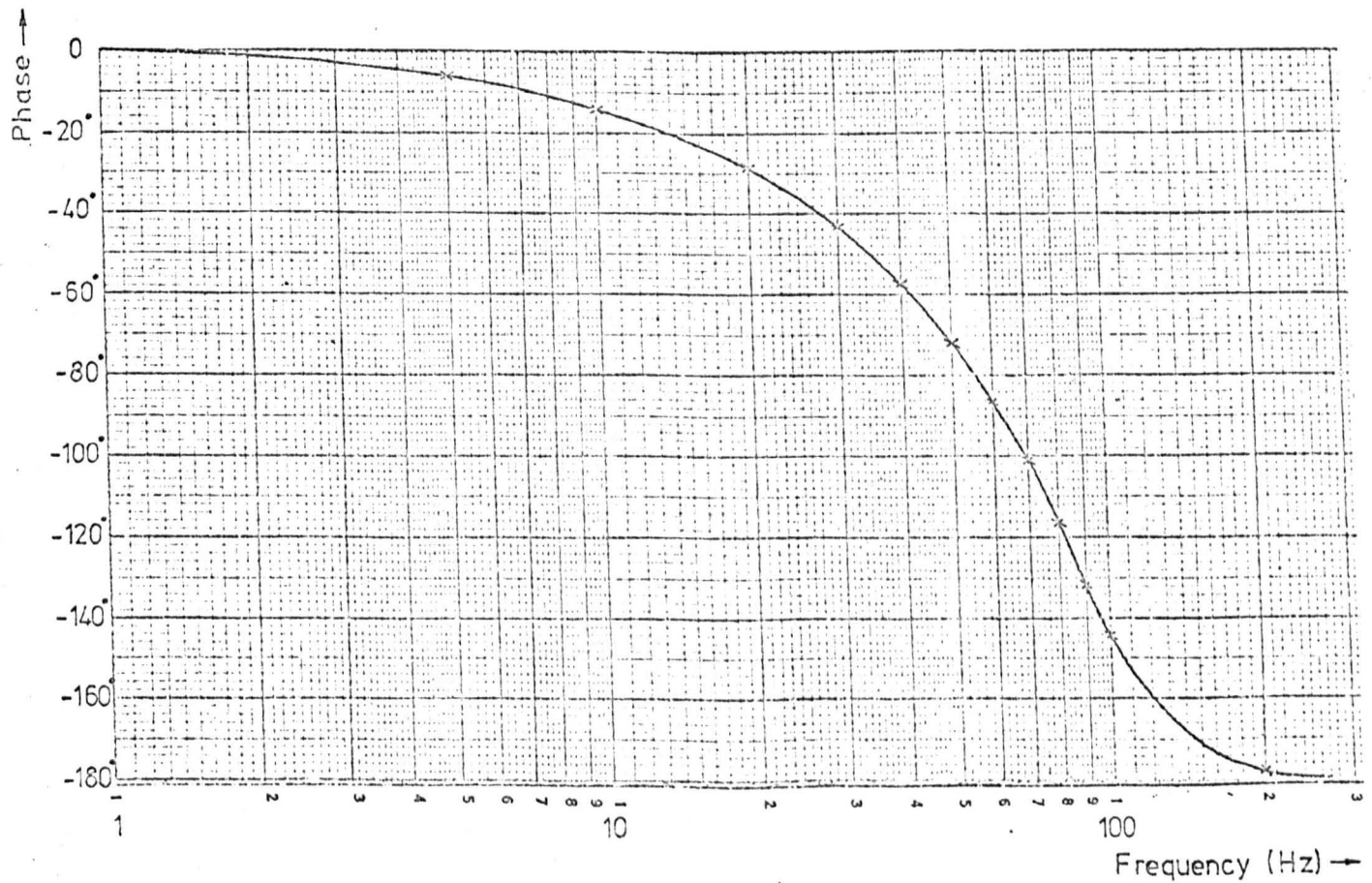


Figure 7.17 Method used for the measurement of the magnitude-frequency and phase-frequency characteristics of the filter.



a) Magnitude response.



b) Phase response.

Figure 7.18 Magnitude and phase response of the filter.

7.5 EEG Results

The differencing and filter routines were combined and the system tested with an EEG signal from the FM tape recorder. Examples of the system performance are given in Figure 7.19.

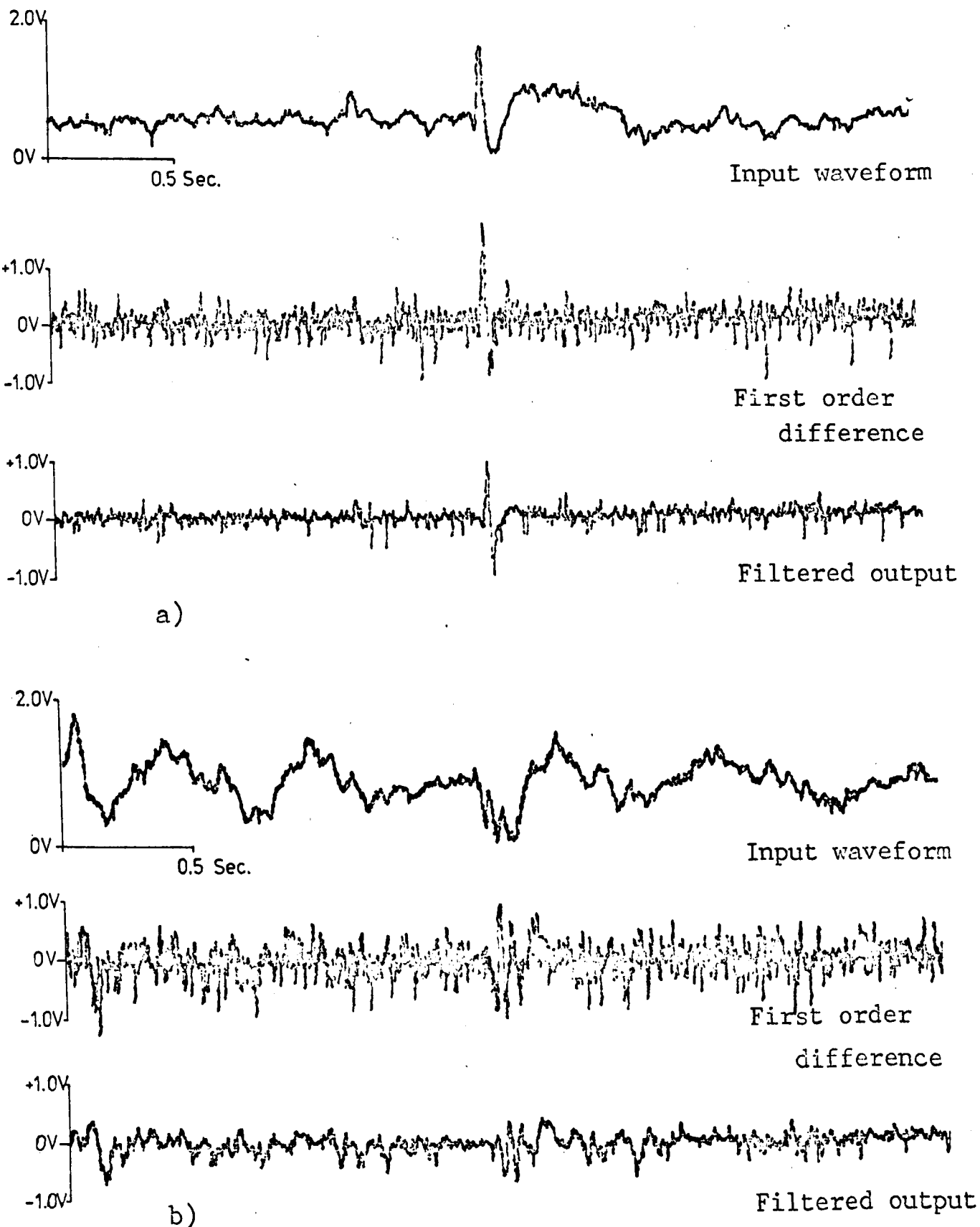


Figure 7.19 Effect of the filter on the differential of an EEG signal.

As discussed in Chapter 5, the spike feature can be further emphasized by the application of a level detection routine. If the filtered output exceeds a predefined threshold level, the system output is set to full-scale otherwise it is set to zero. An example of this process is given in Figure 7.20.

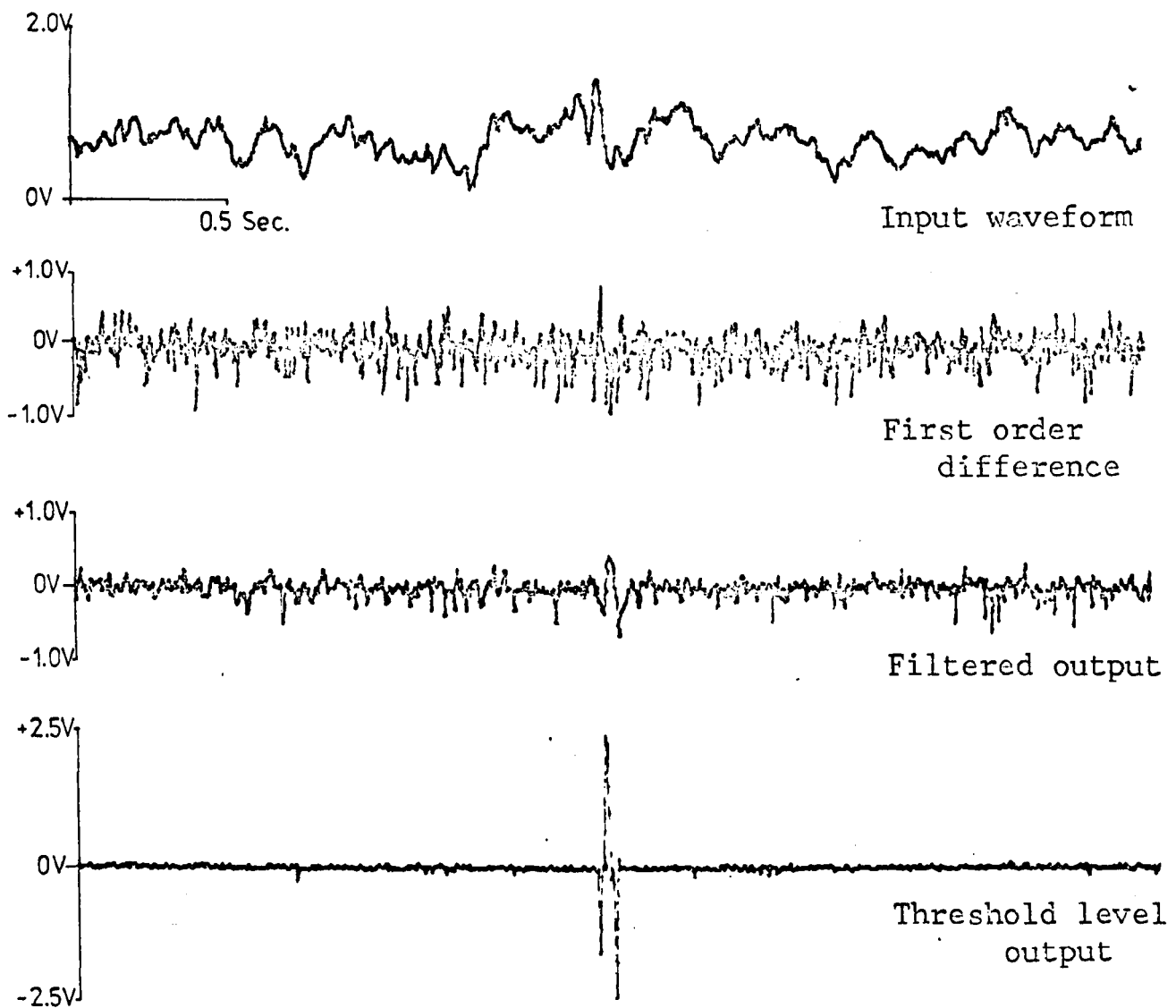


Figure 7.20 Threshold level detection applied to a section of an EEG waveform.

The level detection process presents the EEG record in a form which is comparatively easy to analyse for any abnormal spike activity. The output from the level detector will always be of the form shown in Figure 7.21 when an abnormal spike is present at the input.

The width of the two excursions, a and c, are directly proportional to the magnitude of the positive and negative

going slopes of the spike and the cross-over, b , gives an indication of the sharpness of the spike. When applied to the output from the level detector the effect is as shown in Figure 7.22.

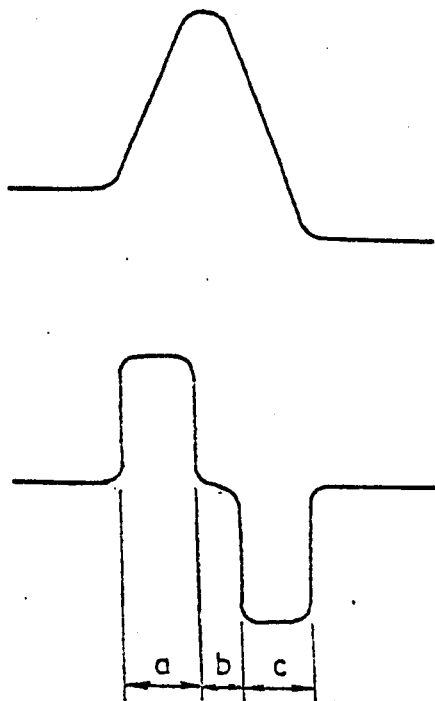


Figure 7.21 Features of a typical level detector output.

The actual magnitude of all three quantities, given in Figure 7.21, is determined by the magnitude of the input, the sampling rate (fixed at 500Hz.) and the choice of threshold level. Obviously, the choice of threshold level is of paramount importance, at this stage. In order to determine the optimum level at which to set the threshold, each of the abnormal records was processed eight times using eight different levels, ranging from 0.3V - 1.0V inclusive, in 0.1V increments. The results of all eight tests plus a copy of the input waveform were recorded in a parallel format on a two-channel, 'flat-bed' plotter.

The mechanical limitations imposed by the plotter necessitated the use of an intermediate storage system so that the results could be output at a rate at which the

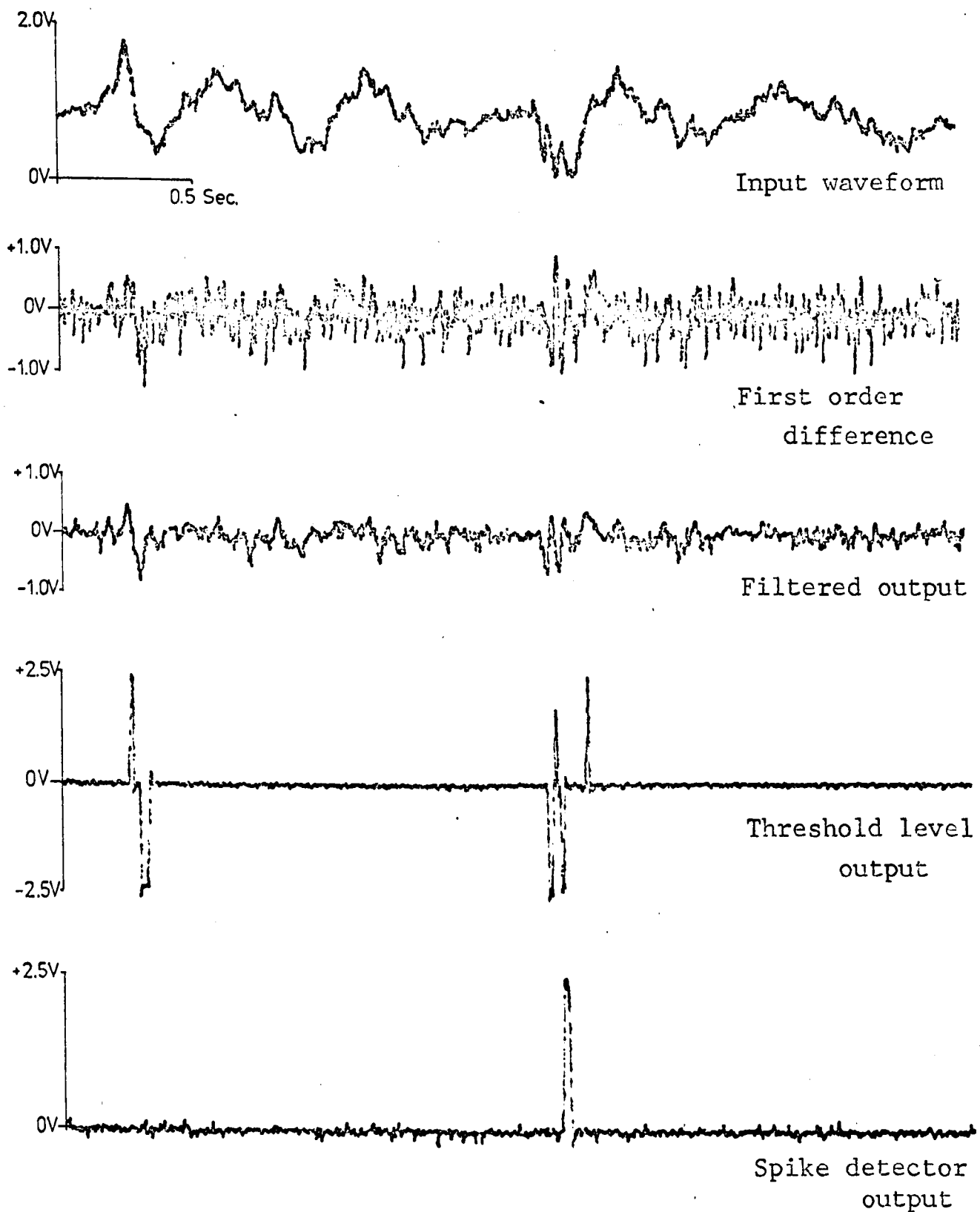


Figure 7.22 The complete spike recognition algorithm applied to an EEG signal.

plotter could respond. The cassette storage system (see Vol. II-7) was used for this purpose, with alternate real-time data samples being recorded at 250Hz. and the results then transmitted to the plotter at 7Hz.. Precautions were taken to ensure that any single-sample level detections were not lost as a result of the fact that only every-other sample was recorded.

The cassette store was loaded in two runs. During the first run, the input waveform was recorded on eight-bits of the cassette word and the 0.3V threshold on a ninth data track. These two records were then plotted before the second run. During the second run, the other seven threshold level outputs were recorded on seven of the possible nine tracks. These results were then plotted in pairs. The configurations used for the recording and playback operations were as shown in Figure 7.23.

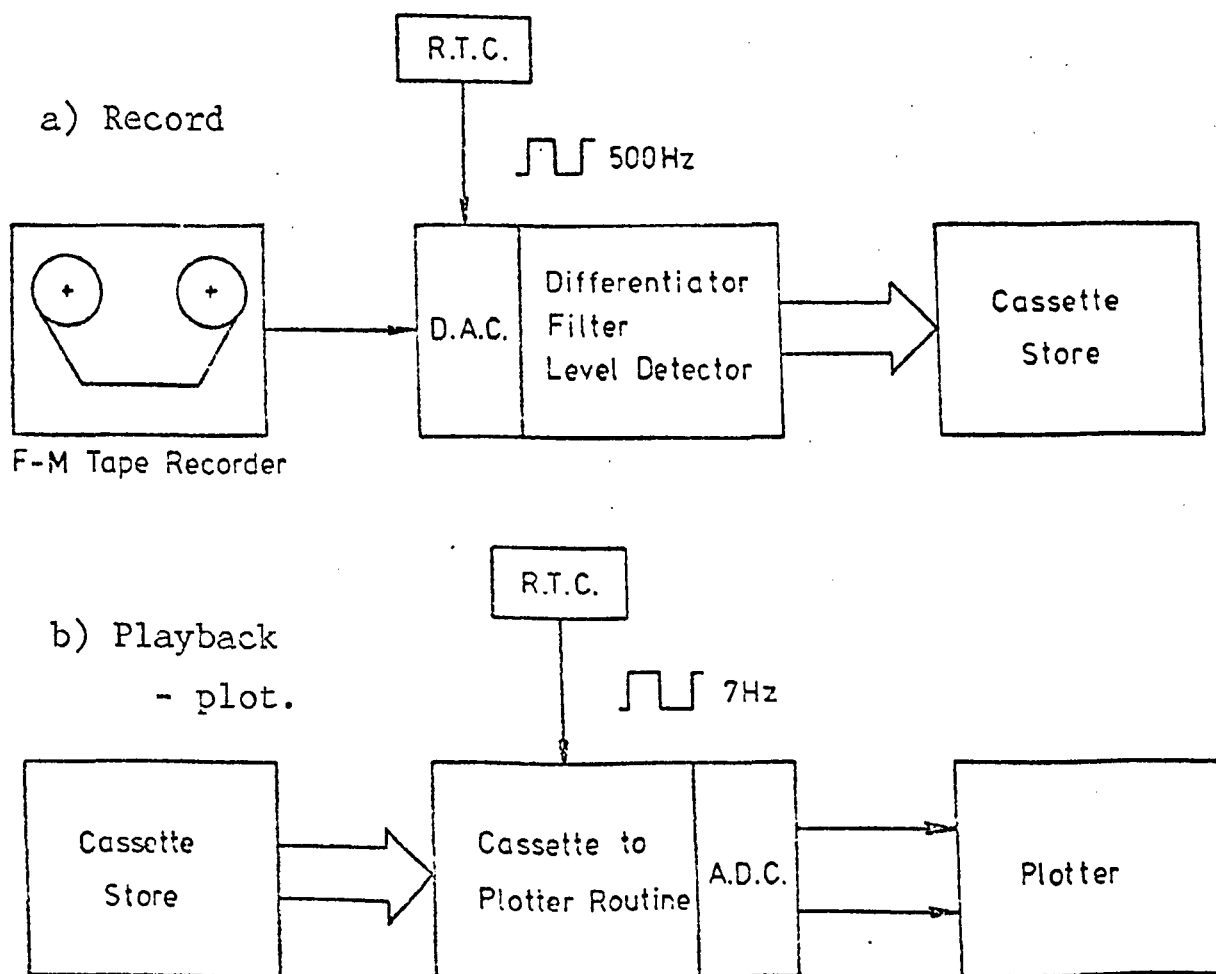


Figure 7.23 Collection and display of threshold level tests.

The plots of the original EEG waveforms were examined by a trained electroencephalographer who identified the abnormal spikes and the plots from the outputs of the level detector were then compared with these features. The results for the four abnormal records analysed in this manner are shown in Figure 7.24 with a plot of their mean values given in Figure 7.25.

Unfortunately, as discussed in Chapter 5, the interpretation of what constitutes abnormal spike activity in an EEG record is a subjective matter and so the results of Figures 7.24 and 25 do not necessarily represent an absolute assessment of the system's performance. Nevertheless, it was decided that the detection process should be capable of detecting all of the abnormal features indicated by individual electroencephalographers, even if these do differ slightly from observer to observer.

If the average false alarm rate and missed feature curves (Fig. 7.25) are superimposed on the same graph (Fig. 7.26) it can be seen that a finite region exists for the threshold level setting which covers the no missed features through to the no false alarms conditions. The intersection of the two curves represents the optimum threshold level setting for the two variables. From the graph (Fig. 7.26), this value can be seen to be 0.56V, for the records tested.

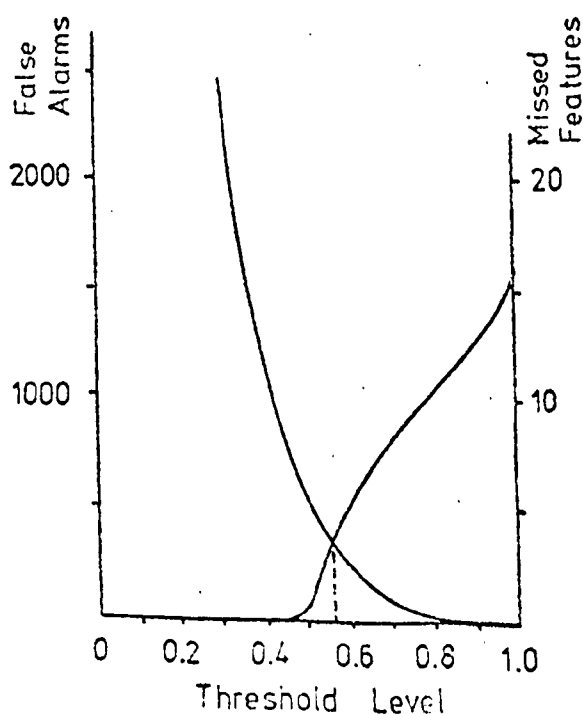
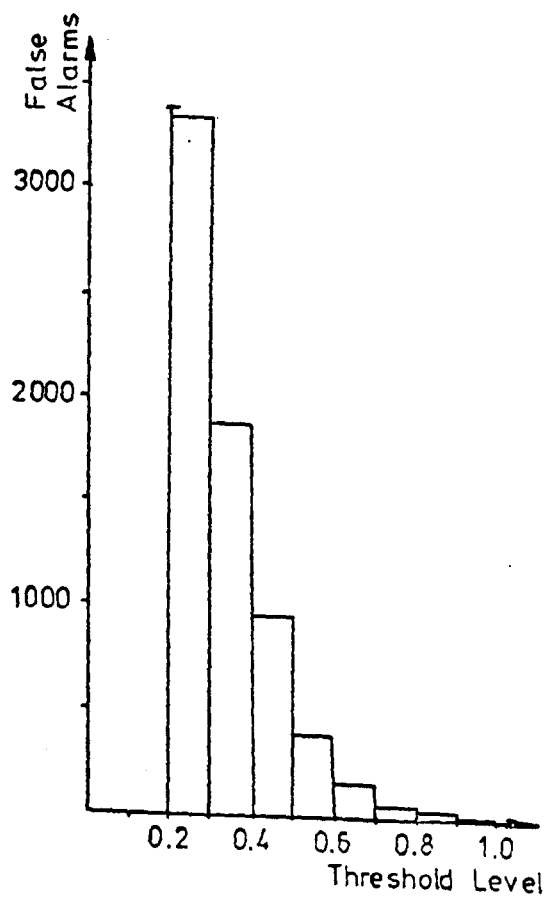
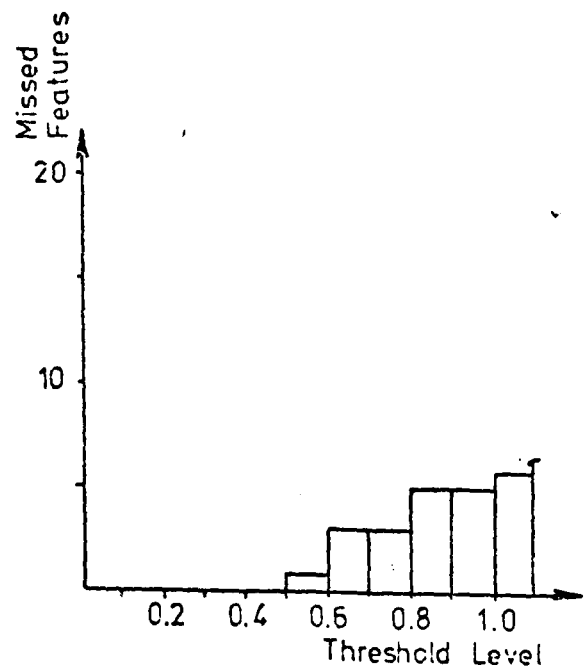


Figure 7.26 Optimum threshold level setting.



Record 1
No. of spikes = 6



Record 2
No. of spikes = 22

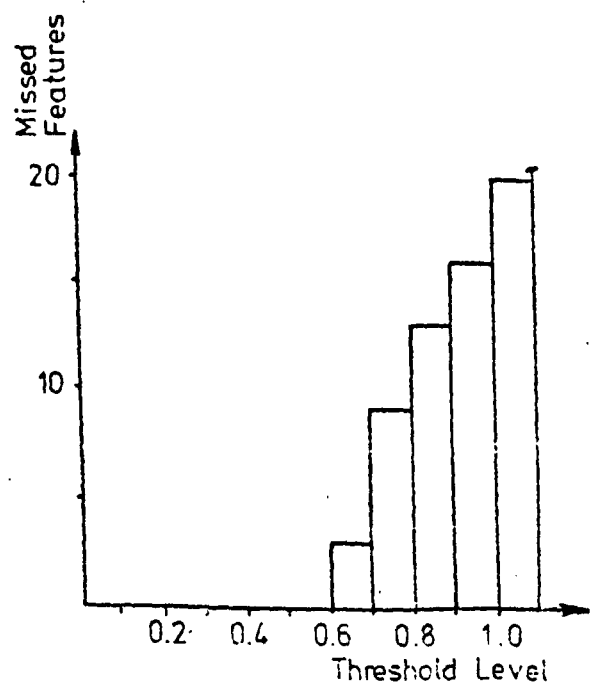
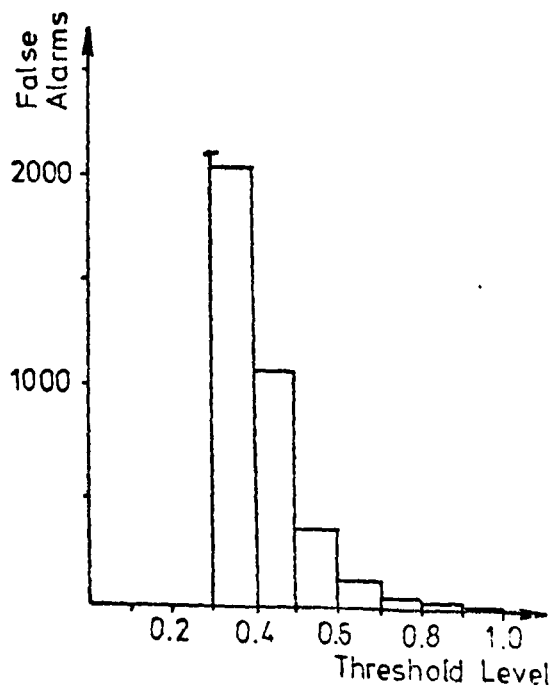
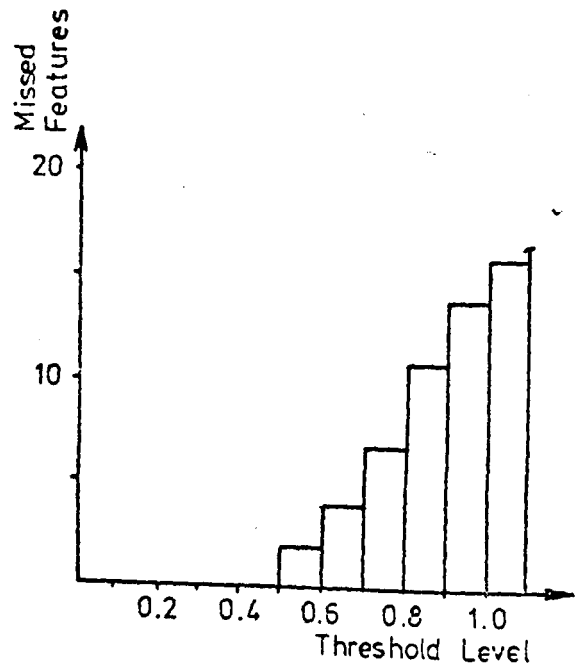
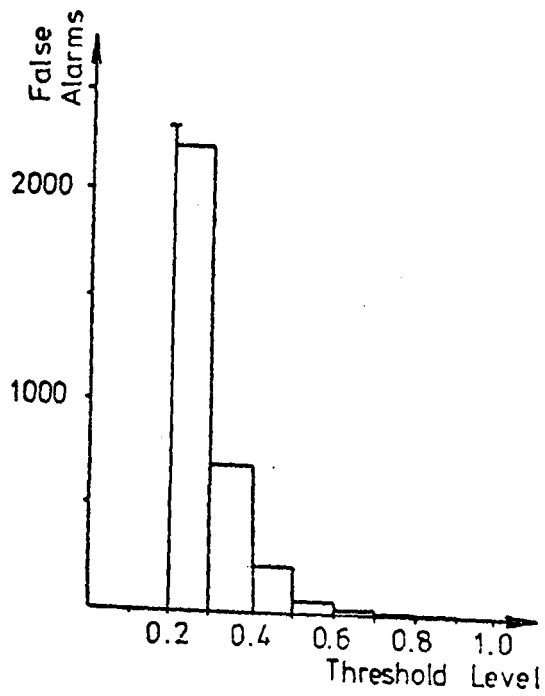


Figure 7.24 Results of threshold level tests - records 1 and 2.

Record 3
No. of spikes = 66



Record 4
No. of spikes = 23

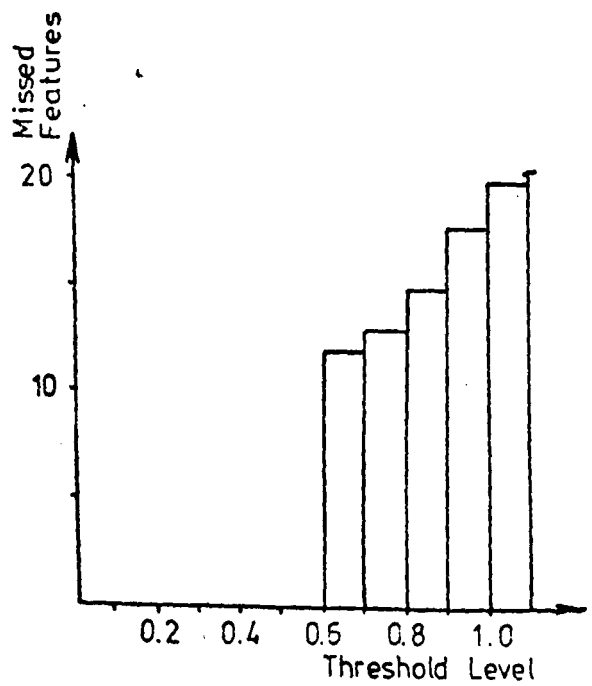
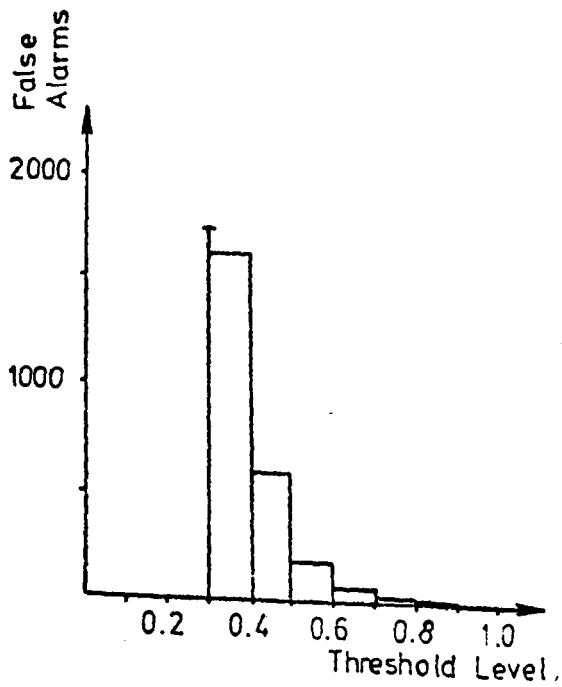
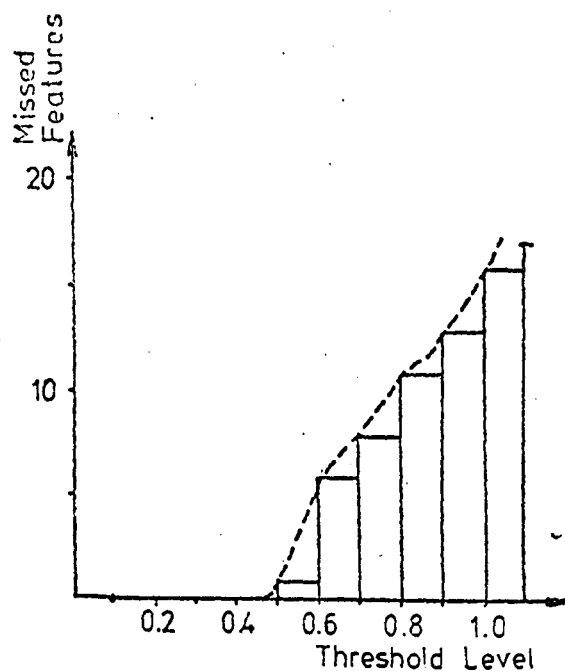
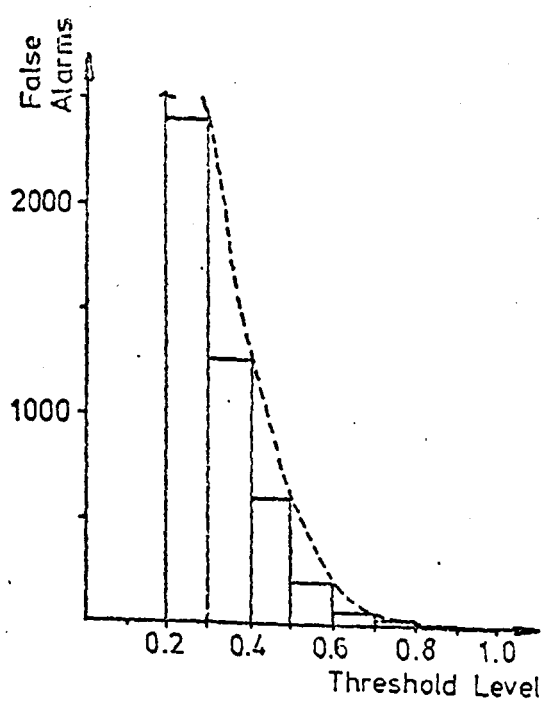


Figure 7.24 (cont.) Results of threshold level tests - records 3 and 4.



Level	Record				Average (mean) values
-------	--------	--	--	--	--------------------------

	1	2	3	4	
0.3	3358	-	2215	1611	2395
0.4	1860	2021	703	578	1290
0.5	934	1079	216	166	599
0.6	380	354	48	54	209
0.7	160	121	12	14	77
0.8	59	46	4	4	28
0.9	27	20	2	2	13
1.0	8	4	0	0	3

False Alarms

0.3	0	-	0	0	0
0.4	0	0	0	0	0
0.5	1	0	2	0	1
0.6	3	3	4	12	6
0.7	3	9	7	13	8
0.8	5	13	11	15	11
0.9	5	16	14	18	13
1.0	6	20	16	20	16

Missed Features

Figure 7.25 Mean values for different threshold levels.

It has been decided that a zero missed feature rate should be aimed for and this requires a threshold setting in the 0.4V - 0.5V region. The false alarm rate for this setting will be in the region of 600 - 1000 on average per 3min. record (i.e. 200 - 300 per min.), but this value can be reduced considerably by the inclusion of the spike detection algorithm in the analysis system. The graphs shown in Figures 7.24 - 26 represent only the number of times that the processed EEG signal has crossed the threshold level during the 3min. test period.

The four abnormal records were re-analysed with the complete spike detection algorithm using three different threshold settings within the selected region; 0.4V, 0.45V and 0.48V. The results were output via the transient recorder and a visual display unit and the number of spike detections counted for each record at the various level settings. The results obtained are tabulated in Table 7.3.

Rec. No.	No. of spikes	No. of spikes detected		
		0.4	0.45	0.48
1	6	37	17	10
2	22	152	79	44
3	66	99	76	67
4	23	31	17	15

Table 7.3 Results for different threshold level settings.

From an inspection of Table 7.3, it would appear that the 0.48V setting offers the best choice of threshold level for all but record 4. A re-analysis of record 4, however, revealed that of the original twenty-three features classified as abnormal, eleven could be re-classified as sharp waves. Similarly, for record 3, some of the features originally classified as spikes were in fact sharp waves. Since the spike detector has been designed to reject sharp waves (see

Chap. 5), the totals for the records had to be amended when this routine was included in the program. The new classification is given in Table 7.4.

Record No.	No. of spikes
1	6
2	22
3	61
4	12

Table 7.4 Re-classified spike counts for the four abnormal records.

It should be noted at this stage that the grossly abnormal nature of the records made the original classification extremely difficult. The analysis and subsequent re-classification of the records only serves to emphasize the subjective nature of visual EEG pattern classification, the very point cited as the main obstacle to the formulation of an automatic detection system, in Chapter 5.

The threshold level was set to 0.48V and the four abnormal and three normal records analysed with the complete spike detection routine. The results were all plotted via the flat-bed plotter using a similar method to that employed for the threshold level tests (see Fig. 7.23), with the input waveform recorded on eight-bits of the cassette word and the spike detector output on the ninth bit. The results for all seven runs are tabulated in Table 7.5.

The false alarm counts have been sub-divided into a definite false alarm column and a 'false alarm but possible spike' column. This second column contains all of the detections which could possibly be defined as spikes but do not correspond to any of those already accounted for in Table 7.4. Four other detections occurring in the abnormal records were re-classified as actual spikes after an inspection of the results and these are indicated in the

'No. detected' column of Table 7.5.

Record No.	No. of spikes	No. detected	No. missed	Possible spikes	False alarms
1	6	6 + 3	0	1	0
2	22	21	1	8	15
3	61	60	1	5	2
4	12	10 + 1	2	2	2
5	0	-	-	-	57
6	0	-	-	-	21
7	0	-	-	-	0

Table 7.5 Results for the complete detection system.

As can be seen from Table 7.5, the success rate of the algorithm is very high with all but four spikes out of an original total of one-hundred and one being correctly detected, i.e. a 96% (approx.) success rate. The situation with regard to the false alarm rate, however, is a little more serious. For the abnormal records, the results for record 2 give the most cause for concern, with a possible total of twenty-three false alarms. Problems were expected with record 2, however, since it is very 'spiky' in appearance and gave a great deal of trouble at the visual classification stage.

7.5.1 Artifact Rejection

For the normal records, all but two of the seventy-eight false alarms can be directly attributed to the presence of muscle artifact in the signal. As a means of reducing this particular cause of false alarms, the method discussed in Chapter 5 was implemented. The rejection scheme is based on a running count of the number of zero-crossings, occurring at the filter output, over a given period of time (number of samples). Since muscle artifact usually exhibits

itself as a burst of high-frequency activity in the EEG, the running count should increase during these periods and hence provide the basis of a rejection system.

The count period was set at $1/4$ sec. (i.e. 125 samples) and a count threshold of twelve chosen as the level above which the signal would be assumed to contain artifact. This threshold setting corresponds to an input frequency of 48Hz..

The realization employed is based on a cyclic buffer system into which the sign-bits (bit 15) of the last 125 samples are stored together with a marker-bit, which is used to indicate that a change in sign has occurred. A separate count is kept of the number of sign-change bits contained in the buffer at any given time. The complete operation is shown in flow-diagram form in Figure 7.27. This implementation of the rejection system is rather wasteful in terms of storage but is very fast in execution and the real-time performance was considered more important than the sacrifice in memory space.

The artifact rejection routine was included as part of the main program but when tested it was found that the threshold count had to be raised to eighty before the system would operate correctly. Counts of up to seventy were obtained for the normal and abnormal records and this rose to ninety (approx.) during periods of artifact behaviour.

These high counts were probably caused in part by the effects of the differencing operation but mainly as a result of noise pick-up at the A-D converter input. The processor, when running, appears to induce a considerable noise component (several tens of millivolts) in the signal leads. Numerous attempts have been made to isolate and remove the cause of this 'system noise' but none have met with any great success. It would appear that the noise is transmitted through the 0V line but attempts at decoupling to ground via ceramic capacitors have had little effect.

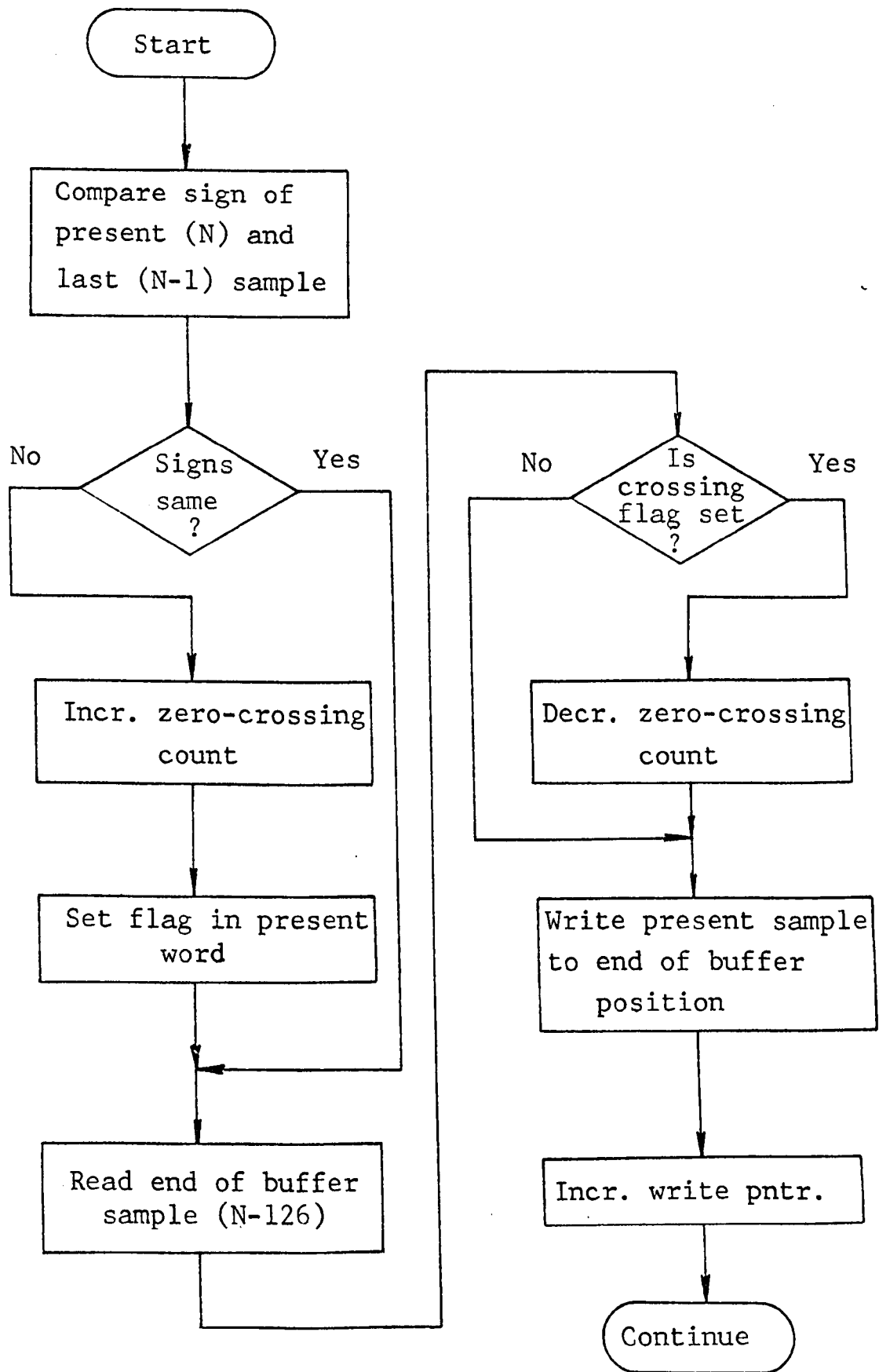


Figure 7.27. Implementation of the zero-crossing count method of artifact rejection.

The best result that could be obtained was to place a simple R-C low-pass filter ($\omega_c \approx 5\text{kHz.}$) on the input to the A-D converter, decoupled to ground and not the system 0V. This at least reduced the noise component present at the input and did appear to improve the performance of the artifact rejection algorithm when applied to both the normal and abnormal records.

The normal records were re-analysed and a reduction in the number of false alarms was observed (see Table 7.6 and Fig. 7.28). Unfortunately, spikes were also occasionally rejected from the abnormal records, apparently on a random basis. This occurred very infrequently and on the occasions that it was observed, a different spike was always involved. This behaviour casts serious doubts on the suitability of the rejection system since one of the prime requirements for any such scheme must be that it does not interfere in any way with the results obtained for artifact-free records. It is probably still due to noise problems at the input, but this has not been conclusively demonstrated. Obviously, a great deal of further research is required on the whole problem of artifact rejection.

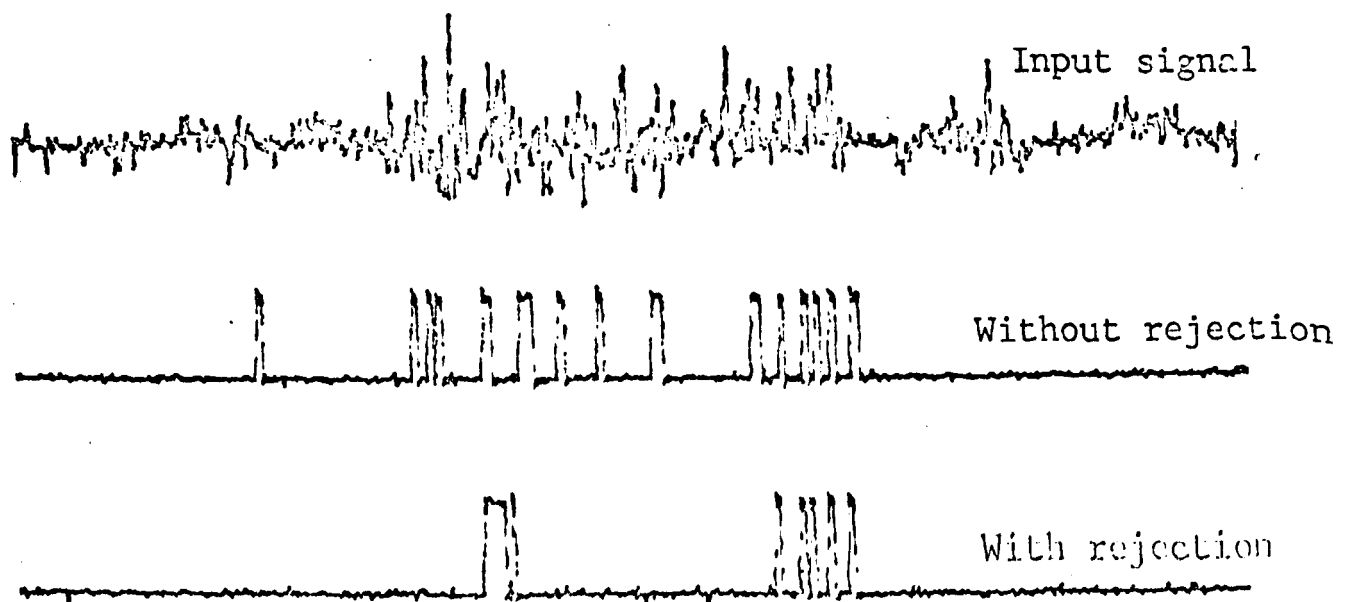


Figure 7.28 Effect of artifact rejection algorithm.

Record No.	No. of false alarms without rejection system	No. of false alarms with rejection system
5	57	41
6	21	18
Totals	78	59

Table 7.6 Effect of artifact rejection routine on the false alarm count for two normal records.

7.6 Concluding Remarks

The routines developed appear to perform to the specifications laid down for the system in previous chapters, although the artifact rejection scheme is not very satisfactory.

One final test was performed and that was to measure the execution speed of the program. This was achieved in a similar manner to that used to measure the execution speed of the filter routine (see sect. 7.4). The complete spike detection program was found to take $\approx 640\mu\text{s}$ for a complete pass and this increased to $\approx 720\mu\text{s}$ when the artifact rejection routine was included.

Chapter 8

Discussion and Suggestions for Further Work

8.1 Discussion

The work recorded in this thesis represents only the beginning of what promises to be a very fast moving and fruitful field of research.

Microprocessors are, as never before, forcing engineers and scientists to consider not only the application but also the processing power of the central processing unit (CPU) in their computer orientated systems. Over the next few years, the microprocessor is likely to revolutionize the field of computer-aided measurement. An approach such as the Roving Slave Processor (RSP) could help to ease the impact of this new technology and make the potential power offered by microprocessors available to a great number of users, who may not have the necessary computing or electronic background in order to configure their own systems. In essence, the RSP is a software definable instrument, which may be quickly and easily reconfigured to perform a whole variety of operations ranging from very simple control functions up to real-time signal-processing.

The essential features of the RSP are given in Chapters 3 and 4 and an application in the very demanding area of real-time signal-processing is described in Chapters 5 to 7. The performance of the system and the results obtained, and their implications, are discussed in this chapter. Considerable attention is also given to areas where further work is required; many problems still remain, some of a development and others of a research nature.

The early confidence shown in the choice of sixteen-bit processors appears to have been well founded; both prototypes have demonstrated their ability to perform complex real-time signal-processing operations. In particular, the CP1600 has proved that it is not only capable of performing all of the computation necessary for the abnormal EEG spike detector application, without the need for special add-on hardware

units, but has also proved very easy with regard to the interface of peripherals (see Vol. II) and in use.

The Gimini microcomputer system has been very effective for the development of both the hardware and software required for the EEG application. Indeed, in its present form the Gimini could assume the role of master computer and service the planned portable and miniature versions. This is a major departure from the original philosophy behind the RSP (see Chap. 3) and is strictly speaking of little relevance to the present research.

The original intention of the RSP approach was to produce a vehicle by which the capabilities of an available minicomputer system, complete with a full range of expensive peripherals, could be enhanced and extended. This is still the purpose behind the research, but a great deal of interest has been shown in the RSPs as the main element of the total computing system, with the master computer operating almost exclusively in an RSP support role. This is particularly true for the miniature version. In many instances, potential users do not possess a minicomputer system and so the possibility of utilizing a cheap microcomputer could be of great relevance to them.

In spite of the relative success of the CP1600 microprocessor, the sixteen-bit wordlength is only just sufficient for the digital filter implementation (see Chap. 7), which is not a particularly demanding configuration with a fairly large ω_c/ω_s ratio (i.e. the poles are not forced close to the unit circle (stability boundary in the z-domain) and so greater errors can be tolerated in their placement before the filter becomes unstable). The errors involved, however, are fairly small, a complete analysis being given in Appendix E. As can be seen, the errors are sufficiently small not to warrant double-length working.

This is a very important result, since although the filter employed was not particularly complicated as far as

digital filters are concerned, digital filtering is none the less a very demanding real-time signal-processing operation. The ability of the processor to deal with this operation within the confines of its single-length working capability is very important; it demonstrates that sixteen-bit microprocessors, in general, possess sufficient computational power to cope with a whole range of real-time problems.

To consider the actual EEG analysis system in more detail, the most important result must be the demonstration of the ability of a microprocessor to deal, successfully, with the problem of detection of abnormal sharp spikes in the EEG signal in real-time. The techniques employed to achieve this result, though, must also rank of great importance.

The fundamental feature of the detection system is that it is implemented as a time domain operation. The arguments against the more commonly employed frequency domain analyses are given in Chapter 5, where the non-stationarity of the EEG signal is cited as a major consideration. In some respects, the non-stationarity is fortuitous since, in general, stationary processes have no Fourier transforms [52]. As discussed, the non-stationarity problem can be overcome by the application of a piecewise approximation to the signal over which period the signal may be assumed to be stationary, in the wide sense (see Chap. 5). The piecewise approximation involves a partitioning of the signal into equal length records in the time domain. The effects of a contraction of the time scale on the frequency domain is expressed as an uncertainty principle, in which it is shown that as the limits imposed on the definition of t are tightened, so the definition in ω becomes more vague (see Chap. 6).

Owing to uncertainties about the period over which the EEG signal may be considered as stationary, it was decided

that very short records should be used; less than 1 sec. in duration. To quote Gevins et al [7], 'Violations of the assumption of stationarity have not been adequately studied. Huber et al suggest that even for 1 sec. epochs, there are significant non-stationarities in EEG data as revealed by bispectral analysis. Until these effects are understood, caution must be exercised in the physiological interpretation of the results of analysis.'

Such a short time scale effectively rules out the use of any frequency domain analysis. Instead, a time domain system was opted for in which a model of the sought feature was used as a 'moving window' of variable width, but of approximately 1/12 sec. (i.e. 40 samples) duration.

The method has proved extremely effective when applied to the test EEG signals. It was particularly attractive in that it employed a very short time scale and was also very easy to realize as a discrete operation. This results in a very simple program and, hence, makes the method ideal for a microcomputer implementation and also very effective as a real-time process.

Another important feature of the spike detection system is the inclusion of a second-order digital filter routine instead of one of the more common techniques, the running mean, to perform the low-pass filtering operation. The poor performance of the running mean as a filter was discussed in Chapter 5, where it was decided that, in spite of the real-time implications, a more elaborate form of digital filter should be used.

Also rejected as a possible method of analysis was the matched filter technique. One reason for its rejection were the problems involved in the formulation of a suitable model with which to define the spike, and the fact that the filter will tend to reinforce any assumptions made about the signal in order to synthesise this model. The other

major reason is the requirement that for a simple implementation, the signal should be uncorrelated between the features of interest. This has been demonstrated not to be the case for the EEG signal by the use of autocorrelation techniques (see Chap. 5, Fig. 5.2).

The autocorrelograms obtained help to explain the problems encountered by Lloyd and Binnie [13] who found that their matched filter performed best with grossly abnormal EEG signals. From the autocorrelograms it can be seen that as the EEG behaviour becomes more abnormal so the correlation decreases until, for the grossly abnormal records, the signal is totally uncorrelated.

8.2 Suggestions for Further Work

The next important phase in the development of the RSP is likely to be the production of a miniature version. The miniature RSP is now a practical proposition, which is much earlier than was anticipated or predicted at the commencement of this project. A great deal of research and development is still required, however, before the first prototype may be produced. Most of the problems to be overcome are of a practical nature, with such factors as power supply requirements and heat dissipation being of major concern.

The memory system still requires some further consideration, with the areas of volatility and reliability being of great importance. The problems involved in the provision of a form of bulk back-up store, which may be demanded in certain applications, is also of major interest, with magnetic bubble or CCD memories (see Appendix B) appearing the optimum choice at present.

An interesting possibility, with regard to the miniaturization problem, is the technique of 'chip-bonding' with which the RSP could be reduced to very small dimensions. Basically, the method involves the use of the individual silicon chips, which go to make up the various components of the RSP (e.g. CPU chip, memory chips, etc.), which are all affixed to a single substrate and then interconnected with very fine gold wires, ultrasonically bonded to small 'pads' around the edge of the chips.

A great many practical problems exist for an approach of this type, for example very expensive equipment is required to secure and interconnect the chips; integrated circuit manufacturers will only release their devices in chip form in very large quantities (e.g. >1000) and then they are usually placed in Nitrogen filled containers in order to avoid any possible contamination. Also, great problems would be experienced in such areas as heat

dissipation of the final device, and the number of attempts required to produce complete working systems from a collection of very delicate chips.

These problems can, however, all be overcome and the technique is quite widely used in industry, particularly for the production of hybrid and military equipment.

Once the problems of miniaturization have been investigated and solutions found, it should be possible to produce RSPs very cheaply and this will emphasize the need for an efficient means of program preparation and generation. This will arise as a natural progression, since, as the availability of cheap and small RSPs increases, so the number of people interested in their capabilities and use will also increase.

A start has been made on a programming system, aimed at fulfilling this need. The system has been called the High-Level Definer, and is described in Chapter 4 and Volume II-4. It is the author's opinion that most of the remaining research to be conducted on the RSP project, lies in the programming and program preparation fields. The High-Level Definer represents just one attempt at a solution to what is possibly the most complex problem of the entire RSP concept.

It would appear that the High-Level Definer is a satisfactory solution as far as it goes, and that the underlying principles are sound. The ability of the system to generate efficient programs from a source containing a few simple statements has been demonstrated (Vol. II-4). There are, however, certain problems and limitations imposed on its effectiveness [24] and the whole topic demands a great deal more fundamental research.

Without a proper programming system, the RSPs will remain 'basic hardware shells' and be of little general use; they will remain, like microprocessors, the province of

persons well versed in computer technology and programming techniques. They could still prove extremely useful, even if confined to this small group, but to realize their full potential the RSPs must be made available to non-expert users. The overall concept behind the RSPs is that they are to be software definable instruments and should, correspondingly, demand no more of the operator than any other piece of complex instrumentation.

Any attempt to suggest possible areas of application for the RSP would be a rather meaningless exercise; anywhere that the processing power of a microcomputer may be put to effective use is a potential area of application. Many of the uses may not exploit the full power of the RSP, obtainable when it is operated in the master-slave mode, but they would be, nonetheless, a possible application.

A great deal more work is required on the EEG analysis system described as an application for the RSP in this thesis. One of the major requirements is the need to perform many more tests on recorded EEG data, with some long term trials (e.g. 24hr. recordings). It may be found necessary to incorporate some small modifications into the basic program developed for this investigation, and it may also be possible to increase the execution speed; this represents an area for continuing research. It is the author's opinion that any modification should not, however, sacrifice the standards set for the performance of the basic algorithm, e.g. the second-order filter should not be replaced by some inferior configuration. Rather, it would be better to await the next generation of microprocessors, with which an improved performance may be attained, e.g. lower power consumption, improved facilities (hardware multiply/divide), easier system configuration, etc..

It is not being suggested that the method employed in the present analysis is necessarily the best or only means

of tackling the spike detection problem with a microprocessor; many other approaches may prove equally effective. The RSP is an ideal vehicle on which to develop and test different routines since, like any computer based system, its operation is determined by a program, which can be altered quite easily.

The RSP also offers the possibility of adaptive or 'learning' type programming, by virtue of the fact that the program store is contained in the read-write memory and so may be modified at run time (dynamic programming). It is more normal practice to store the program in some form of programmable read only memory (PROM) for microprocessor applications. Great care is needed in the use of a technique such as dynamic programming as the problems of debug and software reliability (see Chap. 4) can be daunting. The method does, however, offer interesting possibilities.

The major problem faced by any detection scheme is likely to be that of artifact rejection. This problem represents a research topic in itself, and will be of particular importance for the miniature version which is intended for connection to patients who will be free to move around. A relatively crude method has been proposed and tested, with some degree of success (see Chap. 7), but it is considered that a much better solution is demanded. A very important factor in the synthesis of a better rejection scheme is the capability offered by the system to operate on up to three electrode sites, in its present form (a single input can be processed in $\approx 640\mu\text{s}$ and 2ms are available between samples at 500Hz sampling frequency). Since an abnormal discharge appears over a finite area of the scalp [38] its effects will be observable at a number of adjacent electrodes. This need not be the case for artifact, and so could prove very influential in the formulation of an artifact rejection scheme.

The use of the signal from several electrode sites may

also make it possible to lower the detection threshold level used for the present analysis (see Chap. 7) and thereby reduce the possibility of missed features. This is not practical at present since a lowering of the threshold level would automatically lead to an increased false alarm rate. If several signals were available on which to perform the detection decision, this situation may be improved considerably.

More electrodes may be taken into account if the sampling frequency is reduced, but care will be required should this method be adopted, as it will necessarily increase the ω_c/ω_s ratio, which, if increased too far, could introduce significant errors into the system. The first effects will be experienced in the differentiation (differencing) operation, the errors for which can be seen from an inspection of Figure 6.10. Further, the spike detection routine is likely to be adversely affected since it will have fewer samples on which to base a decision. This disadvantage will be partially offset, though, by the increased emphasis given to the spike feature by the differencing operation.

The consideration of the artifact problem also raises another question, which requires a great deal of research, and that is the type of electrodes to be used, the method of attachment and their placement. The electrodes must be attached, but with the proviso that there must be no damage caused to the scalp, or discomfort to the patient. This is very important if the electrodes are to remain in position over long periods (e.g. 24hr. observations). The problem of electrode placement is also very complicated and it may prove beneficial to locate the epileptic focus (should one exist) by conventional means and arrange the electrodes accordingly, whenever possible.

Before the EEG study is left, two practical problems must be mentioned. The first is the need for a low-noise,

high-gain amplifier and a band-limiting filter (70Hz) to allow the system to be attached to a patient, rather than using the pre-amplified and filtered output from the FM tape recorder. The second, and most important consideration of all, is that of system integrity, and the need to isolate the patient from any injurious voltages which may be present in the system. The highest voltage employed in the RSP is likely to be 12V, with a maximum differential of $\pm 12\text{V}$ (i.e. 24V), which, although not lethal, can be very unpleasant if applied to the scalp.

Finally, the problem of processor noise must be given careful consideration. Special attention must be paid to this problem during the construction of the portable and miniature RSPs in order that this particular source of error may be reduced to a minimum.

Chapter 9

Conclusions

The ability of microprocessors, in particular the sixteen-bit devices, to deal with the very demanding problems encountered in real-time signal-processing, has been demonstrated. Further, the advantages offered by the Roving Slave Processor (RSP) approach to computer-aided measurements in general have been established.

The problems associated with the RSP technique have been identified and solutions obtained to most during the course of this research. The major remaining problem is the method to be employed in order that efficient programs may be generated easily from a high-level type of language. One method has been implemented but this offers only a partial solution.

The possibility of producing a miniature version of the RSP now exists and this should make the device suitable for many interesting applications and provide the basis for a great deal of future research.

Of equal importance to the work on the RSP are the results obtained for the real-time EEG analysis system. Many of the major problems associated with the EEG signal (e.g. its non-stationary nature) and methods commonly employed for their analysis (e.g. the running mean) have been identified. In particular, the problems encountered with frequency domain analyses were examined and it was concluded that a time domain approach would be better suited to this particular application.

The suitability of the time domain approach has been demonstrated by the algorithm employed which isolated, successfully, abnormal spikes occurring in the EEG records used as test samples, in real-time. Most of the problems associated with an automatic analysis of the EEG signal have been overcome but some further research and development work is still required in order to refine the system. The initial results are very encouraging, with a very high

detection rate (96%) established for the test records.

The most immediate problem remaining, is that of artifact rejection. A simple method has been implemented and has met with a reasonable degree of success but a much more elaborate system will be required before any attempt can be made to connect the RSP to a patient.

A further important aspect of the RSP's capabilities is demonstrated by the EEG case study and that is the 'pre-hardening of designs' facility. It would be a comparatively simple task to implement the spike detection scheme, developed and proven on the RSP, in the form of a very small, low-power, dedicated hardware device, should this be desired.

To conclude, it has been shown that the RSP, with the provision of a suitable program, is a viable proposition, capable of performing very complex computational tasks. In its present form, it could provide a very powerful link between the rapidly advancing fields of high technology and practical applications. Care will be needed in their use, however, if the basic problems which can arise with computer-aided measurement techniques are to be avoided. The research work carried out on the EEG serves as a good illustration as to the level of understanding and appreciation of the processes involved, expected of the user.

If 'standard software blocks' are to be employed for the generation of programs to run on the RSP, as proposed by the High-Level Definer system, there is a great danger that many of the fundamental signal-processing constraints will be overlooked (e.g. limit-cycle behaviour in digital filters), which may have very serious effects on any results thus obtained.

Chapter 10

Acknowledgements

The author would like to thank Professor A.J. Ellison for the provision of research facilities and the Science Research Council for financial support, by the provision of a maintenance grant.

Special thanks go to John Brignell for his guidance and many helpful discussions throughout this research project.

Many thanks also go to my colleagues, Richard Young, Colin Buffam, Pat Samwell, Terry Hewish, David Reeves, Robin Tracy and Robert Knight for their comments and assistance at various times.

I would also like to thank the technical staff of the University for their help, in particular Ron Canell for work on the mobile rack units and his many helpful suggestions.

Special thanks also go to David Lloyd and Marion Smith of St. Bartholomew's Hospital for their invaluable assistance and advice during the development of the EEG analysis routine.

My thanks also go to Chris Shelton for his valuable assistance during the past year.

In conclusion, I would like to thank my many friends who, albeit unwittingly, have contributed so much to this research, especially Gillian Wallace, Lorraine Taylor and Christine Cross, and finally my parents for their continual support and encouragement throughout my student career.

Chapter 11

Appendices

Appendix A

Publications

- i) The Roving Slave Processor
- ii) Relieving the Real-Time Processing Load
- iii) Super-tool: the microprocessor is revolutionising industry
- iv) More bits, more power, more precision
- v) Digital Filter Implementation by means of Slave Processors

The roving slave processor

Computer-aided measurement by means of a portable, dedicated processor is a new concept in instrumentation. **John Brignell, Dick Comley and Dick Young** outline the development of the roving slave processor.

The results of two years work on this development in the field of computer-aided measurement are described. The evolution of the concept, experience of building a prototype with the F100-M processor and present work on the hardware and software of fully developed systems based on the F100-L microprocessor are covered.

The cheapness and portability of increasingly powerful microprocessors are shown to lead, via hierarchical computing, to a form of activity in which the online processes are carried out by a subordinate processor totally dependent on a main computer but capable of disconnected operation. The design requirements are discussed with particular reference to existing signal processing bottlenecks, and a powerful multiprocessor format is proposed.

The implications of the software definition of instruments are investigated. It is claimed that the combination of software and hardware developments leads to signal processing methods of great potential power. Reference is made to applications such as continuous patient monitoring.

The emergence of computer-aided measurement (CAM) as a recognizable branch of applied science has its origin in both economic and technical necessity. The economic fact is that most laboratory instruments are very little used, whereas the computer, by confining specialization to software, offers generalized hardware which can be intensively used. The technical fact is that many of today's urgent problems of measurement cannot be solved without the processing power of the computer.

The idea of the roving slave processor (RSP) is an extension of the technique of CAM which also has both economic and technical origins. It comprises two concepts, one of hardware and one of software. Cheap microprocessor technology is just beginning to reach a level of processing power at which it can support some of the real-time signal processing tasks of CAM. Meanwhile, rapid advances in digital signal theory have provided the means of creating software packages which offer the instrument user unprecedented signal processing capabilities.

A typical CAM exercise is divided into three phases:

- Preparation
- Online
- Post mortem.

The computing activities in these phases differ greatly, and it is this difference which leads to the RSP idea.

Preparation consists of analyzing the measurement problem, finding a theoretical solution, preparing programs

to make the solution practicable, assembly, testing and simulation.

Online comprises the measurement phase in which the computer is engaged in data acquisition, condensation, digital filtering, transformation and provision of monitoring information, mostly in real-time.

Post mortem is an optional phase in which data are analyzed and various other offline computations are carried out according to their nature.

The second of these phases is very different in character from the other two. It needs none of the expensive peripherals or software systems they require, but in contrast it does demand a high-level of processor use. Preparation and post mortem can be carried out quite effectively on a time-shared basis, but the time element is so important in the online phase that it is essential to occupy the whole processor.

An obvious step in view of these considerations is to hive-off the online activity to a special processor which is subordinate to the main computer, that is to construct a hierarchical system. Within the master-slave relationship, the slave processor acquires its program through a suitable interface from the master computer, which has carried out the whole process of preparation, compilation and assembly.

Microprocessor technology, however, offers not only cheapness and increasing power, but also portability. The roving slave processor arises from the severing of the umbilical cord, the communication cable, between master and slave computers as soon as the RSP has received its program. It is a small dependent processor with no general purpose peripherals. Its hardware shell houses an extremely versatile instrument whose behaviour is defined by the software. The peripherals it has are used for signal processing such as clocks, digital to analogue (D/A) and analogue to digital (A/D) converters, and transient recorder².

The software implications of this development are interesting. It is possible to design and construct instruments by linking together standard software packages which represent the basic modules of instrumentation such as digital filtering, Fourier transform, signal averaging, waveform recognition and feature extraction. The potential power of the method depends on the organization of the software system.

THE RSP CONCEPT

Most of the elements of the RSP approach have existed in one form or another in computer applications. In 1976 microcomputers of sufficient processing power enabled these elements to be combined into an effective and versatile instrumentation scheme. The roles played by these elements can be illustrated by considering how a typical CAM exercise is carried out.

An essential preliminary is a thorough analysis of the measurement problem. This comprises the physical or informatory entity to be quantified and the number of impediments preventing accurate quantification. Initially some quantitative information is available so that the bounds, for example of amplitude and bandwidth, can be established. The earliest role of the master computer is in evaluating mathematical results or for modelling. This implies that high level language must be available.

When the analysis has been completed, it is necessary to seek a solution. This usually needs a method of: signal acquisition, signal processing, and delivery of information. The degree to which the solution will be imperfect is determined by the physical interpretation made. In many cases the microcomputer will not give the required performance and must use special hardware. When the microcomputer version has been designed it may be implemented, the difference being that here the computing methods are used instead of the traditional computing for instrumentation-support. Up to this point any computing service and language would suffice, but now programs such as cross-compilers, cross-assemblers and link editors are essential.

In the microprocessor implementation, the engineer is usually pre-occupied by trade-off between hardware and software sub-solutions. The RSP approach rejects the possibility of variations in the hardware. As a consequence it is accepted that applications will occur in which the RSP approach is not practical because of limits of the currently available processing power.

Ideally the implementation process consists of the collocation of standard packages representing the fundamental blocks of instrument behaviour. The result is a program residing in the store of the master computer but designed to run on the slave computer. A final, optional, stage before program transfer is the simulation of the instrument in the master computer. A good simulator with accurate timing provisions can be a very powerful piece of support software.

The roving slave processor is now plugged into a special interface of the master computer to be charged with its program by a store-to-store direct memory access (DMA) transfer. This interface is made as efficient and as fast as possible, not because the charging process needs to be carried out quickly, but because it is often useful to carry out a check on the performance of the instrument while it is working in the hierarchical mode with the master computer acting as a monitoring panel.

The basic RSP comprises a central processor, a random access memory (RAM) store and signal input/output (I/O) channels. It has no general purpose peripherals and no computer monitoring panel. Once charged with a particular program it assumes the role of the corresponding instrument and maintains it until recharged. Optionally it may return to the master computer to deliver acquired information, but but more usually it will remain connected to the test object acquiring, returning signals and displaying information. The master-slave relationship is illustrated in Figure 1.

DESIGN CONSIDERATIONS

A long term aim of this research is to develop a portable device, which could, for example, be carried on the person of a patient undergoing continuous monitoring. Hence, size, weight and power consumption are important factors in the choice of technology. At the commencement of the work in 1974 a decision was made to by-pass the then dominant

8-bit form of microprocessor and to aim for a minimum of 16-bits.

There are three aspects of word length which restrict signal processing power: the instruction set, data precision and coefficient precision.

The range and nature of the instructions available greatly influence the efficiency with which real-time processes can be carried out. A variety of addressing modes contributes to the optimization of programs for speed. Bit-testing and bit-setting instructions allow the use of fast flag-based routines in critical program areas. A range of shift orders (arithmetic, logical, double length and cyclic) can also contribute to time saving by speeding up such essential operations as scaling and packing. Alternative forms of jump (link storing, bit-testing, switch, relative, absolute and incremental testing) add to the possibilities of efficient program design. Each of these types of instruction, in addition to the ordinary arithmetic and logical ones, helps to reduce both execution time and stored program length. Eight bits is simply not sufficient to span the range of alternatives.

It is not unusual for an instrumentational precision to be aimed at one part in 10^3 , that is 10 bits of information. Furthermore, if precision is not to be lost through changes of effective length in intermediate calculation, such as multiplying by a coefficient, the capacity must be greater than this, and in practice 16 bits is not generous.

A factor which is often underestimated is the precision required for stored coefficients. A digital filter, for example, is made up in the form of sums of delayed input and output values multiplied by appropriate coefficients. The quality of the filter is determined by the accuracy with which its poles and zeros can be placed. Usually these will be close to the stability boundary (the unit circle in the z domain or the imaginary axis in the s domain^{1,2,3}) and their placement

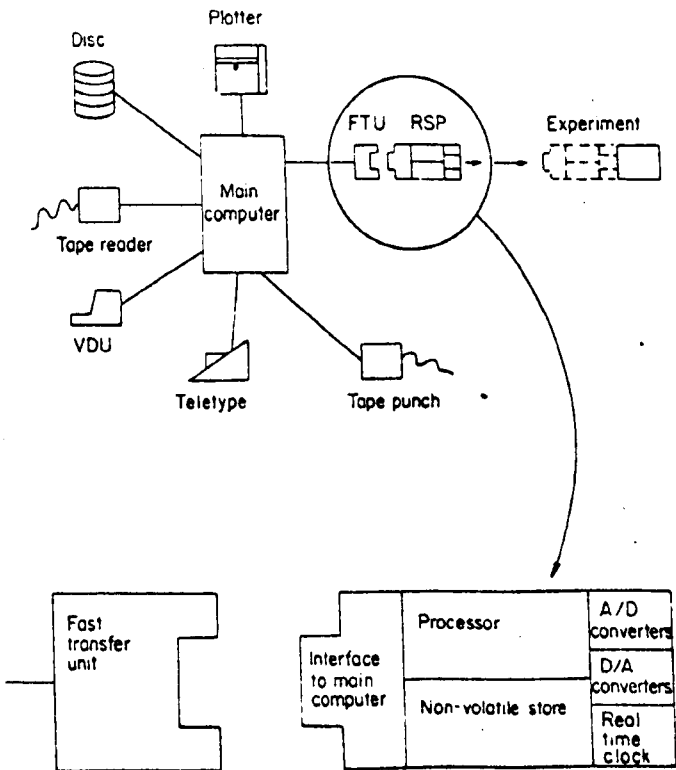


Figure 1. Roving slave processor to laboratory online computer for stages 1 and 3—dotted line position shows connection to experiment for stage 2

is controlled by the coefficients. 16 bits is barely adequate: eight is almost useless.

Taking these and other operational considerations into account an ideal processor will have the following features:

- 16-bit data and fully utilized instruction word
- Simple DMA facility
- Comprehensive vectored interrupt system with priority
- Low power from single supply
- Single clock input
- Internal working registers
- Low implied external component count
- Integrated system philosophy with multiprocessor capability
- Internal multiply and divide.

We were fortunate to have early access to information on the proposed F100-L device which exhibits many of these features although it has no internal multiply/divide and only one internal register. This was used in the development work to replace the F100-M in the first processor that was built using the medium scale integration (MSI) form.

The first 16-bit chip available to us was the CP1600 which has many good features but at present only has a 10-bit wide instruction set, although well geared to real-time use, and a second prototype RSP was constructed incorporating this in the form of the GIMINI micro-computer, see Figure 2.

The TMS9900 is attractive for the partial multiply/divide in hardware, but has compensating disadvantages. We have not yet explored its practical possibilities. Also worth considering is the IM6100, with only a 12-bit word, but, being complementary metal oxide semiconductor (CMOS), is attractive for low power applications.

Bit slice processors offer great signal processing capability, but were considered too expensive in power, size and development effort.

At present, in terms of an overall system for this application, the F100-L appears to be marginally superior, and it is the basis of the ongoing development work. Work is continuing on the CP1600 prototype, however, and future developments may enhance its capabilities.

Software must not be neglected in the design. The assembly language should be powerful but bear a close relationship with the internal operation of the processor. The CP1600 is particularly good in this respect. The F100-L has an extensive cross-assembler and link editor, and, particularly important, a comprehensive simulator with detailed timing facilities.

The RSP is a tool for engineers and it is a matter of some regret that the trend away from algebraic towards literal mnemonics has continued. It is easy to demonstrate that engineers and other numerate professionals find the algebraic variety more scrutable. We have developed an algebraic cross-assembler and de-assembler for the F100-L which has proved a useful practical and descriptive tool.

PROTOTYPES DEVELOPED

This development work has been planned in three stages, the second of which is underway. The first stage was the construction of transportable prototype RSPs. In these no attempt has been made to miniaturize or conserve power: in fact, they have deliberately been made large and accessible and take the form of 482.6 mm (19 inch) rack crates on castors. The CP1600 prototype has been used as a test

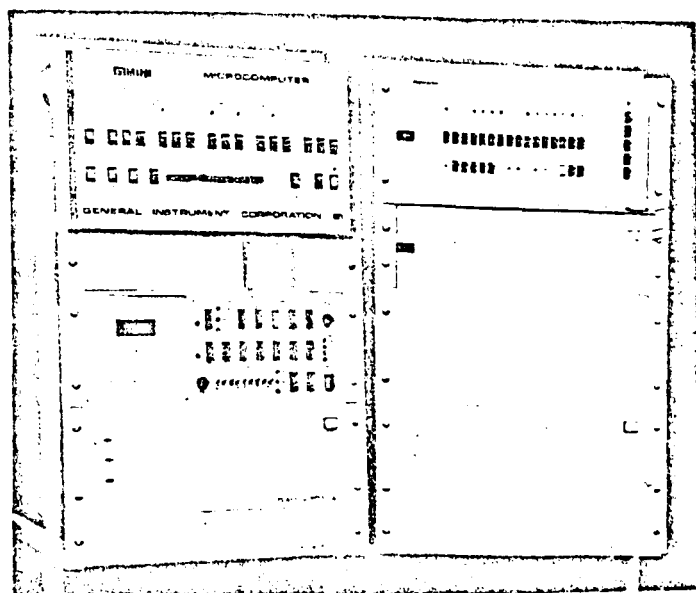


Figure 2. The two prototypes: on the left, CP1600 version carrying cassette and paper tape readers; on the right, F100 version in pure RSP form

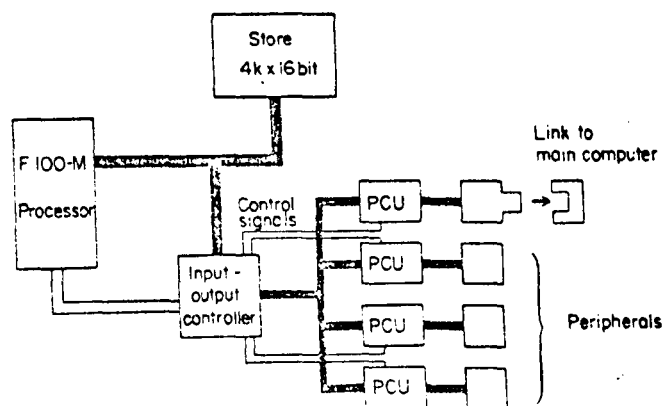


Figure 3. F100-M system with four multiplexed peripherals

bed for alternative methods of overcoming the volatility problem. It has, for example, been equipped with a cassette recorder onto which the current store contents can be copied under battery back-up supply as soon as a power-fail interrupt is received.

The F100-M prototype was begun 18 months before the F100-L processor and its important interface chips were available. In this computer the equivalent of the F100-L chip is a five board MSI processor and I/O is handled by a two board controller. The processor is functionally identical to the large scale integration (LSI) version, but the controller is not as effective as the LSI interface set, which has led to some problems.

The MSI prototype is shown in Figure 3. In the RSP, peripherals are not standard computer types (paper tape, teletype, etc.) but rather signal types (A/D converter, D/A converter, clock) together with certain development monitoring facilities, such as a light emitting diode (LED) to indicate when sampling is too fast for the current program loop, see Figure 4. The master computer, a Ferranti FM1600B, also takes on the status of a peripheral and is connected to the I/O controller via a special detachable interface and a peripheral control unit (PCU).

The second stage is the development of a portable version, and the experience gained with the prototype is being incorporated into a system based on the F100-L and its interface set. This experience has emphasized the two bottlenecks which greatly restrict processing capabilities. One is input/output the other is multiply/divide. As far as the second of these is concerned the saving grace of the F100-L is that a special processor concept is built into its system philosophy. A special external function instruction enables control to be passed to a special processing unit (SPU) connected to the F100-L bus and capable of operating directly on the store. The achievement of signal processing speeds in such activities as digital filtering depends on a multiply/divide processor chip becoming available. Figure 5 shows a prototype RSP based on the F100-L. It will be noted that all devices are connected to the bus via the versatile interface set (IS) of chips and that priority in interrupt and DMA operation is achieved by *daisy chain* connection of the control lines. The memory store developed so far for this version is based on n-type metal oxide semiconductor (NMOS) devices for which the refresh is maintained by battery back-up following a shut-down routine which is initiated by a power-fail interrupt. Obviously static CMOS devices promise a better solution, but at present they suffer from a packing density disadvantage.

The input/output bottleneck requires rather more thought, and once again it is the system philosophy of the F100-L which suggests an answer. This processor is particularly suitable, through the versatility of its interface set, for multiprocessor systems, and it is feasible to provide a separate processor to service the input/output devices. A proposed multiprocessor scheme is shown in Figure 6. The I/O processor services all interrupts and DMA requests from the PCUs according to a program held in its secondary store. Only data for direct calculation are passed to the main store by back-to-back interface sets connecting the primary and secondary buses. The main and special processor units are concerned solely with executing rapidly the calculation loops of the program and they can only be interrupted by the I/O processor. The net result is that signal processing throughput can be considerably increased, though at the expense of an increased buffering delay.

The third stage of the hardware development, which is in the future, is the production of a miniature RSP, which

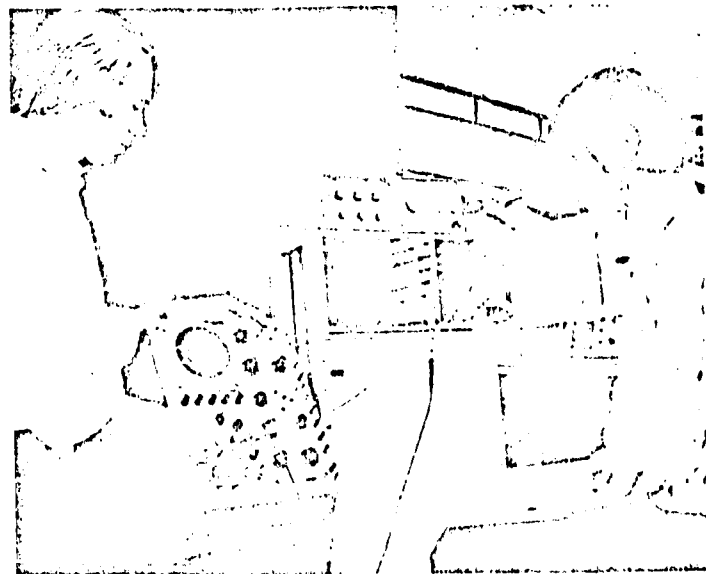


Figure 4. Testing the converter board with a single-pole low-pass digital filter in operation

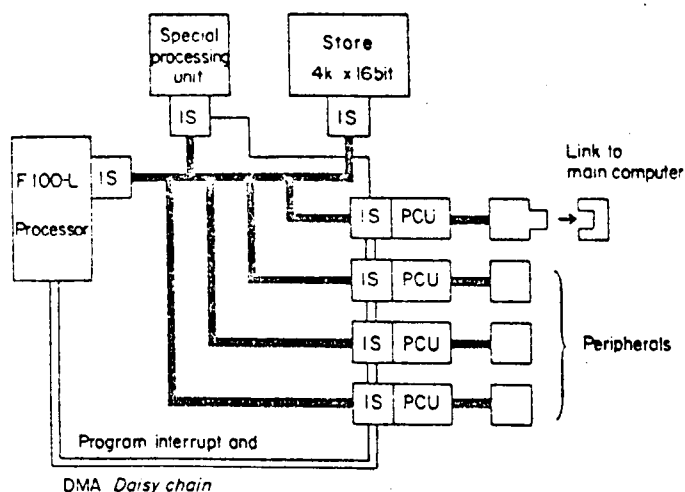


Figure 5. F100-L system with special processor and four peripherals

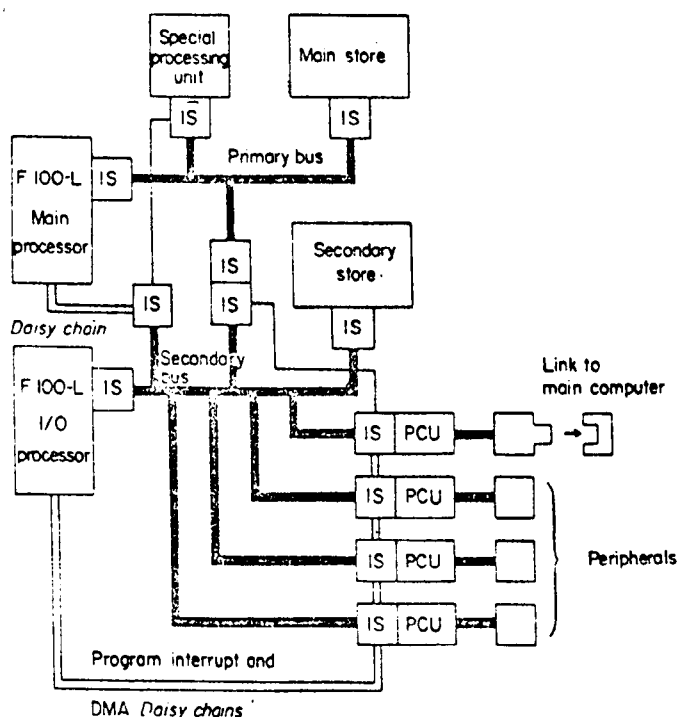


Figure 6. Multiprocessor RSP with special processing unit and second F100-L as I/O processor

could, for example be carried on the person of a patient undergoing continuous monitoring. It would need to be cheap enough for a large number of RSPs to be serviced by one master computer, so that they would be available as tailor made instruments for a variety of applications.

DISCUSSION

Although we have concentrated on the hardware aspects of the RSP, it is the software design which will determine the applicability of the concept. It should be possible to devise a scheme where the instrument user can nominate standard software packages and concatenate them appropriately by typing in relatively simple high-level instructions to the master computer. For example to introduce a low pass digital filter he would specify such factors as: the cut-off

frequency, as a multiple of the sampling frequency; the maximum errors in the pass and stop bands; and the source and destination of the data. The computer would send back warnings about such things as the execution time of the resulting program and generate it as a block suitable for link-editing into the overall program to run on the RSP. This is quite feasible, but it implies a great deal of software development effort.

There are also dangers in such an approach. A simple example is provided by the sampling theorem. If the sampling rate is inconsistent with the bandwidth of the incoming signals, the RSP will carry on processing, unaware that the data are corrupted by aliasing, so it is not a device to be used by the ill-informed without expert support. Similar remarks apply to understanding the limitations of transducers.

One of the most promising areas of application is for tailor-made instruments. In electroencephalograph (EEG) measurements for example there is a waveform which is a characteristic precursor to an epileptic fit. Unfortunately this waveform varies from individual to individual, but RSP techniques would permit the creation of a personalized instrument to give warning in the case of one particular patient.

Another important area is in the testing of 'pre-hardened' instruments. The RSP can be taken out into the field to test several successive forms of an instrument, which when pronounced satisfactory is ready for multiple production with the program hardened in ROM.

The third main area of application is for special instruments which are required urgently but only for a short time, a common occurrence in fundamental research. Fourthly, the RSP can be used to give certain important peripherals a roving commission. An example is the online transient recorder which can be used to solve several difficult problems which occur in the field, such as the capture and recording of acoustic emission waveforms from microcracks in high pressure gas vessels.

CONCLUSIONS

Computer-aided measurement has tended to take the form of *ad hoc* solutions to specific problems. There is a need for a more general approach which will provide insight and foster a cumulative growth of technique. The work described is the result of an attempt to apply such an approach with the incorporation of economic and technological considerations. The RSP concept appears to have a wide range of potential application. Whether this is achieved in practice will depend on the development of a software background. What is particularly required is an emphasis on software for signal processing, whereas most effort in the recent past seems to have been concentrated on the less difficult, though more complicated, organizational aspects of software.

REFERENCES

- 1 Brignell J E and Rhodes G M, *Laboratory online computing*, Intertext, London (1975)
- 2 Rosner R A, Penney B K, Clout P N, editors, *Online computer in laboratory use*, Advance Publications, London (1975)
- 3 Gold B and Rader C M, *Digital processing of signals*, McGraw-Hill (1969)

Relieving the Real-Time Signal Processing Load

R.A. Comley

Abstract: Real-time signal processing places large demands on central processor time, a situation which frequently leads to the inefficient use of computer installations. This paper outlines a novel method of overcoming this problem by hiving-off the real-time activity to a small processing unit. The system is hierarchical in nature but the unit may be detached and operated autonomously while remaining totally dependent upon the master computer for all of its fundamental needs. Hence, the device may be taken to the test object, rather than vice-versa, and dedicated to a particular application for as long as required.

The hardware and software considerations of such an approach are discussed.

Introduction

In medicine, as in other scientific fields, a great deal of time and effort is expended in signal processing tasks. Much of this work involves real-time computations and, as a result, can often lead to the inefficient use of a computer installation. If full benefit is to be obtained from the computer and its expensive peripherals their available time must be used effectively.

In order to allow as many users access to the machine as possible some form of time-sharing is usually adopted. This method is perfectly adequate for most tasks but may be virtually useless for real-time work. For some applications the time element is of such importance that the processing task demands sole occupancy of the Central Processing Unit (CPU) if a solution is to be obtained. Obviously, this will result in very inefficient use of the computer and its facilities, as they may be committed for a considerable period to an application which does not require the use of most of the peripherals.

This situation is further aggravated in the medical field by the need for 'personalised instruments'. In many situations the quantity to be monitored may be similar but the characteristics may differ greatly from patient to patient, i.e. in the EEG waveform the spike-and-wave

precursor to an epileptic attack. This means that the entire computer must be dedicated to an individual patient for as long as is necessary to perform the required measurement.

One solution to these problems is the purchase of a mini-computer which may spend at least some of its time completely dedicated to one particular task, but these still tend to be fairly costly items, as do the rapidly emerging micro-computers (indeed, a usable micro-computer system will often cost as much as a mini-system).

In many situations this high CPU demand may exist for a short period only, or may arise in short bursts, throughout the course of the measurement. Hence, another solution is the addition of a pre-processing unit to relieve the main computer from some of the more mundane tasks while maintaining the ability to grant the real-time problem exclusive use of the CPU, when required, by means of a high-speed interrupt system.

Both of these approaches proffer a solution to the real-time processing problem. A third possibility exists, however, which combines both of the above ideas in one unit and furthermore extends their effectiveness. A closer inspection of a typical real-time computer aided measurement (CAM) problem points the way to this solution as outlined in the following paragraphs.

Stages in CAM

Typically, the solution of a CAM problem can be divided into three stages:-

- i) Preparation
- ii) Measurement
- iii) Post Mortem

The first stage, preparation, consists of defining and analysing the problem and then preparing a means of solving this problem; usually this entails the preparation of a program but may also require the construction of some small piece of hardware.

The next stage is the actual measurement phase for which the computer must be connected to the test object, be it a piece of machinery or a patient, and the programs prepared in stage (i) are used, in real-time, to monitor various parameters and/or control some action.

The final stage involves taking the results obtained in stage (ii) and performing further processing upon them, to provide, for example, some permanent record of the processed data, e.g. a distribution plot. This is an optional stage and may be omitted if not required.

A careful consideration of these three stages shows stage (ii)

to be very different in nature from the other two in that it does not require the use of any of the expensive peripherals or operating systems required by the other two stages but it does demand a high level of CPU time.

Stages (i) and (iii) are not time critical and so may be carried out quite adequately on a time-share basis, but stage (ii) is often so time dependent that it may not.

The obvious next step from here is to 'hive-off' this second phase, in some way, to a small processor which may be dedicated to the real-time task for as long as required and then made to report its findings, or results, at the end of the experiment.

There is nothing really new in the solution so far as this is the basis of many hierarchical systems currently in use. What is proposed in this paper, however, is to take this solution one stage further, i.e. make the hierarchical machine detachable from the master computer and capable of autonomous operation whilst remaining totally dependent upon the master for all of its fundamental needs.

The Roving Slave Processor

The proposed device has as its basis a sixteen-bit microprocessor and a small associated store. It has no general peripherals of its own, not even a front-panel (save for a run/stop switch) and is totally dependent upon the host machine for the provision of all such facilities when they are required.

In contrast, it will be equipped with special signal processing type peripherals (e.g. an ADC/DAC channel) by the addition of boards extra to the basic system, while even more specialised boards may be added if required for some particular task.

This basic hardware unit is termed the Roving Slave Processor, or RSP.

Using the RSP

A typical CAM operation will now proceed as follows:-

The problem is analysed, as before, and programs prepared on the master computer. These are then assembled into a machine language suitable for the RSP via a cross-assembler. At this stage a simulator is a very useful and powerful piece of software support to help 'debug' the program, though it is possible to proceed without one.

The RSP is now connected to the host computer via some suitable link. This could be some general input/output port (e.g. a modified reader/punch or teletype channel) or a specially constructed channel.

The latter solution is preferable as a high speed link can help if the RSP is to be tested in a hierarchical mode. This is useful as a final check can be made on the program whilst it is actually residing and running in the RSP with the host machine controlling and monitoring the activities, for example to single-step the RSP and display the results on its own front-panel.

The assembled machine code is transferred via the link to the RSP's store by means of a DMA (direct memory access) transfer, and, once it appears to be operating correctly, the RSP may be unplugged, disconnected from the mains supply and transported to its required site for operation. The use of a microprocessor based system simplifies this stage considerably as the RSP can be made very small and lightweight. One major problem, however, is the need for a non-volatile store in order to maintain the stored program during transit. Core-store is an obvious choice but it was rejected for this application as it was considered too bulky and that it would add considerably to the power supply requirements of the system. Therefore, a semi-conductor store has been selected, and a battery is used to maintain the supply whilst mains power is removed. No attempt is made to maintain the whole system via this back-up supply: indeed, all non-essential circuits are deliberately switched off during the standby period leaving only the store with a power supply. The instrument in the present state of technology is not intended to run from a battery supply but from the mains.

Upon reaching its destination the RSP may be reconnected to the mains and an ordered start-up procedure performed.

The RSP, now ready for use, may be connected to the test object (i.e. the patient) and the program executed. Once it has completed its task it may be halted, unplugged from the mains and transported back to the host computer. Here, it is reconnected to the link to dump its data via another DMA operation leaving the host machine to proceed with the post mortem stage, if this is required. The RSP is now ready to be reloaded with another program to perform some other real-time task.

Hence, it has now become possible to take the main computer, via the RSP, to the patient rather than vice-versa, which is usually the case at present. It is in this manner that the RSP extends the benefits obtainable from a straightforward hierarchical system, and provides a simple means to construct personalised instruments.

Conclusion

As can be seen from the above discussion the RSP is a very basic

'hardware shell' which may, via a suitable program, take on the guise of any desired instrument. It now becomes possible to describe an instrument as a concatenation of various standard software packages called from a teletype (or similar) - e.g. for a low-pass digital filter the operator would merely type:-

Type:	L.P.
Class:	CH (Chebychev)
Order:	2nd
wc/ws:	0.1 (wc = cut-off frequency)
	(ws = sampling frequency)

Once all of the required routines, plus any peculiar to the task in hand, have been assembled and linked together they may be dumped to the RSP ready for use. In this manner, with correctly organised software, the device becomes very simple to use. Some signal processing knowledge is expected of the user, however, if such basic problems as signal aliasing, for example, are to be avoided.

A further advantage offered by the RSP approach is that, since the hardware is relatively cheap and simple to produce, a great many RSPs could be serviced by one central computer. In this way the effective real-time capability of the main machine is greatly increased, each user being able to dedicate a CPU to his or her problem for a great length of time, if necessary.

Acknowledgments

The author would like to thank Dr. J.E. Brignell for all his help and useful advice and his co-researcher in this project, Mr. R. Young. The author would also like to acknowledge the support of the Science Research Council for the provision of a maintenance grant.

References

1. Brignell J.E., Comley R.A. and Young R., The Roving Slave Processor, Microprocessors, Vol. 1, No. 2, IPC Science and Technology Press, Dec. 1976.
2. Brignell J.E. and Rhodes G.M., Laboratory On-line Computing, Intertext, London, 1975.
3. Proceedings of the IEEE, numerous, e.g. Gevins A.S. et al, Automated Analysis of the Electrical Activity of the Human Brain (EEG): A Progress Report, Vol. 63, No. 10, Oct. 1975.

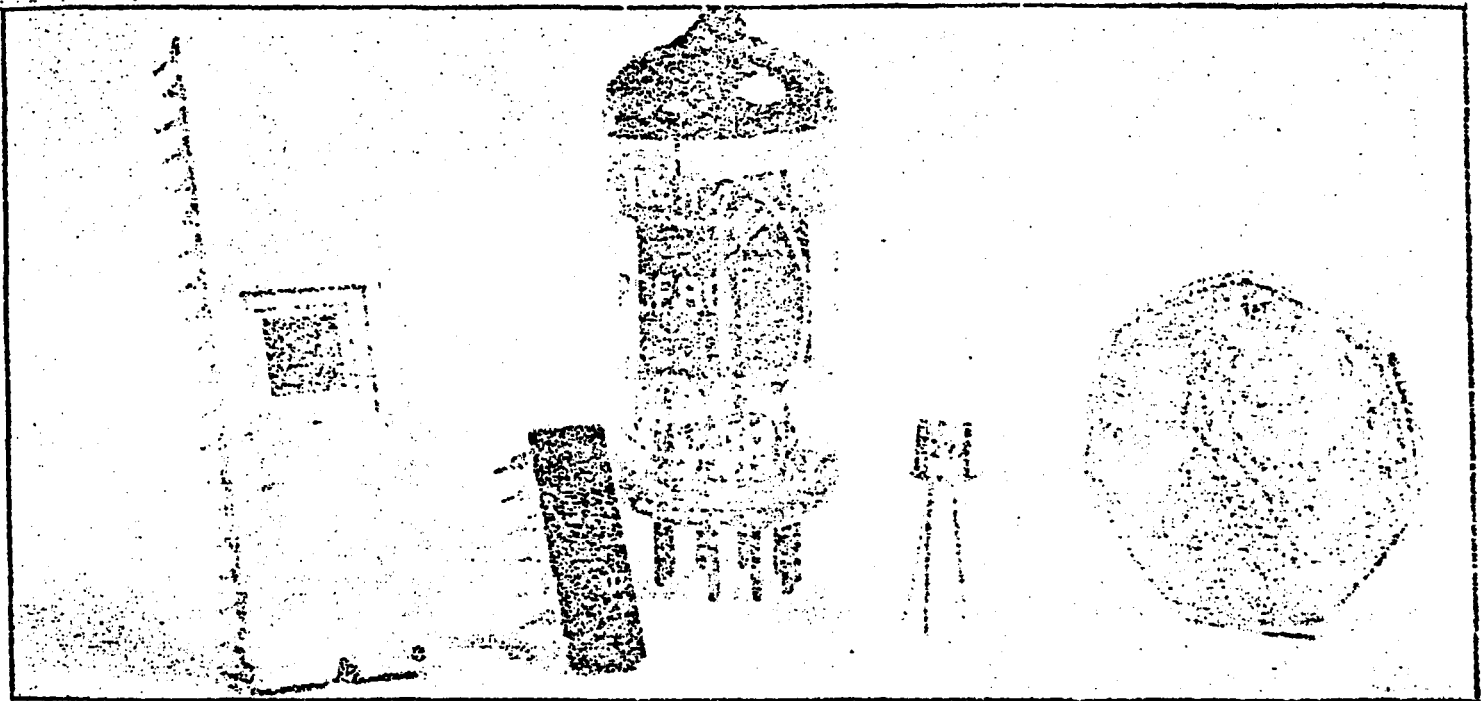


Fig. 1 illustrates the high degree of miniaturisation of the F100L. L to R: F100L, small-scale integrated circuit, valve, single transistor.

Super-tool: the microprocessor is revolutionising industry

The crucial importance of the microprocessor to Britain's future has been underlined by three recent Government moves: the Department of Industry is to make £70m. available to encourage the use of microprocessor technology, the National Enterprise Board is planning an investment of £50m. to build British microprocessor groups into a competitive international industry, and the Central Policy Review Staff (the 'Think Tank') is to study the social implications of this new super-tool. In this article R. A. COMLEY Research Fellow in the Department of Electrical and Electronic Engineering, explains the nature of the microprocessor revolution and describes research at The City University into aspects of its application.

The latest revolution in electronics, which plays an increasing role, both industrial and social, in our society, can be summed up in one word: microprocessor.

The microprocessor is the newest tool available to the computer engineer, whose contribution has had such a profound effect on life in the second half of the 20th century, from the automation of factory and office work to man's landing on the moon.

To trace the microprocessor's origins it is necessary to take a historical view of the electronics industry in general, paying particular attention to the semiconductor industry.

The semiconductor industry is relatively new, having its roots in the early Fifties with the introduction of the first transistors, which made the pocket-size radio pos-

sible, among many other things. The transistor was the starting point for a great research drive that is as intense today as it was two decades ago.

The manufacture of a transistor relies on the diffusion of different regions on to a very small piece (or chip) of silicon or germanium. These two elements are known as semiconductor materials because of their electrical characteristics and, together, they form the basis of the entire semiconductor industry. The methods involved in the diffusion process were refined throughout the Fifties with the result that by the early Sixties several transistors and a few passive components (e.g. resistors) could be diffused on to a single piece of silicon. The first integrated circuits had arrived.

These devices have found

most use in the digital electronics field where 'standard building blocks' are required for linking together in various configurations to perform some logical operation. The standard building blocks required in such applications are manufactured in the form of integrated circuits.

Typically, these early devices contained about 30 components in a single package measuring approximately 20 mm by 6 mm ($\frac{3}{4}$ in $\times$ $\frac{1}{4}$ in). They are still in great demand today, and their cost is now equivalent to that of a single transistor, the major expense now being one of packaging and handling.

This level of technology is known as small-scale integration and from it came medium-scale integration in which up to several hundred components may be incorporated on a single chip.

The processes were further refined and packing densities increased and by the end of the Sixties the first large-scale integrated circuits appeared. These were generally memory devices in which numerous functionally identical units are required. The first devices contained some

1,500 components and this was quickly increased to densities of more than 10,000 components for the devices produced today (Fig. 1).

In 1972 integrated circuit

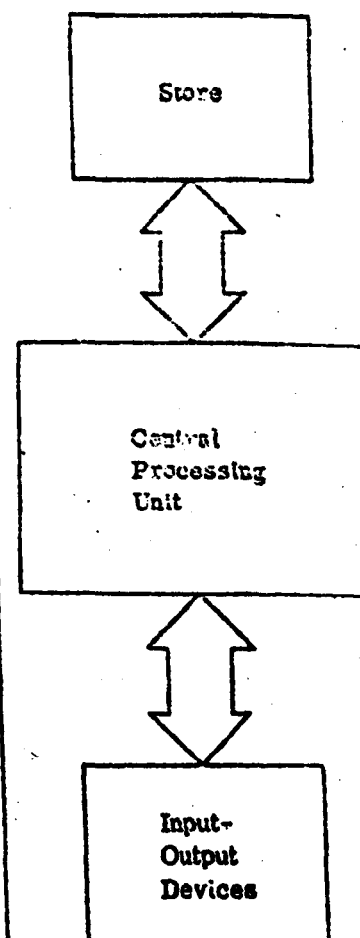


Fig. 2: The basic components of a computer system.

technology had advanced to such a stage that a company called Intel were able to introduce the first microprocessor to the market. This was a fairly basic device, but in the space of only six years it has completely revolutionised the electronics industry. Today's microprocessors, containing some 7,000-10,000 components, are vastly superior to this first device, but a start had been made and other manufacturers and users were quick to respond to this latest innovation.

What is a microprocessor? To answer this question and to understand how a microprocessor works we must look at more conventional processors, or computers as they are more generally known. All digital computers consist of three basic units, as shown in Fig. 2. The central processing unit (CPU) is the part most people think of as being the computer. This is where all the processing, or computation, takes place, e.g. value calculation, decision making, logical operations etc. To control these operations a set of commands, or instructions, are required and these are held in the store. The set of commands is known as the program.

A section of the CPU is set aside for fetching these commands from the store in a logical sequence, decoding them (i.e. deciding what a pattern of voltage levels from the store means) and then setting up the rest of the central processor to perform the requested operation. When one operation has been completed the next instruction is read from the store and in this manner the program is executed.

In addition to the store and CPU some means of communication with the outside world is required—it is of little use having a computer if you cannot ask it to do something and, when it has done it, find out the results. The communication channels are grouped together as input-output devices.

The input-output devices (or peripherals) attached to a computer are many and varied; indeed a whole section of the computer industry exists solely to manufac-

ture and sell such units. In general a means of loading a program into the computer and a method of obtaining the results are required. For loading purposes a card reader is generally used, in which small holes punched in cards are 'read', and the results are usually printed to provide a permanent record.

A microprocessor is the computer's central processing unit, integrated on a single piece of silicon. It is *not* a complete computer on its own—it still needs a program store and some



Fig. 3: A typical design problem.

means of input-output, as does a conventional computer.

Microprocessors operate in exactly the same way as conventional processors in that they fetch a command from their program store, decode the instruction and then perform some appropriate logical action.

To help understand how these new devices may be used, consider the example shown in Fig. 3. A system has three inputs, A, B and C and is required to generate two outputs, X and Y, whose values are to be determined by the state of the three inputs. The inputs and outputs can be considered as switches which may be either on or off.

The electronics engineer now has the choice of either building a circuit to decode the inputs and generate appropriate outputs, or to write a program for a microprocessor which reads the three input values, decodes them and then generates an appropriate output. In this manner a microprocessor can be used as a direct replacement for more conventional circuits, and indeed this is an area where a great number of microprocessor applications exist.

Typical current applications include machine control such as automatic drilling jigs, photocopiers, petrol pumps and the like, and in domestic appliances such as automatic washing machines, sewing machines etc. A

great number of microprocessors are also being incorporated into entertainment equipment, such as TV games, fruit machines, chess games and so on. Although these amusement applications are less important than the machine control examples, from a practical point of view, they generally present more demanding programming problems. A great deal of complicated processing and calculation being required.

Note that for these 'dedicated' types of application, where the microprocessor is required to perform only one task, the input-output device requirements are minimal. The program need be loaded only once, which may be done during the manufacture of the store, and the output may be taken directly from the output of the microprocessor, in digital form.

Microprocessors offer not only economical solutions to the types of problems mentioned, but also provide 'flexible' systems. For example, consider again the problem given in Fig. 3; if a change in the relationship between input and corresponding output states is required it is a much simpler task to amend the control program for a microprocessor-based system than it would be to redesign and rebuild a more conventional circuit. This can be a very important consideration for the drilling jig application, for example, where to reset the machine for a different job all that is required is a new program. These could both be included in the same store and a single switch, read by the microprocessor, could determine which program is to be executed.

A microprocessor, however, is capable of much greater things than simple circuit replacement: it is

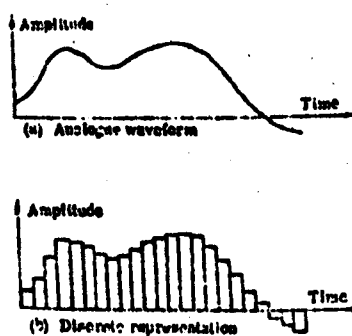


Fig. 4: Analogue and discrete representation of a signal.

after all the central processor of a computer. This is where our interest in microprocessors lies. The aim of our research at The City University is to use them to help solve signal processing problems. This basically involves reading a continuously varying signal from some object under test and using a microprocessor to analyse it. The signals we are dealing with in the real world are known as analogue signals, i.e. they are continuous in time. A continuous signal must be converted to a discrete form before a microprocessor can deal with it. This involves dividing the

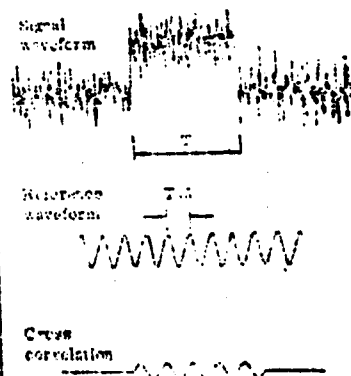


Fig. 5: Detection of the fifth harmonic of a square wave using cross-correlation (from Signal Processing, Beauchamp K.G.).

signal into a series of discrete samples, each of which is then represented as a digital pattern to the microprocessor (Fig. 4). This technique is known as analogue-to-digital conversion.

Once the signal is in a digitised form, the microprocessor can be programmed to perform various mathematical operations and tests on the data: for example, by the use of a technique known as 'cross-correlation' a signal which is not apparent to the human eye can be detected in what appears to be a very noisy waveform (Fig. 5). This is just one of the many mathematical techniques available to the signal processing engineer, but the use of a computer is required to perform the necessary analysis (2), (3).

Many potential applications for these techniques exist at remote or relatively inaccessible points. This effectively rules out the use of conventional computing systems as they tend to be large

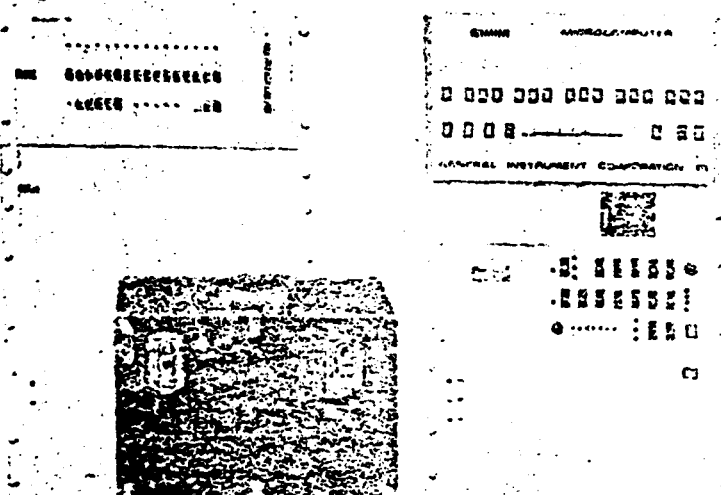


Fig. 6: When miniaturisation of the Roving Slave Processor is completed the equipment on the two panels will fit into the case in the foreground.

and bulky arrangements. With the use of microprocessors, however, the possibility exists of making a small lightweight computer system which may, quite easily, be transported to some remote site to perform measurements. This has been the main theme of our research in the Electrical and Electronic Engineering Department at The City University. Work began almost three years ago on a project to produce a light, portable computer which we named the Roving Slave Processor or RSP (1).

Two prototype systems have been produced and are currently undergoing tests. Work is about to begin on the production of a truly portable version, to be housed in the small case shown in the foreground of Fig. 6. The RSP is a very basic computer system consisting of the three basic functional units shown in Fig. 2. Special input-output devices are generally required for specific applications, but no general purpose peripherals are included (e.g. card read-

ers, printers etc.); instead the RSP receives its program from a large computer system via a special link.

Once loaded with a program the RSP may be disconnected from the main computer and transported to some remote site for connection to the system under test. When the required measurements have been performed, and any necessary data collected, the RSP may return to the main computer and be reconnected to deliver its results. These results may then receive further processing, if desired, or simply be displayed in some appropriate form.

In this manner, the RSP has the use of all the general-purpose peripherals attached to the main computer without having any of its own. Further, numerous RSPs may be serviced by one master computer, which makes very economical use of central resources.

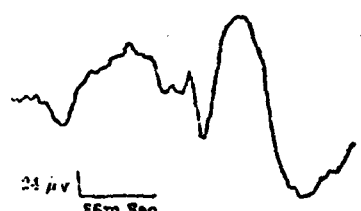
The RSP idea is a very powerful one in that it provides the engineer and scientist with a general-purpose instrument which may have

its mode of operation defined by the provision of a suitable program. With the use of the special link to the main computer it is not necessary to change any of the circuitry contained in the RSP to re-program it, but, as mentioned earlier, special-purpose boards will generally be required for specific applications. For example, a circuit to perform the analogue-to-digital conversion described in Fig. 4 will be required for most signal processing applications.

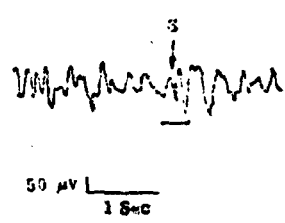
The author's main area of interest, apart from the development of the RSP idea, is in the exploitation of the RSP in the medical research field, in particular for the analysis of the electrical activity of the human brain. It is possible, with electrodes placed at suitable points on the surface of the scalp, to detect and monitor this activity—the principle employed by the electroencephalogram (EEG machine) for many years. The purpose of this research is to provide a means of automatically detecting the onset of an epileptic fit.

Epileptic attacks are generally characterised by a distinctive waveform in the EEG record known as the 'spike and wave' complex, which occurs before an attack. It is given this name as this is the physical appearance the precursor has in the EEG recording, comprising a sharp spike followed by a slow wave (approximately 0.3–1.0 Hz), Fig. 8(a). The spike and wave pattern, in itself, is not too difficult to detect but the problem is greatly complicated by the sheer quantity of data to be analysed, and the fact that the required waveform is hidden in what, to the untrained eye, appears to be a random waveform, Fig. 8(b). The problem is further aggravated by the fact that any movement of the patient's scalp causes a considerable disruption of the EEG record, with numerous sharp spikes being generated. The extraneous signal, or 'artifact' as it is known, presents one of the major problems to be overcome by any system designed to monitor the activity of the brain.

A major advantage offered by the use of an RSP type of



(a) Typical spike and wave



(b) Spike and wave as it appears in EEG record

Fig. 8: EEG trace containing an epileptic precursor.

device for this application is that it may be taken to the patient's bedside, rather than requiring the patient to go to the computer, as happens at present. Another consideration is our hope that future developments in integrated circuit technology will make it possible to reduce the dimensions of the RSP still further, to about the size of a present-day calculator. It will then be practicable to attach the RSP to the patient (for example, it may be carried in a pocket) thereby enabling the continuous monitoring of various biological waveforms in as near a normal environment as possible.

Where next? The introduction of the microprocessor as a basic building block has brought about revolutionary changes in the electronics industry and other allied fields. The pace of the semiconductor industry is such, however, that already, after only six years, the days of the microprocessor appear to be numbered.

The first microcomputers were introduced last year and these incorporate all of the essential elements of a computer system (Fig. 2) in a single package of the same size as current microprocessors. Originally microcomputers were fairly costly items, as were microprocessors when first introduced, but their price has fallen quickly. This is making them the obvious choice for many of the applications at present using microprocessor-based systems. For some applications all that may be required is a single microcomputer

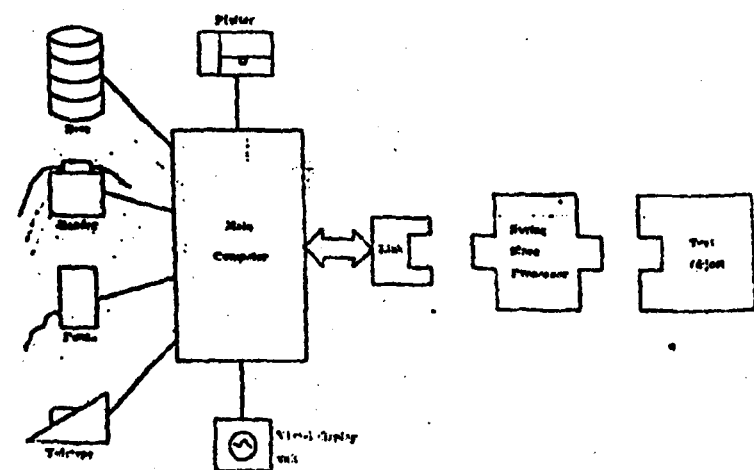


Fig. 7: Overall view of Roving Slave Processor in use.

package!

The dream of a pocket-size RSP may be closer to fruition than we think.

Acknowledgements

The author wishes to acknowledge the contribution of his co-researcher, Mr R Young, to the RSP project and the help and advice given by Dr J. E. Brignell. The author also wishes to acknowledge the support of the Science Research Council by the provision of a maintenance grant.

References

- ¹ Brignell J. E., Comley R. A. and Young R.: The Roving Slave Processor, Microprocessors, *IPC Science and Technology Press*, vol 1, no 2, December 1976.
- ² Beauchamp K. G.: *Signal Processing*, George Allen and Unwin Ltd, London, 1973.
- ³ Brignell J. E. and Rhodes G. M.: *Laboratory Online Computing*, Intetext, London, 1975.

More bits, more power, more precision

C. BUFFAM, R. COMLEY and R. YOUNG*

It is dangerous to exaggerate the analogy between the history of the microprocessor and the minicomputer. One aspect which does however appear to be repeating itself is the enormous pressure on the potential user to accept the status quo or Industry Standard as it is becoming known. While the ubiquitous 8-bit micro is more than adequate to deal with most applications, it is not the panacea which some would have us believe. Some users, through fear of using a sledge hammer to crack a nut, are finding themselves lured into using a nut-cracker to break a rock. In examining the growing usefulness of 16-bit devices, the authors describe the importance of assessing the implications of word length at the outset of a project, in order to arrive at a rational choice of standard.

The word is the packet in which information is passed around a system, and its length has two important influences, corresponding to the two possible interpretations of a word. Firstly, a word may represent data, and in the eight-bit case the representation is very coarse since only 256 different values may be accommodated, while a 16-bit word can take on 65 536 different values. Secondly, the word may represent an instruction, and the same arithmetic applies.

The potential instruction set in a 16-bit word is 256 times bigger than that in an 8-bit word. Of course in either case a multi-word approach may be used, but this is another way of saying that speed may be traded-off against precision.

A typical precision for instrumentation purposes is 0.1% or one part in 10³. This is ten bits of information, and in practice it is found that coping with overflow and underflow in intermediate calculations makes even 16 bits a bare minimum. When eight-bit processors are used for such applications it is conventional to use BCD arithmetic of variable length, but this method is only effective when plenty of time is available.

The sort of area in which a large instruction set is appreciated is in high performance real-time systems. Here such operations as shifts, bit-tests, byte-swapping, auto-increment, etc. can, if available in sufficient variety, each replace long sequences of instructions necessary in a simple processor. If they are coupled with a good priority-assessing interrupt and DMA system the gains are even greater.

Another consequence of having more space available is that the 16-bit instruction set is not only bigger, but it is better structured and more logical. Most 8-bit processors reveal strange omissions and inclusions in their lists of instructions, certainly more so than their bigger brothers. This makes them more difficult to program efficiently, and particularly to debug. It is not always appreciated that system debug is one of the biggest cost components in microsystem development.

Signal processing is one application area where microprocessors will have an enormous effect on technology and life, and it is only with the introduction of the more powerful 16-bit variety that they have begun to make an impact. Techniques such as digital filtering, correlation, transformation and equalisation can be applied to a wide variety of information (visual, aural, control, etc.). Here the speed-precision trade-off is re-interpreted as one of bandwidth and noise. The eight-bit processors are out of their league in such an area, and 16-bit ones are only just in it.

The principal disadvantage of 16-bits is in cost: not so much of the processor, which is rarely a dominant cost component of a system: nor, contrary to popular belief, of storage, since the condensation of program provided by the larger instruction set tends to offset the extra byte per word. The major cost component is in the doubling of the parallel busing around the system and the logic to service it. This makes circuit board layout more difficult and expensive, though the appearance of powerful LSI interface packages has greatly mitigated the problem. On the other hand programming is often cheaper and easier, since the more rational 16-bit languages usually have better self- and cross-software in the form of assemblers, editors, simulators and high-level languages.

Let us briefly look at three 16-bit single-chip processors of which a reasonable amount of experience has been accumulated.

General Instruments' CP1600 uses the well established NMOS technology in a conventional 40-pin DIL package. Full execution time of the 68 basic instructions range from 1.6 to 4.8 μ s for the faster 'A' version.

The processor contains a high-speed ALU, an instruction register, microcontrol unit and TTL compatible input/output buffers. Eight general purpose registers make the device easy to program as they give the programmer a choice of data paths through the CPU, thereby obviating the bottlenecks which occur in simpler architectures.

The associated documentation and after sales care are both excellent. The purchase of a microcomputer system automatically entitles the user to a full set of system software which includes a self-assembler (Super Assembler), numerous test and debug routines and a comprehensive set of useful subroutines, e.g. fixed and floating point multiply/divide, square root, etc. Further, any additional software is made available, free of charge, as it is released. A complete set of cross-software is also available including all of the above plus a simulator for the CP1600.

Disadvantages are a necessity for three supply voltages and the need for a two-phase clock. The major criticism, however, is the fact that only 10-bits of the possible sixteen are used by the instruction set.

It is strongly rumoured that a new version, the CP1616, will make use of the full 16-bits for its instruction set. Perhaps the power

*The authors are with Jebal Microsystems Ltd

supply and clock problems will also be resolved in this device.

CP1600 and its associated system have been available for several years now and are not only readily available but there has been sufficient time for any teething problems to be ironed out.

Texas Instruments' TMS9900 is also NMOS and requires three supplies. The enormous 64-pin DIL package allows separate 15-bit address and 16-bit data buses to be available without the need for external latches. Memory is addressed in 8-bit bytes which may also be organised as 16-bit words. The missing bit on the data bus is used to indicate which byte (upper or lower) is being accessed. There are no internal registers, but this is compensated by the presence of powerful two address operations in the instruction set, which includes separate byte and word versions of the basic orders. The 69 basic instructions include multiply and divide, achieved in 17 μ S and 41 μ S respectively. Unfortunately, this is unsigned; so the signed form, which is usually required for signal processing, is considerably slowed down by extra software. Five addressing modes are used in two-address instructions between workspace and memory, including indirect, indexing and auto-increment. The symbolic assembler is easy to use, but there is a whole range of further software including FORTRAN, COBOL and a simulator.

9900 is basically part of a whole computer family (the 990 range) including the powerful TTL minicomputer 990/10, so it carries great advantages of support and commonality of software. Furthermore, its smaller brother, the 9980, offers the same instruction set in a 40-pin package with reduced i/o facilities, i.e. an 8-bit data bus and fewer interrupt levels; so there is no need to put up with the restrictions of an 8-bit instruction set just because the data environment is 8-bit.

The family is supported by a generous range of peripherals including error correcting memory systems: a very important facility which is as yet only dimly appreciated by applications engineers.

Ferranti F100L is remarkable for being the first indigenous European microprocessor and bipolar, though some of the potential speed is sacrificed by the serial ALU. Its single chip in a 40-pin package offers a single supply, a single-phase clock and TTL compatibility to military specification. DMA and vectored priority interrupt are provided and supported by powerful set of interface chips. The large and well-structured instruction set is tuned to real-time applications, and the indivisible bit-test and set instructions in conjunction with the interface set make this a very suitable basis for multiprocessor systems.

The assembly language is not particularly scrutable, but is excellently supported by comprehensive resident and cross-software. The availability of CORAL 66 adds to the real-time capabilities. Simple instruc-

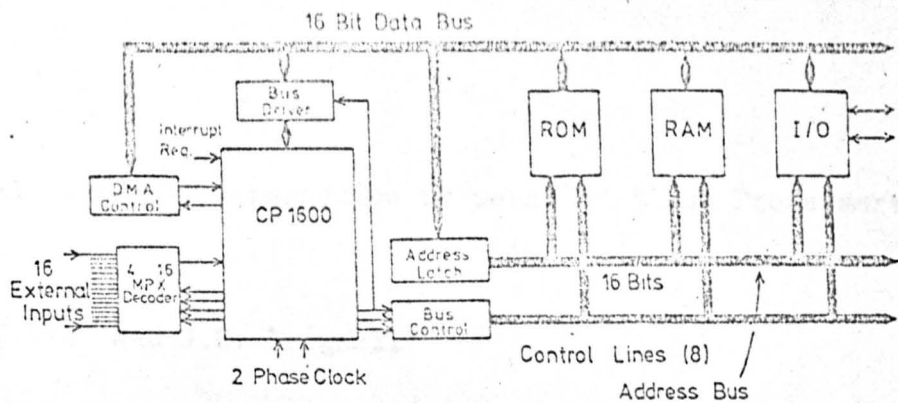
tion times range from 1.2 to 6 μ S, and the availability of a powerful simulator program with comprehensive timing, trace and monitoring facilities further eases the agonies of real-time system development. F100L is best thought of in terms of a system, since the deficiencies of the CPU, such as the single internal register, are more than compensated by the supporting hardware. In addition to the interface set there is provision for the addition of special processors. These connect directly to the bus and obey their own reserved sub-set of instructions. The most important, promised for delivery in 1978, is the multiply/divide chip which will give full 2's-complement results in 6.5 to 14.1 microseconds. This could be a decisive feature, provided it emerges less slowly than some of the other components of the comprehensive F100L development system.

At first sight F100L has fewer instructions in its set than the others, but in fact many features normally represented by separate instructions are carried out simultaneously as sub-instructions. Thus, for example, in a single instruction it is possible to increment the contents of a pointer address, increment the contents of the address now pointed to and jump if that is non-zero. Hence real-time code can be highly condensed and very fast.

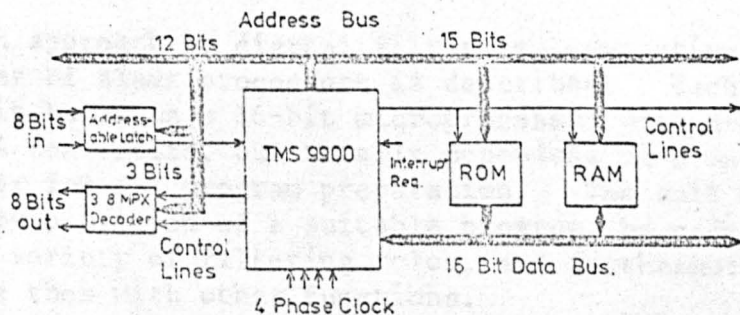
These three brief descriptions are, of course, grossly oversimplified for reasons of space, and it is essential to read the manufacturers' literature for a full appreciation. They should be sufficient, however, to demonstrate the great divergences of these solutions to similar engineering problems. Also they show how all these processors offer a better solution to high-demand problems than the 8-bit variety. The divergences are illustrated in 1, 2 and 3 which shows typical arrangements for small systems based on each of the three processors; equally the optimum program approach for a given problem is quite different for each instruction set.

It should not be assumed that 16-bit processors are more difficult to use. In fact our experience in consultancy practice, which includes a great deal of minimal-hardware 8-bit work, is that the reverse is true. Both software and hardware tend to be better structured and supported, though fewer direct development aids, such as in-circuit emulators, are at present available. For an increasing number of applications, 8-bits is totally inadequate. A simple example is illustrated in 4, in which a real-time EEG signal has to be differentiated as a stage in the identification of the feature arrowed. Such a differencing operation in 8 bits produces a noise-dominated result, particularly if more complex digital filtering operations are included (and this applies even if the A-D and D-A converters are only 8-bit), while multi-word arithmetic produces an unacceptable loss of bandwidth.

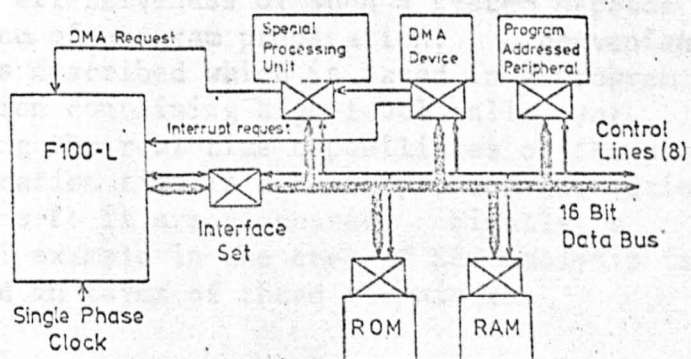
We are not suggesting that 16-bits is always appropriate; indeed, for most applications



1 Minimal systems for CP1600



2 Minimal system for TMS9900

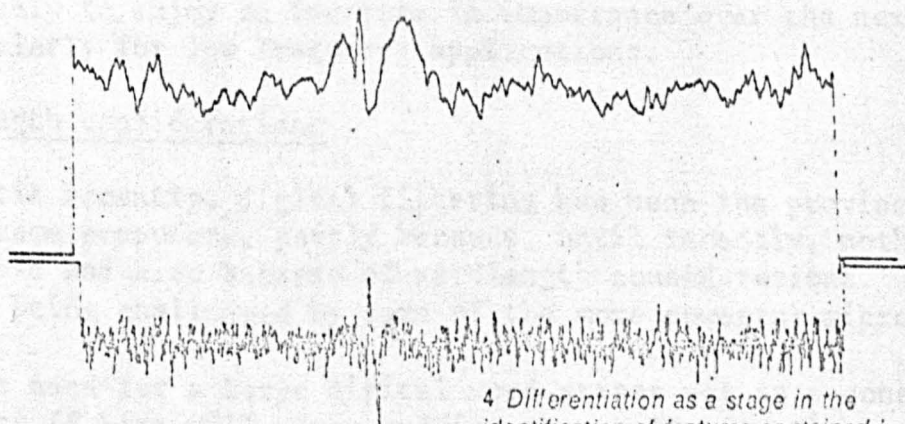


3 Minimal system for F100L

brought to us we find ourselves recommending the simplest of 8-bit systems. However, it can be extremely expensive at a late stage in a project to find oneself saddled with an underpowered system, and if there is any doubt at the outset a thorough feasibility study is an essential prelude to microprocessor selection.

Acknowledgement

We wish to acknowledge a number of helpful discussions, particularly with Dr Brignell and his research group at The City University.



4 Differentiation as a stage in the identification of features contained in an EEG waveform. Note the increased noise in the output waveform (lower)

Digital Filter Implementation by means of Slave Processors

R.A. Comley and J.E. Brignell*

An approach to digital filter implementation by means of slave processors is described. Each slave is based on a 16-bit microprocessor with A-D and D-A converters, but totally dependent on a master computer for all program preparation. The unit may thus, by provision of a suitable program, be made to fill a variety of filtering roles, and furthermore combine them with other functions.

The effectiveness of such a system depends on the system of program preparation. A convenient method is described which is based in a program description containing high-level calls, yet preserving the real-time capabilities of the processor. Multiplication time is a basic problem and various approaches to it are discussed. Finally, a practical example in the area of EEG analysis is discussed in terms of these techniques.

Introduction

The design of time-continuous filters has reached the stage where further improvement is limited by component stability and tolerances. Further, at low frequencies either physical dimensions or active feedback becomes prohibitive. Digital filters do not suffer from such limitation and are more realizable, particularly for time domain operations. They are, however, limited by digital word-lengths and processing speed of the central processor employed and by word-length and conversion rates of analogue-to-digital (A-D) converters. Recent advances have permitted rapid improvements to be made on all these fronts and have also led to a dramatic reduction in costs. As a result, digital filtering techniques are likely to enjoy an increase in importance over the next few years, particularly for low frequency applications.

Wordlength considerations

Until recently, digital filtering has been the province of mini and main-frame computers, partly because, until recently, nothing else was available and also because of wordlength considerations. This monopoly is now being challenged by some of the more powerful microcomputers.

The need for a large digital word arises not as a consequence of A-D accuracy (8 bits will often suffice though 10-bits (1 part in 10^3) is better) but from the need to define the pole-zero locations accurately and to accommodate the length of intermediate results. As digital data

* Department of Electrical and Electronic Engineering, The City University, St. John Street, London, EC1V 4PB.

proceeds through a filter the number of bits required to specify it precisely, increases as a result of the operations performed. For example, each multiplication will produce a result equal in length to the sum of the digits in the two factors and each consequent rounding represents a noise source.

As an example, consider a fourth-order Butterworth filter. Assuming the poles must be placed at an average distance of 0.035 from the unit circle (in z -plane), a peak gain of 10^4 will be experienced through the filter [1]. The output will, therefore, require 14 bits more than the input. If the input is quantised to 8-bits, a wordlength of 22-bits will be required.

One method of reducing this requirement is to divide the filter into k -cascaded sections, with suitable attenuation between stages (Fig. 1). A gain of 100 (7-bits) is now required for each stage.

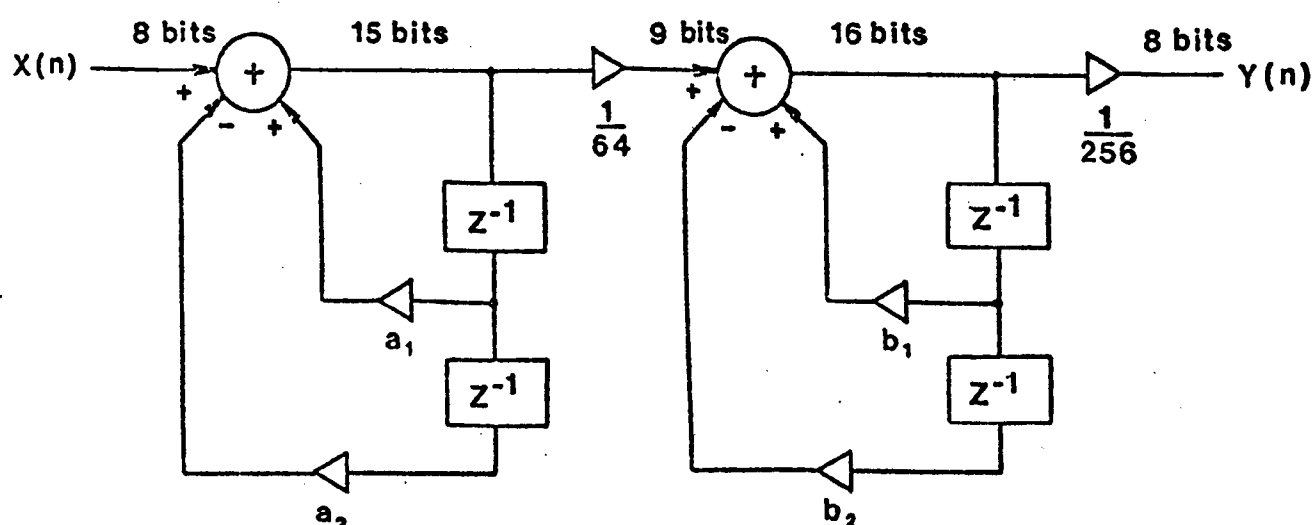


Fig. 1. Fourth-order Butterworth filter.

As can be seen, noise considerations apart, 16-bits is just sufficient for the implementation of such a filter, if organized as shown in Fig. 1. Hence, many digital filtering applications are within the capability of currently available 16-bit microprocessors. If an 8-bit microprocessor were used then double-length working must be implemented, which will complicate the programming task considerably and more than halve the operating speed of the system.

The Roving Slave Processor

The computer-aided measurement research group at the City University have been working on a technique known as the Roving Slave Processor [2] for the past three years. These are very basic hardware devices capable of performing numerous signal processing tasks, among them digital filtering. In their simplest form they comprise a central processor, program and data store and an input/output channel; i.e. a basic microcomputer. Two prototypes have been constructed based on 16-bit microprocessors, the Ferranti F100L and General Instruments CP1600. Both devices are particularly suitable for this application by virtue of their processing power, cheapness, compact size and ease of interfacing.

The RSP's are dependent upon a host computer for provision of all fundamental peripherals (e.g. teletype, reader-punch, front-panel, etc.) and for all program preparation. Special signal processing type peripherals may be

added to the slave system as required (e.g. A-D channel). The slaves are connected to a host computer via a special link which allows data to be transferred to or from the subordinate processor. The program, when ready, is dumped via this link and, once loaded, the RSP may be detached from the host computer and transported to some required site for operation. In this manner the devices can fill a whole variety of filtering (and other) roles by the provision of suitable programs.

High Level Definer

One of the major drawbacks with digital filters is that the weighting coefficients must be re-calculated to accommodate any changes in sampling or cut-off frequency. Conventional analogue filters do not suffer from any sampling frequency restrictions and may be, quite simply, provided with a continuously variable cut-off frequency.

Allied to the basic hardware system of the RSP, and of equal importance, is a programming system designed to ease such problems as this re-definition of filters. The system, known as the 'High-Level Definer', makes use of three well established concepts in computer programming, subroutines, macros and modular programming, to provide a system which allows high-level type commands to be used to generate blocks of efficient machine code [3].

Instead of a single general subroutine being written to perform some signal processing task, a suite of routines is written. The suite consists of various segments written to perform specific operations within the processing task. The required subroutine is built-up from a number of these segments, called by a high-level command and linked together to form a single block of code.

The suite is written in the assembly language of the RSP and is stored on the host computers backing store (disc). A control segment is included, written for the host computer, to organize the calling and linking of the other segments of the suite. The control segment uses parameters specified in the high-level call to determine which of the segments are required and also to perform some checking on the calling command (syntax) and the compiled code. Diagnostics relating to signal processing considerations and constraints can also be provide where relevant, via the control segment; e.g. warnings about bandwidth, accuracy, input range etc. In this manner, efficient machine code programs can be generated from a high-level type of programming language.

A further stage can be included in the control segment to permit the computation of coefficients from a primary specification, included as a parameter string in the high-level call.

The use of this programming system makes the task of re-defining the RSP as a different filter very simple. The user merely inserts a high-level call of the type:-

`FILT (BUTT,2,500,50)`

to specify a second-order Butterworth filter with a sampling frequency of 500 Hz and cut-off of 50 Hz.

Combination of operations

Besides offering a better stability than equivalent analogue filters (in terms of component tolerances and drift) and facilitating the accurate placement of the poles and zeros, digital filters also offer the possibility of combining

several mathematical functions in one computational operation.

EEG analysis - a practical example

Consider, as an example, the application of the techniques discussed to an analysis of EEG records. In particular, for the detection of high-frequency spikes, which occur in the record indicating the onset of an epileptic attack.

The method used for the detection of these spikes is based on a first-order difference of the data, as an approximation to the differentiation process. This involves subtracting the past input sample from the present, giving a rate of change of input with respect to time:-

$$\frac{dx}{dt} \doteq \frac{1}{T} [X(n) - X(n-1)]$$

or $1/T(1 - z^{-1})$ in the z-domain and $\frac{2}{T} j e^{-j\omega T/2} \sin(\frac{\omega T}{2})$ in the frequency domain [4].

The differentiation process amplifies high-frequencies, however, and can result in a very noisy output. To reduce this noise the differentiated signal is filtered via a second-order Butterworth filter, as given in Fig. 2 [5]. The differentiation and filter equations can be combined in one expression:-

$$\begin{aligned} H(z) &= 0.0675 \frac{1 + 2z^{-1} + z^{-2}}{1 - 1.144z^{-1} + 0.414z^{-2}} (1 - z^{-1}) \\ &= 0.0675 \frac{1 + z^{-1} - z^{-2} - z^{-3}}{1 - 1.144z^{-1} + 0.414z^{-2}} \end{aligned}$$

which can be realized as a single computational operation (N.B. this is equivalent to adding an extra zero). This may not be the best way of realizing this particular function but serves merely as an example of the possible combination of operations.

Bottlenecks

Multiplication is the major bottleneck encountered with digital filtering routines. A typical fixed-point, signed multiply in software takes approx. 300 μ s on current microprocessors; floating-point takes about twice this time. Consider the example given in Fig. 2, as can be seen, four multiplications are required. If all are performed by a software multiplication routine, then the processing time involved would be in the order of 1.5 ms per sample, i.e. a maxm. sampling rate of 700 Hz. Since, for a reasonable output waveform (via D-A converter), the sampling rate should be about ten times the highest frequency, generally at the 3 dB point, the cut-off frequency is limited to 70 Hz. This would not normally present any problems as digital filters find most application at lower frequencies. Unfortunately, not all filters are as simple as the one given in the example, and may involve many more stages and hence multiplications.

To overcome this problem it is usual to resort to a method of hardware multiplication. This may not always be necessary, however, as a marked improvement in operating speed can be obtained by defining the coefficients as a series of shift and add operations on the input value. Consider the 0.0675

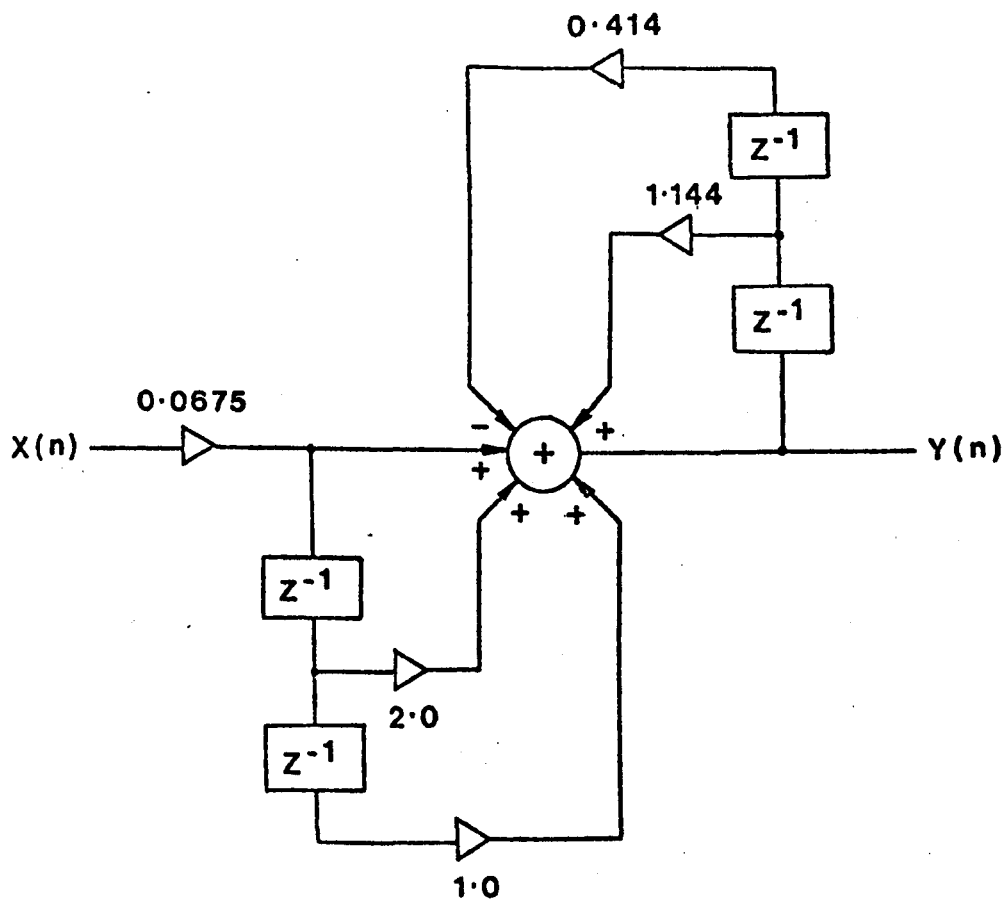


Fig. 2. Second-order Butterworth Filter
($w_s = 500$ Hz, $w_c = 50$ Hz)

0.0675	
0.0625	1/16 (4 shifts)
0.0050000	
0.0039062	1/256 (8 shifts)
0.00109380	
0.00097656	1/1024 (10 shifts)
0.000117240	
0.000061035	1/16384 (14 shifts)
0.000056205	

Fig. 3. Calculation of coefficient as a series of shift and add operations.

coefficient, as shown in Fig. 3 this can be built up from a series of four shift and add operations which involves eleven instructions. For the CP1600 the execution time is 40 μ s approx.; considerably faster than a complete multiply.

Care is needed in the use of this technique, however, to ensure that the coefficients are defined with a suitable accuracy. It has been the authors' experience that 14-bits give a usable accuracy (1 part in 10^4).

Conclusion

The RSP approach should prove a very useful research and development tool. With the use of the High-Level Definer, engineers, signal processing experts, medics, etc. (i.e. individuals whose expertise does not lie in the software field) can program the devices effectively and efficiently starting with only a limited programming knowledge.

It has been shown that many of the demanding problems encountered in digital filtering are just within range of today's 16-bit microprocessors. Care is needed, however, to ensure that coefficients and intermediate results are described with a suitable accuracy and that such problems as signal aliasing are avoided.

References

1. Mick, J.R., Digital Filter Design, Digital Signal Processing Handbook, Advanced Micro Devices Inc., 1976.
2. Brignell, J.E., Comley, R.A., Young, R., The Roving Slave Processor, Microprocessor, Vol. 1, No. 2, pp 79-84, IPC Science and Technology Press Ltd., Dec. 1976.
3. Comley, R.A., Hewish, T.R., The Software Realisation of Instruments, International Minicomputers, Microcomputers and Microprocessors '77, Geneva, May 1977. Proceedings to be published, IPC Science and Technology Press Ltd.
4. Brignell, J.E., Rhodes, G.M., Laboratory On-line Computing, Intertext, London, 1975.
5. McGillem, C.D., Cooper, G.R., Continuous and Discrete Signal and System Analysis, Holt, Rinehart and Winston, Inc., 1974.

Appendix B

Memory Technology Review

B.1 Introduction

The following appendix gives a brief description of device types and technologies of storage elements which have been discussed in connection with the memory system of the RSP. The review is by no means complete; its main function is to provide a general overall view of any relevant types of storage device which are currently available.

Tables are included for comparison and as far as possible, the devices listed are typical of the category they represent.

B.2 Semiconductor Random Access Memories

Semiconductor random access memories (RAMs) can be broadly classified into two main groups:-

- i) Bipolar
 - Transistor-Transistor Logic (TTL)
 - Emitter Coupled Logic (ECL)
- ii) Metal Oxide Semiconductors
 - p-channel (PMOS)
 - n-channel (NMOS)
 - complementary (CMOS)

Bipolar memories exhibit higher operating speeds than MOS circuits at the expense of higher power dissipation and lower packing densities. Bipolar memories are also considerably more expensive than equivalent MOS devices. Their major advantage is that most standard logic circuits are manufactured using the bipolar technology and so no interface problems are encountered. Transistor-transistor logic (TTL) is the commonest form of integrated circuit technology and is a relatively straightforward process (Fig. B.1). Emitter coupled logic (ECL) is also a bipolar technology and so is similar in fabrication, the main difference being that ECL circuits make use of non-saturating switching transistors to perform their functions. This results in much higher switching speeds but at the expense of one or two extra

diffusion stages. The additional manufacturing complexity is reflected in the price of ECL devices, which are considerably more expensive than TTL circuits.

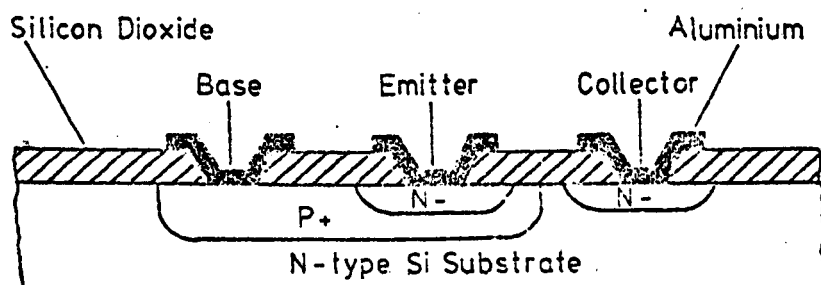


Figure B.1 Basic bipolar transistor.

The main area of use for bipolar memory circuits is in small systems where the low packing density and high power requirements are not of major significance and where high-speed operation is very important. This may often be as a small high-speed 'cache' type of memory to supplement a larger, slower system memory. It is likely that bipolar memories will always find use for this type of application, with the ECL technology becoming the dominant form. Present MOS devices are already approaching the speeds of TTL memories and so are likely to replace them in the near future.

The basic MOS transistor is a very simple device from the point of view of manufacture and is hence much cheaper to produce than the bipolar technologies. The most important advantage offered by the simple MOS transistor, however, is the great increase in packing density which may be achieved.

The operation of MOS devices relies upon the creation of a conduction channel between two diffused regions (source and drain) in a silicon crystal. The conduction channel is formed by the application of a signal voltage to a gate electrode (Fig. B.2). With PMOS, the source and drain are made of p-type material and the channel created depends on 'holes' for conduction; with NMOS, n-type material is used

and electrons are responsible for the conduction process. Since electrons have a much higher mobility than holes, NMOS devices are correspondingly much faster than PMOS circuits. This property is exploited in two ways, one is to manufacture higher speed memories of the same density as PMOS devices, or to decrease the conduction channel width so that higher packing densities may be achieved, at the expense of operating speed.

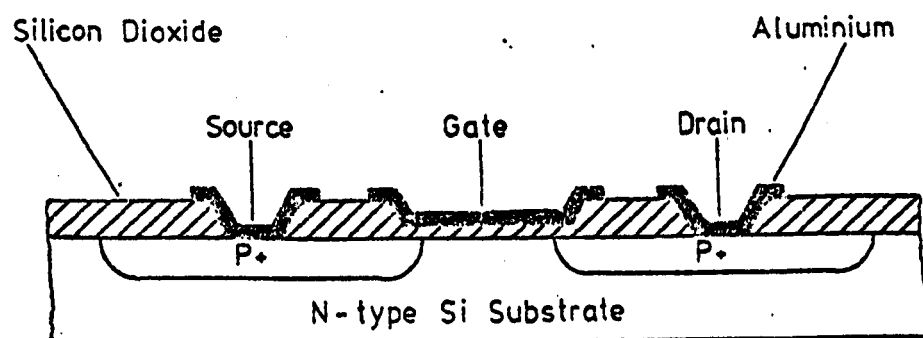


Figure B.2 Aluminium gate MOS transistor.

Many different fabrication techniques are available for the basic MOS cell, with the aluminium gate and silicon gate technologies representing two of the most popular (Figs. B.2 and B.3). Two other important variations are the ion-implanted aluminium gate, which uses ion-implantation to provide accurate control of depth and width of the source and drain regions (Fig. B.4), and VMOS which is based on a technique of etching v-shaped grooves through the various semiconductor layers before the final metalization process (Fig. B.5). Both are reported to give very good yields, which will help to reduce the cost of individual devices.

The basic MOS-FET structures possess several unique characteristics which allows their use for a wide variety of logic circuits. The MOS device may be used as an active amplifying circuit or as a load resistor (Fig. B.6).

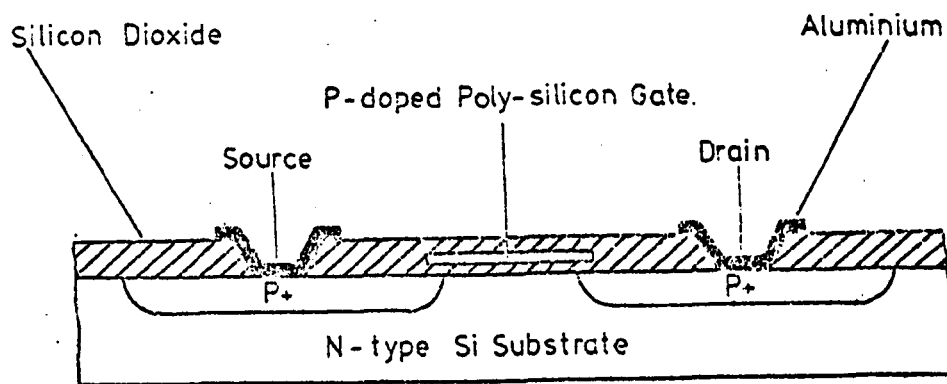


Figure B.3 Silicon gate MOS transistor.

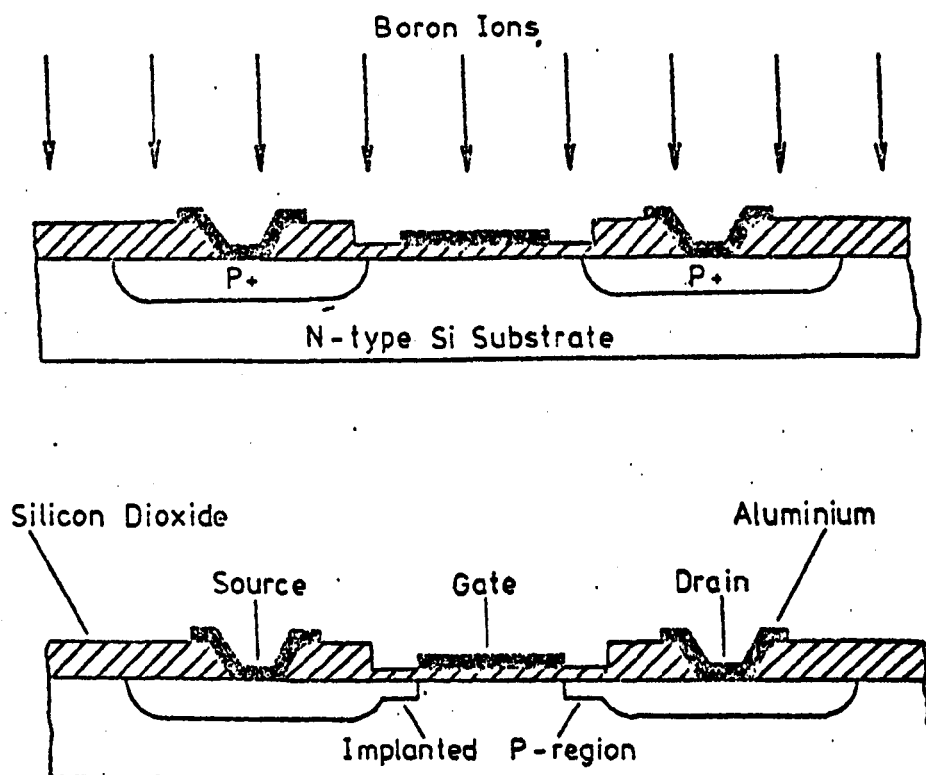


Figure B.4 Ion-implantation technique for producing MOS transistors. Note the advantage offered by the easier mask alignment.

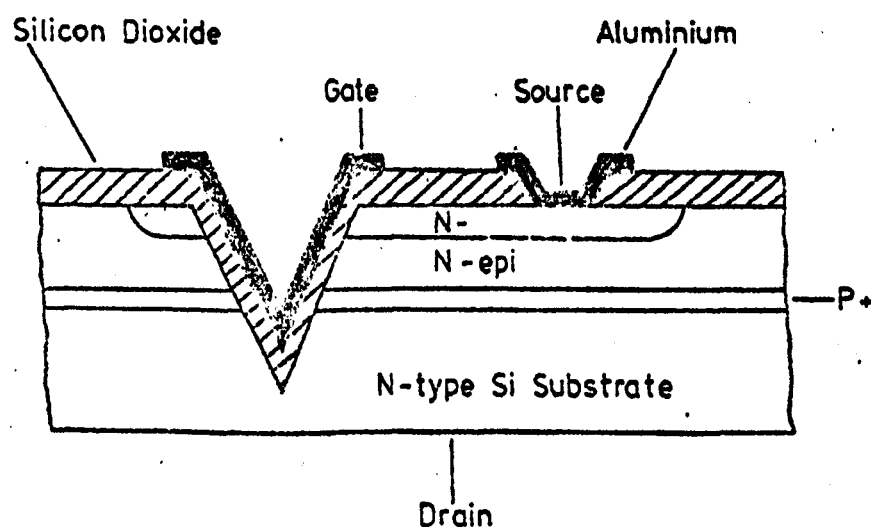


Figure B.5 Basic VMOS transistor.

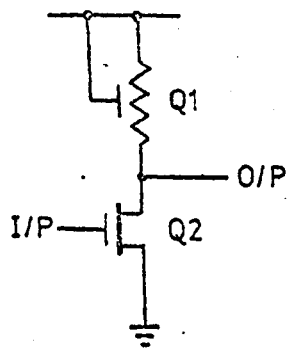
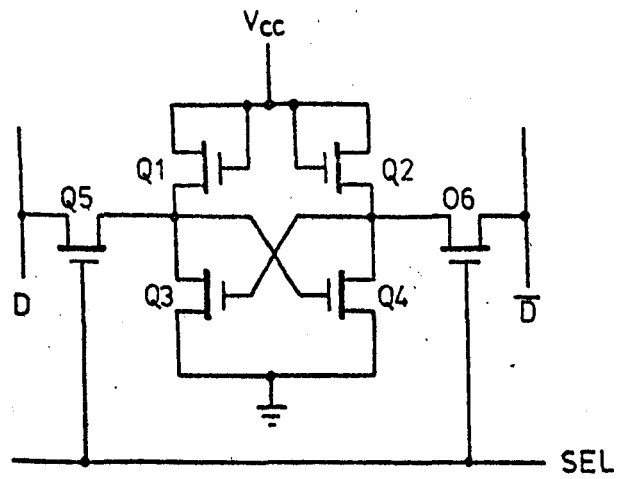


Figure B.6 Use of MOS transistors as either an amplifier or load resistor.

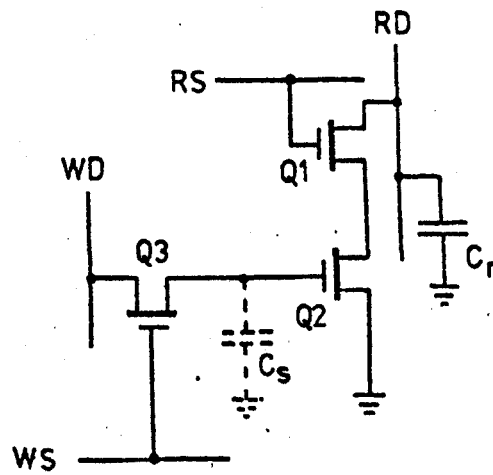
The properties of the MOS-FET devices permit the construction of two basic types of memory cell (Fig. B.7). The static memory cell is the conventional cross-coupled flip-flop cell (Fig. B.7.a) which are arranged as a two-dimensional array in the storage element. When one select line is taken high, the corresponding row of cells is connected to the data bus. To write data, signals are applied to the data lines which set the flip-flop to the desired state. To read data, both data lines are initially charged high prior to enabling the select line, which allows the flip-flop to pull one of the data lines low, depending upon its setting. The read-out is non-destructive and the cell remains in its present state until the next write operation.

The operation of the dynamic memory cell (Fig. B.7.b) is very different from that of the static cell. Data are stored as charge on the parasitic capacitance C_s associated with the gate Q2 and the connected junction of Q1. The data to be written are placed on the WD lines and WS activated. To read data, the RD line (with associated capacitance C_r) is initially charged high. When the RS line is activated, the RD line will be discharged if the capacitor C_r contains a high, and will remain high if C_r is low.

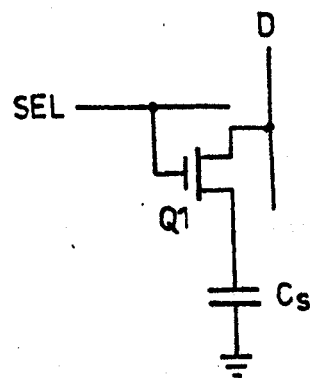
Although the read-out operation from the cell is non-destructive, surface leakage will result in the loss of charge from C_s . To maintain the stored data a periodic refresh is required. This involves reading the contents of a cell, inverting and amplifying the signal and applying it



(a) Static



(b) Dynamic



(c) Dynamic

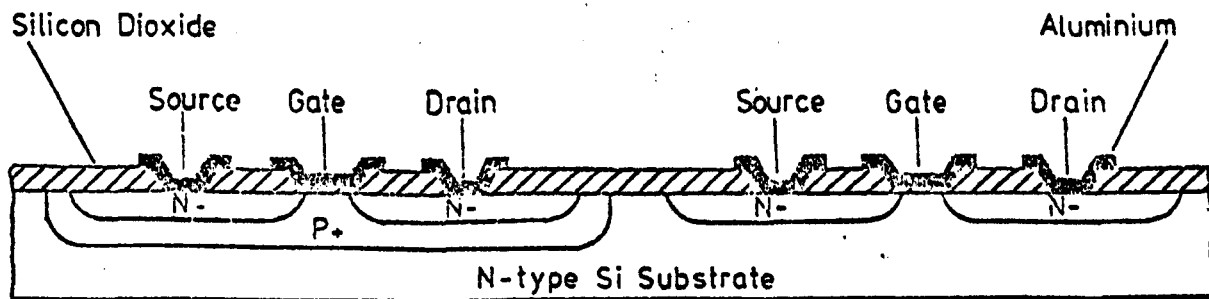
Figure B.7 Different types of memory cell possible with the MOS process.

to the WD line, and then rewriting back into the cell.

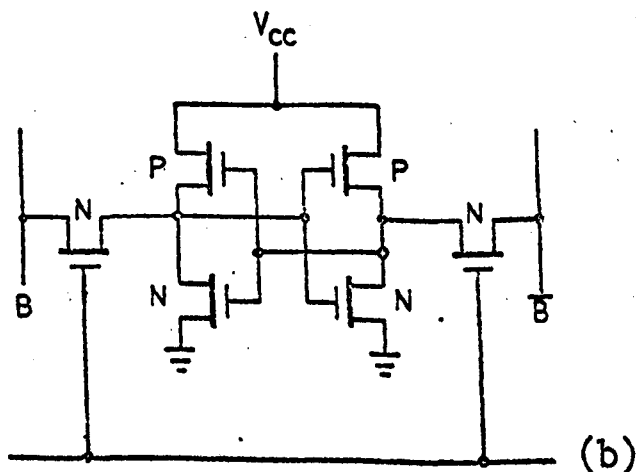
In use, the dynamic cells are laid out in a two-dimensional array. An entire row of cells is refreshed at a time, with one refresh amplifier being provided for each column of cells in the array. To refresh the entire memory, each row of cells must be individually refreshed. Typically the complete refresh operation must be performed every 2ms.

Since the dimensions of the basic dynamic memory cell are smaller than for the equivalent static cell, higher packing densities are possible, at the expense of the refresh operation. The dimensions and complexity of the dynamic cell can be reduced still further, however, with the use of a single transistor storage cell (Fig. B.7.c). For this device, the parasitic capacitance of the single FET is made use of as the storage element.

By a combination of both NMOS and PMOS transistors on the same chip, it is possible to produce circuits with a very low quiescent power requirement. This technique forms the basis of the complementary MOS (CMOS) technology.



(a)



(b)

Figure B.8 Basic CMOS circuit fabrication (a) and single memory element (b).

In order to produce a CMOS device it is necessary to diffuse, or implant, isolated doped substrate regions for either the n or p-channel transistors (Fig. B.8). The extra diffusion processes result in a lower packing density and higher manufacturing cost for CMOS memory devices when compared to other MOS memory types.

The speed of a CMOS circuit is a function of the applied supply voltage and so the user may select the operating characteristics. When operated at high-frequencies, CMOS circuits exhibit a similar power dissipation to that of NMOS circuits. The quiescent power drain may be reduced further by the use of specialized technologies (e.g. silicon-on-sapphire SOS) which minimize surface leakage effects.

As for bipolar memories, it is probable that CMOS memories will continue to find their major use in small, special purpose memory systems, particularly where low power operation and wide supply voltage tolerance are required (e.g. battery back-up, non-volatile stores).

A comparison of typical operating characteristics for each of the devices discussed can be found in Table B.1.

B.3 Semiconductor Read Only Memories

Many applications involve the use of fixed data patterns and it is only necessary to write this information to the memory once, thereafter only read operations need be performed. For these applications the read only memory (ROM) is used. As for RAMs, the memory cells are arranged in a two-dimensional array which is accessed by the use of address and select inputs.

Since the data are to remain fixed, a single transistor may be used as the basic memory cell which is permanently set to either the on or off state during the final metalization process of fabrication, to a specification provided by the user. Any of the technologies discussed in conjunction with RAMs can be, and are, employed to fabricate the transistor

elements.

The simple structure allows memories of very high density to be constructed and, more important, since the data are set by permanent connections, the device is non-volatile (i.e. the information is retained when power is removed). The main disadvantage of this form of ROM is the high initial cost involved in the generation of the mask for the final metalization layer. However, in large quantities (>1000) the ROM is the cheapest form of semiconductor memory available.

For small quantity applications, where the initial masking charge cannot be justified, the programmable ROM (PROM) provides a useful solution. Again, a single transistor memory cell is used, constructed from any of the technologies discussed for the RAMs. PROMs are programmed by selecting each location, or row of locations, in turn via address inputs with the required data supplied as a controlled, over-voltage pulse on the data output lines. The selected group of cells, being connected to the output lines, are hence set to the levels of the applied data.

For bipolar devices, the data are retained by means of a permanent alteration to the interconnecting links within the chip. These are termed 'fusible' memories and three basic methods exist:-

- i) metal links (usually nichrome) which are blown open as bits are programmed.
- ii) polycrystalline silicon links which are fused open.
- iii) semiconductor junctions which are shorted by 'avalanching'.

Once programmed, like the mask programmable ROMs, the stored data cannot be changed.

MOS PROMs retain data by means of stored charge, and are hence, erasable and reprogrammable. The basic MOS memory cell is the same as that used by the RAM devices (single transistor dynamic) except that the gate is isolated in the

oxide layer and has no external connection. The programming procedure is very similar to that used for the bipolar PROMs except that now, the gate is charged via avalanche injection. The MOS FET can hence be turned on, with a conduction path being formed between the source and drain regions (Fig. B.9).

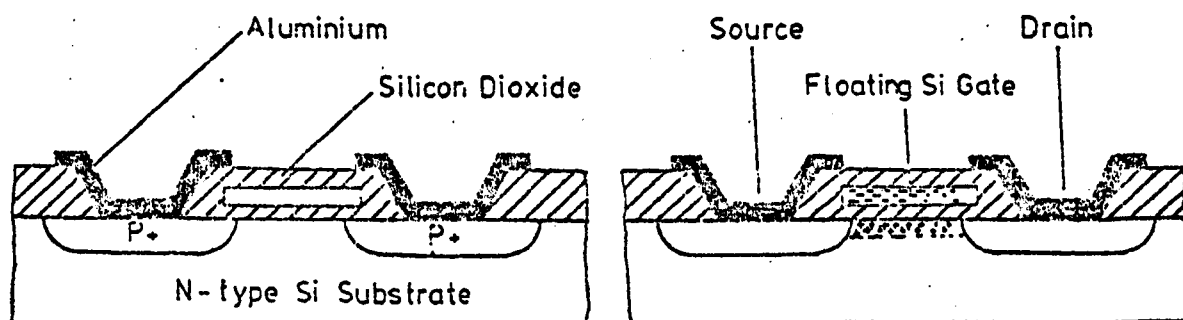


Figure B.9 UV erasable PROM structure.

The MOS FET will remain in this state for a considerable period (>ten years) irrespective of the number of read operations performed. To erase, or discharge, the gate, ultra-violet (UV) light is used which sets up a photo-current between the gate and silicon substrate, causing the gate to be discharged. UV erasable PROMs are fitted with a quartz window over the semiconductor device to permit the erasure process.

Another very useful form of erasable PROM is the electrically alterable PROM, (EAROM) which does not require the use of UV light in order to change its stored data.

These devices, like the UV PROMs, rely upon stored charge as a means of data storage. The basic MOS transistor is modified as shown in Figure B.10, where it can be seen that the normal gate oxide has been replaced with an oxide-nitride structure. The silicon dioxide layer is made very thin, typically $25\text{\AA}$, and this allows charge to tunnel through the insulator when a sufficiently high gate voltage (25 - 30v) is applied. The charges are trapped in the silicon dioxide/

silicon nitride interface and, since both are very good insulators, the charge will remain stored for a considerable period (typically ten years).

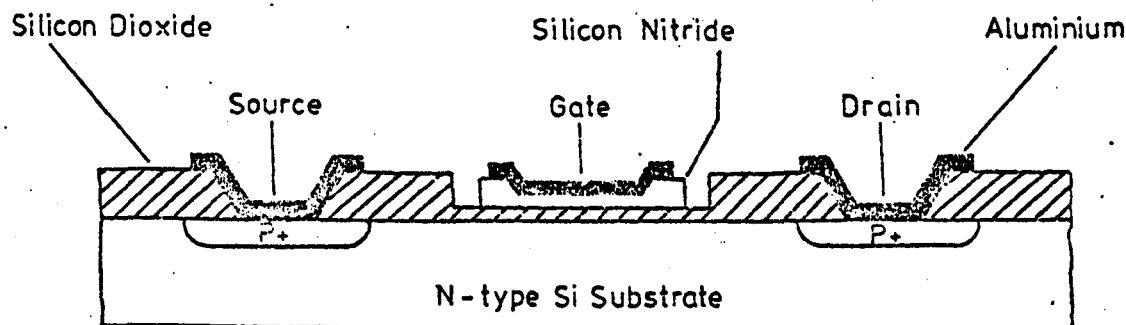


Figure B.10 Electrically alterable PROM (EAROM) structure.

Data are written by the application of a negative voltage to the gate. This causes electrons to be driven from the interface region into the substrate, leaving a net positive charge. Erasure is accomplished by the application of a positive voltage to the gate, which attracts electrons into the interface thus building up a net negative charge.

The chip is organized in such a way that writing and erasure may be performed with the EAROM in circuit, with all inputs and outputs remaining TTL compatible. All that is required are two small DC-DC converters (see Chap. 4) to generate the 25 - 30v (@ 3mA approx.) supply voltages required for the programming operation. .

The actual cell structure is slightly more complicated than the simple system given in Figure B.10, but the principle of operation remains the same.

B.4 Semiconductor, Non-Volatile Random Access Memory

A very new and potentially very important device is the non-volatile RAM. The basic memory cell structure is a hybrid of both the conventional static read-write memory cell and an EAROM store (Fig. B.11). The EAROM store is formed from a pair of MNOS (metal nitride oxide semiconductor) FETs,

inserted in series between the driver and load FETs of the flip-flop, with parallel p-channel FETs (Q3 and Q4) which act as switches.

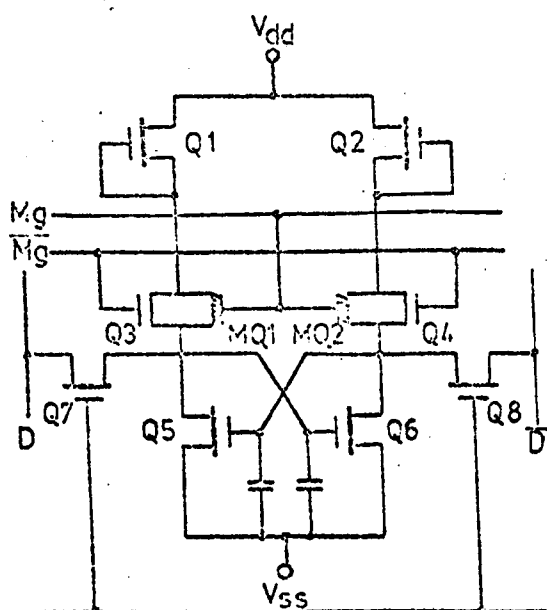


Figure B.11 Cell structure of a non-volatile semiconductor RAM.

When used in the read-write mode, the flip-flop element acts as a normal static memory cell; the switch FETs keep the EAROM cells turned off. During power down, the EAROMs are enabled, via the $\overline{Mg}$ input, supplied by the user, and the contents of the RAM cells transferred. The data may be retained in the EAROM store for up to one year.

When power is re-applied, the data stored in the non-volatile cells are transferred back to their respective read-write memory cells, and the device is again ready for active use. It is necessary to perform an erase cycle at some time during the power on mode in order to clear the EAROMs ready for the next storage operation. This is an entirely electrical process and does not require the use of any UV light source.

At present, only one device of this type is available, the Toshiba TMM142C, and its characteristics are given in Table B.1. There is some doubt as to how many times the EAROM section may be used before permanent damage occurs.

B.5 Core Memory

Very small rings of ferrite material, a mixture of ferric oxide (Fe_2O_3) and oxides of other materials such as magnesium, manganese or zinc, form the basis of a core memory. The rings are of about 2mm diameter and have hysteresis loops which have two stable magnetic states.

The direction of magnetism of a core is controlled by the direction of currents (typically 500mA) in two small wires, threaded through the core (Fig. B.12). The current in each wire is insufficient on its own to produce a switch in direction of magnetization, but where they coincide, cause the core to be polarized in a particular direction. One state is chosen to indicate a binary one and the other a zero.

A large number of cores is threaded on to wires arranged on a frame in x and y planes to form a matrix. For convenience this is usually a square, as shown in Figure B.13. A typical practical memory matrix would have a 64×64 core matrix, giving a $4k \times 1$ bit storage capacity.

To write information to a particular location, currents are applied to the appropriate address lines, e.g. X2 and Y3 if core F is to be accessed (Fig. B.13). No other core will receive sufficient current to cause any change and so only core F is affected.

With this simple system, any accessed location will be set automatically to a logic one, and so a third line, the write (W) or inhibit line, which passes through all the cores, is used to control data input. When the W line carries a current, in an opposite direction to those in the X and Y lines, the core is prevented from changing state. If the W line does not carry a current then the combination of the X2 and Y3 currents will cause a logic one to be stored. (Note the core must be cleared to a logic zero before the write operation takes place).

To read information from the store a fourth line, the

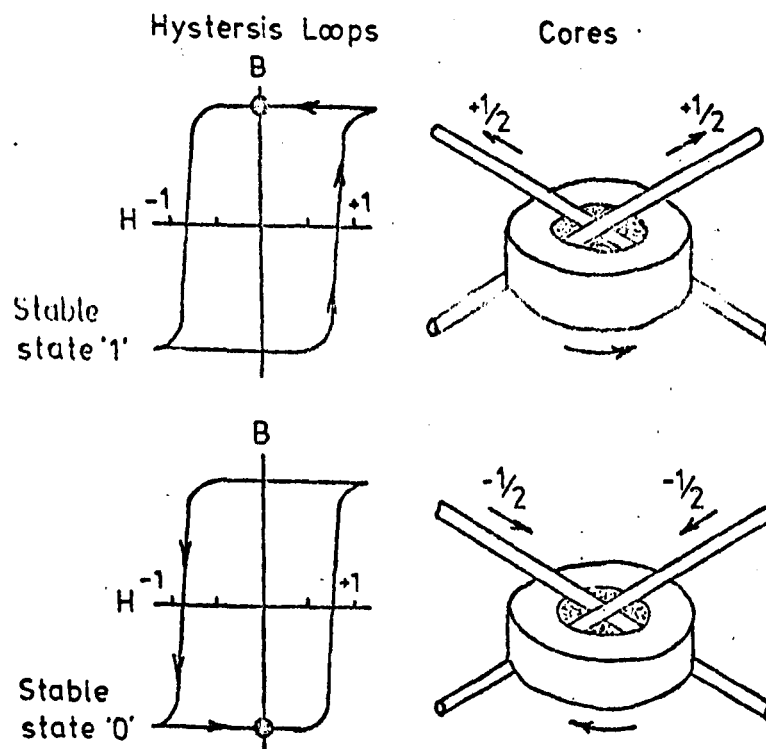


Figure B.12 The two stable states of a magnetic core.

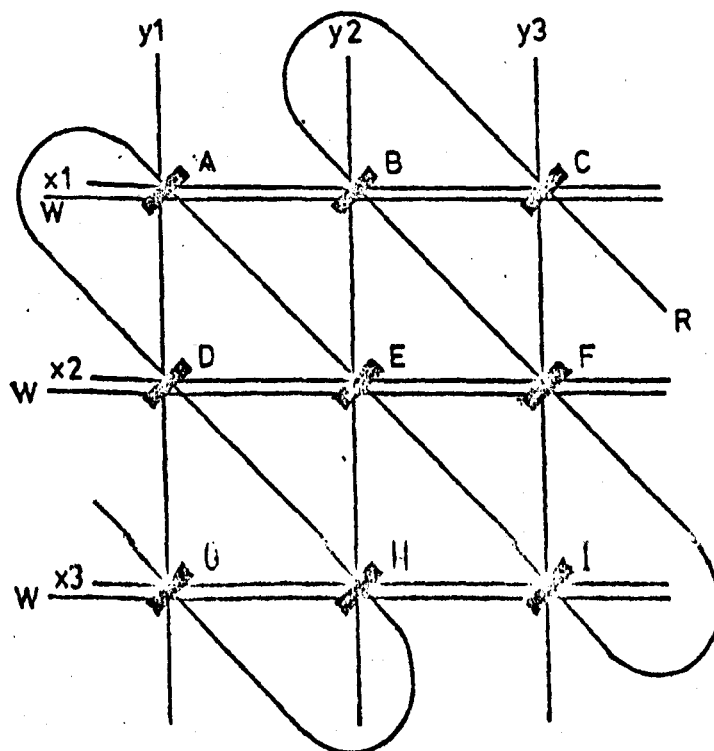


Figure B.13 A nine-core memory matrix.

read (R) line, is used as a sensor and is also threaded through all of the cores. Current is now passed in the opposite direction through the two address lines, X2 and Y3. If core F contains a logic zero then no effect will be seen on the R line, but if magnetized in the logic one direction the application of reverse X and Y currents will cause the magnetic field to be reversed. The abrupt flux reversal in the core induces a current in the R line, which indicates the presence of a logic one at location F. Note, the read-out is destructive, i.e. the stored data are erased, and so a read cycle must be immediately followed by a write cycle to restore the data.

Hence, every memory access must include the time for a read cycle followed by a write cycle. For the write case, the new data are re-written to the store in place of the data just read. This makes core storage rather complicated to use and slow in operation, typical cycle times being in the order of 1 - 2 μ s.

The dimensions of the cores, and hence the memory, can be reduced by combining the R and W lines as one line. This is possible since they are never required together. Problems have been experienced owing to the large difference in magnitude between the inhibit and read sense pulses (1000:1 approx.) but these have now been overcome. The use of this technique allows the core size to be reduced to 0.5mm which helps to increase access speed, improve packing density and reduce power consumption.

Core memories are generally used in the form of complete memory modules, with all necessary drive and sense circuitry included, rather than as single core planes. A typical specification for a core memory module is given in Table B.1. It is unlikely that future developments will be able to offer any significant improvements to these performance figures.

B.6 Charge-Coupled Devices

Charge-coupled devices (CCDs) make use of the controlled movement of electric charge to perform their functions. In their simplest form, they consist of three layers; the semiconductor material, an oxide layer and a metal electrode, i.e. an MOS structure. They are basically junctionless devices except for small diffused p-n junctions at the input and output.

The operation of a CCD requires that charge, in the form of minority carriers, be transferred from the region under one electrode to another. The charge represents the data and is moved by control of the voltages applied to the electrodes.

When data (negative voltage) are applied to the input gate, positive charge is drawn from the input p-n junction to the region under the gate (Fig. B.14.a). When a negative voltage is applied to electrode 1, electrons in the n-type material are repelled, thus forming a depletion region. This depletion region produces a potential well (Fig. B.14.b). The positive charge drawn under the input gate is transferred to electrode 1 and stored in the potential well, provided that the electrodes are close enough to allow coupling.

The charge, or data, can be transferred from electrode 1 to electrode 2 by the application of a negative voltage to electrode 2 and a reduction of the voltage on electrode 1. The transfer voltage forms a deeper potential well under electrode 2 and the charge flows to this well. Thus, by proper manipulation of the electrode voltages, information may be shifted through the CCD.

The simplest CCD is a three-phase device in which electrodes 1, 2 and 3 form a single memory cell. Information is stored under only one of the electrodes while the other two provide isolation.

To form a memory, a number of these cells is connected to form an endless-loop shift-register. Very high packing

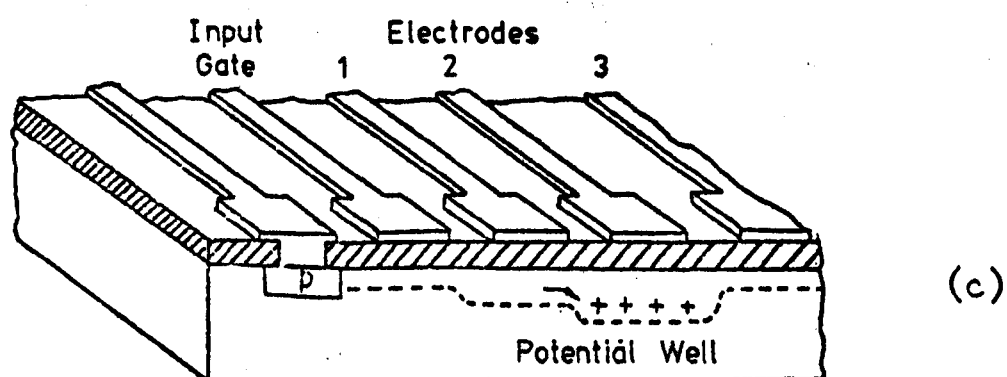
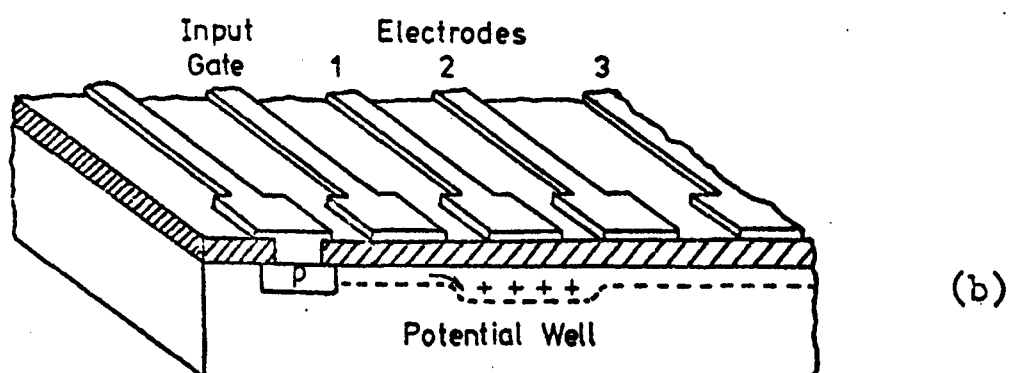
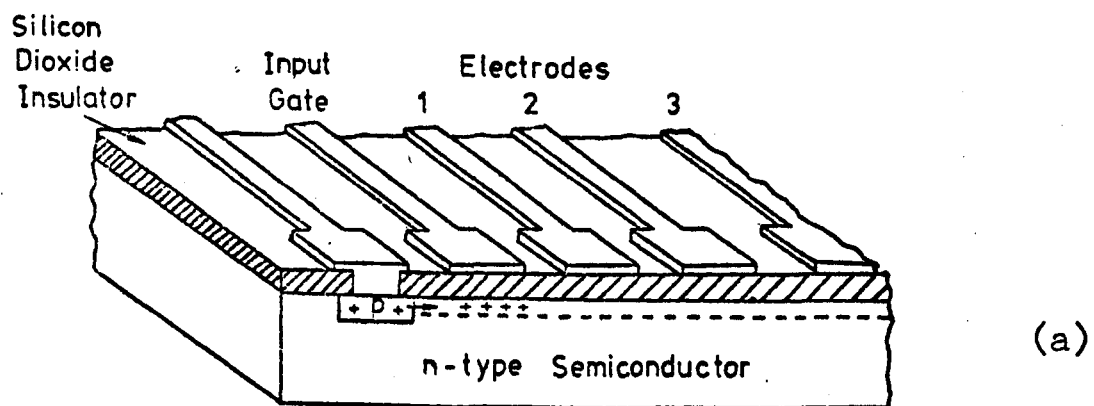


Figure B14 Information storage in a charge-coupled device.

densities are possible using this technology (with some modification to the very simple structure given in Figure B.14), with current devices offering a $64k \times 1$ bit capacity. If the entire memory were organized as a single shift-register, then average access times would be extremely long.

In order to provide reasonable access times the memory is organized as a number of smaller endless-loop shift-registers, any of which may be accessed individually. A typical configuration is shown in Figure B.15, in which a $64k$ device is divided into $16 \times 4k$ bit shift-registers. In order to reduce power dissipation, the $4k$ bit registers are further sub-divided so that a slower shifting frequency may be used. Several arrangements are possible, with the series-parallel-series configuration being the most common. This takes the basic form of two serial and one large, multi-channel parallel shift-register (Fig. B.16). The data are entered serially into the upper high-speed input register and once loaded, transferred in parallel into the slower middle register. All of the vertical, parallel channels are clocked together at a lower frequency than the serial registers. At the output the process is reversed. Hence, power consumption is reduced since the bulk of the data transfers take place in the slow parallel registers.

Note the inclusion of a regeneration amplifier in Figure B.16. This is needed to replace charge lost during transfers between electrodes. These transfers are not 100% efficient and so a periodic recharge is required to maintain the data. In addition to the transfer inefficiencies, charge is lost as a result of surface leakage and this places a minimum value on the clock frequency, below which the stored information will be lost (i.e. the device is dynamic).

The mechanism used for data storage in these devices requires that power be maintained at all times if the data are to be preserved and further, their dynamic nature requires

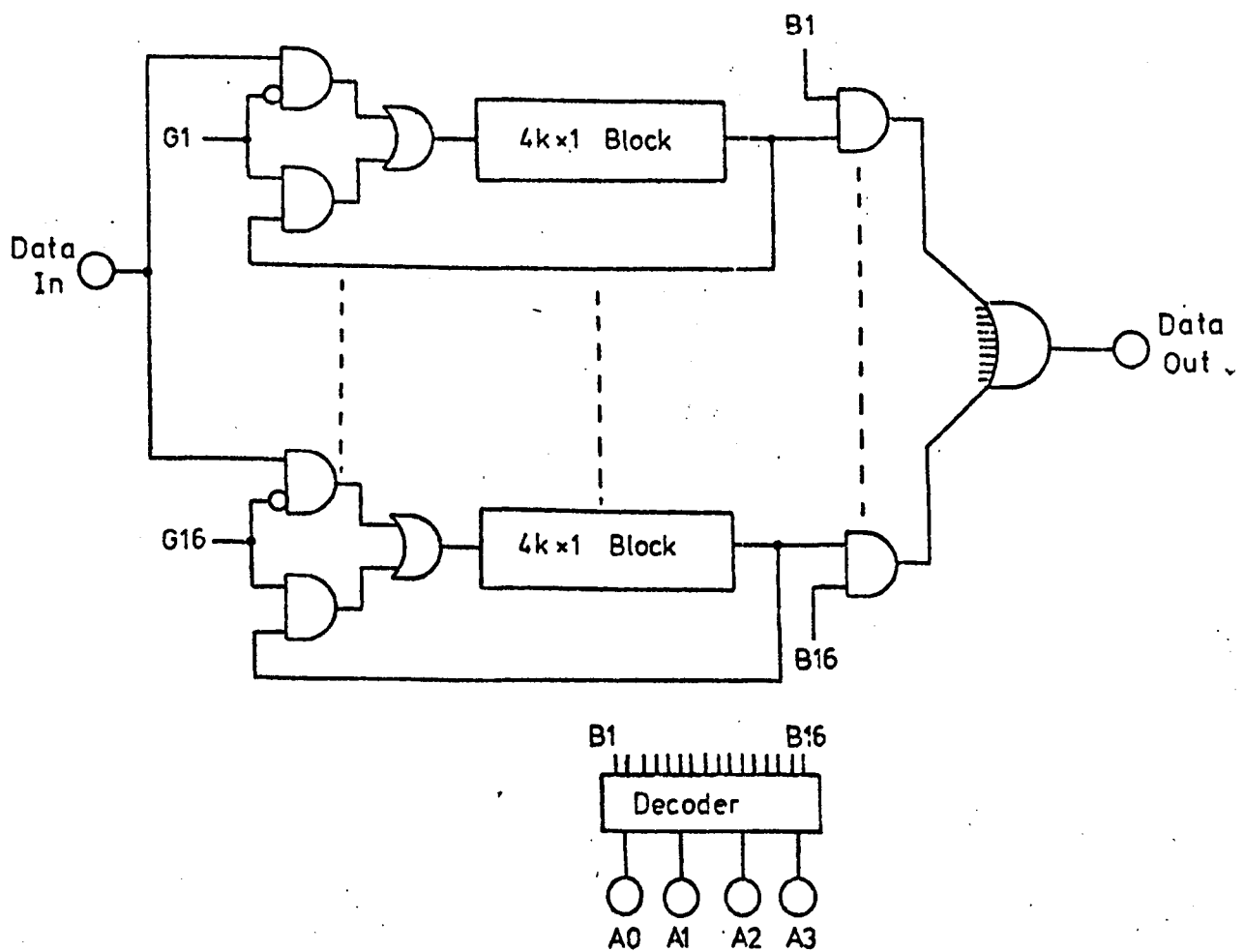


Figure B.15 A typical CCD memory configuration.

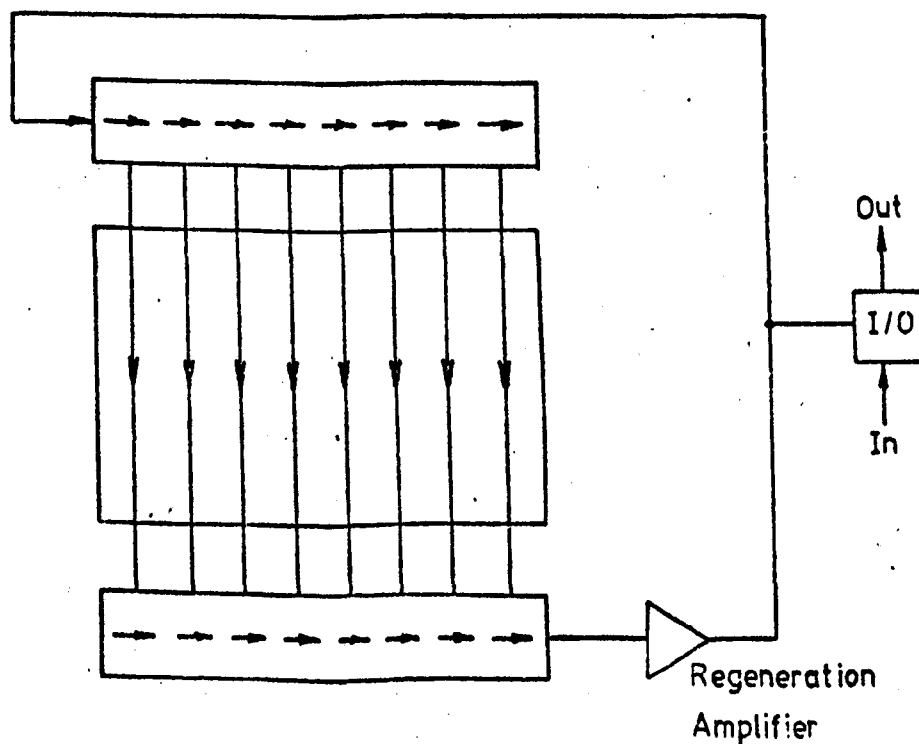


Figure B.16 The series-parallel-series register configuration.

that they are cycled (or clocked) continually. A low-power standby state is, however, generally provided in which the clock input frequency may be reduced to a minimum. The other disadvantage of these devices is the need to perform a 'flushing' operation after power-on which demands that two or three complete refresh cycles be performed in order to achieve synchronization within the memory.

Characteristics for a typical CCD memory are given in Table B.1.

B.7 Magnetic Bubble Memory

A magnetic bubble is a cylindrical magnetic domain with an opposite polarization to that of a thin layer of magnetic material, usually garnet, in which it is embedded. In Figure B.17.a, a single slice of magnetic garnet is shown, with the magnetic polarization arranged normal to the surfaces.

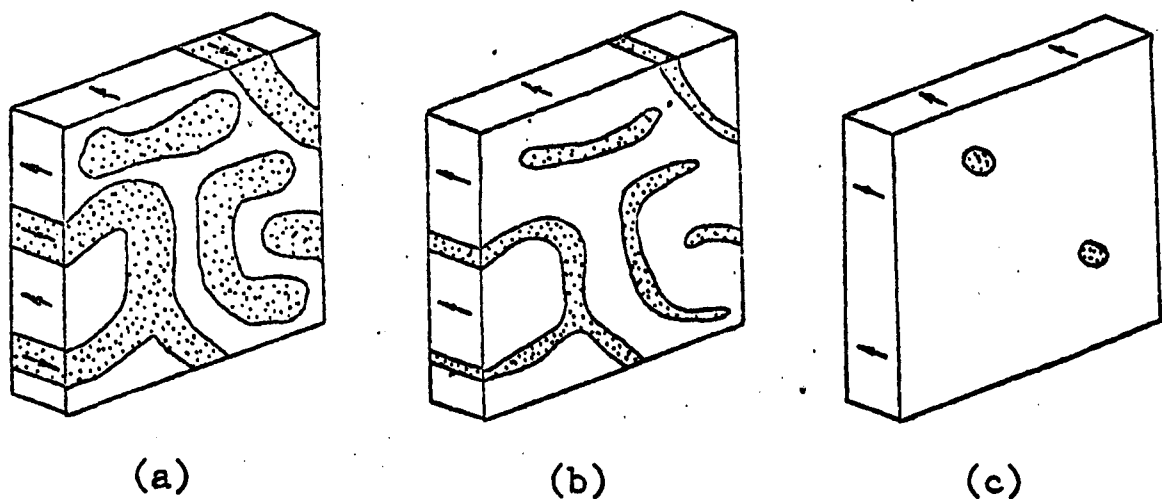


Figure B.17 Formation of a magnetic bubble.

If an external field is applied, in the downward direction, it will tend to shrink the domains of opposite polarity, Figure B.17.b. A critical value of field intensity is eventually reached in which the single walled, or island, domains become cylindrical, resulting in a magnetic bubble of 2 - 3 μ m diameter. Increasing the applied field further will cause the bubble domains to shrink and will lead to the collapse

or 'annihilation' of the bubble (Fig. B.17.c).

The movement of magnetic bubbles is controlled by a pattern of 'soft' magnetic bars (permalloy), deposited on the surface of the garnet slice, and a separate rotating drive field in the plane of the device. The bubbles can be thought of as small bar magnets and as the drive field rotates so the permalloy bars assume the magnetizations shown (Fig. B.18), causing the bubbles to 'move'. No material is actually transferred, the apparent bubble motion is caused by the flipping over of magnetic vectors at successive sites within the slice. Hence, it is possible, for example, to shift bubbles around an endless-loop shift-register (c.f. CCDs).

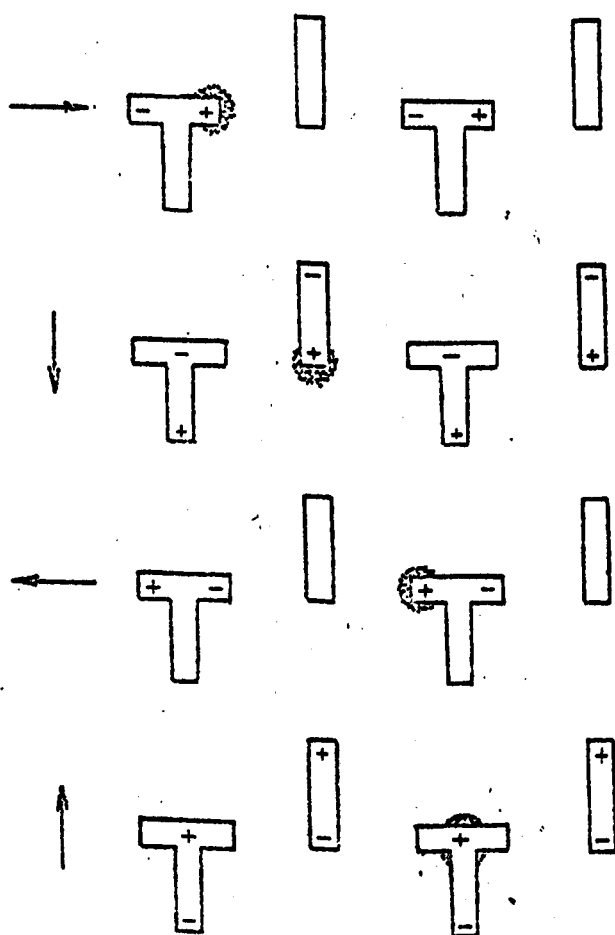


Figure B.18 Movement of a bubble along a row of permalloy bars by the application of a rotating drive field.

Many permalloy patterns have been developed and the spacing of these bars, together with the bubble-to-bubble interaction problems, determines the packing density of the device.

For a practical device, methods of generating, detecting and removing bubbles must also be provided. A bubble generator can either create a bubble directly with a current loop, or by splitting a source bubble, held under a permalloy disc (Fig. B.19). At the correct moment in the field cycle, a current pulse in a nearby hairpin loop, transfers a portion of the source bubble to an adjacent bar and thus into the system. The source bubble is never depleted since it is purely a function of the base magnetic material and bias field.

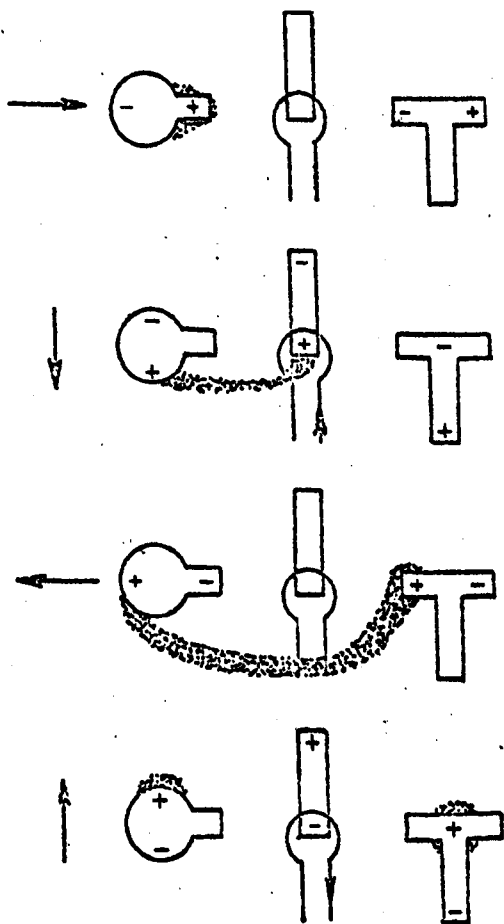


Figure B.19 Bubble generation.

A simple method of removing bubbles is with a current pulse in a hairpin loop in such a way that when a bubble passes under the loop, the high local field collapses it.

There are various methods of detecting the presence of a bubble, the most common being a measure of the change in resistance of the permalloy strip as a bubble passes under it.

Hence, it is possible to read and write information to and from a bubble device and, further, if the bias field is applied by some form of permanent magnet, the bubbles will remain permanently stable and thus a non-volatile memory is possible.

A cross-section of a typical bubble chip is given in Figure B.20 and an exploded view of a complete bubble memory in Figure B.21. The entire memory is housed in a metal case which provides a magnetic shield to protect the stored information.

Very high packing densities are possible with this technology, typically of the order of 10^6 bits/in² for present devices with 10^8 bits/in² predicted for the near future. Hence, it is possible to fabricate non-volatile memory devices of very large storage capacity on a single garnet slice, e.g. Texas Instruments manufacture a 92k x 1 bit device.

At present, bit rates of approximately 100k bits/sec. are possible and so, clearly, if the entire memory were implemented as a single shift-register the average access time would be prohibitively long. To overcome this problem, the memory is organized in a major-minor loop configuration, in which a number of small recirculating shift-registers are interconnected by a larger register (c.f. CCDs) (Fig. B.22).

The address, or position, of the minor loops is held in an external counter which is incremented for each complete 360 degree magnetic field rotation. The counter is normally reset to zero upon power-up and so it is essential that the

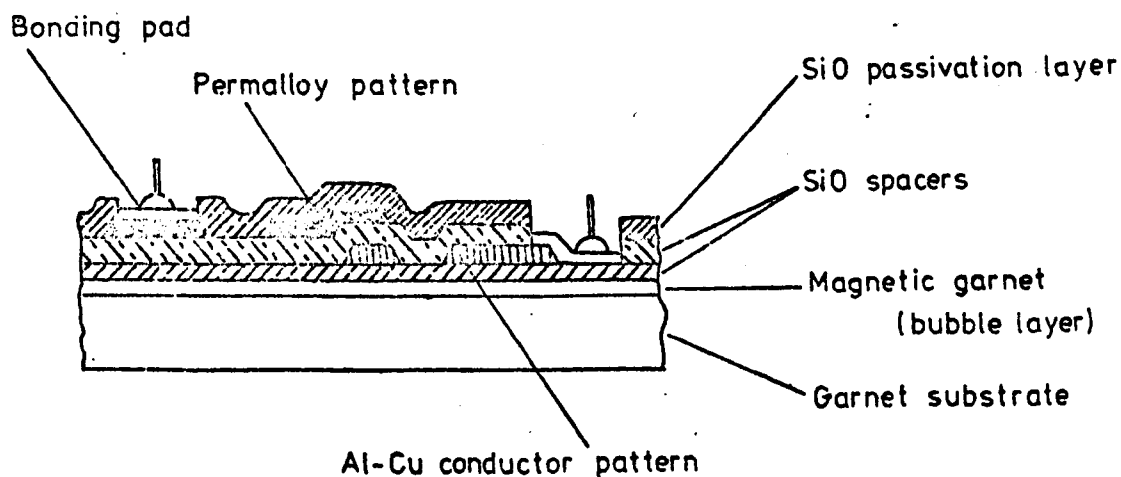


Figure B.20 Cross-section of a typical magnetic bubble device.

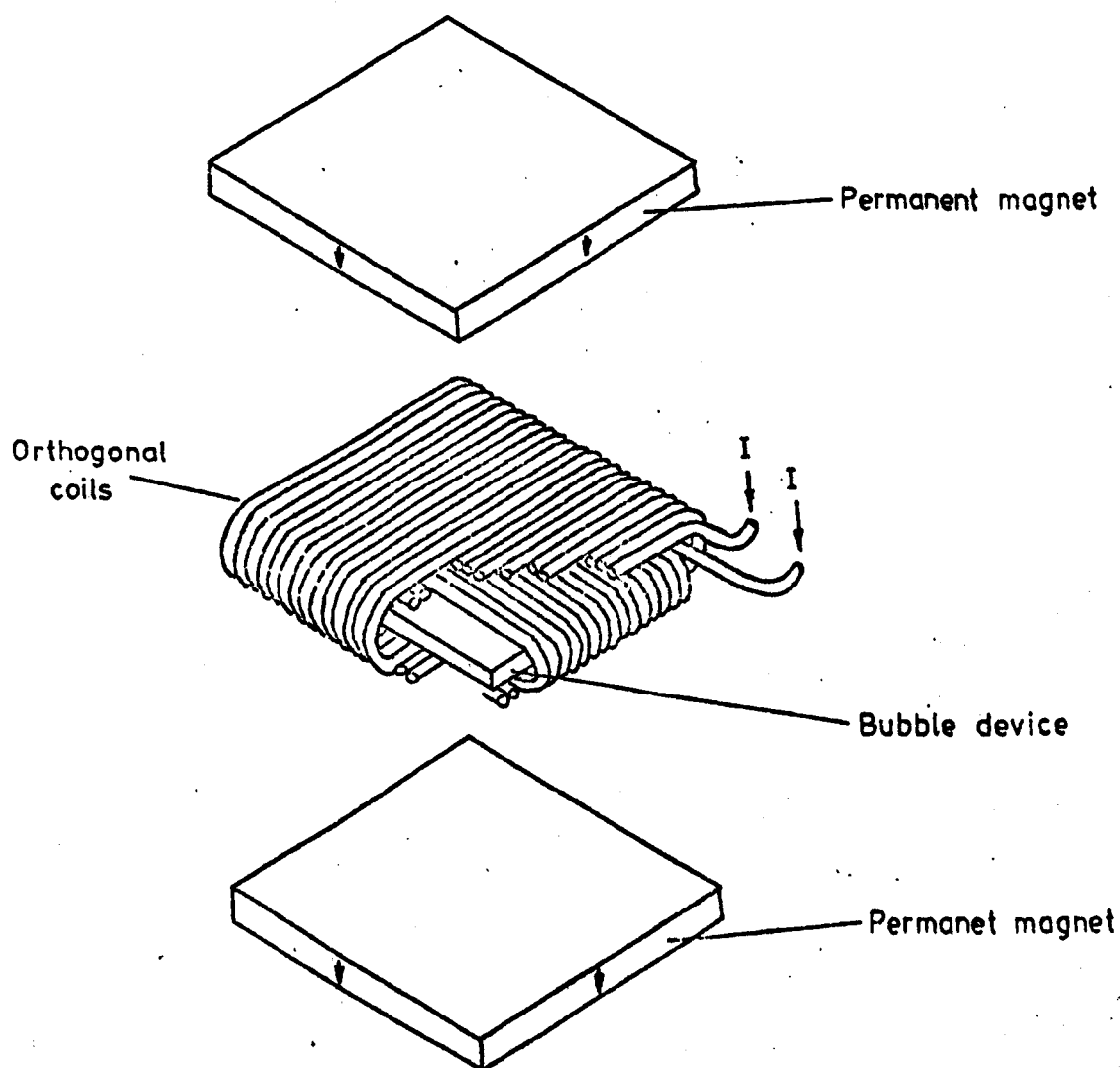


Figure B.21 Exploded view of a complete magnetic bubble memory.

minor loops are set to this position before power is removed.

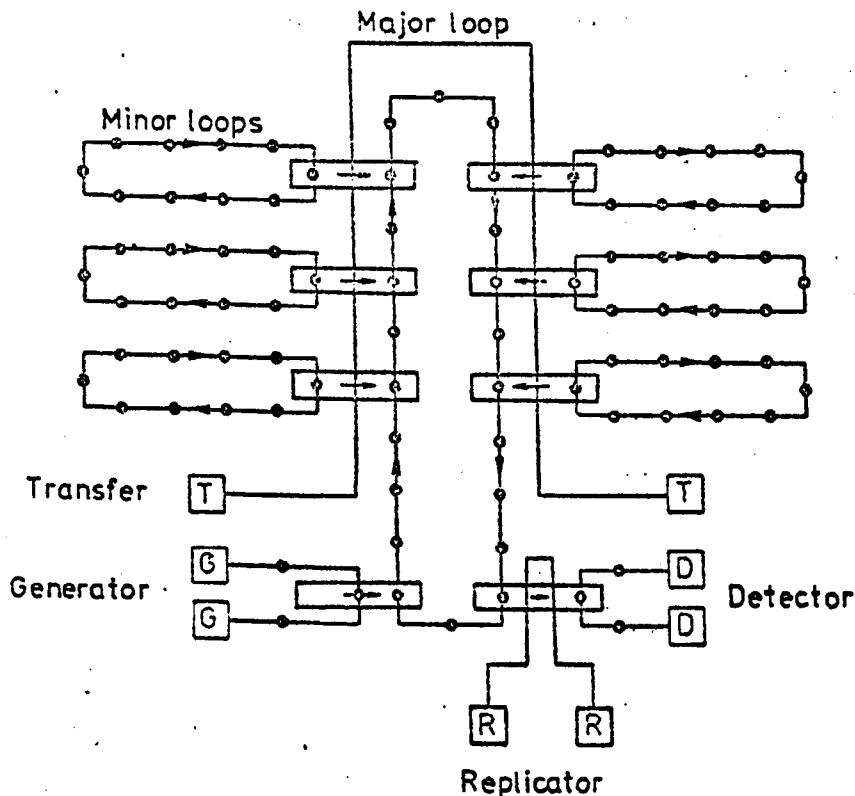


Figure B.22 Major-minor loop configuration of a typical bubble memory.

To write information to the memory it is first introduced in serial form into the major loop. A block of data (or page), equal in length to the number of minor loops, is shifted into the major loop which, when full, is transferred in parallel to the minor loops via the transfer input (T).

To read data, the minor loops must be rotated to place the required block at the top bit position of the minor loops. (Note, the address for block 0 will be $(0 + N)$ where N is the number of minor loops, owing to the shift introduced by the write operation). The transfer input is used to remove data from the minor loops and place it in the major loop. The major loop contains an equal number of bubble positions to the minor loops (assuming a single stage detector) and so the major loop must be shifted around to bring the bubbles to the replicator section. Here, the bubbles may either be transferred to the detector stage, which removes them from the major loop and results in a destructive read-out, or may

be copied into the detector, which results in a non-destructive read-out. When all of the data bits have been read from the major loop, it is shifted further to re-align with the minor loops and the data in the major loop transferred back to the minor loops.

Characteristics for a typical bubble memory are given in Table B.1.

B.8 Cassette, Cartridge and Disc Stores

The bulk data storage field is, at present, dominated by mechanically based storage systems, with cassette, cartridge and floppy disc systems being the most popular for small scale applications (e.g. microprocessor systems). These are relatively slow storage devices (see Table B.2) but are both reliable and very cheap in terms of cost per bit.

The cassette uses 3.81mm mylar tape coated with a magnetic oxide and housed in a standard 'Phillips' type C11 cassette case. The C11 cassette is basically as shown in Figure B.23, with the tape driven from one reel to the other, in either direction, usually by a capstan drive mechanism. Both reel hubs are usually driven, which results, typically, in a three motor cassette drive.

An alternative approach to the cassette drive is the cartridge system. Like cassette, 3.81mm magnetic tape is employed as the storage medium, housed in a '3M DC300A' cartridge case, but now an integral drive band is also included (Fig. B.24). The use of this band allows the cartridge drive mechanism to be reduced to a single motor, but due to other complications, cartridge drives (and cartridges) tend to be more expensive than cassette systems. Cartridges do, however, offer a much better performance than is possible with cassette based systems (see Table B.2) thereby achieving a better cost per bit ratio.

Recently, miniature versions of both the cassette and

cartridge devices have been introduced and these could prove very important in the future where size and weight are major considerations.

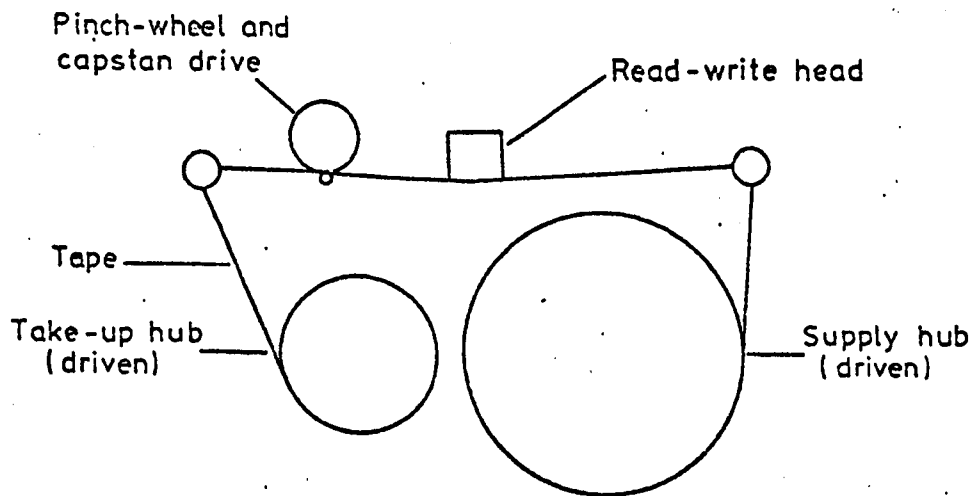


Figure B.23 Standard 'Phillips' type C11 cassette.

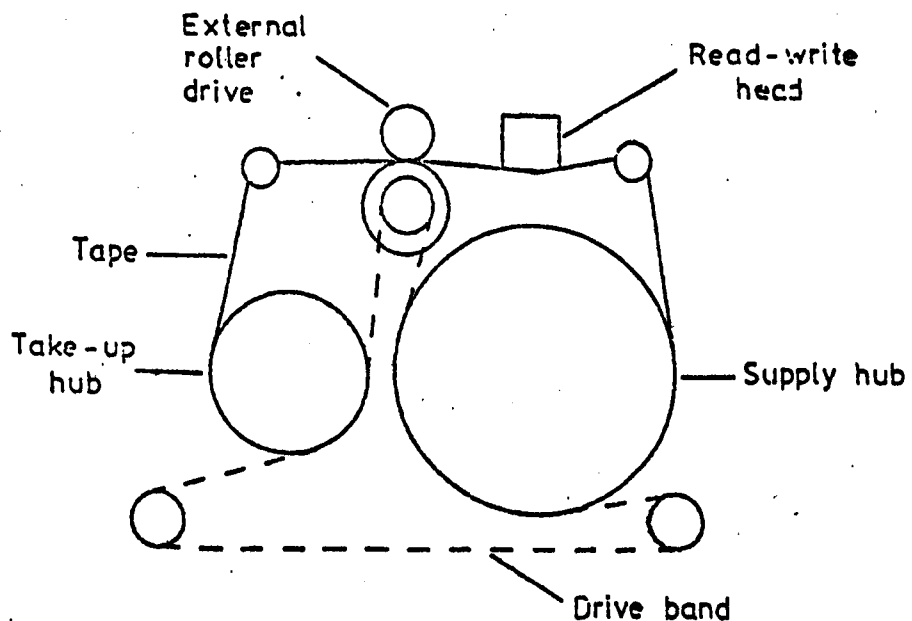


Figure B.24 Standard '3M DC300A' cartridge.

The floppy disc again makes use of a magnetic oxide as its storage medium, but unlike the cassette and cartridge, a flexible mylar disc is used as the base. The disc is 200mm in diameter, and is housed in a 203mm square plastic sleeve. The operation of a floppy disc is very similar to that of the more conventional 'hard' disc system, with the disc rotated at high speed while the heads traverse the surface.

Floppy discs, although bulkier and more costly than cassette or cartridge systems, offer much higher access

speeds (i.e. the average latency is much lower) and data rates (see Table B.2). As for the cassette and cartridge, a miniature version is also available which uses a 130mm disc in a 133mm sleeve.

Numerous recording formats are at present employed for all three devices, although some standards do exist. Every format appears to have relative advantages and disadvantages and, as a result of their diversity, no attempt will be made to explain them here.

Device	Type	Organization	Access Time	Cycle Time	Package	Power Supplies
Bipolar RAM	74S200 (TTL)	256 × 1	31ns	31ns	16 pin dil	+5V @ 110mA
	F10405 (ECL)	128 × 1	11ns	16ns	16 pin dil	-5.2V @ 90mA
MOS RAM						
Static	MK4104-4	4k × 1	250ns	385ns	18 pin dil	+5V @ 27mA
Dynamic	MK4116-3	16k × 1	200ns	375ns	16 pin dil	+12V @ 35mA +5V - load depd. -5V @ 200 A
VMOS RAM (Static)	S2114A-1	1k × 1	150ns	150ns	18 pin dil	+5V @ 50mA
CMOS RAM (Static)	IM6508-1	1k × 1	300ns	465ns	16 pin dil	4V - 11V 1.6mA @ 5V and 1MHz
ROM (MOS)	MK3400-3	2k × 8	350ns	500ns	24 pin dil	+5V @ 60mA

Table B.1 Memory device types and specifications.

Device	Type	Organization	Access Time	Cycle Time	Package	Power Supplies
PROM (Bipolar)	I3608	1k × 8	80ns	120ns	24 pin dil	+5V @ 190mA
UV-PROM	MK2708T	1k × 8	450ns	700ns	24 pin dil	+12V @ 20mA +5V @ 10mA -5V @ 20mA
EAROM	ER2805	2k × 4	1.65μs	3.5μs	24 pin dil	+5V @ 14mA
Non-volatile RAM	TMM142C	256 × 4	1.5μs	1.75μs	18 pin dil	+5V @ 70mA -15V @ 35mA
CORE	TCM409	4k × 9	300ns	800ns	Double-Euro (2 card mod.)	+5V @ 7A
CCD	TMS3064	64k × 1	200ns	Av. latency 820μs	16 pin dil	+12V @ 100mA (pk) +5V @ 2mA -5V @ 60mA (pk)
Bubble	TBM0101	92k × 1	4.0ms (Av)	12.8ms (Av)	14 pin dil	±12V @ 0.5A (coil drive)

Table B.1 (cont.) Memory device types and specifications.

Device	Type	Storage medium	Storage capacity	Data transfer rate	Read/write speed	Rewind speed	Power supplies
Cassette	MFE Model 250B	Phillips C11 cass.	366k bytes	32k bits/sec	80in/sec	120 in/sec	+5V @ 950mA -5V @ 500mA
Mini-Cassette	Mini-Raycorder 6409	Mini-cassette	128k bytes	2.4k bits/sec	3in/sec	20in/sec	+5V @ 300mA
Cartridge	Perifile C 6300	3M DC300A cartridge	2.5M bytes	40K bits/sec	25in/sec	90in/sec	110V - 240V
Mini-Cartridge	(HP 2644A unit)	3M DC100	115k bytes	8k bits/sec	10in/sec	60in/sec	Not avail.
Floppy disc	Shugart 900/901	Standard floppy disc	400k bytes	250k bits/sec	83.3ms (Av. latency)		+24V @ 2A +5V @ 1.5A -5V @ 0.2A
Mini Floppy	BASF 6106	Mini-floppy disc	125k bytes	125k bits/sec	100ms (Av. latency)		+5V @ 0.5A +12V @ 0.65A

Table B.2 Magnetic tape and disc storage systems - device types and specifications.

Appendix C

Memory Reliability

Error detection and correction for memories

The store can be the least reliable element of a microprocessor system. R A Comley describes techniques for detecting and correcting errors in semiconductor memories.

The economic advantages offered by increased packing densities of semiconductor storage devices have been partially offset by doubts over their reliability. However, it is shown that a reliability improvement of 70–80% can be achieved by using relatively simple error detection circuits.

With microprocessors being used in growing numbers for dedicated applications, the need for a method of improving system reliability takes on increasing importance. Microprocessor-based systems are frequently expected to function for many hours over periods of years without error, in what are often very hostile environments (electrically speaking).

When dealing with the reliability of these systems it is usually the microprocessor itself which receives most attention. A great deal of work has been carried out on multi-processor designs aimed at improving overall system reliability. A more subjective view of the situation, however, shows that the microprocessor need not be, and indeed probably is not, the least reliable element of the system.

RELIABILITY CONSIDERATIONS

The reliability of a device can be considered as a function of the number of external connections to the chip, the total power dissipation (as running at high temperatures accelerates ageing) and the internal complexity and layout of the chip, not as a function of itself but due to such problems as localized 'hot-spots', for example.

Most microprocessor systems consist of the microprocessor, a store and some means of I/O (input/output). Reliability of the I/O channel is very much a function of the I/O configuration and is hence specific to a particular system. As a result, only the store and microprocessor will be considered and a comparison of their relative reliabilities made.

Microprocessors, in general, dissipate more power than memory devices and are certainly more complex internally. However, by far the least reliable parts of any device are the connections from the chip to the outside world via fine gold wires and external pins. Most manufacturers have standardized on a 40-pin package for the microprocessor. The number of pins associated with the store is dependent upon the number of memory devices used to make up that store. Clearly, as the number of memory devices used increases so does the number of external connections to

the memory chips. The reliability of the store is thus a direct function of the number of memory devices employed. As the store size increases, memory reliability will become the limiting factor on the overall system reliability. Take as a typical example a 4k X 16-bit store built from 4k X 1-bit devices housed in 16-pin packages. The total number of external connections is 256, a great deal more than for the microprocessor.

Hence a method of improving the reliability, or data integrity, of the store will enhance the overall system reliability. It is possible, by placing error-correcting hardware on the data highway between the store and CPU, to eliminate single-bit errors from the store for a moderate increase in circuit complexity and cost. Further, an approach of this nature also goes a long way towards producing a memory system whose reliability is independent of memory size and individual device reliabilities. This can be a very important consideration since the reliability of individual devices will not only vary with age, but also from manufacturer to manufacturer and between batches from the same manufacturer.

Variations in manufacture can lead to such problems as pattern sensitivity, in which the memory chip only fails when a certain data pattern, or sequence of data patterns, is applied. Faults of this nature are very difficult to locate as they are not easy to reproduce using memory-test routines.

ERROR-CORRECTION SYSTEMS

There are various well established methods of improving data integrity within a system, all of which have their roots in the communications field¹. They all consist of adding check bits to each data word as it is transmitted and then using the extra information to check the received data word (including the check bits). The term 'redundancy' is often used to describe this method of error correction which, in the author's opinion, is an unfortunate misnomer since, although the additional bits convey no useful information as far as the meaning of the data word is concerned, they are essential for the checking process.

Few methods employed in the communications field are of use in digital systems, since they are designed for serial as opposed to parallel data paths, which are the norm for digital equipment. Modifications to some of the basic ideas, however, can produce effective solutions for parallel data paths.

SIMPLE PARITY CHECK

The simple parity check is the most basic method available and consists of performing a count on the number of 1s on the data highway at a given time. An extra bit is

Department of Electrical and Electronic Engineering, The City University, St John Street, London EC1V 4PB

Data word							Even parity	Odd parity
1	0	1	1	0	0	1	0	1
0	1	1	0	1	0	0	1	0
0	0	0	0	0	0	0	0	1

Figure 1. The simple parity check

generated and added to the data word to make the count either even or odd, depending upon the convention employed (see Figure 1). The use of an even parity check has the advantage of giving an all-zero check word for an all-zero data word. Single integrated circuits are available which contain all of the necessary circuitry to perform an 8-bit parity check and so a system may be very easily and cheaply implemented. For example, consider Figure 2 in which a 16-bit RAM (random access memory) with parity checking is shown.

Two 8-bit parity generators are cascaded to perform the 16-bit check and to generate a 17th data bit (DI/O 16) which is added to the data word when it is written to the store.

When the data are read from store, the 16 data-out bits are applied to the inputs of the same parity generators and the output is compared with the 17th data bit (check bit) added during the write operation. If the two are the same (i.e. no error) then the output from the exclusive-OR gate is zero, but if an error exists (i.e. 1 is output) an interrupt is generated indicating to the CPU that an error has arisen.

Note that the interrupt output is only enabled during the read cycle via the ENABLE input. This signal must not be given until after the data read from store has become stable on the data bus. If this precaution is not taken there is a possibility that 'hazard' conditions will cause false interrupts to be issued. This should not present any problems as there will normally be a signal available in the system which is used to inform the CPU that the requested data is stable on the bus. Any signal of this type may be conveniently used to enable the interrupt line.

As can be seen, very little additional circuitry is required to implement a parity checking system; for the 4k X 16-bit memory described all that is required are two parity generators, one RAM chip, one exclusive-OR and one NAND gate. For an 8k system, two RAM packages, two exclusive-OR gates and a three-input NAND gate would be required, and so on.

In addition to the basic hardware shown in Figure 2, an interrupt vector address and a 'daisy chain' for the interrupt request line must be supplied. These are fairly standard pieces of circuitry of few components and should present no problems.

The need for automatic correction

The simple parity check suffers from the major disadvantage that it can only detect single errors. If two or more errors arise simultaneously then they may go undetected by the parity logic. Further, if an error is detected the entire system must be shut down, which in some cases may be very inconvenient and in others simply not possible. What is needed for these situations is a method of detecting

and automatically correcting any errors which may arise during normal operation.

Various other methods of parity checking are often employed in an attempt to improve the effectiveness of the method. A typical example is that of performing an even parity check on all even bits of the data word and an odd parity check on the odd bits. These are reported to be more successful than the straightforward approach described but the author remains sceptical since there is just not sufficient information available to detect with any certainty anything other than single-bit errors, no matter how the parity check is arranged.

The system to be described is based on the well established concept of Hamming codes² and is capable of detecting single or multiple errors in either the data or check word. Further, it can apply automatic correction to the data word for all single-bit errors, only causing an error interrupt when multiple errors are detected.

THEORETICAL BACKGROUND

Codes may be classified by their Hamming distance which, in turn, indicates the ability of the code to detect and correct bit errors. As an example, consider the eight possible combinations of three bits, shown on an n cube in Figure 3. Taking the groups in the order ABCDEFGH gives a Grey code, in which each code differs from the next by one digit position. The Hamming distance in this case is 1 and no error detection is possible since any error gives a wrong but still acceptable code.

Now consider a code formed using only the four groups corresponding to points ACE and G. To change from one group to the next, two bit positions must be changed. The Hamming distance is now two and using this code single errors may be detected since any code with a single error will not form one of the acceptable groups (000, 011, 110 or 101). Correction is still not possible, however, since there is no means of determining whether the received group is one ahead or one behind the required code. To detect and correct a single error, a code of distance three or more must be used, i.e. a minimum of three check digits is required in this case.

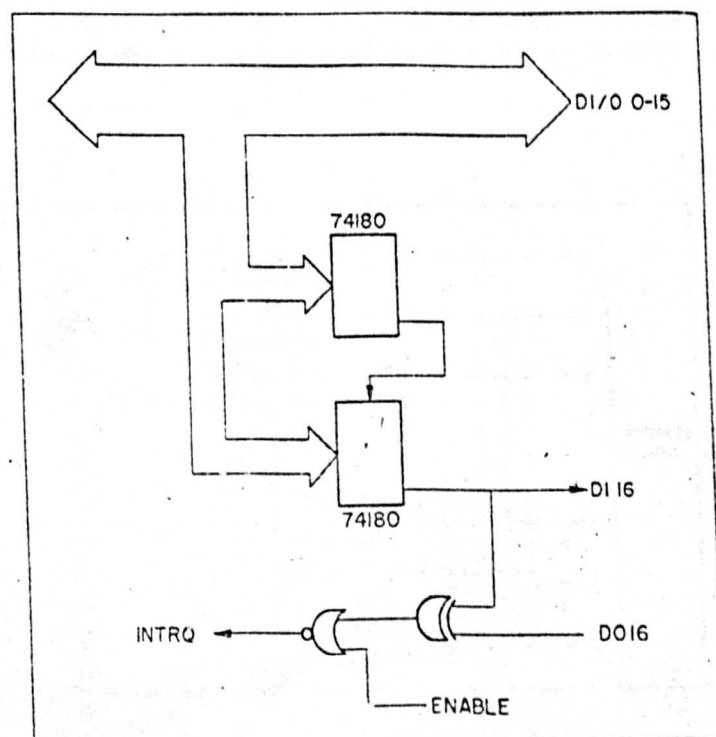


Figure 2. 16-bit parity check system

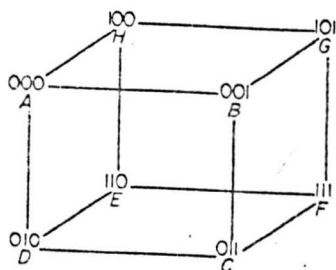


Figure 3. n -cube for three bits

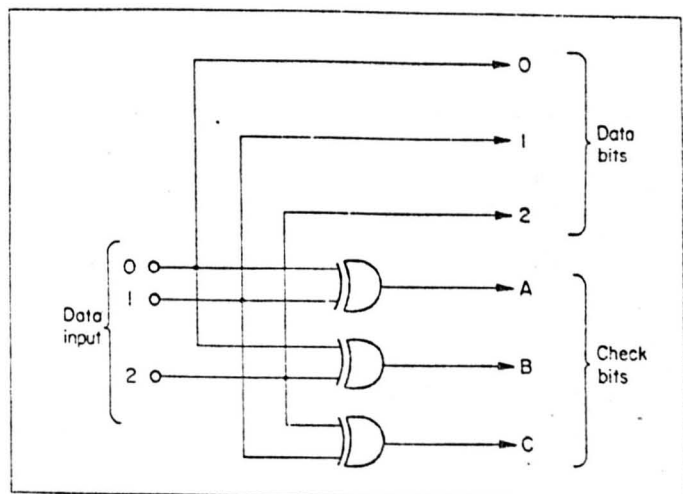


Figure 4. Encoder circuit

Now, any systematic code can be described by its parity check matrix P , of n columns, each corresponding to one of the n digits of the code, and k rows each corresponding to one of the k parity check bits³. The elements of the matrix are 0s and Xs, the position of the Xs in the i th row indicating which digit positions are involved in the parity check. Similarly the positions of the Xs in the j th column indicate to which parity checks the j th digit contributes.

Considering the example of Figure 3, the parity check matrix could be

$$P = I \begin{bmatrix} X & 0 & X & 0 & X & 0 \\ 0 & X & X & 0 & 0 & X \\ 0 & 0 & 0 & X & X & X \end{bmatrix} \begin{matrix} k \text{ rows} \\ n \text{ columns} \end{matrix}$$

The fact that each column is different from all others ensures that any single digit error will cause a unique pattern of parity violations, e.g. an error in the third digit will cause the first and second parity checks to be violated, no other single error can produce the same effect.

Clearly a matrix of k rows can have up to $2^k - 1$ columns corresponding to the set of k -digit binary numbers (except 00...0 which would present no parity check on that column). Hence, the minimum number of check digits needed for an m -digit word is the smallest number satisfying

$$2^k - 1 > m + k$$

Note that the number of check digits required increases slowly as m increases.

Coding methods

The ordering of the rows and columns must now be considered; at first sight any ordering appears equally acceptable. The ordering of the rows and columns can have a considerable effect on the complexity of the encoding and decoding circuitry, however, and so a more systematic approach is required if the additional hardware requirements are to be minimized.

As mentioned earlier, redundancy techniques were traditionally employed in communication channels where the data word is in serial form. Hence, the normal checking codes are cyclic and make use of shift-register circuits as encoders and decoders. These are of little use in digital circuits where parallel data highways are normally used. To produce a code suitable for a parallel system it is useful to consider the parity check matrix divided into columns of data and check bits

$$P = \begin{array}{c|ccc} & \text{data bits} & & \text{check bits} \\ \hline & X & X & 0 & X & 0 & 0 \\ & X & 0 & X & 0 & X & 0 \\ & 0 & X & X & 0 & 0 & X \end{array}$$

If the check bits are organized as a diagonal matrix (as shown) this can considerably simplify the encoding circuit and further, if the number of parity checks performed on each column can be minimized, the hardware required for the decoder circuit may also be minimized. For the code given above, the encoder and decoder circuits are as shown in Figures 4 and 5.

Considering the 4k X 16-bit memory example, used for the simple parity check circuit, we find that a minimum of five check bits are required to detect and correct single-bit errors ($m = 16$ so $2^k > 17 + k$, i.e. $k = 5$). A suitable parity check matrix is as shown in Figure 6 and the implementation of this code for one check digit (A) is as shown in Figure 7. Clearly five such circuits are required for a complete implementation and further, from the matrix, it can be seen that ten two-input and six three-input NOR gates will be required to provide the necessary 16 corrector outputs.

Note that it is not possible to keep the number of check digits to eight or below on all of the rows, but it is possible

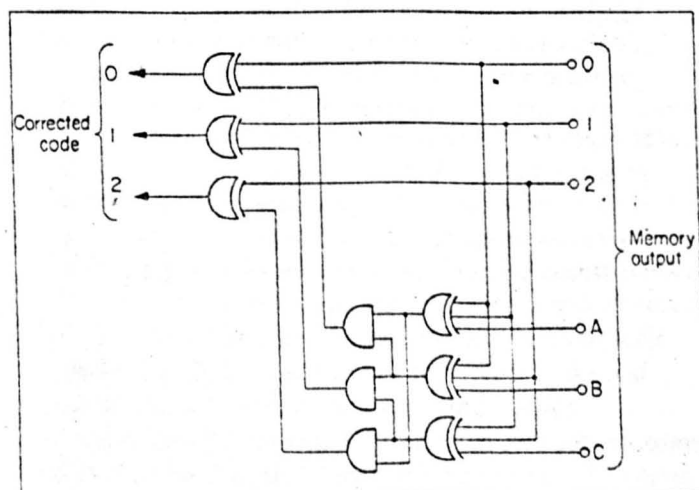


Figure 5. Decoder/corrector circuit

$$P = \begin{bmatrix} X & X & X & X & X & X & X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 \\ X & X & X & X & 0 & 0 & 0 & 0 & X & X & X & X & 0 & 0 & 0 & 0 & 0 & X & 0 & 0 & 0 \\ 0 & X & 0 & 0 & X & 0 & 0 & X & X & 0 & 0 & X & X & X & X & 0 & 0 & 0 & X & 0 & 0 \\ 0 & 0 & X & 0 & 0 & X & 0 & 0 & 0 & X & 0 & X & X & 0 & X & X & 0 & 0 & 0 & X & 0 \\ 0 & 0 & 0 & X & 0 & 0 & X & X & 0 & 0 & X & 0 & 0 & X & X & X & 0 & 0 & 0 & 0 & X \end{bmatrix}$$

Figure 6. Parity check matrix for a 16-bit data word

to limit them to nine. The extra digit can be catered for by taking the appropriate check digit output in via the even parity carry input. For rows four and five, the eighth check input may be used.

Output conditions other than those indicated in Figure 6 are possible ($2^k - 1 = 31$ possible states) and these all represent multiple or other error conditions and additional circuitry is required to decode these states and generate an interrupt should they arise. A complete list of output conditions is given in Figure 8.

The error states may be logged automatically⁴, should this be required, by providing a suitable storage area. Only the check code and error address need be recorded (21 bits in this case) since these provide all the information required to identify the position and nature of the error. Further, as the error states should arise very infrequently, a fairly small logging store will suffice. Including automatic logging can be a very useful aid to detecting 'rogue' memory cells, or packages, as these will produce frequent error outputs.

Reliability improvement with error correction

To provide a quantitative assessment of the improvement possible by including error checking, two commonly occurring parameters must be defined

- Failure rate (λ) is the average percentage of all devices that can be expected to fail per unit of time. Failure rates are usually expressed in percent per thousand hours. Further, semiconductor devices exhibit changing failure rates with time, following a well known reliabili-

ty curve(Figure 9). As a result it is also usual to specify a time frame with the failure rate.

- MTBF (mean time between failures) is the reciprocal of failure rate.

Considering our 4k X 1-bit RAM as a typical example, IC manufacturers' data⁵ indicates that λ for a 4k RAM after a few thousand hours of operation is between 0.05% and 0.2% per 1000 hours. Taking the worst case figure, this represents a MTBF of once every 2.5 years (approximately) assuming an 8h, 5-day working week. If there are 16 such RAMs in the memory system, this figure reduces to approximately one failure every two months!

Other data, collected by RAM manufacturers and Hewlett-Packard⁶ indicate that a large proportion, approximately 75% to 80%, of 4k RAM failures are single-bit failures. Hence, including a single-bit error checking scheme can considerably improve the reliability of the store and thereby the overall system.

Other considerations

The positioning of the error-correcting circuitry can have an effect on the increase in reliability offered by the systems discussed. If, for example, the processor and store are constructed on different boards, the encoder and decoder/corrector circuits should be placed on the CPU board. This will facilitate the elimination of any errors which may arise due to edge-connector problems and noise pick-up over the length of the data bus.

Another important consideration is that of the reliability of the components used in the checking circuits. If the checking logic becomes large and complex it may degrade the overall system performance rather than enhance it. Hence, the encoder-decoder/corrector circuits should be kept as simple as possible. Further, the error-correcting circuits will require a finite settling time, which will increase with increasing circuit complexity.

The settling delay should only be of the order of 30 – 50ns for the circuits described and this will not normally present any problems, since it should be easily absorbed into the settling time allowed for the data bus during memory access operations. It may, however, become a problem for systems utilizing high-speed memory devices, e.g. bipolar memories with typical access times of 70 – 90 ns. For such cases the encoder-decoder/corrector circuits should be constructed from a high-speed logic family (e.g. ECL – emitter-coupled logic). This will reduce the settling time to a few nanoseconds.

ROM and PROM-based systems can also present some problems for the reliability-checking circuits. The main reason for this is that ROMs and PROMs are byte rather than bit organized; most RAMs are bit organized. This

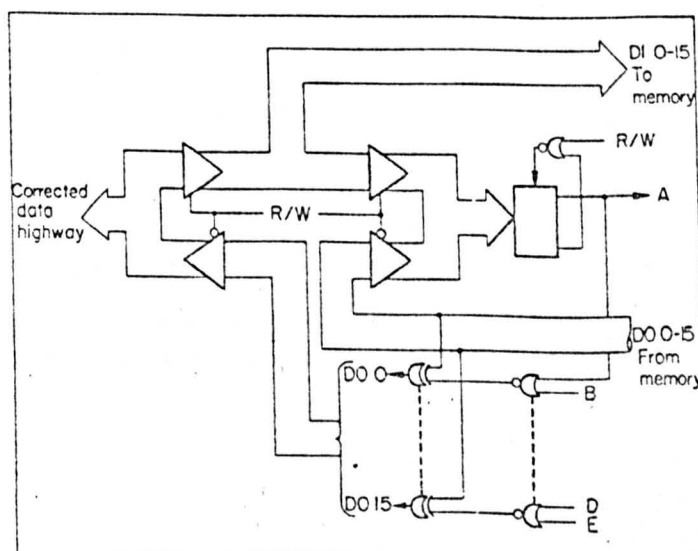


Figure 7. Error detector/corrector for 16-bit memory system

Bit in error	Output code				
	A'	B'	C'	D'	E'
Bit 0	1	1	0	0	0
Bit 1	1	1	1	0	0
Bit 2	1	1	0	1	0
Bit 3	1	1	0	0	1
Bit 4	1	0	1	0	0
Bit 5	1	0	0	1	0
Bit 6	1	0	0	0	1
Bit 7	1	0	1	0	1
Bit 8	0	1	1	0	0
Bit 9	0	1	0	1	0
Bit 10	0	1	0	0	1
Bit 11	0	1	1	1	0
Bit 12	0	0	1	1	0
Bit 13	0	0	1	0	1
Bit 14	0	0	1	1	1
Bit 15	0	0	0	1	1
Check bit A	1	0	0	0	0
Check bit B	0	1	0	0	0
Check bit C	0	0	1	0	0
Check bit D	0	0	0	1	0
Check bit E	0	0	0	0	1
Other Errors (multiple)	1	0	1	1	0
	1	1	1	1	0
	0	1	1	0	1
	1	1	1	0	1
	1	0	0	1	1
	0	1	0	1	1
	1	1	0	1	1
	1	0	1	1	1
	0	1	1	1	1
	1	1	1	1	1
Error-free output	0	0	0	0	0

Figure 8. Possible output codes and their meaning

makes the task of adding the extra storage, required for the check-bits, a little more complicated and will generally mean that some storage space is wasted. This disadvantage, however, is offset by the fact that ROM and PROM-based systems do not require any encoder circuitry.

CONCLUSIONS

Some form of error checking is desirable for most systems, and, as this paper has shown, the additional cir-

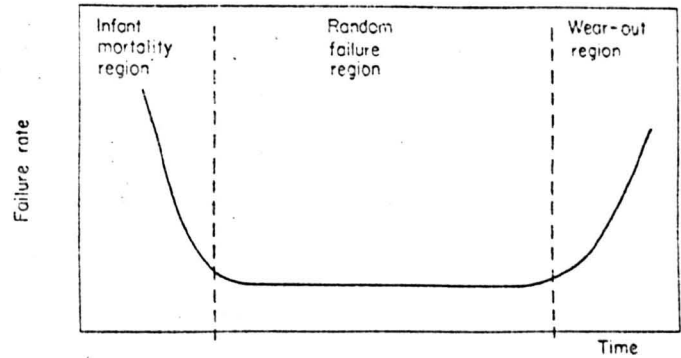


Figure 9. Semiconductor reliability life curve

cuit requirements can be quite small. For most applications the simple parity-check arrangement will be quite adequate. The full checking and corrector circuits become important where higher data integrity is required or for any application using relatively large amounts of storage.

Providing error-correcting circuits for the store is not, however, a complete answer to all reliability problems, since other single-bit errors arising in the system may still cause catastrophic failures (e.g. an erroneous address may cause error-free but incorrect data to be supplied). In situations where this is an important consideration, a checking system, similar to those described, should be applied to the address bus and multiprocessor schemes implemented.

Including a store-checking scheme does, however, considerably increase the reliability of what can be the least reliable element of a microprocessor system.

ACKNOWLEDGEMENTS

The author would like to acknowledge the support of the UK Science Research Council by the provision of a maintenance grant.

REFERENCES

- Peterson, W W *Error correcting codes* Wiley, New York (1961)
- Hamming, R W 'Error detecting and error correcting codes' *BSTJ* Vol 29 (1950) pp 147-160
- Kautz, W H 'Codes and coding circuitry for automatic error correction within digital systems' Wilcox, R H and Mann, W C. (Eds) *Redundancy techniques for computing systems* Spartan Books, Washington (1962)
- Toschi, A E and Watanabe, T 'An all-semiconductor memory with fault detection, correction and logging' *Hewlett-Packard J.* (August 1976)
- Pascoe, W '2107A/2107B N-channel silicon gate MOS 4k RAM' Reliability Report RR-7 (September 1975)
- Palfi, T L 'MOS memory system reliability' *Semiconductor Test Symposium* IEEE (1975)

Appendix D

Program Listings

i) Autocorrelation Program

ii) Spike Detector Program

REL CAUTOC

;;;;

;

AUTOCORRELATION ROUTINE

;

$RXX(T) = 1/N \cdot \sum (X(I) \cdot X(I-T))$

;

CALCULATIONS:- 100 RXX(T) VALUES FROM T = 0

;

TO T = 99

;

SAMPLING FREQU. = 200HZ

;

2MIN 44SECS (APPROX) RECORD

;

N = 32768

;

VALUES TAKEN FROM CASSETTE

;

RESULTS:- STORED IN MAIN MEMORY IN BUFFER RXXT

;

;;;;

R0 = 0

R1 = 1

R2 = 2

R3 = 3

R4 = 4

R5 = 5

SP = 6

PC = 7

TBUF = 1000

RBUF = 1200

RXXT = 1600

MEAN = 2000

TOTL = 2010

STAK = 2020

 GLOB CAUTOC

CAUTOC MVII STAK,SP

 CLRR R0

 MVII MEAN,R5

 MVOO R0,R5

 MVOO R0,R5

 MVII TOTL,R5

 MVOO R0,R5

CRBF MVII RBUF,R5

 MVOO R0,R5

 CMPI RBUF+400,R5

 BLE CRBF

;;;;

;

 CASSETTE SET-UP

;

;;;;

 MVII 20,R0

 MVO R0,167742

WATS BEXT BFIL,1

 B WATS

;DELAYED I/P VALUES

;RESULTS BUFFER

;RXX(T) VALUES

;MEAN VALUE

;LOOP COUNT

;STACK START ADDR

;SET-UP STACK POINTER

;CLEAR MEAN REGS

;CLEAR LOOP COUNT

;CLEAR RESULTS BUFFER

;SET FOR READ : HI-SPEED, INC

;WRITE CONTROL WORD

;ALLOW SET-UP DELAY

;;;		
;		
;	FILL BUFFER	
;		
;;;		
BFIL	MVII TBUF,R5	
CONT	MVO R0,167744	;CASSETTE READ REQUEST
WAIT	BEXT READ,2	;WAIT FOR IDAY
	B WAIT	
READ	MVI 167745,R0	;READ BYTE
	ANDI 377,R0	;CLEAR TOP BITS
	SLR R0,1	;LIMIT I/P TO 7-BITS
	MV00 R0,R5	;STORE VALUE
	CMPI TBUF+144,R5	
	BLT CONT	;BUFFER FULL?
;;;		
;		
;	FETCH X(1) VALUE	
;		
;;;		
	MVII TBUF,R5	
CNT2	MVO R0,167744	;CASSETTE READ RQ
WAT2	BEXT RED2,2	;WAIT FOR IDAY
	B WAT2	
RED2	MVI 167745,R0	;READ BYTE
	ANDI 377,R0	
	SLR R0,1	;LIMIT I/P TO 7-BITS
	MV00 R0,R5	;STORE VALUE
	CMPI TBUF+144,R5	
	BLT CMAN	;TOP OF BUFFER?
	MVII TBUF,R5	
;;;		
;		
;	UPDATE MEAN TOTAL	
;		
;;;		
CMAN	MVII MEAN,R3	;SET POINTER
	MVI0 R3,R1	;FETCH RUNNING TOTAL
	INCR R3	
	MVI0 R3,R2	
	ADDR R0,R1	;ADD PRESENT X(1) VALUE
	ADCR R2	;ADJUST FOR CARRY
	MV00 R2,R3	;STORE RESULT
	DECR R3	
	MV00 R1,R3	
;;;		
;		
;	CALC RXX(T) VALUES	
;		
;;;		
	PSHR R5	;SAVE PRESENT TBUF PNTR
	MVII RBUF,R3	;SET RESULTS BUFFER PNTR
CRXT	MVI0 R5,R1	;FETCH DELAYED VALUE
	PSHR R0	;SAVE X(1) VALUE

```

; ; ; ;
;
; MULTIPLY ROUTINE
;
; ; ; ;
      PSHR R5
      TSTR R1
      BPL MUL0
      NEGR R0
      BOV MUL3
      NEGR R1
      BOV MUL4
MUL0  MOV R0,R2
      CLRR R0
      MVII .17,R5
MUL1  SARC R0
MUL2  RRC R1,1
      DECR R5
      BZE MUL5
      BNC MUL1
      ADDR R2,R0
      BNOV MUL1
      RRC R0,1
      B MUL2
MUL3  MOV R1,R0
      NEGR R0
MUL4  CLRR R1
MUL5  PULR R5
      MVI R3,R0
      INCR R3
      MVI R3,R2
      ADDR R1,R0
      ADCR R2
      MVO R2,R3
      DECR R3
      MVO R0,R3
      PULR R0
      INCR R3
      INCR R3
      CMPI RBUF+310,R3
      BGE FTNX
      CMPI TBUF+144,R5
      BLT CRXT
      MVI TBUF,R5
      B CRXT
; ; ; ;
;
; CHECK FOR ALL PNTS CALC'D
;
; ; ; ;
FTNX  MVII TOTL,R5
      MVI R5,R1
      CMPI 4000,R1
      BGE FINS
      INCR R1
      DECR R5
      MVO R1,R5
      PULR R5
      B CNT2
      ;SAVE PNTR
      ;CHK MULTIPLIER SIGN
      ;BRANCH IF POS
      ;NEGATE MULTIPLICAND
      ;MULTPLICAND TO R2
      ;INIT PARTIAL PROD
      ;LOOP COUNT
      ;SHFT MS PART PROD
      ;SHFT LS PART PROD
      ;DECR LOOP COUNT
      ;FINISHED?
      ;CHK SHFTD OUT BIT
      ;ADD M/CAND TO P.P.
      ;CHK FOR OVERFLOW
      ;SHFT IN CARRY
      ;LARGEST NEG NO.
      ;RESTORE PNTR
      ;FETCH RUNNING TOTAL
      ;ADD NEW VALUE
      ;CARRY ADJUST
      ;SAVE RESULT
      ;RESTORE X(1) VALUE
      ;MOVE PNTR TO NEXT VALUE
      ;TOP OF BUFFER?
      ;TOP OF I/P BUFFER?
      ;RESET PNTR
      ;FETCH RUNNING TOTAL
      ;ALL VALUES CALC'D
      ;INCR TOTAL
      ;RESET TBUF PNTR
      ;READ NEXT X(1) VALUE

```

;
; CHECK FOR ALL PNTS CALC'D

```

FTNX      MV11 TOTL,R5
          MV10 R5,R1
          CMPI 40000,R1
          BGE FINS
          INCR R1
          DECR R5
          MVO@ R1,R5
          PULR R5
          B CNT2

          ;FETCH RUNNING TOTAL
          ;ALL VALUES CALC'D
          ;INCR TOTAL
          ;RESET TBUF PNTR
          ;READ NEXT X(1) VALUE

```

```

; ; ; ;
;
;   CALC FINAL RESULTS
;
; ; ; ;
FINS  MVII MEAN,R3           ;CALC MEAN VALUE
      MVI@ R3,R0
      INCR R3
      MVI@ R3,R1
      SLL R1,1
      SLLC R0,1             ;SHIFT HI-WORD
      ADCR R1               ;SHIFT LO-WORD
                              ;R1 = X(MEAN)
      MOVR R1,R0
      MOVR R0,R2
      CLRR R0
      MVII .17,R5
      SARC R0               ;MULTIPLICAND TO R2
                              ;INIT P.P.
MLT1  RRC R1,1             ;SHFT MS P.P.
MLT2  DECR R5
      BZE MLT5
      BNC MLT1
      ADDR R2,R0
      BNOV MLT1
      RRC R0,1
      B MLT2
      MVII RBUF,R5
      MVII RXXT+144,R3
      MVI@ R5,R0
      MVI@ R5,R2
      SLL R2,1
      SLLC R0,1
      ADCR R2
      SUBR R1,R2
      MVO@ R2,R3
      DECR R3
      CMPI RXXT,R3
      BGE NEXT
      HLT
      HLT
      NOP
      NOP
      END

```

```

;CALC MEAN VALUE

;SHIFT HI-WORD
;SHIFT LO-WORD
;R1 = X(MEAN)

;MULTIPLICAND TO R2
;INIT P.P.

;SHFT MS P.P.

;FINISHED?
;CHK SHFTED OUT BIT
;ADD M/CAND TO P.P.

;SHFT IN CARRY

;SET PNTRS

;FETCH RXX(T) VALUE

;R2 = RXX(T) RESULT
;SUBR MEAN
;STORE RESULT
;SHFT PNTR

;ALL VALUES CALC'D?

```

```

REL EDFPAC
; ; ; ;
;
; FIRST ORDER DIFFERENCE FOLLOWED BY A
; SECOND ORDER BUTTERWORTH FILTER
; WS = 500HZ WC = 50HZ
;
; ARTIFACT REJECTION ROUTINE
;
; SPIKE DETECTION ROUTINE
;
; OUTPUT TO CASSETTE
; - EVERY OTHER SAMPLE
; ; ; ;
R0 = 0
R1 = 1
R2 = 2
R3 = 3
R4 = 4
R5 = 5
SP = 6
PC = 7
FLAG = 1000
DIFP = 1002
PNTR = 1010
LABL = 1020
STOR = 1030
OPST = 1040
PKFL = 1041
CNTA = 1043
CNTB = 1044
ARTB = 1050
ARTP = 1250
ARTC = 1252
GLOB EDFPAC
EDFPAC CLRR R0
      MVII FLAG,R5
      MVO@ R0,R5
      MVO@ R0,R5
      MVII ARTB,R1
      MVO R1,ARTP
      CLRR R1
      MVO R1,ARTC
      MVII ARTB,R5
CART  MVO@ R1,R5
      CMPI ARTB+.125,R5
      BLE CART
      MVII 000017,R0
      MVO R0,167742
WWAT  BEXT DFIL,1
      B WWAT
DFIL  MVI 167732,R0
      ANDI 000377,R0
      MVII STOR,R5
      MVO@ R0,R5
      SLL R0,2
      MVII DIFP+1,R5
      MVI@ R5,R1
      DECR R5
      MVO@ R0,R5
      SUBR R1,R0
;FLAG WORD
;DIFF VALUES
;DELAYED X VALUES
;DELAYED Y VALUES
;INPUT DATA STORE AREA
;OUTPUT COUNT
;PEAK FLAGS
;COUNT A
;COUNT B
;ARTIFACT BUFFER BASE
;ARTIFACT BUFFER PNTR
;ARTIFACT COUNT
;CLEAR OUTPUT FLAG
;CLEAR PEAK FLAG
;INITIALISE ARTF. PNTR.
;CLEAR CROSSING COUNT
;CLEAR ARTIFACT BUFFER
;SET CASS FOR CONT WRITE - 14IPS
;WRITE CONTROL WORD
;WAIT FOR IRDY
;INPUT X(N) VALUE
;CLEAR TOP BITS
;STORE X(N) VALUE
;SCALE INPUT
;LOAD X(N-1) ADDR
;FETCH X(N-1) VALUE
;STORE NEW X(N-1) VALUE
;R0 = X(N) - X(N-1)

```

```

; ; ; ;
;
; FILTER ROUTINE
;
; ; ; ;
FILT CLRR SP                ;CLEAR SIGN FLAG
      TSTR R0                ;SET STATUS
      BPL RESU                ;+VE VALUE?
      NEGR R0                ;2'S COMP
      INCR SP                ;SET SIGN FLAG
RESU  SWAP R0,1                ;SCALE INPUT
      SLR R0,2
      SLR R0,1
      TSTR SP
      BZE CRES                ;CHECK SIGN FLAG
      NEGR R0                ;FLAG NOT SET?
      CLRR SP                ;2'S COMP
      MVI PNTR,R5            ;CLEAR SIGN FLAG
CRES  MVI R5,R1                ;SET X VALUES BASE
      MOVR R1,R3            ;FETCH X(N-1) VALUE
      BPL POS2                ;SAVE VALUE
      INCR SP                ;+VE VALUE
      NEGR R1                ;SET SIGN FLAG
POS2  SLL R1,1                ;2'S COMP
      TSTR SP                ;MULT BY 2
      BZE NEXT                ;CHECK SIGN FLAG
      NEGR R1                ;FLAG NOT SET?
      CLRR SP                ;2'S COMP
NEXT  MVI R5,R2                ;CLEAR SIGN FLAG
      ADDR R0,R1            ;FETCH X(N-2) VALUE
      BOV STOP                ;R1 = X(N) + 2X(N-1)
      ADDR R2,R1            ;OVERFLOW?
      BOV STOP                ;R1 = X VALUES COMP
      MVI PNTR,R5            ;OVERFLOW?
      MVO R0,R5
      MVO R3,R5
      MOVR R1,R0
      BPL POS1                ;STORE NEW X(N-1) VALUE
      INCR SP                ;STORE NEW X(N-2) VALUE
      NEGR R0                ;MOVE RESULT
POS1  SLR R0,2                ;+VE INPUT?
      SLR R0,2                ;SET SIGN FLAG
      MOVR R0,R1            ;2'S COMP
      SLR R0,2                ;4 SHIFTS
      SLR R0,2                ;SAVE RESULT
      ADDR R0,R1            ;8 SHIFTS
      SLR R0,2                ;ADD TO TOTAL
      ADDR R0,R1            ;10 SHIFTS
      SLR R0,2
      SLR R0,2
      ADDR R0,R1
      TSTR SP
      BZE NVAL                ;14 SHIFTS
      NEGR R1                ;R1 = 0.0675 X COMP
      CLRR SP                ;CHECK SIGN FLAG
NVAL  MOVR R1,R3            ;FLAG NOT SET?
                        ;2'S COMP
                        ;CLEAR SIGN FLAG
                        ;SAVE RESULT

```

```

;::::;
;
;  CALC Y VALUE COMPS
;
;::::;
      MVII LABL,R5
      MVI@ R5,R0
      MOVR R0,R1
      BPL POSY
      NEGR R0
      INCR SP
POSY  MOVR R0,R2
      SLR R0,2
      SLR R0,1
      ADDR R0,R2
      SLR R0,2
      SLR R0,1
      ADDR R0,R2
      SLR R0,2
      SLR R0,1
      ADDR R0,R2
      SLR R0,1
      ADDR R0,R2
      SLR R0,1
      ADDR R0,R2
      TSTR SP
      BZE NXTY
      NEGR R2
      CLRR SP
NXTY  MVI@ R5,R0
      TSTR R0
      BPL REPS
      NEGR R0
      INCR SP
REPS  SLR R0,2
      MOVR R0,R5
      SLR R0,1
      ADDR R0,R5
      SLR R0,2
      ADDR R0,R5
      SLR R0,2
      ADDR R0,R5
      TSTR SP
      BZE INVY
      NEGR R5
      CLRR SP
INVY  SUBR R5,R2
      BOV STOP
      ADDR R2,R3
      BOV STOP
      MVII LABL,R5
      MVO@ R3,R5
      MVO@ R1,R5

```

```

;FETCH Y(N-1) VALUE
;SAVE RESULT
;+VE VALUE?
;2'S COMP
;SET SIGN FLAG
;SAVE VALUE

;3 SHIFTS
;ADD TO TOTAL

;6 SHIFTS

;9 SHIFTS
;10 SHIFTS
;12 SHIFTS
;13 SHIFTS

;14 SHIFTS
;R2 = 1.144Y(N-1)
;CHECK SIGN FLAG
;FLAG NOT SET
;2'S COMP
;CLEAR SIGN FLAG
;FETCH Y(N-2) VALUE
;SET STATUS
;+VE VALUE?
;2'S COMP
;SET SIGN FLAG
;2 SHIFTS
;SAVE RESULT
;3 SHIFTS
;ADD TO TOTAL
;5 SHIFTS

;7 SHIFTS
;R5 = 0.414Y(N-2)
;CHECK SIGN FLAG
;FLAG NOT SET?
;2'S COMP
;CLEAR SIGN FLAG
;R2 = Y COMP
;OVERFLOW?
;R3 = Y(N)
;OVERFLOW?

;STORE NEW Y(N-1) VALUE
;STORE NEW Y(N-2) VALUE

```

;;;;;

;

ARTIFACT REJECTION ROUTINE

;

;;;;;

	MOVR R3,R0	;SAVE FILTER O/P
	MOVR R3,R1	;R1 = FILTER O/P
	ANDI 100000,R1	;CLEAR ALL BUT SIGN
	MVII ARTP,R3	
	MVI@ R3,R5	;LOAD BUFFER PNTR
	MVI@ R5,R2	;FETCH N-1 SAMPLE
	SAR R2,1	;CLEAR SIGN FLAG
	CMPR R1,R2	;COMPARE SIGNS
	BZE CKBF	
	MVII ARTC,R3	
	MVI@ R3,R2	;INCR CROSSING COUNT
	INCR R2	
	MV0@ R2,R3	
	XORI 1,R1	;SET SIGN CHANGE FLAG
CKBF	CMP1 ARTB+.125,R5	
	BLE STPN	;TOP OF BUFFER?
	MVII ARTB,R5	;RESET PNTR
STPN	MV0 R5,ARTP	;STORE PNTR
	MVI@ R5,R2	;FETCH N-126 SAMPLE
	CLRC	;CLEAR CARRY BIT
	RRC R2,1	
	BNC SPRS	;SIGN FLAG SET?
	MVII ARTC,R3	
	MVI@ R3,R2	;DECR CROSSING COUNT
	DECR R2	
	MV0@ R2,R3	
SPRS	DECR R5	;MOVE PNTR BACK
	MV0@ R1,R5	;WRITE PRESENT SAMPLE
	MOVR R0,R3	;RESTORE FILTER O/P

;;;;;

;

SPIKE DETECTION ROUTINE

;

;;;;;

	TSTR R3	;SET STATUS
	BPL POSR	;+VE RESULT
	NEGR R3	;2'S COMP
	INCR SP	;SET SIGN FLAG
POSR	SLR R3,2	;RESCALE
	SLR R3,2	
	ANDI 000377,R3	;CLEAR TOP BITS
	CMP1 60,R3	;CMPR WITH +0.48
	BLE CKPS	;PEAK NOT FOUND?
	TSTR SP	;CHECK SIGN FLAG
	BZE PSPK	;+VE PEAK?
	MVII PKFL,R5	
	MVI@ R5,R1	;FETCH +VE PEAK FLAG
	TSTR R1	
	BZE NOPK	;FLAG NOT SET?
	MV0@ R1,R5	;SET -VE PEAK FLAG
	MVII CNTA,R5	
	MVI@ R5,R1	;FETCH COUNT A
	INCR R1	;INCR COUNT A
	DECR R5	
	MV0@ R1,R5	;STORE NEW COUNT A

OUPT	CLRR SP	;CLEAR SIGN FLAG
	MVII STOR,R5	
	MVIO R5,R0	;FETCH X(N) VALUE
	MVII OPST,R5	
	MVIO R5,R1	;FETCH OUTPUT COUNT
	CMPI 0,R1	
	BLE COUT	;COUNT ZERO?
	XORI 000400,R0	;SET NINTH BIT
	DECR R1	;DECR OUTPUT COUNT
	DECR R5	
	MVIO R1,R5	;STORE NEW OUTPUT COUNT
COUT	MVII FLAG,R5	
	MVIO R5,R1	;FETCH OUTPUT FLAG
	TSTR R1	
	BZE SFLG	;FLAG NOT SET
	MVO R0,167743	;OUTPUT VALUE
	CLRR R1	
	DECR R5	
	MVIO R1,R5	;CLEAR OUTPUT FLAG
	B WAIT	
SFLG	DECR R5	
	MVII 1,R1	
	MVIO R1,R5	;SET OUTPUT FLAG
WAIT	BEXT DFIL,0	;NEXT SAMPLE READY
	B WAIT	
PSPK	MVII PKFL,R5	
	MVII 1,R1	
	MVIO R1,R5	;SET +VE PEAK FLAG
	INCR R5	
	MVIO R5,R1	;FETCH COUNT A
	INCR R1	;INCR COUNT A
	DECR R5	
	MVIO R1,R5	;STORE NEW COUNT
	B OUPT	
CKPS	MVII PKFL,R5	
	MVIO R5,R1	;FETCH +VE PEAK FLAG
	TSTR R1	
	BZE NOPK	;FLAG NOT SET?
	MVIO R5,R1	;FETCH -VE PEAK VALUE
	TSTR R1	
	BZE NSET	;FLAG NOT SET?
	MVII CNTA,R5	
	MVIO R5,R1	;FETCH COUNT A
	MVII OPST,R5	
	MVIO R1,R5	;COPY COUNT A TO OUTPUT COUNT
NOPK	CLRR R1	
	MVII PKFL,R5	
	MVIO R1,R5	;CLEAR PEAK A FLAG
	MVIO R1,R5	;CLEAR -VE PEAK FLAG
	MVIO R1,R5	;CLEAR COUNT A
	MVIO R1,R5	;CLEAR COUNT B
	B OUPT	
NSET	MVII CNTB,R5	
	MVIO R5,R1	;FETCH COUNT B
	INCR R1	;INCR COUNT
	DECR R5	
	MVIO R1,R5	;STORE NEW COUNT
	CMPI 4,R1	
	BGE NOPK	;COUNT>4

STOP
MVII ARTC,R5
MVI@ R5,R0
CMP1 120,R0
BGE NOPK
B OUP T
HLT
HLT
NOP
NOP
END

;FETCH ARTIFACT COUNT

;COUNT > 80?
-

Appendix E

Error Calculations for the Second-Order Butterworth Filter

E.1 Introduction

The errors introduced into the signal as a result of the digital filtering operation are calculated in this appendix. The errors arise as a consequence of quantization effects and mean that a slightly different filter than the one desired, will be realized. The errors will manifest themselves as slight shifts in the pole-zero locations from their ideal positions and as small changes in the values of coefficients.

E.2 Cut-off Frequency Error

The shift in the pole-zero locations will mean that the cut-off frequency of the filter is altered from the desired value of $\omega_c = 50\text{Hz}$. This can be found directly from the relationship:-

$$\frac{\omega_c}{\omega_s} = \frac{1}{\pi} \tan \frac{\pi}{10} = 0.10341$$

The 0.00341 error is introduced into the ω_c/ω_s ratio as a consequence of the bilinear transformation technique [21]. The actual cut-off frequency realized is:-

$$\omega_c = \tan^{-1}(0.10341) \cdot \omega_s$$

and if ω_s is set at 500Hz:-

$$= 51.52187\text{Hz}$$

i.e. a 1.52187Hz error in ω_c .

There will be a truncation error involved in the realization of the 0.10341 coefficient, and this will tend to reduce the magnitude of the error in ω_c :-

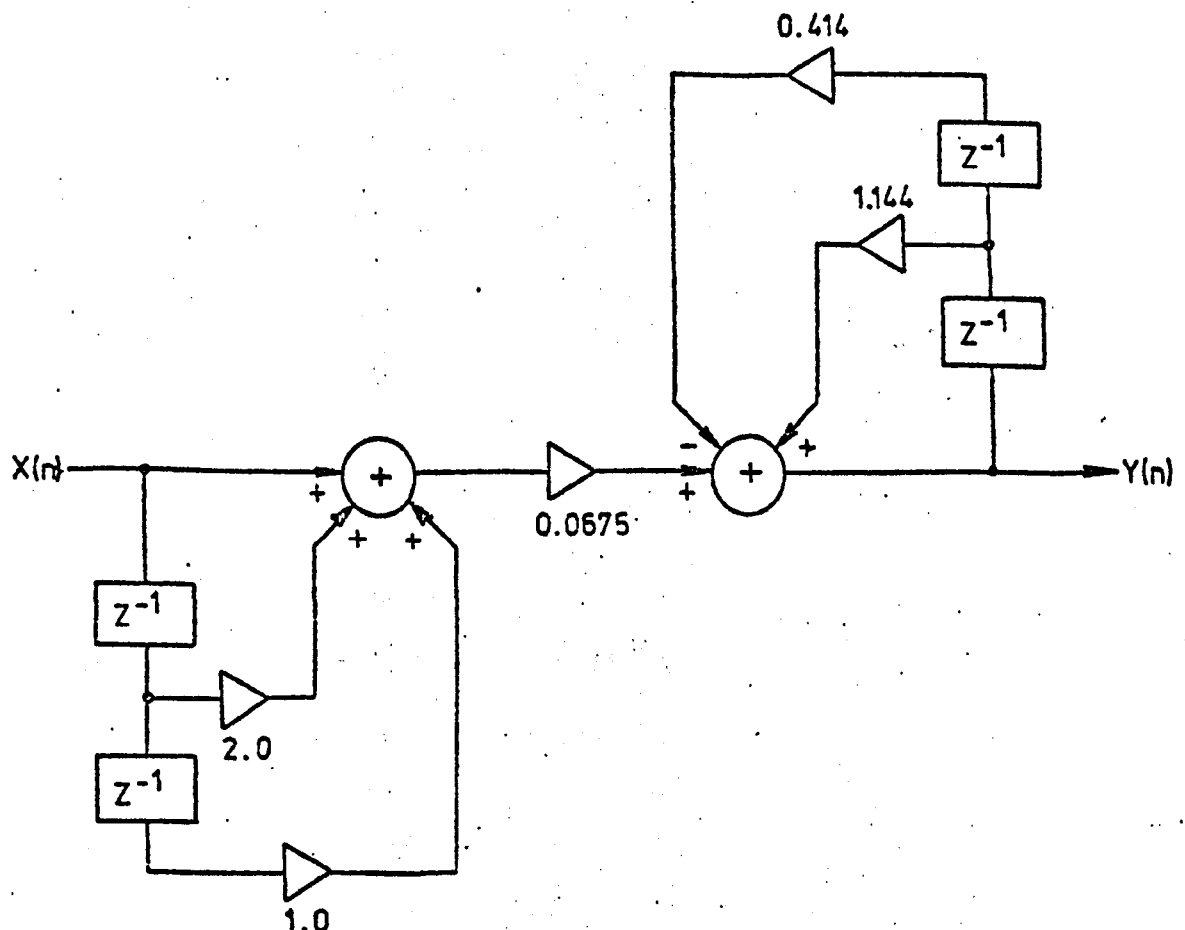
$$\begin{aligned}\omega_c &= \tan^{-1}(0.103410 - 0.000006) \times 500 \\ &= 51.5187\text{Hz}.\end{aligned}$$

As can be seen, there is an error of 1.5187Hz in the value of ω_c but this is not of major importance for the EEG

application; the amplitude response is of far greater significance.

E.3 Amplitude Error

To calculate the amplitude error, the filter configuration must be considered since the total error is dependent upon the structure used. The implementation of the second-order Butterworth filter used in the EEG analysis routine is as shown in Figure 1.E.



$$y(n) = 0.0675 [x_n + 2x_{n-1} + x_{n-2}] + 1.144y_{n-1} - 0.414y_{n-2}$$

Figure 1.E Second-order Butterworth filter implementation.

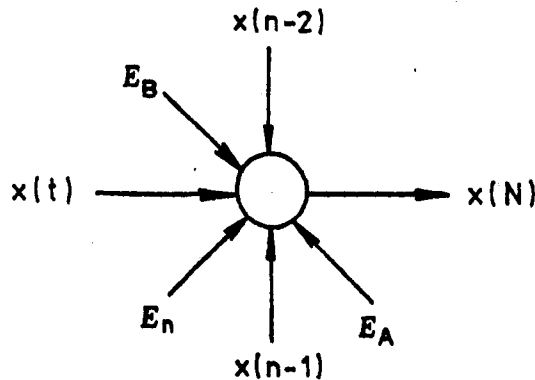
The input to the filter is provided by the output of an A-D converter and so is subject to an input quantization

error. If the input to the A-D converter is $x(n)$ and the corresponding output is $x(t)$, then:-

$$x(n) = x(t) + E_n \quad (i)$$

where E_n is the quantization error.

The error arising as a result of the x term summation is:-



Now
$$x(N) = x(n) + x(A) + x(B)$$

and
$$x(A) = \widehat{x(A)} + E_A$$

where $\widehat{x(A)}$ represents the ideal $x(A)$ value and E_A the error term. Similarly:-

$$x(B) = \widehat{x(B)} + E_B$$

The coefficients are $A = 1$ and $B = 2$ and both can be realized exactly without the introduction of any error.

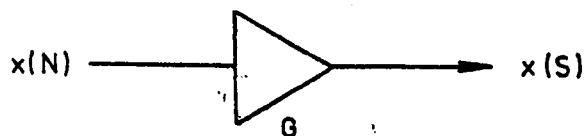
Hence, the only errors are those due to the input quantization error E_n . The E_n term represents the absolute error introduced by the conversion process and so the error due to the summation of the x terms is the sum of the quantization errors:-

$$\begin{aligned} x(N) &= [x(t) + E_n] + 2.0[x(t-1) + E_n] \\ &\quad + [x(t-2) + E_n] \\ &= x(I) + 4E_n \end{aligned} \quad (ii)$$

where $x(I)$ represents the error-free sum of the x terms and $4E_n$ the error term.

This output is now scaled by the 0.0675, or G , coeff-

icient:-



Now $x(S) = x(N).G$

where $G = \hat{G} + E_G$

So, substituting from (ii):-

$$\begin{aligned} x(S) &= [x(I) + 4E_n][\hat{G} + E_G] \\ &= x(I).\hat{G} + x(I).E_G + 4E_n.\hat{G} + 4E_nE_G \end{aligned}$$

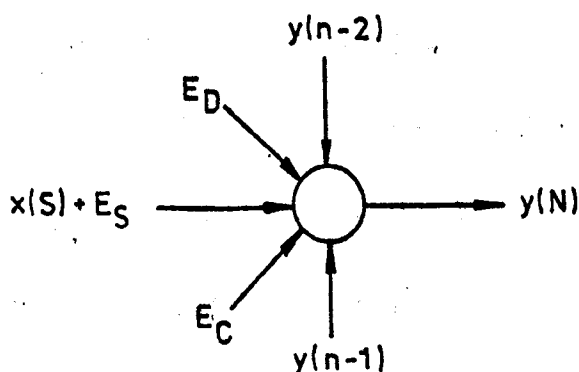
The result $x(S)$ must however be truncated to sixteen-bits and so a further error E_T must be added, which gives:-

$$\begin{aligned} x(S) &= \hat{x(S)} + E_S \\ &= x(I).\hat{G} + x(I).E_G + 4E_n.\hat{G} + 4E_nE_G + E_T \quad \text{(iii)} \end{aligned}$$

and so the error at this stage is:-

$$E_S = x(I).E_G + 4E_n.\hat{G} + 4E_nE_G + E_T$$

The error introduced by the y terms summation is:-



Now $y(N) = x(S) + y(C) + y(D) \quad \text{(iv)}$

and $y(C) = \hat{y(C)} + E_C$

where $\hat{y(C)}$ represents the ideal $y(C)$ value and E_C the error term. Similarly:-

$$y(D) = \hat{y(D)} + E_D$$

The E_C and E_D terms comprise the error involved in the definition of the coefficient and the absolute error in the

value of the output $y(N)$, since the filter is recursive. If the absolute error present at the output is E_y , then:-

$$\begin{aligned} y(C) &= [y(\hat{C}) + E_y][\hat{C} + E_C] \\ &= y(\hat{C}).\hat{C} + y(\hat{C}).E_C + E_y.\hat{C} + E_y E_C \end{aligned}$$

and again, a quantization error must be added. Similarly:-

$$y(D) = y(\hat{D}).\hat{D} + y(\hat{D}).E_D + E_y.\hat{D} + E_y E_D + E_T$$

Substituting in equation (iv):-

$$\begin{aligned} y(N) &= y(\hat{N}) + E_y \\ &= [x(I).\hat{G} + x(I).E_G + 4E_n.\hat{G} + 4E_n E_G + E_T] \\ &\quad + [y(\hat{C}).\hat{C} + y(\hat{C}).E_C + E_y.\hat{C} + E_y E_C + E_T] \\ &\quad + [y(\hat{D}).\hat{D} + y(\hat{D}).E_D + E_y.\hat{D} + E_y E_D + E_T] \end{aligned}$$

and so the total absolute error at the output is given by:-

$$\begin{aligned} E_y &= [x(I).E_G + 4E_n.\hat{G} + 4E_n E_G + E_T] \\ &\quad + [y(\hat{C}).E_C + E_y.\hat{C} + E_y E_C + E_T] \\ &\quad + [y(\hat{D}).E_D + E_y.\hat{D} + E_y E_D + E_T] \end{aligned}$$

which, ignoring the second-order terms:-

$$\begin{aligned} E_y(1 - \hat{C} - \hat{D}) &= x(I).E_G + 4E_n.\hat{G} + y(\hat{C}).E_C \\ &\quad + y(\hat{D}).E_D + 3E_T \end{aligned}$$

Substituting the corresponding values:-

$$\begin{aligned} \hat{C} &= 1.144 \\ \hat{D} &= -0.414 \\ \hat{G} &= 0.0675 \end{aligned}$$

and, for an eight-bit A-D converter the peak error is:-

$$E_n = 0.00391$$

and, setting $x(t) = 1$ and $y(\hat{N}) = 1$:-

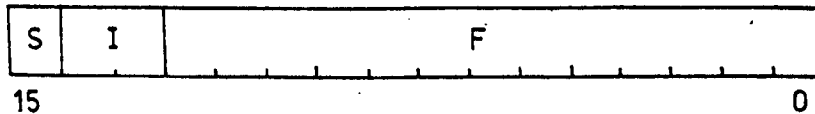
$$E_C = 0.000003$$

$$E_D = 0.000014$$

$$E_G = 0.000056$$

$$E_T = 0.000122$$

The digital word is split up as:-



for the filter calculation and so the last four error values represent the absolute errors for a thirteen-bit representation.

$$\begin{aligned}
 E_y(1 - 1.144 + 0.414) &= 4 \times 0.000056 \\
 &+ 4 \times 0.00391 \times 0.0675 \\
 &+ 1 \times 0.000003 \\
 &+ 1 \times 0.000014 \\
 &+ 3 \times 0.000122
 \end{aligned}$$

$$E_y(0.27) = 0.001665$$

$$\therefore E_y = \frac{0.001665}{0.27} = 0.006167$$

which is the absolute error in y. This error is made up of the error introduced by the filter and the error present at the input due to the A-D conversion process. If the A-D error could be reduced to the theoretical limit of zero, then the remaining error would be due to the filter alone, and would be:-

$$E_f(0.27) = 0.001665 - 0.001058$$

$$\begin{aligned}
 \text{i.e. } E_f &= \frac{0.000607}{0.27} = 0.002248
 \end{aligned}$$

This is the peak error introduced into the signal by the filter and does not take account of any frequency effects (i.e. it is the peak error for zero frequency). It is possible that error-amplification can occur at particular frequencies,

although no serious problems have been encountered in this case (see Chap. 6). In order that the error at different frequencies may be calculated it is necessary to express the error in y in the form of a first-order approximation such as:-

$$\Delta y_n(z) = (Cz^{-1} + Dz^{-2})\Delta y + (\Delta Cz^{-1} + \Delta Dz^{-2})y$$

If z is replaced by $e^{j\omega T}$, the above equation may be evaluated for various values of ω and hence $\Delta y_n(e^{j\omega T})$ found.

Chapter 12

References

1. Berger H., Uber des Eleklrenkephalogramm des Menschen, Arch. Psychiat., pp16-60, 1929.
2. Stowell H., No future in the averaged scalp, Nature, p1074, 1970.
3. Walter W. G., The location of cerebral tumours by electroencephalography, Lancet, pp305-308, 1936.
4. Rabiner L. R. and Rader C. M. (Editors), Digital Signal Processing, IEEE Press, 1972.
5. Digital Signal Processing Committee (Editors), Digital Signal Processing II, IEEE Press, 1977.
6. Cooley J. W. and Tukey J. W., An algorithm for the machine calculation of complex Fourier series, Math. Compt., Vol. 19, pp297-301, Apr. 1965.
7. Gevins A. S., Yeager C. L., Diamond S. L., Spira J-P., Zeithin G. M. and Gevins A. H., Automated Analysis of the Electrical Activity of the Human Brain (EEG): A Progress Report, Proc. IEEE, Vol. 63, No. 10, pp1382-1399, Oct. 1975.
8. Godfrey K. R. and Bruce D. M., Computer analysis of brainwaves, Electronics and Power, pp510-514 and pp607-610, Aug. and Sept. 1976.
9. Buckley J., Saltzberg B. and Heath R., Decision criteria and detection circuitry for multiple channel EEG correlation, IEEE Region 3 Rec., sec. 3.1-3.3, 1968.
10. Kontas P., Smith J. R., King R. L. and Mayersdorf A., Automated detection of abnormal spikes in the electroencephalogram, Epilepsia, Vol. 13, pp522-523, 1971.
11. Carrie J. R. G., A hybrid computer technique for detecting sharp EEG transients, Electroenceph. Clin. Neurophysiol., Vol. 33, pp336-338, 1972.

12. Smith J. R., Automatic Analysis and Detection of EEG Spikes, IEEE Trans. BME-21, pp1-7, 1974.
13. Lloyd D. L. and Binne C. D., Private communication, St. Bartholomew's Hospital, London.
14. Stevens J. R., Kodamo H., Lonsbury B. and Mills L., Ultradian characteristics of spontaneous discharges recorded by radio telemetry in man, Electroencephalogr. Clin. Neurophysiol., Vol. 31, pp313-325, 1971.
15. Gergely S. and Paul R., Delayed signal EEG trigger, Biomed. Eng., Vol. 10, pp105-109, 1975.
16. Grass A. M. and Gibbs F. A., A Fourier transform of the electroencephalogram, J. Neurophysiol., Vol. 1, pp521-526, 1938.
17. Rhodes G. M., PhD Thesis, The City University, 1971.
18. Buffam C. J., PhD Thesis, The City University, 1977.
19. Comley R. A., Super-tool: the microprocessor is revolutionising industry, Quest, The City University, to be published (see Appendix A).
20. Comley R. A., Relieving the Real-time Signal Processing Load, presented at the VII Hospital Physicists Conference, Heriot-Watt University, March 1977 (Not formally published: see Appendix A).
21. Brignell J. E. and Rhodes G. M., Laboratory On-line Computing, Intertext, London, 1975.
22. Brignell J. E., Comley R. A. and Young R., The Roving Slave Processor, Microprocessors, IPC Science and Technology Press, Vol. 1, No. 2, pp79-84, Dec. 1976. (See Appendix A).
23. Allen T. A., BSc Project, No. 973, The City University, 1977.

24. Comley R. A. and Hewish T. R., The Software Realisation of Instruments, International Microcomputers/Minicomputers/Microprocessors '77, Geneva. Proceedings published by IPC Science and Technology Press, pp41-47, 1977. (See Chap. 4.7).
25. Comley R. A. and Brignell J. E., Digital Filter Implementation by means of Slave Processors, IEE Colloquium, Digest No. 1977/50, Nov. 1977. (See Appendix A).
26. Buffam C. J., Comley R. A. and Young R., More Bits, more power, more precision, Electron, IPC Electrical-Electronic Press, pp45-46, Dec. 1977. (See Appendix A).
27. Mears B. R., PhD Thesis, The City University, 1977.
28. Tracey R. N., BSc Project, No. 968, The City University, 1977.
29. Comley R. A., Error detection and correction for memories, Microprocessors, Vol. 2, No. 1, pp29-33, IPC Science and Technology Press, Feb. 1978. (See Appendix C).
30. Young R. and Brignell J. E., An Unorthodox Approach to Microprocessor Language, IERE Conference publication No. 36, pp43-52, Computer Systems and Technology, 1977.
31. Brignell J. E., The Case for an Algebraic Assembler, Proceedings of the Technical Program International Microelectronics, ppl35-139, Brighton, 1977.
32. General Instrument Microelectronics, Series 1600 Microprocessor System - Documentation, GIC, 1976.
33. Freeman P., Software reliability and design: a survey, Proceedings of the 13th. Annual Design Automation Conference, pp484-494, San Francisco, 28-30th. June 1976.
34. Kopetz H., Error detection in real-time software, Proceedings of the 1975 IFAC-IFIP Workshop on Real-Time Programming, pp81-87, Boston Mass., 21-22nd. Aug. 1975

35. Beauchamp K. G., Signal Processing, George Allen and Unwin, London, 1973.
36. Whalen A. D., Detection of Signals in Noise, Academic Press, London, 1971.
37. Storm van Leeuwen W. et al, Proposal for an EEG terminology by the terminology committee of the International Federation for Electroencephalography and Clinical Neurophysiology, Electroencephalogr. Clin. Neurophysiol., Vol. 20, pp293-320, 1966.
38. Maulsby R. L., Some Guidelines for Assessment of Spikes and Sharp Waves in EEG Tracings, The Proceedings of the Electro-Physiological Technologists Association, Vol. 18, No. 2, pp70-81, Aug. 1971.
39. Smith J. R., Negin M. and Nevis A. H., Automatic analysis of sleep electroencephalograms by hybrid computation, IEEE Trans., SSC-5, pp278-283, 1969.
40. Campbell J., Bower E., Dwyer S. J. and Lago G. V., On the sufficiency of autocorrelation functions as EEG descriptors, IEEE Trans. BME-14, pp49-52, Jan. 1967.
41. Elul R., Gaussian behaviour of the electroencephalogram: Changes during performance of mental task, Science, Vol. 164, pp328-331, Apr.-June 1969.
42. Huber P. J., Kleiner B., Gasser T. and Dumermuth G., Statistical methods for investigating phase relations in stationary stochastic processes, IEEE Trans. Audio Electroacout., Vol. AU-19, pp78-96, March 1971.
43. Kawabata N., A Nonstationary Analysis of the Electroencephalogram, IEEE Trans. BME-20, No. 6, pp444-452, Nov. 1973.
44. Beiser A., Concepts of Modern Physics, McGraw-Hill, New York, 1967.

45. Bracewell R. N., The Fourier Transform and its Application, McGraw-Hill, 1965.
46. McGillem C. D. and Cooper G. R., Continuous and Discrete Signal and System Analysis, (pp113-114), Holt, Rinehart and Winston, New York, 1974.
47. Mick J. R., Digital Filter Design, Digital Signal Processing Handbook, Advanced Micro Devices Inc., USA, 1976.
48. Titchmarsh E. C., The Theory of Functions, Oxford, 1964.
49. Gold B. and Rader C. M., Digital Processing of Signals, McGraw-Hill, 1969.
50. Blackman R. B., Linear Data-Smoothing and Prediction in Theory and Practice, Reading Mass., Addison-Wesley, 1965.
51. Rader C. M. and Gold B., Effects of parameter quantization on the poles of a digital filter, Proc. IEEE, Vol. 55, pp688-689, May 1977.