# DIGITAL MOMENTS ANALYZER

# DIGITAL MOMENTS ANALYZER:
# DESIGN AND ERROR CHARACTERISTICS

by

Jayantilal Majithia, B.Sc. Eng.Hons. (London)

M.Eng. (McMaster), Member I.E.E.

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

March 1971

DOCTOR OF PHILOSOPHY (1971) McMaster University
(ELECTRICAL ENGINEERING) Hamilton, Ontario

TITLE: Digital Moments Analyzer: Design and Error Characteristics

AUTHOR: Jayantilal Majithia, B.Sc. Eng. Hons. (London)
M.Eng. (McMaster)
Member I.E.E.

SUPERVISOR: Professor R. Kitai, D.Sc.

NUMBER OF PAGES: xvi, 190

SCOPE AND CONTENTS:

A special purpose computer (s.p.c.) is described which provides decimal readouts of the first four time-averaged moments and of the cumulative amplitude distribution of a randomly varying voltage. There is no theoretical low frequency limit, the upper frequency being about 5 kHz for a 99.73% confidence limit of a 1% error. Measurements can be made in a one cycle mode (for periodic inputs) or in a fixed-time or fixed-sample-size mode. Readouts of all moments are available immediately at the end of the measurement time so that the s.p.c. can be used for real-time applications. A simple method for the direct computation of standard deviation from the measured values of the first and second moments is also described. The errors arising in the s.p.c. are investigated theoretically and it is shown that for many commonly-encountered signals the overall error is within 1% for all moments. Iterative and near-iterative arrays using universal arithmetic cells are proposed; these would simplify the design of the s.p.c. considerably.

## ABSTRACT:

A portable special purpose computer (s.p.c.) is described which provides decimal readouts of the first four moments of a fluctuating voltage v on four separate registers. A fifth register provides a readout of the measuring time which can be within the range 10 ms. to 30 Hrs. The s.p.c. can be switched to another mode which provides a measure of the cumulative amplitude distribution of v within sixteen positive and negative levels. Salient characteristics of the s.p.c. are as follows:

(a) There are no low frequency limitations. The upper frequency limit, established by error considerations, is about 5 kHz with 99.73% confidence that the error is within 1%.

(b) At the end of the measuring time T, all the four moments are immediately available in magnitude and sign.

(c) The outputs can be available in any code, the only change necessary being in the code of the counting readout registers.

(d) All computations for a sample are completed before the next sample arrives so that programming and unnecessary storage facilities are eliminated.

The voltage input v is rectified and sampled systematically by an equi-interval a.d. converter. The samples, together with the sign bit, are fed into special purpose digital multipliers based on a "weighted feed" principle. The outputs from these multipliers, with the sign bit, are fed to accumulators via parallel adders for each of the moments. The overflows of these accumulators are shown to be contributions to the various moments and are fed to the decimal display registers.

Direct computation of the standard deviation ($\sigma$) of the input, from measured first and second moments has also been investigated.

A theoretical analysis of the various errors which occur in such an s.p.c. has been made. Results indicate that for most signals the overall error is within 1% for all four moments.

Finally, the development of a universal arithmetic cell, for use in iterative and near-iterative arrays, is reported in this thesis. It is shown that use of such arrays in the arithmetic units of the s.p.c. can lead to a considerably simplified design.

ACKNOWLEDGEMENTS

The author particularly wishes to thank Dr. R. Kitai for his helpful advice, criticism and encouragement received during the course of this work. The author is also grateful to Dr. T.J. Kennett, Dr. A.R. Elliott and Dr. N.K. Sinha, who, as members of the Supervisory Committee, have helped considerably by way of valuable and stimulating discussions. Thanks are due to Dr. S.O. Rice of the Bell Telephone Laboratories, New Jersey, U.S.A., for his assistance in the theoretical error analysis of the moments' algorithms by correspondence.

Special thanks are also due to Karl-Hans Siemens and Desmond Taylor for their comments and suggestions during the implementation of the special purpose computer.

Finally, the author is indebted to the National Research Council of Canada and to McMaster University for graduate scholarships received during the course of this work.

# TABLE OF CONTENTS

## FIGURE CAPTIONS

| FIGURE | | PAGE |
|---|---|---|

## LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| $m_k$ | : The kth order statistical moment of a random variable |
| $v$ | : A random signal assumed ergodic |
| $E\{g(v)\}$ | : The expectation of the random function $g(v)$ |
| $T$ | : Integration time in time-averages |
| $\tau_d$ | : Delay time used in the definition of the autocorrelation function |
| s.p.c. | : Special purpose computer |
| a.d. converter | : Analog to digital converter |
| W.F.L. | : Weighted feed logic |
| $\sigma$ | : Standard deviation of a statistical distribution, defined as $\sigma = \sqrt{m_2-m_1^2}$ |
| LSI | : Large scale integration |
| $\Delta V_\ell$ | : Voltage interval corresponding to the ith interval of an amplitude quantiser |
| $v_q$ | : Quantiser output |
| $n_q$ | : Error in the quantiser output |
| $N$ | : Number of samples in a statistical measurement |
| $\overline{v^k}$ | : $E\{v^k\}$ i.e., mean value of $v^k$ |
| $R_{mm}(\tau)$ | : Autocorrelation function of $v^k$ |
| $Var\{m_k\}$ | : Variance of the statistical distribution of $m_k$ |
| $K_k$ | : A constant of proportionality for the kth moment |
| $f(v)$ | : Probability density function of the random voltage $v$ |
| $p(v_r)$ | : Discrete valued probability density function |

| | |
|---|---|
| $m_{kd}$ | : Discrete-valued k$^{th}$ moment |
| $m_{k+}$ | : Positive contribution to $m_{kd}$ |
| $m_{k-}$ | : Negative contribution to $m_{kd}$ |
| n | : Number of quantisation intervals |
| $V_n$ | : Voltage at the n$^{th}$ level, assumed to be unity |
| $P_r$ | : Probability that the r$^{th}$ level is exceeded |
| $V_{mr}$ | : Mid-interval voltage corresponding to the r$^{th}$ interval |
| $P_{r+}$ | : Probability $P_r$ for positive level r |
| $P_{r-}$ | : Probability $P_r$ for negative level r |
| $C_r$ | : Number of samples for which the r$^{th}$ level is exceeded |
| $C_o$ | : Number of samples in a measurement and would generally equal N |
| $\mu_k$ | : The k$^{th}$ moment about the mean value |
| $\beta_1$ | : Skewness factor |
| $\beta_2$ | : Coefficient of kurtosis |
| $W_{k,r}$ | : Weighting number for the k$^{th}$ moment and r$^{th}$ level exceeded |
| $\ell$ | : Number of bits used in the s.p.c. and is given by $2^{\ell}$=n. |
| $Z_q$ | : q$^{th}$ output of a logic system |
| a,b,c,d,e,f | : Logic variable inputs to a system; also the 6 bits output of the a.d. converter |
| $\bar{a}$ | : Complement of a |
| MOPI | : Multiple Output Prime Implicants |
| $\psi$,X | : General product terms (minterms) |
| $\sigma_q$ | : Presence factor for the q$^{th}$ minterm |

| | |
|---|---|
| $\bigcup_{j=1}^{n}$ | : Logical sum of n minterms |
| $\Phi_j$ | : $j^{th}$ MOPI |
| $w_i$ | : Logic weighting factor for $i^{th}$ minterm |
| $\oplus$ | : Exclusive-OR operation |
| $[x]_1$ | : One's complement of the binary number x |
| $e_{k,z}$ | : Error in $k^{th}$ moment due to an error source z |
| $m^1_{k,z}$ | : Measured $k^{th}$ moment, using a system in which only the z error source is present |
| $e_{k,n}$ | : System quantisation error |
| $e_{k,\ell}$ | : Error due to level offsets |
| $e_{k,T}$ | : Error due to finite time (T) of measurement |
| $e_{k,p}$ | : Error due to finite sampling rate |
| $f_i$ | : Frequency of the input signal |
| p | : Sampling rate |
| $\delta V_r$ | : Level offset at the $r^{th}$ level |
| $t_r$ | : Time interval for which the signal is in the interval specified by the level offset |
| $V^1_r$ | : Slope of the signal at the $r^{th}$ level |
| $p(F_1)$ | : Probability that $F_1$ occurs |
| q | : Number of cycles of the input signal |
| $g(\theta)$ | : A periodic function of $\theta$ |
| B | : Bandwidth of input noise signal |
| $\tau_r$ | : Time for which the $r^{th}$ level is exceeded by the input signal |
| $\mu_a$ | : Aperture time of the a.d. converter |

| | |
|---|---|
| L,M,K | : General binary numbers |
| $T_m$ | : Total processing time for binary multiplication |
| $T_d$ | : Total processing time for binary division |
| BCD | : Binary-coded decimal |

# CHAPTER 1

# INTRODUCTION

In recent years, statistical methods have been successfully used to solve engineering problems especially in the fields of optimal control, communication and detection systems. One aspect of these applications is the need for measured statistics of a random signal. The more commonly required statistics are the probability density function and the various time averages (e.g., auto- and cross-correlation, moments). Numerous techniques for probability density function and correlation analysis have been reported in the engineering literature. However, for the analysis of statistical moments, the trend has been towards the use of a general purpose computer with a.d. conversion facilities. In this thesis, the development of a special purpose computer for the analysis of statistical moments, is reported. Programming requirements are entirely eliminated; also the computer may be used in real-time applications.

Assuming that the random signal is at least quasi-ergodic, the time averages and ensemble averages are equivalent. Thus the time average of $v^k$ where v is the random signal, is defined as

$$m_k = \lim_{T\to\infty} \frac{1}{T}\int_0^T v^k \, dt \tag{1.1}$$

This may also be defined as an expectation integral.

Thus $$m_k = E\{v^k\} = \int_{-\infty}^{+\infty} v^k f(v)\,dv \tag{1.2}$$

where f(v) is the probability density function of v. The quantity $m_k$ is called the $k^{th}$ order moment of v.

The importance of moments in statistical analysis is well established[1]. Thus a statistical distribution is completely specified, once all its moments are known. In some cases it is possible to represent the physical phenomenon under investigation by a statistical distribution of known properties and theoretically justify such a representation. However, in cases where such a theoretical justification can not be made, the moments may be used to obtain an empirical distribution. Such empirical distributions are useful in simulation studies where the random variables to be used are selected from the derived distributions. Their use has also been suggested recently by Kuo and Rowland, in a suboptimal adaptive filtering algorithm[2]. This algorithm requires only the first four moments of a specified random variable.

Analytic techniques[3] for approximating the statistical distribution have also been developed. These use an orthogonal expansion whose coefficients are the moments of the random variable. Karl Pearson[4], on the other hand has suggested the use of a family of curves, each member of which is completely specified by the first four moments of the random variable. Such approximate representations are useful in evaluating a system performance in terms of its component performances.

The advantages of digital techniques are now well recognized for performing arithmetic operations. These features, coupled with readily available and accurate analog to digital converters, make the use of digital methods very attractive for certain types of measurements. One particular area for such an application is that of statistical analysis.

Earlier efforts in instrumentation for this purpose have been mainly in the sequential determination of the probability density function and correlation measurements. In the measurement of probability density function[5,6,7,8,9], slicing circuits or comparators are used to simulate a voltage window (or interval) $\Delta V_\ell$ about a voltage reference $V_\ell$. The time spent by the random voltage in this interval is measured digitally and represents the probability that the voltage lies in the interval $\Delta V_\ell$ about $V_\ell$. Several analog[10,11] and digital methods have been proposed for autocorrelation and cross-correlation measurements. The correlator design proposed by Cheney[12] is significant since it uses the residue number system for the arithmetic processing. This number system[13] offers the advantage that multiplication can be as fast as addition, but has the disadvantage of special storage requirements. Furthermore division and overflow are not clearly defined in the residue number system. A substantial improvement in correlator design is that proposed by Kitai and Masuko[14]. This design uses a unique arithmetic processing system and requires very little storage for the actual processing. Furthermore, the reading of autocorrelation function for a given time delay $\tau_d$ is available immediately after the sampling of the input is terminated. Clearly for $\tau_d = 0$, this system yields the value of the second moment.

Instrumentation for the statistical moments by ensemble averaging has received very little attention in the past, the emphasis always being on the use of a general purpose computer for such analysis. One of the earliest efforts was by Deist and Kitai[15], who proposed a digital r.m.s. voltmeter. An algorithm for the second moment was developed and an iterative procedure was proposed for calculating the r.m.s value of a voltage. A special purpose digital instrument using this algorithm was reported by Kitai and Braithwaite[16]. An extensive theoretical and experimental error analysis of this instrument was carried out by the present author and has been reported elsewhere[17]. This analysis indicated the necessity for use of higher number of quantisation levels in order to reduce the errors. Feasibility of the basic processing used in the 16-level instrument[16], for use in the measurement of the higher order moments was also established. Hanrahan[18] has recently considered the use of various types of quantisers in real time averaging. Several standard numerical integration formulae are analysed and the error characteristics are given for first two moments, for autocorrelation and Fourier analysis. Some design aspects of individual modules for use in the Tangent Formula for numerical integration are also described.

This thesis deals with an extension of the above mentioned work[17] and describes the design and error characteristics of a special purpose digital computer (s.p.c.) for the measurement of the first four moments of a random signal. The salient features of this computer are as follows.

1. There are no low frequency limitations. The upper frequency limit is established by certain error considerations and is about 5kHz.

2. At the end of the measurement time T, all four moments are immediately available in a sign and magnitude form. The s.p.c. is therefore particularly suitable for real-time applications.

3. The outputs are in a decimal code but can be displayed in any other code, the only change required being in the logic of the display units.

4. The above-level amplitude probability distribution is readily measured in an alternate mode of operation, incorporated in the s.p.c.

5. Real time measurement of the standard deviation $\sigma$ from the continuous computation of first and second moments can also be made. A technique for this purpose has been developed and is reported in this thesis.

6. The sample size or the measurement time can be varied so that the special purpose computer can be programmed for long measurement runs.

7. The s.p.c. can be readily modified to accept either a continuous signal and use its own a.d. converter or to use ready-quantised data as input.

8. Since all computations for a sample are complete before the next sample arrives, all programming and unnecessary storage facilities are eliminated.

The properties of amplitude and time quantisation of a continuous signal are reviewed in Chapter 2. The algorithms, for the first four moments, to be used in the s.p.c. are developed and general design requirements are considered. Calculation of the $k^{th}$ moment generally requires at least k multiplications per sample of the input signal. Such a method is not suitable for real time computations since it requires excessive computation time. The weighted feed concept developed in this chapter is particularly powerful for such purposes since it requires only one multiplication and addition cycle for each sample to accumulate the $k^{th}$ moment. Furthermore, use of precise rectification of the input signal simplifies the design of the arithmetic units considerably.

Chapter 3 deals with the analysis of the weighted feed concept used in the arithmetic unit of the s.p.c. Simultaneous binary multipliers using combinational logic have been designed, for the purpose of calculating the weighting numbers, using general minimisation techniques. These techniques are suitable for multiple output - multiple input logic system minimisation. Computer programs for this minimisation are also discussed in this chapter.

In Chapter 4, a hardware implementation of the s.p.c. is considered. It is shown that use of one-step parallel binary adders in circulating accumulators considerably simplifies the timing requirements. Complete design details of the individual units of the s.p.c. are treated. Details of performance tests by using direct decimal readouts for d.c. inputs are also given. A novel technique for computation of standard deviation is described in this chapter.

Results of a theoretical error analysis are given in Chapter 5. Individual sources of errors are discussed. Extensive use has been made of a general purpose computer for calculation of these errors. The computer programs and the results obtained are also described. These results indicate that little improvement in errors is achieved beyond a quantisation of 128 levels, and that the errors due to level inaccuracies in the a.d. converter tend to swamp the other errors in such cases.

In the design of special purpose computers, the most critical unit is the arithmetic processor. This is also true of the s.p.c. designed for the analysis of moments. The arithmetic unit described in Chapter 3 is a hard-wired special purpose simultaneous multiplier, which cannot be easily extended for, say, a higher number of quantisation levels. Hence arithmetic processes using iterative arrays of logic cells were investigated and the results are described in Chapter 6. Fully iterative and nearly-iterative arrays have been developed for binary multiplication, division and square root extraction. These arrays use a universal logic cell based on the principles of ordinary binary arithmetic. Use of such arrays, implemented as LSI functions, would simplify the design of the s.p.c. Methods of interconnecting such arrays for the realisation of the weighting numbers for the moments are also described in this chapter.

In Chapter 7, the significant aspects of the s.p.c. are reviewed. Possible areas for further investigation are suggested. It is felt that design of special purpose computers for other measurements is feasible and that the weighted feed concept is an extremely powerful technique for use in such designs. Investigation of cellular arrays for complex arithmetic operations would considerably simplify the design of special purpose computers.

# CHAPTER II

# REAL TIME ALGORITHMS FOR STATISTICAL MOMENTS

## 2.1 Introduction

Digital computation from analog data requires that the signal be sampled at discrete time and quantised in amplitude. The number of bits used in the quantisation process is restricted in order to reduce the complexity of the arithmetic processing units. In special purpose computers unnecessary storage should also be eliminated. Furthermore since real time applications are an important consideration, it is necessary that the results be available immediately after sampling is terminated.

In this chapter, the theory of amplitude quantisation is reviewed with specific reference to special purpose computers. Based on this discussion, the discrete models for the first four moments are developed and the design of a system for these algorithms is outlined. A powerful arithmetic processing technique i.e., the use of weighted feeds, which is particularly useful in time averaging is also introduced.

## 2.2 Amplitude Quantisation

For digital computation the analog signal has to be sampled and the sample converted to a digital value of a finite word length. Thus the range of a random voltage v is subdivided into class intervals, specified for the $i^{th}$ interval by

$$\left[\sum_{q=1}^{i} \Delta V_q\right] < v < \left[\sum_{q=1}^{i} \Delta V_q\right] + \Delta V_{i+1}$$

$$(i=0,\pm1,\pm2,\ldots,\pm n-1) \qquad (2.1)$$

where $\Delta V_q$ is the quantisation interval corresponding to the $q^{th}$ interval. The quantity $[\sum_{q=1}^{i} \Delta V_q]$ is the threshold voltage at the $i^{th}$ level. The manner in which these threshold voltages are disposed in amplitude, determines the quantiser characteristic. A typical example of such a characteristic is shown in Fig. (2.1).

Amplitude quantisation is a non-linear operation, in which the quantised output $v_q$ can be regarded as the sum of the input v and a round off error $(n_q)$ such that

$$n_q = v_q - v \qquad (2.2)$$

The round off error $n_q$ is often referred to as the quantisation noise or the quantisation error. Any computation involving the quantised data, therefore, will be subject to errors.

In statistical analysis random data are often grouped into classes for further analysis. To compensate for errors in the computation of moments due to grouping, corrections known as Sheppard's corrections[19] are usually applied. Amplitude quantisation of continuous signals, for use in analysis has been considered by Widrow[20] and also by Watts[21]. Both have shown that if the quantisation is sufficiently fine, Sheppard's corrections may also be applied in these cases, provided the sampling rate exceeds the Nyquist Rate. An equi-interval quantiser is usually assumed in such analyses.

The quantiser characteristic i.e., the threshold voltages to be used, must be such that the error due to quantisation is minimised for as

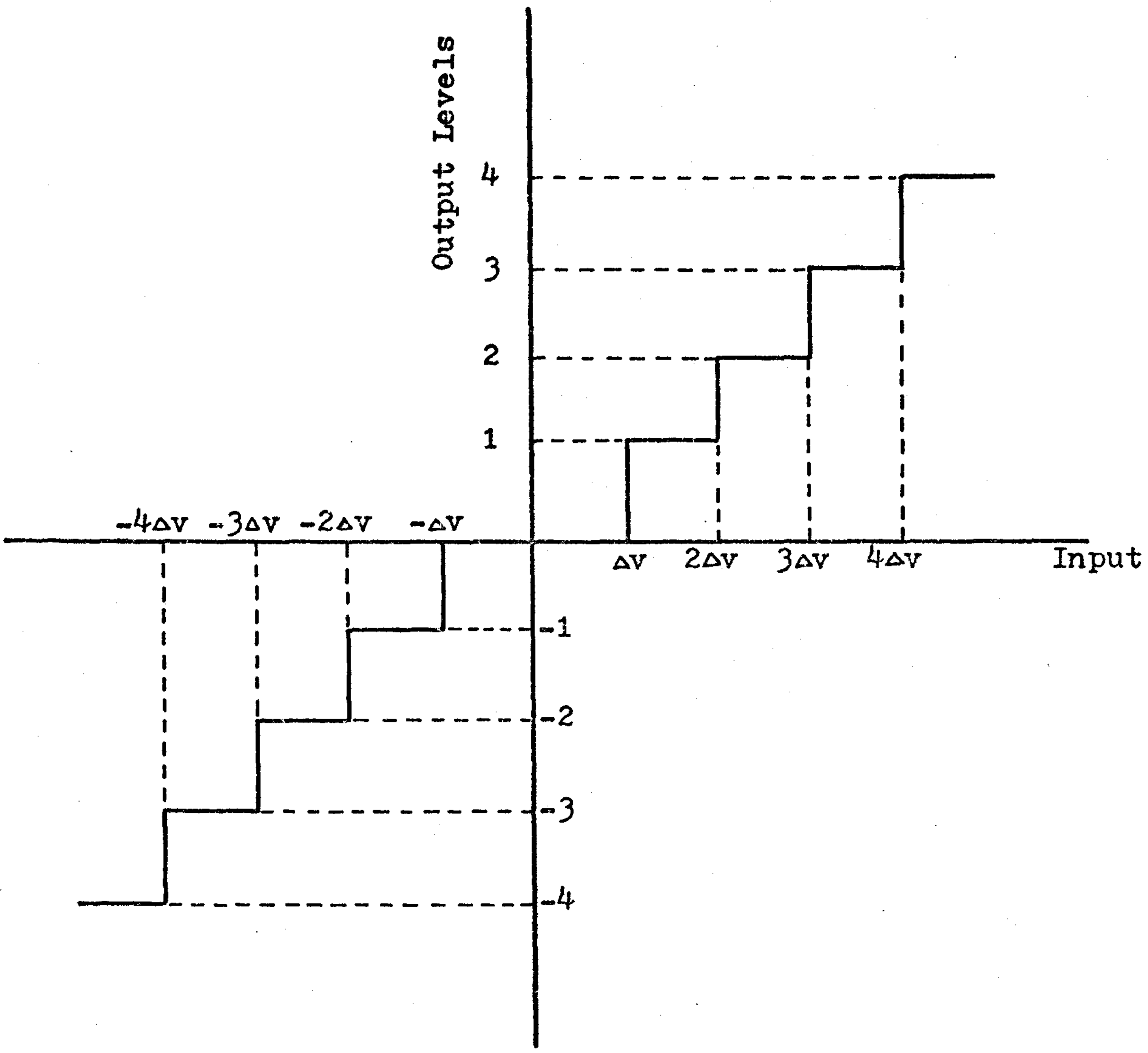


Figure 2.1: An equi-interval quantiser characteristic

wide a range of signals as possible. Although it is possible to construct the quantiser characteristic such that the error in the measured statistics is zero[22,23], it is found that this requires a prior knowledge of the probability distribution of the input signal. In statistical measurements, this is seldom the case and therefore it is desirable that within a range of values prescribed, the quantiser thresholds be spaced at equal intervals. In this manner all values are equally emphasised. An equi-interval quantiser is simple to realise and commercial a.d. converters[24] with excellent stability and accuracy are now available. The special purpose computer, designed for the analysis of moments, uses such an a.d. converter.

## 2.3 Finite Sampling Time

Sampling of the voltage is usually systematic and at a finite rate. The sample size or the total time of computation is also finite. The statistics, (in this case the first four time-averaged moments) which are computed using a finite sample size are not invariable i.e., they can not be reproduced by performing the measurement anew. The values obtained are governed by the probability distribution of the voltage under measurement.

Effect of finite sample size has been considered by R.A. Fisher[25] who has shown that the deviation in the measured statistics is proportional to $\frac{1}{\sqrt{N}}$. For systematic sampling N can be related to the time of measurement T. A second approach[26] uses the variance expression,

$$\mathrm{Var}\{m_k\} = \frac{2}{T}\int_0^T \left(1 - \frac{\tau}{T}\right)\{R_{mm}(\tau) - (\overline{v^k})^2\}d\tau \qquad (2.3)$$

The above equation indicates that the variance in the measured value of $m_k$ can only be zero if $T \to \infty$. In practice where T is finite the variance is also finite and therefore a confidence interval for the measured statistic is usually specified. Excessive measurement times, on the other hand, may endanger the validity of the stationarity assumption usually made in the derivation of the real time algorithms. It is therefore necessary to determine the minimum measurement time T or sample size N required to produce a specified confidence limit for a measured statistic.

The use of equation 2.3 for calculation of the confidence limits is possible only if the autocorrelation function $R_{mm}(\tau)$ of $v^k$ is known. If this information is not available then the results of sampling theory may be applied. It has been shown[25] that if the sample size is large, then the distribution of the first four moments follows a very nearly Normal Law. In these cases the variance is given by

$$\text{Var}(m_k) = \frac{K_k}{N} \tag{2.4}$$

where $K_k$ is a constant depending on the value k i.e., the order of the measured moment. Again the sample size to be used in a measurement should be large in order to assure only small variations in the measured values.

## 2.4 Real Time Algorithms for Statistical Moments

If the random signal to be analysed is at least quasi-ergodic then the time averages and ensemble averages are equivalent[27]. For such a signal, the k$^{th}$ statistical moment can be defined either as an expectation integral or as a time average. Thus

$$m_k = E\{v^k\} = \int_{-\infty}^{+\infty} v^k f(v)\,dv \tag{2.5}$$

where f(v) is the probability density function of the random voltage v.

Also

$$m_k = \lim_{T\to\infty} \frac{1}{T}\int_0^T v^k dt \tag{2.6}$$

Assuming quantisation into n equal intervals on either side of zero as shown in Fig. (2.2.a.), the discrete-value expectation integral form for equation (2.5) may be written as

$$m_{kd} = \sum_{r=-n}^{+n} v_r^k p(v_r) \tag{2.7}$$

where $p(v_r)$ is the discrete value probability density function.

The quantiser assumed in Fig. (2.2.a.) is of the equi-interval type with the quantisation interval being $\frac{V_n}{n}$ where $V_n$ is the voltage corresponding to the nth level. In the subsequent analysis it is assumed that $V_n = 1$.

Using mid-interval values for $v_r$ in equation (2.7), the kth moment may be expressed as

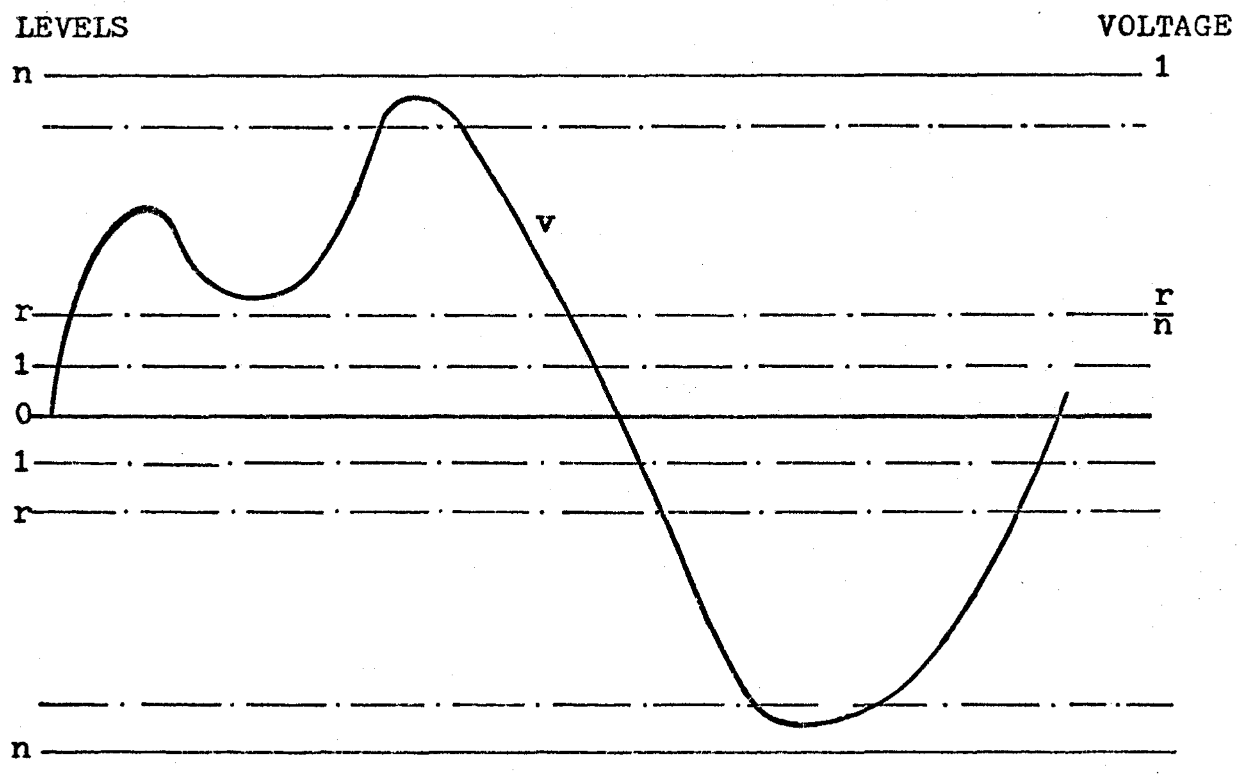
$$m_{kd} = \sum_{r=-n}^{+n} (P_r - P_{r+1}) V_{mr}^k \tag{2.8}$$

where $P_r$ is the probability that the voltage exceeds the rth level and $V_{mr}$ is the mid-interval value corresponding to the rth interval.

Now

$$V_{mr} = \frac{2r + 1}{2n}$$

$$\therefore \quad m_{kd} = \sum_{r=-n}^{+n} (P_r - P_{r+1}) \left\{ \frac{2r + 1}{2n} \right\}^k \tag{2.9}$$



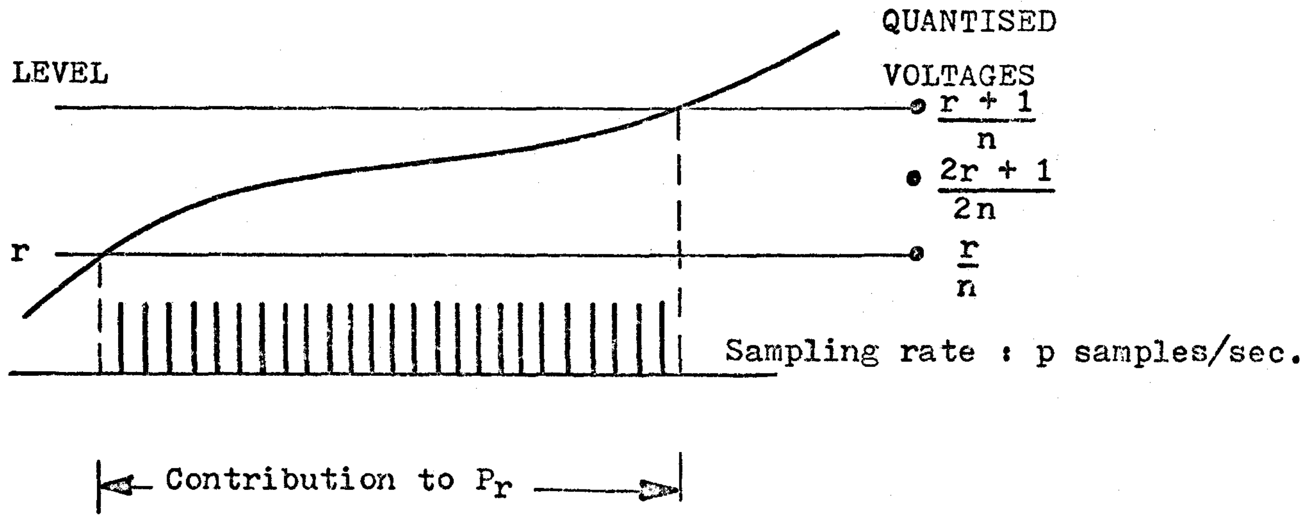


Figure 2.2: (a) Voltage under measurement with quantisation levels.

(b) Sampling at the rth level.

Separating the positive and negative parts of equation (2.9), we have

$$m_{kd} = m_{k+} + m_{k-} \tag{2.10}$$

$$= \sum_{\substack{r=0 \\ r+}}^{n} (P_r - P_{r+1})\{\frac{2r+1}{2n}\}^k + (-1)^k \sum_{\substack{r=0 \\ r-}}^{+n} (P_r - P_{r+1})\{\frac{2r+1}{2n}\}^k \quad \ldots \tag{2.11}$$

Since the expressions for the positive and negative parts are identical except for a possible sign difference, the algorithm is developed for $m_{k+}$ only. Thus

$$m_{k+} = (P_{o+} - P_1)(\frac{1}{2n})^k + (P_1 - P_2)(\frac{3}{2n})^k + \ldots$$

$$\ldots + (P_r - P_{r+1})(\frac{2r+1}{2n})^k + \ldots$$

$$\ldots + (P_{n-1} - P_n)(\frac{2n-1}{2n})^k \tag{2.12}$$

If the nth level is never exceeded than $P_n = 0$ and equation (2.12) may be written as

$$m_{k+} = \frac{P_{o+}}{(2n)^k} + \frac{1}{(2n)^k} \sum_{\substack{r=1 \\ r+}}^{n-1} P_{r+}\{(2r+1)^k - (2r-1)^k\} \tag{2.13}$$

The first term $\frac{P_{o+}}{(2n)^k}$ in equation (2.13) represents the contribution to $m_{k+}$ due to the voltage in the first quantisation interval. Its contribution to $m_{k+}$ is usually small except for k=1 and small n. In general, however, if n is large, this term may be neglected even for k=1. It is found that this assumption simplifies the design of the special purpose computer considerably. In a systematic sampling $P_r \simeq \frac{C_r}{C_o}$ where $C_r$ denotes the number of samples occurring while the input signal exceeds the voltage for the r^th level and $C_o$ is the total number of samples. Substituting for $P_r$ in equation (2.13), one obtains

$$m_{k+} = \frac{1}{(2n)^k} \sum_{\substack{r=1 \\ r+}}^{n-1} C_{r+}\{(2r + 1)^k - (2r - 1)^k\}/C_o \qquad (2.14)$$

The algorithms for various values of k may now be derived using the general expression (2.14).

2.4.1. <u>k = 1: First Moment.</u>

From equation (2.14)

$$m_{1+} = \frac{1}{2nC_o} \sum_{r=1}^{n-1} C_r \cdot 2$$

$$= \frac{1}{nC_o} \sum_{\substack{r=1 \\ r+}}^{n-1} C_r \qquad (2.15a)$$

Similarly,

$$m_{1-} = \frac{-1}{nC_o} \sum_{\substack{r=1 \\ r-}}^{n-1} C_r \qquad (2.15b)$$

and $$m_1 = |m_{1+}| - |m_{1-}| \qquad (2.16)$$

2.4.2. <u>k = 2: Second Moments.</u>

The second moment of a random variable v is used together with the mean value $m_1$ to describe the spread of the statistical distribution of v. The standard deviation $\sigma = \sqrt{m_2 - m_1^2}$ is the measure of this spread. For periodic signals with zero mean, $\sigma$ is the r.m.s. value.

From equation (2.14), for k = 2

$$m_{2+} = \frac{1}{C_o \cdot 4n^2} \left[ \sum_{\substack{r=1 \\ r+}}^{n-1} C_r \{(2r+1)^2 - (2r-1)^2\} \right]$$

$$\therefore \quad m_{2+} = \frac{2}{C_o n^2} \sum_{r=1}^{n-1} rC_r \tag{2.17a}$$

Similarly

$$m_{2-} = \frac{2}{C_o n^2} \sum_{\substack{r=1 \\ r-}}^{n-1} rC_r \tag{2.17b}$$

and

$$m_2 = |m_{2+}| + |m_{2-}| \tag{2.18}$$

2.4.3. <u>k = 3: Third Moments.</u>

The third moment is used to describe the asymmetry or skewness of the distribution for v. A useful formula is the third moment defined about the mean value, $m_1$. Thus,

$$\mu_3 = E\{(v - m_1)^3\}$$

$$= m_3 - 3m_2 m_1 + 2m_1^3 \tag{2.19}$$

and a skewness factor $\beta_1$ is defined as

$$\beta_1 = \frac{\mu_3}{\sigma^3} \tag{2.20}$$

A single-peaked distribution with $\beta_1 < 0$ is said to be skewed to the left or has a left 'tail' and with $\beta_1 > 0$, it is skewed to the right. If $\beta_1 = 0$, the distribution is symmetrical. The algorithm for $m_3$ is now derived. From equation (2.14), for k = 3,

$$m_{3+} = \frac{1}{(2n)^3 C_o} \Big[ \sum_{\substack{r=1 \\ r+}}^{n-1} C_r\{(2r+1)^3 - (2r-1)^3\}\Big]$$

$$\therefore \quad m_{3+} = \frac{1}{C_o n^3} \Big[ \sum_{\substack{r=1 \\ r+}}^{n-1} C_r\Big( \frac{24r^2 + 2}{8} \Big)\Big] \qquad (2.21)$$

If the input signal is of low level such that only a few of the available n levels are utilised then the errors in the time averages become excessive. It is therefore expected that a scaler is included at the input so that all n levels are used. In this case $3r^2 \gg ¼$ and equation (2.21) may be further simplified to

$$m_{3+} \simeq \frac{1}{C_o n^3} \sum_{\substack{r=1 \\ r+}}^{n-1} 3r^2 C_r \qquad (2.22a)$$

Similarly

$$m_{3-} \simeq \frac{1}{C_o n^3} \sum_{\substack{r=1 \\ r-}}^{n-1} 3r^2 C_r \qquad (2.22b)$$

and

$$m_3 = |m_{3+}| - |m_{3-}| \qquad (2.23)$$

### 2.4.4. <u>k = 4: Fourth Moment.</u>

The fourth moment ($m_4$) is used to describe the peakedness of a distribution. Thus the fourth moment about the mean ($\mu_4$) is used to define the coefficient of kurtosis ($\beta_2$), where

$$\mu_4 = E\{(v - m_1)^4\}$$

$$= m_4 - 4m_3 m_1 + 6m_2 m_1^2 - 3m_1^4 \qquad (2.24)$$

and

$$\beta_2 = \frac{\mu_4}{\sigma^4} \qquad (2.25)$$

Using equation (2.14), one obtains

$$m_{4+} = \frac{1}{C_o(2n)^4}\left[\sum_{\substack{r=1\\r+}}^{n-1} C_r\{(2r+1)^4 - (2r-1)^4\}\right] \tag{2.26}$$

$$= \frac{1}{C_o n^4}\left[\sum_{\substack{r=1\\r+}}^{n-1} C_r(4r^3 + r)\right] \tag{2.27}$$

Assuming that scaling is used, then $4r^3 >> r$ and

$$m_{4+} \doteq \frac{1}{C_o n^4}\sum_{\substack{r=1\\r+}}^{n-1} 4r^3 C_r \tag{2.28a}$$

and

$$m_{4-} \doteq \frac{1}{C_o n^4}\sum_{\substack{r=1\\r-}}^{n-1} 4r^3 C_r \tag{2.28b}$$

$$\therefore \quad m_4 = |m_{4+}| + |m_{4-}| \tag{2.29}$$

A general expression for $m_{k+}$ can be deduced from equations (2.15a), (2.17a), (2.22a) and (2.28a). Thus

$$m_{k+} = \frac{1}{C_o n^k}\sum_{r=1}^{n-1} k\, r^{(k-1)}\, C_r \tag{2.30}$$

A similar expression holds for $m_{k-}$. Furthermore,

$$m_k = |m_{k+}| + (-1)^k\, |m_{k-}| \tag{2.31}$$

The expressions (2.30) and (2.31) are used for the moments analysis in the special purpose computer.

## 2.5 The Weighted Feed Concept

Consider the general moments equation (2.30). For each sample exceeding the $r^{th}$ level it is necessary to compute $kr^{k-1}$ and add this quantity to an accumulator. The evaluation of $kr^{k-1}$ requires k multiplications, except for $k = 1$ when no multiplication is necessary. Furthermore, such an implementation of equation (2.30) would, also require either level selectors (so that r can be varied) or n comparators with preset references corresponding to the n levels. The design of a computer based on such a method is extremely wasteful of hardware and the time of processing would be excessive. An alternative is to use weightings for each level which would allow use of a single comparator and which considers all the available levels simultaneously at a sampling instant. Thus consider sampling of the input signal, at the $r^{th}$ level shown in Fig. (2.2(b)). If a sample occurs while the input exceeds the $r^{th}$ level, then all the levels from 0 to (r-1) are also exceeded. Thus this sample contributes to $C_1, C_2, \ldots, C_r$. The total contribution to $m_{k+}$, due to a sample exceeding the $r^{th}$ level, therefore is given by

$$W_{k,r} = \sum_{q=1}^{r} k\, q^{k-1} \qquad (2.32)$$

$W_{k,r}$ is termed the weighting number for the $k^{th}$ moment and $r^{th}$ level. Thus it can be seen that computation of $m_{k+}$ or $m_{k-}$ is an accumulation process in which, given the information of r the highest level exceeded at the sampling instant, a corresponding weighting number $W_{k,r}$ is added to the accumulator.

The weighting numbers of equation (2.32) can be realised either by a combination logic system i.e., simultaneous multipliers or by general purpose polynomial evaluators. Both these methods are considered in this thesis (Chapters 3 and 6). If k is even then the same logic is used in conjunction with a single accumulator (see equations (2.18) and (2.29). If k is odd then the output of the weighted feed logic system is gated to either a positive accumulator or a negative accumulator, by the predetermined sign bit.

The weighting numbers for the first four moments may be deduced from equation (2.32). Thus,

$$\begin{aligned} W_{1,r} &= r \\ W_{2,r} &= r(r+1) \\ W_{3,r} &= \frac{r(r+1)(2r+1)}{2} \\ \text{and} \quad W_{4,r} &= r^2(r+1)^2 \end{aligned} \tag{2.33}$$

Division by $C_o$ in equation (2.30) is a trivial operation and can be implemented if required. The only operation remaining is, therefore, division by $n^k$. The size of the accumulator for the kth moment and $\ell$-bit quantisation (i.e., $2^\ell=n$), is $\ell k$ bits. For such an accumulator, a carry out or overflow occurs when the accumulated total exceeds or just equals $2^{\ell k}$ (i.e., $n^k$). Thus the overflow represents the contribution to $m_k C_o$ at a sampling instant. Inputs at a sampling instant, to the binary adder of the accumulator, are the weighting number and the remainder of the division by $n^k$ of the accumulated number. Complete design of such accumulators is considered in Chapter 4.

## 2.6 General Description of the S.P.C.

A schematic diagram of the special purpose computer for the measurement of moments is shown in Fig. (2.3). A scaler is included for precision amplification or attenuation so that the full range of the a.d. converter (i.e. all available levels) is utilised. The sign of the input voltage v is detected at the input and is required in the computation of odd-order moments. Since only the magnitude of r, the highest level exceeded, is required for the determination of the weighting numbers, a precision rectifier with low distortion is used. The sampling rate of the a.d. converter is controlled by a master clock system.

The level r information determined by the a.d. converter, is used as an address input to the weighted feed logic (W.F.L.) unit which determines the weighting number for each moment. For k odd, the output of the corresponding W.F.L. unit is gated by the sign bit to either the positive or the negative accumulator. For k even, the outputs of the W.F.L. unit feed directly to an accumulator. The overflows from these accumulators represent the contribution to $m_kC_o$ and are used to gate a delayed clock pulse (or the a.d. done pulse usually produced by an a.d. converter) to display registers. Clearly, since computation of $m_kC_o$ is a simple counting process, it can be performed in any code.

For odd-numbered moments, the sign information is used to control the count direction of up-down counters. For even numbered moments, the sign information is redundant and only unidirectional counters are required. The master clock also feeds to a counter for $C_o$ display.

Control circuits are required to change the sampling rate, provide start and stop facilities and also change the mode of operation by which

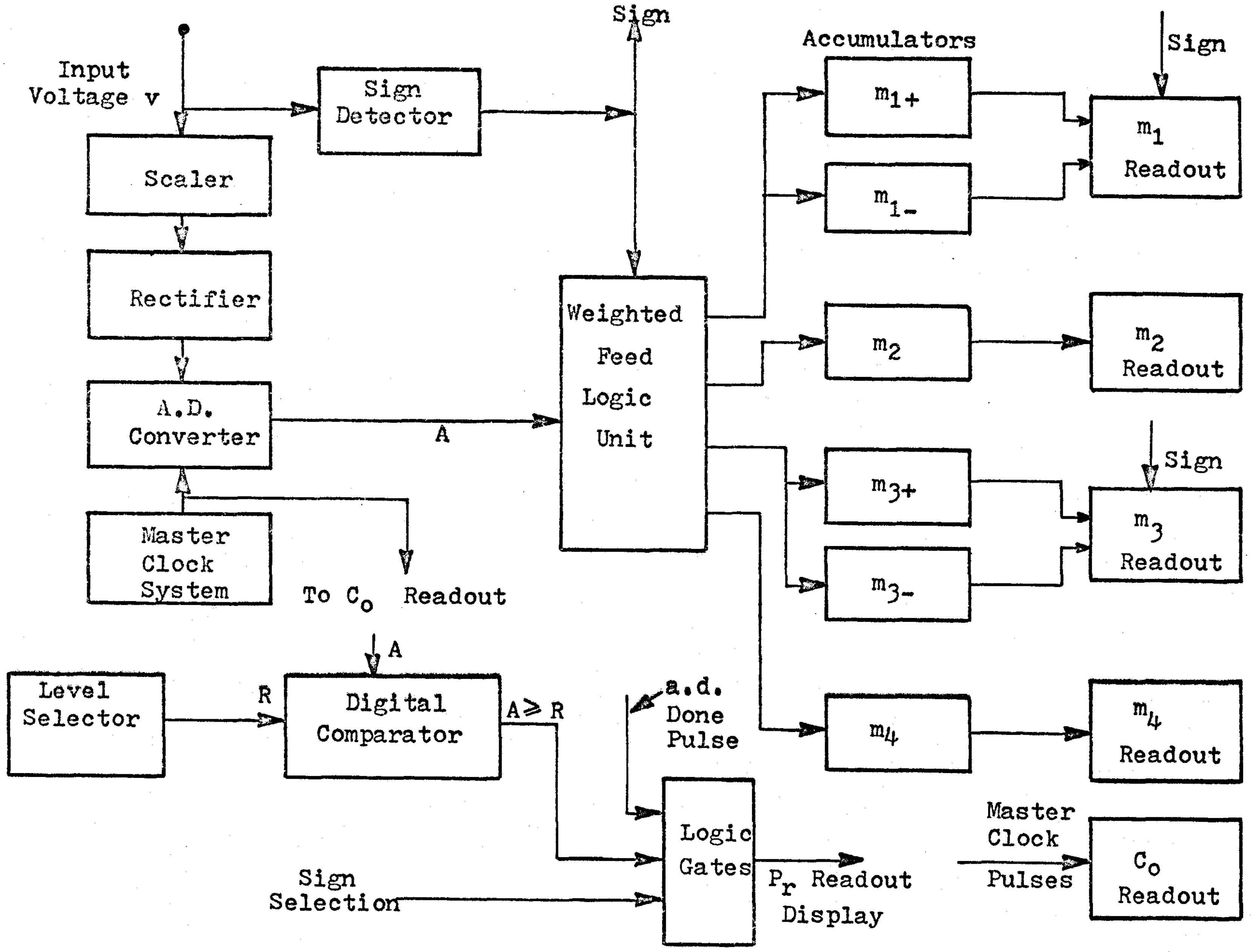


Figure 2.3: Block diagram of the S.P.C. for measurement of moments.

the s.p.c. can be used as an above level probability analyser.

## 2.7 Above Level Probability Measurements

Above level probabilities can be measured by comparing the output of the a.d. converter to a reference level R selected externally by means of a level selector switch. The reference level information is required in a binary code so that the difficulties of precision reference levels setting are eliminated. In Fig. 2.3, referring to the probability section of the circuit, the A inputs are the a.d. converter bits for a sample and the R inputs are the binary bits of the reference level. When $A \geq R$, the digital comparator switches to the logic state 1. This, together with the sign bit is used to gate the clock to the output counter. For a sample size $C_o$, the counter reading $C_r$ represents the time for which the input exceeds the rth level i.e., $\frac{C_r}{C_o}$ is the approximate above level amplitude probability. Use of the sign bit allows measurements to be made on either side of the zero level. If a second level selector is used to produce the reference (R + 1) for a second comparator, then the output of the latter can be used to inhibit the $C_r$ counter. In this way, the probability that the input signal is within a quantisation interval above the Rth level may also be determined.

## 2.8 Hardware Details

The individual units of the special purpose computer of Fig. (2.3) are implemented using currently available TTL integrated logic circuits, which have a logic 0 state at 0 v and a logic 1 state at +3v. The a.d. converter used is an 8-bit successive approximation type[28]. Precision rectification is performed using high slew-rate operational amplifiers. Design and implementation of these units are considered in the next two chapters.

# CHAPTER III

# DESIGN OF THE WEIGHTED FEED LOGIC

## 3.1 Introduction

The concept of weighted feed logic for use in the computation of the first four moments has been introduced in Chapter II. The weighting numbers to be used for each sample, can be realised either by general purpose binary multipliers or by binary polynomial evaluators. Both these methods are difficult to implement with available logic circuits. Furthermore conventional multipliers[29] tend to be slow for real time applications. In view of these limitations, the use of simultaneous multipliers, to be implemented with combinational logic modules, was investigated. The design and implementation of such multipliers are considered in this chapter.

The weighted feed logic concept is analysed in detail and recognised as a problem in multiple output combinational logic design. To minimise cost and improve the reliability of the system it is necessary to minimise the number of logic gates required. Minimisation using a computer-aided approach has been investigated and the various programs used for this purpose are described in this chapter. An overall design of the weighted feed logic for a 6-bit a.d. conversion in terms of circuit requirements is also considered.

## 3.2 Analysis of the Weighted Feed Concept

It has been shown in Chapter II, Section (2.5), that for the computation of the kth moment, each sample has to be weighted by the weighting number $W_{k,r}$ given by

$$W_{k,r} = \sum_{q=1}^{r} k\, q^{k-1} \tag{3.1}$$

where the highest level exceeded by the sample is r. The expressions for the weighting numbers for the four moments were also derived. These are,

$$\begin{aligned} W_{1,r} &= r \\ W_{2,r} &= r(r+1) \\ W_{3,r} &= \frac{r(r+1)(2r+1)}{2} \\ W_{4,r} &= r^2(r+1)^2 \end{aligned} \tag{3.2}$$

The range of r is finite and is determined by the a.d. converter used. Thus for a six bit a.d. converter, the highest number of bits required for the weighting numbers is for $W_{4,r}$ and equals 24, as can be seen from equation (3.2).

Cheney[12] has suggested use of the residue number system in cases where the range of numbers is restricted. However, this approach requires special coding and decoding circuits, which would complicate the design of the special purpose computer. The conventional shift and add multiplying technique[29], on the other hand, is simple to implement but requires considerable processing time, a factor which limits its use in systems where real time applications are important. The remaining possibility is the use of special simultaneous multipliers where each bit of the product is realised with an AND/OR logic configuration. Thus consider a simple design for $W_{2,r}$ where r is restricted to three bits (i.e., $r_{max} = 7$). The weighting numbers and their binary equivalents are shown in Table 3.1. Each output line of this table can be written as a Boolean expression. Thus, consider the output line $Z_4$. This line is in logic state 1 when r = 3, 5, 6 and 7. Thus the sum of products type Boolean expression for $Z_4$ may be derived as shown,

$$Z_4 = \sum(3, 5, 6, 7) \tag{3.3}$$

$$\therefore \quad Z_4 = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$$

This may be simplified to

$$Z_4 = ab + ac + bc \tag{3.4}$$

where a, b and c represent the three bits of the level information r.

The logic minimisation for a single output line can be carried out using well known rules of Boolean algebra[30] (as in equation (3.4)), the Karnaugh maps[31] or by iterative techniques[32,33]. However, these methods do not take into account the possibility of a logic product term (i.e., a minterm) being shared by two or more output lines. Thus, referring to Table 3.1, it is seen that the minterm ab is also required in the output line $Z_6$ as well as in $Z_4$ and therefore need not be reproduced for $Z_6$. Such shared minterms can be easily recognised for simple systems but for systems with a large number of outputs this is seldom the case and a systematic minimisation procedure is called for, which takes into account all of the outputs of the system.

| $r$ | Binary Equivalent of $r$ | $W_{2,r}$ | Binary Equivalent of $W_{2,r}$ | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | $Z_7$ | $Z_6$ | $Z_5$ | $Z_4$ | $Z_3$ | $Z_2$ | $Z_1$ |
| 0 | 0 0 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 0 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 1 0 | 6 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 3 | 0 1 1 | 12 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 0 0 | 20 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 5 | 1 0 1 | 30 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 6 | 1 1 0 | 42 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 7 | 1 1 1 | 56 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Table 3.1: The Input/Output table for design of $W_{2,r}$ with $r_{max} = 7$.

| Input | | | Output | | | | | |
|---|---|---|---|---|---|---|---|---|
| a | b | c | $Z_6$ | $Z_5$ | $Z_4$ | $Z_3$ | $Z_2$ | $Z_1$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Table 3.2: Input/Output representation for $W_{2,r}$.

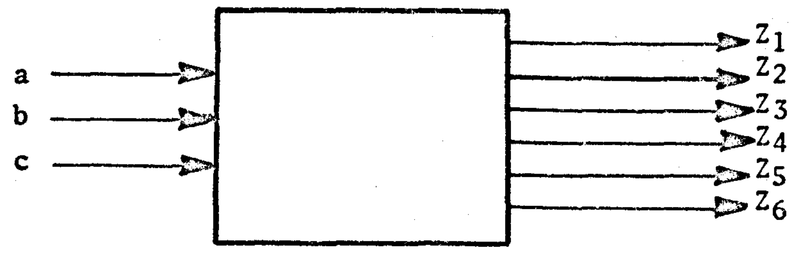


Fig. 3.1: Multiple Input-Multiple Output logic unit for $W_{2,r}$.

Table 3.1 may be reorganised as shown in Table 3.2 and the logic system is represented as shown in Fig. (3.1). The reorganised table is a standard multiple output-multiple input logic truth table. Truth tables similar to Table 3.2 are produced for $r_{max} = 63$ and for each weighting number expression of equation (3.2), except for $k = 1$ where no multiplication is required. An a.d. converter with 6-bit conversion has been assumed. The truth tables are produced using a tabulating program whose flow chart is shown in Fig. (3.2). Thus each output line of the weighted feed logic is available in the sum of products form of the type of equation (3.3).

The logic minimisation technique considered in the next section has been developed for a general multiple output-multiple input problem.

## 3.3 Multiple Output-Multiple Input Logic Minimisation

Minimised logic design at present and in particular reference to special purpose computer design, is required for economy and reliability. The development of LSI functions and microminiaturised circuits will, however, reduce the importance of the cost factor and therefore the need for absolute minimisation. However it will still be necessary to eliminate redundancies, in order to produce a reliable logic system. The minimisation technique developed in this section does not produce an absolute minimum answer but one in which a near minimum solution is obtained without requiring excessive computer time. A three-output system is used as an example.

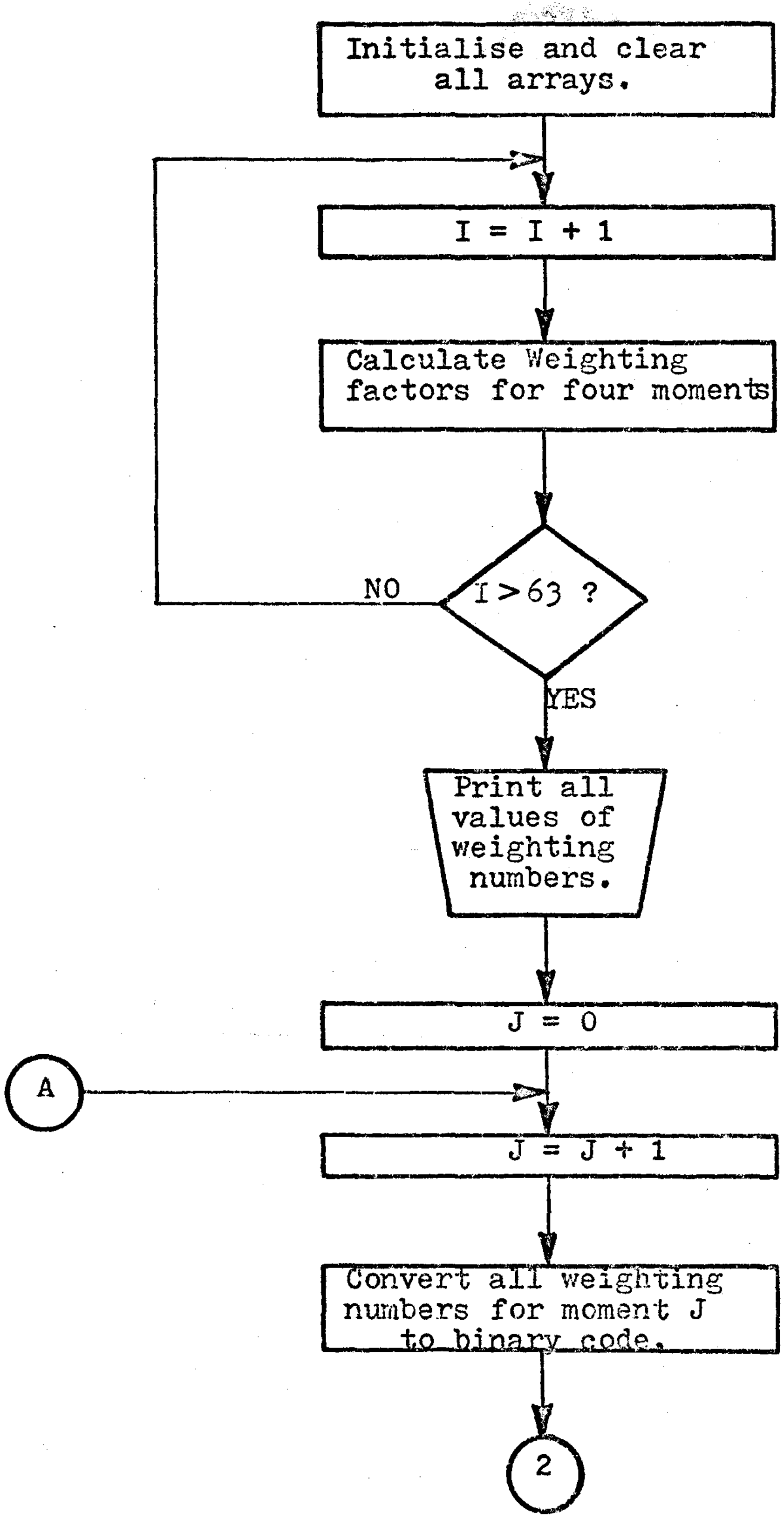

Initialise and clear all arrays.
I = I + 1
Calculate Weighting factors for four moments
I > 63 ?
NO
YES
Print all values of weighting numbers.
J = 0
A
J = J + 1
Convert all weighting numbers for moment J to binary code.
2

Figure 3.2: Flowchart of procedure for determining weighting numbers and gating functions.

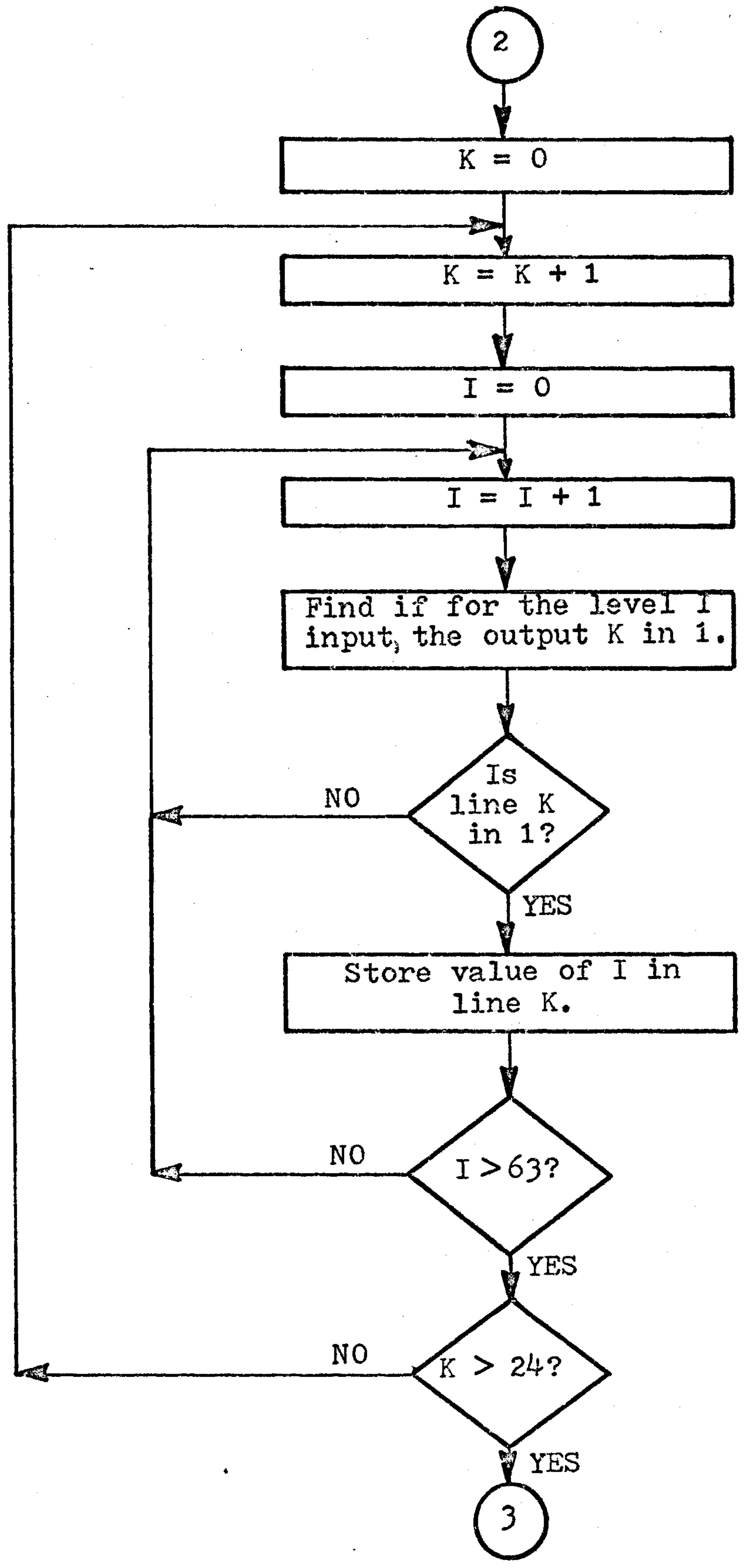

2
K = 0
K = K + 1
I = 0
I = I + 1
Find if for the level I input, the output K in 1.
Is line K in 1?
NO
YES
Store value of I in line K.
I > 63?
NO
YES
K > 24?
NO
YES
3

Figure 3.2: Contd.

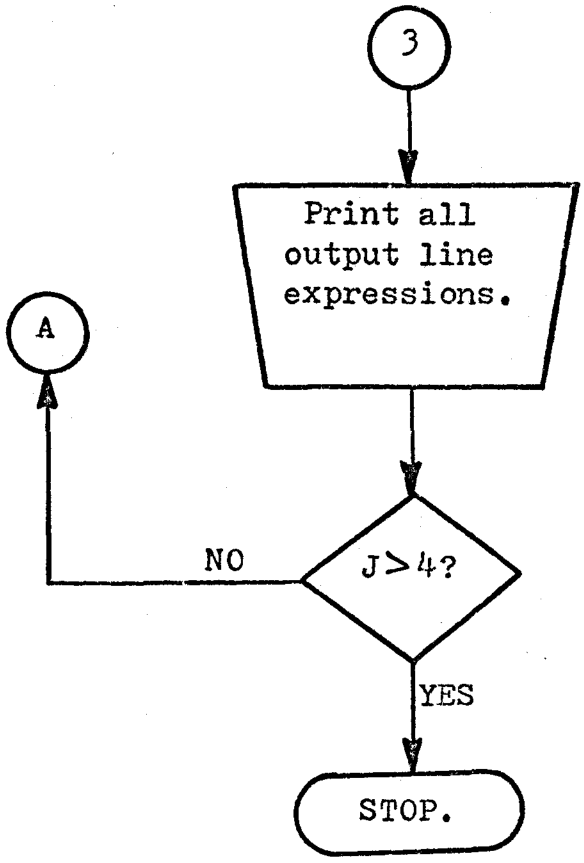


Figure 3.2: Contd.

### 3.3.1. Multiple Output Prime Implicants (MOPI)[34]

Each output $Z_1, \ldots, Z_n$ of the combinational logic system is expressed as a logical sum of several products (minterms), the number of input variables being specified. A sum of products type representation usually implies a realisation with AND/OR logic. The first step in a minimisation process is to generate prime implicants from the given set of output functions. This is usually a simple process for a single output system with up to four or five input variables. However, the standard tabulation method[32] becomes extremely tedious when the logic system has a large number of outputs with more than five logic variables and where the multiple output prime implicants are to be generated.

The multiple output prime implicant (MOPI) has been defined by Bartee[35]. In this definition use is made of E terms which are obtained from the output functions. An E term consists of a V-section which is comprised of the logic variables and their complements and a book-keeping or KP-section which denotes the output lines. Thus a minterm $\bar{a}bcd\bar{f}$ used in the output lines $Z_1$, $Z_3$ and $Z_5$ has an E term given by $\bar{a}bcd\bar{f}Z_1\bar{Z}_2Z_3\bar{Z}_4Z_5$ or by 011120,20202 in the notation adopted in the program.

Two basic operations which are used in the MOPI program are now defined.

(i) Consensus: If two minterms X and Y contain only a variable which is complemented in one and not in the other, then the consensus is the minterm formed by the product of X and Y omitting the opposed variable e.g.,

$$X = abc - -\bar{Z}_1\ Z_2\ Z_3, \quad Y = - - \bar{c}de\ \bar{Z}_1\ Z_2\ \bar{Z}_3$$

then the consensus exists and is given by

$$C = ab - - de\ \bar{Z}_1\ Z_2\ \bar{Z}_3$$

(ii) Subsumption: A minterm X subsumes another minterm Y if all the literals in Y are also in X. Thus X = abc subsumes Y = ab. In a multiple output system, the KP-section of X must be completely included in the KP-section of Y if X is to subsume Y.

The MOPI is an E term formed from the original output functions or by repeated consensus operation. The MOPI is such that its V-section subsumes no shorter V-section of another E term having the same KP-section or a KP-section with fewer outputs. It has been shown by Bartee[35] that the Quine-Mcluskey iterative consensus method may be used to generate the MOPI. This method is adopted in the program with a flow chart shown in Fig. 3.3.

The MOPI program requires as input data, the output functions $Z_1$, $Z_2$, etc. expressed as a row of minterms, in their decimal equivalent. It is to be noted that the tabulating program of Fig. 3.2 produces the output lines for the W.F.L. unit in this form and therefore these may be used directly for MOPI generation.

A typical example to illustrate the MOPI generation is given.

3.3.2. Example

$Z_1$ = 1, 6, 7, 9, 11, 14, 15

$Z_2$ = 4, 5, 6, 7, 10, 12, 13, 14, 15

$Z_3$ = 1, 4, 5, 9, 10, 11, 12, 13, 14, 15

The MOPI obtained by the program are given in Table 3.3 on page 41.

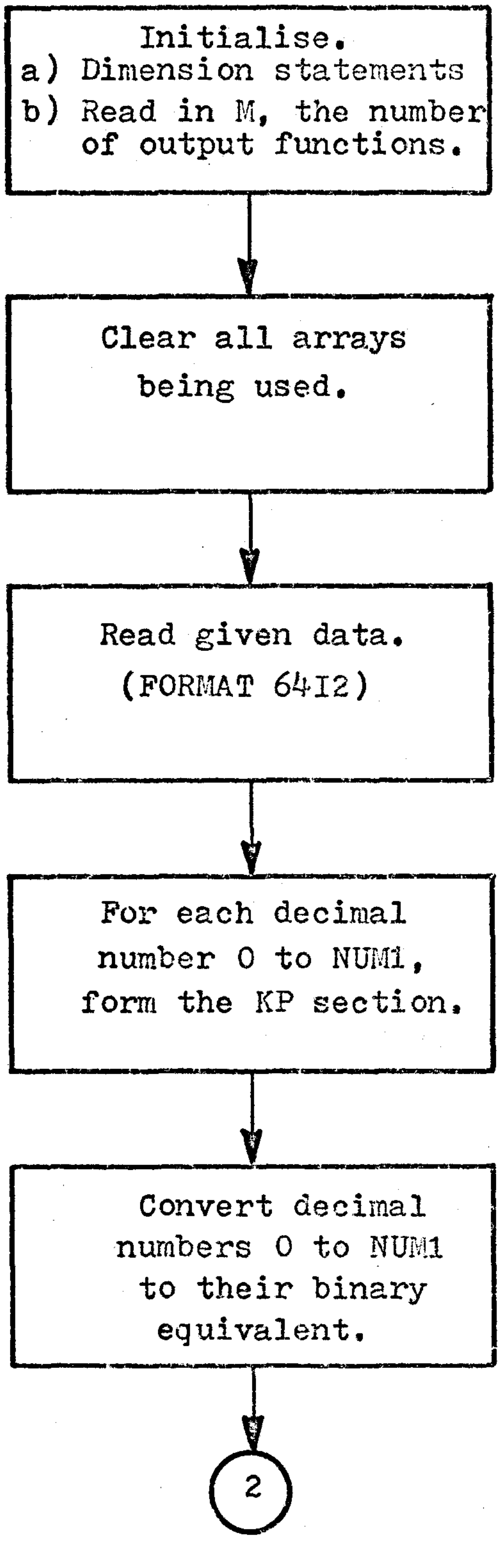

Initialise.
a) Dimension statements
b) Read in M, the number of output functions.
Clear all arrays being used.
Read given data. (FORMAT 64I2)
For each decimal number 0 to NUM1, form the KP section.
Convert decimal numbers 0 to NUM1 to their binary equivalent.
2

Figure 3.3: Flow-chart procedure for MOPI generation.

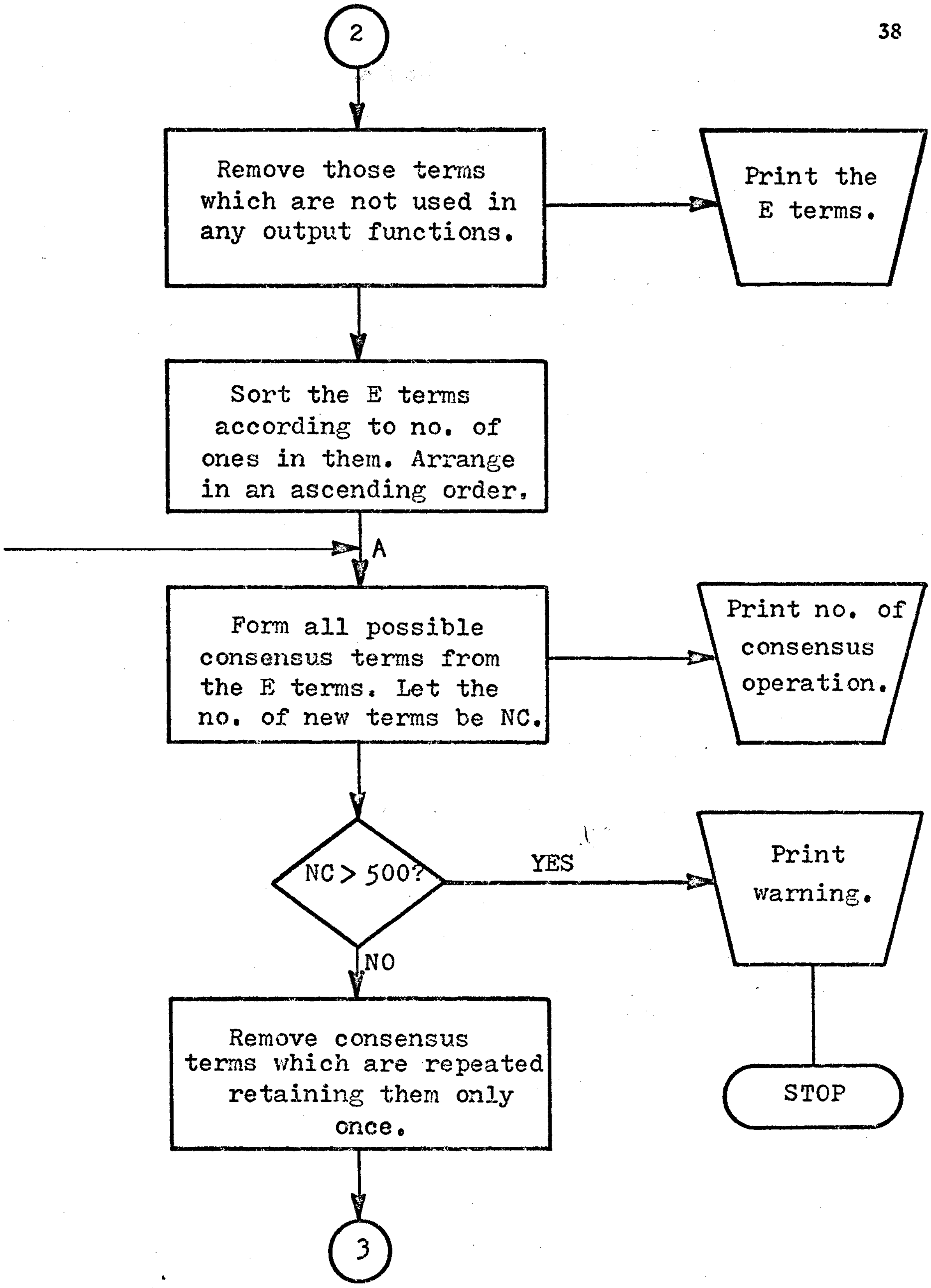

2
Remove those terms which are not used in any output functions.
Print the E terms.
Sort the E terms according to no. of ones in them. Arrange in an ascending order.
A
Form all possible consensus terms from the E terms. Let the no. of new terms be NC.
Print no. of consensus operation.
NC > 500?
YES
Print warning.
STOP
NO
Remove consensus terms which are repeated retaining them only once.
3

Figure 3.3: Contd.

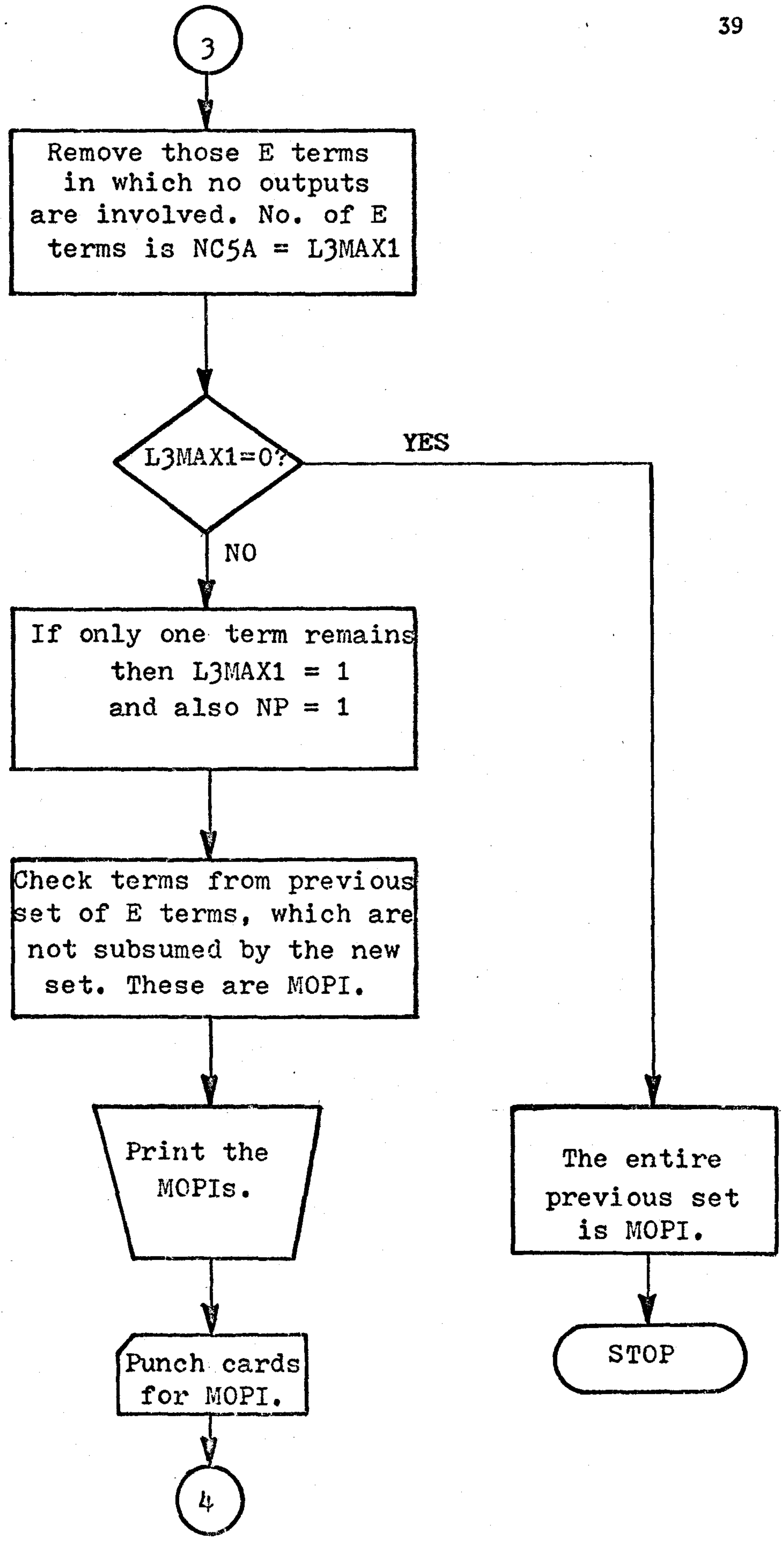

3
Remove those E terms in which no outputs are involved. No. of E terms is NC5A = L3MAX1
L3MAX1=0?
YES
NO
If only one term remains then L3MAX1 = 1 and also NP = 1
Check terms from previous set of E terms, which are not subsumed by the new set. These are MOPI.
Print the MOPIs.
Punch cards for MOPI.
4
The entire previous set is MOPI.
STOP

Figure 3.3: Contd.

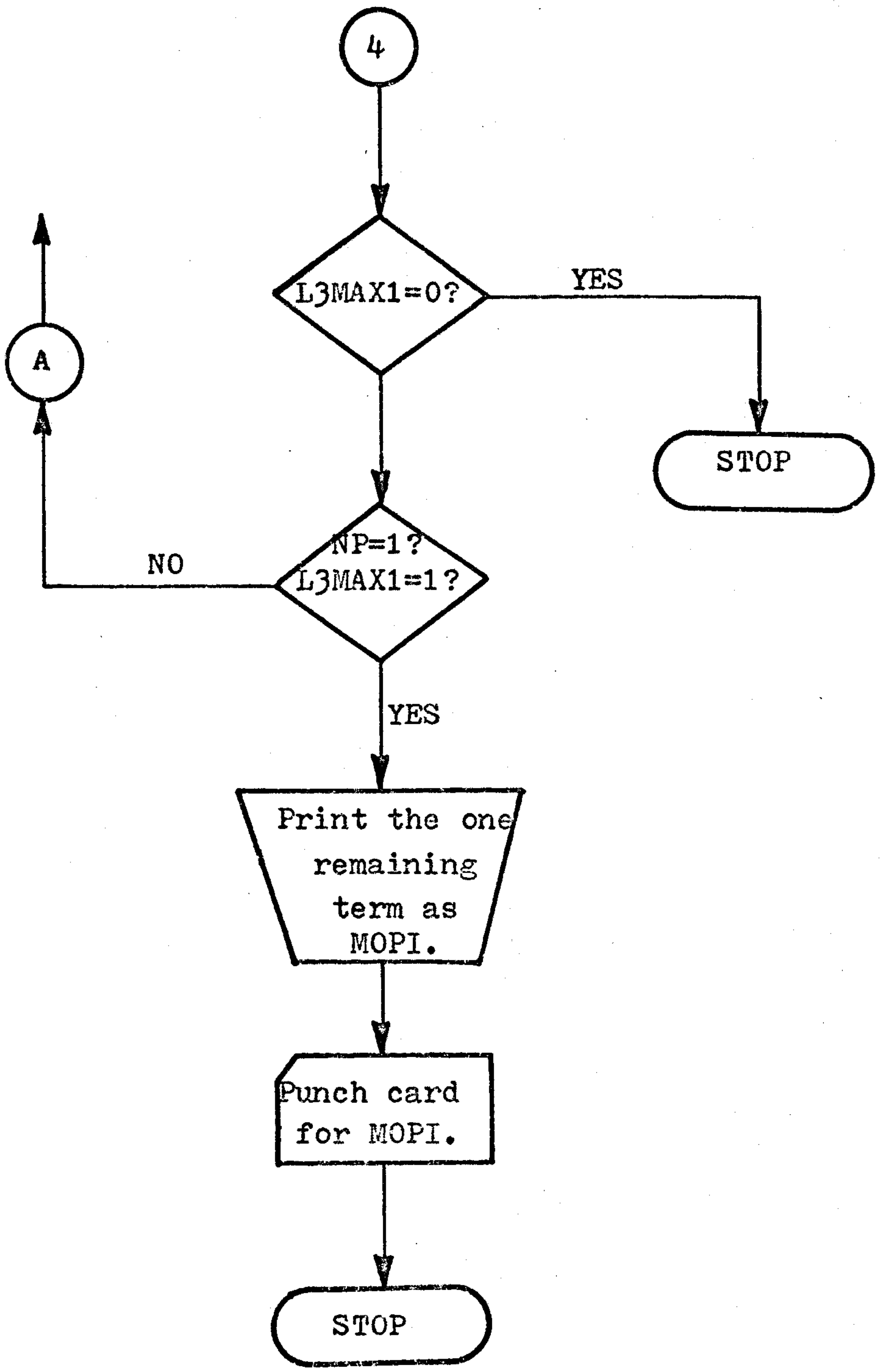


Figure 3.3: Contd.

| V-section | KP-section |
|---|---|
| 2122 | 020 |
| 2122 | 220 |
| 2102 | 022 |
| 2201 | 002 |
| 1122 | 022 |
| 1212 | 002 |
| 1221 | 002 |
| 1021 | 202 |
| 2001 | 202 |
| 1112 | 222 |
| 1211 | 202 |
| 1210 | 022 |

Table 3.3: MOPI for example of Section 3.3.2.

The MOPI program considers up to twelve outputs at a time, this limit being placed in order to operate the program within permissible computing facilities.

### 3.3.3. Logic Minimisation

The problem of logic minimisation has been treated by many authors[36,37,38]. The more commonly used methods are the Karnaugh maps and the tabulation techniques referred to earlier. Several reduction algorithms have also been described recently in the literature[37,38]. These methods generally aim at an absolute minimum solution and therefore require considerable computational effort. However the hardware cost saved by using an absolute minimum solution, instead of a near-minimum solution is often trivial compared to the computation cost. This is even more true because of the tremendous cost reductions of available integrated circuit logic modules. Thus, the minimisation technique should be such that a near minimum solution is obtained with the emphasis being on computation time.

The minimisation technique developed uses the MOPI obtained in the previous section. It separates the MOPI set into two subsets. The first consists of essential prime implicants, all of which must be used in the final answer. The second set contains dispensable minterms all of which are not required in the final answer. Thus, in the near-minimum solution, a minimum number of these dispensable terms are retained together with the first subset of essential minterms.

In order to develop the criterion for dispensability[39], consider the case of a single output network. The output g may be written as a logical sum of the prime implicants. Thus

$$g = Q_1 + Q_2 + Q_3 + \cdot \cdot \cdot Q_n$$

i.e., $$g = \bigcup_{i=1}^{n} Q_i \tag{3.4}$$

When a minterm $Q_j$ implies another minterm $Q_\ell$, the following two identities result.

$$Q_j + Q_\ell = Q_\ell \tag{3.5}$$

and

$$Q_j \cdot Q_\ell = Q_j \tag{3.6}$$

If $Q_j$ implies $Q_\ell$, i.e., $Q_j \rightarrow Q_\ell$ then $Q_j$ is redundant. This result can be extended to a general set of prime implicants as follows:

If $Q_j \rightarrow \bigcup_{\substack{i=1 \\ i \neq j}}^{n} Q_i$ then $Q_j$ is a redundant or dispensable prime implicant, since

$$Q_j + \bigcup_{\substack{i=1 \\ i \neq j}}^{n} Q_i = \bigcup_{\substack{i=1 \\ i \neq j}}^{n} Q_i \tag{3.7}$$

To check if a prime implicant is redundant, the concept of Boolean ratios is introduced. For this purpose, equation (3.7) is divided by $Q_j$. Thus

$$\bigcup_{\substack{i=1 \\ i \neq j}}^{n} \frac{Q_i}{Q_j} = 1 + \bigcup_{\substack{i=1 \\ i \neq j}}^{n} \frac{Q_i}{Q_j} \tag{3.8}$$

Since if A = A + 1 then A = 1, one obtains from equation (3.8) that

$$\bigcup_{\substack{i=1 \\ i \neq j}}^{n} \frac{Q_i}{Q_j} = 1 \tag{3.9}$$

Equation (3.9) is the necessary condition for redundancy of $Q_j$. The Boolean ratio used here is quite different from the normal arithmetic ratios and should be considered further. Consider as an example a system with two prime implicants only, in which case the ratio $\alpha_1 = \frac{Q_2}{Q_1}$ needs to be analysed. If all the literals in $Q_1$ are set to logic 1, then the ratio $\alpha_1$ is given by $Q_2$.

Examples:

$$\alpha_1 = \frac{ab\bar{c}}{b} = a\bar{c} \qquad \text{since } b = 1$$

$$\alpha_2 = \frac{ab\bar{c}}{c} = 0 \qquad \text{since if } c = 1,\ \bar{c} = 0$$

$$\alpha_3 = \frac{a}{abc} = 1 \qquad \text{since } a = b = c = 1$$

Obviously the ratio $\alpha_i$ can also be obtained by forming the logical conjunction of $\frac{Q_i}{Q_\ell}$ and suppressing $Q_\ell$ in it. The dispensability criterion may now be restated.

If $\alpha_i$ denotes the Boolean ratio $\frac{Q_i}{Q_j}$ then the prime implicant $Q_j$ is redundant if the logical sum of all the ratios $\alpha_i (i = 1,2,\ldots,n,\ i \neq j)$ is valid. Standard rules of Boolean Algebra are used to check this validity.

Essential prime implicants are those minterms for which the logical sum of the Boolean ratios can not be valid by any combination of the ratios. Such essential prime implicants must all be present in the final answer for the logic system.

Prime implicants for which the logical sum of the ratios is valid are dispensable terms. If the ratios which cause this validity involve only the essential prime implicants, then the dispensable prime implicant is completely redundant or absolutely dispensable and need not be considered any further in the analysis. If the dispensability is due to ratios which involve essential prime implicants and absolutely dispensable prime implicants, then again that prime implicant is absolutely dispensable. These absolutely dispensable prime implicants are removed from the original list and only the remaining dispensable terms need to be analysed further. The minimisation procedure is now outlined in the Fig. (3.4). The remaining analysis is, therefore, applied to the dispensable terms only.

It has been shown that for a dispensable prime implicant the logical sum of its Boolean ratios is valid (see Eqn. 3.9). This validity can occur due to the logical sum of some of the ratios being valid (since A+1 = 1). Furthermore several combinations of these ratios can cause this validity. These combinations can be represented by presence factors. Thus if a prime implicant $Q_j$ is dispensable due to the logical sum of $\frac{Q_n}{Q_j}$ and $\frac{Q_\ell}{Q_j}$ and also due to the logical sum of $\frac{Q_m}{Q_j}$, $\frac{Q_p}{Q_j}$ and $\frac{Q_q}{Q_j}$ say, then the presence factors for $Q_j$ are $\sigma_j$, $\sigma_n\sigma_\ell$ and $\sigma_m\sigma_p\sigma_q$. These factors imply that in a logical expression involving $Q_j$, either $Q_nQ_\ell$ or $Q_mQ_pQ_q$ may be used. The presence function $(S_j)$ is defined as

$$S_j = (\sigma_j + \sigma_n\sigma_\ell + \sigma_m\sigma_p\sigma_q) \tag{3.10}$$

Use of these presence functions may be understood from the following example.

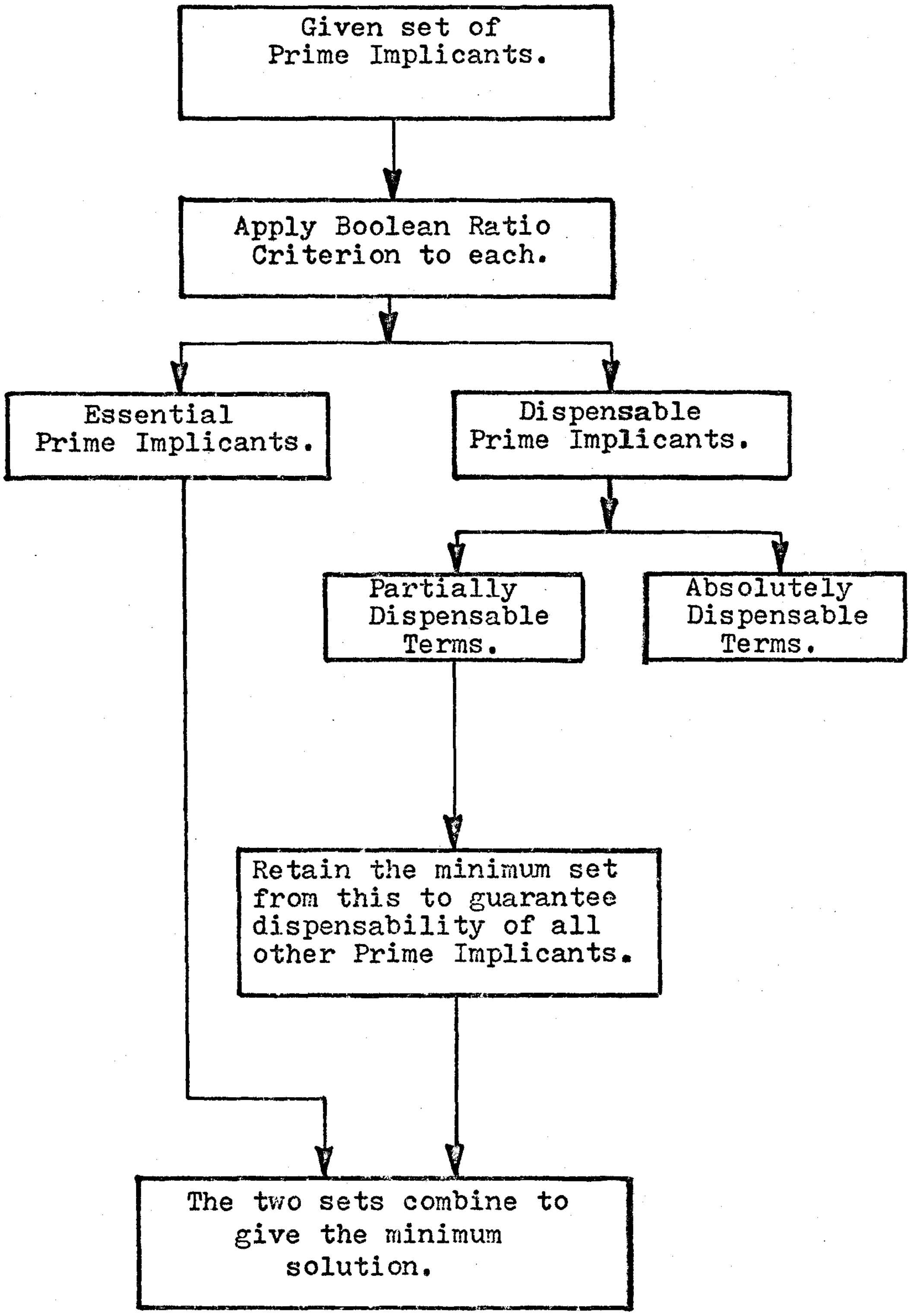

Given set of Prime Implicants.
Apply Boolean Ratio Criterion to each.
Essential Prime Implicants.
Dispensable Prime Implicants.
Partially Dispensable Terms.
Absolutely Dispensable Terms.
Retain the minimum set from this to guarantee dispensability of all other Prime Implicants.
The two sets combine to give the minimum solution.

Figure 3.4: Outline of the general minimisation procedure.

### 3.3.4. Example

To minimise $F = \bar{X}Y + X\bar{Y} + XZ + YZ + WZ$

$$\text{i.e. } F = Q_1 + Q_2 + Q_3 + Q_4 + Q_5$$

The Boolean ratio test gives $Q_1$, $Q_2$ and $Q_5$ as essential implicants. $Q_3$ and $Q_4$ are dispensable with their presence functions being,

$$S_3 = \sigma_3 + \sigma_2\sigma_4$$

and $$S_4 = \sigma_4 + \sigma_1\sigma_3$$

The system presence function ($S_s$) is given by

$$S_s = (\sigma_1\sigma_2\sigma_5)(\sigma_3 + \sigma_2\sigma_4)(\sigma_4 + \sigma_1\sigma_3)$$

$$\therefore \quad S_s = \sigma_1\sigma_2\sigma_3\sigma_5 + \sigma_1\sigma_2\sigma_4\sigma_5 + \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5$$

Either $\sigma_1\sigma_2\sigma_3\sigma_5$ or $\sigma_1\sigma_2\sigma_4\sigma_5$ give the minimum logic realisation for F.

$$\therefore \quad F_{min} = \bar{X}Y + X\bar{Y} + XZ + WZ$$

or $$F_{min} = X\bar{Y} + \bar{X}Y + YZ + WZ.$$

### 3.3.5. Further Analysis of Dispensable Minterms

In the example of section 3.3.4., the dispensabilities are due to combinations of two minterms only. This is not usually the case, especially for large numbers of prime implicants, where some of the combinations causing a validity involve many minterms. In these cases, the problem of choosing an irredundant solution tends to be very time consuming. A near minimum solution obtained within a small computation time is usually quite adequate in such cases. For example when a prime implicant is made dispensable by a combination of a large number of prime implicants then the dispensability is a weak one and it would be sufficient to treat the prime implicant as an essential term. Thus in a practical minimisation approach, for each dispensable prime implicant, all the combinations of prime implicants which cause the dispensability are found. A restriction is placed on the maximum number of prime implicants in each combination. If in each combination, the number of terms involved exceeds this limit, then the dispensable prime implicant is treated as an essential prime implicant.

The dispensabilities are converted into presence factors. Presence factors which involve the least number of other dispensable prime implicants are retained. The process is repeated for all the other dispensable prime implicants. The final presence function calculated from these presence factors will consist of

(a) all essential prime implicants

(b) a minimum or near minimum number of dispensable prime implicants required in addition to (a), to completely define the original problem.

The example of the previous section illustrates the use of (a) and (b).

The general concept of dispensable prime implicants is now extended to MOPI minimisation.

## 3.3.6.. MOPI Minimisation

The multiple output prime implicants of the logic system to be minimised are generated by the program of section 3.3.1 Logic minimisation of large systems can be carried out by subdividing the system into smaller subsystems and applying the minimisation technique to each. The minimisation technique discussed here can handle systems with six logic variables and twelve outputs.

The dispensability criterion can be extended to the MOPI terms. A MOPI $\Phi_j$ consists of a V section, $V_j$ and a KP-section $\zeta_j$. The output vector [F] may be written as

$$[F] = \bigcup_{i=1}^{n} V_i \zeta_i \qquad (3.11)$$

The MOPI $\Phi_j$ can be dispensable if and only if its V-section $V_j$ implies the logical sum of the V-sections of all the MOPI whose $\zeta$ section (i.e., KP section) includes at least $\zeta_j$. This is equivalent to stating that the Boolean ratio for a MOPI can be defined as $\frac{V_i}{V_j}$ if $\zeta_i$ includes at least $\zeta_j$, i.e., if both $V_i$ and $V_j$ are used together in any output line. The dispensability criterion of equation (3.9) may now be generalised to include the MOPI. Thus, if

$$\bigcup_{\substack{i=1 \\ i \neq j}}^{n} \frac{V_i}{V_j} \beta_i = 1 \qquad (3.12)$$

$$\text{where} \quad \beta_i = 1 \text{ if } \zeta_i \text{ includes } \zeta_j$$
$$= 0 \quad \text{otherwise}$$

then $\Phi_j$ is a dispensable MOPI.

For each MOPI, the dispensability criterion of equation (3.12) is applied and the given MOPI set is subdivided into essential MOPI and dispensable MOPI. Absolutely dispensable terms as defined earlier are removed entirely from the set. The remaining dispensable prime implicants are now considered for further analysis.

The flow chart for a minimisation program using the dispensability criterion is shown in Fig. (3.5). The given MOPI set is rearranged so that the output involving the least number of MOPI can be considered first. The set is subdivided into an essential subset and dispensable subset, using the dispensability criterion of equation (3.12). The absolutely dispensable terms are removed from the set of dispensable prime implicants.

For each remaining MOPI a weighting factor $\omega_i$ is defined which denotes the number of times the MOPI is used in the unminimised logic system. For each output line starting with the one with the least number of prime implicants, minimisation is carried out as outlined in section 3.33, using the presence functions. If several minimum solutions result, then all these are retained until all other outputs have been minimised.

Based on the new set of solutions, the weighting factors are re-evaluated. For each output line the solution with the highest sum of the weighting factors is retained. The procedure is repeated for all the output lines. At this stage each output may again have more than one solution. The weighting factors for the MOPI are therefore re-evaluated.

A final step in minimisation is now carried out. If a MOPI has a weighting factor $\omega_i=1$ then tests are made to check if it can be replaced by two or more prime implicants used elsewhere in the system. For example, a MOPI BC with $\omega_i=1$ can be replaced by BCD and $BC\bar{D}$ both of which are used in other output lines. In this way, all the MOPI with a

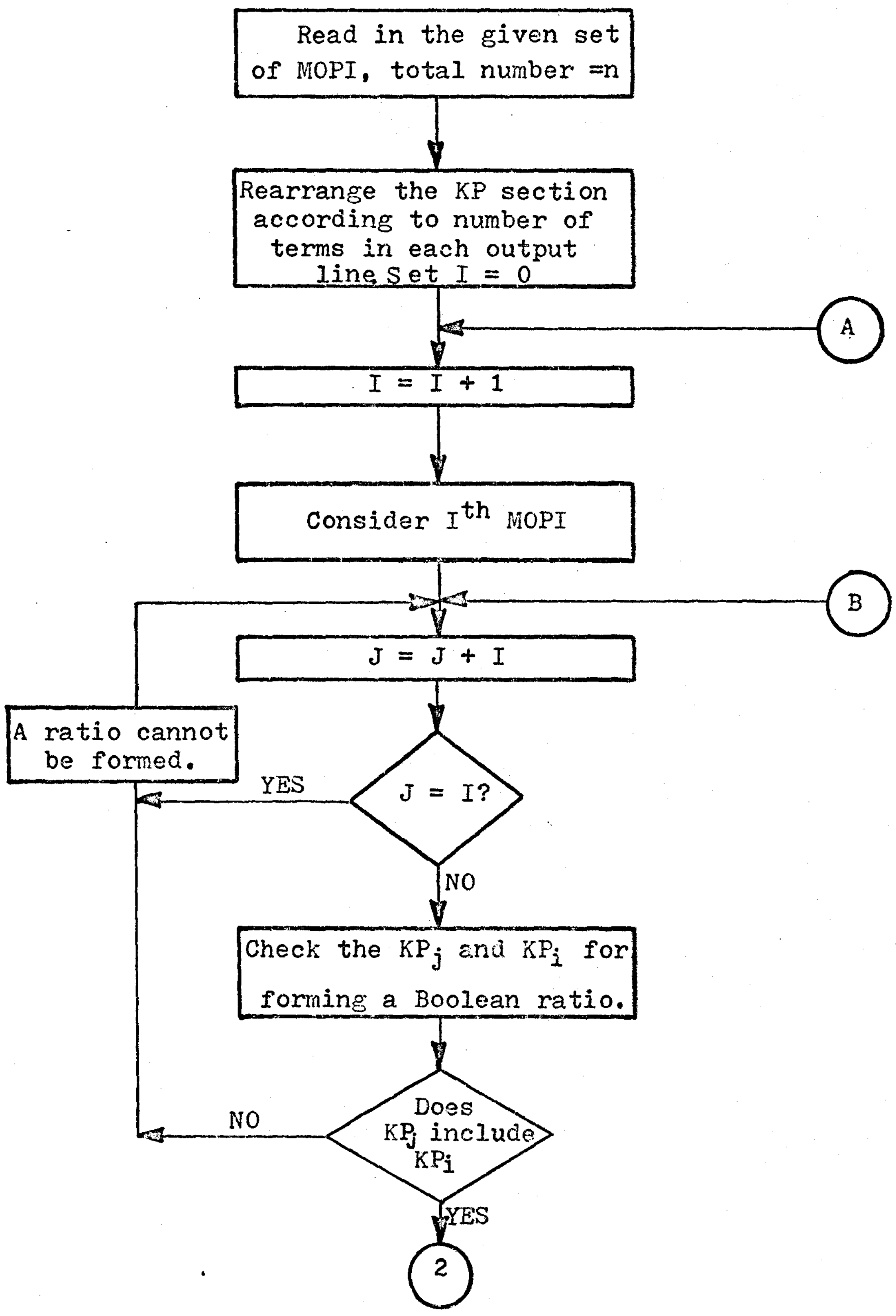

Read in the given set of MOPI, total number =n
Rearrange the KP section according to number of terms in each output line, Set I = 0
A
I = I + 1
Consider I^th MOPI
B
J = J + I
A ratio cannot be formed.
YES
J = I?
NO
Check the KP_j and KP_i for forming a Boolean ratio.
NO
Does KP_j include KP_i
YES
2

Figure 3.5: Flow-chart representation for MOPI minimisation.

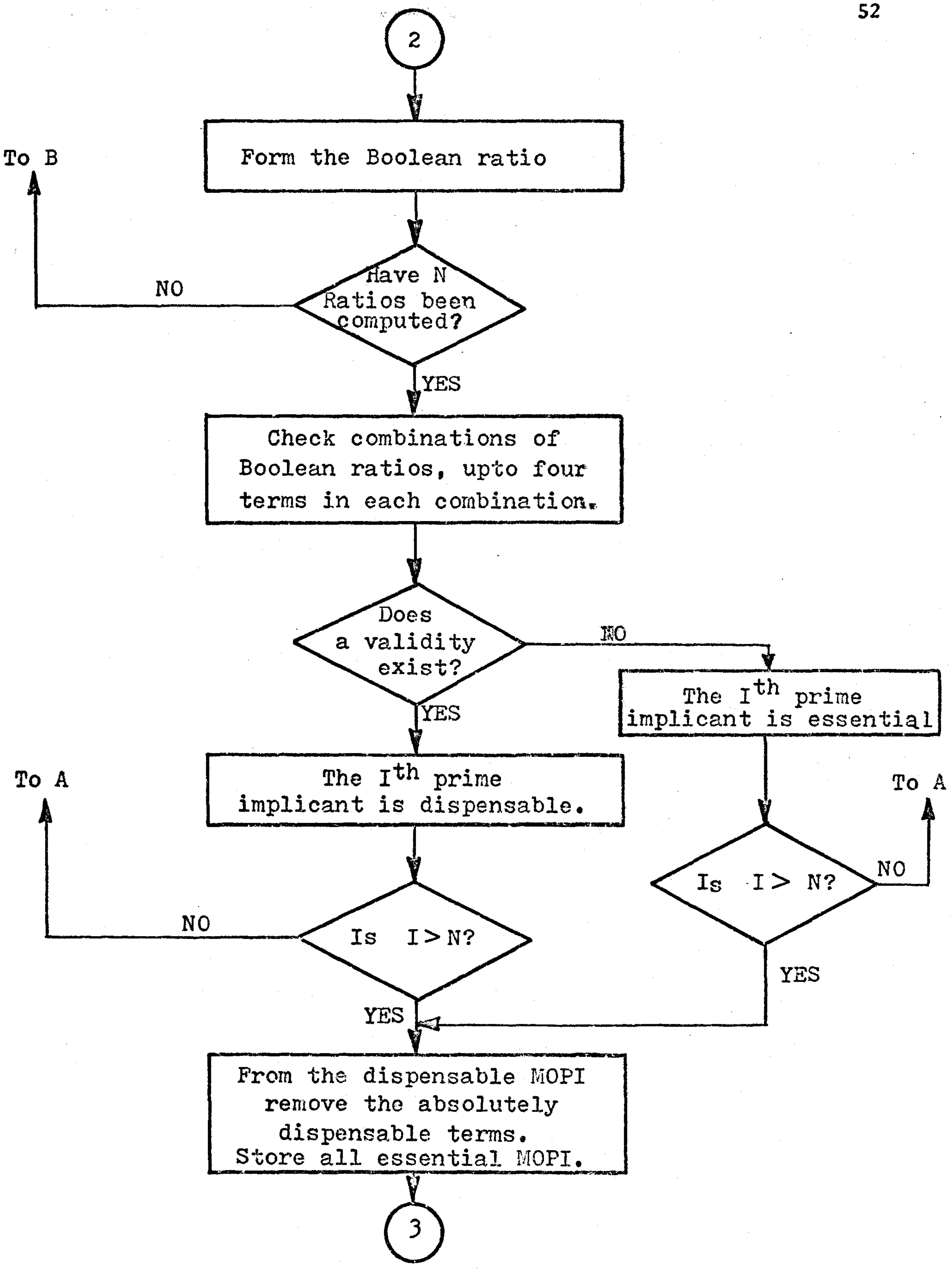

2
Form the Boolean ratio
To B
NO
Have N Ratios been computed?
YES
Check combinations of Boolean ratios, upto four terms in each combination.
Does a validity exist?
NO
YES
The Ith prime implicant is essential
The Ith prime implicant is dispensable.
To A
To A
NO
Is I > N?
NO
Is I > N?
YES
YES
From the dispensable MOPI remove the absolutely dispensable terms. Store all essential MOPI.
3

Figure 3.5: Contd.

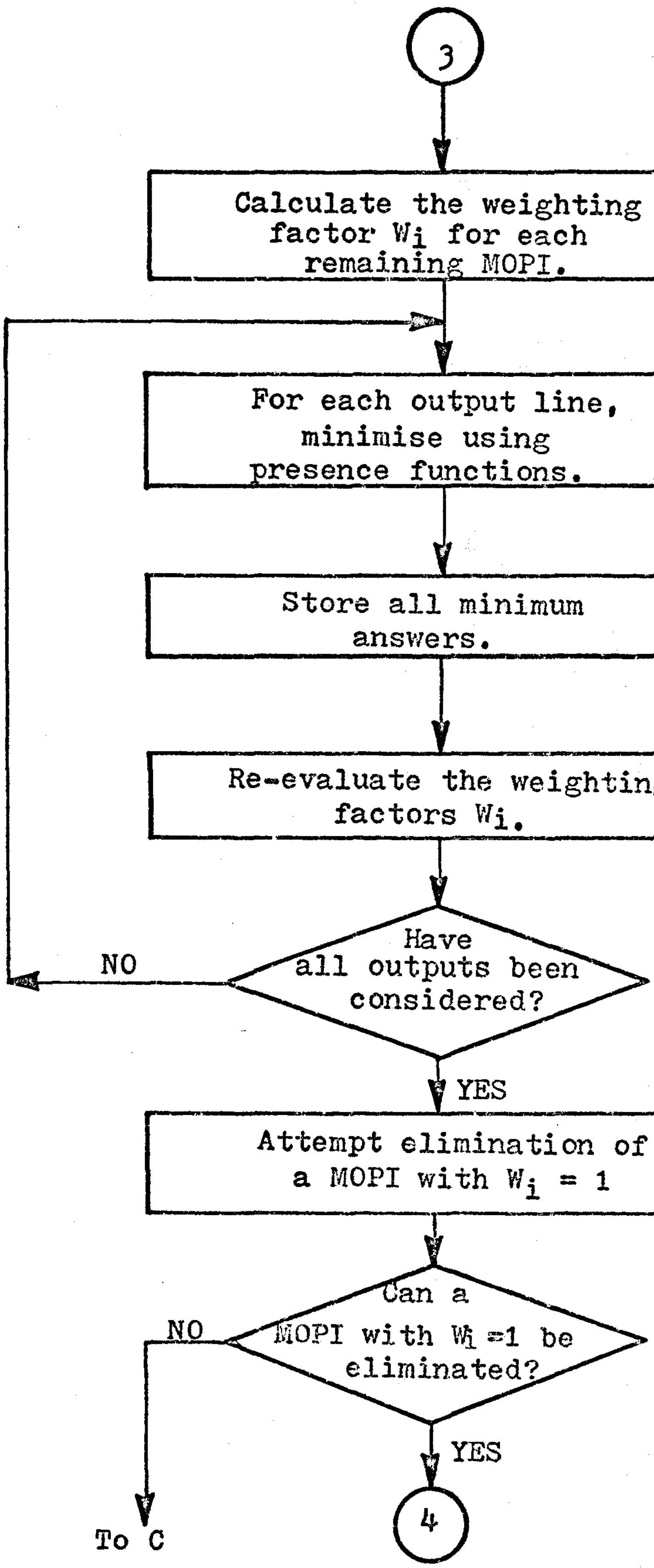
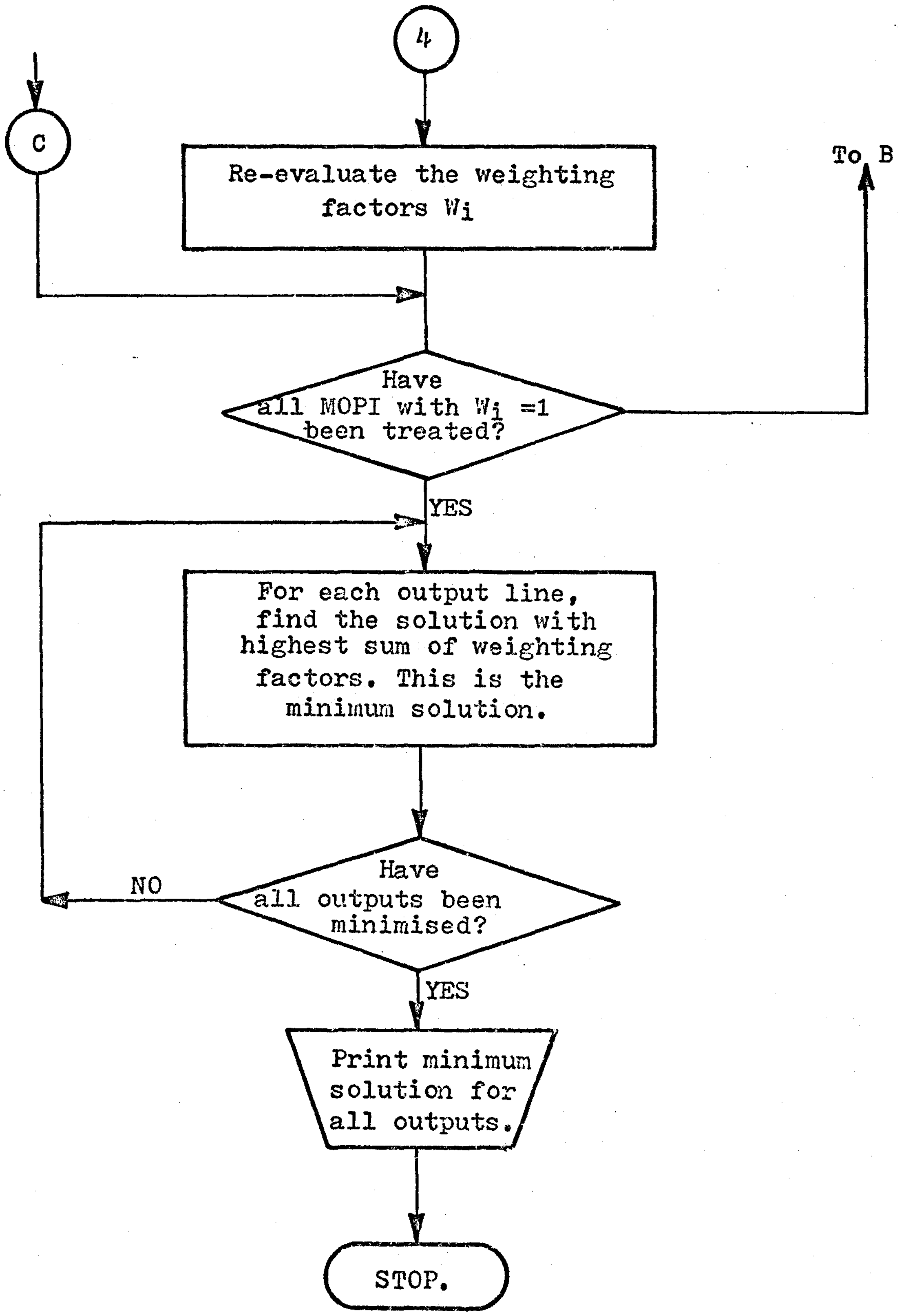

3
Calculate the weighting factor $W_i$ for each remaining MOPI.
For each output line, minimise using presence functions.
Store all minimum answers.
Re-evaluate the weighting factors $W_i$.
Have all outputs been considered?
NO
YES
Attempt elimination of a MOPI with $W_i = 1$
Can a MOPI with $W_i = 1$ be eliminated?
NO
YES
To C
4

Figure 3.5: Contd.



4
C
Re-evaluate the weighting factors $W_i$
To B
Have all MOPI with $W_i$ =1 been treated?
YES
For each output line, find the solution with highest sum of weighting factors. This is the minimum solution.
Have all outputs been minimised?
NO
YES
Print minimum solution for all outputs.
STOP.

Figure 3.5: Contd.

weighting factor of unity are removed if possible and the weighting factors re-evaluated. For each output line the solution with the highest sum of the weighting factors is retained as the best minimised solution.

### 3.3.7. Example

The MOPI determined in section 3.3.2. were minimised using the program of the previous section. The results are

$$Z_1 = bc + \bar{b}\bar{c}d + acd$$
$$Z_2 = bc + b\bar{c} + ac\bar{d}$$
$$Z_3 = acd + ac\bar{d} + b\bar{c} + \bar{b}\bar{c}d \ .$$

## 3.4 Minimised Design of the W.F.L. Unit

The weighted feed logic for the three higher moments (k=2,3,4) requires 12, 18 and 24 output lines respectively for six bit level inputs. Therefore, it is necessary to subdivide the W.F.L. units for the third and fourth moments into two subsystems each. The logic minimisation for the W.F.L. unit, requires five MOPI and five minimisation program runs. The results of the five minimisation programs are combined together to form a listing of all the prime implicants required. In this listing procedure, a number is assigned to each prime implicant. The output lines where a prime implicant is used are also indicated. The prime implicants list is used to design the NAND/NAND logic configuration for each output line and also to evaluate the loading requirement for each MOPI and the input information a, b, c, d, e and f.

| K | TYPE OF IC PACKAGE. | MOPI MINIMISATION. | MAP MINIMISATION. | |
|---|---|---|---|---|
| 2 | 2 I/P NAND(4) | 2 | 2 | |
| | 3 I/P NAND(3) | 7 | 7 | |
| | 4 I/P NAND(2) | 29 | 33 | |
| | 8 I/P NAND(1) | 46 | 46 | |
| | Inverters (6) | 2 | 2 | |
| NUMBER OF PACKAGES FOR 12 OUTPUT LINES. | | 86 | — | 90 |
| 3 | 2 I/P NAND(4) | 1 | 1 | |
| | 3 I/P NAND(3) | 3 | 5 | |
| | 4 I/P NAND(2) | 40 | 55 | |
| | 8 I/P NAND(1) | 97 | 121 | |
| | Inverters (6) | 4 | 3 | |
| NUMBER OF PACKAGES FOR 18 OUTPUT LINES. | | 125 | — | 185 |
| 4 | 2 I/P NAND(4) | 0 | 1 | |
| | 3 I/P NAND(3) | 2 | 3 | |
| | 4 I/P NAND(2) | 10 | 35 | |
| | 8 I/P NAND(1) | 85 | 214 | |
| | Inverters (6) | 5 | 5 | |
| NUMBER OF PACKAGES FOR 24 OUTPUT LINES. | | 102 | — | 258 |
| TOTAL PACKAGE REQUIREMENT FOR W.F.L. UNIT. | | 313 | — | 533 |

NOTE: Number of gates in each package is given in brackets.

Table 3.3: Logic requirements for W.F.L. unit.

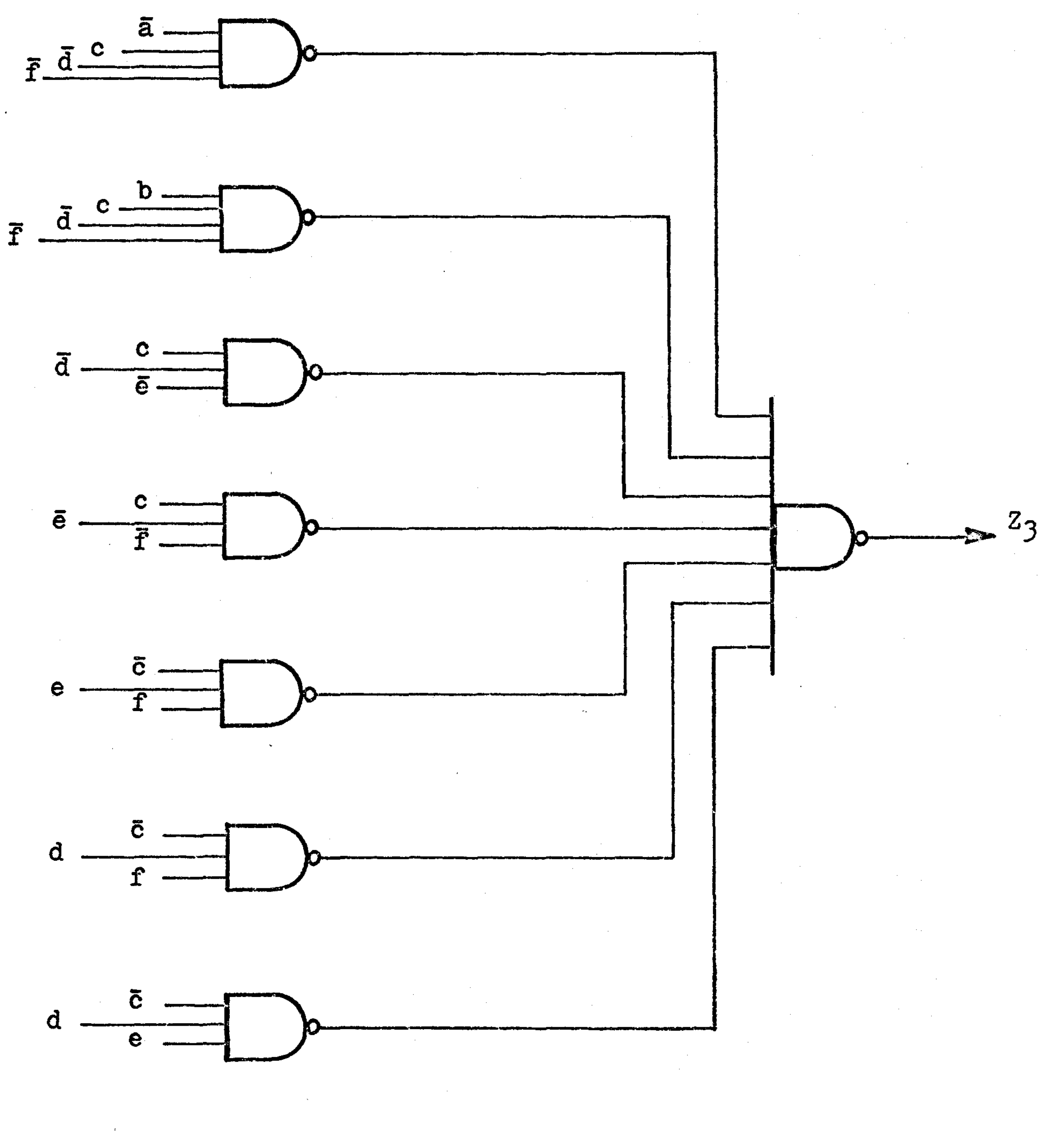


Figure 3.6: NAND/NAND implementation for $Z_3$ of the second moment W.F.L. Unit.

Integrated circuit module requirements for the minimised W.F.L. unit are given in Table 3.3. Results are compared with the module requirements obtained by minimisation of each output line using a six-variable Karnaugh map. It is evident that considerable economy has been achieved. A typical output line implemented with NAND/NAND logic is shown in Fig. (3.6).

## 3.5 Some Comments on Logic Minimisation

Experience with the logic minimisation programs described has shown that absolute minimisation usually requires long computer runs. In such cases, the saving in component costs is often outweighed by the cost of computer time and effort. Any minimisation process, therefore, must take this factor into account. There will certainly be systems where the direct logic realisation of the outputs in terms of all possible minterms may prove more economical. For a six input system, sixty-four minterms need to be realised. If in such a system, the number of outputs is considerably greater than 64 and if the outputs are not well ordered* logical sum of the minterms then the direct realisation should also be investigated. For the weighted feed logic units, direct realisation was discarded since it required a large number of buffer gates thereby increasing the I.C. package requirements.

*A well ordered function is one for which all the minterms combine to give a minimum answer of the type $(A \oplus B) + (C \oplus D) + - - .$

# CHAPTER IV

# DESIGN OF THE SPECIAL PURPOSE COMPUTER

## 4.1 Introduction

The design of the various units of the special purpose computer are considered in detail in this chapter. A precision rectifier is used at the input of the system so that a single a.d. converter is needed for both positive and negative inputs. It is shown that use of one-step parallel binary adders in circulating registers simplifies the timing requirements and also achieves the division by $n^k$ required in the equations for the $k^{th}$ moment.

The evaluation of standard deviation ($\sigma=\sqrt{m_2-m_1^2}$) is also investigated. Since computation of all the moments is continuous, the evaluation of $\sigma$ should also, preferably, be continuous with the final result available soon after the computation of the moments is terminated. A technique for such a computation of $\sigma$ is also described. The circuit for measurement of $\sigma$ is capable of following fluctuating numbers and furthermore the standard deviation can be available in any code.

A general description of the special purpose computer has already been given in Chapter II and a schematic diagram of an implementation was also presented. In this chapter, the individual units are considered and their design and operation are described.

## 4.2 Rectification

Implementation of n-level quantisation on either side of the zero level has the disadvantage of requiring separate weighted feed logic units to obtain the weighting factors for positive and negative inputs. Furthermore the logic required in these units is complicated by the need to code negative levels for 1's or 2's complement arithmetic.

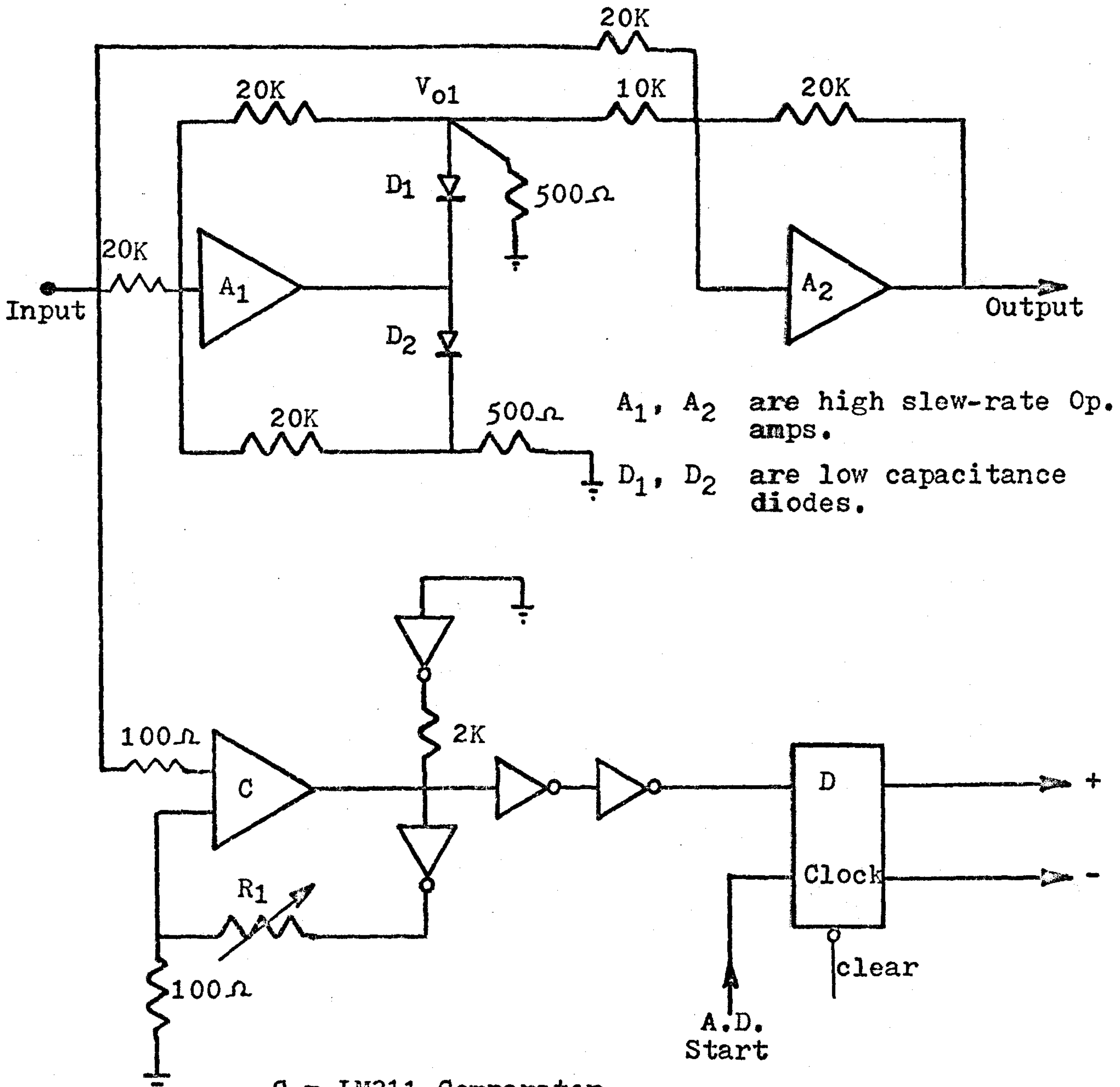
A possible alternative investigated, is the use of levels on one side of zero level only and shift the input by adding a d.c. voltage $\frac{V_n}{2}$. This possibility was discarded because of the following disadvantages.

(i) For a six-bit a.d. converter, one has effectively only five-bit quantisation when the input is shifted by adding the d.c. voltage. Thus, although the entire s.p.c. is designed for six-bit quantisation, the accuracy obtained is that of a five-bit s.p.c.

(ii) For a fixed upper voltage (+10V), the quantisation intervals are halved in voltage range and hence the errors due to level offsets will be increased.

(iii) All the final readings for moments must be corrected for the d.c. voltage added to the input. This can be done in the weighted feed logic by assigning negative weights to samples below the half level value, again complicating the design of the W.F.L. units considerably. If however, the W.F.L. units discussed in Chapter III are used, then the special purpose computer is no longer a direct reading system.

Rectification is, therefore, preferred. A precision rectifier[40], consisting of two high slew-rate operational amplifiers (250V/μsec) and stable precision resistors, is used. The rectifier circuit is shown in Fig. (4.1)



$A_1$, $A_2$ are high slew-rate Op. amps.

$D_1$, $D_2$ are low capacitance diodes.

C = LM311 Comparator.

$R_1$ is adjusted for 25 mv. hysteresis.

Figure 4.1: Rectifier and Sign detection circuit.

When the input ($V_{in}$) to the rectifier is negative, the voltage $V_{o1}$ is essentially zero and the second operational amplifier acts as an inverting amplifier with unity gain and an output given by

$$V_o = -V_{in} \quad . \tag{4.1}$$

When the input ($V_{in}$) is positive, the diode D1 conducts making $V_{o1}=-V_{in}$, since the gain of the first inverting amplifier is also unity. The second amplifier now acts as a summing and inverting amplifier with an output given by

$$V_o = -V_{in} + 2V_{in} \quad . \tag{4.2}$$

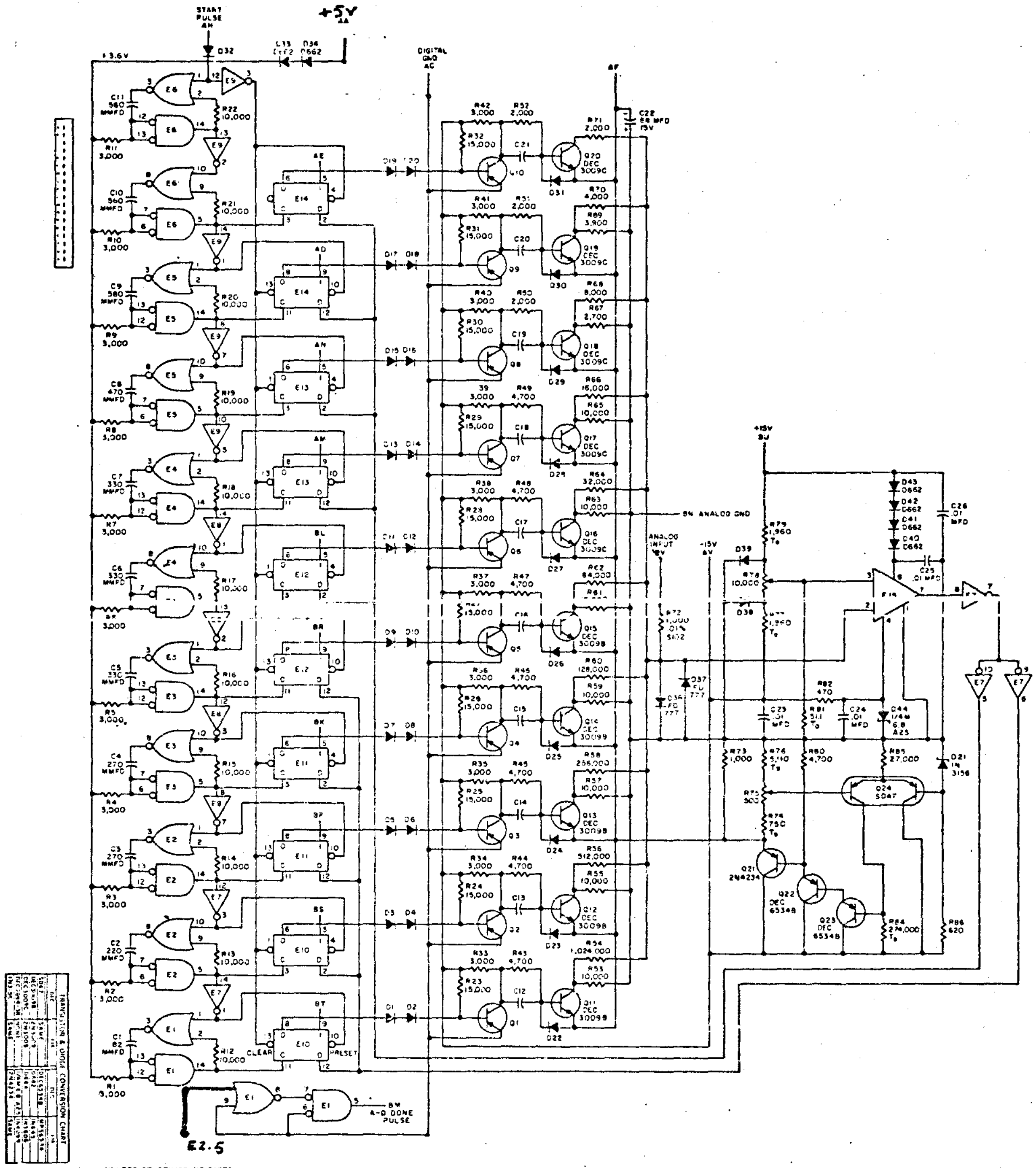
With the precision resistors selected to maintain the appropriate ratios accurately, the error and the distortion in the output can be made very small. Thus the rectifier designed for the special purpose computer has a maximum error of 5 mV for the input in the range ±25mV to ±10V. Distortion in the output is negligible for the frequency range D.C. to 5 kHz. The loss of 5 mV in the signal causes errors in the measured values of the moments. These errors are treated in Chapter V.

The sign detection circuit is also shown in Fig. (4.1). The resistor $R_1$ is adjusted to produce a hysteresis of about 25 mV in the switching characteristic of the comparator. This hysteresis is necessary in order to eliminate erratic switching of the comparator due to additive noise in the input signal. A D-type flip-flop, used to store the sign information at the beginning of each sampling operation, eliminates the change of sign occurring during the sampling and conversion process.

## 4.3 Analog-Digital Converter

An 8-bit successive approximation a.d. converter with a built-in reference supply is used[41]. The converter circuit is shown in Fig. (4.2). Conversion is initiated by raising the convert input (i.e., a.d. start) to logic 1. The digital output bits for a sample are available at 1 μsec. rate, with the conversion completed within 8 μsec. The maximum sampling rate is, therefore, limited to 100 kHz. Of the bits available after conversion, all eight are used in the computation of the first moment, whereas only six most significant bits are required in the computation of the higher moments. After the completion of an 8-bit conversion, an a.d. done pulse is produced by the converter as shown in the timing diagram of Fig. (4.3). The a.d. done pulse is used in the accumulation for the various moments. The samples are taken on the rectified signal. At the end of an a.d. conversion, therefore, the highest level exceeded information is available in magnitude and sign form. The highest level exceeded information is used as an address input to the W.F.L. units. For odd-order moments, the output lines of the corresponding W.F.L. units are gated by the sign bit to positive or negative accumulators. For the even-order moments, the sign bit information is redundant and is therefore not used.

Control of the a.d. converter sampling rate is provided by the clock rate circuit.



UNLESS OTHERWISE INDICATED:
TRANSISTORS ARE DEC2894-38
RESISTORS ARE 1/4W, 5%
DIODES ARE D664
CAPACITORS ARE 47MMFD
PIN 7 = GND ON E10, E11, E12, E13 & E14
PIN 14 = +5V ON E10, E11, E12, E13 & E14
PIN 4 = GND ON E1, E2, E3, E4, E5, E6, E7, E8 & E9
PIN 11 = +3.6V ON E1, E2, E3, E4, E5, E6, E7, E8 & E9
E1, E2, E3, E4, E5 & E6 ARE MC724
E7, E8 & E9 ARE MC789
E10, E11, E12, E13 & E14 ARE DEC7474N
E15 IS μA710
R66, R68, R70 & R71 ARE VISHAY M.F. .01% 5PPM
R58, R60, R62, R64, R74 & R76 ARE M.F., 1/8W, 25 PPM 1%
R56 IS 1/4W, M.F. 25PPM 1%
R54 IS 1/2W, M.F. 25PPM 1%
R77, R79, R81 & R84 ARE M.F., 1/8W, 100PPM 1%
R75 IS A BOURNS TRIMPOT TYPE 224L DEC PART # 13-5136
R78 IS A BOURNS TRIMPOT TYPE 224L DEC PART # 13-05391

Figure 4.2: A.D. Converter.

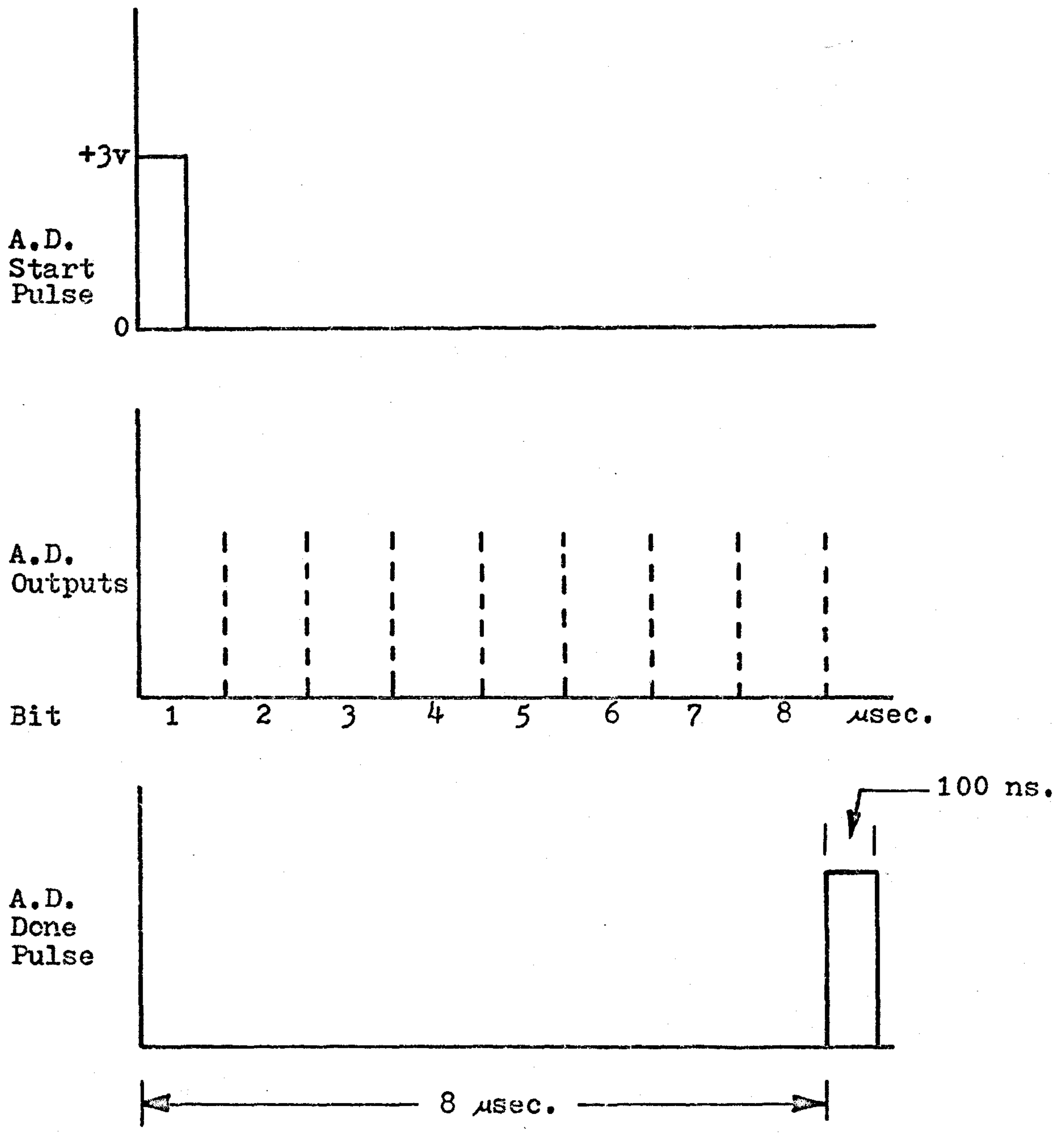


Figure 4.3: Timing diagram for the A.D. Converter.

## 4.4 Clock-rate Circuit

A master clock is used in the clock rate circuit. The pulse rate of the master clock is fixed at 1 MHz and divide-by-ten TTL circuits are used to provide lower rates down to 1 Hz. A two-input OR gate is provided in the master clock circuit for start and stop control, which in the case of the special purpose computer, is obtained from a timing control circuit.

The complete clock rate circuit is shown in Fig. (4.4). The clock output is selected by a rotary switch. Since the a.d. converter start pulse must not exceed a width of 500 ns., a pulse shortening circuit is used at the output of the rotary switch. The output of this circuit provides the a.d. start pulse for the a.d. converter. It also feeds an output counter which measures the total sample size and therefore the measurement time.

## 4.5 Control Circuits

Two control circuits are provided in the special purpose computer:

(i) Mode control which allows the system to be used either for the measurement of the four moments or for the above level probability measurements (see section 4.8).

(ii) Timing control which enables the measurement to be made for one cycle of the input (if this is periodic) or for a predetermined sample size $C_o$.

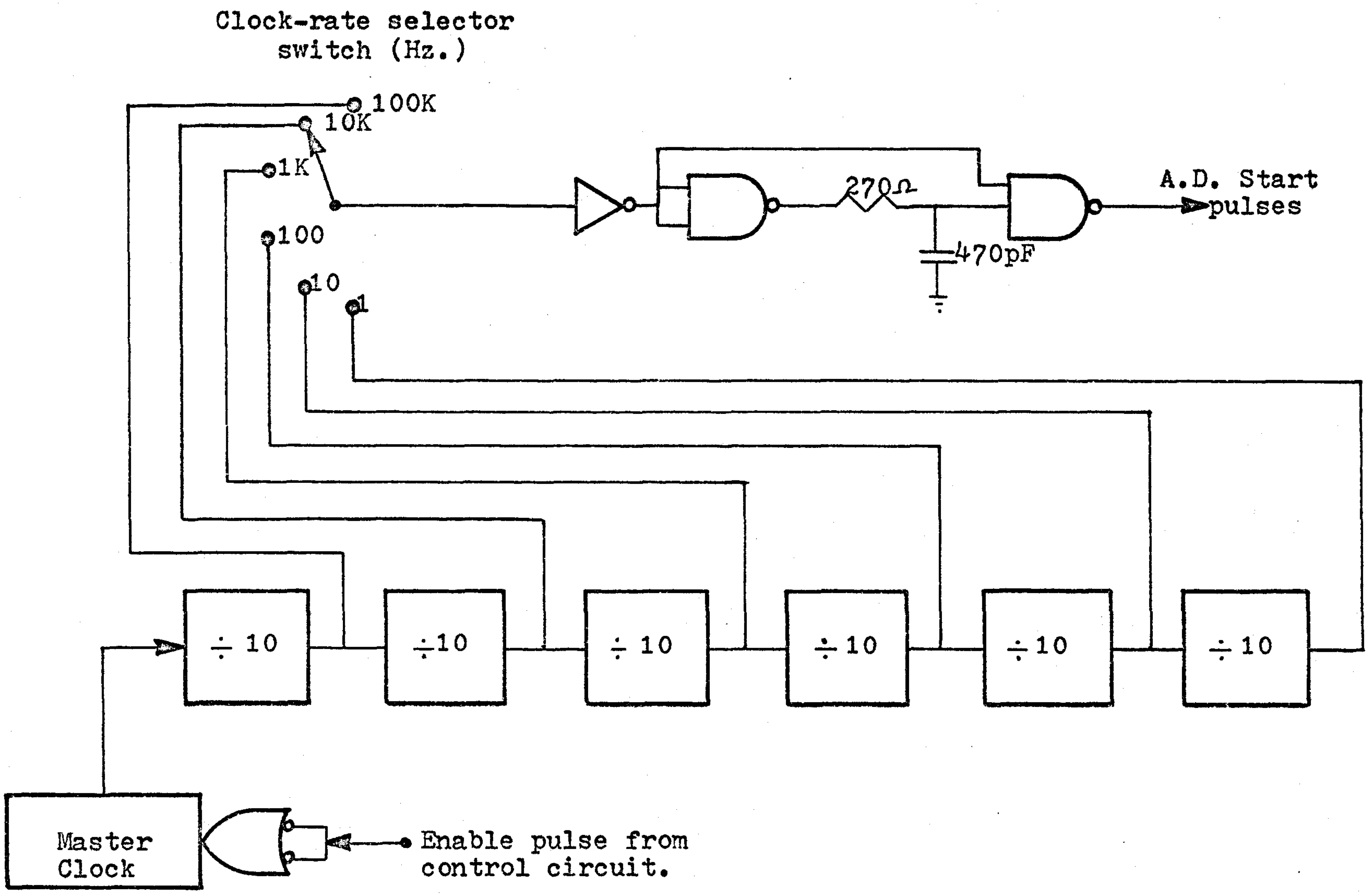


Figure 4.4: The clock-rate system.

In the probability mode, the a.d. done pulse is transferred by a selection switch to the output gate of the probability measuring circuit and all the accumulators used in the computation of the moments are inhibited. The first moment counter and display is also used for the probability display. In the moments' mode, the a.d. done pulse is transferred to the gating circuits of the various accumulators. A sign selection switch used in the probability mode only, allows measurements to be made on either side of the zero level.

The s.p.c. can be operated either in the one cycle mode or in a fixed-sample-size mode in which the sample size is chosen externally and can be $10^3$, $10^4$, $10^5$ or $10^6$. The timing control circuit is shown in Fig. (4.5). The sign detection comparator is used as a zero-crossing detector. Both the flip-flops are initially cleared in either timing mode. In the one cycle mode, the comparator output's negative going edge (β-transition) occurs at a zero crossing and triggers the first JK flip-flop. The $\bar{Q}_1$ output of this flip-flop goes to 0 and enables the sample-rate master clock. At the next β-transition which occurs precisely at the end of one cycle of input, $\bar{Q}_1$ goes to logic 1 and thereby inhibits the master clock, terminating the sampling process. At the same time $Q_1$ triggers the second flip-flop whose $\bar{Q}_2$ output goes to logic 0 which in turn inhibits the first flip-flop. The s.p.c. can be re-started only after clearing the two flip-flops.

In the fixed-sample-size or the fixed time mode, measurements commence upon pressing the manual start switch and setting the R-S flip-flop. The $Q_R$ output goes to logic 0 and enables the master clock. Stop pulses are produced by the sample size counter at $10^3$, $10^4$, $10^5$ or $10^6$

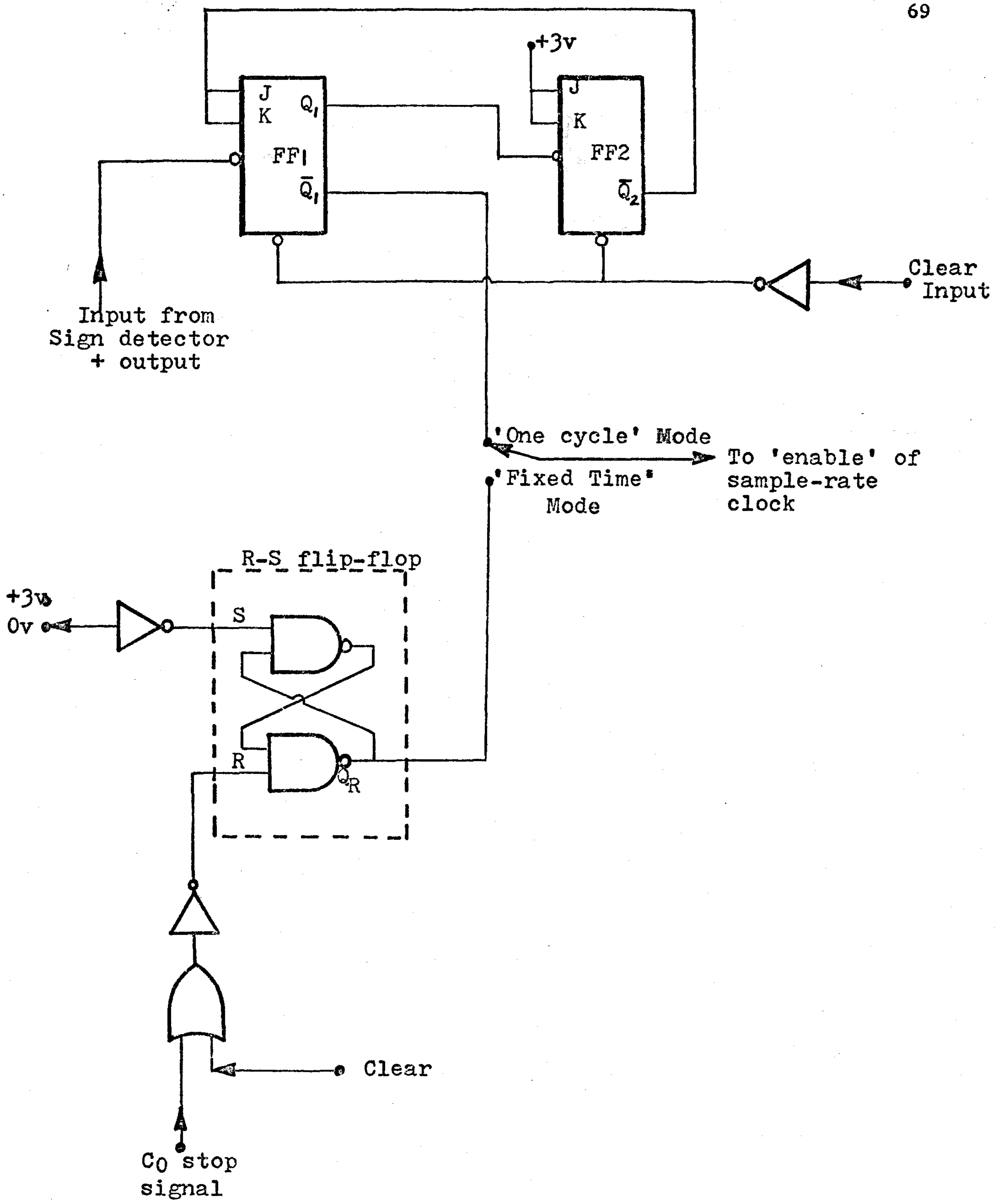


Figure 4.5: Control circuit for the S.P.C.

samples. Any one of these pulses is used to reset the R-S flip-flop which in turn disables the master clock. The R-S flip-flop also eliminates any contact bounce effects in the start switch.

## 4.6 Accumulators

The a.d. converter output gives r the highest level exceeded at the sampling instant. This together with the sign bit, feeds the W.F.L. units. The weighting numbers are computed using the magnitude of r, with the sign bit being used to gate the weighting number output lines for odd-order moments, to the positive or negative accumulators.

If a serial type multiple feed accumulator[17] is used, then for the kth moment and $\ell$-bit quantisation, $\ell k$ clock pulses are required to add the weighting numbers for a sample to the previous accumulation. Assuming a propagation delay of $\tau_m$ sec. for each stage of the serial accumulator then the total time required before the next sample can be taken, is $2.\ell.k.\tau_m$ sec. This is calculated by taking into consideration the requirement that the feed lines for the weighting numbers, must be applied serially into the accumulator and furthermore, the least significant bit applied at $(\ell k)^{th}$ pulse must be allowed to propagate through the entire length of the $\ell k$-bit register before the next sample can be taken. Typically $\tau_m$=50 ns. and for the fourth moment, the total delay is 2.4 μsec. An additional clock system for each moment is also required to produce the $\ell k$ pulses following every a.d. done pulse. At this advanced state of the art, the serial feed accumulator has been superseded and is, therefore, not economical to implement. It would also result in considerable reduction in the upper frequency limit of the s.p.c.

A one-step addition and accumulation is easily implemented with TTL circuit modules. A complete M-bit accumulator, requiring an M-bit adder and storage, is shown in Fig. (4.6). Initially all the JK flip-flops are cleared. The A inputs to the binary adder are therefore 0 and the B inputs are the output lines of the W.F.L. unit for a given $k^{th}$ moment, gated by the sign bit where necessary. The 4-bit adder modules used in the adder circuit have a delay of 50 ns. so that for the fourth moment, the maximum total delay for addition is 300 ns. The final carry-out of the M-bit adder occurs when the total accumulated, exceeds or equals $2^{\ell k}$ $(M=\ell k)$. A similar overflow will occur for the accumulation of negative samples. Since $2^{\ell k}=n^k$, the overflow represents contributions to the $k^{th}$ moment. These overflow signals, gate the a.d. done pulse to the display counters and display units. The remainder which is available as the $\sum$ outputs of the binary adder, is shifted to the JK flip-flop outputs by the a.d. start pulse. Thus, in general, the inputs to the adder after the $(i+1)^{th}$ sample are the weighted feed outputs for the $(i+1)^{th}$ sample as B inputs and the remainder of

$\frac{1}{2^{\ell k}} \sum_{j=1}^{i} W_{k,j}$ as the A inputs.

Division by $n^k$ required in equation (2.30) is therefore easily achieved. Furthermore, since the computation of the various moments is a simple counting procedure, any code may be used in the final display. The fractional value, remaining in the accumulator after processing is complete, is usually negligible for sample size exceeding $10^3$ and therefore is seldom required.

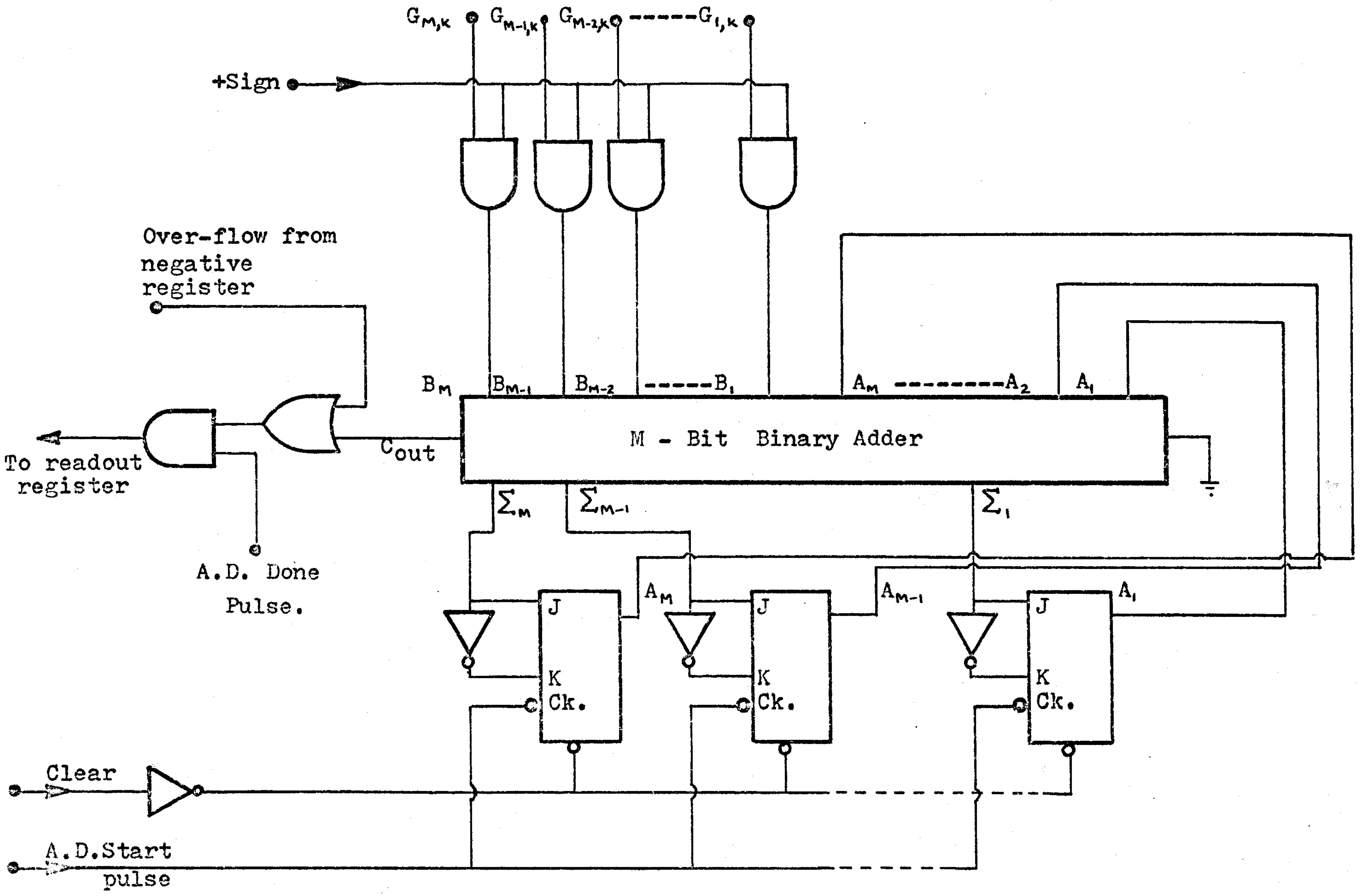


Figure 4.6: A generalised M-bit accumulator.

The total delay in accumulation is equal to the time required to carry out $\ell k$-bit addition. In the s.p.c. where TTL binary adders and flip-flops are used, the worst case delay is in the fourth moment and equals 300 ns. Thus all computations for a sample are completed soon after the sixth bit of the level r is available.

For the first and third moments, two separate accumulators are required to accumulate $m_{k+}$ and $m_{k-}$. The overflows from these accumulators, together with the a.d. done pulse, feed an UP/DOWN counter whose count direction is controlled by the sign bit. For even-order moments, only UP counters are required.

## 4.7 Above-level Amplitude Probability Distribution Measurements

Above-level probabilities (i.e., cumulative probability distribution) are measured using a digital comparator. In the s.p.c. designed, 4-bits or 16 levels are used. The circuit is shown in Fig. (4.7). The A inputs to the 4-bit digital comparator are the four m.s. bits of the a.d. converter and the R inputs are the four reference bits selected by a rotary switch. When $A \geq R$, the comparator output switches to logic 1. This together with the sign bit gates the a.d. done pulse to a counter. Thus, for a sample size $C_o$, this counter readout $C_r$ divided by $C_o$ is approximately the probability that a reference level R is exceeded. The reference level can be varied sequentially and measurements are repeated to obtain sixteen values of the above-level probability on either side of the zero level.

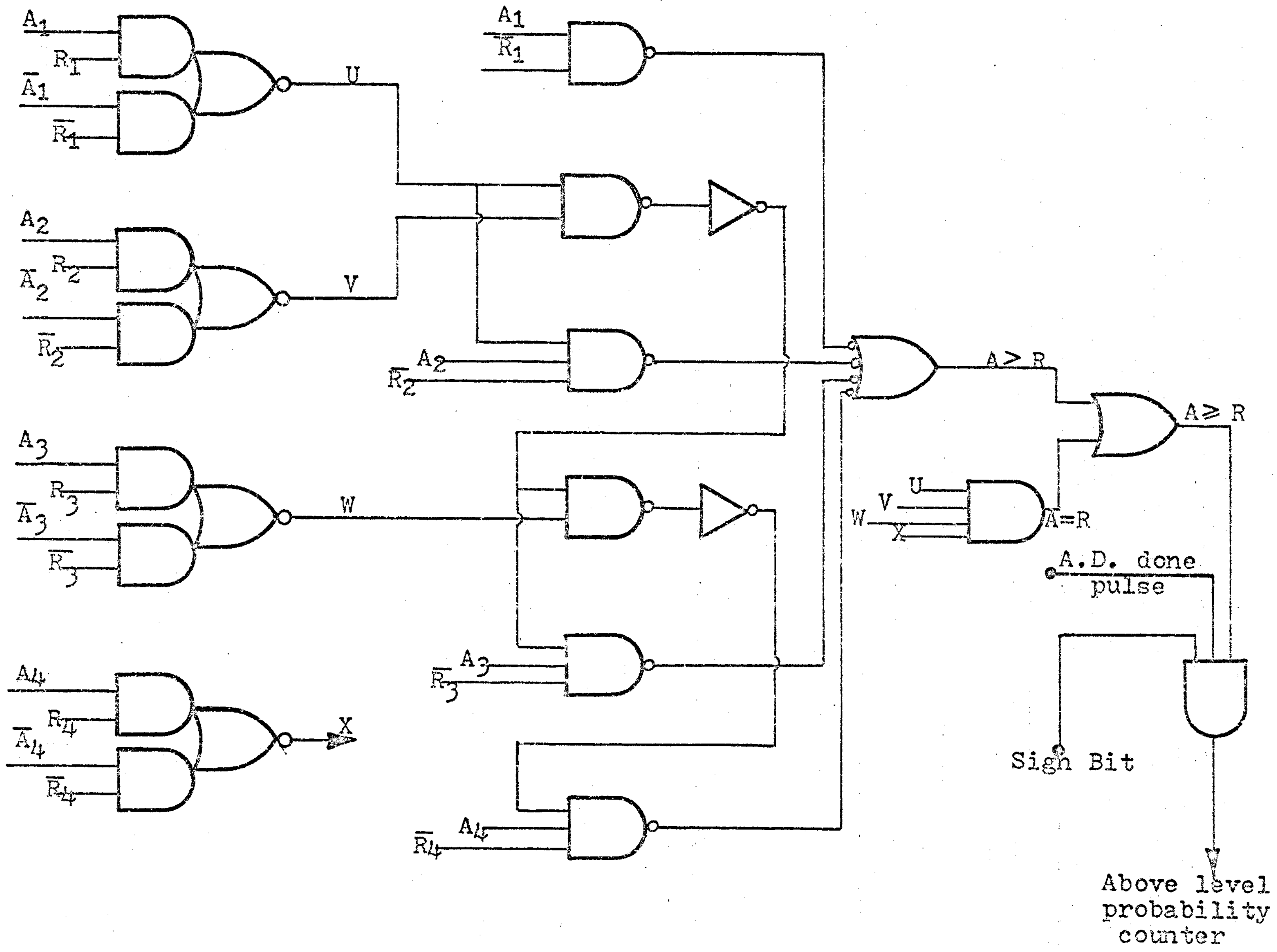


Figure 4.7: Probability distribution circuit.

## 4.8 Display Registers and Readouts

The display counters built use TTL circuit modules throughout, including the drivers for the Nixie display tubes. Standard designs are used in both the UP/DOWN and the UP counters. The upper frequency limit is about 5 MHz. The UP counters have the additional facility of output pulses when the count reaches $10^3$, $10^4$, $10^5$ or $10^6$ counts. These pulses are used in the control circuit, to terminate computation.

The s.p.c. is shown in open rack form in Fig. (4.8) and Fig. (4.9). The a.d. converter and the master clock are on two plug-in cards. The logic for the entire s.p.c. uses TTL circuits and is on eight 6" x 6" printed circuit boards (see Fig. (4.9)). The complete system is contained in a 4-rack 19" x 5" stand and includes all necessary power supplies. Two indicator lights are also provided on the control panel, one to indicate that sampling is in progress and the other a warning to indicate if the input exceeds +10V, the maximum voltage limit of the a.d. converter.

The accumulators, displays and the control circuit are all cleared by a single clear pulse.

## 4.9 Performance Tests

Direct decimal readouts (D.D.R.) for d.c. input voltages, provide a very convenient method for testing the entire s.p.c. If the d.c. input voltage is between the $r^{th}$ and $(r+1)^{th}$ level, then the readouts for the various moments correspond to the mid-interval voltage $\frac{2r+1}{2n}$ corresponding to the $r^{th}$ interval. The expression for the $k^{th}$ moment, developed earlier, gives

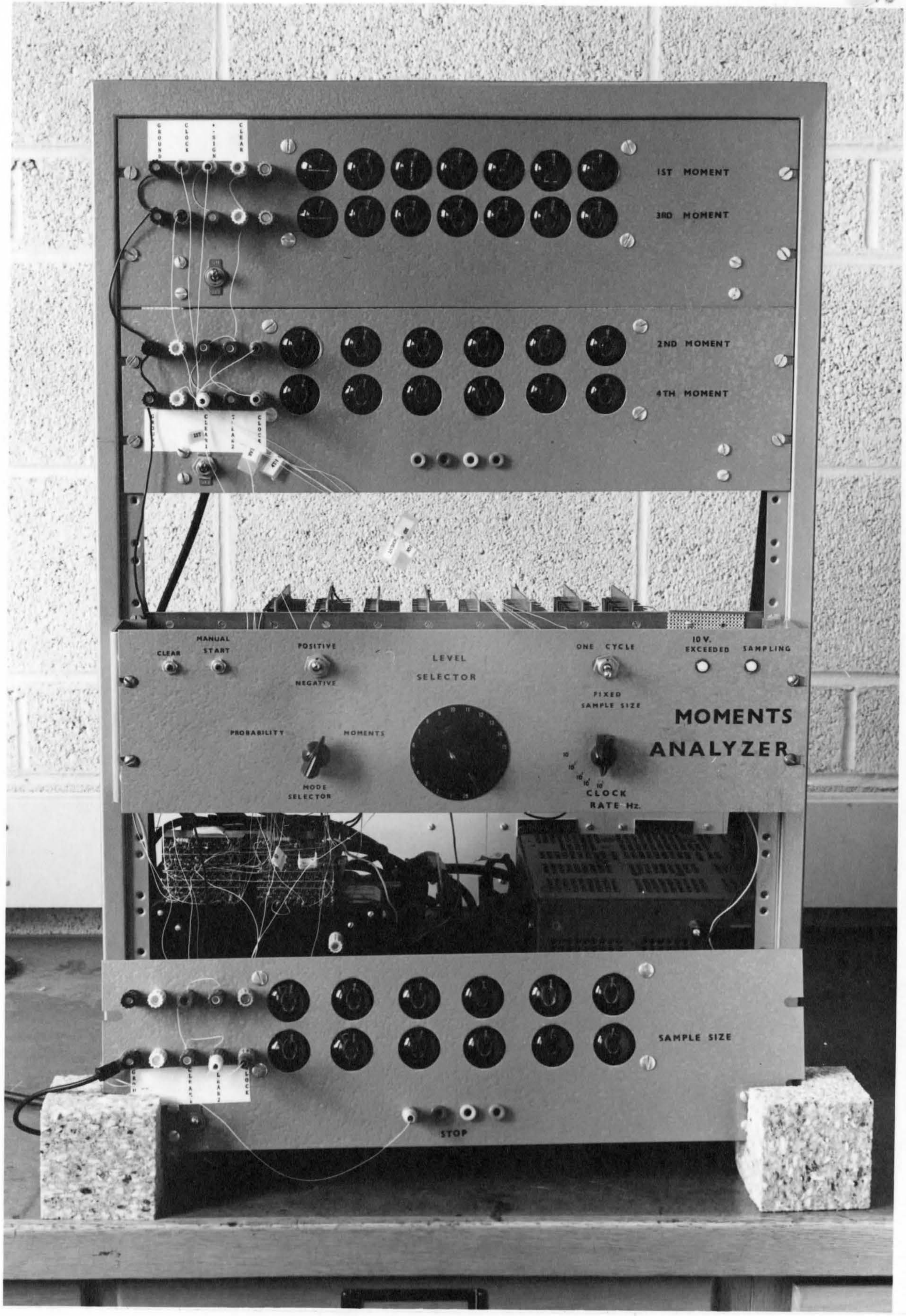


Figure 4.8: Photograph of the S.P.C.

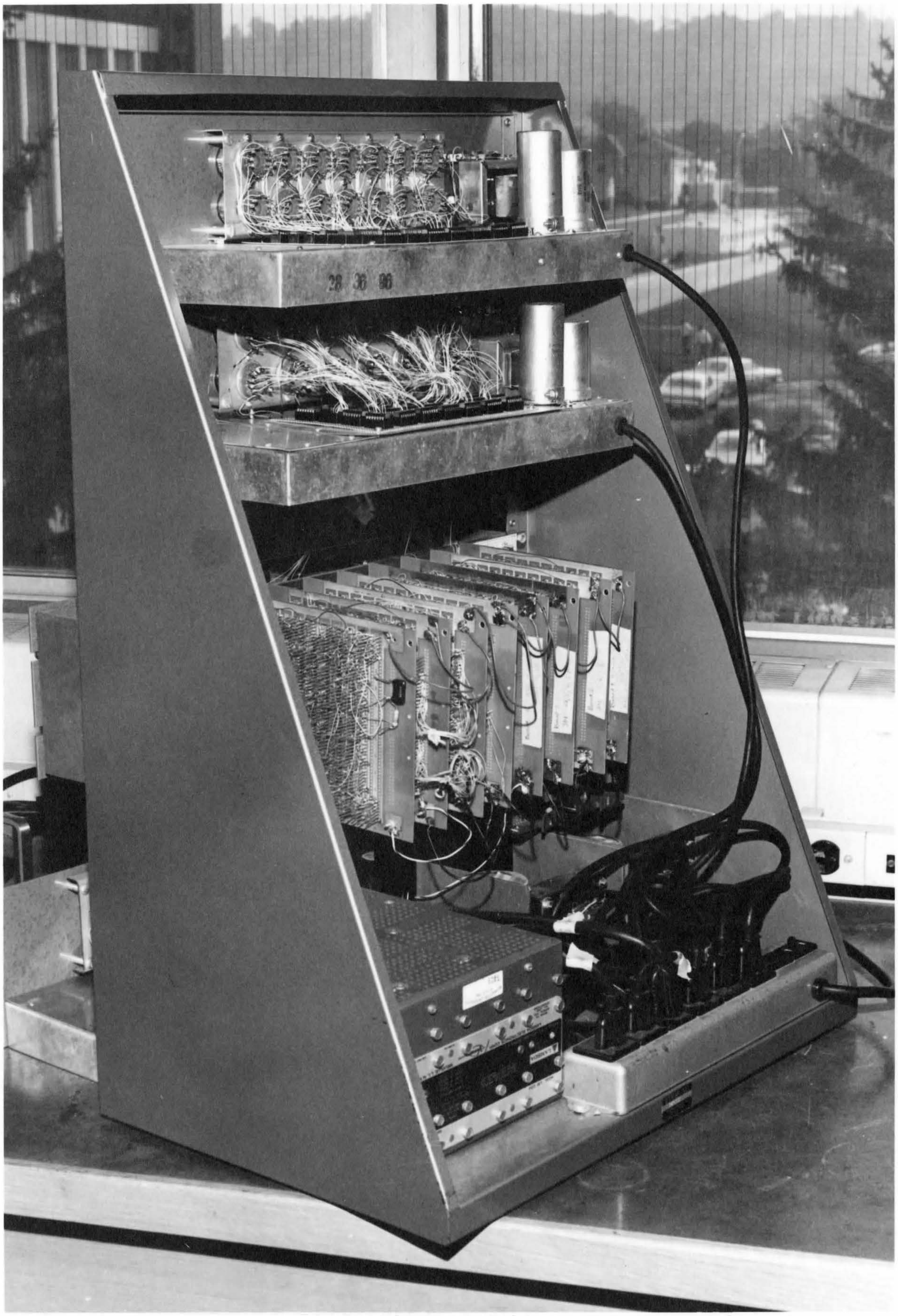


Figure 4.9: Rear-end view of the S.P.C.

$$\{ \frac{2r+1}{2n} \}^k = \frac{1}{(2n)^k} + \frac{1}{n^k C_o} \sum_{r=1}^{n-1} k\, r^{k-1} C_r \, . \tag{4.3}$$

The direct decimal readout $(DDR)_{k,r}$ is defined as

$$(DDR)_{k,r} \triangleq \frac{1}{n^k} \sum k\, r^{k-1} C_r \tag{4.4}$$

Thus,

$$(DDR)_{1,r} = \frac{r.C_o}{n} \tag{4.5}$$

$$(DDR)_{2,r} = \frac{C_o.\, r(r+1)}{n^2} \tag{4.6}$$

$$(DDR)_{3,r} = \frac{C_o.\, r(4r^2 + 6r + 3)}{4n^3} \tag{4.7}$$

and

$$(DDR)_{4,r} = \frac{C_o.\, r(2r^3 + 4r^2 + 3r + 1)}{2n^4} \tag{4.8}$$

The direct decimal readouts are computed for d.c. voltage inputs in the range 0 to 10V with $C_o=10^4$ and n=64. The values are given in Appendix A together with the computer program flow chart for this computation. These values are used to check the entire s.p.c. The a.d. converter switching characteristic can also be checked by measuring the voltage levels at which the output readings change, while the rectifier performance may be checked by using d.c. voltage inputs of both polarities.

The s.p.c. of Fig. (4.8), has been tested for these direct decimal readouts and has also been used to measure the moments of known signals available in the laboratory. The overall performance is within the specifications of 1% overall error for input signals of frequencies up to 5 KHz[42].

## 4.10 Extension to Measure the Standard Deviation

The standard deviation ($\sigma$) of a random signal is defined as

$$\sigma = \sqrt{m_2 - m_1^2} \tag{4.9}$$

Several methods have been proposed for obtaining the square-root of a binary number. These use either the Newton approximation[43] or the rational Chebyshev approximations valid in a given range of the number[44].

Recently direct restoring and non-restoring square root extractions using cellular arrays have also been considered (see Chapter VI). In all these methods, the square root answer is in a binary code and therefore code conversion circuits are necessary if the final readout is to be in any other code. Additional circuits, to compute $m_1^2$ and perform the subtraction in equation (4.9), are also required. Such methods are, therefore, not suitable for special purpose computations.

The square-root method described here[45] is especially suitable for such applications. It is capable of following a fluctuating number N whose square-root is desired. Furthermore, subtraction, squaring and square-root extraction are all implemented in a single circuit, making the method economical for implementation for large numbers.

Consider first the square-root algorithm.

### 4.10.1. The Square-root Algorithm

The algorithm is an extension of one proposed by Phister[46]. This extension provides a particularly fast and economical means of implementation. In the algorithm a number S is compared with N and one of two procedures is carried out, depending on whether $S<N$ or $S>N$. Let j denote the jth step in the process.

(i) For $S<N$

Let $$x_{j+1} = x_j + 1 \tag{4.10}$$

and $$S_{j+1} = x_{j+1}^2$$

$$= x_j^2 + 2x_j + 1$$

These may be written

$$\begin{aligned} x_{new} &= x_{old} + 1 \\ S_{new} &= S_{old} + 2x_{old} + 1 \end{aligned} \tag{4.11}$$

(ii) For $S>N$

Let $$x_{j+1} = x_j - 1 \tag{4.12}$$

and $$S_{j+1} = x_{j+1}^2$$

or $$S_{j+1} = x_j^2 - 2x_j + 1 \tag{4.13}$$

These may be written

$$x_{new} = x_{old} - 1$$

and

$$S_{new} = S_{old} - 2x_{old} + 1 \tag{4.14}$$

In the original algorithm[46], equations (4.11) and (4.14) are intended for implementation. In a hard wired special purpose computer, however, equation (4.14) is found to present difficulties in complement arithmetic. These are overcome by rewriting equation (4.13) as

$$S_{j+1} = x_j^2 - 2(x_{j+1} + 1) + 1$$

$$\text{where} \qquad S_{new} = S_{old} - 2x_{new} - 1 \tag{4.15}$$

Let $[D]_1$ denote the ones-complement of a binary number D. Then using twos-complement arithmetic for subtraction, equation (4.15) becomes

$$S_{new} = S_{old} + [2x_{new}]_1 + 1 - 1 \quad .$$

To summarise, the algorithm for S>N uses

$$\begin{aligned} x_{new} &= x_{old} - 1 \\ S_{new} &= S_{old} + [2x_{new}]_1 \end{aligned} \tag{4.16}$$

Equations (4.11) and (4.16) are implemented in the square-rooting process.

### 4.10.2 Squaring Algorithm

The square root algorithms can be used, with a small modification, to calculate the square of an m-bit binary number M which may be fluctuating. The basic algorithms of equations (4.11) and (4.16) are still used but the choice between the two is now decided according to the states $x<M$ and $x>M$. Thus when $x<M$, equation (4.11) is implemented whereas $x>M$ required use of equation (4.16). The processing is stopped when x=M, at which instant, the S register contains $M^2$ exactly. The x-counter must be cleared before the square-root or the squaring operation is commenced.

### 4.10.3. Computation of Standard Deviation

The square-root algorithms can also be used to calculate $\sigma$. For this purpose, the equation (4.9) is rewritten as

$$\sigma^2 + m_1^2 = m_2^2 \tag{4.17}$$

Initially computation of $m_1^2$ is carried out using the squaring mode of section 4.10.2. On completion, the S register contains $m_1^2$. The entire system is cleared, except the S register and the square-root mode is implemented with $m_2$ being compared with S. Since the S register contained $m_1^2$, its value will increase to $m_1^2 + x_{j+1}^2$ after each step. The square-root process is stopped when $S \geq m_2$. The S register will then contain $m_1^2 + x_{j+1}^2 \geq m_2$ and equation (4.17) shows that the x-register content is $\sigma$ with a maximum error of 1. Again, since the computation of $\sigma$ is a simple counting procedure any readout can be used.

### 4.10.4. Implementation

A block diagram of a 2m-bit system is shown in Fig. (4.10). The square-root mode is used in the following description and only minor modifications are necessary for $\sigma$ computation.

The system comprises two 2m-bit binary registers for N and S, a buffer register for N, a 2m-bit binary comparator to indicate the states $S<N$, $S=N$, $S>N$, a 2m-bit binary adder, 2m true/complementing (T/C)

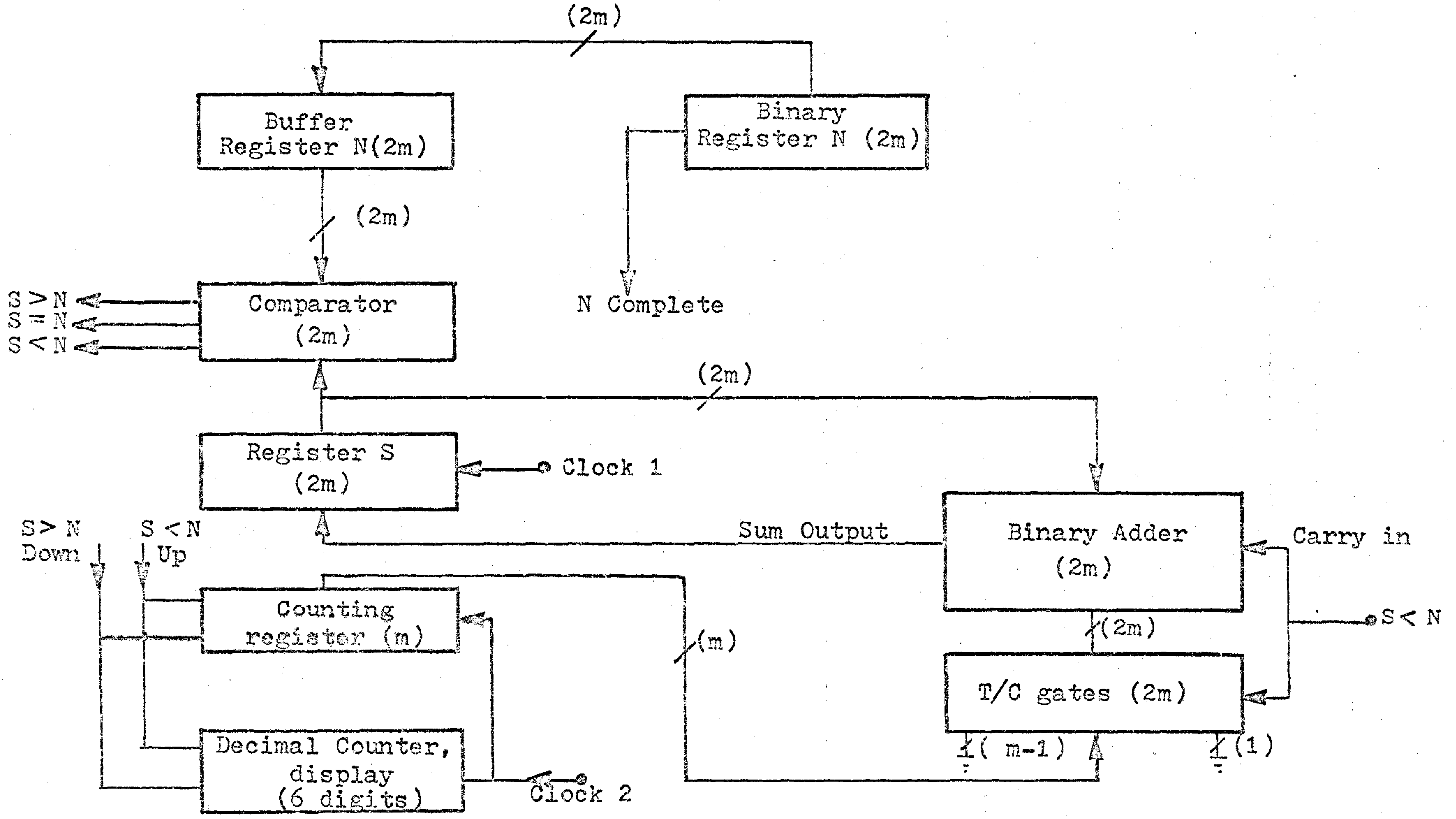


Figure 4.10: Block diagram of the $\sigma$ measuring system.
A 2m-bit number is assumed.
Inclined bars denote number of connecting lines.

gates, and an m-bit synchronous UP/DOWN counter for x and a clock system. In the diagram, a counter for readout of $\sqrt{N}$ in the decimal code is shown. However, if the readout is to be in binary code, then it may be directly obtained from the x counter. Register S is comprised of D type clocked flip-flops. The buffer register is made up of bistable latches.

(i) <u>Calculation of $S_{new}$</u>

A one step parallel adder is used in conjunction with parallel true/complementing (T/C) gates. The output Y of a T/C gate is given by

$$Y = \bar{C}\bar{X} + CX = \overline{\bar{C}X + C\bar{X}}$$

C=0 implies S>N for which $Y=\bar{X}$ and C=1 implies S<N for which Y=X. The inputs to the T/C gates are the various bits of the x counter and are wired with a left shift of one bit as shown in Fig. (4.11). This shift accomplishes multiplication by 2. The l.s.b. input to the T/C gates is permanently in state 0. The control signal is S<N. The T/C gates have 2x as output when S<N and $[2x]_1$ when S>N.

The various inputs to the parallel adder are as follows:

a. The addend is $S_{old}$ from the output of the storage register S.

b. The augend is the output from the T/C gates.

c. The "carry in" is the state of the comparator S<N since from equation (4.11), $C_{in}=1$ only in the up-count mode.

(ii) <u>Clock System</u>

In the down-count mode, i.e., when S>N, $x_{new}$ is used in the algorithm, so that some delay must be incorporated to allow x to change

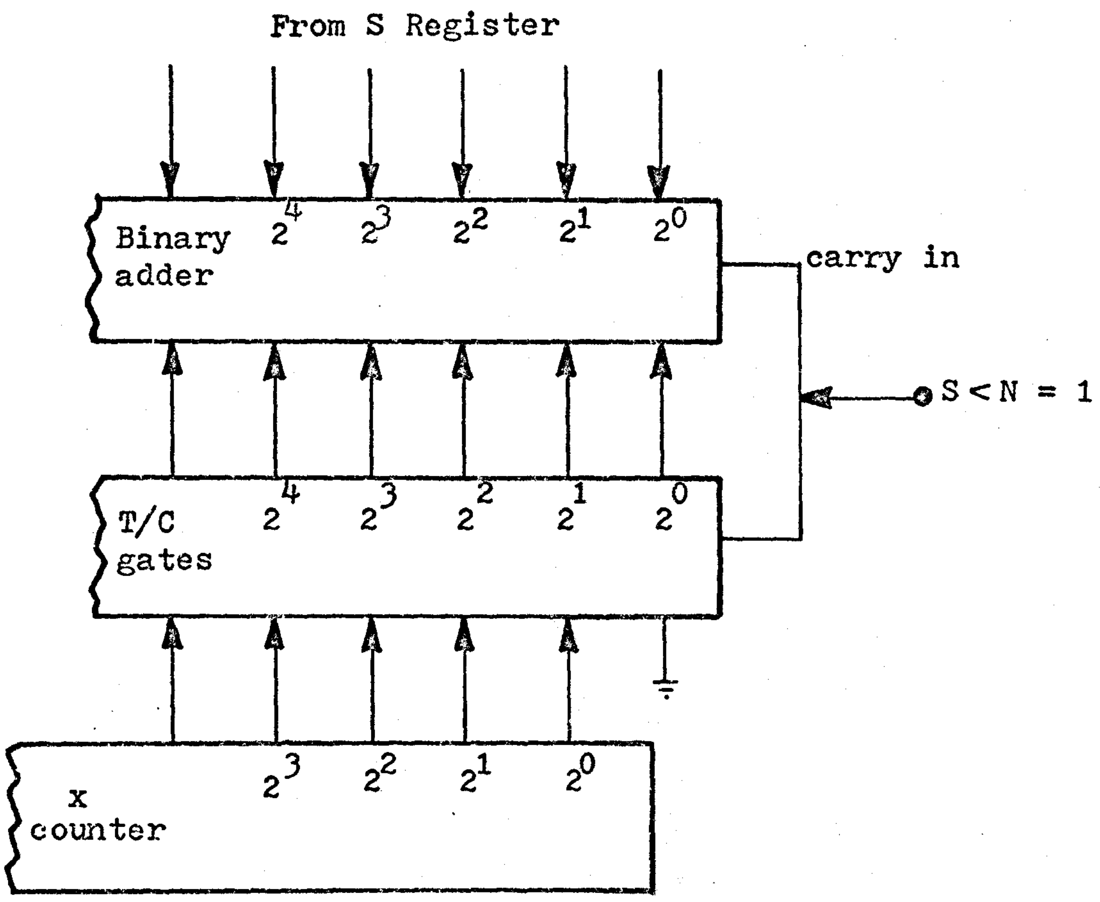


Figure 4.11: Shifted feeds to the T/C gates.

before S. In the up-count mode no delay is necessary since delay in the T/C gates and the adder ensure that $x_{old}$ and not $x_{new}$ is used to form $S_{new}$ when the S register is clocked on the leading edge of the clock pulse. Timing diagrams for these conditions are shown in Fig. (4.12).

Fig. (4.13) shows the clock system. Clock 1 is used to transfer $S_{new}$ to the storage register S while clock 2 is used to clock the x counter. A delay is incorporated in the clock system, which delays the pulse for transferring $S_{new}$ when $S>N$. Computation of $\sqrt{N}$ is stopped by inhibiting the clock, according to any one of the following conditions:

a. $S=N$

b. N is "complete" and transitions from $S<N$ to $S\geq N$ occur at a later time

c. N is "complete" and transitions from $S>N$ to $S\leq N$ occur at a later time.

The inhibiting conditions (b) and (c) prevent the oscillation in the least significant bit of $\sqrt{N}$ when N is complete and when $\sqrt{N}$ is not an integer. Since these conditions require a transition from an initial state to a final state, some memory must obviously be provided. The two D-type clocked flip-flops F1 and F2 are used for this purpose. In the absence of an "N complete" signal the clear inputs of the two flip-flops are held at state 1 so that $\bar{Q}_1=1$ and $\bar{Q}_2=1$. However, if N is fixed, denoted by "N complete"=1, the clear inputs are in state 0 and the flip-flops are free to change state. The "N complete" signal is delayed by 200 ns. to allow the comparator to settle. Assume that when N is complete, $S<N$ (indicated by state 0). At the subsequent transition to $S\geq N$ the clock input to flip-flop F1 changes from 0 to 1, so that $\bar{Q}_1$ changes from 1 to 0 and inhibits the clock. The clock input to F2 changes from 1 to 0 leaving $\bar{Q}_2=1$. The clock remains inhibited until N changes again,

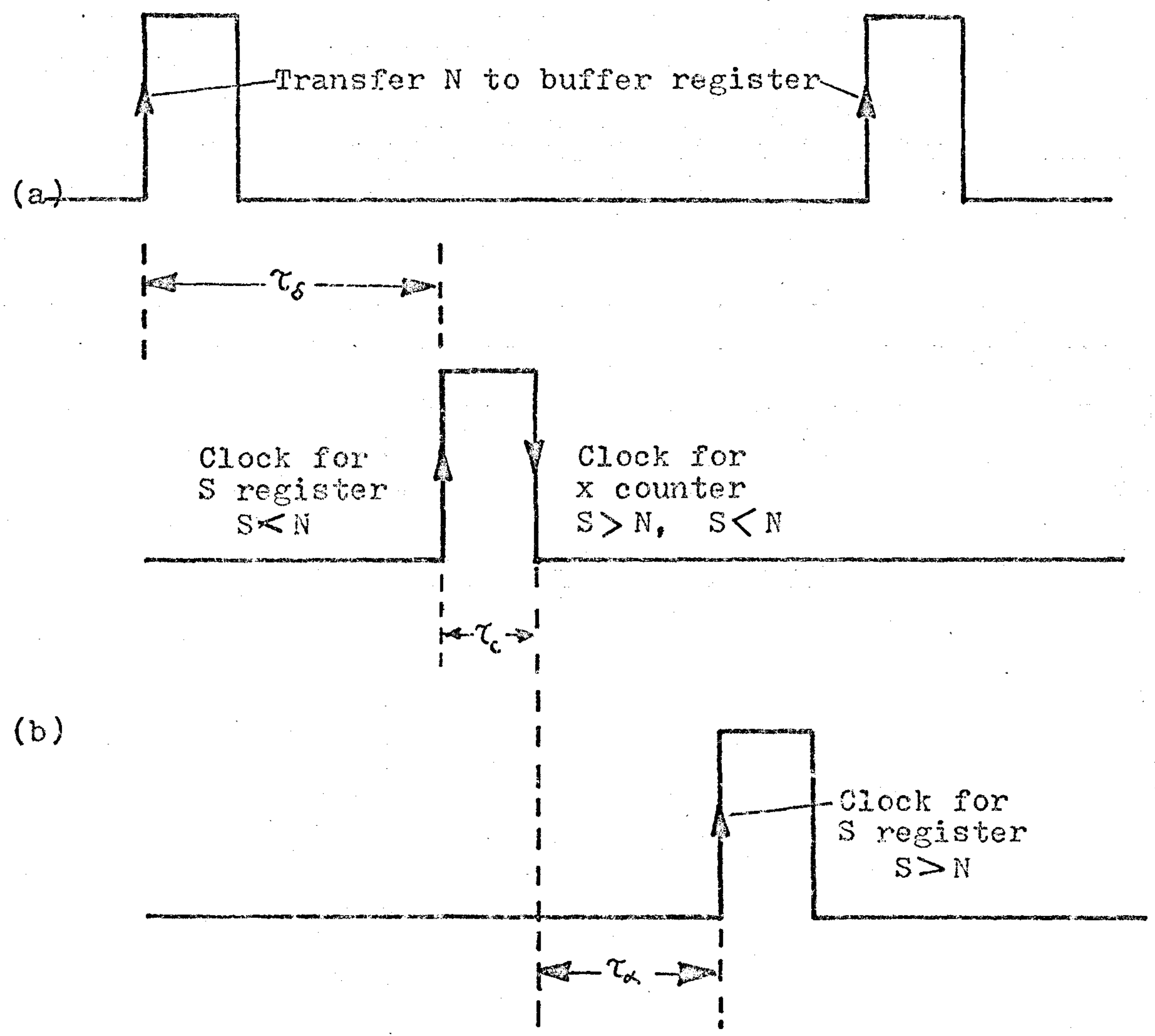

Transfer N to buffer register
(a)
$\tau_\delta$
Clock for S register $S < N$
Clock for x counter $S > N$, $S < N$
$\tau_c$
(b)
Clock for S register $S > N$
$\tau_\alpha$

Figure 4.12: Timing diagram for one step.

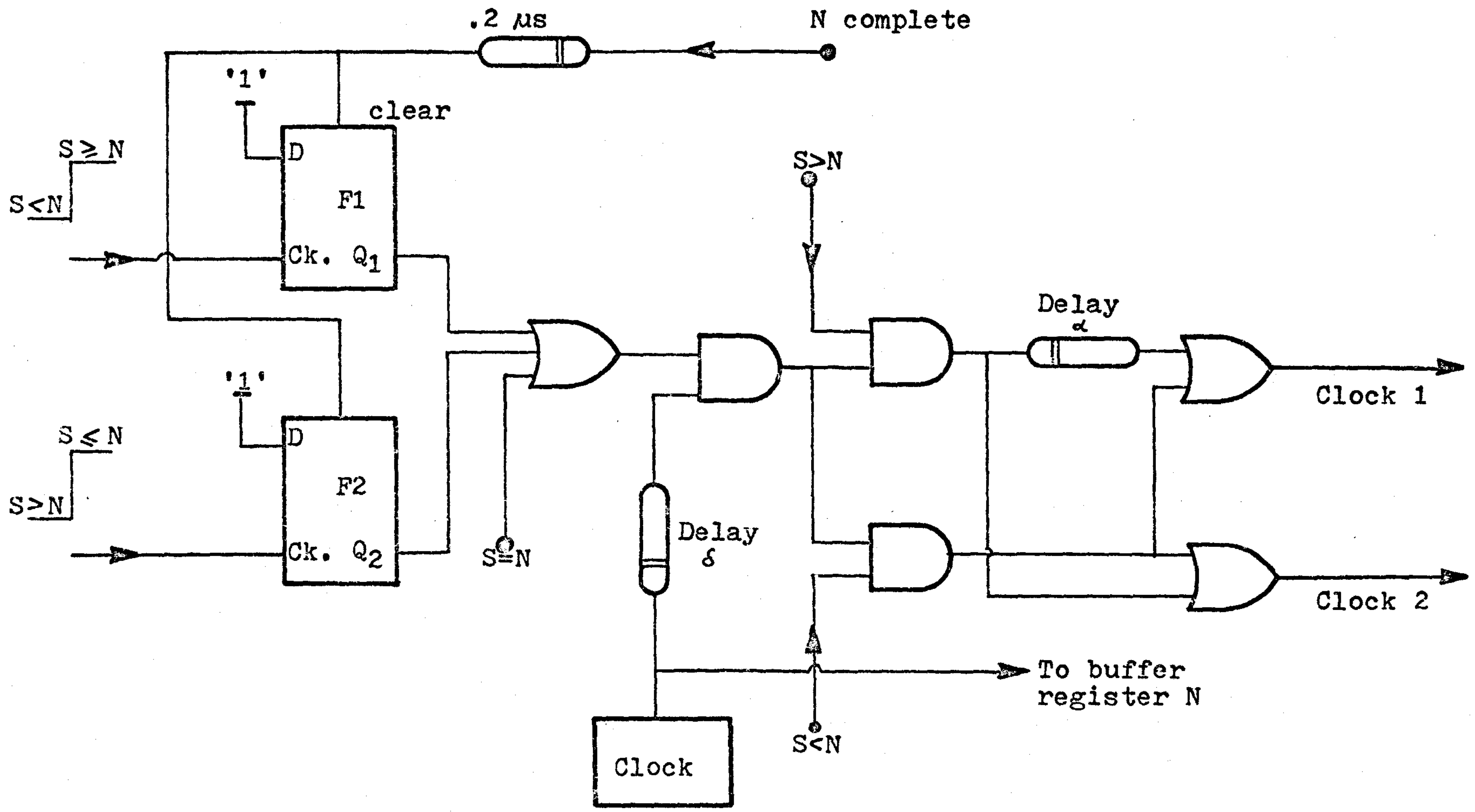


Figure 4.13: Clock System: Clock 1 transfers $S_{new}$ to storage register S.

Clock 2 is used for the UP/DOWN x counter and for the display register.

thereby resetting the flip-flops and releasing the clock.

Referring to Fig. (4.12), the time delay $\tau_\alpha$ must be sufficient for the following to be completed.

1. Synchronous UP/DOWN x counter to settle to the new value of x.
2. The new value of x presented via the T/C gates to the binary adder.
3. Binary addition to be completed and the carry propagated through all stages of the adder.

(iii) Buffer Register for N

The number N may be permitted to change between any steps of the algorithm but not during a step, for if the state of the comparator should change between clocking of the x counter and the S register, the end result will be in error. A convenient way of allowing N to change only at allowed times is to use a buffer register to hold it. The buffer register is controlled by the clock system and a delay $\tau_\delta$ (Fig. (4.12) and Fig. (4.13)) allows the N buffer register and the binary comparator to settle before processing of the algorithms commences. The timing diagram of Fig. (4.12) shows the complete sequence in the processing.

(iv) Modification for Standard Deviation Measurement

The basic circuits of Fig. (4.10) and (4.13) are retained but some additions are needed. Thus initially in the squaring mode, the first moment $m_1$ is transferred into the N register and the comparator inputs are $m_1$ (i.e., N) and x. When computation of $m_1^2$ is complete, all registers except the S register are cleared. The "computation of $m_1^2$ completed" signal is used to load $m_2$ into the N register, and change the comparator inputs to $m_2$ (i.e., N) and S. Thereafter since $S<N$ (i.e., $\sigma\neq0$),

the algorithm of equation (4.11) is automatically implemented. This continues until S=N when the clock is inhibited or when S>N. If the latter obtains, the algorithm changes to that of equation (4.16). Thus if N does not change, the x value will oscillate by ±1. However, if a signal "N complete" is applied then the clock is inhibited when the first transition from S<N to $S \geq N$ or from S>N to $S \leq N$ occurs. The final $\sigma$ value contained in the x counter will be in error by ±1 maximum.

The minimum processing time for one step can be derived from the timing diagram of Fig. (4.12) and is given by

$$T_p = \tau_\delta + \tau_\alpha + \tau_c$$

where $\tau_\delta$ = delay to allow buffer register and comparator to settle after N is transferred to buffer register

$\tau_\alpha$ = delay to allow correct computation of $S_{new}$ in the S>N mode

$\tau_c$ = clock pulse width.

All delay elements in the clock system comprise open collector NAND gates loaded with suitable small capacitors, and are therefore easily implemented.

For the s.p.c. considered, the maximum sample size is $10^6$. Therefore a 40-bit square-root system is required to compute the standard deviation, since $m_1^2$ can have up to 40 bits. The $m_1^2$ computation can run simultaneously with the computation of $m_1$. However, the square-root operation commences only after $m_1^2$ is computed, so that the standard deviation is available shortly after change-over to the square-root mode.

If TTL integrated circuits are used, then the process can be operated at 1 MHz clock-rate which is already available in the master clock system of the s.p.c.

# CHAPTER V

# ERROR ANALYSIS

## 5.1 Introduction

The use of mid-interval values of the quantised signal and the simplifications made in the derivation of the algorithms for the moments, result in an error in the measured values of the moments. Also, the measurements are terminated after a finite time, which causes further statistical fluctuations in the measured values. Additional errors due to level offsets, finite sampling rate etc., can be reduced by improvements in the design of the various units of the s.p.c.

In this chapter, the predominant errors are analysed for several standard input signals. The analysis assumes an exact correspondence between time averages and expectation integrals (using ensemble averaging). The errors are analysed on the basis of statistical independence. This assumption allows each error to be treated individually.

The error equations do not, in general lend themselves to closed-form solutions. They are, therefore, best evaluated by means of a general purpose computer. In these computation procedures, the errors due to truncation are insignificant.

Since the errors depend on the amplitude distribution of the signal and its peak value, it is desirable that waveforms of as wide a range of probability distribution as possible are treated. It is shown that full wave rectified (FWR) signals are suitable for a complete error analysis of the s.p.c.

The errors that arise in the s.p.c. are essentially stochastic in nature, so that if an overall error characteristic is required, then the various error probability distributions must be convolved together. However, this overall error distribution is seldom needed since the worst case errors are sufficient specifications for the s.p.c.

## 5.2 The Error Equation

Let the error due to any one source for the k$^{th}$ moment be $e_{k,z}$ where z represents the source of error. Then $e_{k,z}$ is defined by

$$e_{k,z} = \frac{m^1_{k,z} - m_k}{m_k} \tag{5.1}$$

where $m^1_{k,z}$ is the measured value for a system in which only the z source of error is present and $m_k$ is the actual k$^{th}$ moment.

Since it is usual to define the error in terms of full scale readings, it is tacitly assumed that all the quantisation intervals of the s.p.c. are used in the measurements.

## 5.3 Standard Signals

The errors arising in the s.p.c. are dependent on the signal waveform. Thus in any investigation of the errors, it is necessary to specify the signal. The following criteria are useful in selecting the waveform to be used in the analysis.

(i) All four moments are finite and non-zero.

(ii) The probability density function of the signal can be expressed in closed-form.

(iii) The signals used in the theoretical analysis should be available in the laboratory for experimental verification.

Full-wave rectified (FWR) signals have non-zero and finite four moments, and the s.p.c. has a precision rectifier at its input. Furthermore, the only modification necessary, in the s.p.c., for the measurements on FWR signals is to put the sign control of the odd-order moments counters in a fixed UP-count mode. The signals used for the theoretical analysis are:

1. FWR rectangular waveform or D.C. input.
2. FWR Sine wave.
3. FWR Triangular wave.

Each of these periodic waveforms has a normalized peak voltage $\zeta$ such that the peak lies in the last quantisation interval.

4. For non-periodic signal analysis, bandlimited FWR Gaussian noise of standard deviation $\sigma$ is used.

The four moments for these standard signals are given in Table 5.1. If $P_r$ denotes the probability that the r th level is exceeded and if all levels are used, then the following distributions can be easily derived.

(a) FWR Rectangular Waveform or D.C.

$$\begin{aligned} P_{r+} &= 1 \qquad 0 \le r \le n-1 \\ P_{r-} &= 0 \end{aligned} \tag{5.2}$$

(b) FWR Sine Wave

$$v(t) = \zeta|\cos(\omega t)|$$

and $$P_{r+} = \frac{2}{\pi} \cos^{-1} \frac{r}{n\zeta} \qquad 0 \leq r \leq n-1$$

$$P_{r-} = 0 \tag{5.3}$$

(c) FWR Triangular Wave

$$P_{r+} = 2(1 - \frac{r}{n\zeta})$$

$$P_{r-} = 0 \tag{5.4}$$

In general $$\zeta = \frac{n-1+\varepsilon}{n} \tag{5.5}$$

where $\varepsilon$ is a fraction of the highest level occupied by the input.

(d) Normal Wave or Bandlimited Gaussian Signal

$$P_{r+} = 1 - 2 \int_{o}^{\frac{r}{n\sigma}} \frac{1}{\sqrt{2\pi}} \exp.(-\frac{v^2}{2})dv \tag{5.6}$$

$$P_{r-} = 0$$

A normal waveform in the above case is defined as a periodic signal which has a Gaussian probability density function when sampled at random in time. If the standard deviation $\sigma$ is small, then the normal waveform approaches a unit impulse for each half cycle.

The various errors are now considered individually.

## 5.4 Error due to Algorithm Approximations and Amplitude Quantisation

The general equation for the four moments has been derived using two approximations, viz,

(a) The contribution, due to the voltage in the first interval, is negligible for all four moments.

(b) The weighting number expressions were simplified on the

| F.W.R. Signal Input | First Moment ($m_1$) | Second Moment ($m_2$) | Third Moment ($m_3$) | Fourth Moment ($m_4$) |
|---|---|---|---|---|
| Rectangular Wave or D.C. | $\zeta$ | $\zeta^2$ | $\zeta^3$ | $\zeta^4$ |
| Sine Wave | $\frac{2\zeta}{\pi}$ | $\frac{\zeta^2}{2}$ | $\frac{4}{3\pi}\zeta^3$ | $\frac{3}{8}\zeta^4$ |
| Triangular Wave | $\frac{\zeta}{2}$ | $\frac{\zeta^2}{3}$ | $\frac{\zeta^3}{4}$ | $\frac{\zeta^4}{5}$ |
| Normal Wave or Bandlimited Gaussian Noise | $\sqrt{\frac{2}{\pi}}\,\sigma$ | $\sigma^2$ | $\sqrt{\frac{2}{\pi}}\,2\sigma^3$ | $3\sigma^4$ |

Note: $\zeta = \frac{n-1+\varepsilon}{n}$ where $0 \leq \varepsilon \leq 1$

and $\sigma$ is standard deviation of Normal Wave or Gaussian Noise.

Table 5.1.: The four moments of the FWR signals used in the error analysis.

basis that all available levels are used.

Both these assumptions cause an 'approximation error' in all four moments, which is independent of the s.p.c. system.

In the s.p.c., samples are classified within n intervals (i.e., the discrete value expectation integral is used) on each side of the zero level, and a mid-interval value is assigned to each sample. In the computation of moments the $k^{th}$ power of each sample value is used. Since n is finite, a quantisation error occurs in the computed moments. For the purpose of analysis, the approximation and quantisation errors are grouped together and referred to as the system quantisation error $e_{k,n}$.

For periodic waveforms, the peak of the input signal may lie anywhere within the $n^{th}$ interval. Since this peak value influences the probability distribution values as seen from section 5.3, it is to be expected that the system quantisation error, $e_{k,n}$, is also a function of the position of the peak value, within the last interval.

For bandlimited Gaussian noise or for the normal wave, the mechanism of the occurence of system quantisation error is somewhat different, since the voltage is not confined within the n intervals. Thus, in addition to the approximations and quantisation errors mentioned earlier, a level limit error arises. This is due to the fact that samples beyond the $n^{th}$ level are all grouped together and classified into the $n^{th}$ interval. For small standard deviation $\sigma$, this limit error is negligible. In such cases the effect of using n intervals can also be obtained by the Sheppard correction formulae[19]. These correction formulae, however, do not account for errors due to algorithm approximations.

The system quantisation error analysis in most cases, yields intractable expressions except for d.c. inputs. Simplification[16] is possible in some cases but yields very little useful information regarding the general behaviour of the error. Instead two computer programs are used to obtain the system quantisation characteristics. The error definition uses the calculated $m^1_{k,z}$ as used in the s.p.c., i.e.,

$$m^1_{k,z} = \frac{1}{n^k} \sum_{r=1}^{n-1} k \cdot r^{k-1} \; P_r \tag{5.7}$$

The first program used for periodic signals requires $P_r$ and the actual moments to be specified. The second program is for Gaussian signals where $P_r$, $\sigma$ and the actual moments need to be specified. In both programs n, the peak position $(n-1 + \varepsilon)$ or $\sigma$ are varied. The flow-charts for these two programs are given in Appendix B and C.

## 5.4.1. Results for $e_{k,n}$.

The computer analysis results for $e_{k,n}$ are divided into two categories, viz

(a) For a fixed n, the variation of the error due to the peak value changing from $\frac{n-1}{n}$ to 1 for periodic signals and due to variation of standard deviation $\sigma$ for Gaussian signals, is investigated.

(b) Effect of variation of n for a given peak position for periodic signals and a given $\sigma$ for Gaussian noise, is also investigated. This analysis establishes a design criterion for the special purpose computer.

The results of the two analyses for the standard signals are presented in Figures (5.1) to (5.8).

The dependence of the system quantisation errors on the position of the peak values is seen from the graphs of Figures (5.1), (5.2), (5.3) and (5.4). Results indicate that it is possible to reduce $e_{k,n}$ to zero by properly locating the peak value position of the signal. However, such a technique requires a knowledge of the input and would also invariably involve an iterative scaling procedure. Hence in practice, it is likely that the assumption of the peak being situated anywhere within the last interval with uniform probability distribution will be more valid. To obtain the statistical distribution of the error, Monte Carlo methods[17] of simulation can be used.

For all four FWR signals the maximum error in the first moment varies inversely with n. The graphs of Figures (5.5), (5.6), (5.7) and (5.8) also show that it is necessary to use n=256 i.e., 8-bit quantisation in order to keep the system quantisation error for first moments, below 1%.

The maximum errors in all the other moments vary as $\frac{1}{n^u}$ where u is a function of the periodic signal. Thus for FWR Sine Wave u=1.5, for FWR Triangular Wave u=2 and for D.C. inputs u=1. The errors are largest for D.C. inputs as would be expected.

In the case of FWR Gaussian signals, a normalised standard deviation of 0.25 is used so that level limit errors are negligible. The system quantisation error is large for small values of n but begins to settle to a steady value for all higher moments, indicating that above a certain value of n, very little reduction in $e_{k,n}$ is achieved.

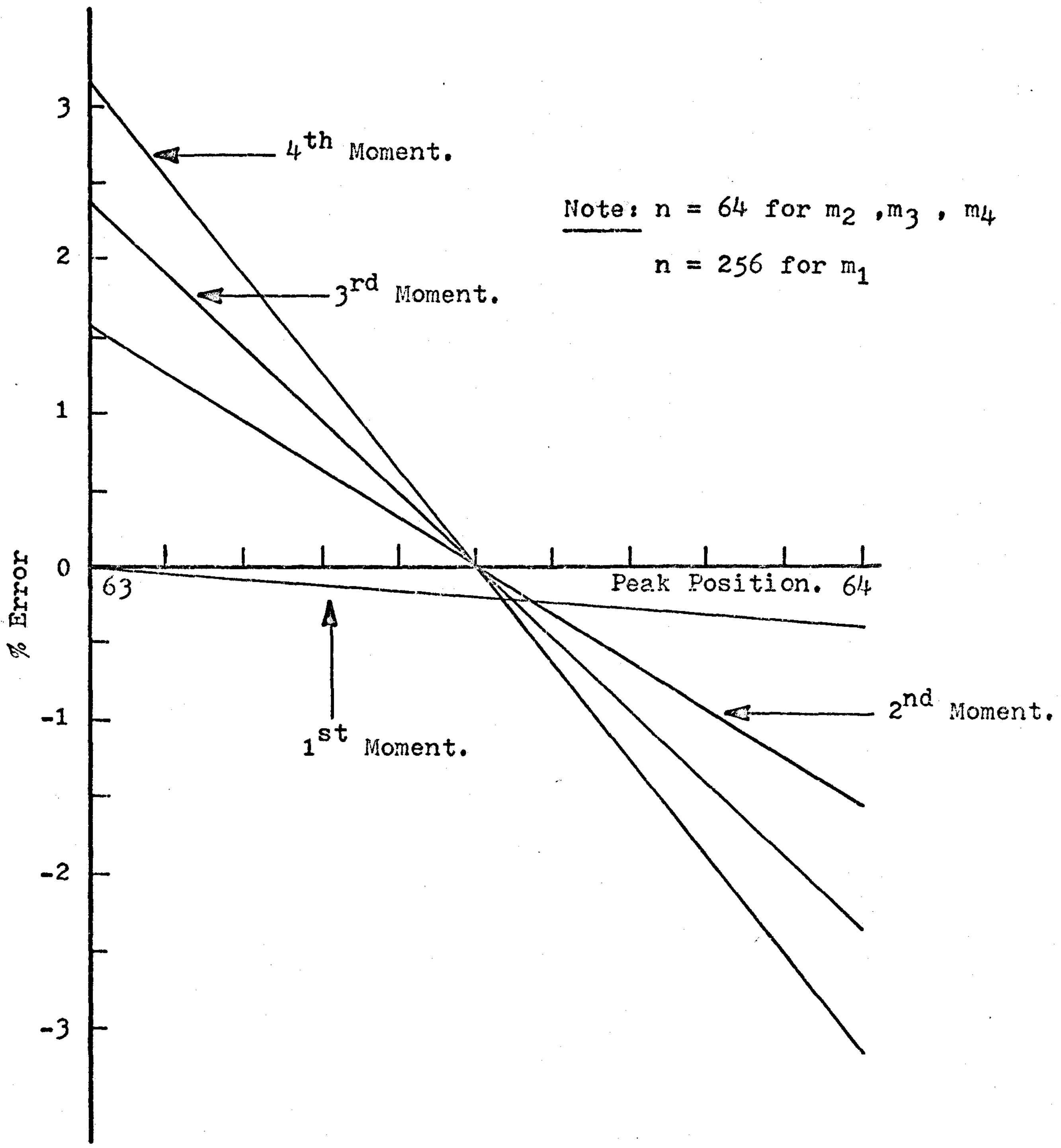


Figure 5.1: Quantisation Errors for FWR Rectangular Wave.

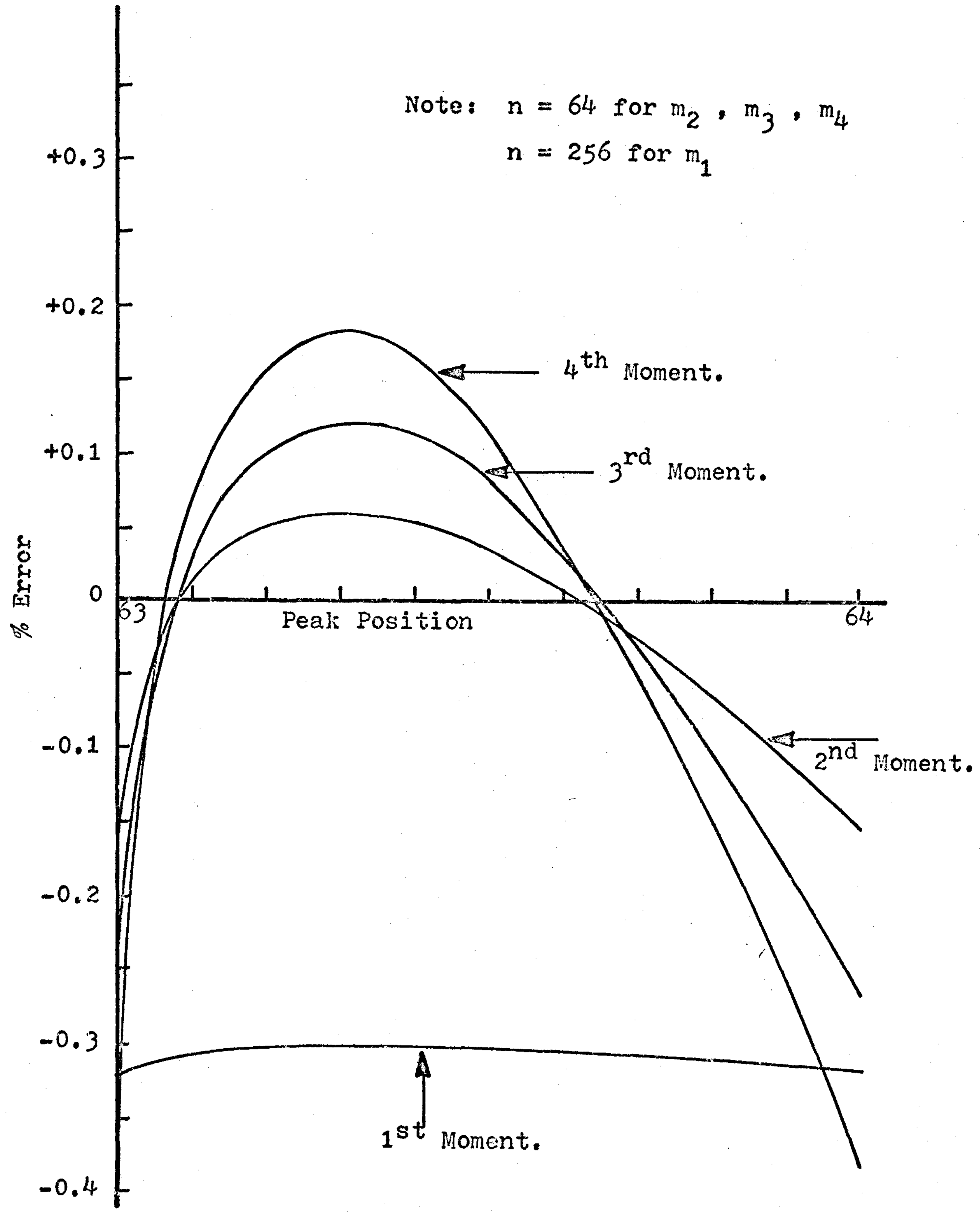


Figure 5.2: Quantisation Errors for FWR Sine Wave.

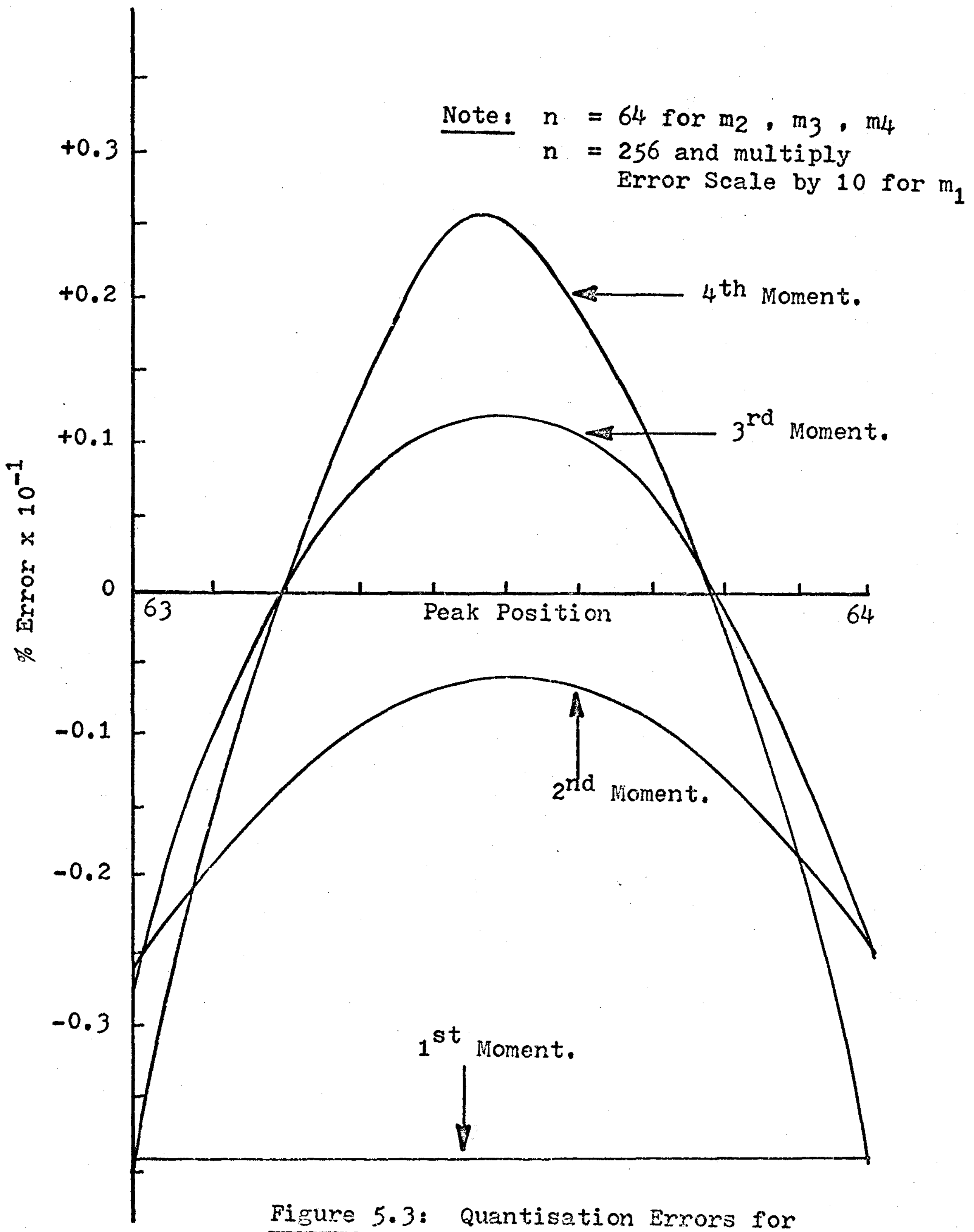


Figure 5.3: Quantisation Errors for FWR Triangular Wave.

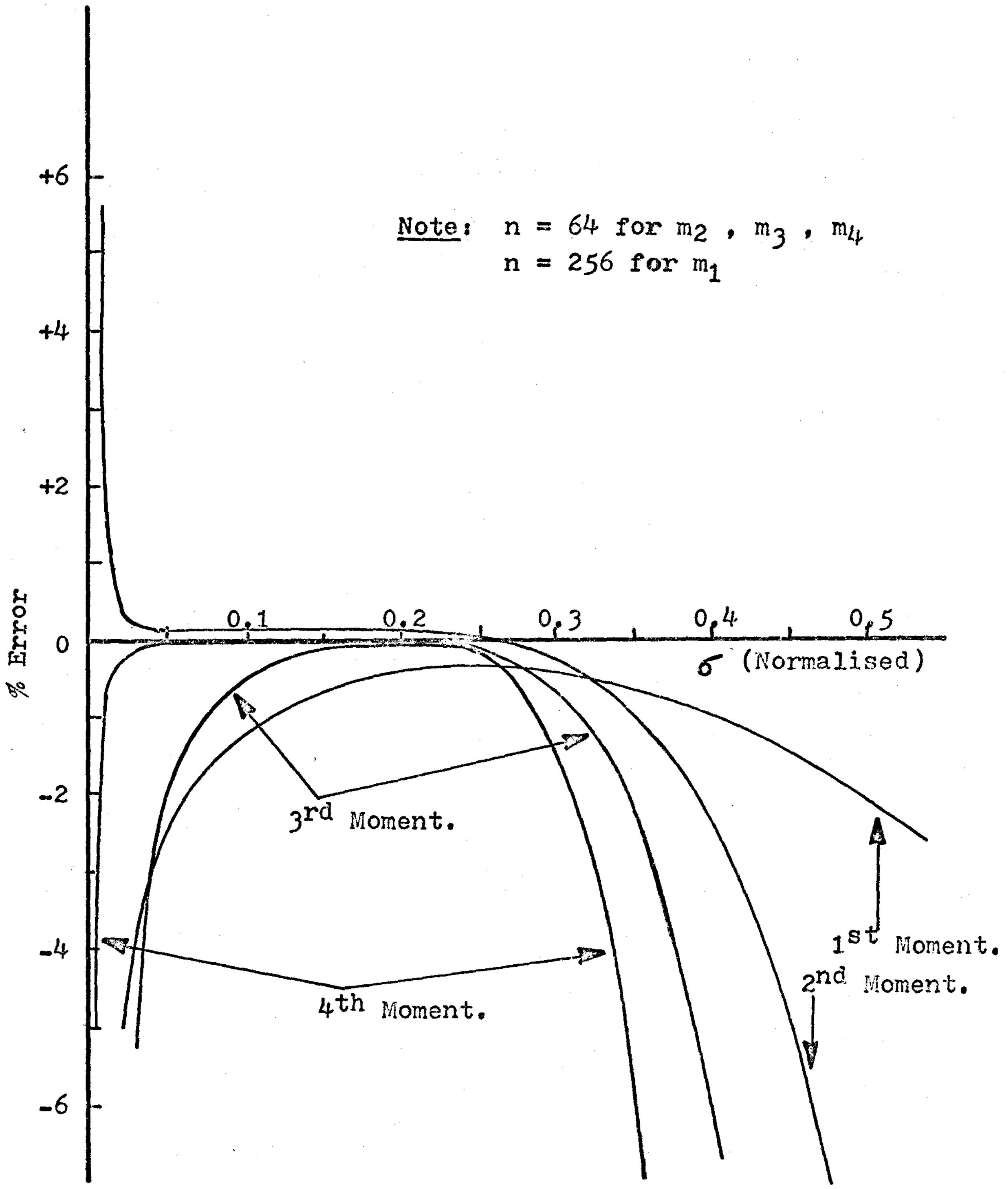


Figure 5.4: Quantisation Errors for FWR Gaussian Noise.

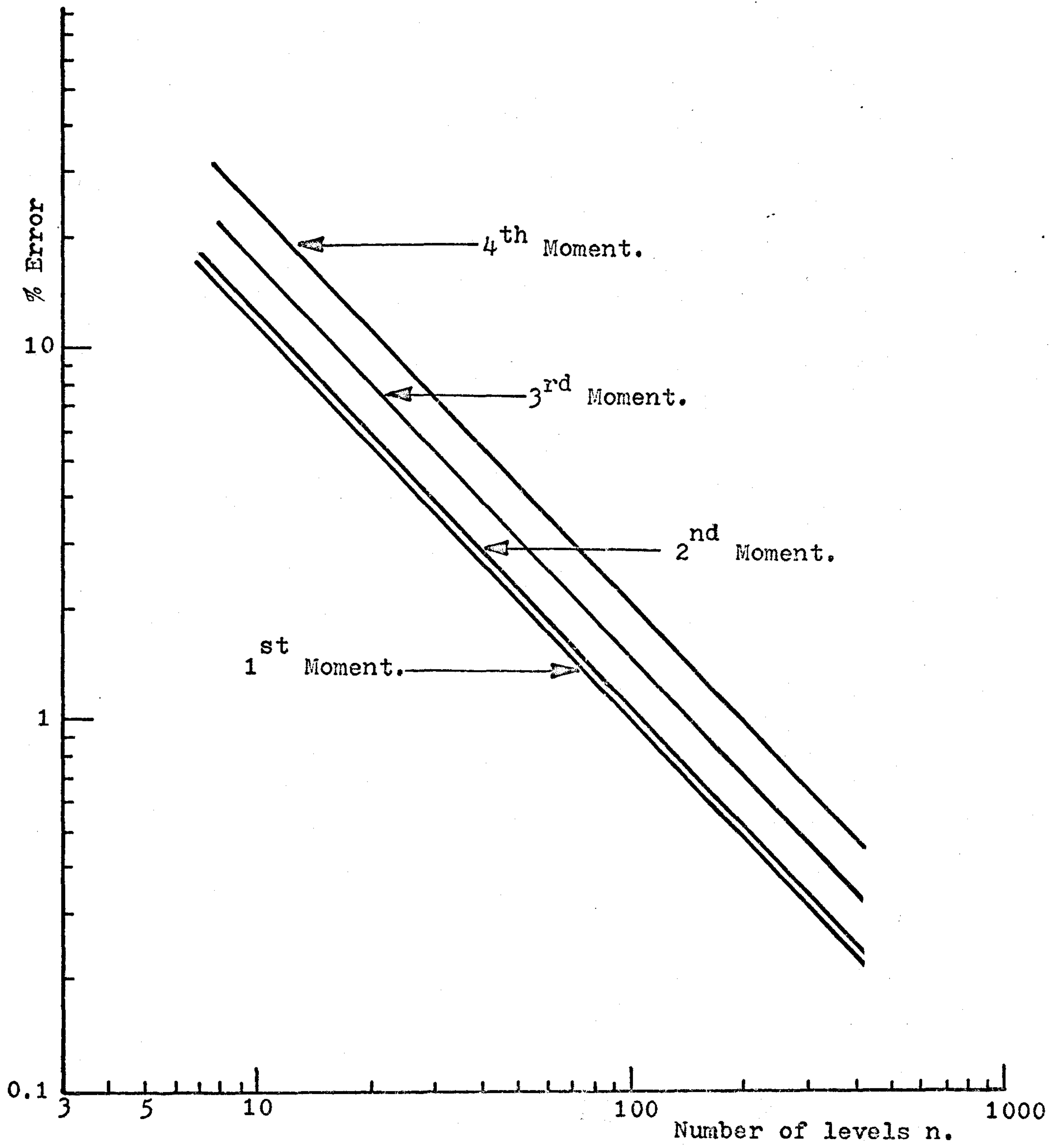


Figure 5.5: Quantisation Errors for FWR Rectangular Wave of 10 V. amplitude.

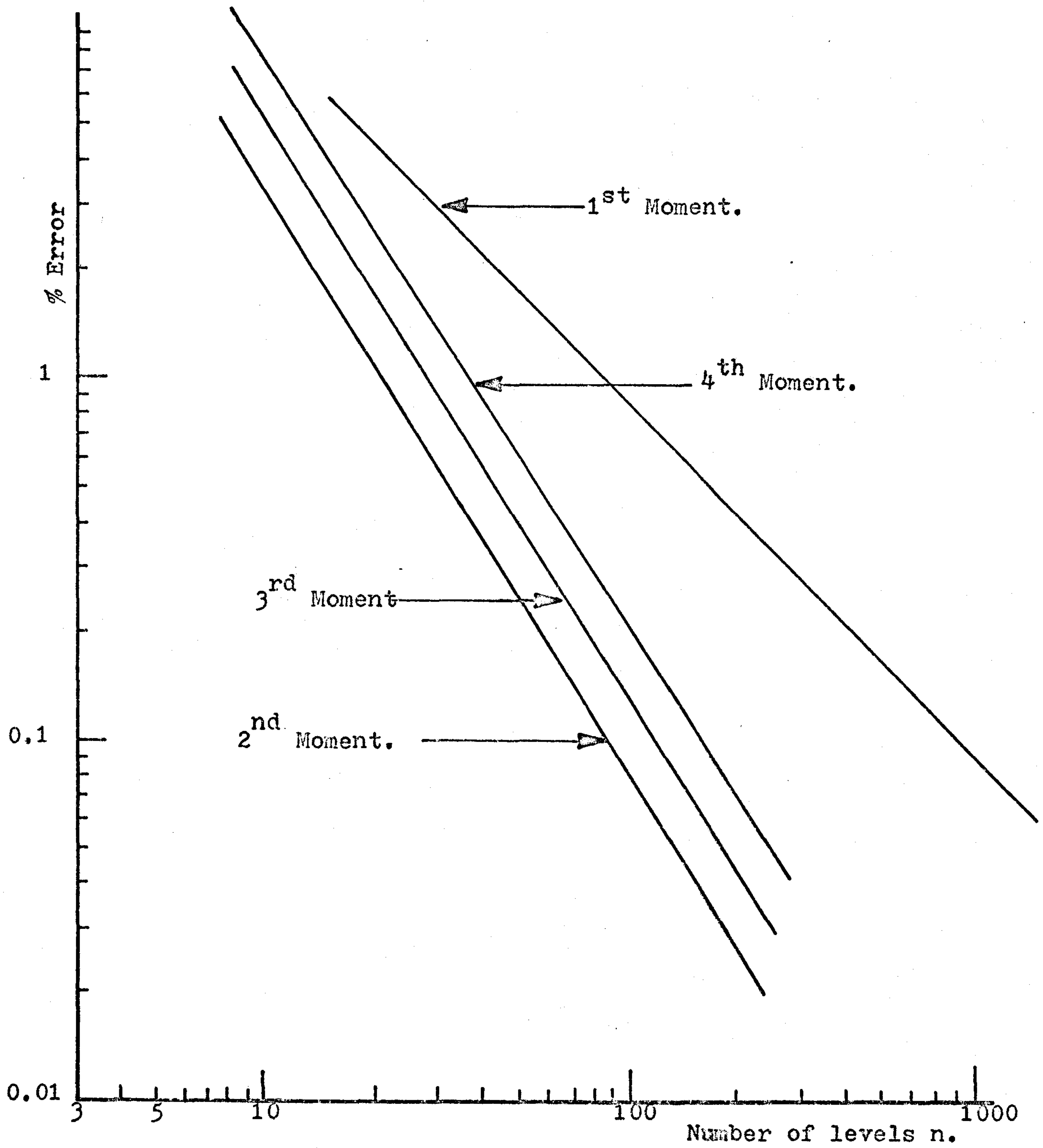


Figure 5.6: Quantisation Errors for FWR Sine Wave of 10 V. Peak.

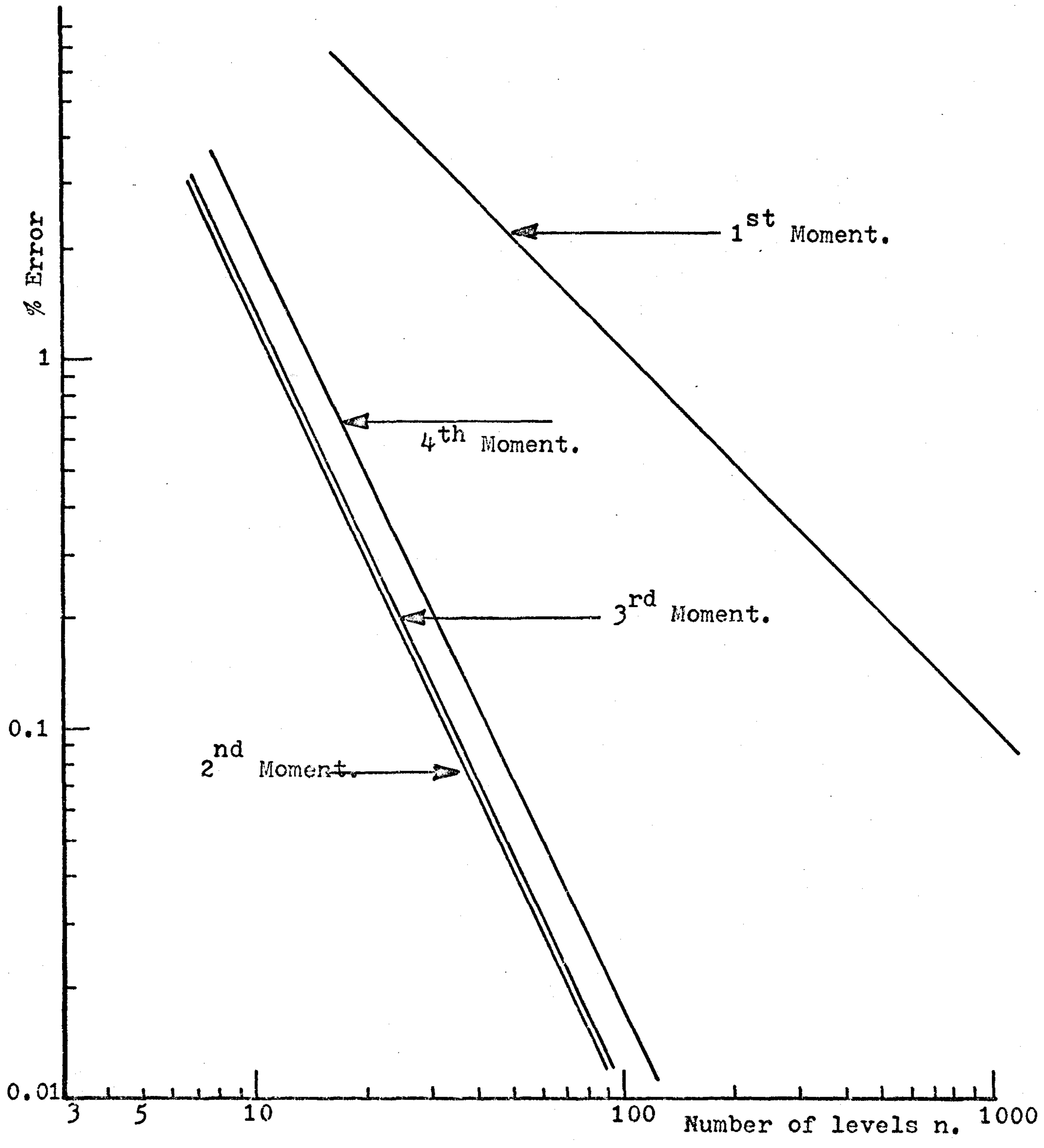


Figure 5.7: Quantisation Errors for FWR Triangular Wave of 10 V. Peak.

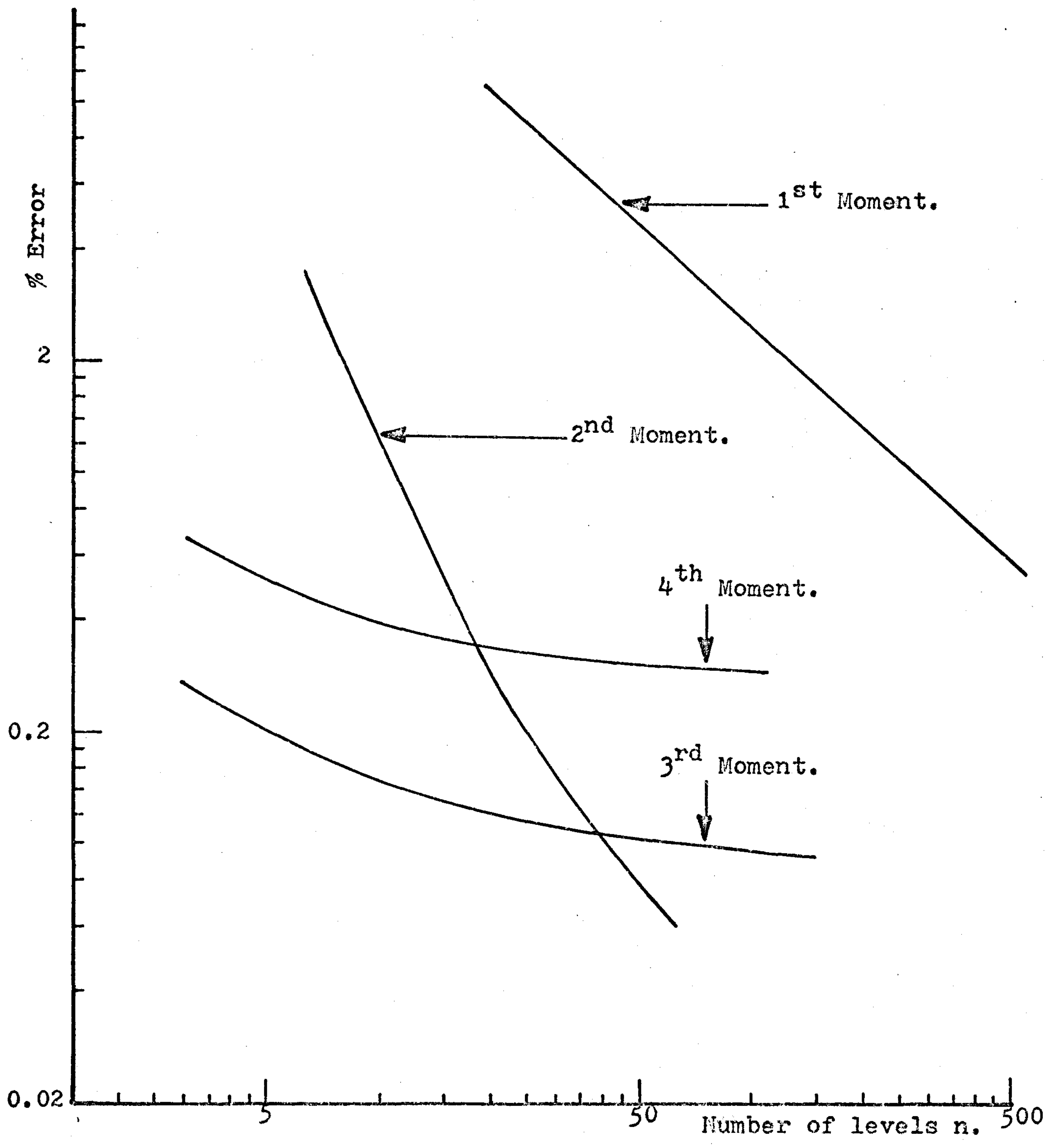


Figure 5.8: Quantisation Errors for FWR Gaussian Noise of $\sigma = 0.25$.

### 5.4.2. Level Limit Errors

(a) Periodic Waveforms:

If the peak value of the input exceeds the n$^{th}$ level voltage, then a negative error occurs. A typical example of this error is shown for FWR Sine Wave, in Figure 5.9. This error is easily eliminated in practice by scaling down the input.

(b) Gaussian Signals:

The level limit errors are most serious for these signals since there is always a finite probability that at some instant the highest level available will be exceeded. For large $\sigma$, this probability increases rapidly and the negative level limit error swamps the system quantisation error as shown in Fig. (5.4). It is seen that if the standard deviation is within 1.1 v to 2.9 v, the total system quantisation error (including the level limit error) is within 1% for all moments.

## 5.5 System Errors

The system quantisation error, as mentioned earlier, is independent of the method adopted in the actual computation. System errors depend on the system design and on the components used in the processing. The main errors in this category are

(a) Level offset errors

(b) Finite sampling time errors

(c) Finite sampling-rate errors

(d) Aperture-time errors.

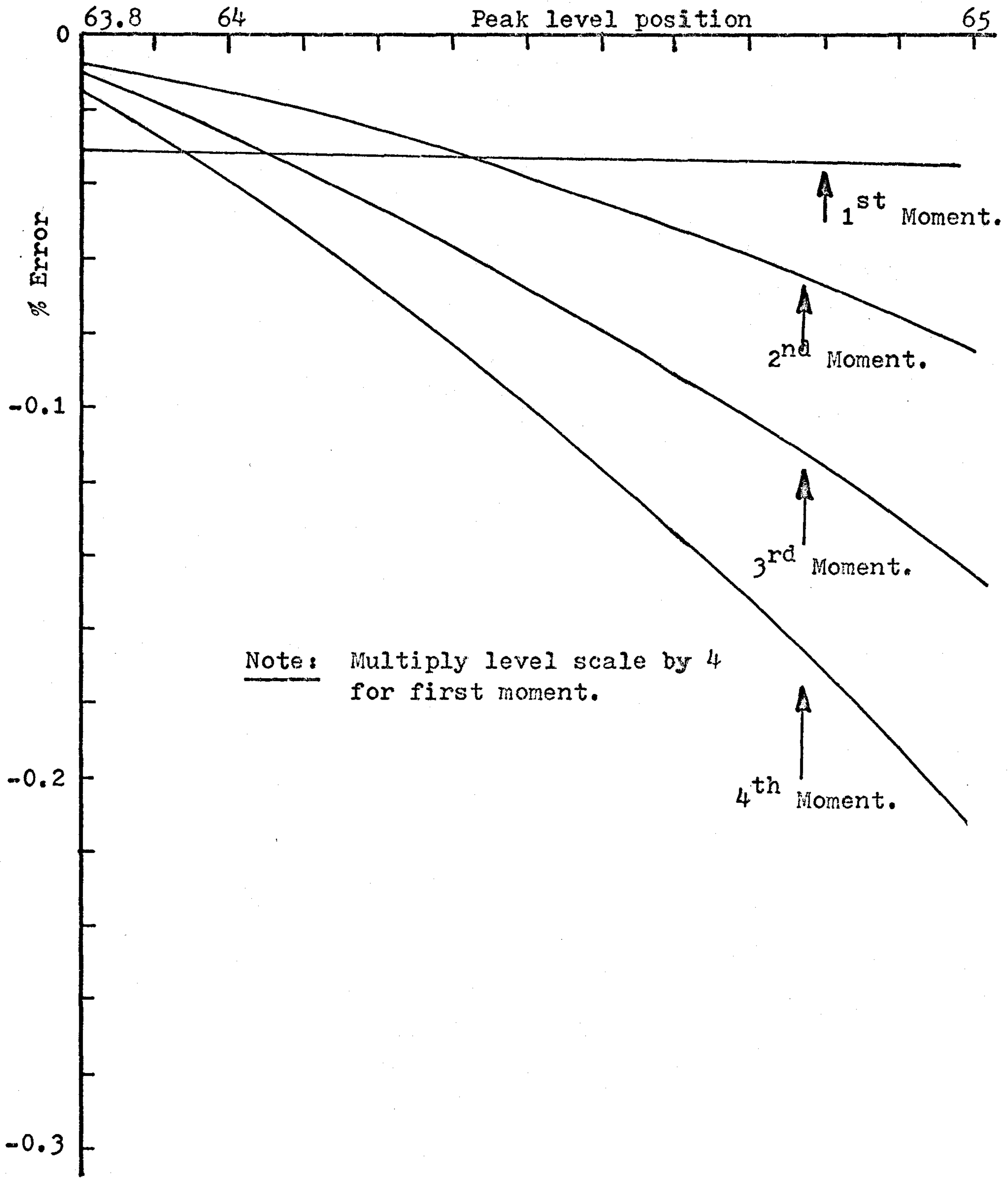


Figure 5.9: Errors for FWR Sine Wave of Peak Exceeding 64th level.

Errors due to finite sampling time and finite sampling rate are regarded for the purpose of analysis as system errors since in practice the upper limit in measurement time is fixed by the size of the $C_o$ register, while the sampling rate is limited by the a.d. converter selected in the design of the s.p.c.

### 5.5.1. Level Offset Errors

The offset at any given level can be due to inaccuracies in the rectifier and due to the a.d. converter resolution capabilities. Thus let the total offset at the $r^{th}$ level be $\delta V_r$. An incorrectly weighted sample occurs for all moments if the sample falls in the range $V_r$ to $V_r + \delta V_r$ where $V_r$ is the voltage at the $r^{th}$ level. The error in the weighting number for the $k^{th}$ moment, for a positive $\delta V_r$ is

$$\delta W_{k,r} = -W_{k,r} + W_{k,r-1}$$

$$= - \sum_{q=1}^{r} k\, q^{k-1} + \sum_{q=1}^{r-1} k\, q^{k-1}$$

$$\therefore \quad \delta W_{k,r} = -k\, r^{k-1} \tag{5.8}$$

Thus all the measured values of the moments will be lower than the actual values when $\delta V_r$ is positive. The total error depends on the number of samples occurring in the level interval $\delta V_r$. Obviously such offsets can occur at all levels. For signals which are symmetrical about the zero level these offset errors tend to cancel out since both $m_{k+}$ and $m_{k-}$ tend to be lower than the actual values for a positive offset. Similarly, in the one cycle mode for periodic signals, the offset error is small since the probability of a sample occurring in the offset interval is very small except at high sampling rates. The level offset errors are therefore, important only in the fixed time mode of operation of the s.p.c.

The error due to level offsets depends on the number of times a level is crossed and the time spent in the offset interval at each crossing. The offset can also be regarded as causing an error in the value of $P_r$, the probability that the r^th level is exceeded. For periodic signals it is relatively easy to obtain the number of crossings whereas for Gaussian signals, the error is best analysed by considering $P_r$.

(a) FWR Periodic Signals

A level is crossed twice for each half cycle. Therefore, for a total measurement time of T sec., the number of crossings of any level is 4 $f_i$ T where $f_i$ is the frequency of the input. If the time spent by the signal, in the offset interval $\delta V_r$ is $t_r$ then the expected number of samples within this interval, for a sampling rate $1/\tau$ is $\frac{t_r}{\tau}$. The total expected number of samples in the interval $\delta V_r$ is given by

$$N_r = 4t_r\ f_i\ C_o \tag{5.9}$$

Each of the $N_r$ samples is incorrectly weighted, the weighting error being given by Equation (5.8). If the slope of the waveform at the r^th level is $V_r^1$ then assuming $\delta V_r$ is small

$$t_r = \frac{\delta V_r}{V_r^1} \tag{5.10}$$

Hence the error in the k^th moment for periodic signals is

$$e_{k,\ell} = \frac{1}{n^k C_o m_k} \sum_{r=1}^{n-1} 4C_o f_i \frac{\delta V_r}{V_r^1} k r^{k-1} \tag{5.11}$$

$$\therefore \quad e_{k,\ell} = \frac{1}{n^k m_k} \sum_{r=1}^{n-1} 4f_i \frac{\delta V_r}{V_r^1} k r^{k-1} \tag{5.12}$$

Since $V_r^1$ involves the frequency, $f_i$, it is easy to show that $e_{k,\ell}$ is independent of $f_i$ the signal frequency.

(b) FWR Gaussian Signals

The expression for the expected number of crossings developed by S.O. Rice[48] can be used; however, difficulty arises in the evaluation of the slope at a level crossing. In such cases, therefore, the probability analysis is more convenient. Thus the error $\delta P_r$ in $P_r$ due to a level offset $\delta V_r$ is given by

$$\delta P_r = \frac{2\,\delta V_r}{\sqrt{2\pi}\,\sigma} \exp.\{\frac{-r^2}{2n^2\sigma^2}\} \tag{5.13}$$

This is derived from the probability density function for FWR Gaussian Signals.

The error in the kth moment for FWR Gaussian Signals is therefore, given by

$$e_{k,\ell} = \frac{1}{n^k m_k} \sum_{r=1}^{n-1} \frac{2\delta V_r}{\sqrt{2\pi}\,\sigma} \cdot [\exp.\{\frac{-r^2}{2n^2\sigma^2}\}]\cdot k\, r^{k-1} \tag{5.14}$$

(c) Results for $e_{k,\ell}$.

The error expressions (5.12) and (5.14) are best evaluated with a general computer program. The flow chart of the procedure is given in Appendix D. The analysis assumes that the level offsets are equal for all levels; however if the actual offsets are known then their values can be used in the analysis, the program modification required, being a very trivial one. The offset $\delta V_r$ and the number of levels n are considered as the variables in the analysis. For a fixed number of quantisation levels n, the error $e_{k,\ell}$ varies linearly with $\delta V_r$. Thus

$$e_{k,\ell} = k_\ell \cdot \delta V_r \tag{5.15}$$

where $\delta V_r$ is the level offset in millivolts. The constant of proportionality for various waveforms are given in Table 5.2.

Errors due to level offsets do not occur for d.c. or rectangular waveforms since only one level is occupied at any time. However, the offset will affect the highest level exceeded information and thereby affect the system quantisation errors, if the peak lies in the offset region.

The level offset errors increase only slightly as n is increased as shown in Fig. 5.10. However, a large n implies a smaller quantisation interval in which case there is every possibility that $\delta V_r$ will also increase. Therefore, little improvement can be achieved by using a higher n. For this reason, n=64 has been chosen in the s.p.c. designed.

## 5.6 Errors due to Finite Time of Measurement

Statistical analysis using the s.p.c. assumes a finite time of averaging. However, time averaging without errors, requires a time of integration which tends to infinity. In practice, measurements are always terminated after a finite time. If the input is periodic then this error can be regarded as arising from the use of a non-integral

| FWR Waveform | $m_1$ | $m_2$ | $m_3$ | $m_4$ |
|---|---|---|---|---|
| Sine | $14.3 \times 10^{-3}$ | $22.3 \times 10^{-3}$ | $30.4 \times 10^{-3}$ | $37.2 \times 10^{-3}$ |
| Triangular | $19.7 \times 10^{-3}$ | $29.5 \times 10^{-3}$ | $39.1 \times 10^{-3}$ | $48.4 \times 10^{-3}$ |
| Gaussian Noise $\sigma=0.25$ | $4\ 9 \times 10^{-3}$ | $6\ 3 \times 10^{-3}$ | $7\ 6 \times 10^{-3}$ | $8\ 1 \times 10^{-3}$ |

Table 5.2.: Constants of proportionality for equation (5.15) to calculate $e_{k,\ell}$.

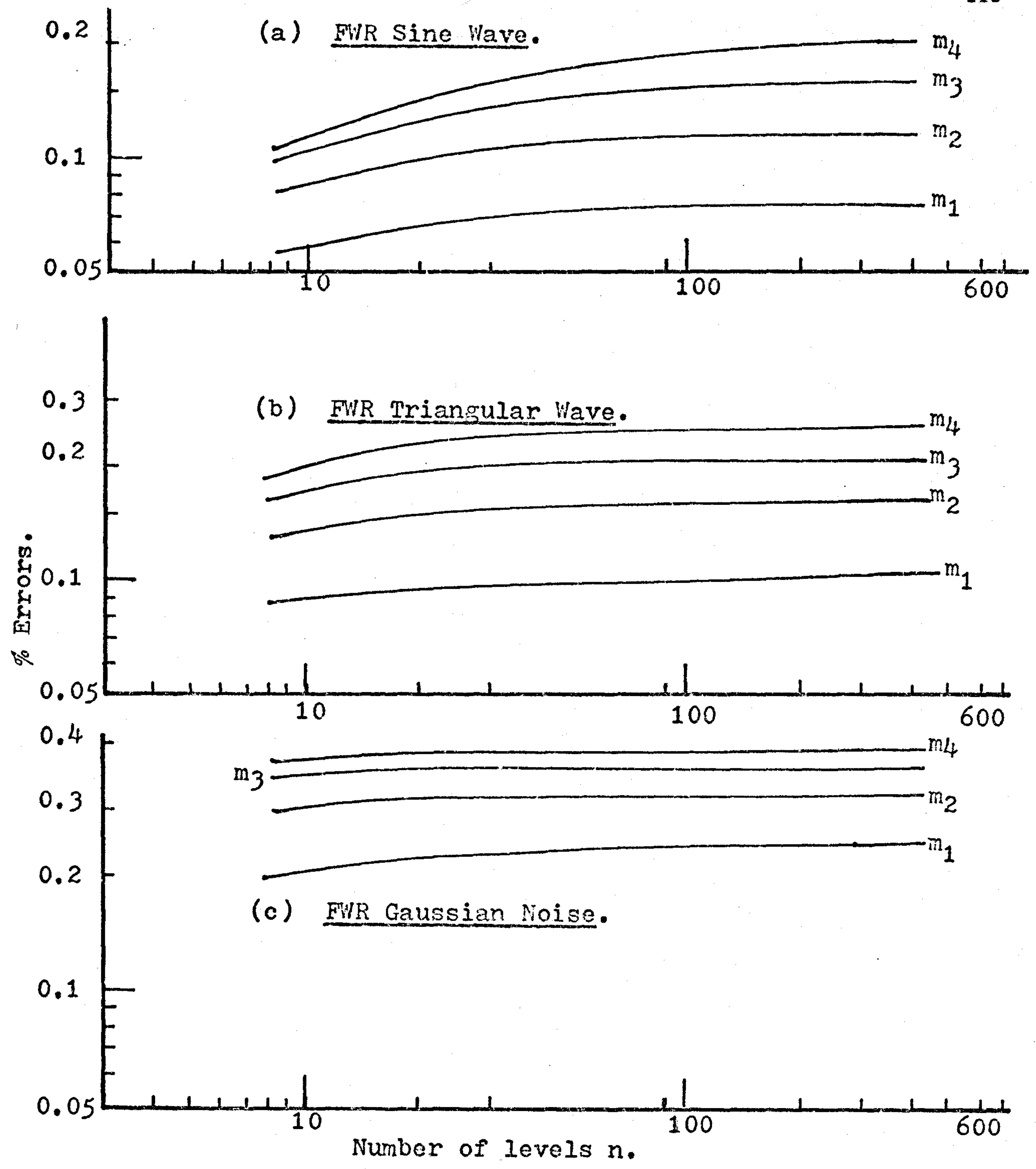


Figure 5.10: Level offset errors for the three standard signals for a - 5 mV. offset at all levels.

number of cycles in the measurement process. In general cases, the effect of finite time is to cause a random error which has a near Normal statistical distribution and whose variance[27] is given by

$$\mathrm{Var}(e_{k,T}) = \frac{2}{T} \int_0^T (1 - \frac{t}{T})(R_{mm}(t) - \bar{v}_k^2)dt \tag{5.16}$$

where $R_{mm}(t)$ is the autocorrelation function of $(v^k)$ corresponding to the k^th^ moment of v and $\bar{v}_k$ is the mean value of $v^k$. If $R_{mm}(t)$ is known then the variance can be calculated and a confidence limit can then be specified for $e_{k,T}$. However, difficulties usually arise in calculating $R_{mm}(t)$, unless suitable approximations can be made[27]. Rice[47], on the other hand, has suggested a direct evaluation of the variance. Again, the same difficulty as in Eqn. (5.16) is experienced.

A simpler analysis can be made using the equivalence between time averages and ensemble averages and then applying the results of large sampling theory. This approach has been adopted in the following analysis (section 5.6.2.).

## 5.6.1. Error $e_{k,T}$ for FWR Periodic Signals

The $k^{th}$ time averaged moment for a finite time T is given by

$$m_{k,T} = \frac{1}{T} \int_0^T v^k \, dt \tag{5.17}$$

and the actual $k^{th}$ moment is

$$m_k = \lim_{T\to\infty}[m_{k,T}] \tag{5.18}$$

Let the measurement process run for $(q+\alpha)$ periods as shown in Fig. (5.11) where $\alpha = \eta - \phi$. The starting point $\phi$ and stopping at $\eta$

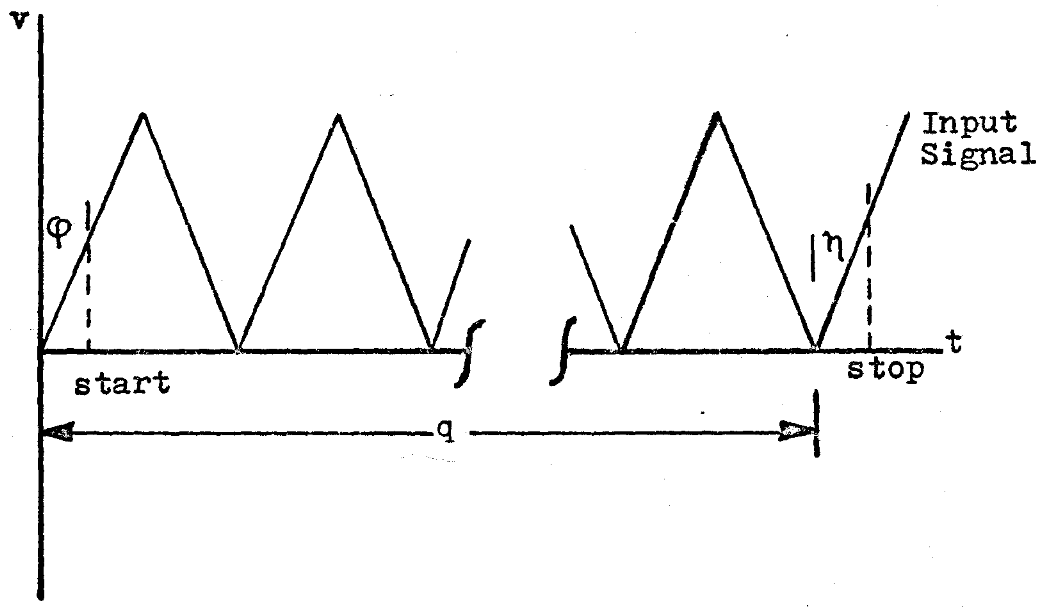


Figure 5.11: Finite time of measurement for a periodic signal.

are completely random. For a periodic signal

$$v = g(\theta)$$

$$\therefore \quad m_{k,T} = \frac{1}{2\pi(q+\alpha)} \int_0^{2\pi(q+\alpha)} g^k(\theta)\, d\theta \tag{5.19}$$

where $0 \leq \alpha \leq 1$

and

$$m_k = \frac{1}{2\pi q} \int_0^{2\pi q} g^k(\theta)\, d\theta\ . \tag{5.20}$$

The error $e_{k,T}$ due to finite time is

$$e_{k,T} = \frac{m_{k,T} - m_k}{m_k}$$

$$= \frac{m_{k,T}}{m_k} - 1 \tag{5.21}$$

Substituting for $m_{k,T}$, one obtains the general expression for $e_{k,T}$, viz

$$e_{k,T} = \frac{\int_0^{2\pi\alpha} g^k(\theta)\, d\theta + 2\pi q\, m_k - 2\pi(q+\alpha) m_k}{m_k \cdot 2\pi(q+\alpha)} \tag{5.22}$$

If $q >> \alpha$, the equation (5.22) reduces to

$$e_{k,T} = \frac{1}{2\pi q\, m_k} \int_0^{2\pi\alpha} g^k(\theta)\, d\theta - \frac{\alpha}{q} \tag{5.23}$$

For FWR signals used in the analysis, $g(\theta)$ is known and therefore the product $q \cdot e_{k,T}$ can be evaluated for various $\alpha$. The maximum values of this error for the four periodic signals are given in Table 5.3. For D.C. inputs $g(\theta)=1$, assuming peak value is at the $n^{th}$ level, $m_k=1$ and

| | Maximum error $e_{k,T}$% | | | |
|---|---|---|---|---|
| FWR Waveform | First Moment | Second Moment | Third Moment | Fourth Moment |
| Rectangular Waveform | 0 | 0 | 0 | 0 |
| Sine Waveform | $\pm \frac{0.0526}{q}$ | $\pm \frac{0.0796}{q}$ | $\pm \frac{0.0966}{q}$ | $\pm \frac{0.257}{q}$ |
| Triangular Waveform | $\pm \frac{0.0624}{q}$ | $\pm \frac{0.0961}{q}$ | $\pm \frac{0.1181}{q}$ | $\pm \frac{0.1337}{q}$ |

q = no. of cycles of input, used in measurement process.

Table 5.3.: The error $e_{k,T}$ for the four waveforms.

therefore $e_{k,T}$ is always zero.

### 5.6.2. Error $e_{k,T}$ for FWR Gaussian Noise

Since the autocorrelation function $R_{mm}(t)$ for $v^k$ cannot be easily calculated for Gaussian signals, the general variance expression of Equation (5.15) yields little useful information. An alternate method suggested by S.O. Rice[47], attempts at a direct evaluation of variances. An example of this approach is given in Appendix E. It is seen that this method also has the same disadvantages as for the general expresion of Equation (5.16).

A simple analysis can be made using the statistical theory of large sample sizes. It is known that the four statistics $m_1$, $m_2$, $\sqrt{\beta_1}$ and $\beta_2$ obey the Normal distribution for large sample sizes. R.A. Fisher[25] has given the results of the variances of the distributions of these statistics. When the sample size is large the results can be simplified further. For systematic sampling of a Gaussian signal with a bandwidth of B Hz., the minimum sampling rate is the Nyquist rate 2B Hz. Thus if measurement time is T secs., the sample size N is

$$N = 2BT \tag{5.24}$$

The variance results for large sample size are given in Table 5.4. For a 99% confidence limit, the measured statistics will be within $\pm 2.58\sigma_s$ of the true value. Thus, if the Gaussian signal has a very narrow bandwidth, then the time of measurement must be increased considerably, if the fluctuations in the measured statistics are to be small.

| Statistic | General Expression | Expression for samples at Nyquist Rate |
|---|---|---|
| Mean Value $(m_1)$ | $\sigma_s = \frac{\sigma}{\sqrt{N}}$ | $\frac{\sigma}{\sqrt{2BT}}$ |
| Standard deviation $(\sigma)$ | $\sigma_s = \frac{\sqrt{2}\,\sigma}{\sqrt{N}}$ | $\frac{\sigma}{\sqrt{BT}}$ |
| Skewness $(\sqrt{\beta_1})$ | $\sigma_s = \sqrt{\frac{6}{N}}$ | $\sqrt{\frac{3}{BT}}$ |
| Flatness $(\beta_2)$ | $\sigma_s = \sqrt{\frac{24}{N}}$ | $\sqrt{\frac{12}{BT}}$ |

Table 5.4.: Expressions for standard deviations $(\sigma_s)$ for Gaussian inputs.

## 5.7 Error due to Finite Sampling Rate

If the input signal crosses a quantisation level during an a.d. conversion process before the next a.d. start pulse is produced, then an incorrectly weighted sample occurs, causing an error in each moment value. Thus consider sampling at the $r^{th}$ level as shown in Fig. 5.12. The $\delta_r$ and $\beta_r$ sections of the input signal are incorrectly weighted. In practice the values of $\delta_r$ and $\beta_r$ occurring at all levels are uniformly distributed in the range 0 to 1. Of course, the error due to finite sampling rate occurs only if a sample is taken during the crossing interval. The exact analysis which takes into account the number of crossings of each level and the variation of $\delta_r$ and $\beta_r$ from one level to another is extremely complex. A simpler analysis to evaluate only the worst case maximum errors can be carried out, assuming that a sample occurs at each level crossing. It can be readily shown that the worst case errors occur when $\delta_r=0$ and $\beta_r=1$. Assuming also that fast sampling approximates continuous time averaging, the error $e_{k,p}$ due to a finite sampling rate is given by

$$e_{k,p} = \frac{1}{n^k\ T\ m_k} \sum_{r=1}^{n-1} (W_{k,r+1} - W_{k,r})\tau \cdot N_{r,\beta} \qquad (5.25)$$

where $N_{r,\beta}$ is the number of negative going crossings of the $r^{th}$ level. For a FWR periodic signal of input frequency $f_i$ Hz,

$$N_{r,\beta} = 2f_i\ T$$

and

$$e_{k,p} = \frac{1}{n^k\ m_k} \frac{2f_i}{p} \sum_{r=1}^{n-1} (W_{k,r+1} - W_{k,r}) \qquad (5.26)$$

$$= \frac{2f_i}{p\cdot m_k} \left[ \sum_{r=1}^{n-1} \frac{W_{k,r+1} - W_{k,r}}{n^k} \right] . \qquad (5.27)$$

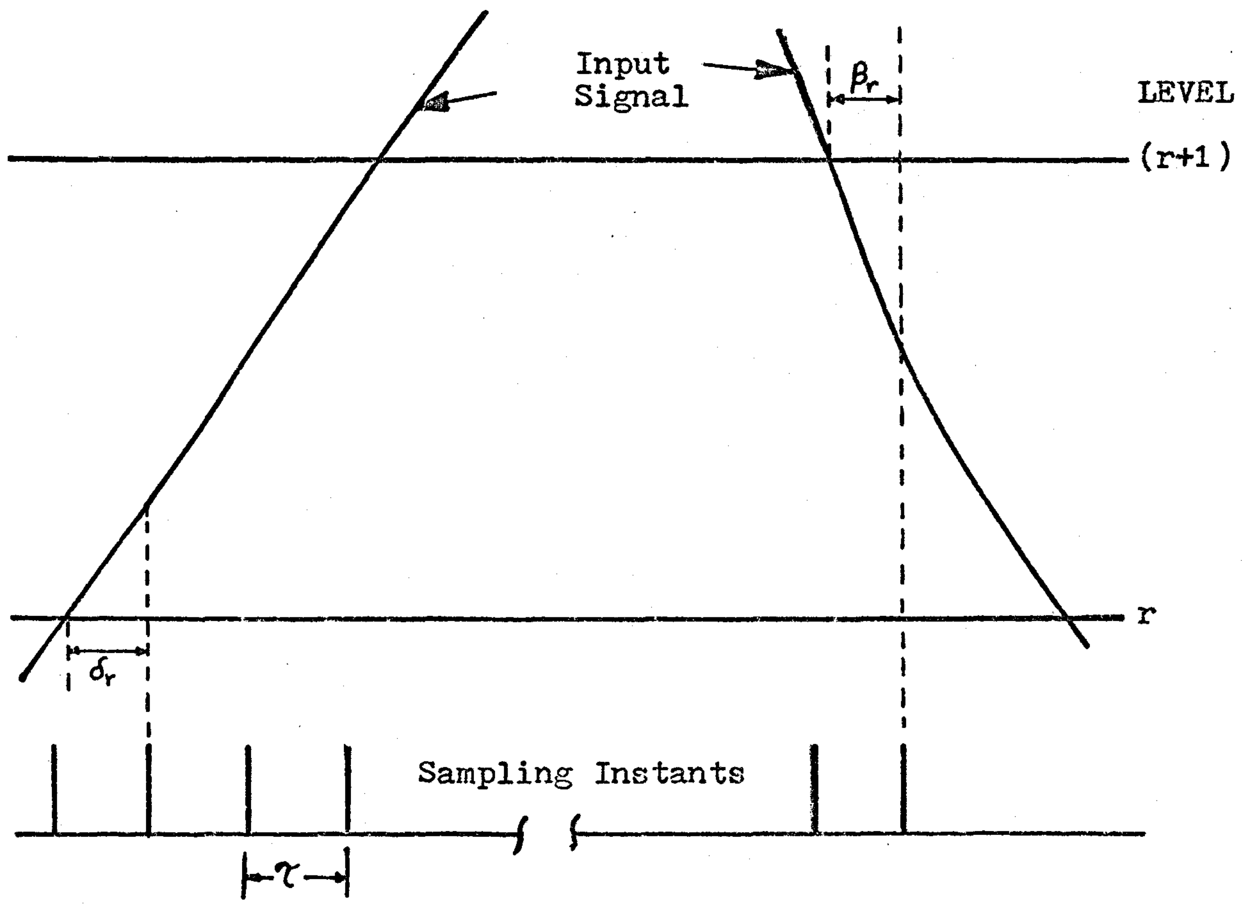


Figure 5.12: Error due to finite rate of sampling.

For large n it can be shown that the quantity within the square brackets approaches unity. Therefore, the worst case maximum error is

$$\text{max. } e_{k,p} = \frac{2\ f_i}{p \cdot m_k} \tag{5.28}$$

For a FWR Gaussian bandlimited signal of bandwidth B Hz, the expected number of negative crossings is given[48] by

$$N_r = \frac{N_o}{2}\ e^{-\frac{r^2}{2n^2\sigma^2}} \tag{5.29}$$

$$\text{where} \qquad N_o = \frac{2B}{\sqrt{3}} \tag{5.30}$$

The worst case error $e_{k,p}$ for this signal is

$$e_{k,p} = \frac{1}{n^k}\ \frac{B}{\sqrt{3}\ p \cdot m_k}\ [\{\sum_{r=1}^{n-1} \frac{W_{k,r+1} - W_{k,r}}{n^k}\} \cdot e^{-\frac{r^2}{2\sigma^2 n^2}}\ ] \ . \tag{5.31}$$

Again it can be shown that the quantity within the square brackets in Equation (5.31) can never exceed unity. The worst case error in the limit is therefore given by

$$\text{max. } e_{k,p} = \frac{B\ K_k}{p \cdot m_k} \qquad \text{where } 0 \leq K_k \leq 1 \tag{5.32}$$

The analysis for $e_{k,p}$ suggests that the ratio f/p should be made small in order to reduce the error due to finite sampling rate. It must also be pointed out that this analysis considers the worst possible situation at each level crossing, i.e., $\delta_r = 0$ and $\beta_r = 1$ and that a sample occurs at every $\beta$ crossing. In practice, however, this is seldom the case and the average error $e_{k,p}$ is much less than the worst case error.

For example, it has been shown[17] that for the second moment the error $e_{k,p}$ has a Gaussian statistical distribution with an average value $\frac{2f_i}{np}$ and a standard deviation of $\frac{4f_i}{np}$ . Thus a 99.73% confidence limit for the error $e_{k,p}$ is $\frac{12f_i}{np}$ . For the s.p.c. designed with a maximum value of p being $10^5$ Hz and n=64, the highest input frequency, $f_i$, is slightly over 5 KHz.

Deliberate jittering of the samples is a simple method for preventing a systematic build up of errors due to finite sampling rate. In the s.p.c. designed, the master clock frequency is voltage controlled, so that a random voltage can be used as an input to this clock and thereby produce the required jittering of the samples.

## 5.8 Errors due to Aperture Time

The a.d. converter, used in the special purpose computer, employs a successive approximation technique, in which the digital output is determined one bit at a time starting with the most significant bit. For this type of a.d. converter, the digital output corresponds to some previous value of the analog input during the conversion process. The aperture time $\mu_a$ is the total conversion time and equals 8 μsec. for the a.d. converter used.

If, during this aperture time, the input changes from the $r^{th}$ to $(r+1)^{th}$ level then an incorrectly weighted sample occurs. Thus, if the slope of the input signal exceeds $\frac{V_a}{\mu_a \cdot n}$ v/sec. an error will occur if a sample is taken at this slope.

The error due to aperture time depends on the probability that a sample occurs at the instant where the slope exceeds the limiting value. Thus for a triangular waveform of 10 V peak and a frequency of

500 Hz, the aperture time errors begin to occur. The magnitude of this error depends on several factors and is therefore not easily analysed. However, aperture time errors have a tendency to cancel, since the errors for positive going slopes usually differ from the errors for negative slopes, in sign only. In practice, therefore, a much higher frequency input can be analysed. Thus in the s.p.c. designed, signals up to 5 KHz have been analysed with a total error being within 1% for any moment. Aperture time errors can be minimised by using a sample and hold circuit before the a.d. converter.

## 5.9 Errors in the Above-level Probability Measurements

Errors in the above level probability arise from two sources viz:

(a) Level offsets at each level

(b) Finite time of measurement.

If the level offsets are identical at each level then the effect will be, simply a shift in the quantisation interval by a fixed amount. However, this is seldom the case and in practice the above-level probability is measured for a level $V_r+\delta V_r$ exceeded instead of the level $V_r$. For coarse quantisation (16 levels) $\delta V_r << V_r$ and therefore the error due to level offsets is negligible. If, on the other hand, $\delta V_r$ is known, then a correction can be easily made in the final results.

The effect of finite time of measurement will be to cause a statistical variation in the measured probability, in the same way as in the measurement of the four moments (section 5.6). For a negligible $\delta V_r$, the s.p.c. produces $\frac{C_r}{C_o} \simeq \frac{\tau_r}{T}$ where T is the total time of measurement and $\tau_r$ is the time for which the rth level is exceeded. It now remains to assess whether $\frac{\tau_r}{T}$ is a good estimate of the above level probability. The quantity $\frac{\tau_r}{T}$ will depend radically on T and of course on the amplitude distribution of the input. A detailed analysis of the fluctuations is not necessary since one would expect the following precautions to be taken during a measurement of the above level probability distribution.

1) The measurement time T will be at least as long as that required for a prescribed confidence limit in the first moment, since the latter is computed by a simple accumulation of the above level probabilities. The measured probability will have a confidence limit comparable to that for the first moment.
2) Several $C_r$ measurements for one level r are made and the average of the $\frac{C_r}{C_o}$ values is used as the above level probability value.

## 5.10 Comments on the Error Analysis

In order to obtain the overall error distribution characteristics, the statistical distribution of the individual errors are required. A convolution approach[17] can be used to obtain the distribution of the sum of these errors, assuming their statistical independence. For design purposes, however, the worst case errors are usually more meaningful.

Accurate experimental error analysis requires a precision signal source with very small but known distortion. The measuring equipment for the actual moments, must have an overall accuracy which is at least an order of magnitude better than that of the s.p.c. Measurements made for the four moments for the standard FWR signals with the highest frequency of 1 KHz indicate that the total error is well within 1% for the fourth moment for a sample size $C_o = 10^6$ and a sampling rate of 10 KHz. The r.m.s. value of the input signal was measured by the Hewlett Packard Model 3450 Multifunction digital meter, which has an overall accuracy of 0.01% in the 10V range.

# CHAPTER VI

# ITERATIVE ARRAYS FOR BINARY ARITHMETIC

## 6.1 Introduction

The weighted feed logic, implemented for the s.p.c. uses special purpose simultaneous multipliers which complete a 24-bit binary multiplication within 50 ns. The design of such multipliers makes an efficient use of the fact that for the weighting numbers, the multiplier and the multiplicand are interrelated and are in a restricted range. However the W.F.L. units cannot be easily generalised to accommodate a larger number of bits. Thus if instead of a 6-bit s.p.c., an 8-bit machine were required, then the entire design procedure outlined in Chapter III for the W.F.L. units would need to be carried out again, resulting in a cumbersome and costly hardware system.

In this chapter, the use of iterative and near-iterative cellular arrays for arithmetic operations is investigated. A universal arithmetic cell (U.A.C.), based on the rules of binary addition and subtraction, has been developed for use in cellular arrays. The arithmetic operations considered are multiplication, division and square-root extraction. A method of intercoming such arrays for the realisation of the weighted feed logic is also described.

## 6.2 Review

It has now been recognised that there are many problems in the area of computer structures, especially useful in information processing[49], that involve the presentation of data in the form of a single uniform array. The arithmetic operations of binary multiplication and division are typical problems in this area. Their problem structure suggests that it would be possible to design a group of processing arrays in which each array is composed of a number of identical cells interconnected in a regular fashion. An iterative array has been defined by Hennie[49] as one that is composed of a number of identical logic cells with all connections to the neighbouring cells being regular.

Arrays constructed in an iterative or in a near-iterative fashion have several advantages. Being made up of identical cells, they are economical to manufacture and repair. An array can be easily enlarged to accommodate more variables, e.g., an increased number of binary bits, by simply adding more cells. Since some of the processing can be carried out in parallel, a considerable improvement in speed usually results. Use of cellular arrays for arithmetic operations was initially suggested by Hoffmann, Csillag and Lacaze[50], who proposed a simple multiplier in which the conventional add and shift algorithm was implemented in an iterative array. Significant contributions by several authors followed this work[51-62]. The array structures proposed by these authors begin the multiplication process with the l.s.b. of the multiplier in the conventional manner and obtain a speed of multiplication for two n-bit numbers as $(3n-1)\tau$ where $\tau$ is the cell delay. Similarly division is implemented using a restoring type algorithm[57] which tends to be slow. The array structures proposed were not easily applicable to other arithmetic operations, e.g., it is not possible to use a single array for both multiplication and division.

The investigation described in this chapter differs somewhat from the above-mentioned work, in that it emphasises the concept of a universal logic cell for use in all arithmetic arrays. This method of implementation of arrays allows the same array to be used for multiplication and division.

## 6.3 The Universal Arithmetic Cell (U.A.C.)

The Universal Arithmetic Cell (U.A.C.) proposed here has been developed[60] using the truth-table for binary addition or subtraction shown in Table 6.1. In this table, C represents the carry-in for addition and a borrow-in for subtraction. A and B are the primary inputs.

### 6.3.1. Addition

The logic requirements for the sum $S_o$ and the carry-out $C_o$ can be easily simplified using three-variable Karnaugh maps, giving the following expressions.

$$S_o = \bar{C}[A\bar{B} + \bar{A}B] + C\overline{[A\bar{B} + \bar{A}B]}$$

$$S_o = A \oplus B \oplus C \qquad (6.1)$$

where $\oplus$ denotes "exclusive OR" operation

and $$C_o = A[B + C] + BC \text{ .}$$

| Inputs | | | Addition | | Subtraction | |
|---|---|---|---|---|---|---|
| A | B | C | $S_o$ | $C_o$ | $D_o$ | $B_o$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 6.1.: Binary Addition and Subtraction

### 6.3.2. Subtraction

A similar logic minimisation as for addition gives

$$D_o = A \oplus B \oplus C$$

and (6.2)

$$B_o = \bar{A}(B + C) + BC$$

### 6.3.3. Generalised Arithmetic Operation and the U.A.C.

Comparison of Equations (6.1) and (6.2) shows that the sum and difference expressions are identical whereas the carry-out and borrow-out expressions differ only in the variable A. Thus a generalised addition/subtraction operation can be described by the following equations.

$$\begin{aligned} S &= A \oplus B \oplus C \\ P &= W[B + C] + BC \\ W &= A \oplus F \,. \end{aligned} \tag{6.3}$$

The input F is a control input and decides the mode of operation.

Binary arithmetic also requires the facility to inhibit an operation if desired, i.e., to let the primary inputs A and B go through the processor unaltered. This inhibiting decision is to be provided by an external control state D. Thus the generalised adder/subtractor with this inhibit control has logic equations which are

$$S = [A \oplus B \oplus C]D + A\bar{D}$$

i.e., $S = A \oplus BD \oplus CD$ (6.4)

and $P = W[B + C] + BC$

To complete the design of the universal cell, only a few minor additions are required. Thus the states B, D and F are also made available as outputs. The U.A.C. has five inputs and five outputs and is described by the following logic equations

$$\begin{aligned} S &= A \oplus BD \oplus CD \\ P &= W[B + C] + BC \\ \text{where} \quad W &= A \oplus F \\ U &= D \\ V &= B \\ G &= F \end{aligned} \tag{6.5}$$

This cell together with one logic realisation is shown in Fig. (6.1).

A useful modification of the U.A.C. is to replace A in the expression for S in Equations (6.5), by W. This modification requires no extra components and can be easily implemented using a binary adder and some additional logic gates. Its primary use is in cellular arrays for complement arithmetic operations[60].

## 6.4 Functions of the U.A.C.

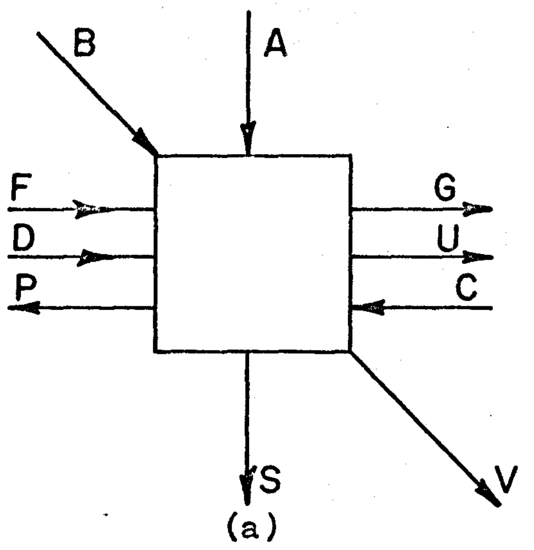
The arithmetic operations of the U.A.C. are controlled by the inputs F and D. The following operations are relevant.

### 6.4.1. Controlled Adder F = 0

The cell equations for S and P are

$$\begin{aligned} S &= A \oplus BD \oplus CD \\ P &= A[B + C] + BC \end{aligned} \tag{6.6}$$

The other outputs are as given in Equations (6.3).



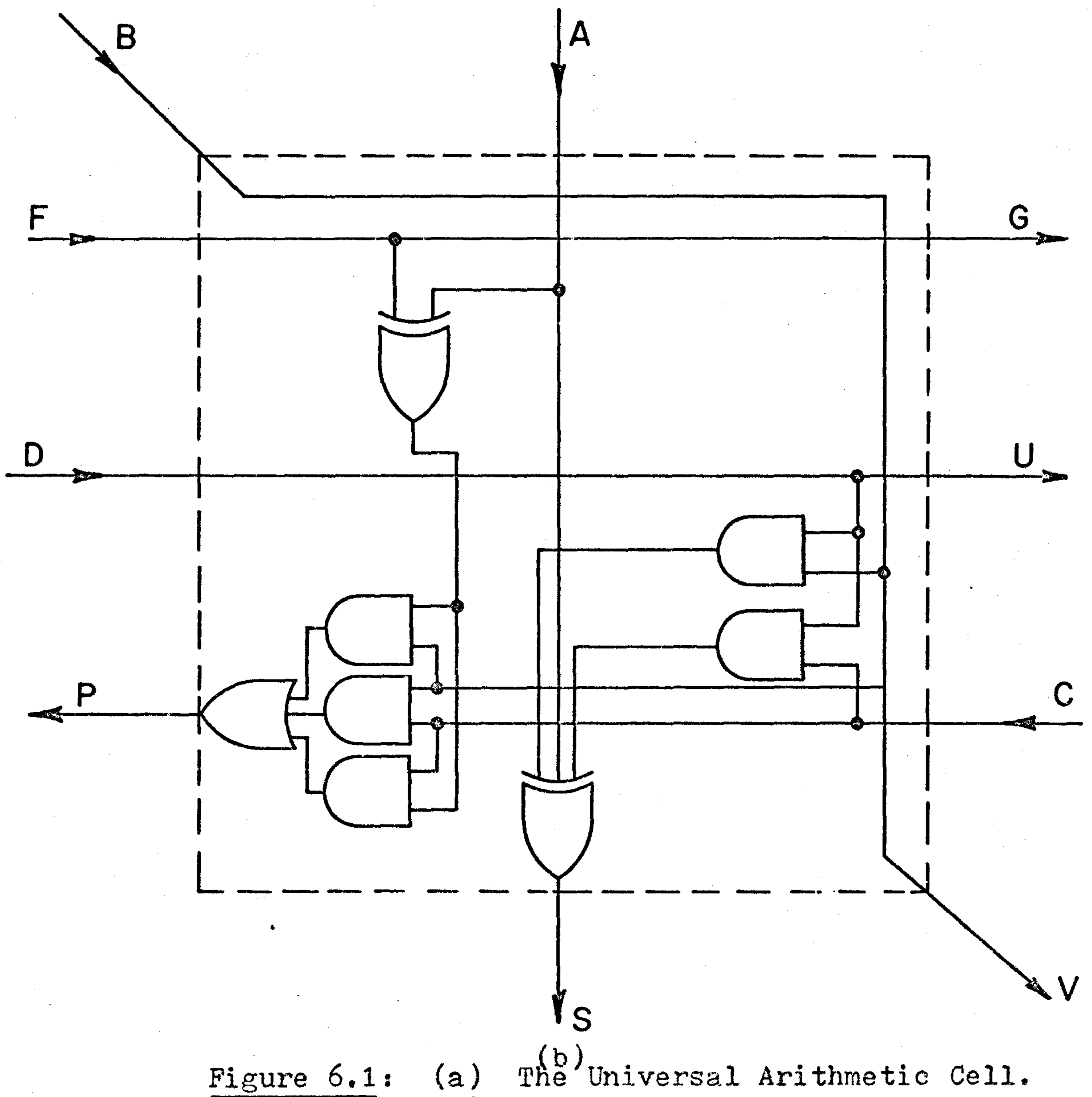


Figure 6.1: (a) The Universal Arithmetic Cell.
(b) A logic realisation of the U.A.C.

The cell is a controlled adder in which S is the sum and P is the carry-out. If D = 0, S = A and no arithmetic is performed. This mode is useful in binary arithmetic.

### 6.4.2. Controlled Subtractor F = 1

The logic equations for S and P become

$$\begin{aligned} S &= A \oplus BD \oplus CD \\ P &= \bar{A}[B + C] + BC \end{aligned} \tag{6.7}$$

The cell is a controlled subtractor in which S is the difference and P is the borrow-out. If D = 0, no arithmetic is performed. The cell can be used in a division process using the restoring algorithm[57].

### 6.4.3. Conditional Adder/Subtractor D = 1

The logic equations for S and P are

$$\begin{aligned} S &= A \oplus B \oplus C \\ P &= [A \oplus F][B + C] + BC \end{aligned} \tag{6.8}$$

The controlled U.A.C. is a full adder when F = 0 and when F = 1 it is a full subtractor. This is the most useful mode of operation since it enables a single array to be used as a multiplier or divider.

### 6.4.4. Ternary Operation

When both F and D are used as control lines, the U.A.C. can perform addition, subtraction or leave the inputs A and B unaltered. This ternary operation is useful in multiplication of signed binary numbers[62].

## 6.5 Cellular Arrays Using the U.A.C.

The universal cell operated as a controlled adder or subtractor may be used in any of the cellular arrays mentioned earlier[51-60]. However, these arrays implemented the slower algorithms for multiplication and division. One method of increasing the speed is to use some form of "carry save" techniques[63] for multiplication. For division or square-root extraction, considerable speed improvement can be obtained by using non-restoring algorithms[63].

### 6.5.1. Array for Multiplication

Figure 6.2 shows a three row array with the additional logic required to perform multiplication of two 3-bit binary numbers, using a 'carry save' technique. The order of multiplication is reversed with the m.s.b. of the multiplier being considered first. The F inputs to all the cells are in state 0, i.e., they are used as controlled adders. To obtain the product LM, M is applied to the B inputs of the top row and all A inputs of this row are in state 0. The L inputs are applied to the row control lines D. The least significant bit C inputs are obviously zero. In Figure 6.2 L = 101 and M = 111 have been chosen. All the intermediate states are also shown. The product is obtained at the S outputs of the lowest row, with the m.s.b. being the logical OR of the 'carry outs' from the m.s.b. cells of rows whose D inputs are 1. This is implemented in each row by an AND gate fed from the D input and the m.s.b. carry output. The AND gate outputs constitute the inputs to the final OR gate. No AND gate is required in the top row since its carry out P is always zero.

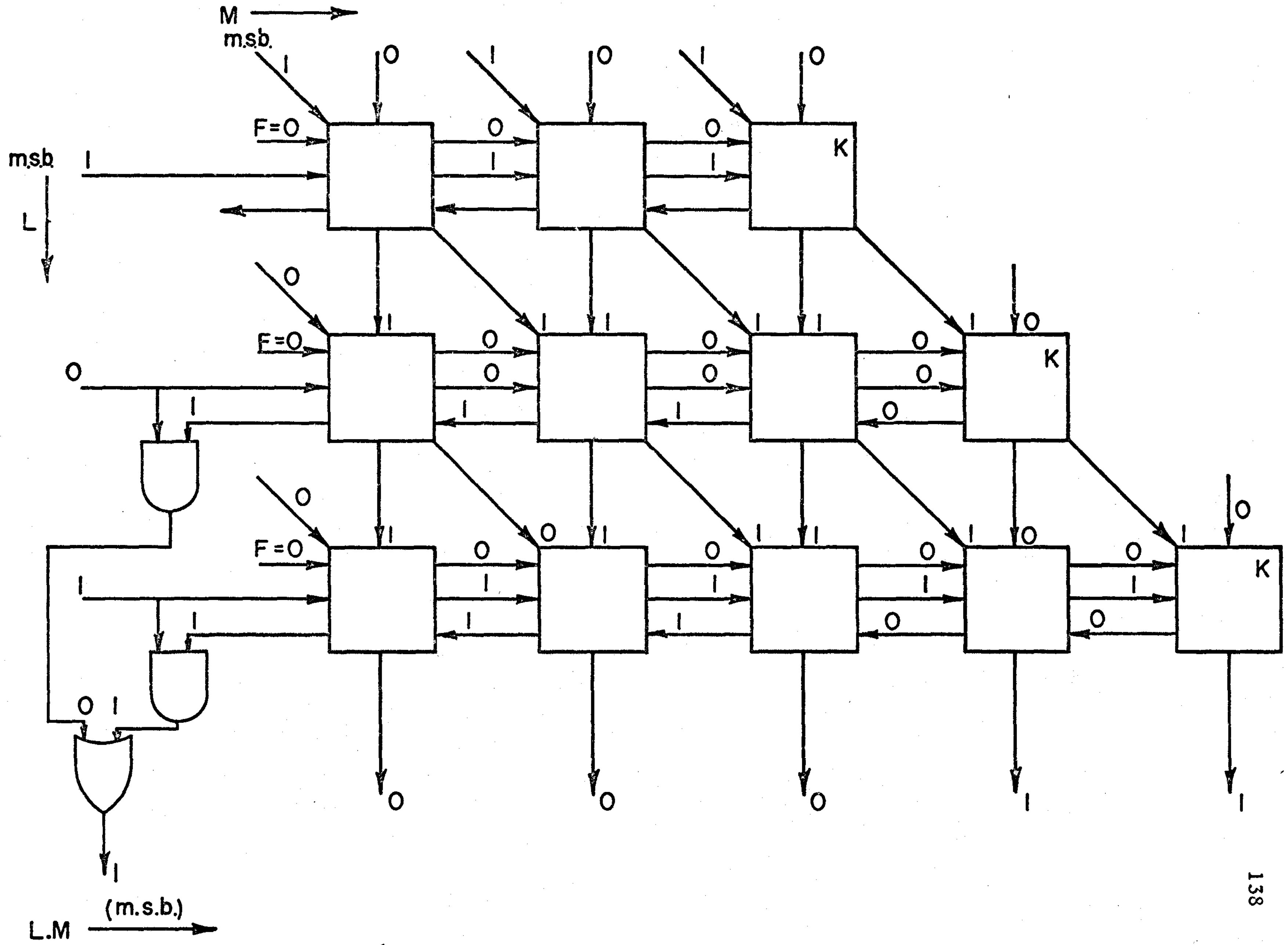


Figure 6.2: A 3 - bit Multiplier Array.

The array requires $\frac{n(3n-1)}{2}$ cells for multiplication of two n-bit numbers. Also (n-1) 2-input AND gates and one (n-1)-input OR gate are required. The states on the diagonal inputs and outputs of the cells appear without any delay. Propagation of the carry outs from one diagonal of cells to the next m.s.b. diagonal is simultaneous. Thus the l.s.b. of the product is available after a delay equal to a cell delay $\tau$. The next bit is available after $2\tau$. The carry out of the m.s.b. cell of the last row is available after a delay $(2n-1)\tau$. If $\tau_g$ is the delay of the external AND and OR gates then the maximum total delay for multiplication of two n-bit numbers is

$$T_m = (2n-1)\tau + 2\tau_g \text{ sec.}$$

The multiplier array is also capable of performing the arithmetic operation LM+K where L, M and K are all n-bit numbers. The K inputs are applied to the A lines which are marked K in Figure 6.2. Since a carry can result from the first row, an additional AND gate is required for this row.

### 6.5.2. Non-restoring Division Array

For a non-restoring algorithm the U.A.C. is used as a conditional adder/subtractor. The non-restoring algorithm[60,64] requires addition for the divisor when the remainder is negative. The division is performed by shifting the remainder to the left at each step (or by shifting the divisor one position to the right) and either adding to or subtracting from the partial remainder. The D inputs of all the cells in the array for division are in state 1 and the F inputs are used as control lines. If $F_n$ is the control input to the nth row and $P_n$ is the borrow out/carry out signal then the quotient bit $Q_n$ and the control $F_{n+1}$ for the next row may be derived from the following truth table.

| Control to $n^{th}$ row $F_n$ | Borrow/Carry out signal $P_n$ | Quotient $Q_n$ | Control to $(n+1)^{th}$ row $F_{n+1}$ |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |

Table 6.2.: Quotient bit & control states for $(n+1)^{th}$ row.

Note that $F_n=1$ implies subtract mode and $F_n=0$ implies add mode. From the truth table one obtains

$$Q_n = F_{n+1} = F_n \oplus P_n$$

The array for division by the non-restoring algorithm is shown in Fig. 6.3. The dividend is applied to the A inputs and the divisor is at the B inputs of the top row. The F input to the first row is at 1. For each row the borrow out/carry out signal P from the m.s.b. cell and the control signal F constitute the inputs to an exclusive-OR gate. The output of this gate is the quotient bit and also the control signal for the next row. Clearly the number of rows may be increased to obtain more bits of the quotient. In Fig. 6.3, division of binary L=011 by M=100 is treated. All intermediate states are also shown.

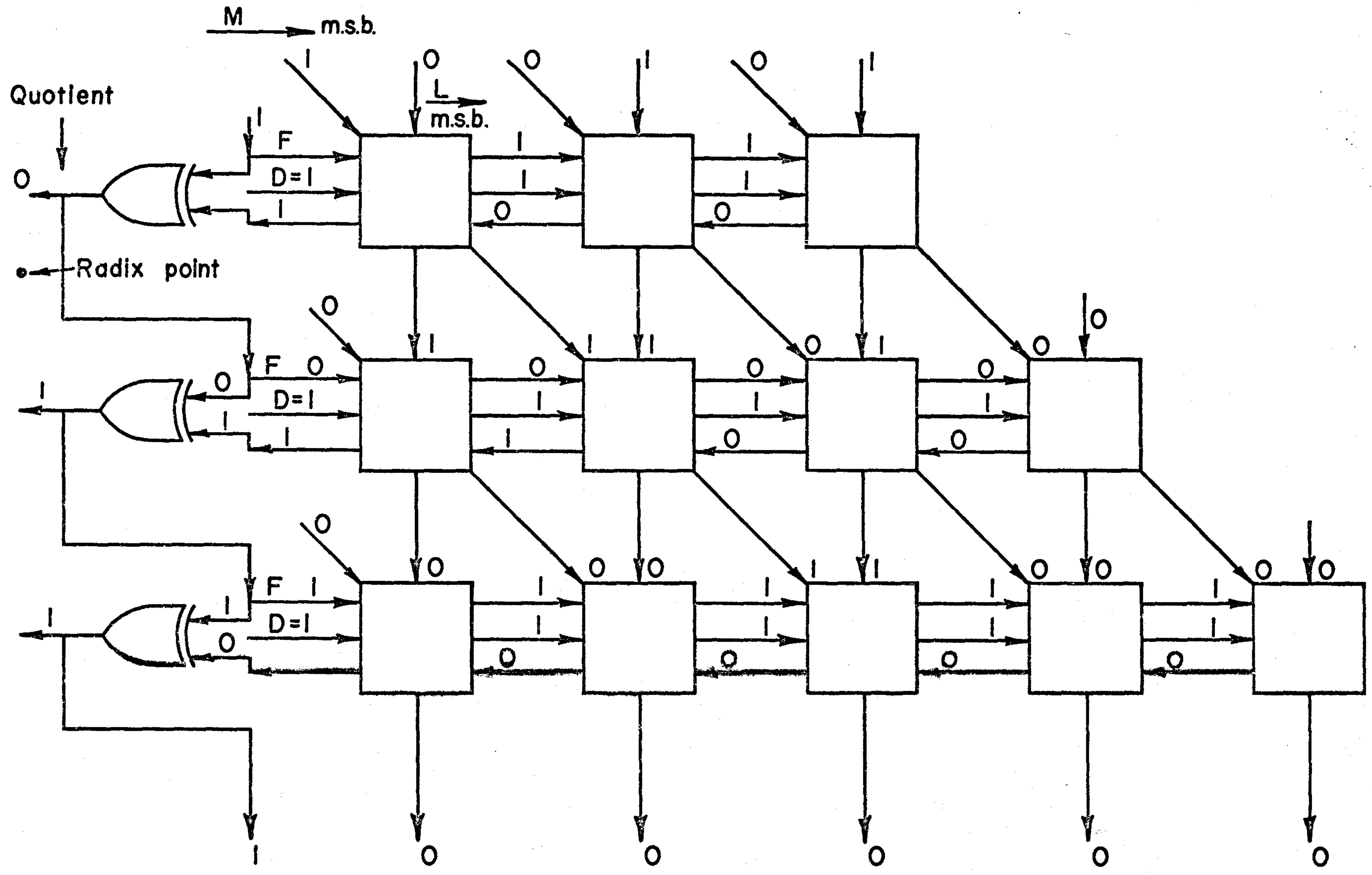


Figure 6.3: A 3-bit Non-restoring Division Array.

The number of cells required for division of two n-bit numbers to produce an n-bit quotient is the same as that for the multiplier. If however, m bits of the quotient are required, then the array will have $\frac{m(m + 2n - 1)}{2}$ cells.

The total delay to produce the m-bit quotient will be

$$T_d = \frac{m(m + 2n - 1)\tau}{2} + m\tau_g \text{ sec.}$$

where $\tau$ = cell delay and

$\tau_g$ = exclusive-OR gate delay.

### 6.5.3. General Array

A significant feature of the multiplier and divider arrays of Fig. 6.2 and 6.3 is that their cell interconnections are the same, so that a multiplication or division operation may be readily performed by a small control logic block for each row of the array[63]. An example of such a logic block is shown in Figure 6.4. The nth digit $L_n$ of number L is fed to the block for the nth row of the array. The block outputs for feeds to the nth row are $D_n$ and $F_n$. The $A_n$ output of the block is applied to the A input of the nth cell of the top row. The block carry in/borrow in is $P_n$. Control of the array is exercised by Z. For Z=0, $A_n=0$, $D_n=L_n$ and the array acts as a multiplier with L as the multiplicand and M the multiplier. When Z=1, $A_n=L_n$, $D_n=1$ and the array is a non-restoring divider with L as the dividend and M as the divider. The input M is applied to the first row in both cases so that the M inputs do not feature in the control logic. The outputs marked $R_n$ feed an OR gate which is required for the m.s.b. of the product LM in the multiplication mode.

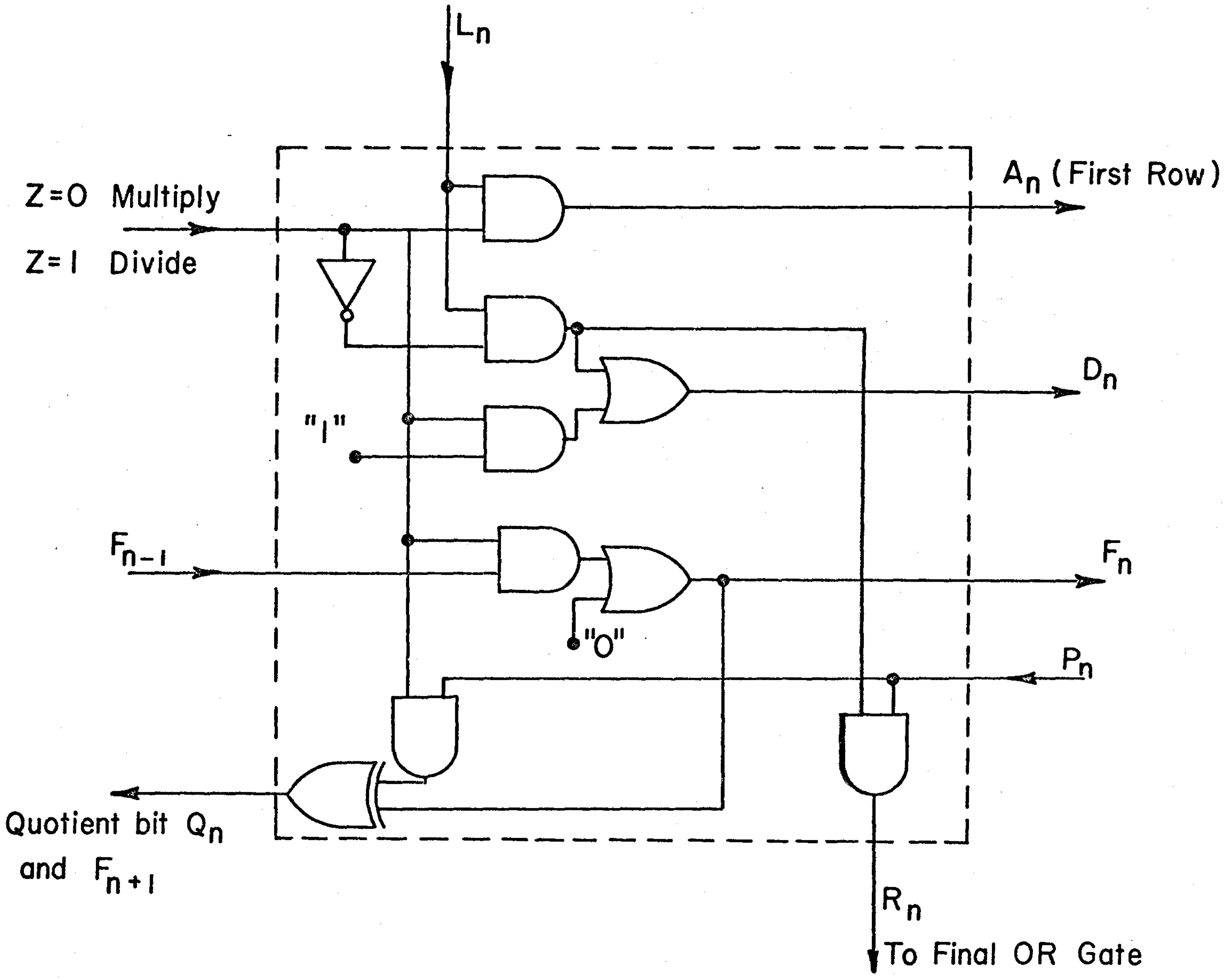


Figure 6.4: Logic block required for each row in the general array.

A general array for processing two n-bit numbers will require n control logic blocks in addition to the n-row array. The product will have 2n bits and the quotient n bits. The speed of processing is slightly reduced due to propagation through the control logic blocks. The array structure can be extended readily for larger numbers or for increased accuracy in division.

## 6.5.4. Array for Multiplication of Signed Binary Numbers

For the multiplier of Section 6.5.1., if a negative number (multiplier or multiplicand) is in two's complement form, then it has first to be converted to a sign and magnitude form before multiplication can be performed.

The general multiplier considered here requires no prior knowledge of the sign of the multiplier or multiplicand. If the final answer is negative, then the product is in two's complement form. Such arrays can be used in data processing and would also eliminate the use of the precision rectifier in the s.p.c. The array uses an algorithm due to A.D. Booth[65] and is summarised below for multiplication of two signed n-bit numbers x and y. The number x is assumed to be a binary fraction having (n+1) bits and is written

$$x = x_o 2^o + x_1 2^{-1} + \ldots x_n 2^{-n} \qquad (x_r = 0,1)$$

where $x_o$ is the sign bit. The rules for multiplication apply to each digit $x_k$ starting with the least significant bit.

(i) If $x_k = x_{k+1}$, shift the existing sum of partial products one place to the right.

(ii) If $x_k = 1$, $x_{k+1} = 0$ i.e., $x_k > x_{k+1}$, subtract y from the existing sum of partial products and shift the new sum one place to the right.

(iii) If $x_k = 0$, $x_{k+1} = 1$, i.e., $x_k < x_{k+1}$, add y to the existing sum of partial products and shift the new sum one position to the right.

No shift is required after the last operation is carried out. Clearly the operation can start with the most significant bit of x, instead of with the least significant bit as originally proposed. Furthermore, if the multiplicand is shifted instead of the partial sum, the same end result is obtained with the advantage that some parallel processing may be used, resulting in an improvement in the speed of multiplication. It should be noted that when a number with a 0 in its m.s.b. position is right-shifted, there will also be a 0 in the m.s.b. position of the new number. Similarly, for a number with a 1 in its m.s.b. position, there is also a 1 in the m.s.b. position after the right shift.

<u>Example</u>: Consider multiplication of $y = 0.01$ by $x = -0.11$. In two's complement form $y = 0.01$ and $x = 1.01$.

(a) $x_o = 1$, $x_1 = 0$, subtract y from 000 to give $y_{p1} = 1.110$

Shift y one place to give $y_1 = 0.001$

(b) $x_1 = 0$, $x_2 = 1$, add $y_1$ to $y_{11}$

$y_{p2} = 1.111$

Shift $y_1$ to give $y_2 = 0.0001$

(c) $x_2 = 1$, $x_3 = 0$, subtract $y_2$ from $y_{p2}$

$y_{p3} = 1.1101$ No shift is required at this step.

The answer is 1.1101 (i.e., -0.0011)

The array for multiplication of signed binary numbers uses the U.A.C.'s in the ternary mode of operation[62]. The iterative array is shown in Fig. 6.5. It is capable of multiplying two three-bit signed numbers. The A inputs to the first row are in state 0. The Y inputs are at the B lines of the first row with the m.s.b. at the extreme left cell. The B inputs to the m.s.b. cell of any row other than the first are obtained from the B inputs to the next cell in that row. Thus the diagonal lines present the multiplicand Y in a correctly shifted position for further processing. Furthermore, since no shift is required after the last operation, the final product is available at the S outputs of the last row, without any processing. A one-bit comparator is used to obtain the results of comparison of $x_k$ and $x_{k+1}$ for the $k^{th}$ row and provide the D and F inputs to that row. Thus $[x_k > x_{k+1}]$ is the F input and $[\overline{x_k = x_{k+1}}]$ is the D input. If $x_k > x_{k+1}$ is denoted by $[x_k > x_{k+1}] = 1$ then F = 1, D = 1 and the row with these inputs will be in the "subtract" mode. If $x_k < x_{k+1}$ is denoted by $[x_k > x_{k+1}] = 0$, then F = 0, D = 1 and the row is in the "add" mode. When $x_k = x_{k+1}$, $x_k = x_{k+1}$, $[\overline{x_k = x_{k+1}}] = 0$, therefore, D = 0 and the row performs no arithmetic. In Fig. 6.5, multiplication of Y = 0.01 by X = 1.01 (i.e., +0.01 by -0.11) is treated. All intermediate states are also shown.

If both X and Y are two n-bit numbers including the sign bit, then the array requires $\frac{n(3n-1)}{2}$ cells and n 1-bit magnitude comparators.

If the comparators have a delay of $\tau_c$ sec. each, then the mode of operation of each row is settled after $\tau_c$ sec. Assuming a cell delay of $\tau$ to produce either S or T output, the l.s.b. of the produce appears

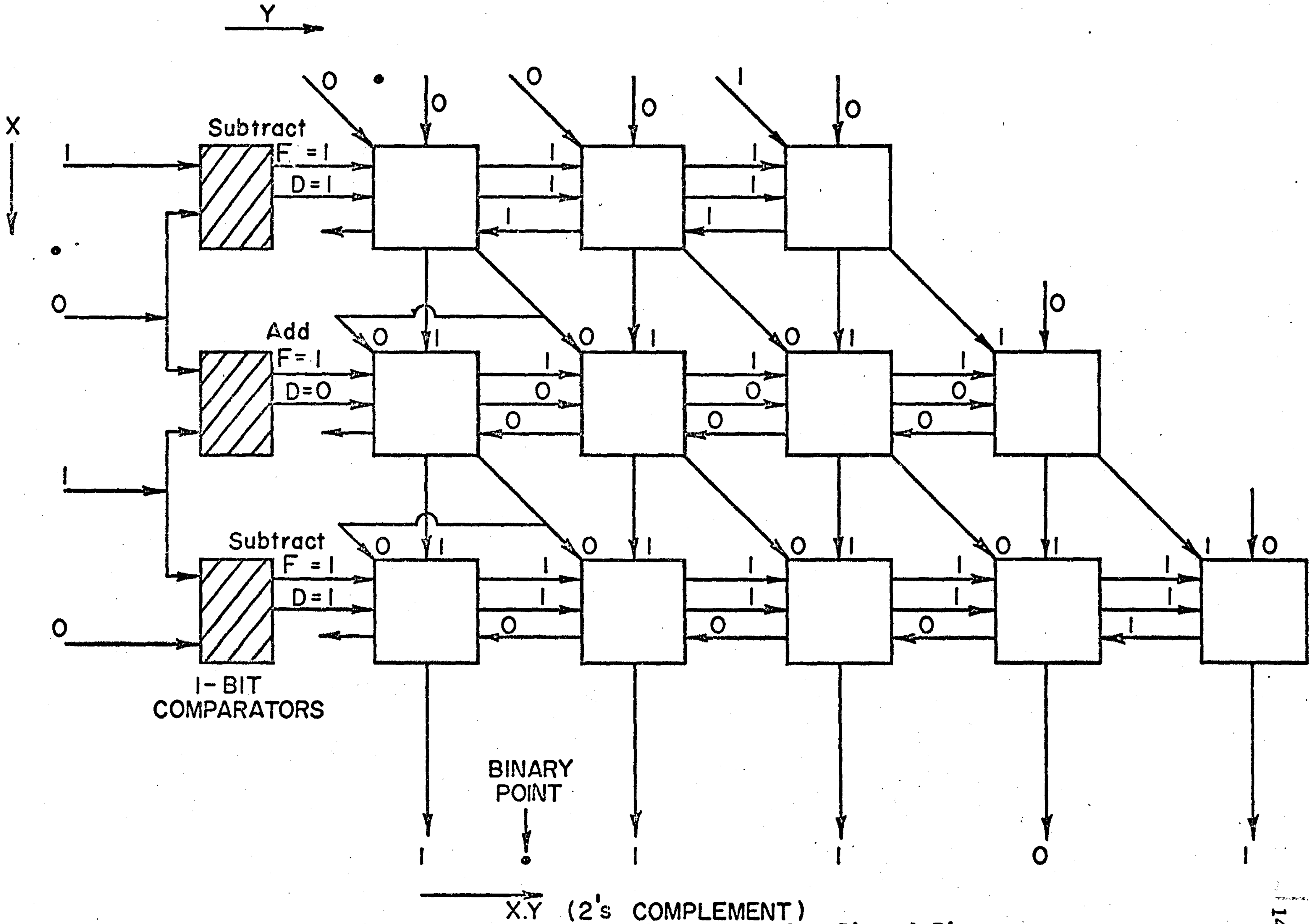


Figure 6.5: A 3-bit Multiplier Array for Signed Binary Numbers.

after a time $\tau+\tau_c$ sec. The states on the diagonal lines appear without any delay. Therefore, the next bit of the produce will be available after a delay $2\tau+\tau_c$, the third bit after a delay $3\tau+\tau_c$ etc. Since there are (2n-1) diagonals of cells, the delay in obtaining the m.s.b. of the product will be

$$T_{gm} = (2n-1)\tau + \tau_c \text{ sec.}$$

This delay is comparable to the delay of the general multiplier/ divider array described earlier in this chapter.

## 6.5.5. Array for Square-root Extraction

A non-restoring division array can be readily extended to yield a square-rooting array which is near-iterative, but has a simple inter-cellular connection pattern.

The non-restoring square-root algorithm may be outlined as follows:

(a) The binary number N is paired off starting from the radix point. Let these pairs be $A_n$, $A_{n-1}$, etc.

(b) The first minuend is 01. Subtract this from $A_n$. If the remainder is positive, 1 is entered in the square-root answer. 01 is appended to the square-root developed so far, shifted by the correct number of times and a further subtraction is attempted.

(c) If the result of subtraction is negative, then 0 is entered in the square-root. 11 is appended to the square root developed so far and this number is added to the remainder after being shifted correctly.

When the remainder is positive, the square-root is always 1, 01 is appended and a subtraction is performed on the next cycle. If the remainder is negative, the square-root bit is 0, 11 is appended and an addition performed on the next cycle.

Fig. 6.6 shows a decision tree of the possible numbers to be added or subtracted in each cycle. The square-root bit developed in the previous cycle is SA and is also indicated.

An iterative array for square-root extraction is shown in Fig. 6.7. The D inputs are at 1 in each row. The number N is applied in pairs to each row. The least significant number to be added or subtracted is always 1 (Figure 6.6). Furthermore, the next two bits are always SA and $\overline{SA}$, SA being the square-root bit developed in the preceding cycle. Since SA is also the F inputs to a row, the next two feeds may be obtained from these lines, as shown in Figure 6.7. The remaining bits of the numbers of Figure 6.6 are obtained from the diagonal outputs of the preceding row. As for non-restoring division, an exclusive-OR gate is required for each row to produce the square-root bit and F input to the next row. In Figure 6.7, the square-root of 1001 is treated. All intermediate states are also shown.

The square-root array requires n(n+1) cells to extract an n-bit square-root of a 2n-bit number. If the cell delay is $\tau$ and the exclusive-OR gate delay is $\tau_g$ then the total delay in obtaining the n-bit square-root will be

$$T_s = n(n+1)\tau + n\tau_g \text{ sec.}$$

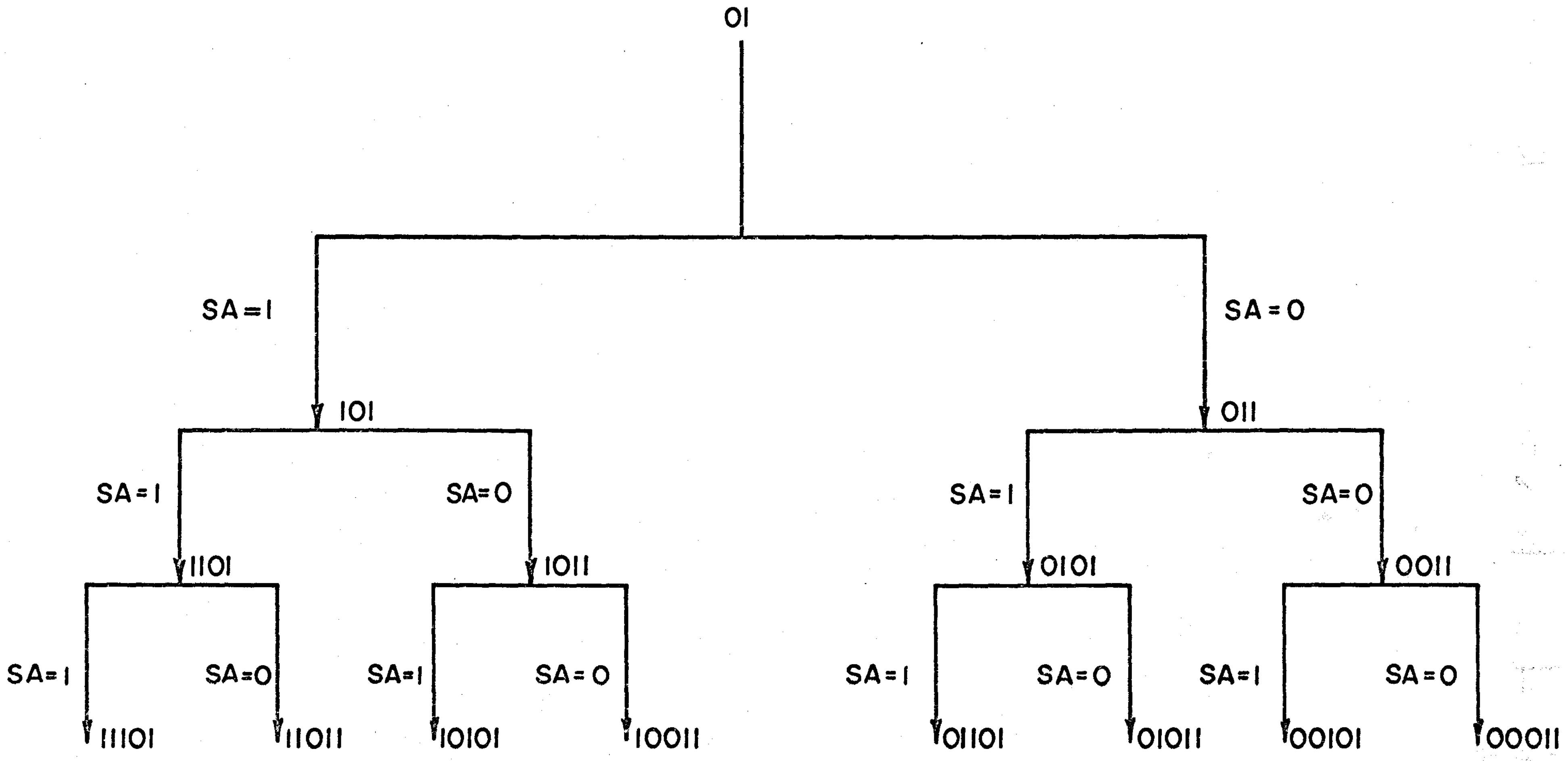


Figure 6.6: A Decision Tree for Non-restoring square-root extraction.

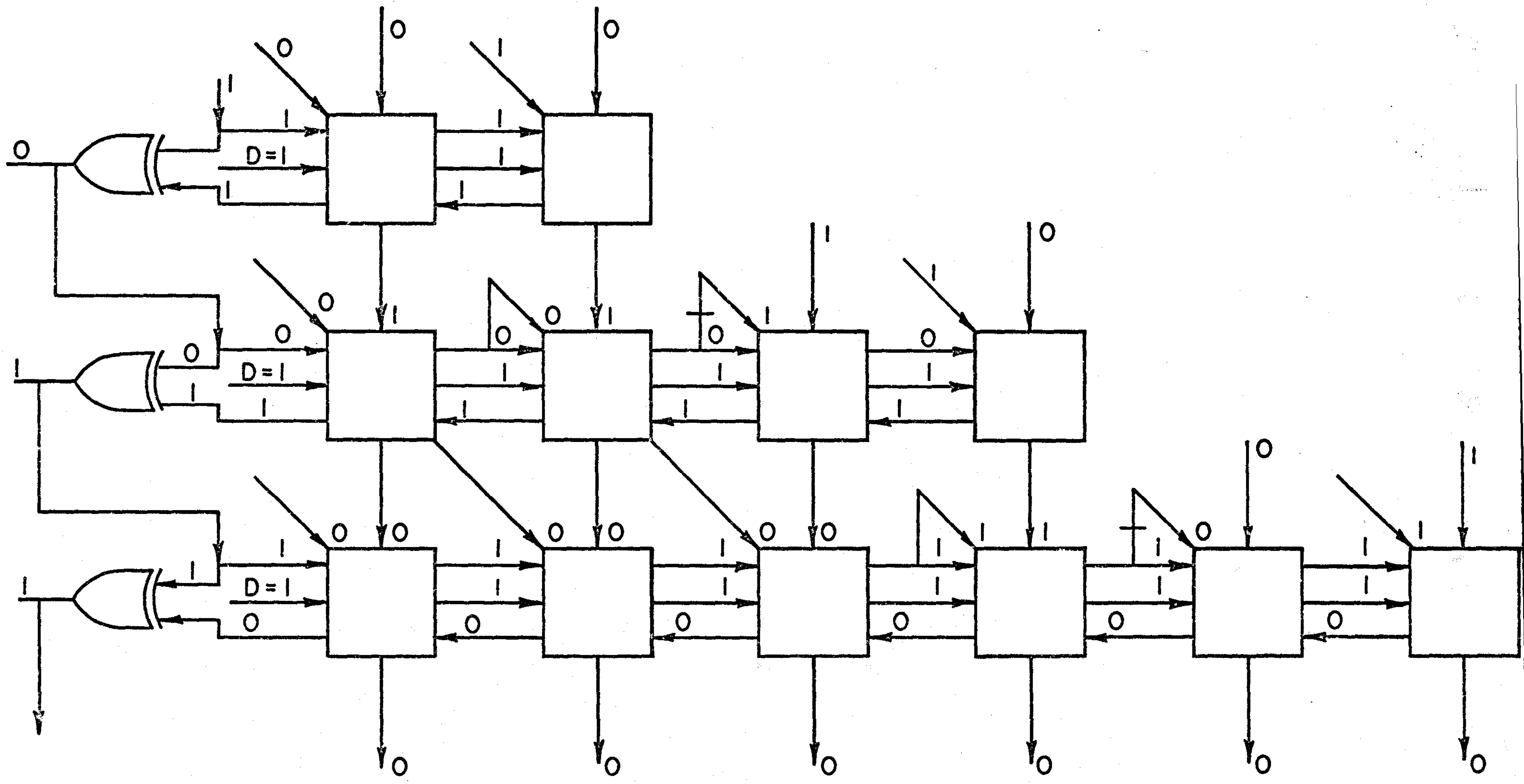


Figure 6.7: A 6-bit array for Non-restoring Square-root Extraction.

## 6.6 BCD Multiplication

It is possible to perform complex arithmetic operations such as BCD multiplication by connecting the arrays of the previous section in an iterative system. The iterative array proposed here for BCD multiplication, uses sub-arrays, each of which has two basis operations. The first operation is multiplication of two binary numbers and additions of any carry-ins. The second operation is division by a fixed divisor decimal 10(1010 binary), of the multiplication results. The general multiplier/divider arrays can be used in this application. Thus, referring to Fig. 6.8, the multiplier together with an extra adder produces [A·D + B + C] and the divider, divides this by 10 and produces a remainder P and a quotient Q. The operation of such a sub-array is then given by the following equations.

$$P = \text{Rem.}[\{A \cdot D + B + C\}/10]$$

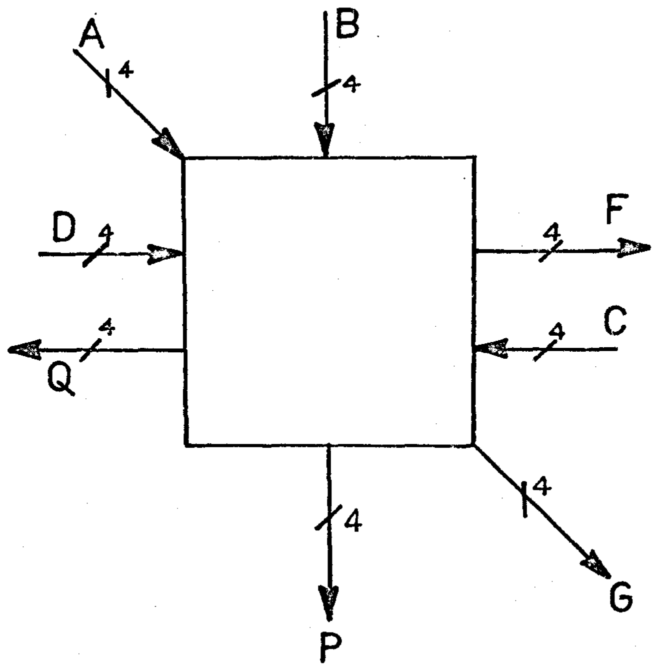
$$Q = \text{Quo.}[\{A \cdot D + B + C\}/10]$$

$$F = D$$

$$G = A$$

Note that in Fig. 6.8 all the literals are BCD variables, i.e., each input/output line represents four lines of a BCD digit.

The iterative array for BCD multiplication is shown in Fig. 6.9. Decimal equivalents of the BCD digits are used. In Figure 6.9, the operation [999 x 999 + 999 + 999] is treated. All intermediate states are also shown. Each sub-array multiplies two BCD numbers and adds any carry-ins to produce a result which cannot exceed 99. This result is then divided once by decimal 10 to produce the BCD remainder and a BCD quotient. The quotient becomes the carry-in to the next m.s.b. sub-array.



<u>Figure 6.8:</u> Generalised Multiplier/Divider Cell for BCD arithmetic.

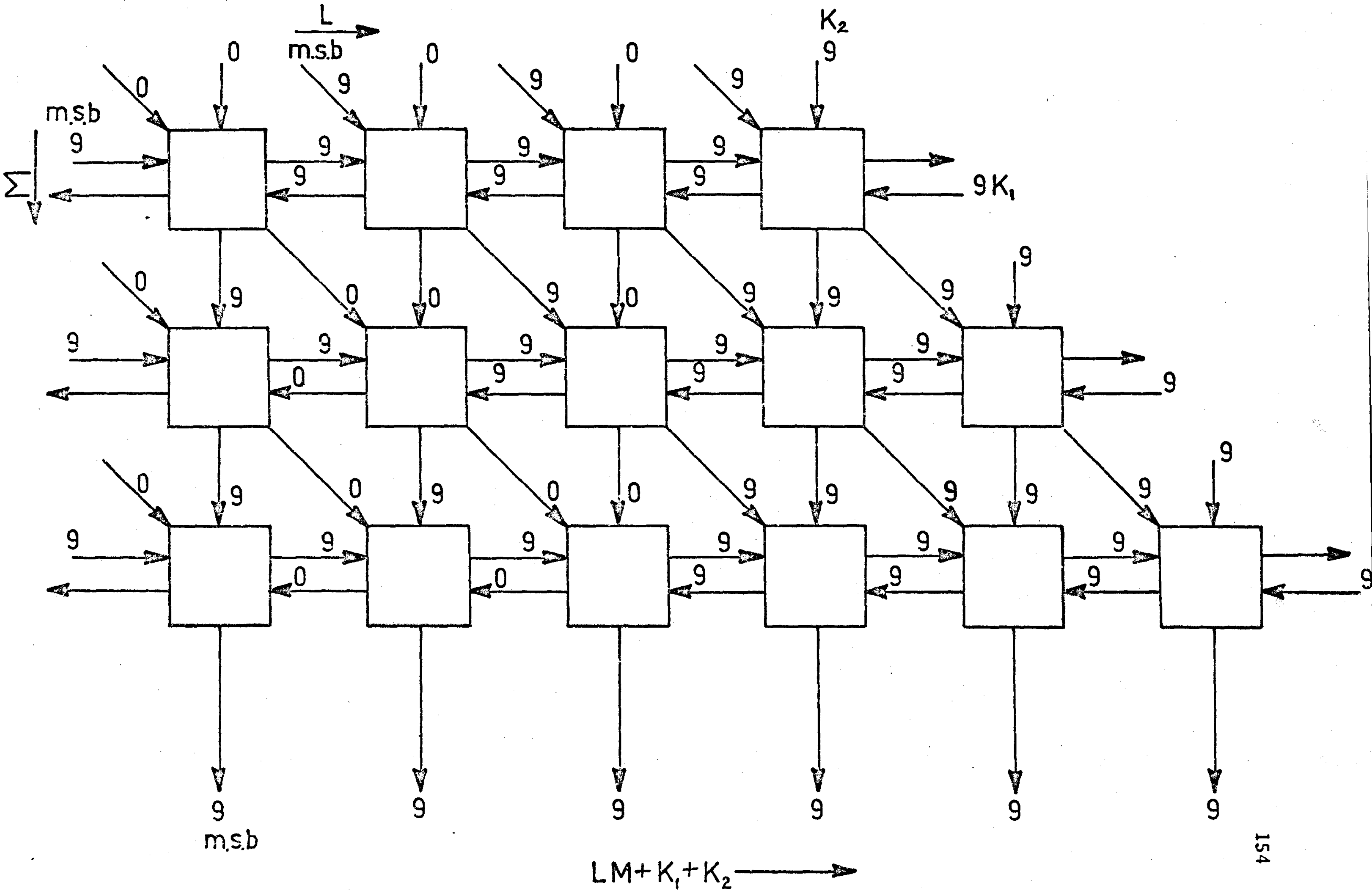


Figure 6.9: A 3-digit BCD Multiplier Array.

Each sub-array thus consists of a 4-bit by 4-bit binary multiplier and a divider to accommodate a seven-bit dividend and a four-bit divisor. Since both of these processes can use the same logic structure, the sub-array can be realised as a single LSI function.

The BCD multiplier requires $\frac{n(3n+1)}{2}$ sub-arrays to multiply two n-digit numbers. For a sub-array delay of $\tau_s$, the first digit of the product is available after a time $\tau_s$. The carry-outs from one diagonal propagate simultaneously to the next m.s.b. diagonal, there being parallel processing in each diagonal set of sub-arrays. The BCD multiplication is therefore completed within $2n\tau_s$ sec. Addition of $K_1$ and $K_2$ involve no extra time delay.

## 6.7 Application of Iterative Arrays in W.F.L. Units

The cellular arrays can be interconnected to obtain the weighting numbers for the three higher moments. Since the arrays required are all identical to each other, the entire W.F.L. unit design is considerably simplified.

The weighting number expressions for the three higher moments are

$$W_{r,2} = r(r + 1)$$

$$W_{r,3} = \frac{r(r + 1)(2r + 1)}{2} \qquad (6.9)$$

$$W_{r,4} = r^2(r + 1)^2$$

The maximum value of r is 63 for a 6-bit quantisation process. It is seen that only multiplier arrays are required. The array described in section 6.5.1. is capable of realising the product LM+K where L, M and K are all n-bit numbers.

The weighting number $W_{r,2}$ can be realised with a single multiplier array in which L=M=K=r. A 6-bit array is required. To realise $W_{r,3}$, the expression r(r+1) for $W_{r,2}$ may be used. Thus let

$$R = r(r + 1)$$

$$\text{then} \quad W_{r,3} = \frac{R}{2}\{2r + 1\} \tag{6.10}$$

Division of R by 2 is a simple right shift of R by one bit and can be easily accomplished by wiring R into an array, with a right shift. Thus, $W_{r,3}$ is realised as

$$W_{r,3} = R \cdot r + \frac{R}{2} \quad .$$

The implementation of $W_{r,4}$ requires the realisation of R·R, where R = r(r + 1). It is possible to simplify a multiplier array and obtain a special squarer. However the general multiplier array can also be used. The design of the W.F.L. unit, using arrays is illustrated in Fig. 6.10. The arrays required are a 6-bit by 6-bit multiplier for $W_{r,2}$, a 6-bit by 12-bit array for $W_{r,3}$ and a 12-bit by 12-bit multiplier for $W_{r,4}$.

The total delay is obtained by calculating the delay in obtaining $W_{r,4}$. Thus

Delay to compute $W_{r,2}$ is

$$T(W_{r,2}) = (2n-1)\tau + 2\tau_g$$
$$= 13\tau$$

Assuming $\tau_g = \tau$

and $T(W_{r,4}) = 13\tau + (2 \times 12 - 1)\tau + 2\tau$ sec.

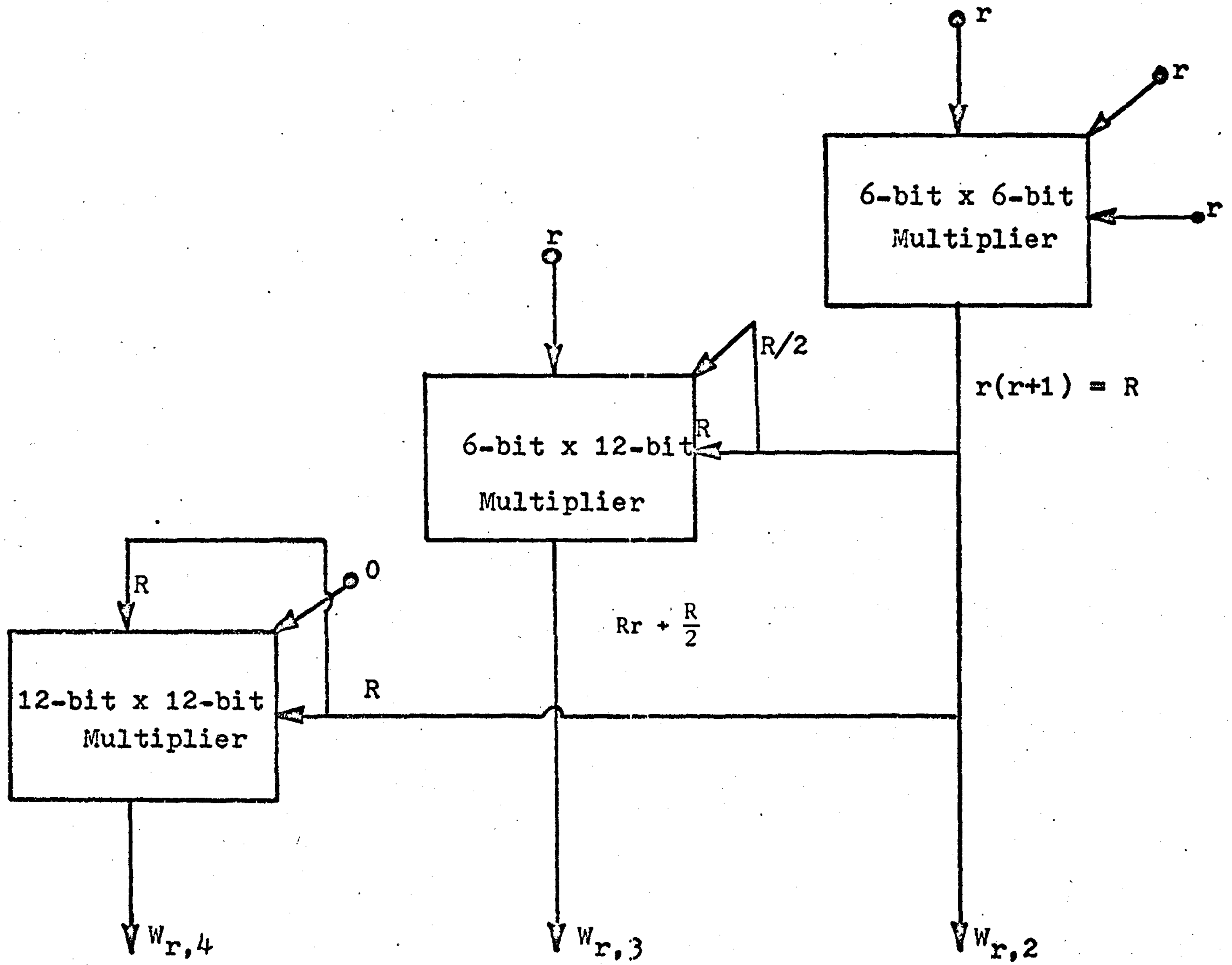


Figure 6.10 : Realisation of the weighting numbers.

$\therefore \quad T(W_{r,4}) = 38\tau$ sec.

Typically $\tau$=10 nsec. Hence the total delay is 380 ns., to realise all the weighting numbers.

In the special purpose computer designed, the first moment uses all eight bits of the a.d. converter, each bit being available at 1 μsec. interval. The other three moments use 6-bits. Therefore, the weighting numbers for these three moments would be available before the accumulation process which begins after the eight-bit conversion is complete.

## 6.8 Comments

The iterative arrays offer a convenient method for implementing the arithmetic operations which arise in special purpose computers. The simplification achieved by using such arrays is seen from Fig. 6.10. It is quite conceivable that in the near future, such arrays will be available as LSI functions. The universal arithmetic cell developed in this chapter is shown to have a versatility which was not available in the cells proposed earlier.

# CHAPTER VII

# CONCLUSIONS

The use of above-level probabilities results in simple algorithms for the moments of a random signal. A special purpose computer using such algorithms for the first four moments has been designed and constructed. Some tests have been carried out which indicate that the overall accuracy of the s.p.c. is within 1% for all four moments for commonly-encountered random and periodic signals.

The concept of the weighted feeds is simple but a powerful one since it enables fast processing without storage of the samples of the input signals. Simultaneous parallel multipliers have been used in the weighted feed logic units. Design of these multipliers makes an efficient use of the fact that the multiplier and multiplicand are interrelated. Logic minimisation of these units, therefore, results in economy.

Logic minimisation has received considerable attention in the past. However, with the reduction in cost and improved reliability of integrated circuit modules, any multiple output logic problem must be analysed carefully before attempting a minimisation. In many cases, cost of computer time becomes excessive compared to savings in component costs resulting from a minimisation. In the minimisation approach adopted in Chapter III, a compromise has been used in which a near-minimum logic solution is sought for the multiple output weighted feed logic within a reasonable computer-run time.

In the implementation of the s.p.c., the significant feature is the direct computation of the standard deviation ($\sigma$) from first and second moment. The counter-equation algorithm used in this system is simple and can be readily used in other direct computations (e.g., reciprocals, approximations to functions, etc.).

A comprehensive error analysis for the four moments has been carried out. The most important error sources are

(a) Approximations and quantisation of the input,

(b) Level offsets in rectifier and a.d. converter,

(c) Finite time of measurement,

(d) Finite sampling rate and finite aperture time of the a.d. converter.

Errors due to (c) and (d) can often be reduced in practice. Thus, depending on the frequency of the input, the time of measurement should be chosen such that an appropriate sample size is obtained. Use of a sample and hold circuit before a.d. conversion, would reduce finite aperture time effects and if the ratio f/p does not exceed $\frac{1}{20}$ , then the finite sampling rate errors are also negligible. Analysis of the other remaining errors shows that beyond six-bit quantisation for higher moments, the reduction in errors is small whereas the complexity of the W.F.L. units increases considerably.

Design of W.F.L. units, using the simultaneous parallel multipliers, although compatible with the presently available IC components, tends to be complex and time consuming. Furthermore, the design cannot be easily extended to a higher number of bits. Simplicity of design would be achieved by using iterative or near-iterative arithmetic arrays. The universal arithmetic cell developed for use in these arrays is extremely versatile; its use in other arithmetic operations has been demonstrated. The W.F.L. units for a s.p.c. using 6-bit quantisation would require only three multiplier arrays. With the current progress in LSI technology, it is possible that in the forseeable future such arrays will be available as LSI modules.

The use of such LSI arrays in arithmetic operations would also make the design of substandard digital instruments an attractive possibility which should be investigated further. In such designs it is apparent that methods of improving speed of operation should be sought. A possibility in this area is the design of data-dependent arrays in which some arithmetic cycles are eliminated depending on input data[66,67]. Special purpose computers using such arrays are also possible for other applications such as multivariate averaging and convolution, the latter being useful in digital filtering.

Another area for possible investigation is the use of the s.p.c. for the four moments for real time applications. An example is the use of the second moment algorithm to compute $\int_0^T v^2 dt$, v being an error voltage and obtain an optimum control strategy. Finally, the first four moments may also be used for approximation of the probability density functions to be used in non-linear filtering algorithms[2].

# <u>A P P E N D I C E S</u>

## APPENDIX A

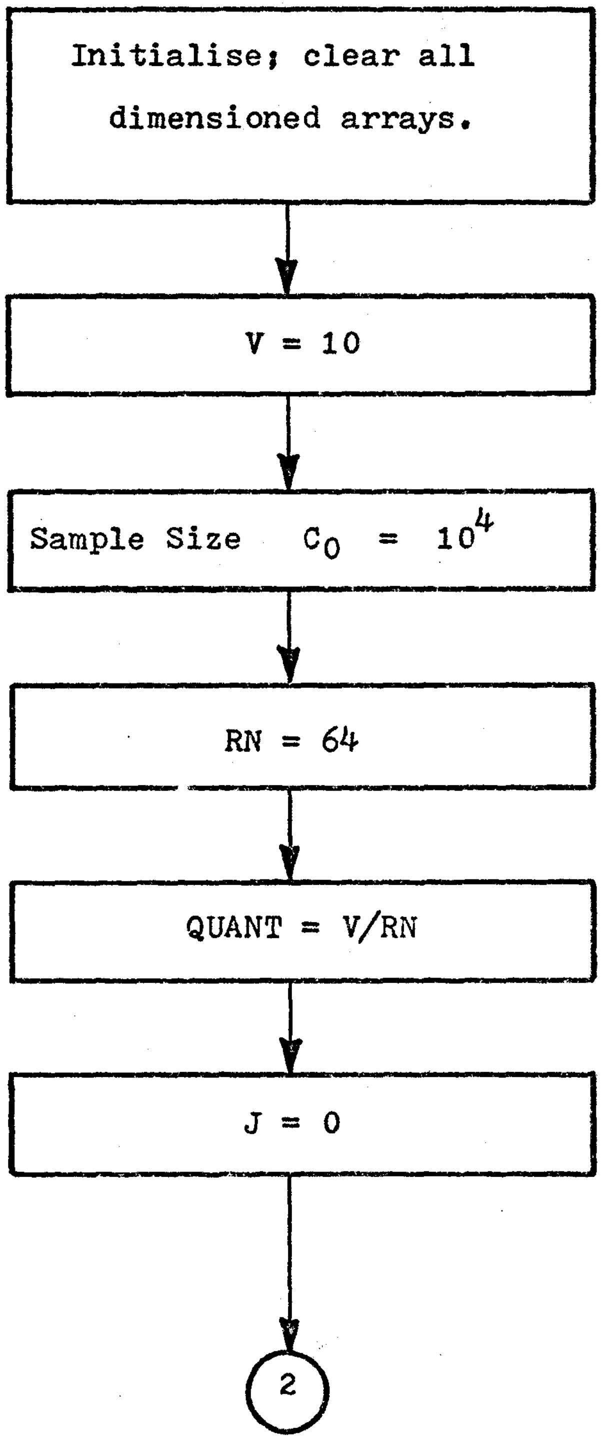

Initialise; clear all dimensioned arrays.
V = 10
Sample Size $C_0$ = $10^4$
RN = 64
QUANT = V/RN
J = 0
2

Figure A : Flow-chart procedure for evaluation of Direct Decimal Readouts for $C_0 = 10^4$

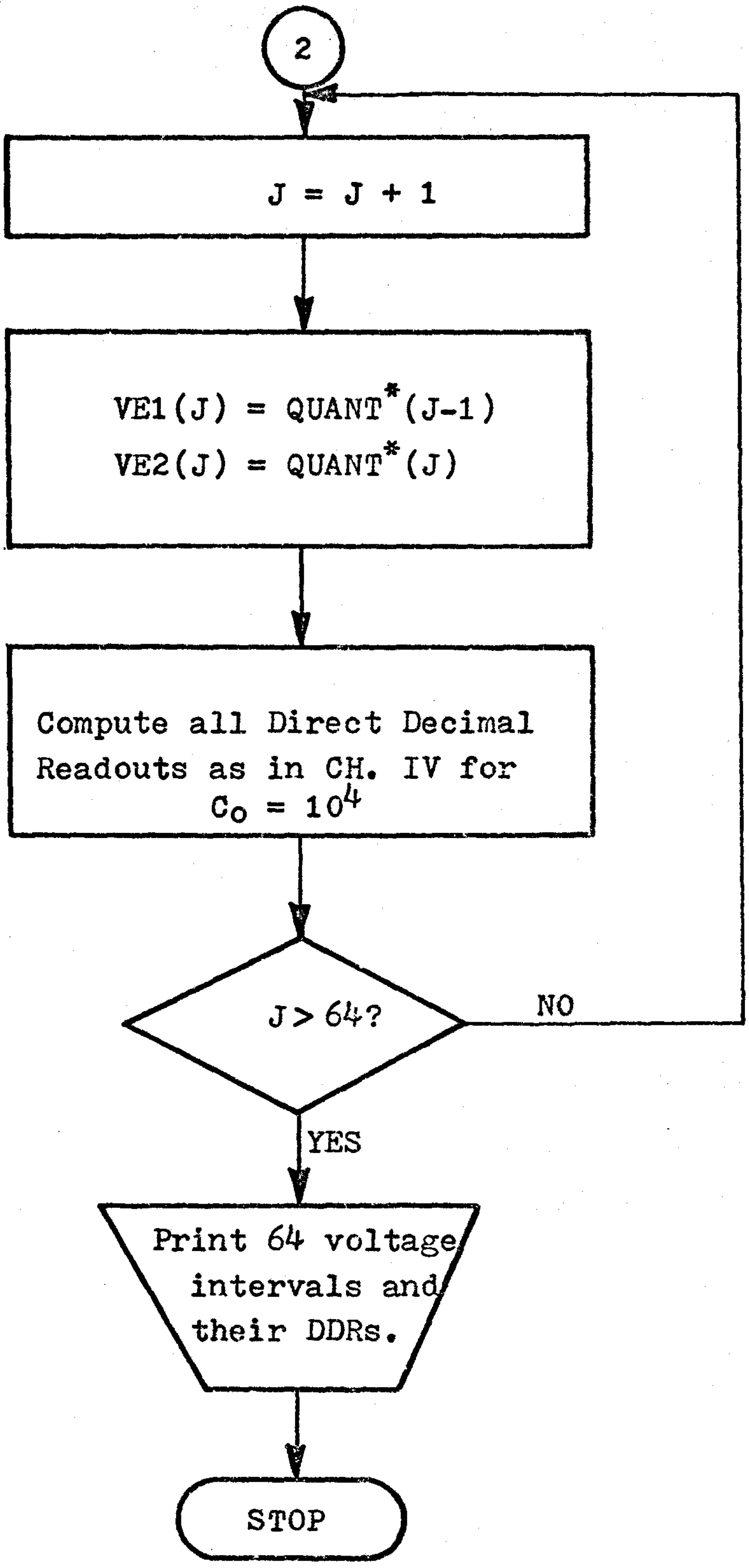

2
J = J + 1
VE1(J) = QUANT*(J-1)
VE2(J) = QUANT*(J)
Compute all Direct Decimal Readouts as in CH. IV for $C_o = 10^4$
J > 64?
NO
YES
Print 64 voltage intervals and their DDRs.
STOP

Figure A : Contd.

| VOLTAGE RANGE (VOLTS) | | FIRST MOMENT | SECOND MOMENT | THIRD MOMENT | FOURTH MOMENT |
|---|---|---|---|---|---|
| 0.0000 | 0.1563 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.1563 | 0.3125 | 156.25 | 4.88 | .12 | .00 |
| 0.3125 | 0.4688 | 312.50 | 14.65 | .59 | .02 |
| 0.4688 | 0.6250 | 468.75 | 29.30 | 1.63 | .09 |
| 0.6250 | 0.7813 | 625.00 | 48.83 | 3.47 | .24 |
| 0.7813 | 0.9375 | 781.25 | 73.24 | 6.34 | .55 |
| 0.9375 | 1.0938 | 937.50 | 102.54 | 10.47 | 1.06 |
| 1.0938 | 1.2500 | 1093.75 | 136.72 | 16.09 | 1.89 |
| 1.2500 | 1.4063 | 1250.00 | 175.78 | 23.42 | 3.11 |
| 1.4063 | 1.5625 | 1406.25 | 219.73 | 32.70 | 4.85 |
| 1.5625 | 1.7188 | 1562.50 | 268.55 | 44.16 | 7.24 |
| 1.7188 | 1.8750 | 1718.75 | 322.27 | 58.01 | 10.42 |
| 1.8750 | 2.0313 | 1875.00 | 380.86 | 74.50 | 14.55 |
| 2.0313 | 2.1875 | 2031.25 | 444.34 | 93.85 | 19.80 |
| 2.1875 | 2.3438 | 2187.50 | 512.70 | 116.29 | 26.35 |
| 2.3438 | 2.5000 | 2343.75 | 585.94 | 142.05 | 34.40 |
| 2.5000 | 2.6563 | 2500.00 | 664.06 | 171.36 | 44.18 |
| 2.6563 | 2.8125 | 2656.25 | 747.07 | 204.44 | 55.90 |
| 2.8125 | 2.9688 | 2812.50 | 834.96 | 241.53 | 69.82 |
| 2.9688 | 3.1250 | 2968.75 | 927.73 | 282.85 | 86.18 |

Table A: Direct Decimal Readouts for 37 four moments, n = 64, Co = $10^4$

| VOLTAGE RANGE (VOLTS) | | FIRST MOMENT | SECOND MOMENT | THIRD MOMENT | FOURTH MOMENT |
|---|---|---|---|---|---|
| 3.1250 | 3.2813 | 3125.00 | 1025.39 | 328.64 | 105.27 |
| 3.2813 | 3.4375 | 3281.25 | 1127.93 | 379.11 | 127.36 |
| 3.4375 | 3.5938 | 3437.50 | 1235.35 | 434.51 | 152.76 |
| 3.5938 | 3.7500 | 3593.75 | 1347.66 | 495.06 | 181.78 |
| 3.7500 | 3.9063 | 3750.00 | 1464.84 | 560.99 | 214.76 |
| 3.9063 | 4.0625 | 3906.25 | 1586.91 | 632.52 | 25202 |
| 4.0625 | 4.2188 | 4062.50 | 1713.87 | 709.90 | 293.94 |
| 4.2188 | 4.3750 | 4218.75 | 1845.70 | 793.33 | 340.89 |
| 4.3750 | 4.5313 | 4375.00 | 1982.42 | 883.06 | 393.24 |
| 4.5313 | 4.6875 | 4531.25 | 2124.02 | 979.32 | 451.41 |
| 4.6875 | 4.8438 | 4687.50 | 2270.51 | 1082.32 | 515.80 |
| 4.8438 | 5.0000 | 4843.75 | 2421.88 | 1192.31 | 586.84 |
| 5.0000 | 5.1563 | 5000.00 | 2578.13 | 1309.51 | 664.99 |
| 5.1563 | 5.3125 | 5156.25 | 2739.26 | 1434.14 | 750.69 |
| 5.3125 | 5.4688 | 5312.50 | 2905.27 | 1566.45 | 844.42 |
| 5.4688 | 5.6250 | 5468.75 | 3076.17 | 1706.65 | 946.66 |
| 5.6250 | 5.7813 | 5625.00 | 3251.95 | 1854.97 | 1057.92 |
| 5.7813 | 5.9375 | 5781.25 | 3432.62 | 2011.65 | 1178.71 |
| 5.9375 | 6.0938 | 5937.50 | 3618.16 | 2176.91 | 1309.55 |
| 6.0938 | 6.2500 | 6093.75 | 3808.59 | 2350.99 | 1451.00 |

Table A: Contd.

| VOLTAGE RANGE (VOLTS) | | FIRST MOMENT | SECOND MOMENT | THIRD MOMENT | FOURTH MOMENT |
|---|---|---|---|---|---|
| 6.2500 | 6.4063 | 6250.00 | 4003.91 | 2534.10 | 1603.62 |
| 6.4063 | 6.5625 | 6406.25 | 4204.10 | 2726.49 | 1767.96 |
| 6.5625 | 6.7188 | 6562.50 | 4409.18 | 2928.37 | 1944.62 |
| 6.7188 | 6.8750 | 6718.75 | 4619.14 | 3139.98 | 2134.21 |
| 6.8750 | 7.0313 | 6875.00 | 4833.98 | 3361.55 | 2337.33 |
| 7.0313 | 7.1875 | 7031.25 | 5053.71 | 3593.30 | 2554.62 |
| 7.1875 | 7 3438 | 7187.50 | 5278.32 | 3835.47 | 2786.71 |
| 7.3438 | 7.5000 | 7343.75 | 5507.81 | 4088.28 | 3034.27 |
| 7.5000 | 7.6563 | 7500.00 | 5742.19 | 4351.96 | 3297.97 |
| 7.6563 | 7.8125 | 7656.25 | 5981.45 | 4616.74 | 3578.50 |
| 7.8125 | 7.9688 | 7812.50 | 6225.59 | 4912.85 | 3876.55 |
| 7.9688 | 8.1250 | 7968.75 | 6474.61 | 5210.52 | 4192.85 |
| 8.1250 | 8.2813 | 8125.00 | 6728.52 | 5519.98 | 4528.11 |
| 8.2813 | 8.4375 | 8281.25 | 6987.30 | 5841.46 | 4883.10 |
| 8.4375 | 8.5938 | 8437.50 | 7250.98 | 6175.17 | 5258.55 |
| 8.5938 | 8.7500 | 8593.75 | 7519.53 | 6521.37 | 5655.25 |
| 8.7500 | 8.9063 | 8750.00 | 7792.97 | 6880.26 | 6073.99 |
| 8.9063 | 9.0625 | 8906.25 | 8071.29 | 7252.09 | 6515.56 |
| 9.0625 | 9.2188 | 9062.50 | 9354.49 | 7637.08 | 6980.77 |
| 9.2188 | 9.3750 | 9218.75 | 8642.58 | 8035.46 | 7470.47 |

Table A: Contd.

| VOLTAGE RANGE (VOLTS) | | FIRST MOMENT | SECOND MOMENT | THIRD MOMENT | FOURTH MOMENT |
|---|---|---|---|---|---|
| 9.3750 | 9.5313 | 9375.00 | 8935.55 | 8447.46 | 7985.49 |
| 9.5313 | 9.6875 | 9531.25 | 9233.40 | 8873.30 | 8526.69 |
| 9.6875 | 9.8438 | 9687.50 | 9536.13 | 9313.22 | 9094.95 |
| 9.8438 | 10.0000 | 9843.75 | 9843.75 | 9767.45 | 9691.14 |

Table A: Contd.

# APPENDIX B

## COMPUTATION OF SYSTEM QUANTISATION ERRORS

The following notes apply to the flow-charts for the computer programs, of Figure B.

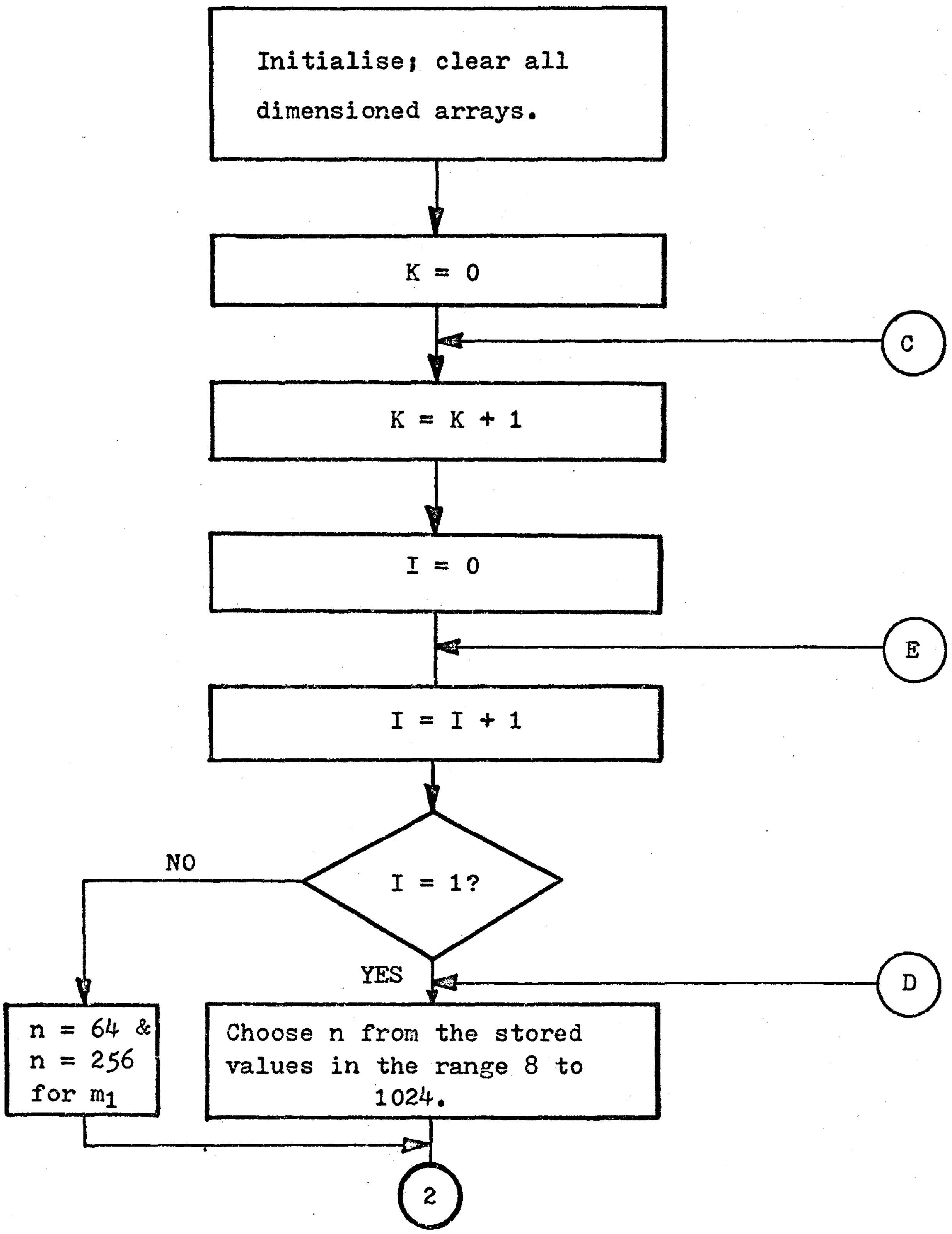
(a) Periodic Waveforms have been assigned numbers as follows.

| WAVEFORM | NUMBER |
|---|---|
| Sine Wave | 1 |
| FWR Sine Wave | 2 |
| Triangular Wave | 3 |
| FWR Triangular Wave | 4 |
| Rectangular Wave | 5 |
| FWR Rectangular Wave or DC Input | 6 |

(b) Two passes are made through the computation procedure.

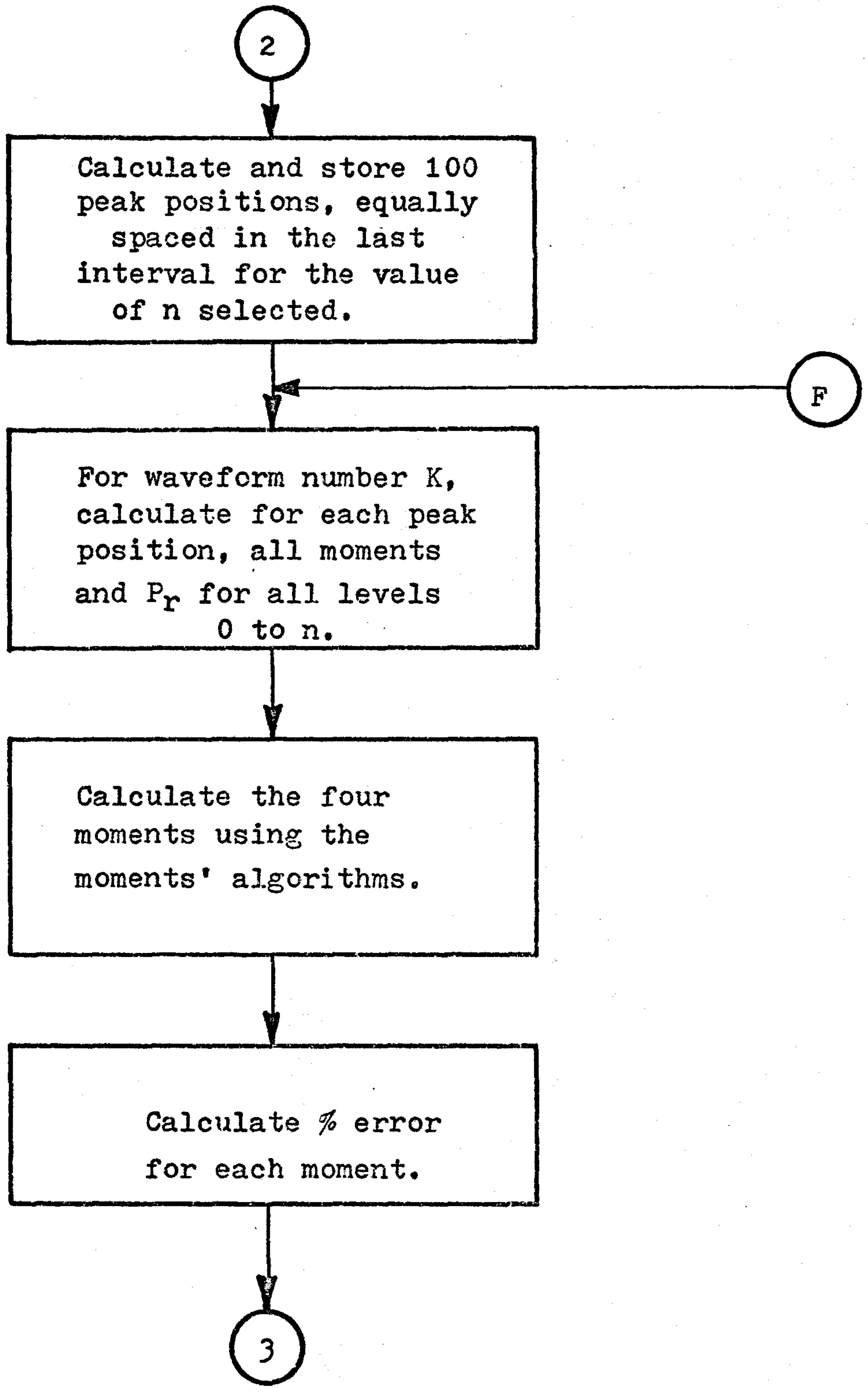
(i) First Pass: Peak of input waveform is varied in the level interval 255-256 for first moment and in the interval 63-64 for the other three moments. For each peak position, the quantisation error is calculated using the actual moments algorithms.

(ii) Second Pass: The number of quantisation levels n is varied in the range 8 to 1024. The peak position of input is varied in the interval (n-1) to n. The maximum and minimum errors are calculated for each value of n.



Initialise; clear all dimensioned arrays.
K = 0
C
K = K + 1
I = 0
E
I = I + 1
I = 1?
NO
YES
D
n = 64 & n = 256 for m1
Choose n from the stored values in the range 8 to 1024.
2

Figure B: Flow-chart procedure for calculation of system quantisation errors.



2

Calculate and store 100 peak positions, equally spaced in the last interval for the value of n selected.

F

For waveform number K, calculate for each peak position, all moments and $P_r$ for all levels 0 to n.

Calculate the four moments using the moments' algorithms.

Calculate % error for each moment.

3

Figure B: Contd.

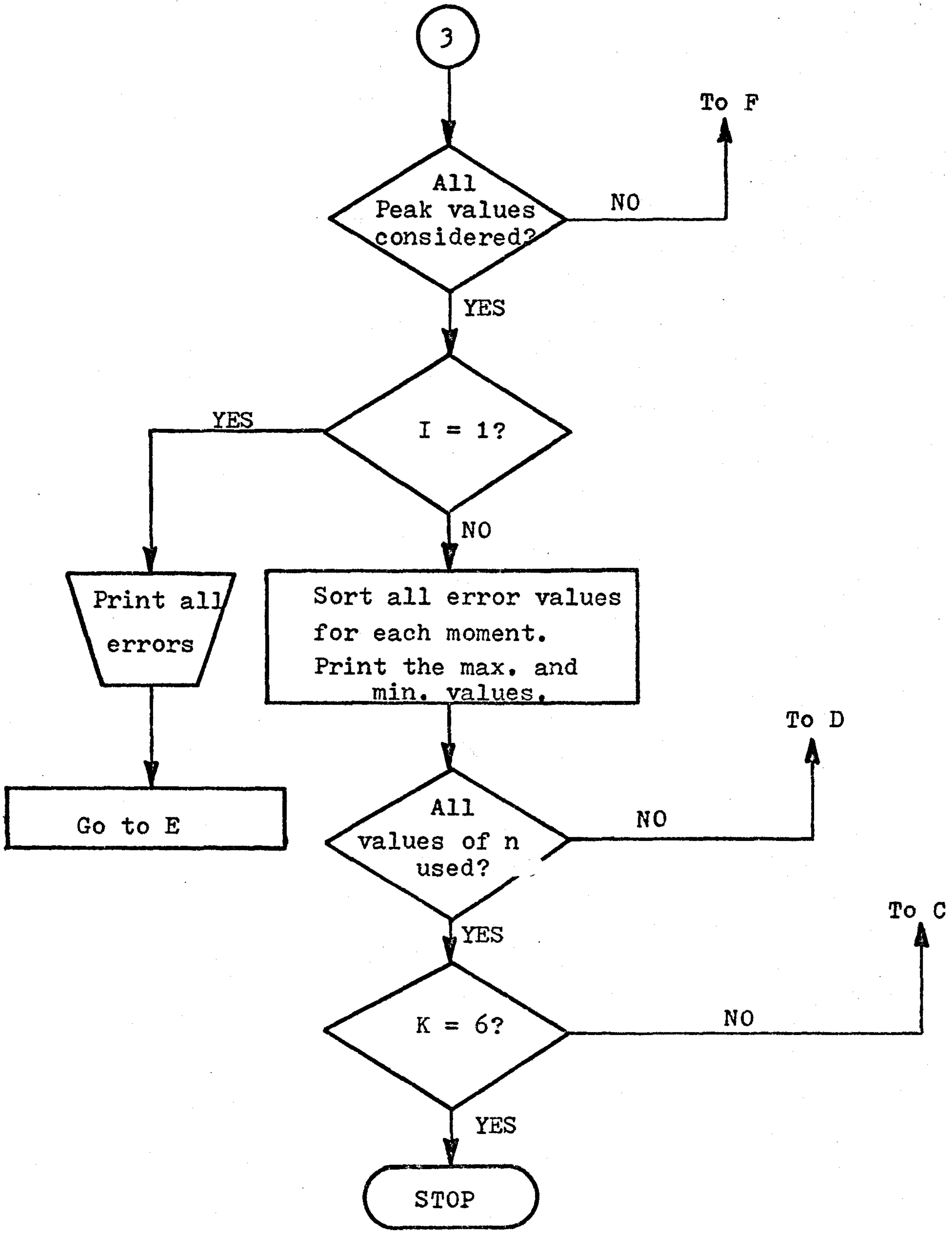
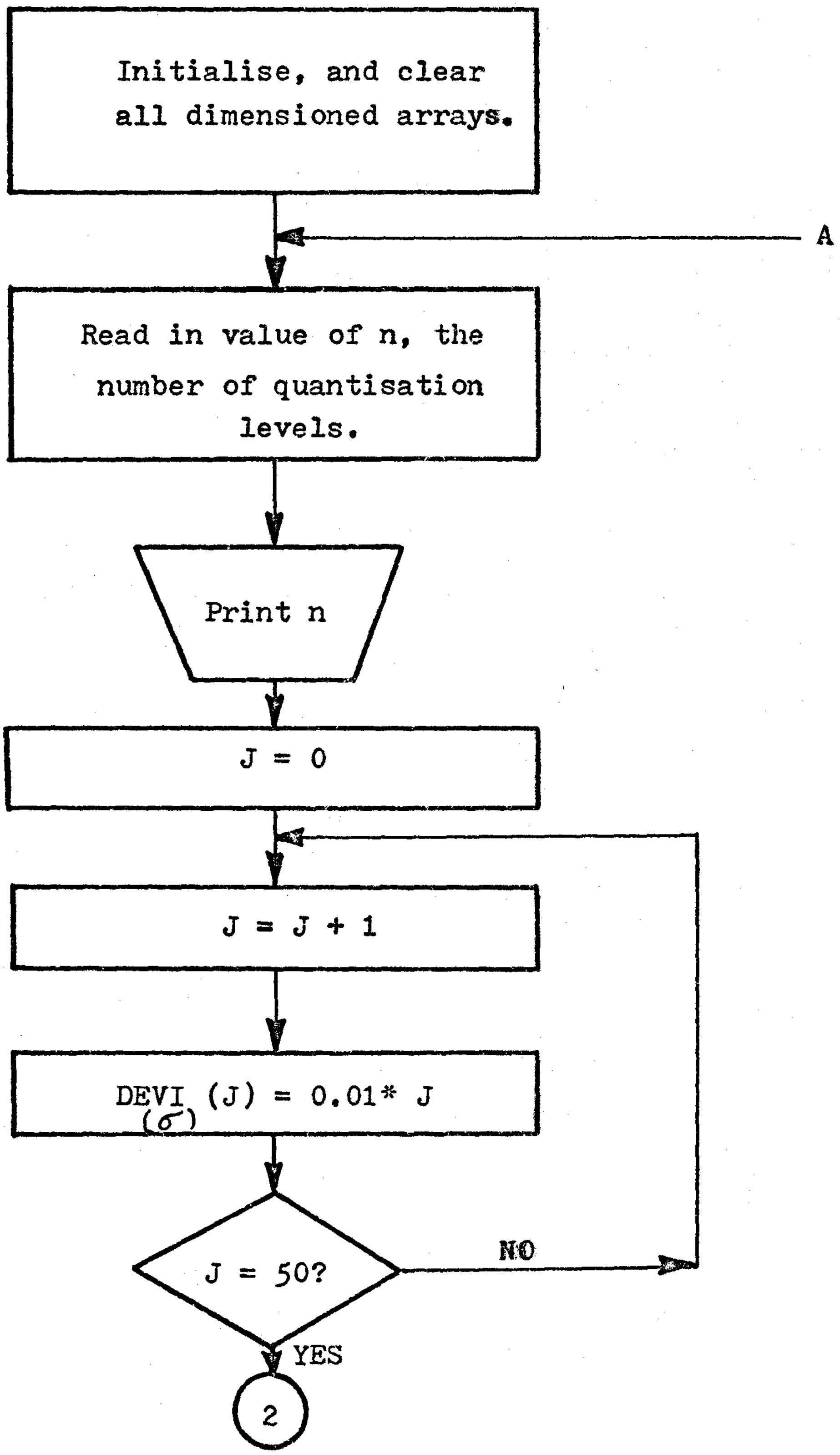

3
All Peak values considered?
NO
To F
YES
I = 1?
YES
NO
Print all errors
Sort all error values for each moment. Print the max. and min. values.
Go to E
All values of n used?
NO
To D
YES
K = 6?
NO
To C
YES
STOP

Figure B: Contd.

APPENDIX C.



Initialise, and clear all dimensioned arrays.
A
Read in value of n, the number of quantisation levels.
Print n
J = 0
J = J + 1
DEVI (J) = 0.01* J
(σ)
J = 50?
NO
YES
2

Figure C: Flow-chart procedure for calculation of quantisation errors for Gaussian Noise.

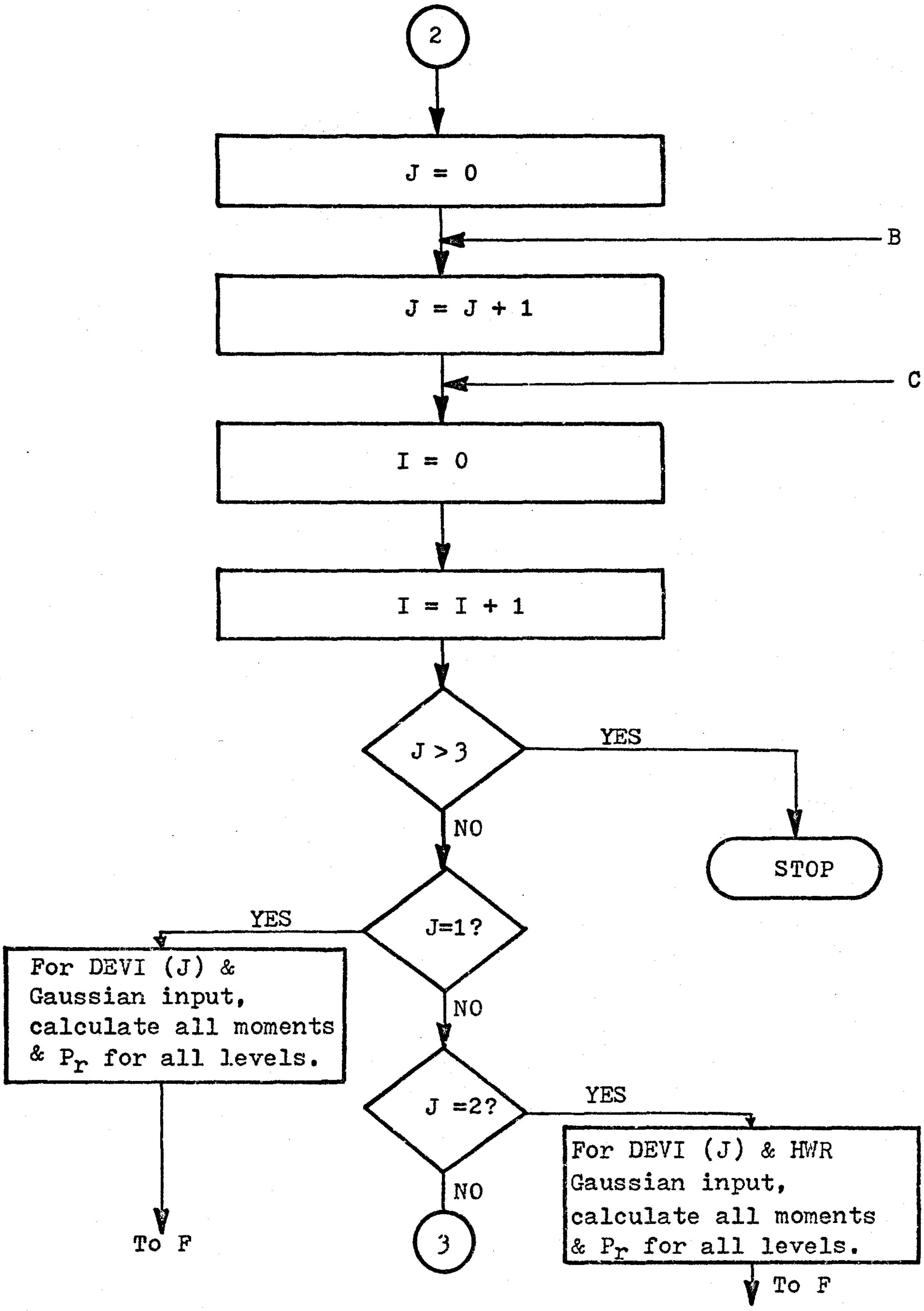

2
J = 0
B
J = J + 1
C
I = 0
I = I + 1
J > 3
YES
STOP
NO
J=1?
YES
For DEVI (J) & Gaussian input, calculate all moments & $P_r$ for all levels.
To F
NO
J =2?
YES
For DEVI (J) & HWR Gaussian input, calculate all moments & $P_r$ for all levels.
To F
NO
3

Figure C: Contd.

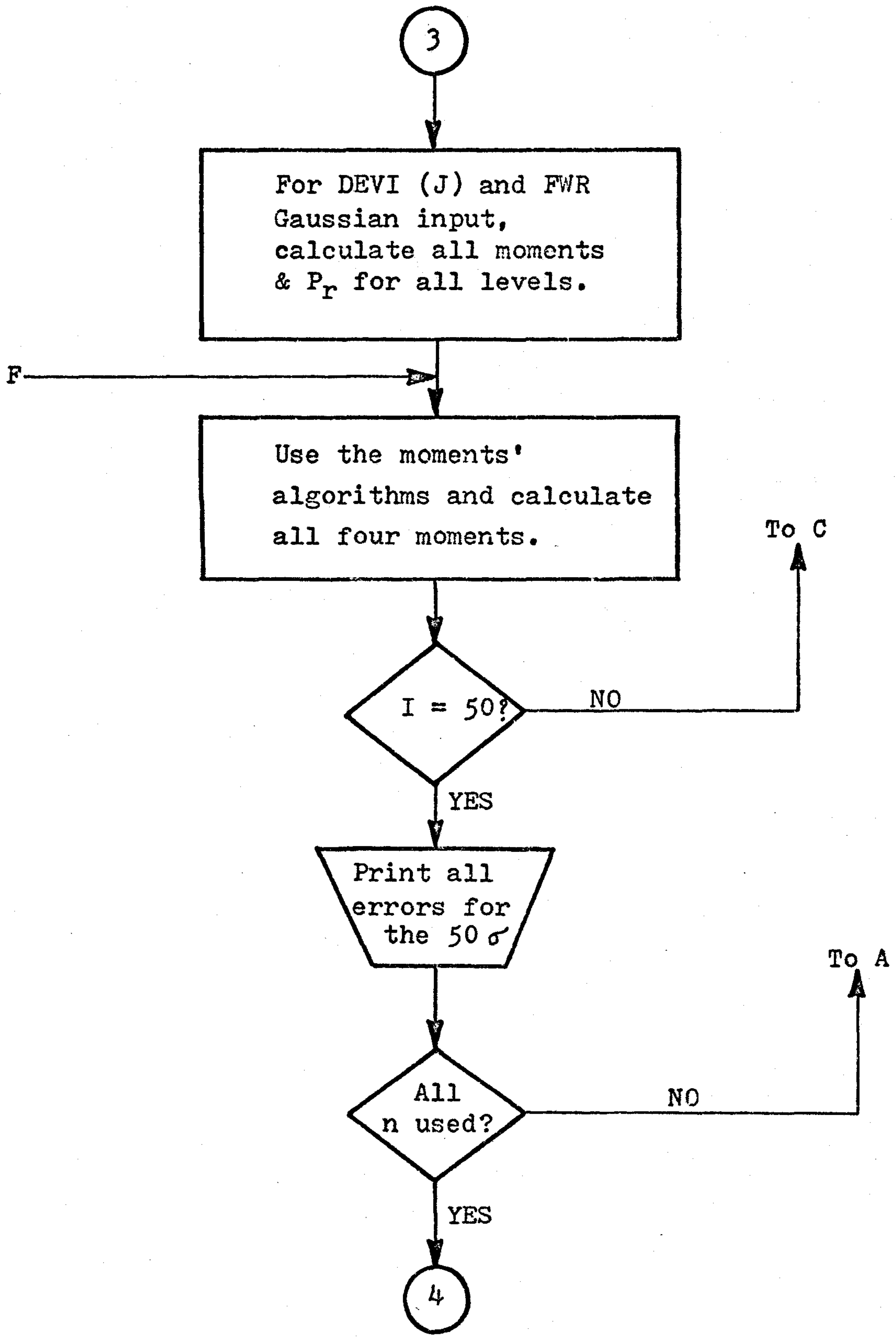

3
For DEVI (J) and FWR Gaussian input, calculate all moments & $P_r$ for all levels.
F
Use the moments' algorithms and calculate all four moments.
To C
I = 50?
NO
YES
Print all errors for the 50 $\sigma$
To A
All n used?
NO
YES
4

Figure C: Contd.

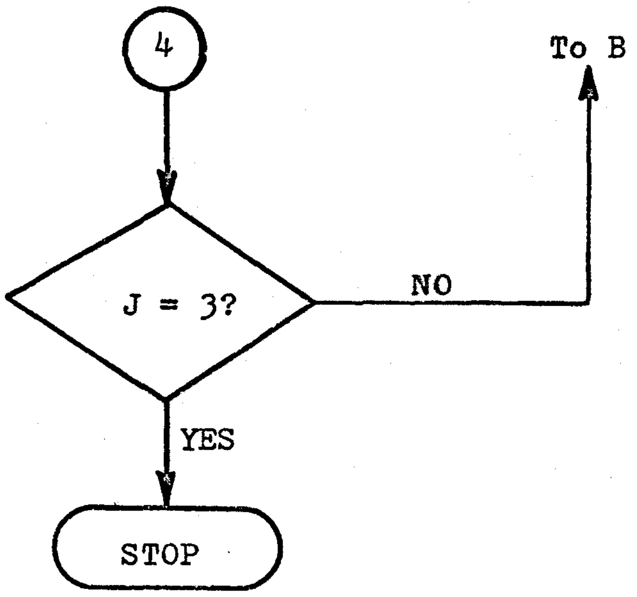


Figure C: Contd.

## APPENDIX D

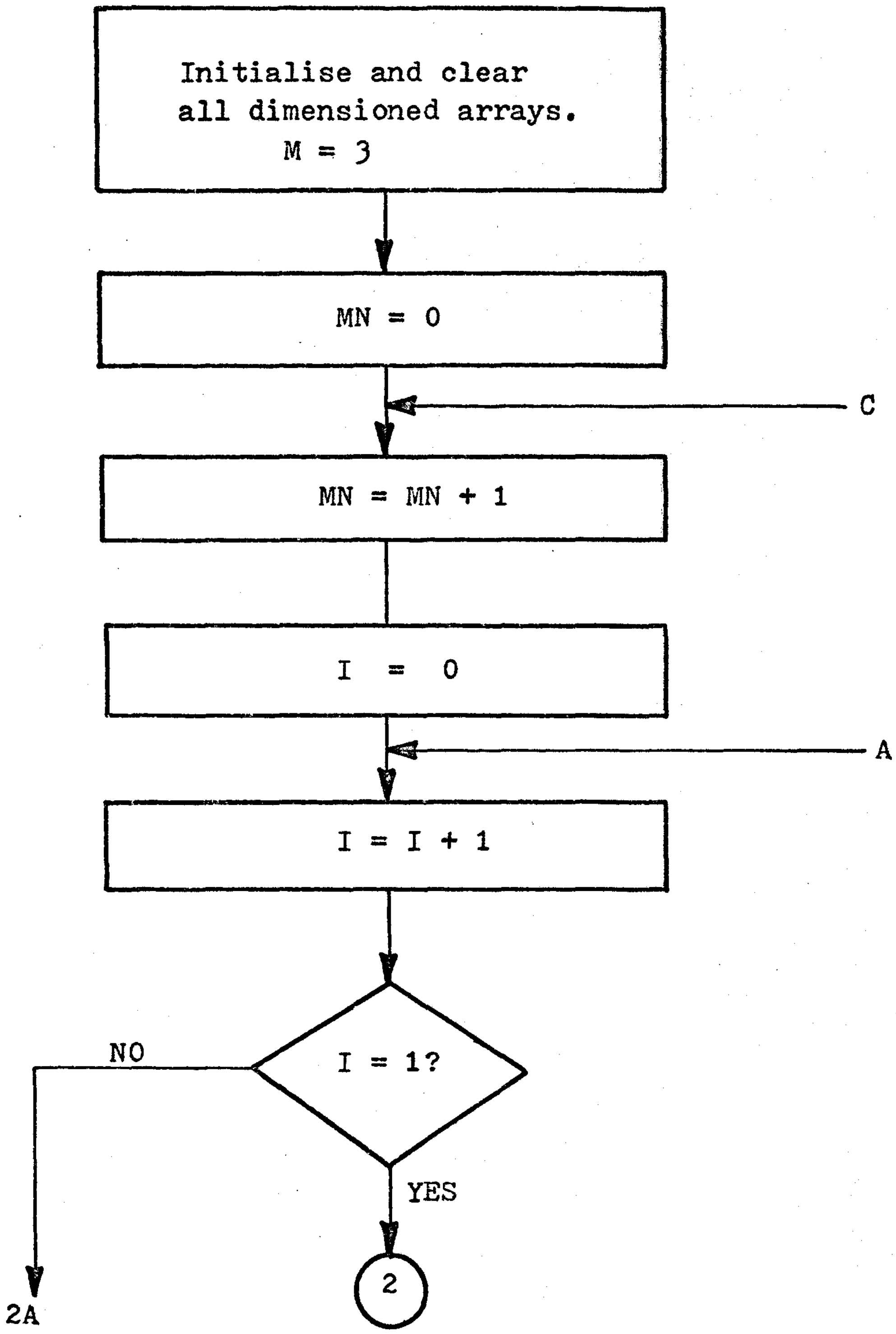


Figure D: Flow-chart procedure for evaluation of errors due to level offsets.

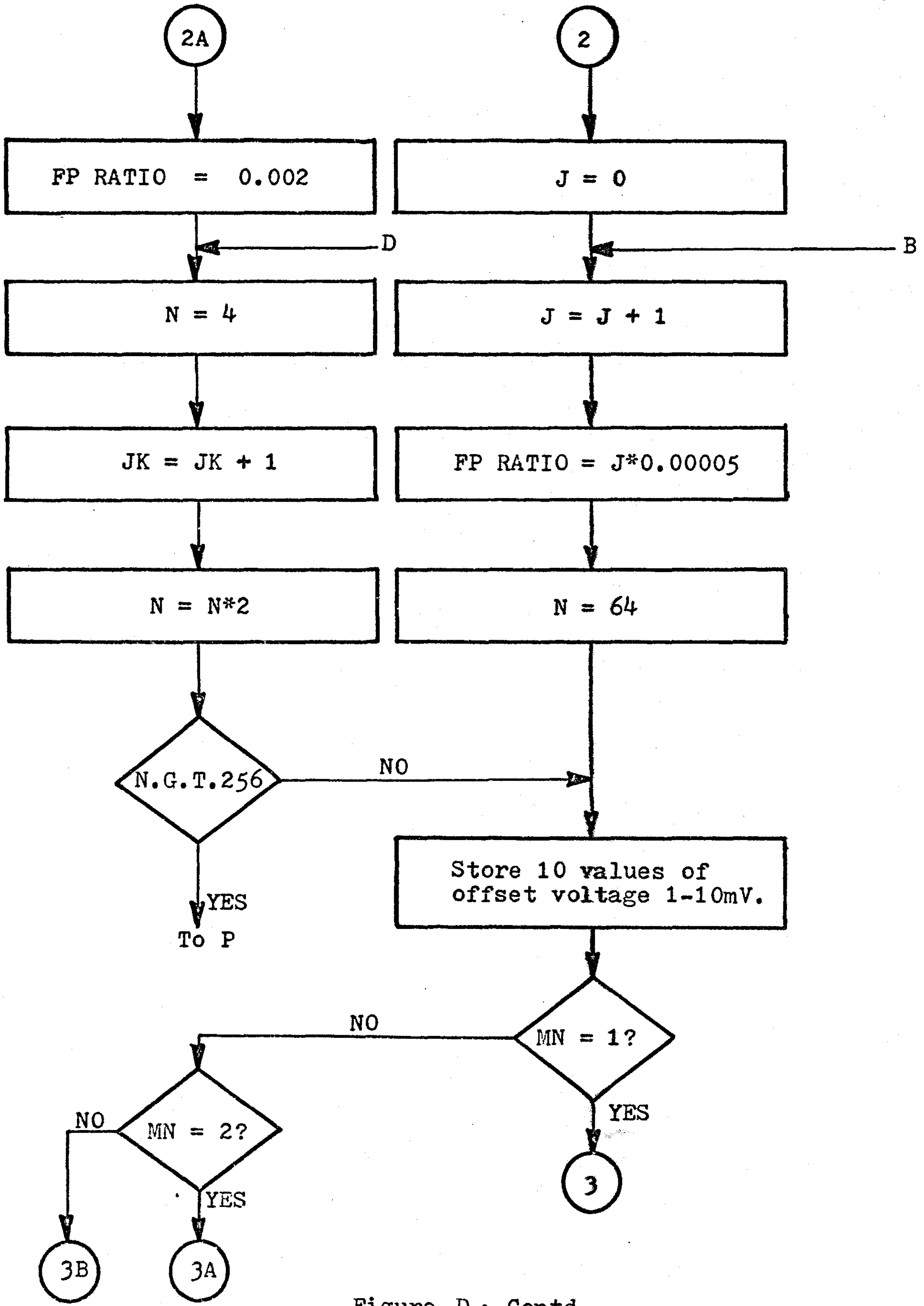


Figure D : Contd.

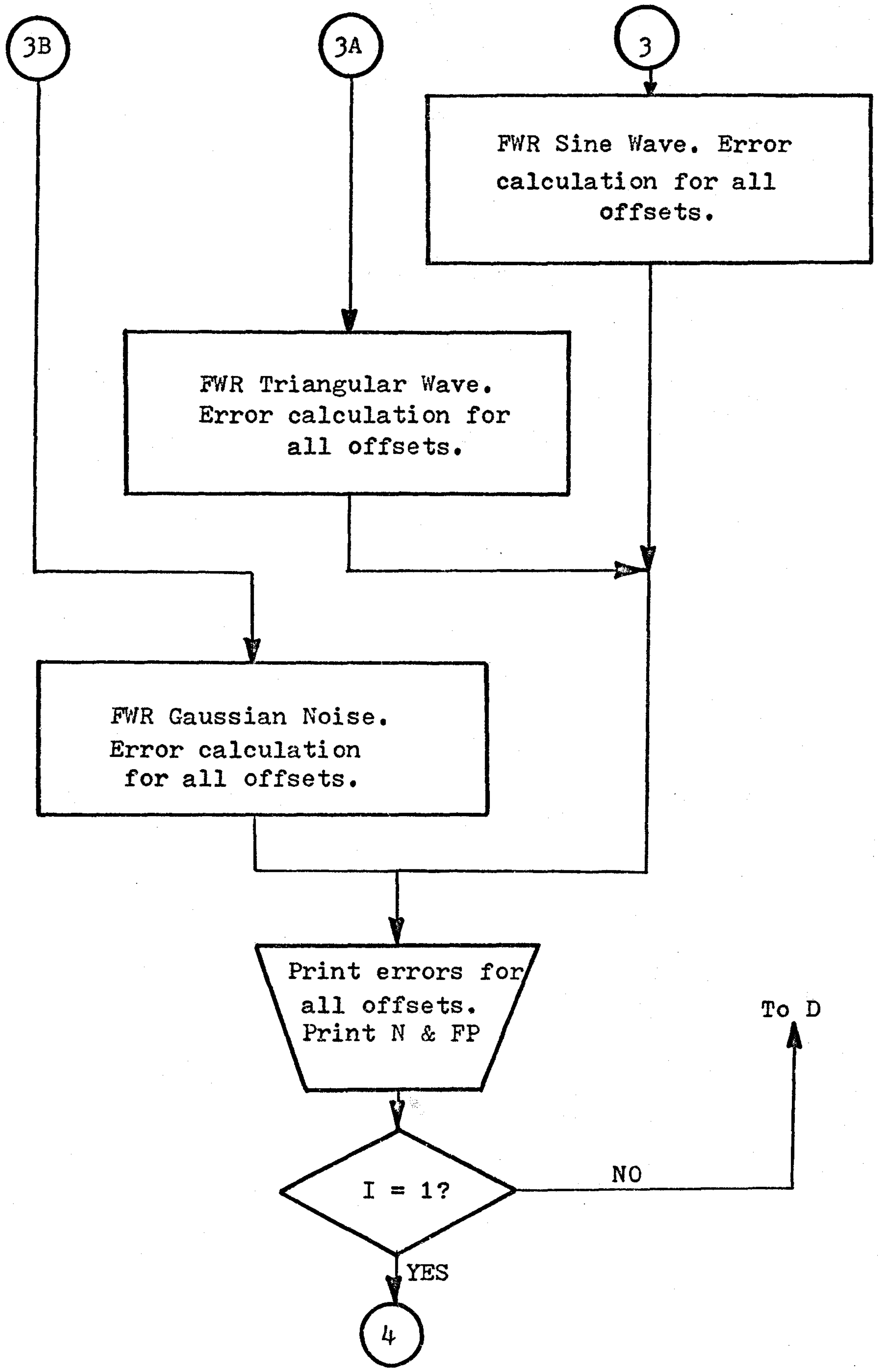

3B
3A
3
FWR Sine Wave. Error calculation for all offsets.
FWR Triangular Wave. Error calculation for all offsets.
FWR Gaussian Noise. Error calculation for all offsets.
Print errors for all offsets. Print N & FP
To D
I = 1?
NO
YES
4

Figure D: Contd.

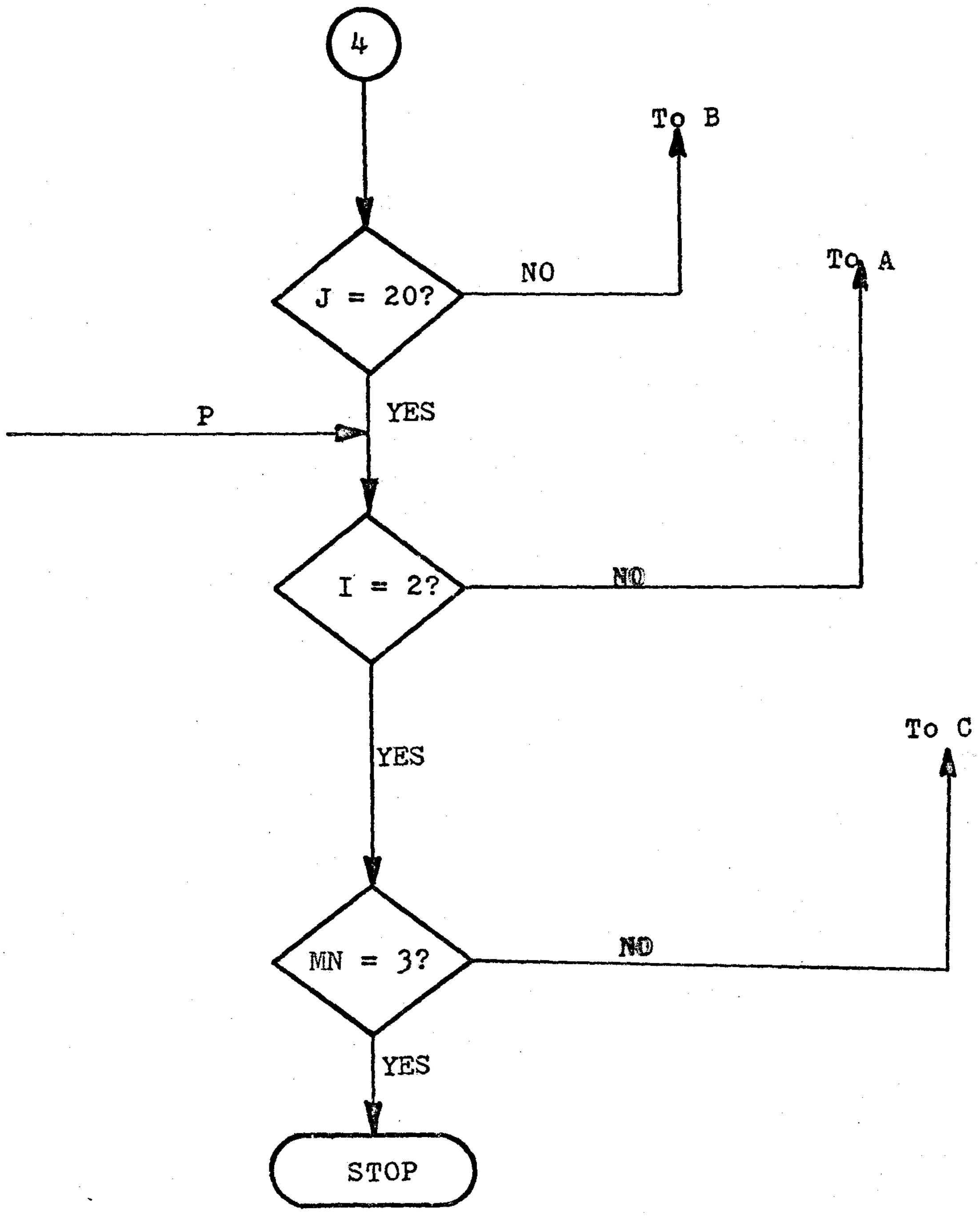


Figure D: Contd.

# APPENDIX E

## ERRORS IN HIGHER ORDER MOMENTS DUE TO FINITE MEASUREMENT TIME.

The following analysis has been suggested by Dr. S.O.Rice. of the Bell Telephone Laboratories, U.S.A.

Consider for example the forth moment of a random variable x measured over a finite time T.

Let

$$y = \int_0^T x^4(t)dt \qquad (1)$$

where x(t) is a stationary Gaussian process with two-sided power spectrum W(f) and auto-correlation R( τ ). Then

$$R(\tau) = \int_{-\infty}^{\infty} e^{+i2\pi f\tau} W(f)df$$

$$(2)$$

$$\sigma^2 = \int_{-\infty}^{\infty} W(f)df$$

The ensemble average $\langle y \rangle$ is

$$\langle y \rangle = \int_0^T \langle x^4(t) \rangle dt = \int_0^T (3\sigma^4)dt$$

$$(3)$$

$$= 3\sigma^4 T$$

The second moment of y is

$$\langle y^2\rangle = \int_0^T dt_1 \int_0^T dt_2 \langle x^4(t_1)x^4(t_2)\rangle$$

$$= \int_0^T dt_1 \int_0^T dt_2 \left[9\sigma^8+72\sigma^4R^2(t_1-t_2)+24R^4(t_1-t_2)\right]$$

$$= \left[\int_0^T dt_1(3\sigma^4)\right]^2 + \int_0^T dt_1 \int_0^T dt_2 \left[72\sigma^4R^2(t_1-t_2) + 24R^4(t_1-t_2)\right]$$

$$= [\langle y\rangle]^2 + 2\int_0^T d\tau\,(T-\tau)\left[72\sigma^4R^2(\tau) + 24R^4(\tau)\right]$$

The variance of y is

$$\langle y^2\rangle - \langle y\rangle^2 = 48\int_0^T (T-\tau)\left[3\sigma^4R^2(\tau) + R^4(\tau)\right]d\tau$$

For "flat" band-limited noise with

$$W(f) = \begin{cases}\sigma^2/(2f_c), & |f| < f_c \\ 0 \quad , & |f| > f_c\end{cases}$$

we have

$$R(\tau) = \int_{-f_c}^{f_c} e^{i2\pi f\tau}\ \frac{\sigma^2}{2f_c}\ df$$

$$= \frac{\sigma^2}{2\pi f_c\tau}\ \sin(2\pi f_c\tau)$$

and the integral for the variance of y can probably be evaluated in terms of Ci and Si functions.

The expression for $\langle x^4(t_1)x^4(t_2)\rangle$ can be obtained as the coefficient of $(iv)^4(iv)^4/(4!4!)$ in the expansion of

$$\langle e^{iux_1+ivx_2}\rangle = \exp\left[-\frac{\sigma^2}{2}(u^2+v^2) - R(t_1-t_2)uv\right]$$

When T becomes large we have

$$\frac{\langle y^2\rangle - \langle y\rangle^2}{\langle y\rangle^2} \approx \frac{48\ T}{9\sigma^8 T^2}\int_0^\infty \left[3\sigma^4R^2(\tau) + R^4(\tau)\right]d\tau$$

The integral of $[x(t)]^3$ can also be evaluated in the same way.

## REFERENCES

1. Hahn, G.J. and Shapiro, S.S.: "Statistical Models in Engineering", John Wiley & Sons, 1967, pp. 225-252.

2. Kuo, J. and Rowland, J.R.: "A Moment Technique for Suboptimal Adaptive Non-linear Filtering Algorithm", 1970 SWIEEECO Record of Tech. Papers. IEEE Cat. No. 70C5, pp. 105-109.

3. Pugachev, V.S.: "Theory of Random Functions and its Application to Control Problems", Pergamon Press, 1965, pp. 228-278.

4. Hahn, G.J. and Shapiro, S.S.: "Statistical Models in Engineering", John Wiley & Sons, 1967, pp. 220-224.

5. Wolf, A.A. and Dietz, J.H.: "A Method of Measurement and Display of Probability Functions of Ergodic Random Processes by Orthogonal Series Synthesis", Proc. I.R.E., No. 50, Dec. 1962, pp. 2503-2504.

6. Kama, Ya'coub.: "Application of a Quasi-peak Detector to the Measurement of Probability Density Functions", I.R.E. Trans. I-8, No. 1, March 1959, pp. 19-20.

7. Clarke, K.K.: "An Electronic Probability Density Machine", I.E.E.E. Trans. IM-15, 1966, pp. 25-28.

8. Caldwell, W.F., Korn, G.A., Latoree V.R. and Peterson, G.R.: "A Precision Amplitude Distribution Amplifier", IRE Trans. EC-9, June 1960, pp. 252-255.

9. Yang, E.S.: "A Probability Density Analyzer", I.E.E.E. Trans., IM-18, No. 1, March 1969, pp. 15-19.

10. Keist, D.N.: "An Analog System for the Analysis of Random Data Signals up to 10 Kc/s.", I.R.E. Trans. Instrumentation, vol. I-11, No. 2, Sept. 1962, pp. 52-57.

11. Mayorov, F.V.: "Digital Integrating Computers - Digital Differential Analyzers", Illiffe, 1964, pp. 1-49.

12. Cheney, P.W.: "A Digital Correlator Based on the Residue Number System", I.R.E. Trans. Elect. Computers, vol. EC-11, March 1961, pp. 63-70.

13. Garner, H.L.: "The Residue Number System", I.R.E. Trans., EC-8, June 1959, pp. 140-147.

14. Kitai, R. and Masuko, A.: "Digital Instrument for Measurement of Autocorrelation and Moments", Proc. I.E.E., vol. 116, No. 11, Nov. 1969, pp. 1950-1956.

15. Deist, F. and Kitai, R.: "Digital Transfer Voltmeters: Principles and Error Characteristics", Proc. I.E.E., vol. 110, No. 10, Oct. 1963, pp. 1887-1904.

16. Kitai, R. and Braithwaite, D.J.: "Digital Instrumentation for the Time Integral of the Square of a Slowly Fluctuating Voltage", I.E.E.E. Trans., IM-17, Sept. 1968, pp. 177-185.

17. Majithia, J.C.: "Digital Instrumentation for the Time Integral Squared of a Voltage and its Error Characteristics", M.Eng. Thesis, McMaster University, Hamilton, Ontario, 1968.

18. Hanrahan, H.E.: "The Principles and Error Characteristics of a Technique for Determining the Time Averages of Amplitude Quantised, Time Sampled Sequences", Ph.D. Thesis, University of the Witwatersrand, Johannesburg, South Africa, 1969.

19. Fry, T.C.: "Probability and its Engineering Uses", B. Van Nostrand Co., New York, 1928, pp. 310-312.

20. Widrow, B.: "A Study of Rough Amplitude Quantisation by Means of Nyquist Sampling Theory", I.E.E.E. Trans., CT-3, 1956, pp. 266-276.

21. Watts, D.G.: "A General Theory of Amplitude Quantisation with Application to Correlation Determination", Proc. I.E.E., vol. 109б, 1962, pp. 209-218.

22. Max, J.: "Quantising for Minimum Distortion", I.E.E.E. Trans., IT-6, pp. 7-12.

23. Wood, R.C.: "On Optimum Quantization", I.E.E.E. Trans., IT-15, pp. 48-52.

24. Renschler, E.: "A/D Conversion Techniques", Application Note AN471, Motorola Semiconductor Products, Inc.

25. Fisher, R.A.: "Statistical Methods for Research Workers", Oliver & Boyd, Edinburgh, 1948, pp. 73-76.

26. Papoulis, A.: "Probability, Random Variables and Stochastic Processes", McGraw-Hill, 1965, pp. 323-332.

27. Korn, G.A.: "Random-Process Simulation and Measurements", McGraw-Hill, New York, 1966, pp. 95-149.

28. DEC Logic Handbook: "Model A811 A.D. Converter", Digital Equipment Corp., Maynard, U.S.A., 1969, pp. 248-249.

29. Scott, N.R.: "Analog and Digital Computer Technology", McGraw-Hill-Kogakusha, International Students Edition, 1960, pp. 348-353.

30. Marcus, M.P.: "Switching Circuits for Engineers", Prentice-Hall, U.S.A., 1967, pp. 7-23.

31. Marcus, M.P.: "Switching Circuits for Engineers", Prentice-Hall, U.S.A., 1967, pp. 97-110.

32. Marcus, M.P.: "Switching Circuits for Engineers", Prentice-Hall, U.S.A., 1967, pp. 71-96.

33. McLuskey, E.J.: "Introduction to the Theory of Switching Circuits", McGraw-Hill, 1965, pp. 165-174.

34. Bartee, T.C.: "The Automatic Design of Logical Networks", Lincoln Lab. MIT Tech. Report, No. 191, Dec. 1958.

35. Bartee, T.C.: "Computer Design of Multiple Output Logical Networks", I.R.E. Trans., No. EC10, March 1961, pp. 21-30.

36. Pyne, I.B. and McCluskey, E.J., Jr.: "The Reduction of Redundancy in Solving Prime Implicant Tables", I.R.E. Trans., vol. EC-11, No. 1, Aug. 1962, pp. 473-482.

37. Gimpel, J.F.: "A Reduction Technique for Prime Implicant Tables", I.E.E.E. Trans., vol. EC-14, No. 4, Aug. 1965, pp. 535-541.

38. Schneider, P.R. and Dietmeyer, D.L.: "An Algorithm for Synthesis of Multiple Output Computer Logic", I.E.E.E. Trans., vol. EC-17, No. 2, Feb. 1968, pp. 117-128.

39. Ghazala, M.J.: "Irredundant Disjunctive and Conjunctive Forms of a Boolean Function", IBM Journal of Res. & Dev., vol. 1, No. 2, April 1955, pp. 171-176.

40. Burr-Brown Research Corp.: "Handbook of Operational Amplifier Applications", 1963.

41. Digital Equipment Corp.: "Positive Logic Handbook", 1969, pp. 238-249.

42. Kitai, R. and Majithia, J.C.: "Digital Analyzer for Statistical Moments: Design and Error Characteristics", Presented at I.E.E.E., 1971 Elect. and Electronic Meas. and Test Inst. Conference, Ottawa, Ontario, June 1-3, 1971.

43. Dean, K.J.: "Some Electronic Logic Circuits for Serial-Parallel Arithmetic", Radio & Electronic Engineer, Feb. 1969, pp. 95-98.

44. Cody, W.J.: "Double Precision $\sqrt{x}$ for the CDC 3600", Com. A.C.M., vol. 7, No. 12, Dec. 1964, pp. 715-718.

45. Kitai, R. and Majithia, J.C.: "Digital Square Root Computer for a Fluctuating Number", Electronic Engineering, To be published.

46. Phister, M.: "Logical Design of Digital Computers", John Wiley & Sons, New York, 1958, pp. 375-387.

47. Rice, S.O.: Private Correspondence, see Appendix E. of this thesis.

48. Rice, S.O.: "Mathematical Analysis of Random Noise", see Selected Papers on Noise and Stochastic Processes, Edited by Nelson Wax, Dover Press, 1954, pp. 133-294.

49. Hennie, F.C.: "Iterative Arrays of Logical Networks", M.I.T. Press, 1961, pp. 1-14.

50. Hoffman, J.C., Lacaze, B. and Csillag, P.: "Multiplieur Parallele a Circuits Logicques Iteratifs", Electronics Letters, 4, May 1968, pp. 178-179.

51. Dean, K.J.: "Logical Networks for Use in Iterative Arrays", Electronics Letters, vol. 4, No. 5, March 1968, pp. 81-82.

52. Dean, K.J.: "Versatile Multiplier Arrays", Electronics Letters, vol. 4, No. 16, Aug. 1968, pp. 333-334.

53. Dean, K.J.: "Design of a Full Multiplier", Proc. I.E.E., vol. 115, No. 11, Nov. 1968, pp. 1592-1594.

54. Dean, K.J.: "Cellular Logical Array for Extracting Square Roots", Electronics Letters, vol. 4, No. 15, July 1968, pp. 314-315.

55. Burton, D.P. and Noaks, D.R.: "High Speed Iterative Multiplier", Electronics Letters, vol. 4, No. 12, May 1968, p. 262.

56. De Mori, R.: "Suggestion for an I.C. Fast Parallel Multiplier", Electronics Letters, vol. 5, 1969, pp. 50-51.

57. Dean, K.J.: "Binary Division Using a Data-Dependent Iterative Array", Electronics Letters, vol. 4, No. 14, 1968, pp. 283-284.

58. Guild, H.H.: "Fully Iterative Fast Array for Binary Multiplication and Addition", Electronics Letters, vol. 5, No. 12, June 1969, pp. 253-

59. Guild, H.H.: "Cellular Logical Array for Nonrestoring Square-root Extraction", Electronics Letters, vol. 6, 1970, pp. 66-67.

60. Majithia, J.C.: "Non-restoring Binary Division Using a Cellular Array", Electronics Letters, vol. 6, 1970, pp. 303-304.

61. Ramamoorthy, C.V. and Economides, S.C.: "Fast Multiplication Cellular Arrays for LSI Implementation", Fall Joint Computer Conference AFIPS, 1969, pp. 89-98.

62. Majithia, J.C. and Kitai, R.: "An Iterative Array for Multiplication of Signed Binary Numbers", I.E.E.E. Trans. on Computers, To be published.

63. Majithia, J.C. and Kitai, R.: "Fast Multiplier/Divider Using a Controlled Iterative Array", I.E.E.E. Trans. on Computers, Submitted for publication.

64. Flores, I.: "The Logic of Computer Arithmetic", Prentice-Hall, 1963, pp. 249-253.

65. Booth, A.D.: "A Signed Binary Multiplication Technique", Quart. Jour. of Mech. and Applied Math, vol. IV, Pt. 2, 1951, pp. 236-240.

66. Walker, P.A.W.: "Asynchronous Binary Dividing Array", Electronics Letters, vol. 6, No. 16, Aug. 1970, pp. 515-517.

67. MacSorley, O.L.: "High Speed Arithmetic in Binary Computers", Proc. I.R.E., Jan. 1961, pp. 67-91.