BROWN, A.W. 1975. *Design of a digital stochastic computer*. Robert Gordon's Institute of Technology, MPhil thesis. Hosted on OpenAIR [online]. Available from: https://doi.org/10.48526/rgu-wt-1993296

Design of a digital stochastic computer.

BROWN, A.W.

1975

The author of this thesis retains the right to be identified as such on any occasion in which content from this thesis is referenced or re-used. The licence under which this thesis is distributed applies to the text and any original images only – re-use of any third-party content must still be cleared with the original copyright holder.

OpenAIR @RGU

This document was downloaded from https://openair.rgu.ac.uk

CC BY NC

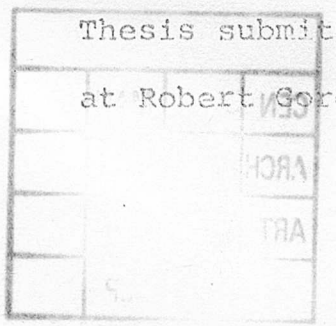
DESIGN OF A DIGITAL STOCHASTIC COMPUTER

Angus W Brown

September, 1975

Thesis submitted for the degree of M.Phil.

at Robert Gordon's Institute of Technology



ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr P Mars, for his invaluable help and constructive criticisms during the course of this project. I would also like to express my gratitude to fellow postgraduate students A J Miller and T Baxter for their consultations with regard to the ADDIE and the automatic patching system.

Finally my appreciation is due to Miss Pam Swanson for her patience and diligence in typing this thesis.

SUMMARY

This thesis describes the design and construction of an automatically patched, general purpose, digital stochastic computer. Also described are the automatic patching system and the software which has been written to operate the machine in conjunction with a PDP/8E minicomputer.

Some applications of the machine are investigated and suggestions as to further work are made.

CONTENTS

INTRODUCTION

In 1965, research teams working independently in the fields of pattern recognition and machine learning, suggested a new approach to digital computation.

They found that by using standard digital gates and counters, and using probability to represent the variables, they could build analogue type modules and combine them to form a stochastic computer.

In certain applications, such as process control, the amount of data to be processed simultaneously is so large that normal computation methods result in intolerable computation times. Digital computers may be very accurate, but are generally slow for on-line applications, because they use sequential operations for addition, subtraction, etc. Analogue machines are inaccurate when large systems are used, because of the build up of drift errors and offset errors in the amplifiers. The solution time however, is virtually instantaneous.

Both systems have their advantages and disadvantages. The digital stochastic computer, it is hoped, combines the advantages of both - the fast processing time of the analogue machine with the accuracy of the digital machine. It does this by using standard digital integrated circuits arranged to form modules of analogue type, such as summers, multipliers and integrators.

The stochastic computer does not represent its variables as a binary number as in a normal digital computer, nor as a voltage level as used in an analogue machine, but uses the probability that the voltage on a line will be a logic 1 or a logic 0 after any clock pulse.

A variable represented by this method will have a mean probability corresponding to its value, and a variance about this mean. By using parallel processing of many variables, /

variables, the stochastic computer makes computation time faster than the digital computer. This time, however, is slower than the analogue machine, since probability must be measured over a finite time for reasonable accuracy. A graph, showing the relative areas of use of analogue, stochastic and digital computers is shown in Figure (i).

The value of a variable represented as a stochastic sequence will also be less accurate than that of a digital computer using the same number of bits, but will be more accurate than an analogue computer.

The digital stochastic computer, DISCO, to be described later, (DISCO is an acronym for DIgital Stochastic COmputer) is run under the supervision of a PDP8/E minicomputer through standard digital input/output interfaces. All operations are controlled via a visual display unit consisting of an alphanumeric display on a cathode ray tube and a standard keyboard. The interconnection of the stochastic modules, the setting up of initial conditions on the integrators, and the reading of results, are all controlled from the V.D.U.

All the software written to control these operations, has been written in machine language, since no readily available programming language provided the flexibility and ease of communication that was required. All the machine language programs, to control DISCO, and the V.D.U., are included for reference in Appendix I.

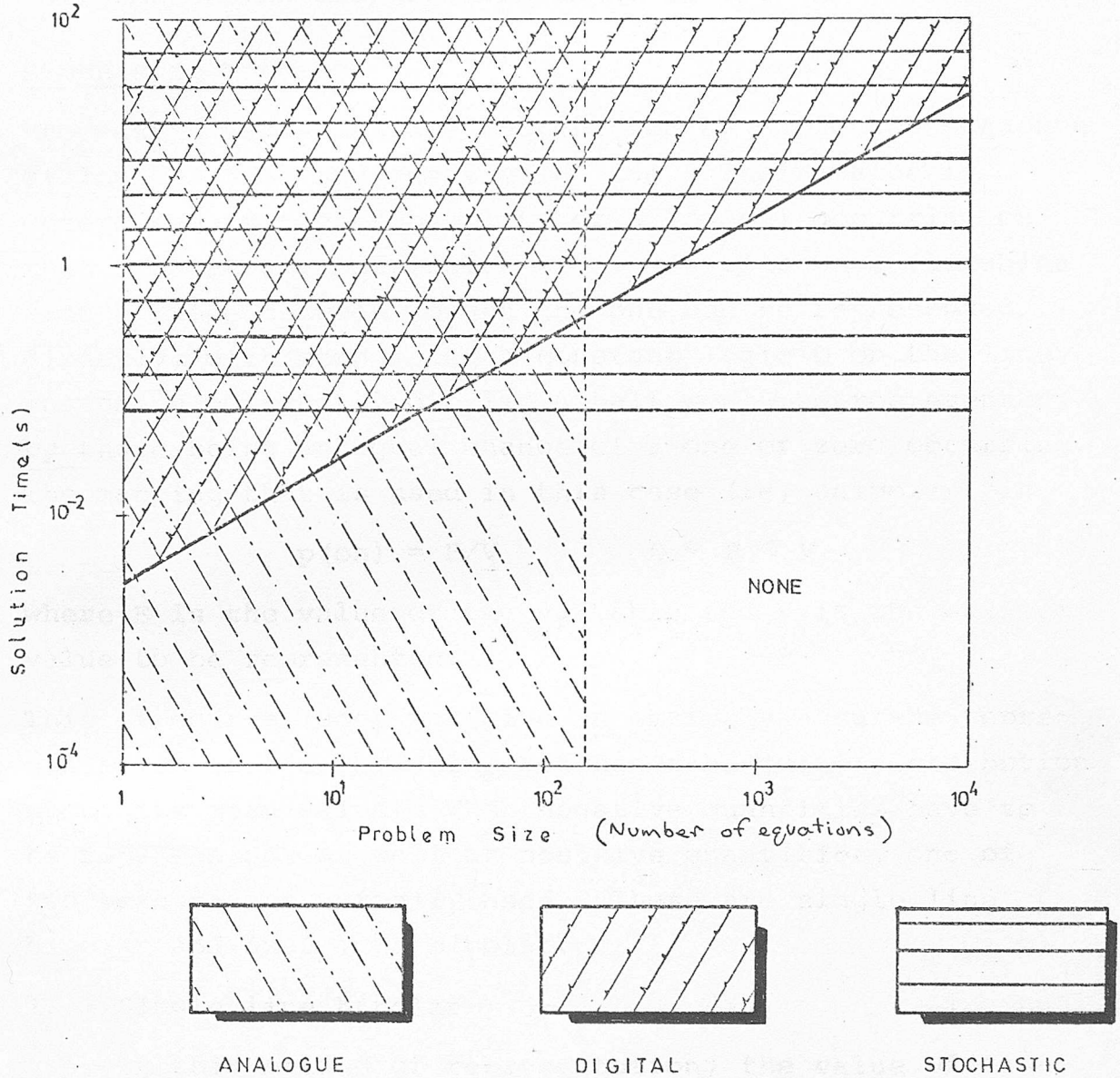


Figure (i) PROBLEM SIZE Vs. COMPUTING TIME

CHAPTER 1

REVIEW OF STOCHASTIC COMPUTATION

A stochastic computer can perform all the standard operations that can be accomplished by an analogue computer, such as integration, multiplication, etc. These operations are performed using standard TTL logic in simple configurations.

1.1 Stochastic Mappings[2,3,8,9]

Stochastic variables are represented by a clocked sequence of logical ones and zeros on a line. Their value is determined by the probability of a logic 1 occurring on that line at any particular time. In this way, variables with a value between nought and one can be represented directly, with zero being a constant logic 0 on the line, and one a constant logic 1. A half would be represented by there being an equal chance of a one or zero occurring. The mapping that is used in this case (ie, unipolar) is

$$p(\text{on}) = E/V \qquad 0 \leqslant E \leqslant V$$

where E is the value of the variable, and V is the maximum value to be represented.

This of course, does not give an entirely accurate representation of the variable, but has a binomial distribution about its mean value. When negative quantities have to be represented, as well as positive quantities, one of two methods are normally used. These are single line bipolar and dual line bipolar.

1. Single line bipolar

   In this method of representation, the value of the variable, whether positive or negative, is represented on a single line. This is done by using the mapping

   $$p(\text{on}) = \tfrac{1}{2} + \tfrac{1}{2}.\frac{E}{V}$$

   where p(on) is the probability that a one will occur, V is the maximum value that may be represented, and E is the variable to be represented.

A /

A variable may therefore lie in the range -V to +V. In the case when E = -V, the probability p(on) will always be zero. When E = +V, p(on) will always be one. In this representation, zero will be represented by

$$p(on) = \frac{1}{2} + \frac{1}{2}.\frac{E}{V} = \frac{1}{2} + \frac{1}{2}.\frac{0}{V} = \frac{1}{2}$$

2. Dual line bipolar

In this case, two lines are used to represent a variable, one for positive quantities and the other for negative quantities. If the quantity to be represented is positive, then

$$p(UP) = \frac{E}{V}$$

$$p(DOWN) = 0$$

and if the quantity is negative,

$$p(UP) = 0$$

$$p(DOWN) = \frac{E}{V}$$

In this representation, if the variable is zero, both the up and down lines will be at logic 0.

For the machine to be described, the single line bipolar mapping was adopted.[8] This mapping requires less hardware for implementation and thus kept down costs, and simplified construction.

The basic elements used in the stochastic computer, are the inverter, summer, multiplier, squarer, and integrator. The operation of these elements is explained below.

1.2 Inverter - Figure 1.1[a]

This element simply negates the input sequence to give the negative of the number. If the input probability is p then

$$p' = 1 - p$$

where p' is the output probability since p' + p = 1.

If /

If we substitute the mapping $p(on) = \frac{1}{2} + \frac{1}{2}.\frac{E}{V}$ then

$$\frac{1}{2} + \frac{1}{2}.\frac{E'}{V} = 1 - (\frac{1}{2}+\frac{1}{2}.\frac{E}{V})$$

$$\therefore \quad E' = -E$$

1.3 The Summer - Figure 1.2 (2,8)

This element has two inputs and one output, which is the normalised sum of the two inputs. The inputs are effectively switched through to the output by a stochastic sequence with the probability of $\frac{1}{2}$. The result is:-

$$p(out) = \frac{1}{2}p(A) + \frac{1}{2}p(B)$$

Substituting the single line bipolar mapping:-

$$\frac{1}{2} + \frac{1}{2}.\frac{E_O}{V} = \frac{1}{2}(1 + \frac{1}{2}.\frac{E_A}{V} + \frac{1}{2}.\frac{E_B}{V})$$

$$\therefore \quad \frac{E_O}{V} = \frac{1}{2}.\frac{E_A}{V} + \frac{1}{2}.\frac{E_B}{V}$$

$$\therefore \quad E_O = \frac{1}{2}.(E_A + E_B)$$

The factor of $\frac{1}{2}$ allows for the condition that the inputs may be greater than $\frac{V}{2}$ which would result in the output being meaningless.

1.4 The Multiplier - Figure 1.3 (2)

This element also has two inputs and provides a normalised output. The circuit is effectively an inverted exclusive-OR. The output is a logic 1 if the two inputs are either both ones or zeros.

$$p(out) = p(A).p(B) + (1 - p(A)).(1 - p(B))$$

Substituting:-/

Substituting:-

$$\frac{1}{2} + \frac{1}{2}.\frac{E_O}{V} = (\frac{1}{2}+\frac{1}{2}.\frac{E_A}{V}).(\frac{1}{2}+\frac{1}{2}.\frac{E_B}{V}) + (\frac{1}{2}-\frac{1}{2}.\frac{E_A}{V}).(\frac{1}{2}-\frac{1}{2}.\frac{E_B}{V})$$

$$\therefore \quad \frac{1}{2}.\frac{E_O}{V} = \frac{1}{2}.\frac{E_A.E_B}{V.V}$$

$$\therefore \quad E_O = \frac{E_A.E_B}{V}$$

The output is normalised by the division of V.

1.5 The Squarer - Figure 1.4 (3,8)

It is not possible to obtain the square of a number by simply connecting the two inputs of a multiplier together. This would result in the output always being at logic 1. It is necessary that the input sequences to a stochastic element be completely independent to achieve unbiased results. In this case, however, all that is required is to delay the sequence by one clock pulse, and use the resulting sequence, and the original sequence as the inputs to a multiplier. The squarer then becomes an element with only one input and one output.

As will be explained subsequently, due to modifications made to the PRBS generators in DISCO, this method of squaring is not possible.

1.6 The Integrator - Figures 1.5, 1.6 (2,8)

Integration is performed by using twelve bit binary up/down counters, since it is necessary to integrate negative as well as positive quantities. Figure 1.5 shows a simple integrator whose input is connected to the up line, and the inverse connected to the down line. Therefore, if a one is received, it counts up, and if a zero is received, it counts down. It cannot stay in the same state for two successive clock pulses.

If /

If we assume that the counter has N+1 states in the range -1 to +1, and that the input sequence is stochastic and conforms to the mapping

$$p(on) = \frac{1}{2} + \frac{1}{2}.\frac{E}{V}$$

then:-

$$p(UP) = \frac{1}{2} + \frac{1}{2}.\frac{E}{V}$$

$$p(DOWN) = \frac{1}{2} - \frac{1}{2}.\frac{E}{V}$$

Let us say that the counter starts in some arbitrary state $\Pi(O)$, then after m clock pulses, the expected output will be:

$$\Pi(m) = \Pi(O) + \sum_{n=O}^{m} (p(UP)_n - p(DOWN)_n).x$$

where x represents the value of one change of state of the counter.

$$\therefore \quad \Pi(m) = \Pi(O) + \sum_{n=O}^{m} ((\frac{1}{2}+\frac{1}{2}.\frac{E}{V}) - (\frac{1}{2}-\frac{1}{2}.\frac{E}{V})).x$$

$$\therefore \quad \Pi(m) = \Pi(O) + \sum_{n=O}^{m} (\frac{E}{V}).x$$

But $x = \frac{2}{N}$ so

$$\Pi(m) = \Pi(O) + \frac{2}{N} \sum_{n=O}^{m} \frac{E}{V}$$

$$= \Pi(O) + \frac{2}{N.T} \sum_{n=O}^{m} \frac{E.T}{V}$$

where T is the time for one clock period. The clock period is high, in the order of 1MHz, so that $T = 1/10^6 = 10^{-6}$s, and so we can say $T \rightarrow dt$.

$$\therefore \quad \Pi(t) \doteq \Pi(O) + \frac{2}{N.T} \int_O^t \frac{E(t)}{V}\, dt$$

$$\therefore \quad V \times \Pi(t) = V \times \Pi(O) + \frac{2}{N.T} \int_O^t E(t)\, dt$$

$$\therefore \quad E_{out} = E(O) + \frac{2}{N.T} \int_O^t E(t)\, dt \qquad \text{---------- (1.1)}$$

We /

We now see that the expected value of the output from the integrator is the starting value plus the time integral of the input, multiplied by a factor which is proportional to the clock frequency, and inversely proportional to the number of states in the integrator minus one.

A modification to the integrator can be made, Figure 1.6, to make it a two input device. These two inputs are summed and then integrated as can be shown:-

$$\Pi(m) = \Pi(0) + \sum_{n=0}^{m} (p(UP)_n - p(DOWN)_n).x$$

$$\therefore \Pi(m) = \Pi(0) + \sum_{n=0}^{m} (p(A).p(B) - p(\bar{A}).p(\bar{B})).x$$

$$\therefore \Pi(m) = \Pi(0) + \sum_{n=0}^{m} ((\tfrac{1}{2}+\tfrac{1}{2}.\tfrac{E_A}{V}).(\tfrac{1}{2}+\tfrac{1}{2}.\tfrac{E_B}{V}) - (\tfrac{1}{2}-\tfrac{1}{2}.\tfrac{E_A}{V}).(\tfrac{1}{2}-\tfrac{1}{2}.\tfrac{E_B}{V})).x$$

$$\therefore \Pi(m) = \Pi(0) + \sum_{n=0}^{m} \tfrac{1}{2}(\tfrac{E_A}{V} + \tfrac{E_B}{V})x$$

As for the previous configuration:

$$\Pi(t) = \Pi(0) + \frac{1}{NT}\int_0^t (\frac{E_A}{V} + \frac{E_B}{V})\,dt$$

$$\therefore E_{out} = E_O + \frac{1}{NT}\int_0^t (E_A + E_B)\,dt \qquad \text{---------- (1.2)}$$

As the above shows, it is very easy to add, multiply, etc. using stochastic elements. The variable inputs, which may be available in either digital or analogue form, must now be converted into stochastic sequences.

Conversion from analogue to stochastic may be done using the circuit in Figure 1.7.[8] This involves the use of what is called a true noise source, such as a microplasma diode. This is compared with the analogue signal in a comparator which will give a logic 1 out, if the analogue voltage is greater than the true noise source voltage, and /

and a logic 0 out, if it is not. This method has the advantage that all sequences generated will be completely independent.

Digital to stochastic conversion may be performed in one of two ways. A digital to analogue converter may be used to provide an analogue signal, and then used in the above method (Figure 1.8), or the digital number may be compared directly with a pseudo random number in a digital comparator (Figure 1.9).[2,8] This works in the same way as in the analogue comparator in that it produces a logic 1 if the digital number is greater than the pseudo random number.

The operations in the computer are performed using these sequences, and therefore there must be a conversion of the outputs to digital or analogue form. These outputs take the form of a sequence of ones and zeros. To convert these sequences to an accurate probability, it would be necessary to count the number of ones occurring over a long interval of time, and the solution would be the number of ones that occurred, divided by the number of clock pulses in that time. This would be a satisfactory solution for sequences that remain constant, but would still involve a considerable length of time. For sequences whose probability is changing with time, this method would not be feasible.

The output can be regarded as a steady probability with an inherent variance, that corresponds to noise on an analogue signal. A low pass filter is therefore required. The element used to do this is called the ADDIE (an acronym for ADaptive DIgital Element).[3,8,15] The circuit is shown in Figure 1.10. This is, in effect, a two input summing integrator with negative feedback to give an exponential average of the input sequence. This can be seen from the following analysis, assuming a simple analogue model.

$E_O(t)$ /

$$E_O(t) = E_O(0) + \frac{1}{NT} \int_0^t (E(\tau) - E_O(\tau))\,d\tau$$

$$\therefore \dot{E}_O(t) = \frac{1}{NT} (E(t) - E_O(t))$$

Taking Laplace transforms:-

$$sE_O(s) = \frac{1}{NT} (E(s) - E_O(s))$$

$$\therefore (s+\frac{1}{NT})\, E_O(s) = \frac{1}{NT} E(s)$$

$$\therefore E_O(s) = \frac{\frac{1}{NT} E(s)}{(s + \frac{1}{NT})}$$

If we now consider the response to a step input E.

$$E_O(s) = \frac{\frac{1}{NT} E}{s(s + \frac{1}{NT})} \qquad E(s) = \frac{E}{S}$$

$$= \frac{E}{S} - \frac{E}{(s + \frac{1}{NT})}$$

$$\therefore E_O(t) = E(1 - \exp(\frac{-t}{NT}))$$

It can now be seen that the output will be exponentially weighted with respect to past inputs, and that, if the number of states of the integrator is increased (N) the time constant and accuracy will increase with a corresponding decrease in bandwidth. This ADDIE is called the 'noise' ADDIE, because it uses a pseudo random binary number to give the feedback sequence. There is a modified version which uses a binary rate multiplier instead of a random number source and comparator and is called the BRM ADDIE.[8] This is discussed further in Chapter four.

1.7 Pseudo-Random Binary Sequence Generation [2,8]

Integration, summation and the input and output interfaces all require either a single random sequence, or a twelve bit random number. The sequences must have a probability of a half, which may be obtained by using pseudo random binary sequence (PRBS) generators. These generators consist of N-bit serial shift registers with feedback; usually taking the form of a single exclusive OR gate whose inputs are the Nth bit and one other, which is chosen as follows.

In an n bit shift register, there are 2^n possible states that it may take. When the feedback is connected in the correct way, the shift register will cycle through every possible state except all zeros. The reason for this being that the all zeros state is one in which the shift register would remain indefinitely.

If the shift register does cycle through every possible state, this would be called a maximal length sequence. It is possible to connect the feedback in a different manner, however, and the generator may only cycle through a small fraction of all possible states. The generator may also start in a state which is not part of the final cyclic pattern.

The maximum length that a sequence can have is 2^n-1, where n is the number of bits in the shift register. Take for example, a three bit PRBS generator. This is shown in Figure 1.11. The feedback is taken correctly from bits 1 and 3. The sequence would take the following form, assuming it was started in state 001.

Time /.

Time	Bit 1	Bit 2	Bit 3
0	0	0	1
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	1	0	1
6	0	1	0
7	0	0	1

If we take the feedback from bits 1 and 2 however, the shift register will take the following sequence, assuming it starts in state 001.

Time	Bit 1	Bit 2	Bit 3
0	0	0	1
1	0	0	0

This reverts to all zeros, and remains there. It can be shown that the shift register will take a three state cycle if started in any other state. This is shown in the state diagram in Figure 1.12.

This shows the general method of PRBS generation which is used in the stochastic computer, but of course with much larger shift registers. The detailed design considerations and construction peculiar to this particular system are discussed further in Chapter Four.

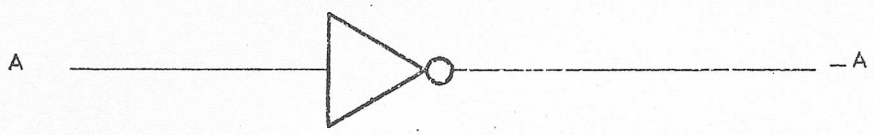


FIGURE 1-1
INVERTER

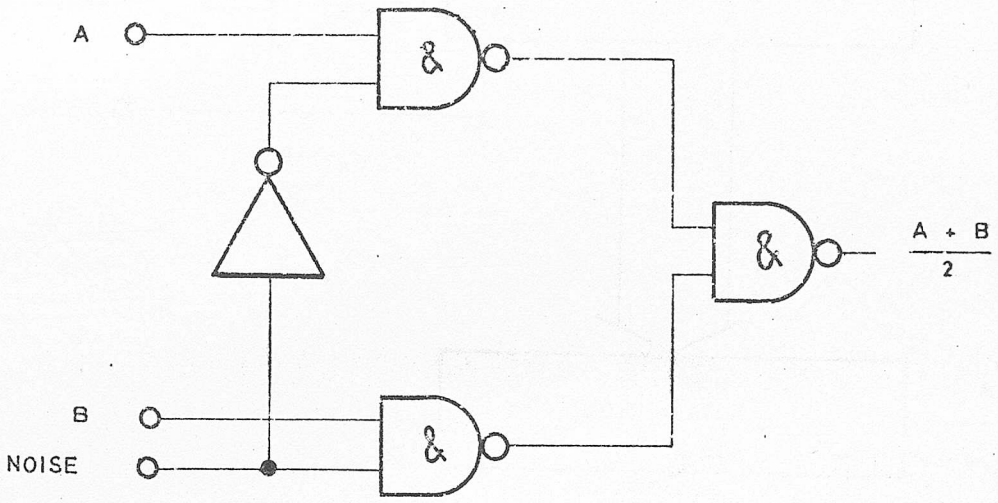


FIGURE 1-2
SUMMER

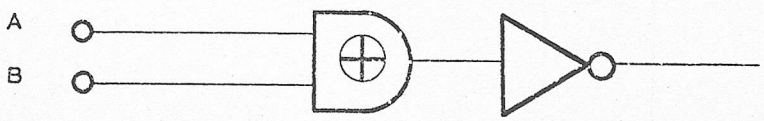


FIGURE 1-3
MULTIPLIER

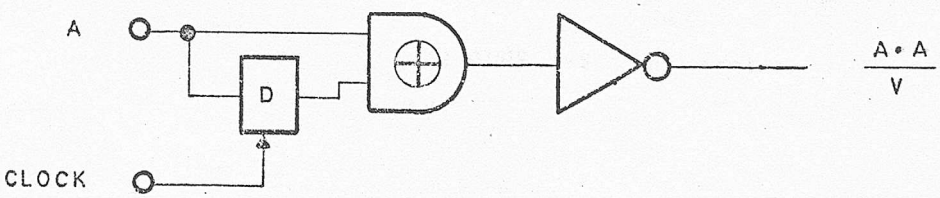


FIGURE 1-4
SQUARER

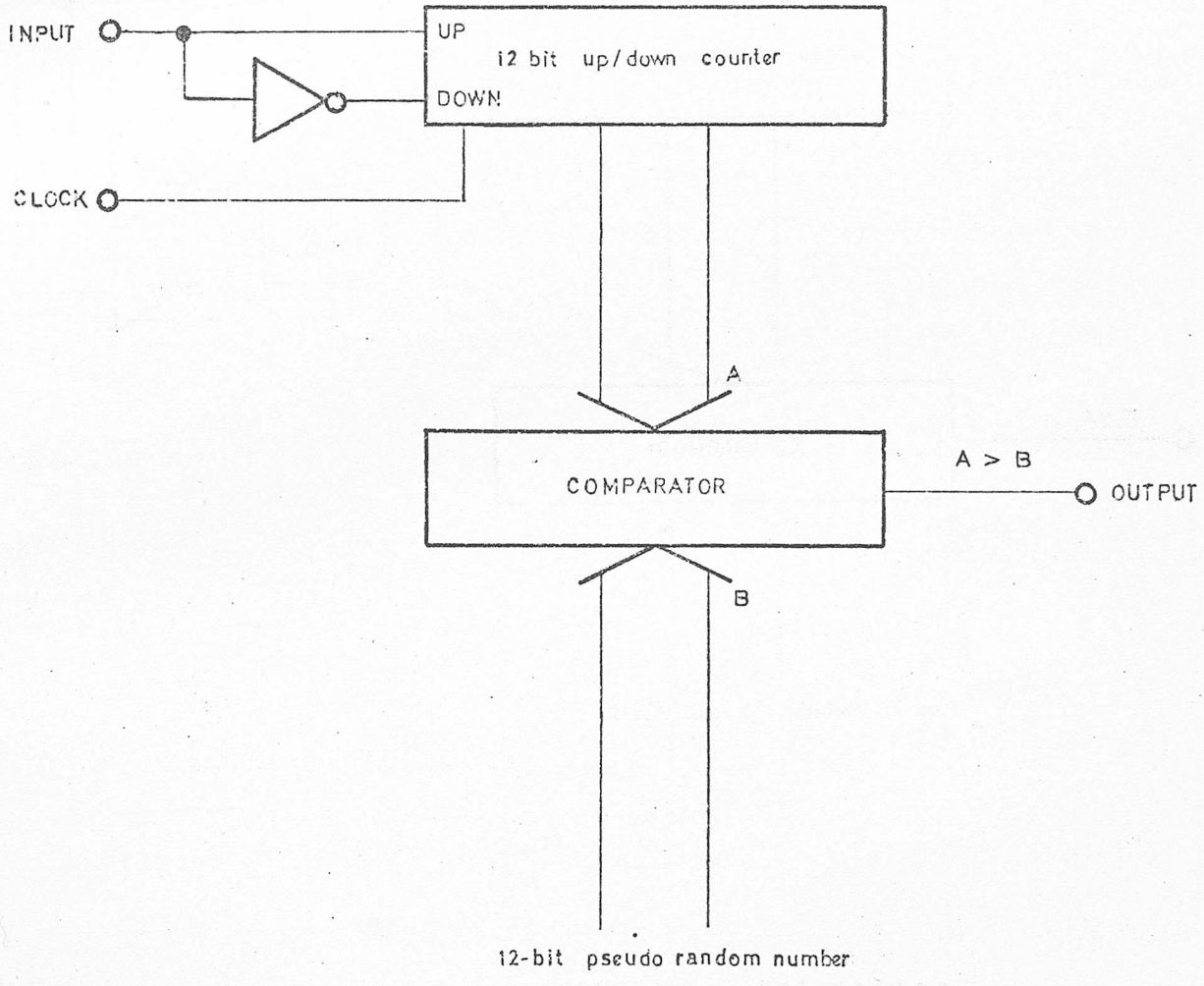


FIGURE 1·5 Integrator

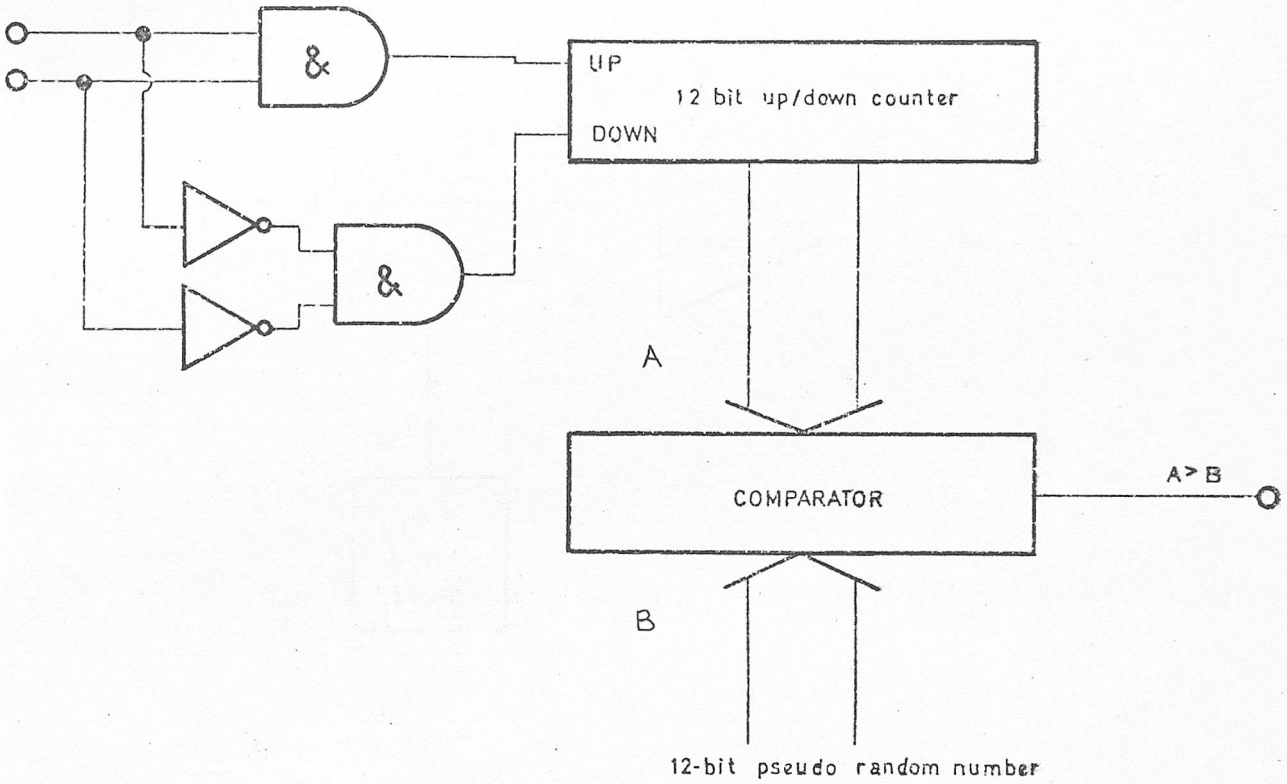


FIGURE 1.6 Integrator

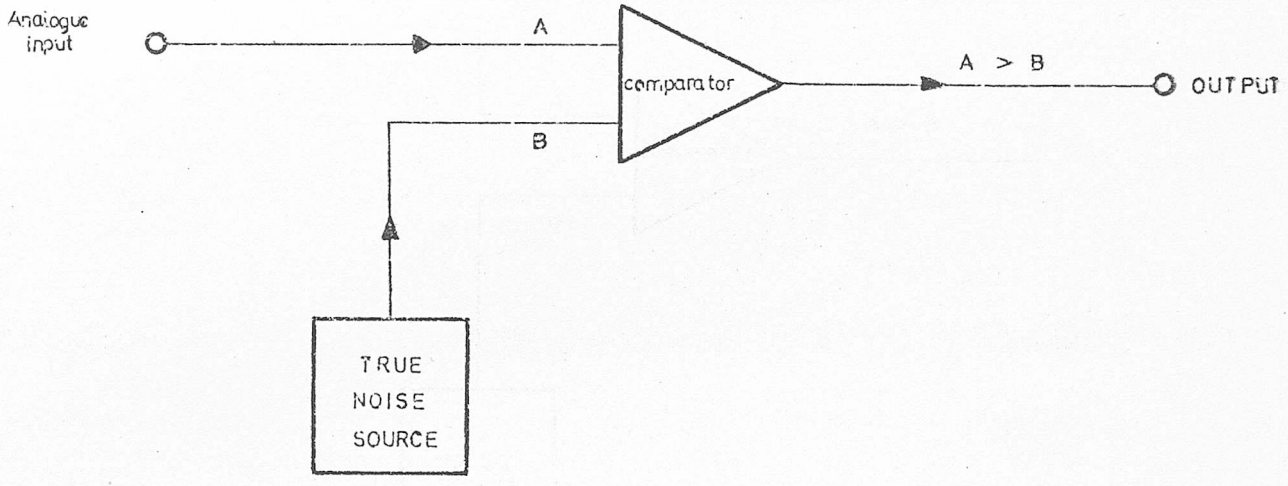


FIGURE 1.7 Analogue to stochastic converter

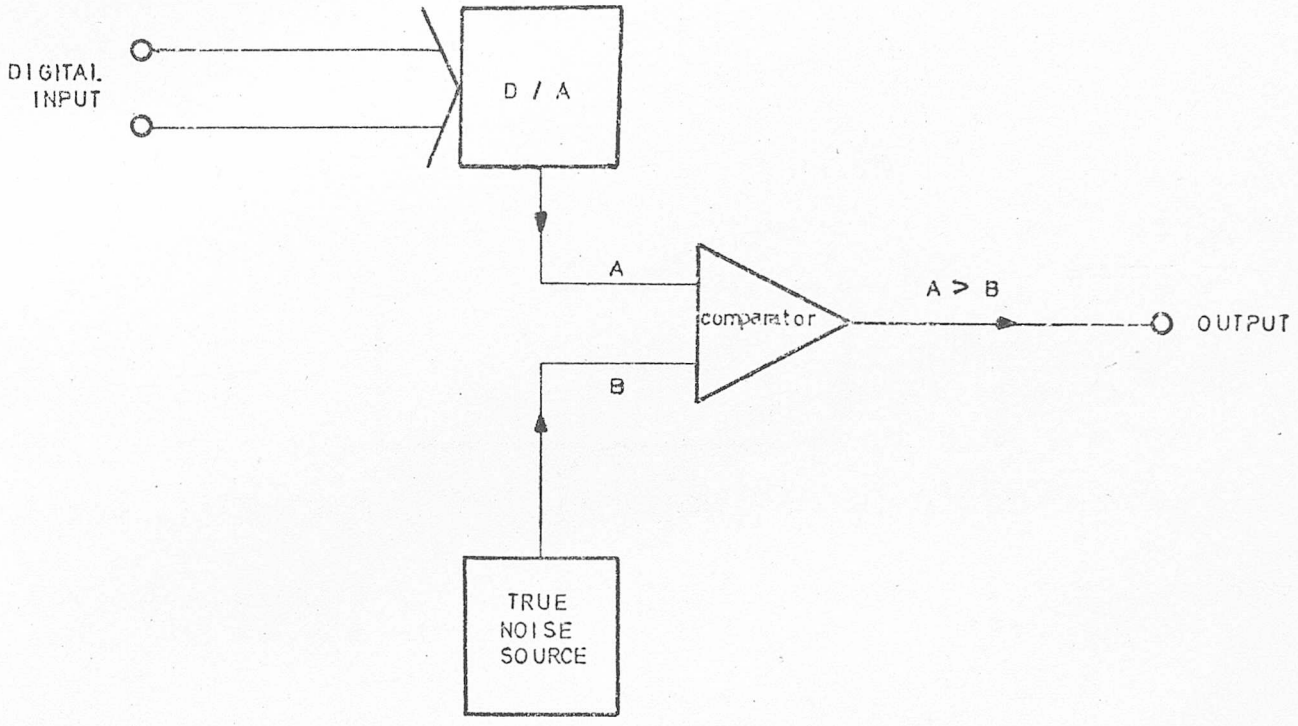


FIGURE 1.8 Digital to stochastic converter

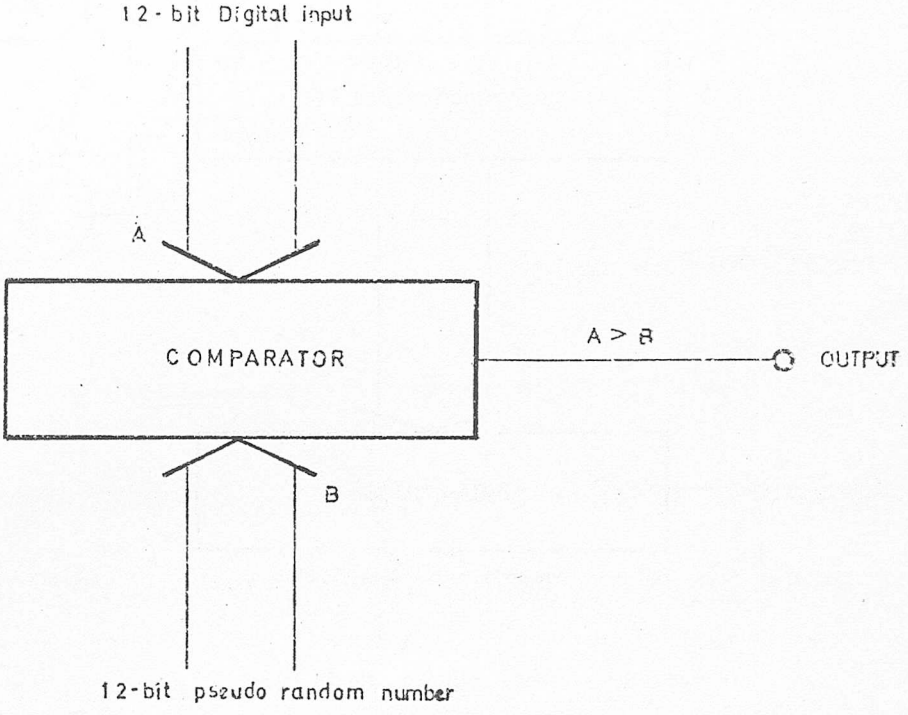


FIGURE 1.9 Digital to stochastic convertor

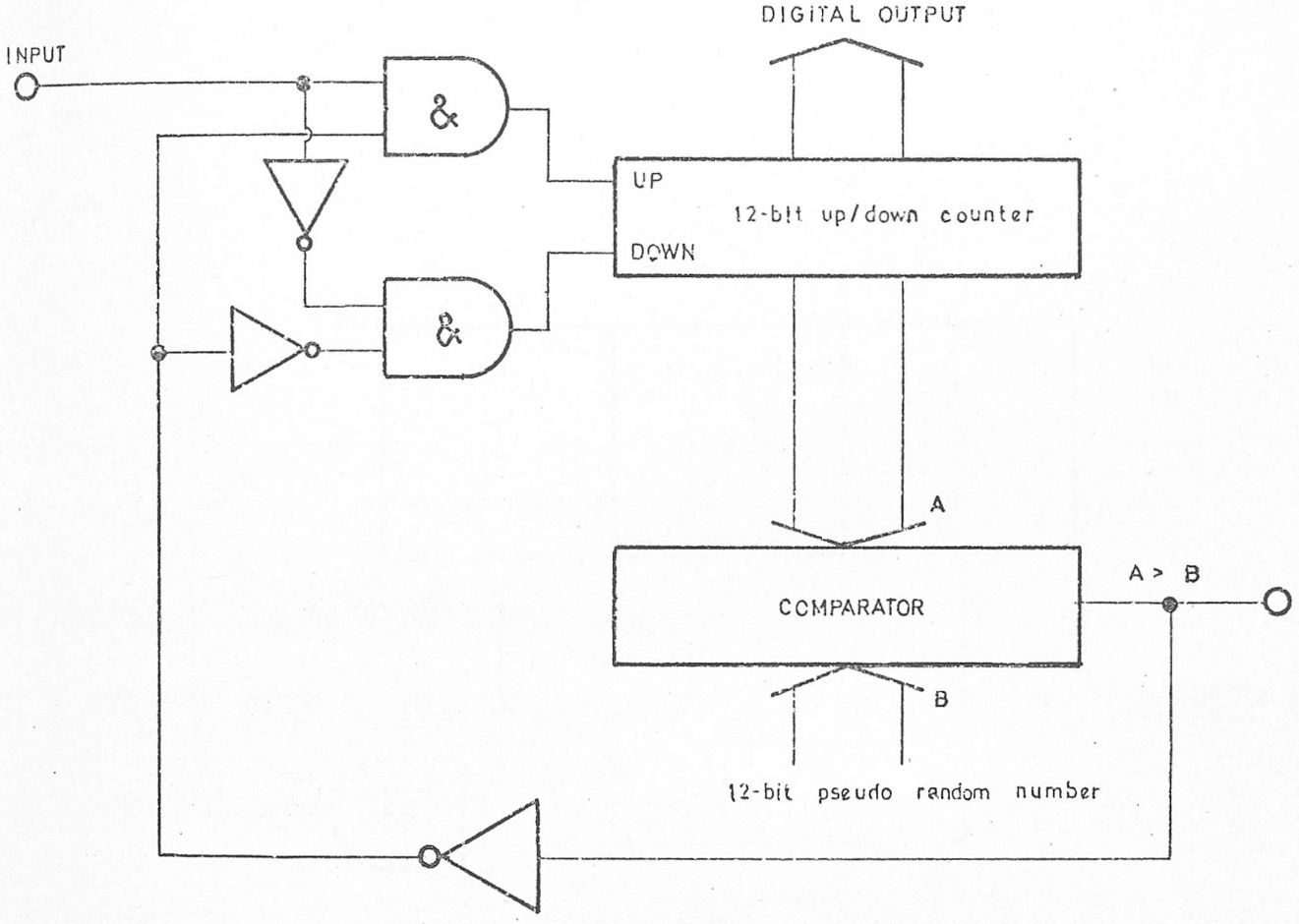


FIGURE 1.10 NOISE ADDIE

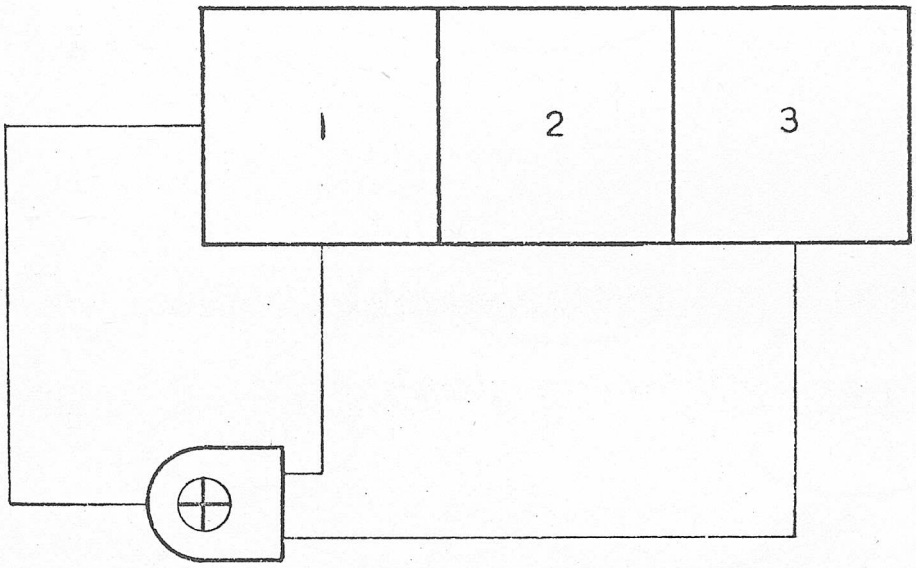


FIGURE 1·11 PRBS Generator

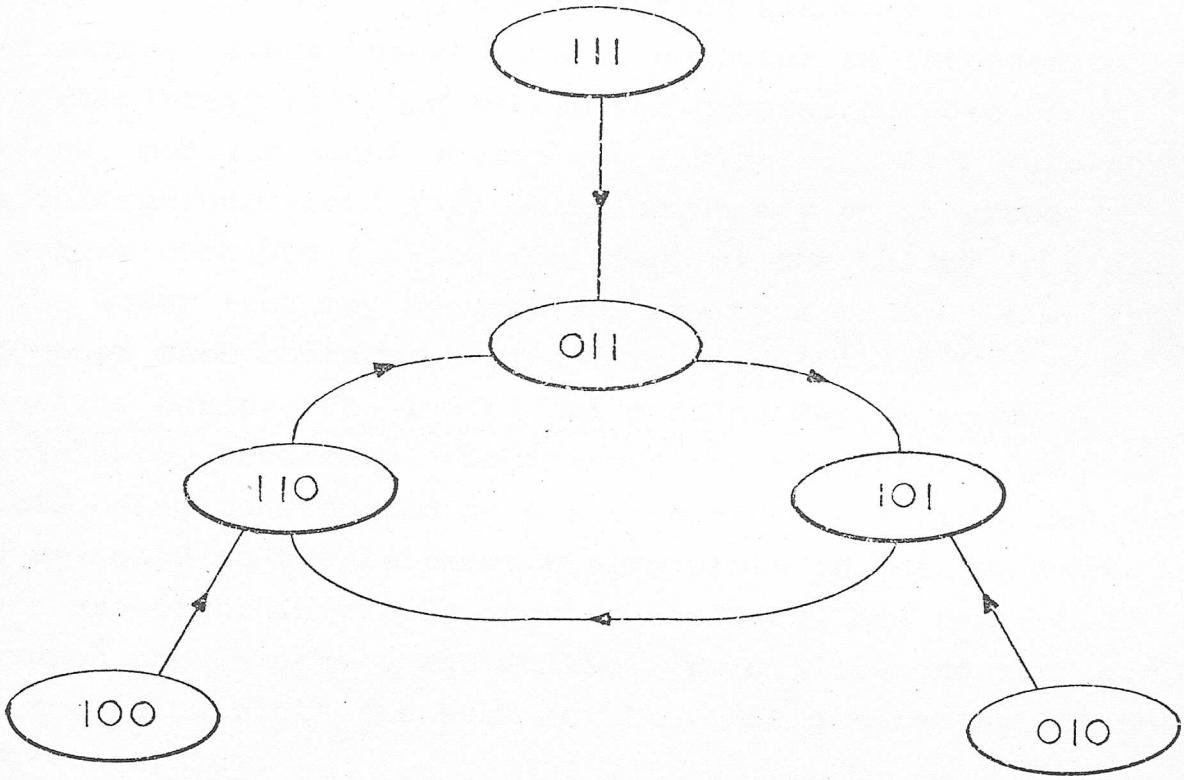


FIGURE 1·12 PRBS Sequence

CHAPTER 2

GENERAL SYSTEM ORGANISATION OF DISCO

This chapter describes the general organisation and design of DISCO, the factors influencing the particular choice of construction method, and the overall operation of the system.

The DISCO system is shown in Figures 2.1 and 2.2. It is built using verorack modules and mounted in a standard 19" rack. Three of the modules contain the complete system, one being totally utilised for the automatic patching,[9] and the other two containing the computing elements and PRBS generators. Since the stochastic computer is intended to be a completely flexible system for general purpose simulations, modular construction was envisaged. Each verorack has thirty-four slots into which cards may be inserted. To ensure complete flexibility, each of the thirty-four slots in the lower rack may be used for any type of module whilst the upper rack contains twelve fixed modules and the PRBS generator cards. The individual modules may be summers, multipliers, inverters, comparators or integrators, each module being constructed on one verocard. The fixed modules are assigned to the leftmost twelve slots in the top rack. There are eight inverters, ten multipliers, and ten comparators, of which the inverters and multipliers utilise one card each. The remaining slots are used for the PRBS generators for the complete system.

Some of the modules require a twelve bit random number, whilst others only require a single bit random sequence. In addition, modules may require a clock pulse, or a hold line, or a connection for the input of data. Since the system is to be completely flexible, each modular slot in the system must have all of these functions available. The pin assignment for each modular slot is shown in Figure 2.3. We are limited to one operation per card even though the inverter, for example, uses only one sixth of an integrated circuit. The reason for this unfortunate /

unfortunate fact is that the integrator circuitry is so complex that only one may be constructed on a verocard. If the system is to remain truly modular, therefore, we may only assign two inputs and one output to each slot.

The automatic patching system[9] has the capacity of connecting any of 96 input nodes to any of 64 output nodes. The modular section of DISCO uses thirty-four of these outputs and sixty-eight inputs. The fixed inverters use eight inputs and eight outputs, the fixed multipliers twenty inputs and ten outputs, and the fixed comparators no inputs and ten outputs. This gives a total utilisation of 96 inputs and 62 outputs. The two connections remaining may be used for external inputs which are already in stochastic form.

The comparators, which are used to provide the conversion between a digital number and a stochastic sequence, must have their data loaded from the PDP8/E under program control. This is done serially, and thus each comparator contains a twelve-bit, serial in, parallel out shift register, with data in and data out connections. Each slot has the pin corresponding to the data out line, connected to the data in line on the adjacent slot. If the comparators in a system are inserted in adjacent slots, this will then form a large, serial in, parallel out shift register. On modules which are not comparators, the data in and data out connections are short circuited, so that they may be inserted between comparators if so desired. A similar method is used for the scaling information required by the integrators. In this case, however, only four bit shift registers are required. This method of loading the data requires that there be no slots left unused between modules, so that the serial data line is not broken. An exception to this rule is that if the modules to the left of the unused slot do not require scaling or input information. Since the data is entered serially from right to left, the slots must be used in that order. Each slot is assigned one output number and two input numbers for use in patching. The leftmost slot in the lower rack is output number one, increasing to output thirty-four at the far right slot. The input numbers of any slot n are 2n-1 and 2n.

When /

When one wishes to use the computer, there is available a layout of the system slot assignments, giving the input and output numbers of the fixed modules, so that interconnections may be easily made. This is shown in Figure 2.4. The modules that are required should be inserted into the modular part of the system, from slot thirty-four to slot one. The block diagram of the required configuration should be marked with the input and output numbers corresponding to the connections in DISCO. An example of this procedure is shown in Figure 2.5. The required connections may now be typed in on the keyboard using the 'C' command, as shown in Figure 2.6. Using the 'S' command, the integrators should then be scaled. Even in the case of unity scaling an integrator must be scaled to ensure that any information in the program, perhaps entered by a previous user, is not used inadvertently. If a connection is made to an output by mistake, it may be corrected by simply retyping the instruction. This erases any previous connection. If there are comparators in the system, the input information is entered by the 'I' command. The comparator numbers that appear on the V.D.U. refer to the comparators in the system, numbered from left to right in ascending order. The final operation corresponds to the setting up of the initial conditions on the integrators.[9] This is essential, because any further instructions will cancel the automatic compute-reset cycle initiated by this command.

There are three distinct methods which may be used to obtain an output from the system. Firstly, the output to be investigated may be read directly by the use of a stochastic to analogue converter.[9] The converter is simply connected to any node using the automatic patching. The analogue output may be connected to a digital voltmeter for an accurate representation of the variable. Unfortunately, the resultant information must be converted from a voltage to a physical variable by manual calculation. However, if necessary, it would be simple to modify the program to read this voltage and convert it automatically to the required value.

A /

A second method of output interface involves direct reading of a twelve bit output if the output is from an integrator, and it is plugged into a slot with twelve connections to the PDP8/E. This slot may also be used with an ADDIE to which a single line output has been connected using the patching system. The particular twelve bit number is automatically converted to the proper value and displayed on the V.D.U.

The final output interface method is normally used only for diagnostic purposes. This method utilises the distribution curve as a means of determining the accuracy and variance of an output. A twelve bit number is read continuously and a distribution curve built up on an oscilloscope. This curve may be graphed on an X-Y plotter if necessary. Further explanation of this output method, and some examples of results are given in Chapter Four.

Having considered the overall structure of the DISCO system, we will now consider the specific design of an automatic patching system for the interconnection of individual modules.

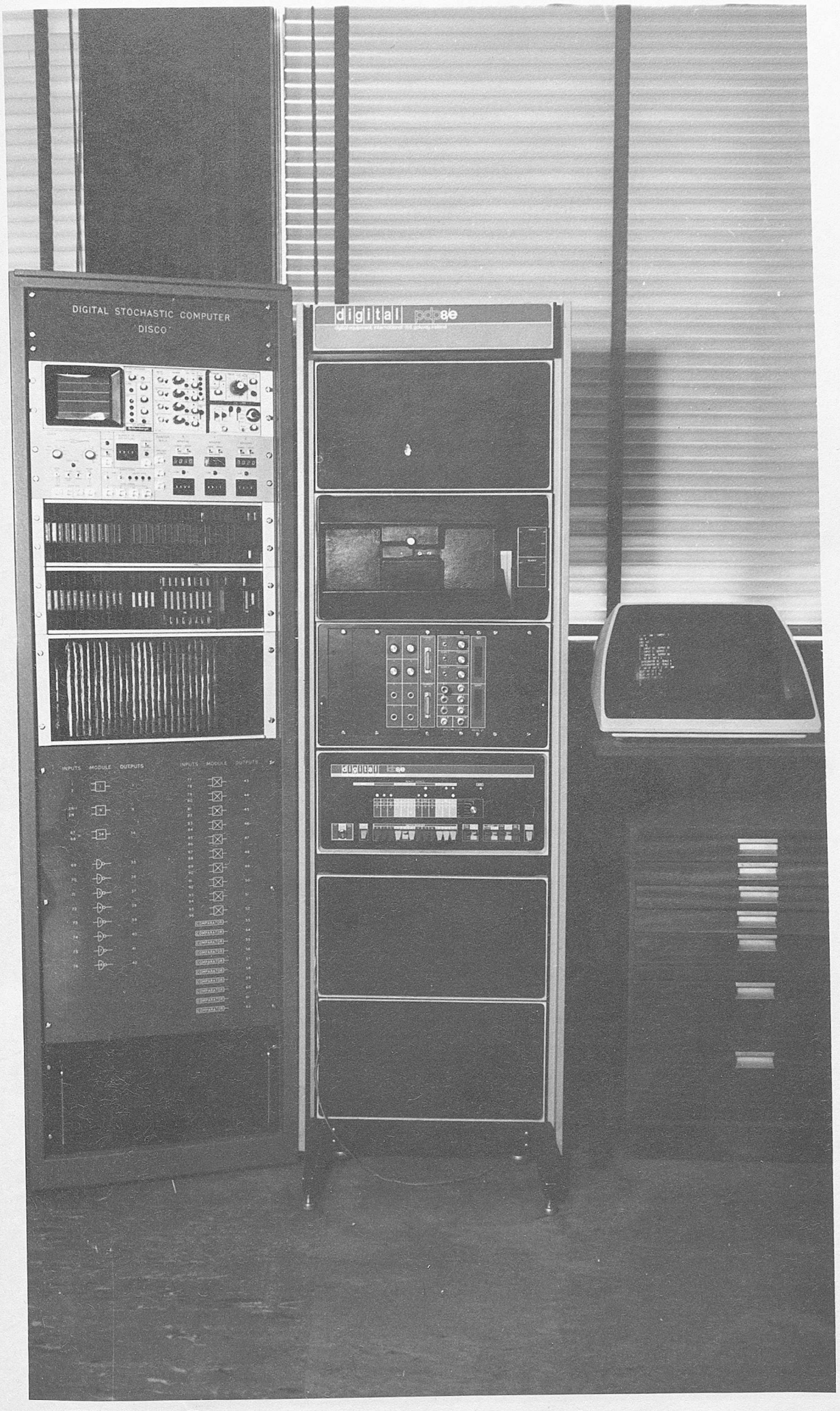

DIGITAL STOCHASTIC COMPUTER
DISCO
digital pdp8/e
INPUTS MODULE OUTPUTS
INPUTS MODULE OUTPUTS

FIGURE 2·1

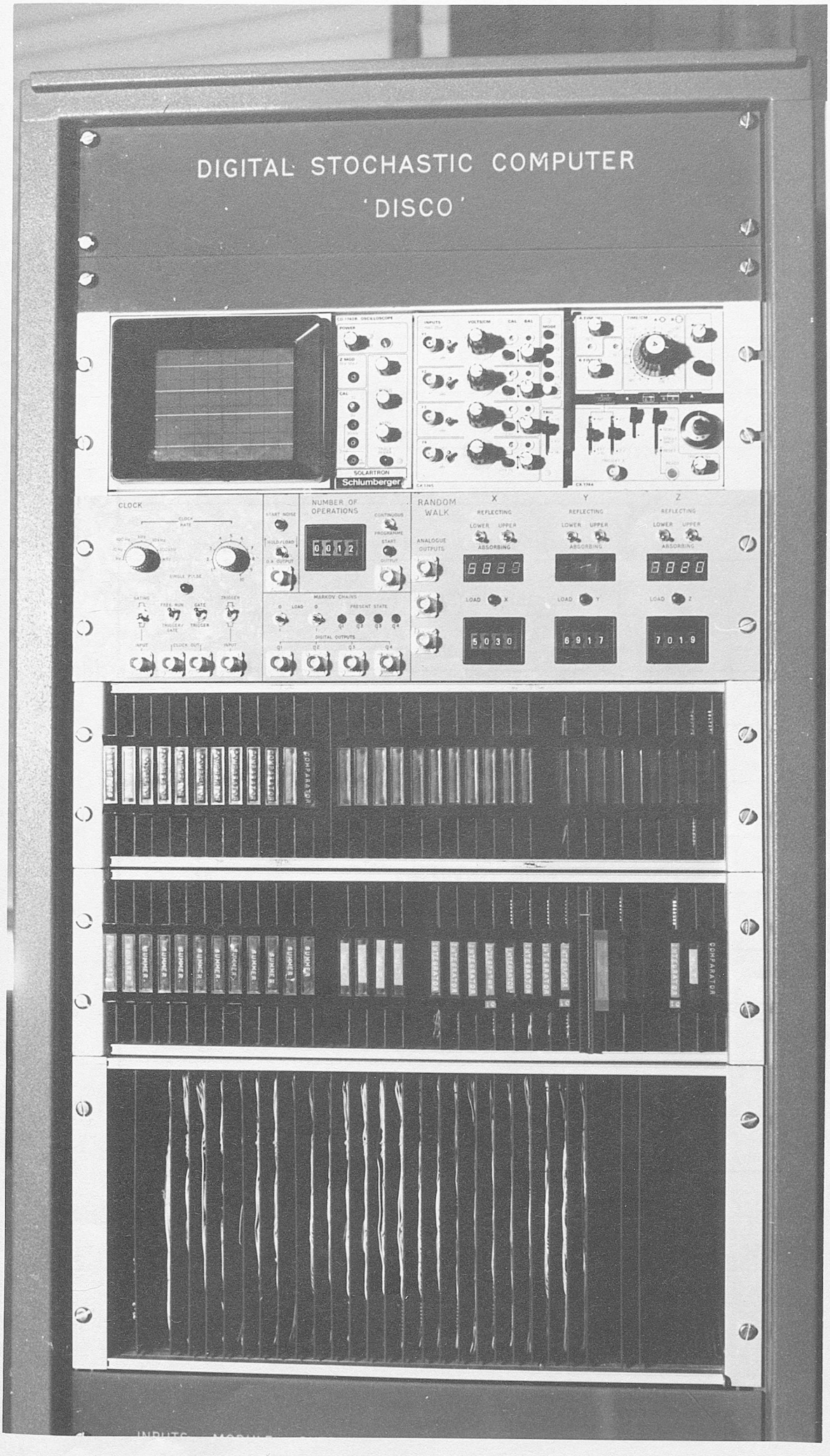

DIGITAL STOCHASTIC COMPUTER
'DISCO'
SOLARTRON
Schlumberger
CLOCK
NUMBER OF OPERATIONS
RANDOM WALK
X
Y
Z
REFLECTING
ABSORBING
ANALOGUE OUTPUTS
MARKOV CHAINS
DIGITAL OUTPUTS
5030
6917
7019
COMPARATOR
INTEGRATOR
BUNKER

FIGURE 2·2

Pin No.	Use
1	+5 Volts
2	Scale input
3	Scale output
4	Master clock
5	Hold
6	Scale clock
8, 10, 12, – –, 30	12-bit noise input
9, 11, 13, – –, 31	12-bit digital output
32	Comparator input
33	Comparator output
34	Comparator clock
35 – 37	Not used
38	Stochastic output
39	Not used
40	Stochastic input
41	Not used
42	Stochastic input
43	Ground

FIGURE 2.3 Pin assignments

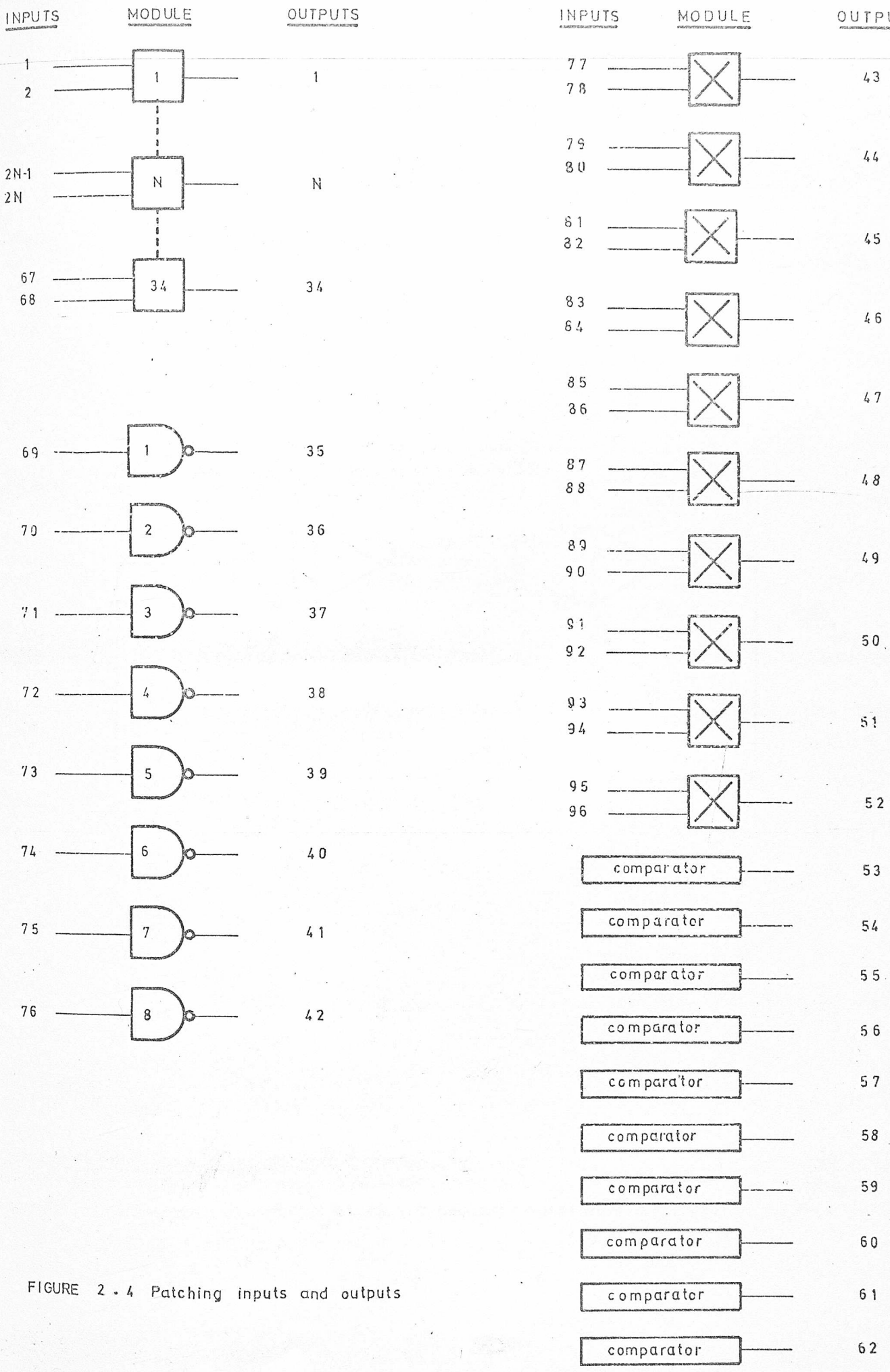


FIGURE 2.4 Patching inputs and outputs

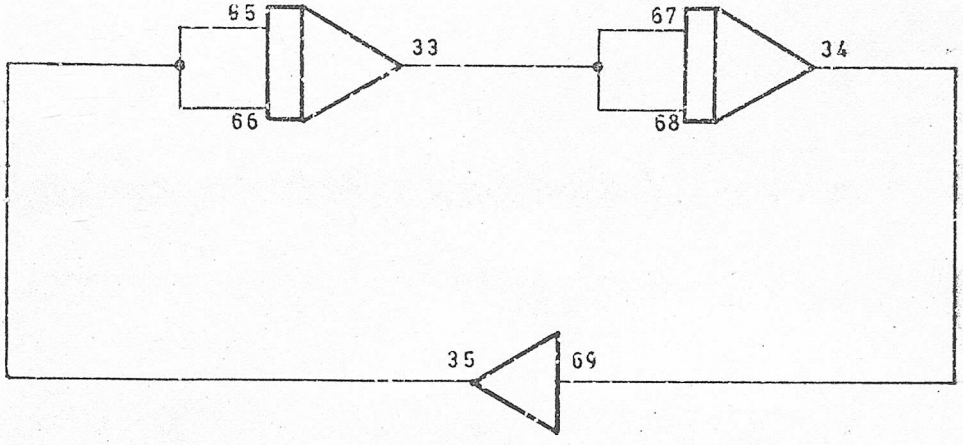


FIGURE 2.5 Stochastic sine wave generator

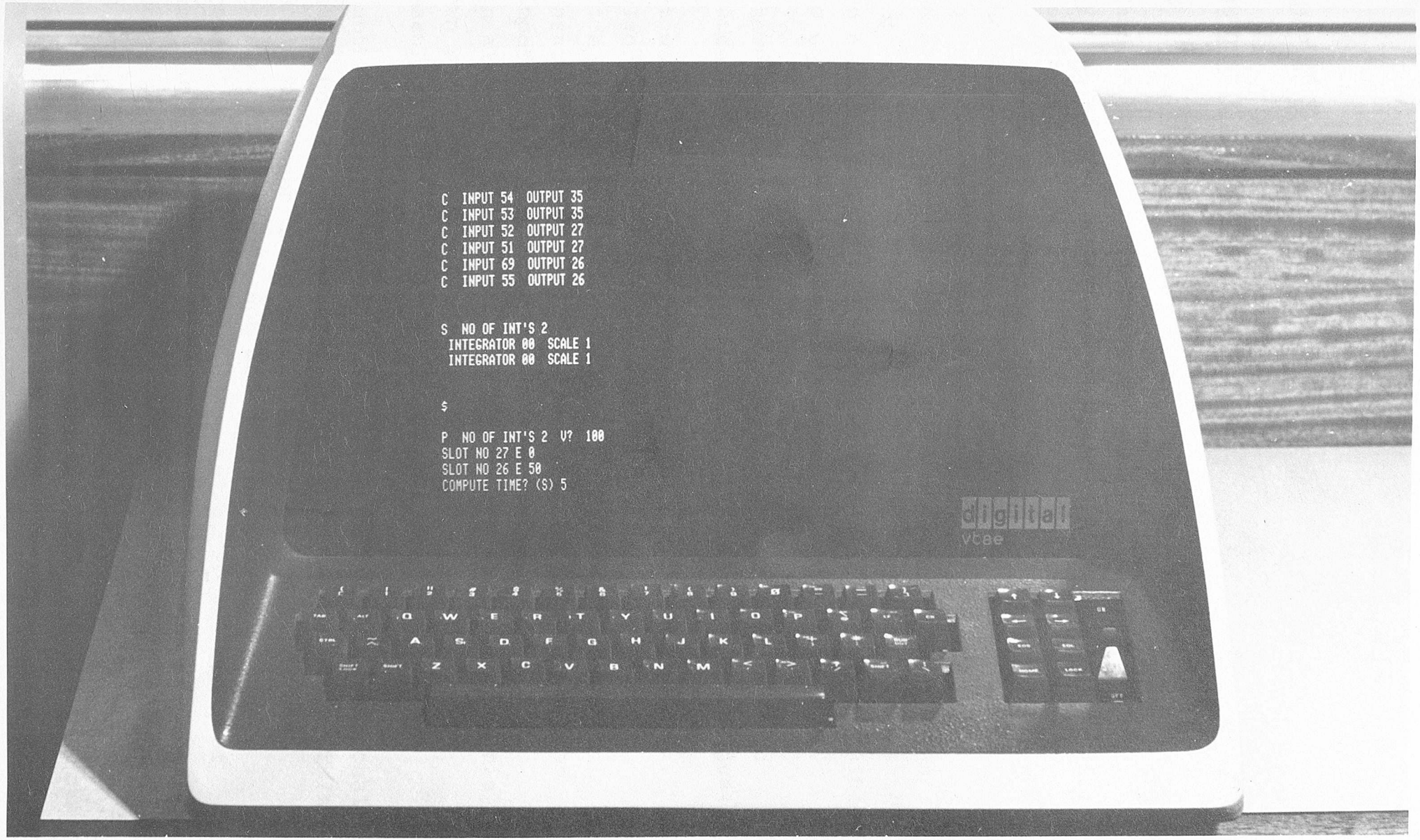

C INPUT 54 OUTPUT 35
C INPUT 53 OUTPUT 35
C INPUT 52 OUTPUT 27
C INPUT 51 OUTPUT 27
C INPUT 69 OUTPUT 26
C INPUT 55 OUTPUT 26
S NO OF INT'S 2
INTEGRATOR 00 SCALE 1
INTEGRATOR 00 SCALE 1
$
P NO OF INT'S 2 V? 100
SLOT NO 27 E 0
SLOT NO 26 E 50
COMPUTE TIME? (S) 5
digital
vtae

FIGURE 2·6

CHAPTER 3

AUTOMATIC PATCHING

This chapter describes the design of an automatic patching system for the connection of stochastic modules and the routing of output data from the stochastic computer.[9]

3.1 General System Organisation

The complete stochastic computing system involves a number of interchangeable modules, with an effective total of 96 input connections and 64 output connections. This represents a total of 96x64 possible connection paths. To obtain a completely flexible system it must be possible to connect any output to any input in a simple manner.

There are several methods that could have been used to patch the system, but almost all have some disadvantages.

It would of course have been possible to hardwire any algorithms on the machine, but this would have been a time consuming process, and generally result in an unsatisfactory system. For example, it would have been impossible, using this method, to have a rapid change in algorithm, and each user would have had to rewire his own particular problem each time he wished to use the machine.

Another possibility would have been to use a conventional patch board system as used in an analogue computer. This would have been a convenient system to use, had the problem of crosstalk not arisen. An analogue machine generally operates at low frequencies, in the order of tens of hertz, while it was envisaged that the stochastic computer would operate at one to ten megahertz. However, a normal patchboard system was tried experimentally, but with little success, due to the predicted crosstalk between adjacent /

adjacent lines on the patchboard.

The problem could also have been solved by the use of a 96x64 switching matrix. A particular input and output node would have been connected by the switch at the junction of their particular row and column. Part of this switching matrix was built, using standard TTL logic circuitry. Two input NAND gates were used as the switches, one NAND gate input acting as a control line and the other input connected to one of the 64 output nodes. A schematic diagram of this system is shown in Figure 3.1. Any particular output from the computer would be connected to one of the inputs of 96 NAND gates, the outputs of the gates being wire-OR'ed. Each output would be connected to one of the 96 input nodes of the stochastic computer. For this system, 64 modules would be required, one for each output of the computer.

The control lines would be activated by a command from the PDP8/E minicomputer, the actual information bits being loaded into a serial shift register, 6144 bits long.

The main disadvantage of this form of patching is the high cost both in terms of labour and materials. A typical breakdown of the cost of such a system is given in Figure 3.2.

3.2 Hardware Organisation

The apparent complexity of the system is reduced by the fact that the stochastic machine never requires two outputs to be patched to a given input. It is thus possible to consider using some form of multiplexing system, whereby one, and only one, output is patched to a specific input. This problem may be more easily understood if we say that only one output node may be connected to a given input node. However, any output may be connected to several input nodes. For the 96x64 system we require that a sixty four-to-one multiplexer system be connected to each input, to allow it to be patched to one and only one of the 64 outputs.

We /

We therefore require 96 of these multiplexers, and although this would appear to be a complex system, it is simplified by the use of MSI integrated circuits. Each multiplexer unit is constructed using four sixteen to one data selectors, and a four to one data selector. (Texas Instruments type SN74150 and SN74151). The schematic diagram is shown in Figure 3.3.

The configuration established by these multiplexers is decided by a six bit binary code, contained in a serial in, parallel out shift register, loaded from the PDP8/E on command from the keyboard. The four least significant bits of the code are common to each of the SN74150 data selectors, and the two most significant bits are connected to the final SN74151 data selector. This integrated circuit is an eight to one data selector, with the most significant bit grounded. By using these MSI integrated circuits, it allows the construction of four multiplexers on one large verocard. There are therefore twenty four cards required, a considerable saving on the previous method.

Since each of the 64 outputs must have a connection to 96 inputs on the multiplexers, they must be buffered. On each of the boards, a single inverter is used for each input to drive the four multiplexers on that board. This reduces the number of gates to be driven by each output to 24. This is achieved by the use of the SN7440 dual four input NAND buffer. These power gates each have an output driving capability of 25, and so only one is required for each output. The buffers are located on two separate cards, adjacent to the patch boards. Two cards are necessary since there are 64 inputs and 64 outputs to be connected, and there are only 72 edge connections on each card.

As well as being smaller and easier to construct, this system is also much less expensive. The cost breakdown of this system is shown in Figure 3.4.

Although /

Although this system is more compact and uses less integrated circuits than the switching matrix, it still requires a large amount of power. Each board of multiplexers requires approximately one amp at five volts, the complete system requiring approximately twenty two amps. It was decided to use a single power supply for the complete computer, and so thirty amps was considered a reasonable figure. A commercial power supply was purchased for use in the system, rated at thirty amps at five volts. These large currents would not of course be necessary in a situation where stochastic computation is used to control a process. The algorithm would then be hardwired, and the current consumption would be approximately one or two amps, since no patching system would be required.

3.3 Software Organisation

The complete computing system and patching, is controlled by a PDP8/E minicomputer via a visual display unit and keyboard. This is shown in Figure 3.5. The software for the system was written so that the PDP8/E would respond to certain control characters typed on the keyboard. The flowchart for this is shown in Figure 3.6, and the operation of the system is as follows.

When the system is switched on and initialised, (ie the program started and reset), all previous patching information stored in the core memory is reset to zero. This ensures that no patching other than that required will occur.

A flashing cursor is displayed on the VDU to signify readiness to accept data. This also has the function of showing where a displayed character will next appear. The character that is typed on the keyboard is tested to determine which command has been given. A list of command characters and their meaning is given in Figure 3.7. Only the first two commands will be dealt with now, the others in their respective chapters.

If /

If the character typed is not one of the listed characters, the software will cause a question mark to be displayed on the VDU and the command ignored. If the character is a C, the connection routine will be entered, and the legend "INPUT" displayed. This signifies that the command has been accepted and that the software is waiting for a one or two digit number to be typed on the keyboard. This routine is so often used in the program, that a special subroutine was written for it.

This special subroutine will only accept one or two digit numbers. An attempt to enter a third digit will result in a question mark being displayed, and the routine terminated. If an error is made while entering the data, it may be deleted one character at a time from right to left, by means of the "RUBOUT" key on the keyboard, or wholly deleted by use of the 'back arrow' (←). New data may then be entered. The information typed is stored in a single memory location as a six bit binary word, signifying the "INPUT" on the stochastic computer which is to be connected to an output to be specified. After this connection has been stored, a similar routine is used to allow the OUTPUT connection to be entered. This information is also stored as a six bit binary word.

The final part of the connection routine is to collate the information, and set up the required bit pattern in memory. This bit pattern will eventually be loaded into the serial shift registers on the patching boards. The actual information is stored in forty eight twelve bit words in memory, each word corresponding to two input connections. Each twelve bit word is divided into two six bit codes, which correspond to the outputs to be connected.

The /

The "INPUT" data therefore defines a specific location in memory, and the "OUTPUT" data, the six bit code to be entered in that location. On completion of this operation the program will return to the start and await the typing of another control character.

If the character typed in is $, the computer will enter a routine which transfers the input information from memory to the patching system. This information transfer is performed serially because the speed of patching is limited by the typing speed of the human operator. It is also necessary to conserve the output connections on the buffered digital input/output on the PDP8/E. There are twelve outputs and twelve inputs on the interface, the outputs being used in pairs. One of each pair is used for information, and the other the clock pulse for that information. The code shift registers on the patching cards are connected together to form one large serial in, parallel out, shift register of 576 bits. Serial shift registers are also used for, scaling of the integrators, input information to comparators, and the setting up of the initial conditions. This requires a total of ten output connections.

3.4 Automatic Testing

Because the patching cards are of such complexity, it was not feasible to test them by manual methods. A machine code program was written in assembly language to test each of the four multiplexers on each card, with varying inputs and different combinations of codes. The flowchart for this program is shown in Figure 3.8.

The PDP8/E stores in memory the bit patterns of the code inputs, ie, four six bit codes, one for each multiplexer on the cards. These bit patterns will be clocked into the serial shift registers on the cards. These four codes are taken sequentially, and each one varied from nought to sixty three, the other three remaining at zero. For /

For every one of these 256 possible codes, a logic zero is applied sequentially to each of the sixty four inputs. The PDP8/E is limited to twelve digital outputs, two of which must be used for the code information and clock pulse. It was therefore necessary to construct an interface card, which contains a sixty four bit, serial in, parallel out, shift register. Each of the outputs on this shift register connects to one of the sixty four inputs on the multiplexer cards. This interface also contains buffers for the clock pulse and shift register inputs.

The sixty four bit shift register is loaded serially from the PDP8/E under program control. This reduces the number of interconnections to two. The schematic diagram of the interface is shown in Figure 3.9.

The program commences by initialising all code patterns to zero, and loading the interface shift register with logic ones, except for the first bit, which is set to zero. This zero is used as the test input, since there is an overall inversion in the patch boards. In this manner, one can ensure that the output will only go to a logic one, if the correct coding is used, and the board is functioning correctly. In addition, open circuit outputs can be conveniently detected as a logic one.

The four six bit codes are now entered into the code shift register and the four outputs interrogated to see if they correspond to the computed outputs. The program computes the expected result from the code patterns and input position. Unless a fault exists, the outputs should remain at zero at all times except when the input position is the same as the decimal equivalent of the bit pattern in the code shift registers.

If /

If the expected result is not received, then an error routine is entered, and all pertinent information about the fault typed on the teletype. This information consists of:- expected output, received output, multiplexer number (1-4), input number (1-64), and the four six bit codes. The program will then continue with the test routine.

If the received output is correct, then the remaining sixty three inputs are tested, and the code pattern updated. This procedure is repeated for every code pattern in one multiplexer. When every code pattern has been tested, the following multiplexer is tested in the same manner. After all four multiplexers have been tested, the computer may halt, or alternatively, it may continue to test the board by returning to the start of the program.

The machine language coding for this program, which was written in PALIII assembly language, is shown in Appendix two.

This test program does not indicate the specific connection or integrated circuit that is causing the fault, but by consideration of the printout, the general nature of the fault may be found. More detailed investigation may then be performed. In practice, a fault is not limited to a single multiplexer, since the inputs are common to all four, and thus by the repetitive nature of a fault, it may be narrowed down to a particular area.

The program has proved its usefulness in the course of time, since approximately 75 per cent of the patch boards exhibited at least one fault immediately after construction. The program is also used to provide a routine check of the patching system every month or so, to ensure no unnoticed faults upset the operating of the stochastic computer.

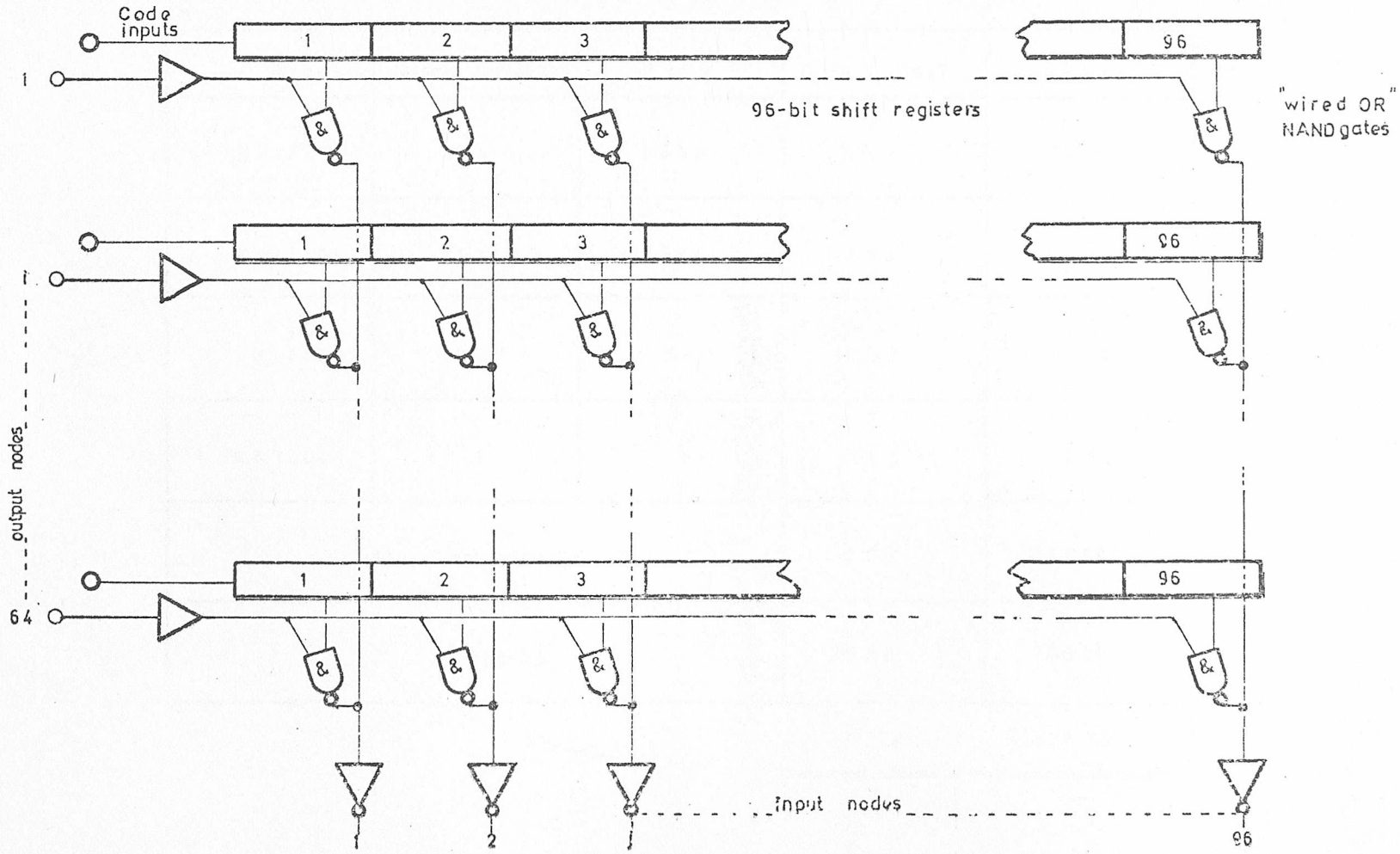


Figure 3.1 Initial patching system

I.C. types	Number per board	Unit total	Unit cost	TOTAL COST
SN7409	24	1536	0·14	215.04
SN74164	12	768	1.90	1459.20
SN7440	2	128	0.12	15.36
SN7404	1	64	0.12	7.68
64 Veroboards			2.00	128.00
2 Veroracks			25.00	50.00
			TOTAL	£1875.28

FIGURE 3.2 Costing of initial patching system

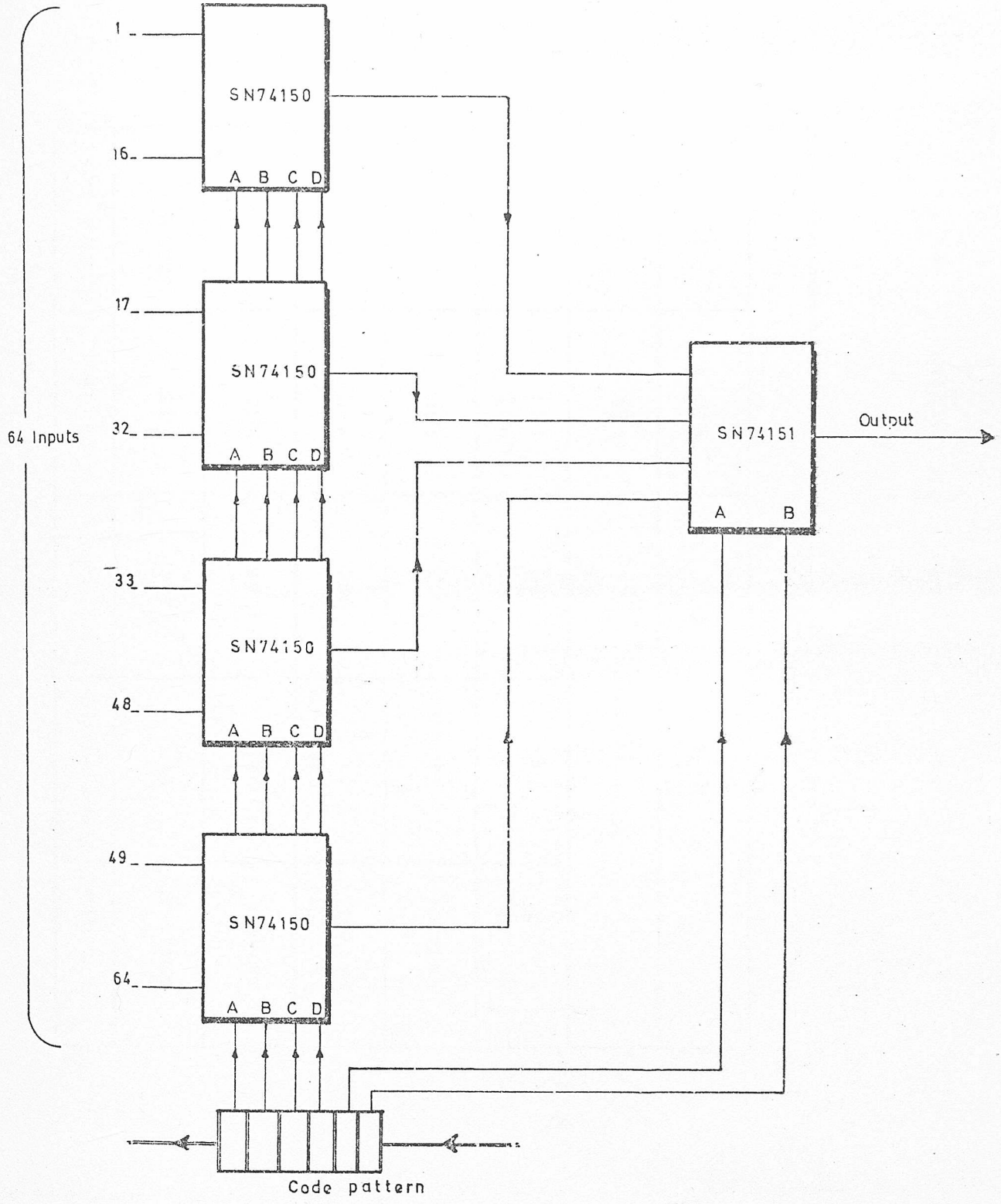


Figure 3.3 Multiplexer

I. C. types	Number per board	Unit total	Unit cost	TOTAL COST
SN74150	16	384	2.15	825.60
SN74151	4	96	0.90	86.40
SN7404	11	264	0.12	31.68
SN74164	3	72	1.90	136.80
Buffers SN7440	16	32	0.12	3.84
Buffers SN7404	6	12	0.12	1.44
26 Veroboards			2.00	52.00
1 Verorack			25.00	25.00
			TOTAL	£1162.76

FIGURE 3.4 Costing of final patching system

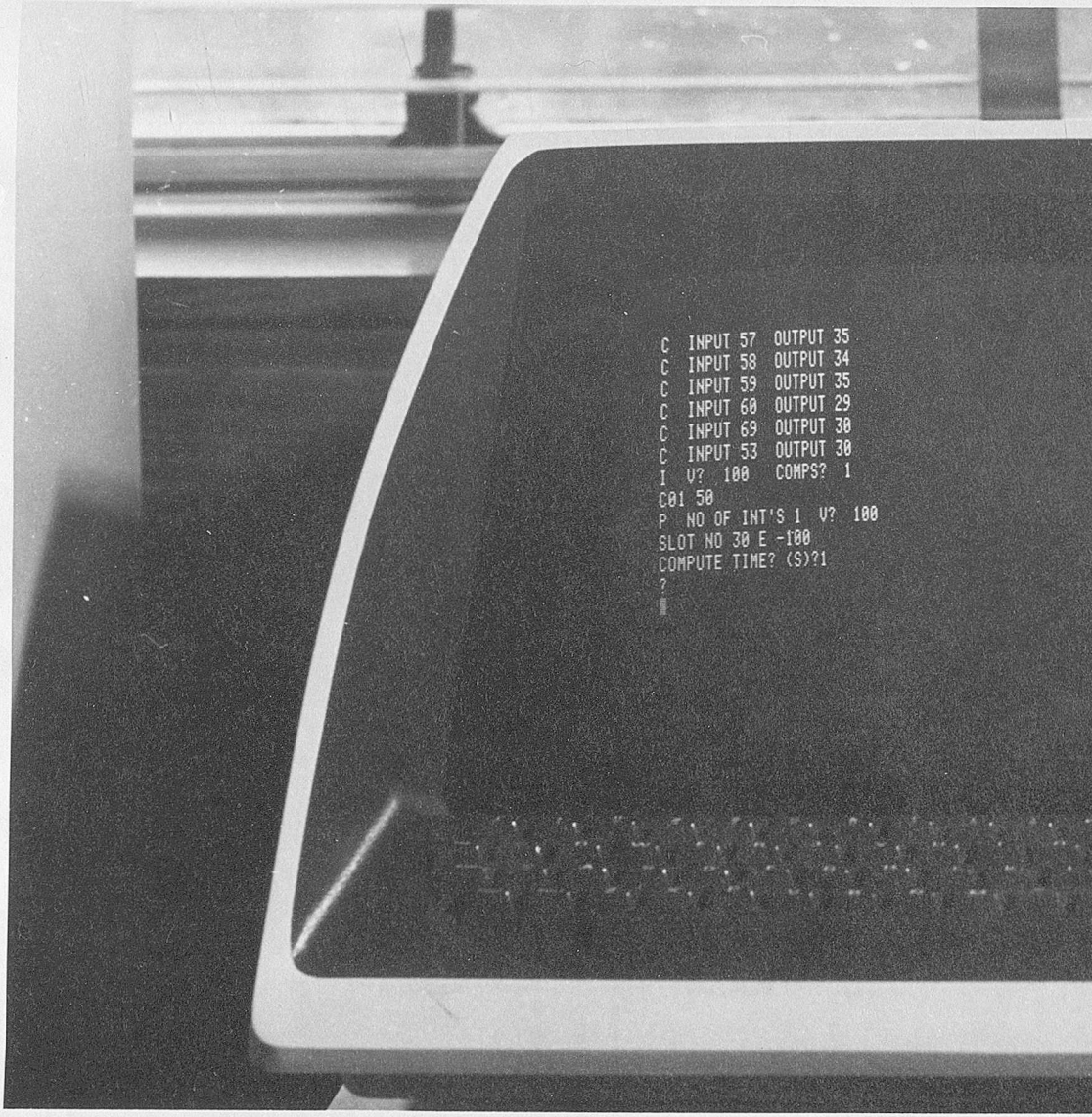

C INPUT 57 OUTPUT 35
C INPUT 58 OUTPUT 34
C INPUT 59 OUTPUT 35
C INPUT 60 OUTPUT 29
C INPUT 69 OUTPUT 30
C INPUT 53 OUTPUT 30
I V? 100 COMPS? 1
C01 50
P NO OF INT'S 1 V? 100
SLOT NO 30 E -100
COMPUTE TIME? (S)?1
?

FIGURE 3·5

digi
vt320

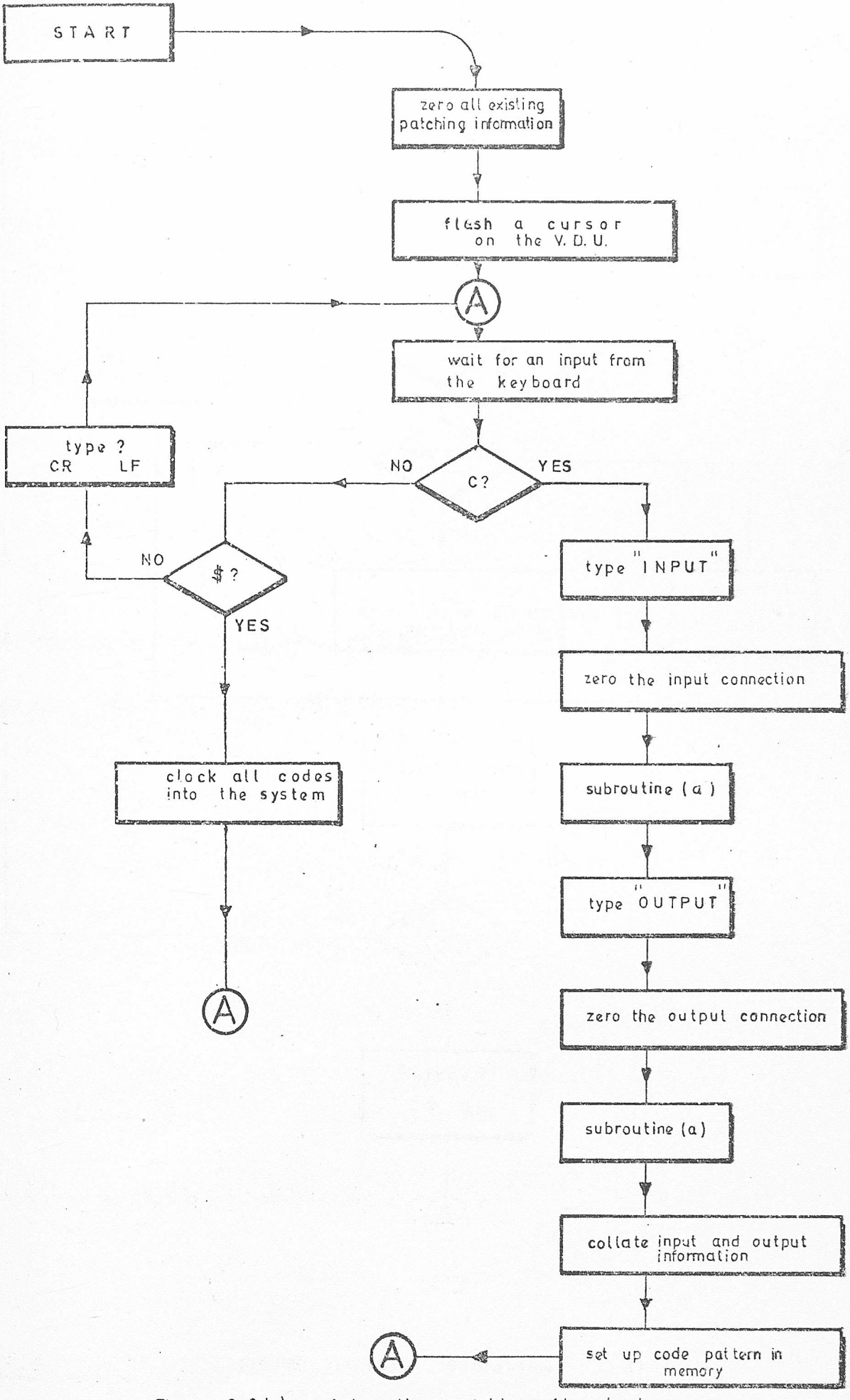


Figure 3.6 (a) Automatic patching flowchart

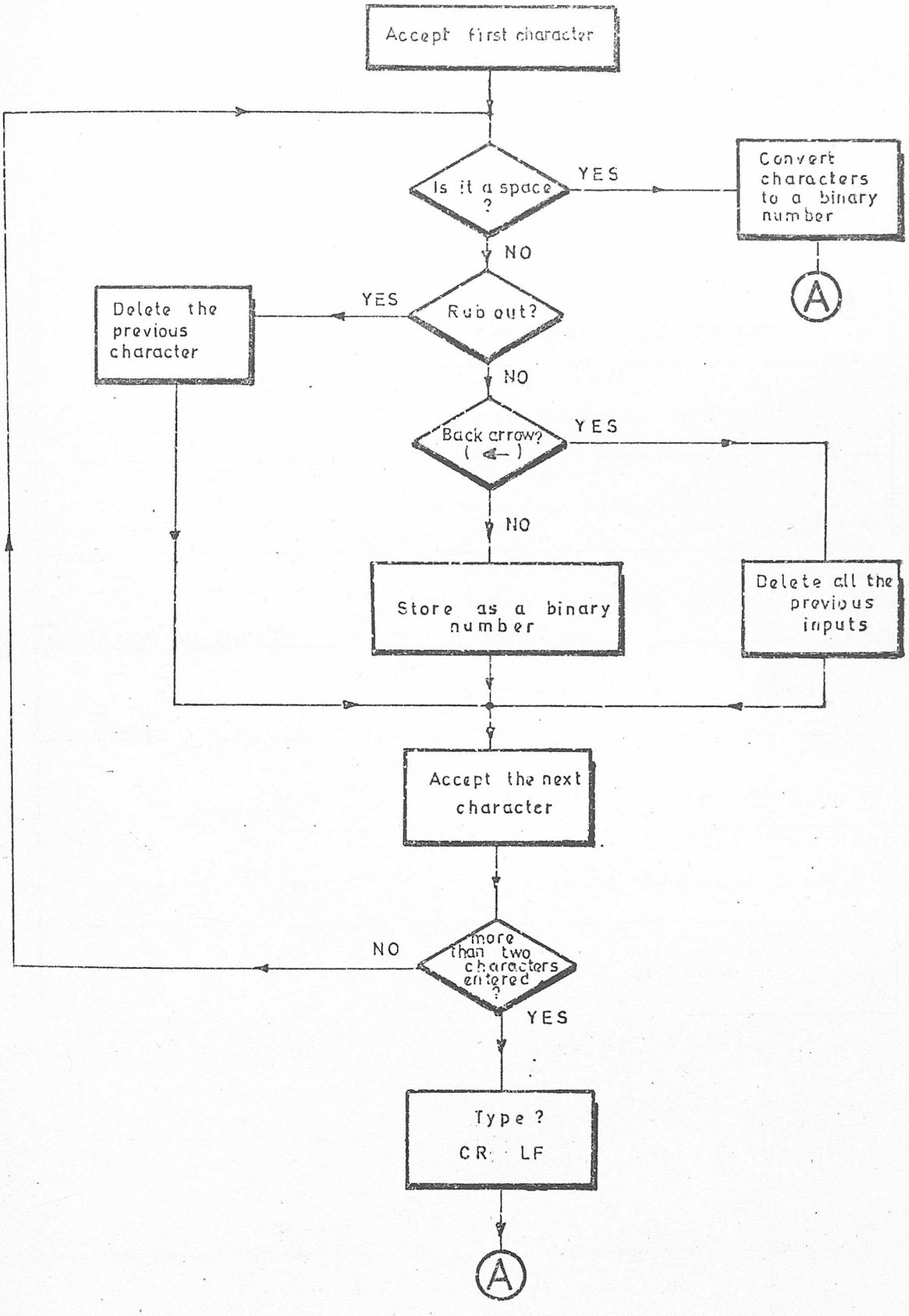


FIGURE 3.6 (b) Subroutine (a)

CONTROL CHARACTER	FUNCTION
C	Interconnection of elements
$	Enter information into DISCO
S	Scale the integrators
R	Read a twelve bit number
D	Distribution curve
G	Graph of distribution curve
P	Set up initial conditions
I	Input to the comparators

FIGURE 3.7 Control commands

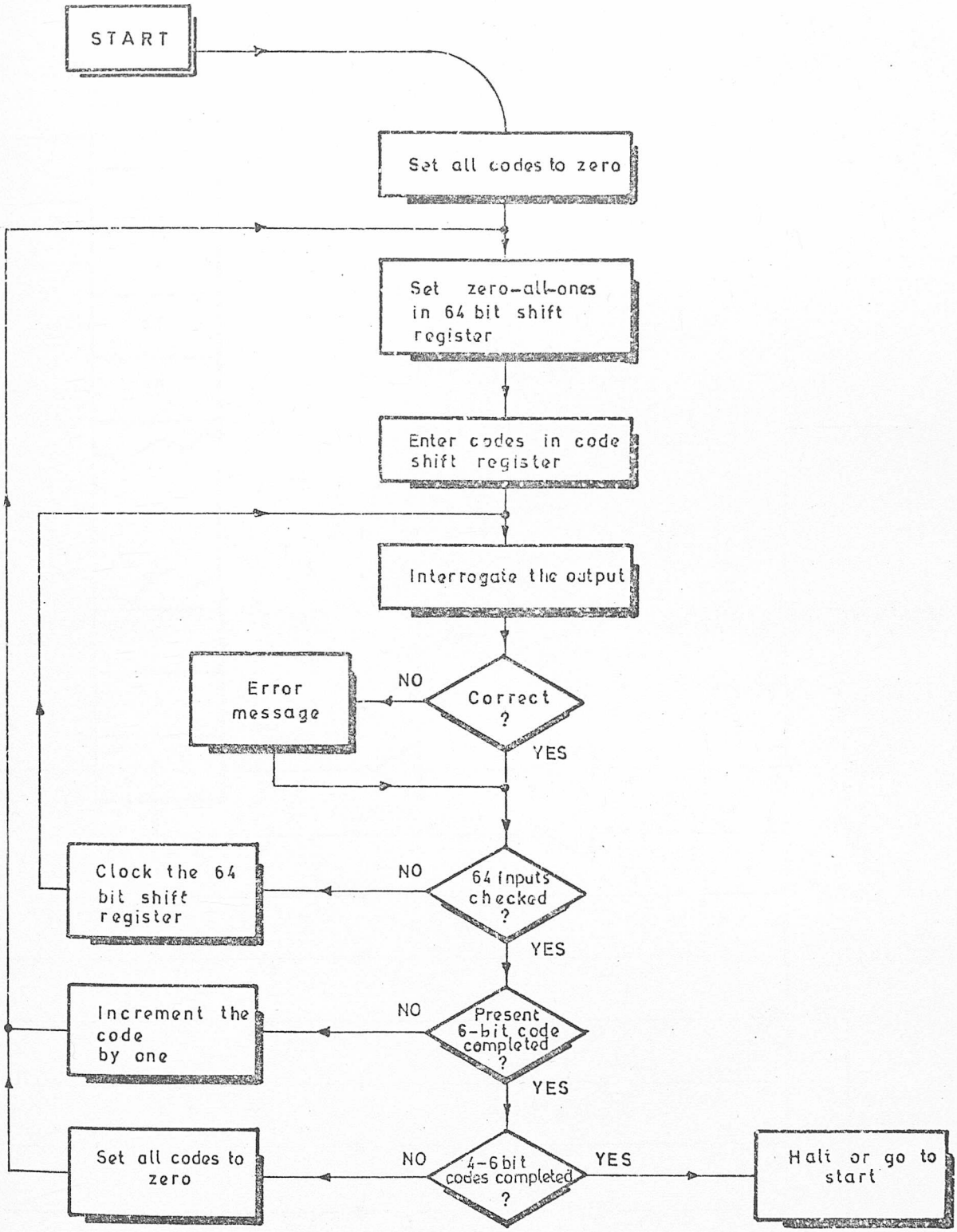


FIGURE 3.8 Automatic testing flowchart

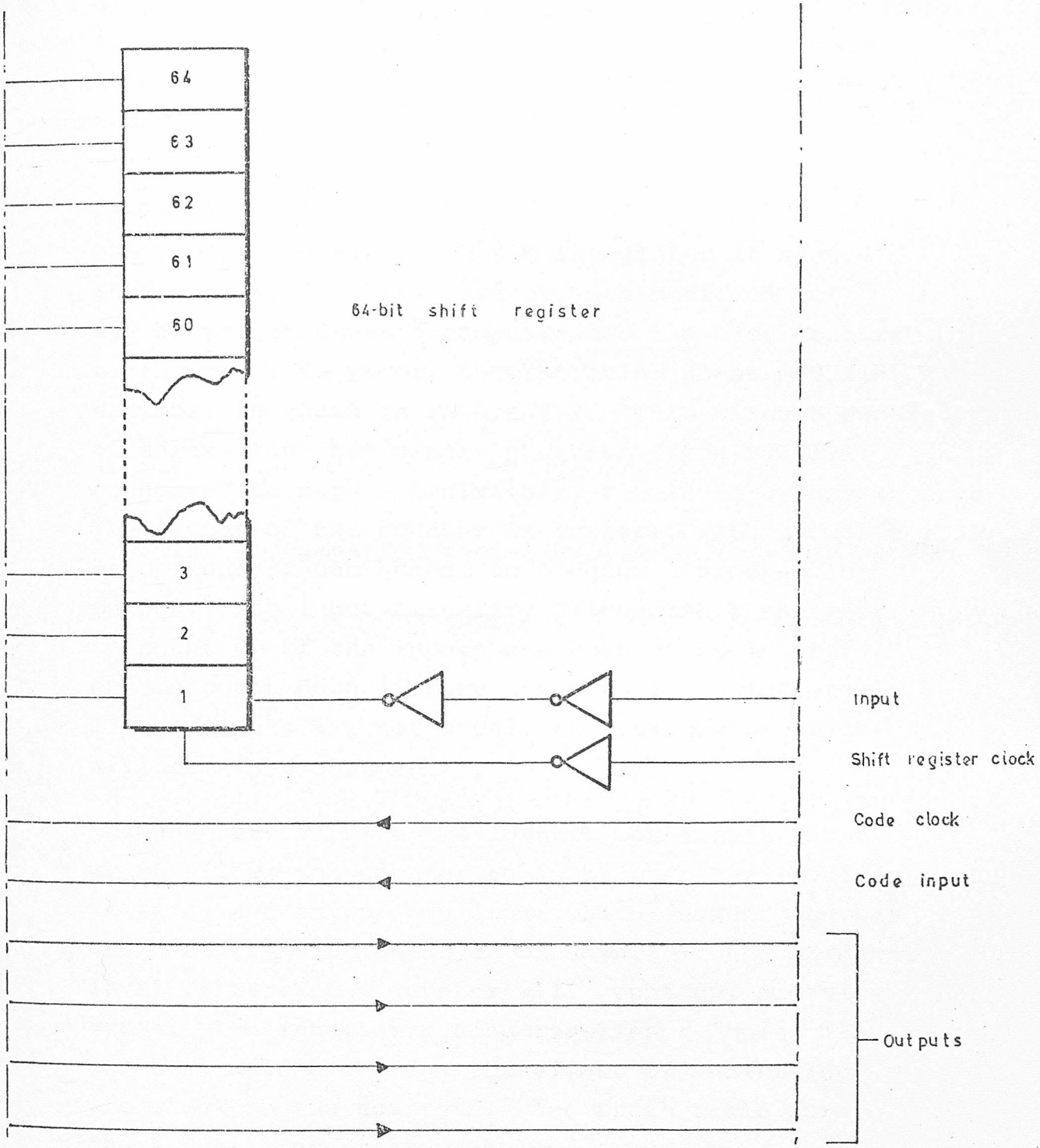


Figure 3.9 Automatic testing interface

CHAPTER 4

BASIC COMPUTING ELEMENTS

This chapter considers the basic elements of the stochastic computer. In particular we will describe the design of the integrator, the ADDIE, and the PRBS random number generator.

4.1 Integration [3,8]

The block diagram of a basic integrator is shown in Figure 1.6. This, however, was modified for use in the stochastic computer and the block diagram of the new integrator, incorporating these modifications, is shown in Figure 4.1. This circuit consists of three, four bit binary counters (type SN74191) connected to form a twelve bit, ripple through counter. The output of the counter is compared with a twelve bit pseudo-random number to produce a stochastic output. The input circuitry allows the integrator to count up if the inputs are both at logic one, and to count down if they are both at logic zero. If the inputs are not equal, the integrator state will remain unchanged.

The counters require two inputs; an enable input which allows the counter to count when the level is at zero, and an up/down input which decides whether the counter is to count up or down. If the up/down level is zero, the counter will count up, and vice versa. The integrator also requires a 'hold' line, which if held high, has the effect of inhibiting the clock to the counters. The truth table for these functions is shown below.

Inputs /

Inputs		Required Outputs	
A	B	Up/down (U/D)	Enable (E)
0	0	1	0
0	1	Ø	1
1	0	Ø	1
1	1	0	0

By inspection, the required logic expressions are:

$$U/D = \bar{A} \qquad \bar{E} = \bar{A}\bar{B} + AB$$

$$\text{or } E = A \oplus B$$

Thus the enable output E is derived from the output of an exclusive OR gate, and the up/down line is simply the inverse of the input A. The clock to the system is routed through a NAND gate which serves a dual function. The gate provides the inverted clock pulse form required by the SN74191 integrated circuit, and also enables the clock to be gated by the hold line.

If we now consider equation 1.2,

$$E_{out} = E_O + \frac{1}{NT}\int_O^t (E_A + E_B)dt$$

It may be seen that the output of the integrator is dependent upon 1/T which is the clock period, and 1/N which is the inverse of the number of states in the integrator. Since for an n-bit integrator, $N = 2^n$, scaling of the integrator may be simply achieved by varying n. Thus for example, a reduction of n by one has the effect of dividing N by two. This fact can be used to provide a scaling function for the integrator. It can be accomplished practically by the modification to the basic integrator circuit, shown /

shown in Figure 4.2. The first four bits of the integrator are replaced by discrete J-K flip-flops. (type SN7476). This allows the blocking off of one to four bits of the counter so that the integrator may have 8, 9, 10, 11 or 12 bits.

The number of bits that the integrator uses is determined by the code pattern set up in a four bit serial in/parallel out shift register. As with the patching system, these shift registers are loaded serially from the PDP8/E, and thus the scaling factor for each integrator may be entered at the keyboard console. The code patterns and resulting scaling factors are shown in Figure 4.3. Only the first four bits of the integrator are shown in Figure 4.2, since the remaining eight bits are not affected, and only require an enable and up/down signal from the preceding stages. These scaling factors are used to compensate for the normalised outputs produced by some circuit elements, as explained previously in Chapter One.

4.2 PRBS Generation [3,14]

The generation of pseudo-random binary sequences for use as twelve bit random numbers is of essential importance in the design of DISCO. Since the system is so large, and modular, it is necessary to have a twelve bit random number available at every position in the machine. These numbers must not be correlated in any manner, that is, they must be completely independent.

The basic principles of PRBS generation were discussed in Chapter One. There are several methods possible for the generation of large quantities of random numbers. One solution would be to have a large PRBS generator, /

generator, and generate twelve individual sequences for each number by means of cascaded exclusive OR gates. This obviously would involve a large number of logic gates and complex wiring problems, although one advantage of this system would be that bit independence is provided over a considerable time period.

Another alternative technique would be to provide an individual PRBS generator for each twelve bit number. Thus each PRBS generator would have the same number of bits, but would start in a different initial state. For example, consider a PRBS generator 31 bits long. The number of states that this generator may take is $2^n-1 = 2^{31}-1$. The minus one is included since the all zero state is a null condition.

If there are thirty two numbers required, then each PRBS generator must start $(2^{31}-1)/2^5$ states from the previous one. This would ensure that no correlation would occur until after $(2^{31}-1)/2^5$ clock pulses. If the clock rate was one 1MHz, then this would be equivalent to a time of

$$\frac{2^{31}-1}{2^5} \times \frac{1}{10^6} \text{ seconds}$$

that is, approximately 67 seconds. This would be sufficient time for most computations to be completed before correlation effects occurred. Unfortunately, this method would also require a considerable amount of logic gates and shift registers. Another disadvantage would be the difficulty in ensuring that each PRBS generator started in the correct state.

The actual method finally selected for use in DISCO consists of fifteen identical modules and one master generator. The schematic diagram of this system is shown /

shown in Figure 4.4. The master generator board contains the main thirty one bit PRBS generator from which is derived the $2^{31}/32$ delay using cascaded exclusive OR gates. This delay is routed into a second thirty one bit shift register whose outputs are exclusive OR'd in exactly the same manner as the main shift register to produce the delay $(2^{31}/32) \times 2$. As shown in the schematic diagram, this arrangement is duplicated to obtain 32 sequences.

Practically, it is possible to have two banks of exclusive OR gates on one verocard. Thus only sixteen cards are required for the complete PRBS system. The twelve bit random numbers are the first twelve bits of each shift register, with the most significant bit inverted. This is to give a small degree of negative correlation to improve computational accuracy.

Having selected this basic configuration for PRBS generation, the problem of delay generation must still be solved, that is, in what manner are the exclusive OR gates to be connected to generate the required delays from the basic shift register. The method of calculating delays may be explained using the characteristic equation of the PRBS generator. The feedback on the PRBS generator is derived from the modulo two addition of bits 31 and 4, so that the equation is

$$D_0 = D_{31} \oplus D_4$$

where D_0 is the input to the generator, D_{31} and D_4 are delays 31 and 4 respectively. Rearranging the equation gives

$$D_{31} = D_0 \oplus D_4$$

ie

$$D_n = D_{(n-31)} \oplus D_{(n-27)}$$

This /

This equation may be used to find any delay n. For example, delay 87:

$$D_{87} = D_{(87-31)} \oplus D_{(87-27)}$$

$$= D_{56} \oplus D_{60}$$

We now have to use the equation repeatedly until the delays are less than or equal to 31. Therefore,

$$D_{56} = D_{(56-31)} \oplus D_{(56-27)}$$

$$= D_{25} \oplus D_{29}$$

and

$$D_{60} = D_{(60-31)} \oplus D_{(60-27)}$$

$$= D_{29} \oplus D_{33}$$

and

$$D_{33} = D_2 \oplus D_6$$

Hence

$$D_{87} = D_2 \oplus D_6 \oplus D_{25} \oplus D_{29} \oplus D_{29}$$

Since

$$D_{29} \oplus D_{29} = 0$$

we can ignore these delays.

Therefore

$$D_{87} = D_2 \oplus D_6 \oplus D_{25}$$

This algorithmic technique was programmed on a mini-computer and run satisfactorily. The program listing is given in Appendix Three. However, several disadvantages are involved with this method. Since the program was written in an interpretive language, FOCAL, the execution time /

time increased rapidly as the magnitude of the input increased. This resulted in a computation time of approximately five minutes for an input involving a delay greater than 10^5.

We require to find the delays to be exclusive OR'ed to produce delays around 2^{26}, so that this method was not feasible. Another disadvantage was the fact that FOCAL could not operate accurately with integers greater than 999999.

The PDP8/E system had only 4096 x 12 bits storage capacity, so that the quantity of numbers that could be stored was limited. The method previously outlined requires a large amount of storage, since the number of variables to be stored approximately doubles at each succeeding step. In some cases there are cancellations which reduce the storage, but not significantly.

It was noted that at each level of 2^n, as shown in Figure 4.5, where n is a positive integer, the number of delays to be gated together reduced to two. In the pyramid structure shown in Figure 4.5, level 0 represents the initial delay, in this case, 2^{26}. The delays on the other levels are found by subtracting 31 from the previous delay as we move diagonally left down the pyramid, and subtracting 27 as we move diagonally right down the pyramid. This reduction to two delays at certain levels was utilised to allow the program to start at the lowest pair using the equations,

$$d_1 = D - 2^n \times a \qquad \text{where } d_1 > 0$$
$$d_2 = D - 2^n \times b$$

where a = 31, the length of the shift register, and b = 27, the feedback connection. The objective is to find the smallest $d_1 > 0$. This method reduced the computing time and storage required considerably but was not yet the ideal solution.

The /

The next stage was to use these delays d_1 and d_2 as original delays, and use the previous equations. This was repeated continuously, each new pair of delays being utilised as original delays, until every d could be tapped off directly from the master PRBS generator.

Cancellations were automatically made by storing an array of values ST(1), ST(2),...., ST(31) corresponding to each bit of the generator. When a delay d was found in the correct range the corresponding value of ST was incremented by one.

That is, $ST(d) = ST(d) + 1$ where $1 \leqslant d \leqslant 31$

When every d was in the range one to thirty one, the printout routine in the program would print a value I if the value ST(I) was odd. This meant that the cancelled delays were ignored. By using this cancellation method, the storage required was reduced dramatically, but was not yet of the order to enable large delays to be found.

The final modifications to the program resulted in minimum storage capacity, although the computing time was increased slightly. This enabled the program to be run with large numbers, of the order required by the DISCO system.

After computing the first d_1 and d_2 from the original delay, instead of immediately forming four new delays from d_1 and d_2, d_2 was stored, and d_1 used to form two new delays. This process was repeated, d_1 forming two delays, and d_2 being stored, until d_1 was in the required range, that is, 1 to 31. When this occurred, the corresponding d_2 was used as an original delay, and the process repeated. If it occurred that d_1 and d_2 were both in the required range, the previous d_2 would be used. Automatic cancellation took place as before. The flowchart for this final program is shown in Figure 4.6, and the FORTRAN listing given in Appendix Four.

This /

This explanation is helped by the numerical example shown in Figure 4.7.

4.3 Output Interface Design (ADDIES) [3,8,15]

The ADDIE is the element used to convert a stochastic sequence into a deterministic twelve bit binary number. It was discussed briefly in Chapter One, where it was shown that the output was exponentially weighted with respect to past inputs. A more detailed diagram is shown in Figure 4.8.

The counter in the ADDIE, which is a cascade of three SN74191 up/down counters, will count either up or down, or stay in the same state, depending on the state of the input and the feedback. If the input and feedback are both logic one then the ADDIE will count up. If they are both logic zero, then it will count down. Any other condition will result in the counters being disabled and remaining in the same state. The counter is continuously compared with a twelve bit random number using three SN7485, four bit comparators. The output from these comparators will be a logic one if the state of the counter is greater than the random number, and logic zero if the state of the counter is less than or equal to the random number.

It can be seen that if the counter is almost full, there will be very few logic 0's at the output of the comparator. However, the output is negated, and so the feedback will tend to be almost all logic 0's. This will tend to stop the counter counting up. A similar effect occurs when the counter is almost empty, but this time tending to stop the counter counting down.

As previously mentioned, a modification was made to the ADDIE, by using binary rate multipliers (BRM's) as the feedback element.[8] The modified circuit is shown in Figure 4.9.

The /

The BRM gives an output rate proportional to the value of the input, which is in this case the output from the up/down counters. The output from the BRM however, is gated internally by the clock, and this is not suitable for this application, since we require a full-width pulse. The circuitry also shown in Figure 4.9, was used to lengthen the output pulses from the BRM. It consists of an S-R flip-flop constructed using NAND gates, which is set by the output from the BRM, and only reset if both the output and the inverted clockpulse are at zero. An example of its operation is shown in Figure 4.10.

Using the BRM has the advantage that it removes the necessity for a twelve bit random number. This introduces deterministic feedback, and so improves the accuracy of the ADDIE.

It was necessary to have some means of determining the accuracy of the output, and also to test the output for variance. First, the output was converted to an analogue signal by means of a digital to analogue converter, and then plotted on an X-Y plotter. This gave a useful indication of the accuracy and variance but was a laborious procedure. A more sophisticated method was devised, whereby the output of the ADDIE under test was read and operated on by the PDP8/E minicomputer. The experimental arrangement is shown in Figure 4.11.

The object was to produce a distribution curve of the output of the ADDIE, to be displayed on an oscilloscope. A machine language program was written to accomplish this task and the flowchart for the program is shown in Figure 4.12. The assembly listing for the program is in Appendix Five.

Since /

Since it is reasonable to assume that the variance of the ADDIE will never be more than plus or minus 100 states either side of the mean value, the distribution curve displayed is limited to these values. The expected mean value is entered into the PDP8/E by means of the front panel console switches.

The twelve bit output from the ADDIE is interrogated and examined to see if it lies within the specified range of states. If it does, a location in memory corresponding to that particular state is incremented by one. If the output is outside the specified range then the reading is ignored. A routine is now entered which produces a display of the distribution curve on the oscilloscope by taking each of the 200 locations in turn and using a digital to analogue converter to provide a corresponding voltage level. This results in a continuously increasing distribution curve. Instead of producing a display for each individual reading, the display characteristics are only exhibited for every thirty samples. This has the effect of producing a flicker-free display and an improvement in the speed of the build up of the distribution curve.

The distribution curve gives a good visual indication of the accuracy of the output interface since the correct estimate of the output value is displayed in the centre of the screen, and a quick guide to variance is evident from the width of the display.

If a permanent record of a distribution curve is required then the program may be slowed down by means of software delays so that the X and Y outputs may be used as the inputs to a graph plotter. Some typical distribution curves for the noise and BRM ADDIE structures for a range of input probabilities are shown in Figures 4.13 to 4.16.

It /

It can be seen that there is a decrease in variance for the BRM ADDIE due to the deterministic feedback. The discontinuity in the distribution curves for the BRM ADDIE is a characteristic of the BRM. It is due to a change of phase of the output sequence, when the input to the BRM changes from a 'one-all-zeros' state to a 'zero-all-ones' state.[8]

We have seen from the distribution curves that there will be certain variance about the mean value. A more accurate method of estimating the mean value was required, and so a program was written to take a specified number of readings and form an average. The result was also converted to a real value by the transformation $E = V(2p(on) - 1)$ where V is the maximum value of the real variable, and p(on) is the reading taken from the output of the ADDIE. Figure 4.17 shows typical errors for a range of input probabilities, for three ADDIE configurations. These results were obtained by taking 100 readings of the ADDIE states and evaluating an average. The errors are expressed as a percentage of the full scale value V. The third ADDIE configuration, that of the BRM with negative correlation, was obtained by using a negatively correlated random number as the PRBS input to the comparator. The generation of this number was accomplished as shown in Figure 4.18. A single PRBS sequence is clocked along a serial in parallel out shift register, the outputs all being negated except the most significant bit.

The tests previously described refer only to the steady state response of the ADDIEs. The response to various step inputs was also investigated. Some results are shown in Figure 4.19. These curves were obtained using a 12 bit digital to analogue converter and an X-Y plotter. It can be seen that the response time of the BRM ADDIE is almost identical to that of the noise ADDIE.

The /

The effect of varying the number of bits in the ADDIE counters was also investigated. As expected, if the number of bits in the counters were decreased, then the response time decreased, and the accuracy deteriorated. The results for three lengths of counter are shown in Figure 4.20. The frequency response of the ADDIEs were also measured for varying clock frequencies, and the results are shown in Figure 4.21. It may be seen that the BRM and noise ADDIE are almost identical in their response characteristics.

The BRM ADDIE shows an improvement in variance over the noise ADDIE, without any deterioration in bandwidth, and the BRM ADDIE was therefore chosen as the output interface to be used in DISCO.

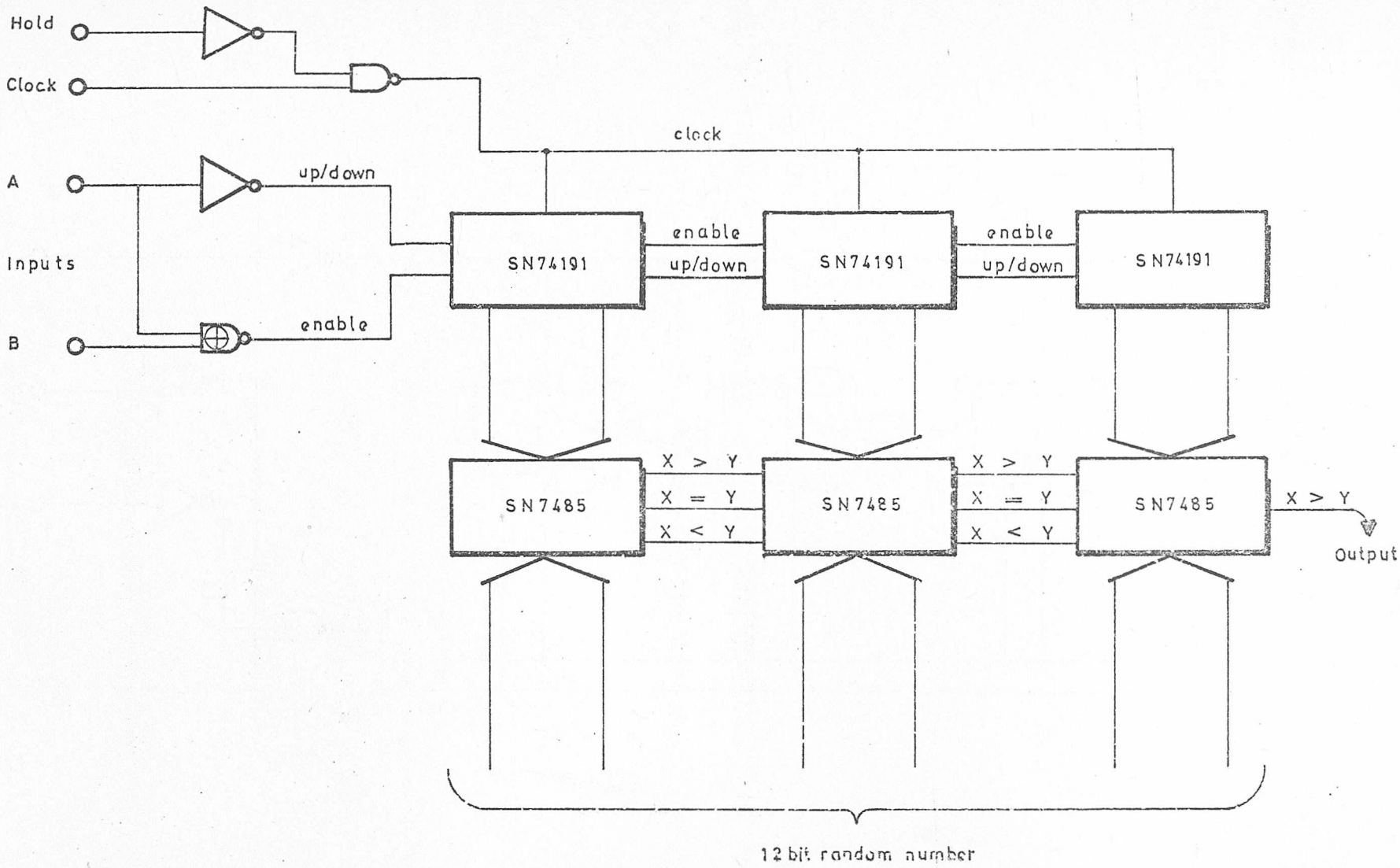


Figure 4.1 Integrator

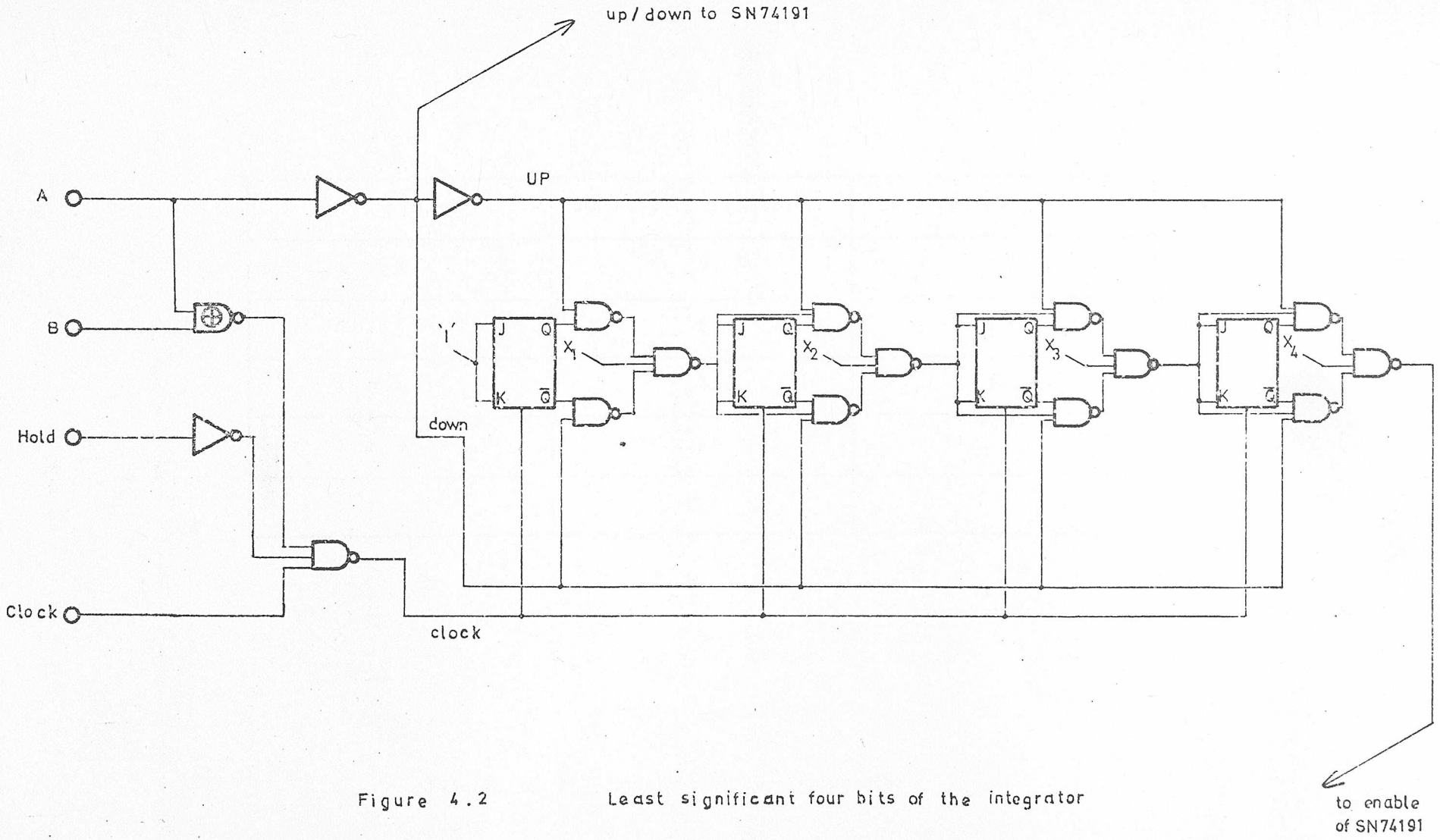


Figure 4.2 Least significant four bits of the integrator

Code	Multiplying factor	Effective length of integrator
1 1 1 1	1	12
0 1 1 1	2	11
1 0 1 1	4	10
1 1 0 1	8	9
1 1 1 0	16	8

FIGURE 4.3 Scaling of the integrator

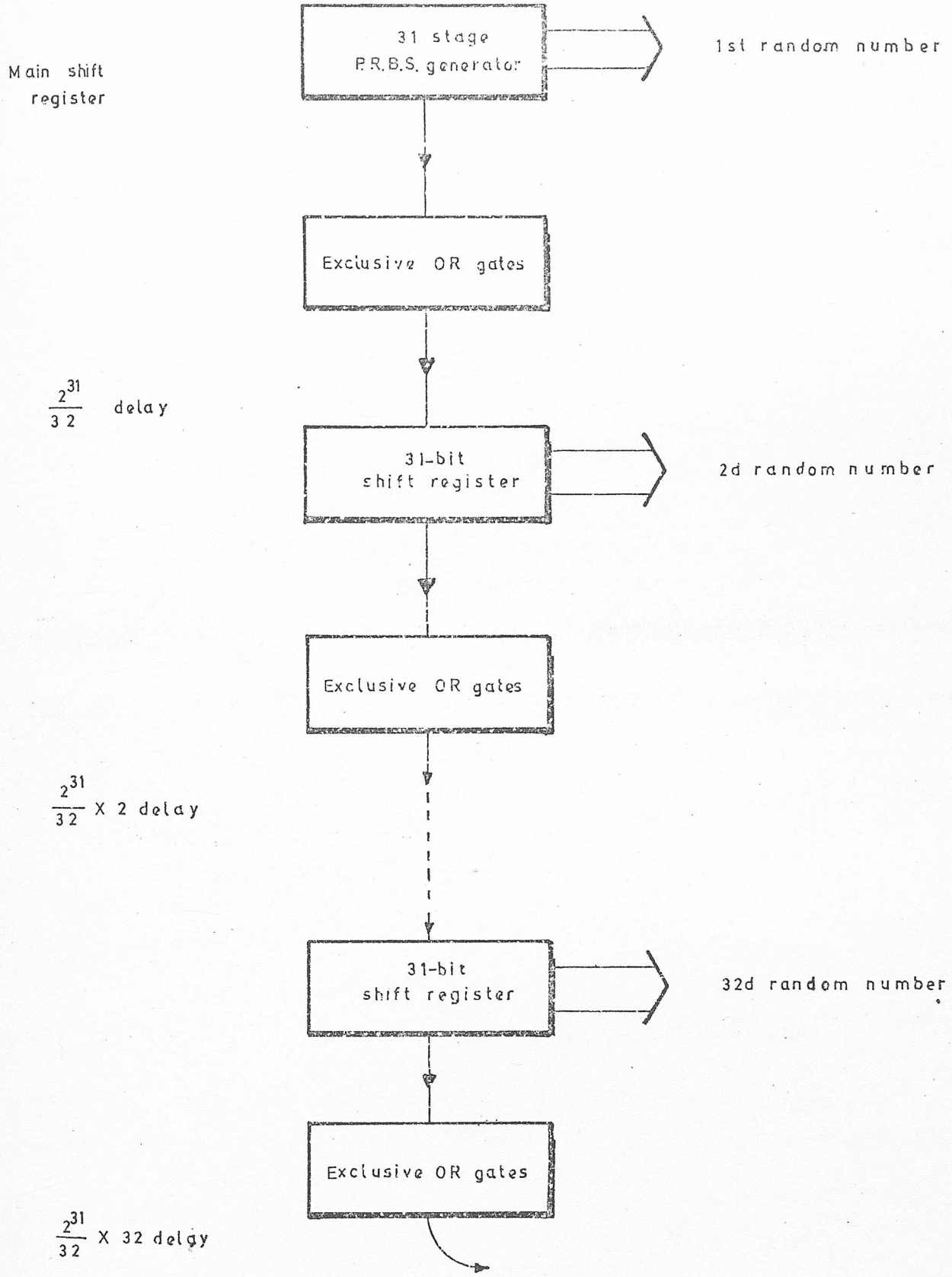


Figure 4.4 Generation of P.R.B.S. sequences

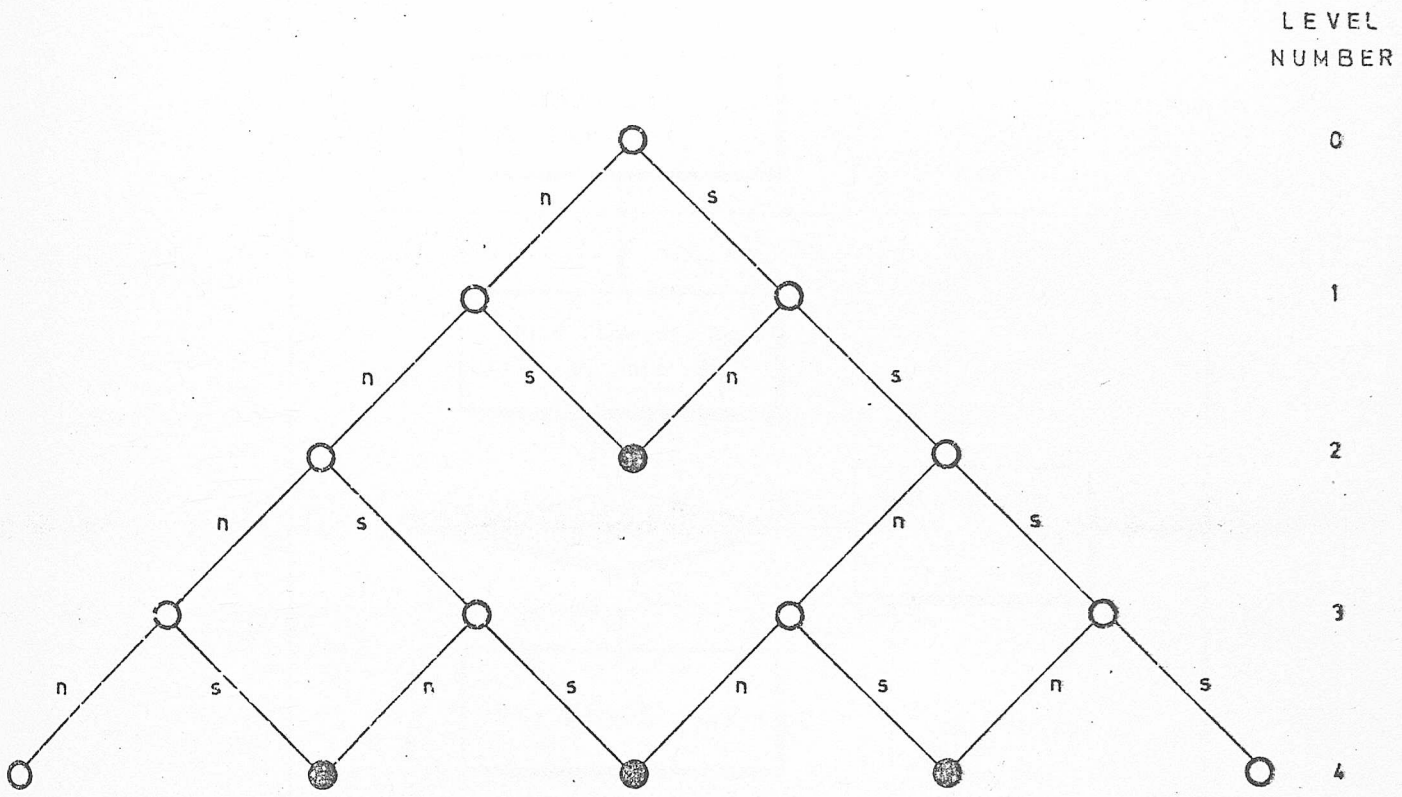


FIGURE 4.5 Delay cancellation

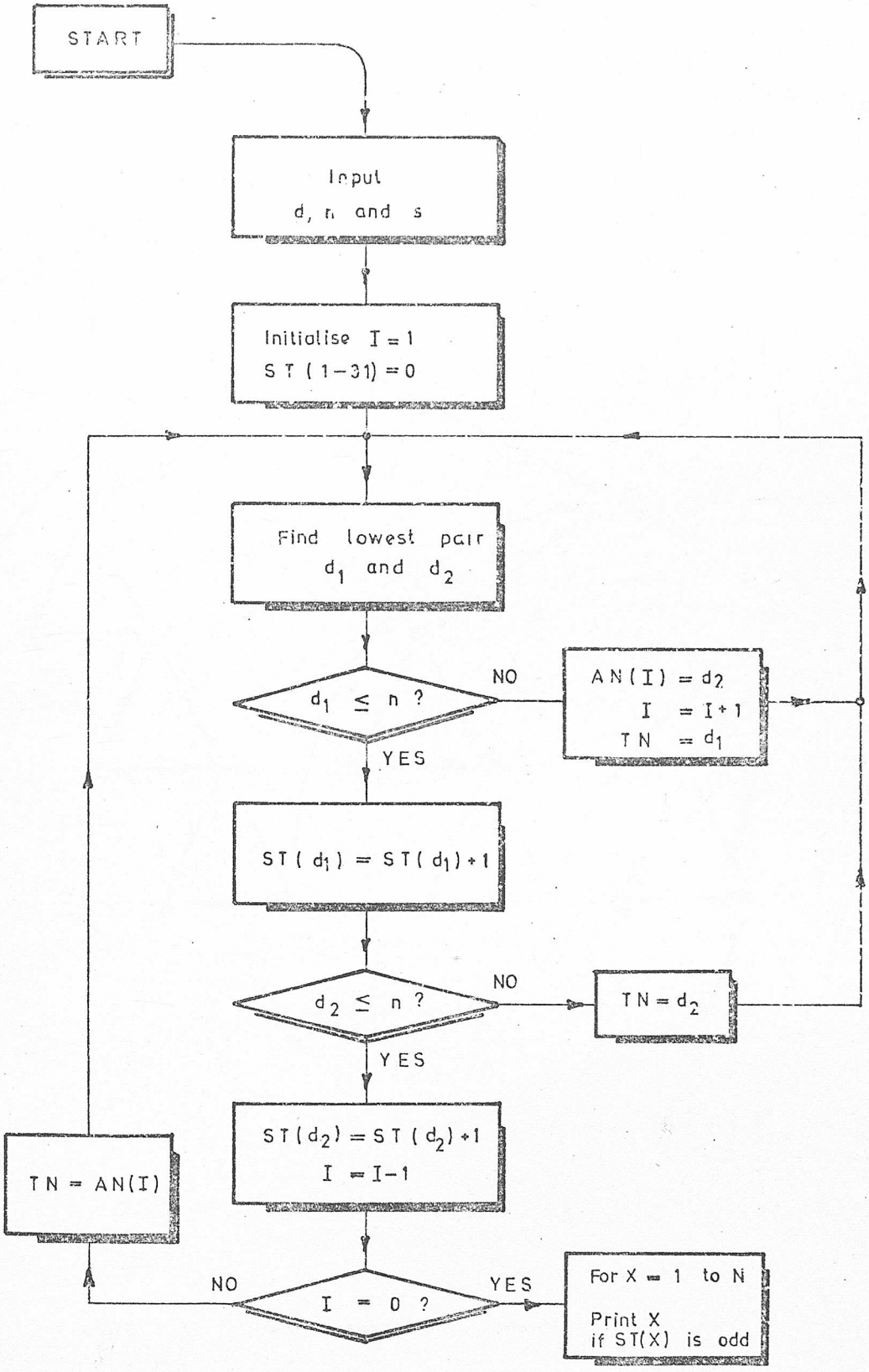


FIGURE 4.6 Delay program flowchart

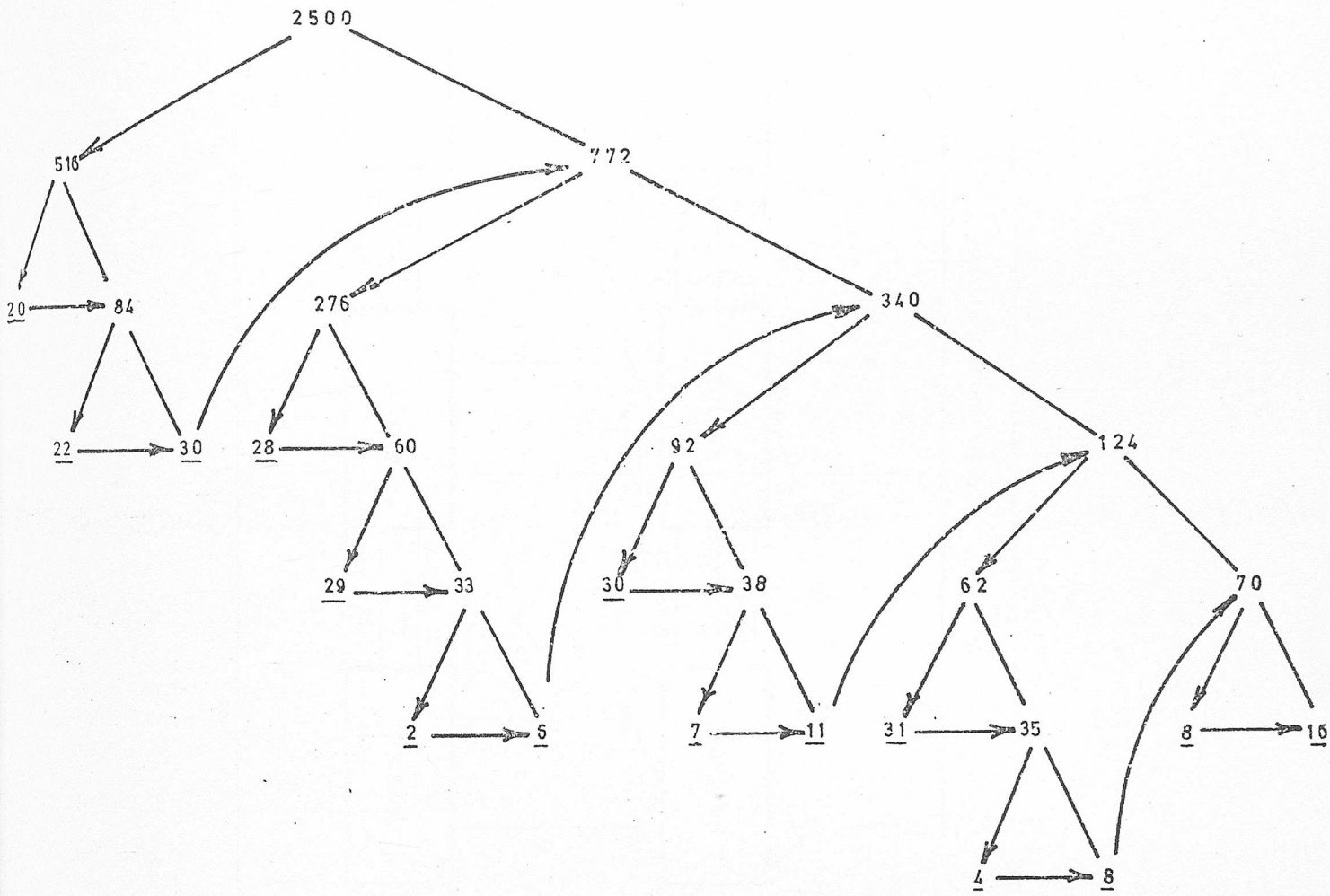


FIGURE 4.7 Delay generation example

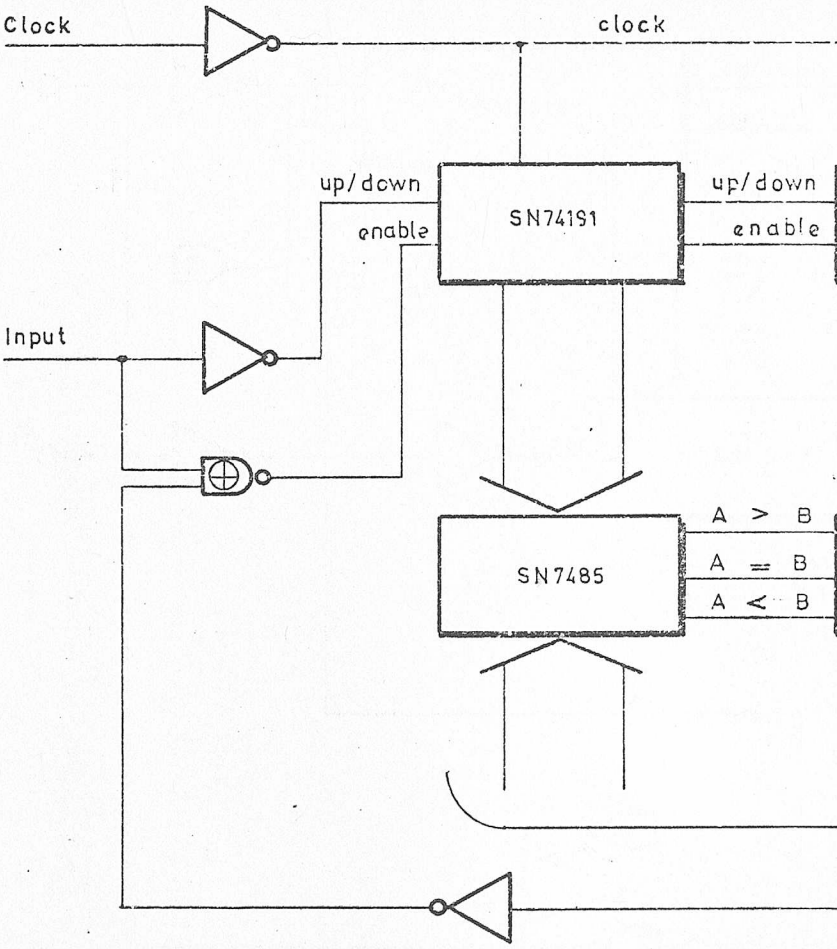
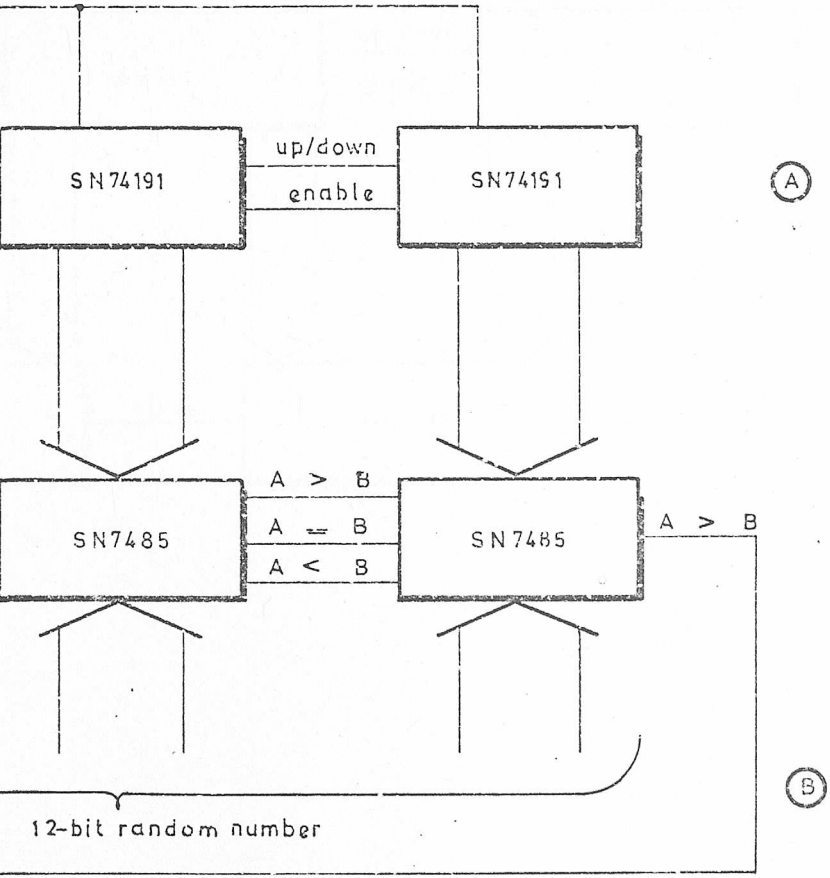

Clock
clock
up/down
enable
SN74191
up/down
enable
Input
A > B
A = B
A < B
SN7485

Figure 4.8


SN74191
up/down
enable
SN74191
A
SN7485
A > B
A = B
A < B
SN7485
A > B
B
12-bit random number

Noise ADDIE

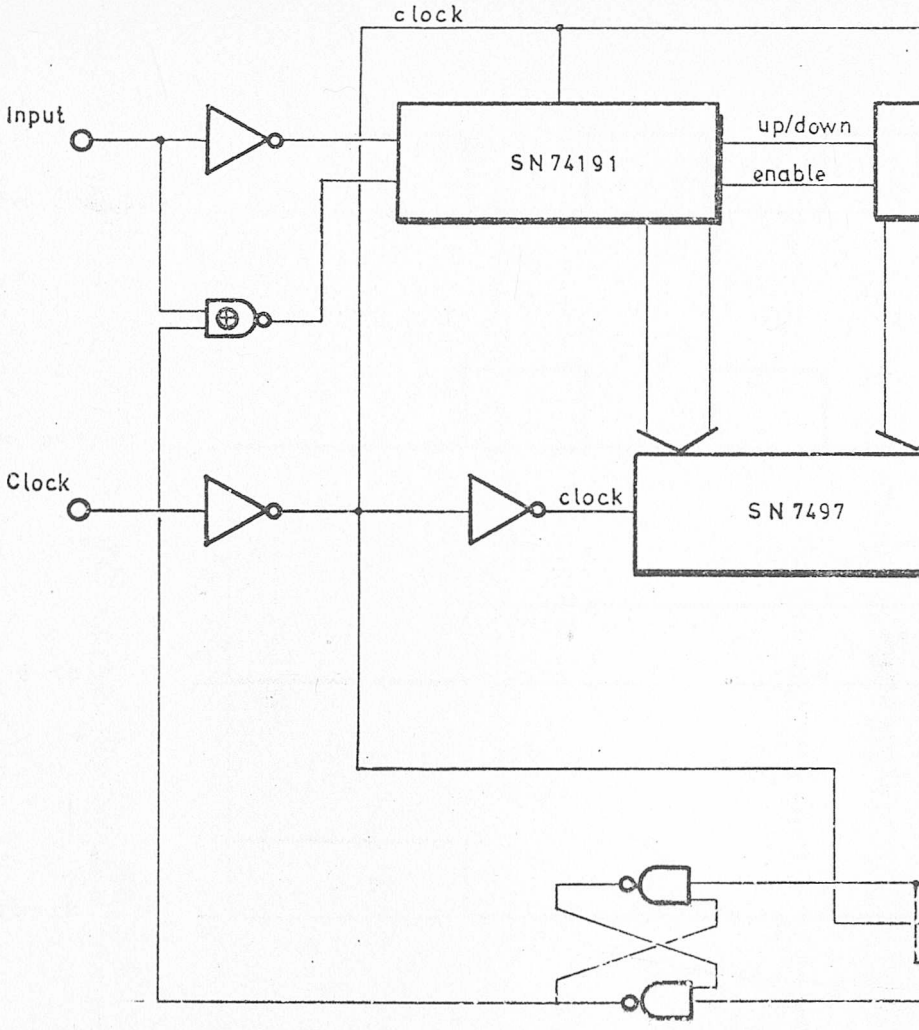
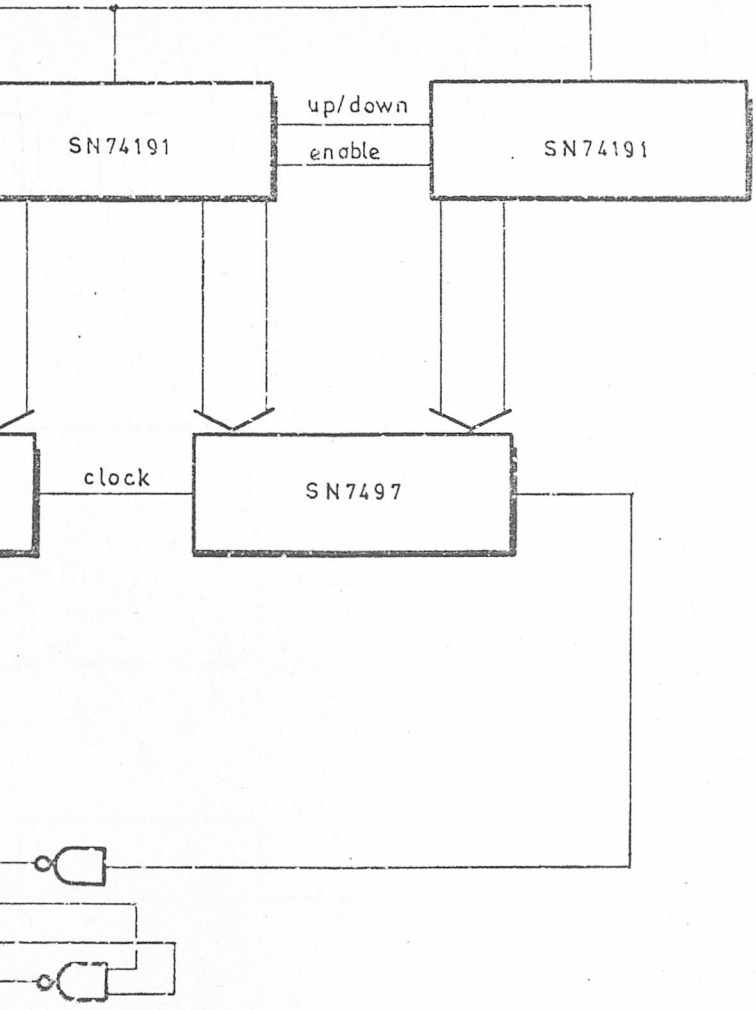
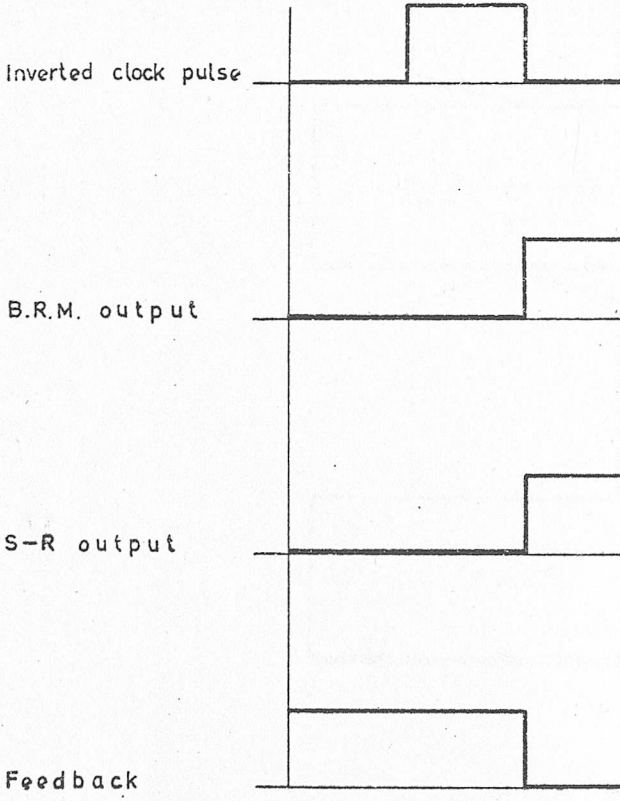


Figure 4.9 B.R.M.



ADDIE


Inverted clock pulse
B.R.M. output
S–R output
Feedback

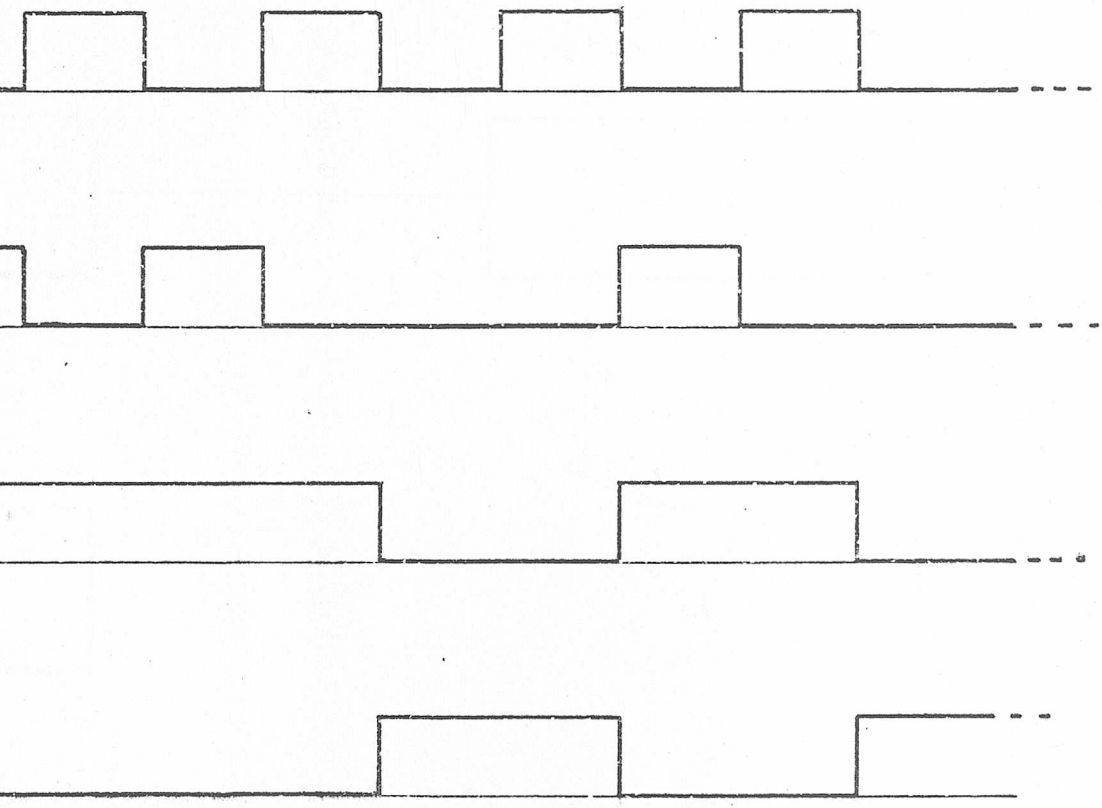


Figure 4.10 B.R.M. waveforms

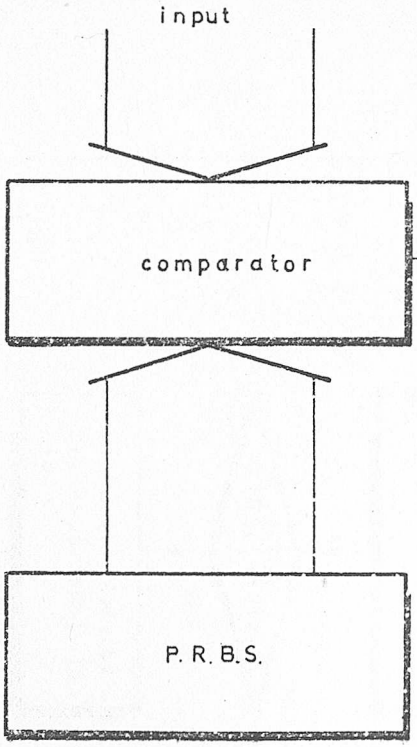
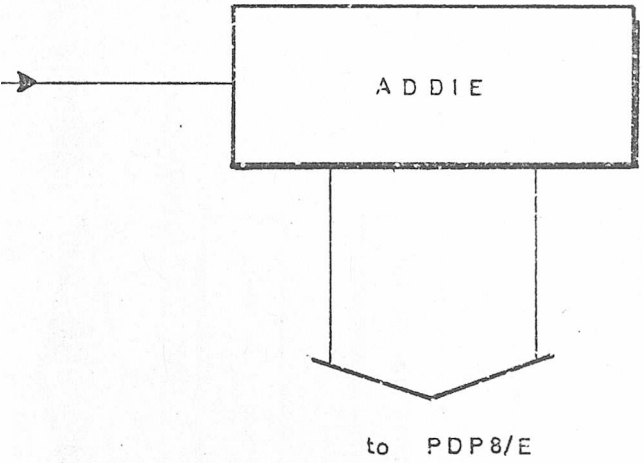


Figure 4.11



ADDIE testing configuration

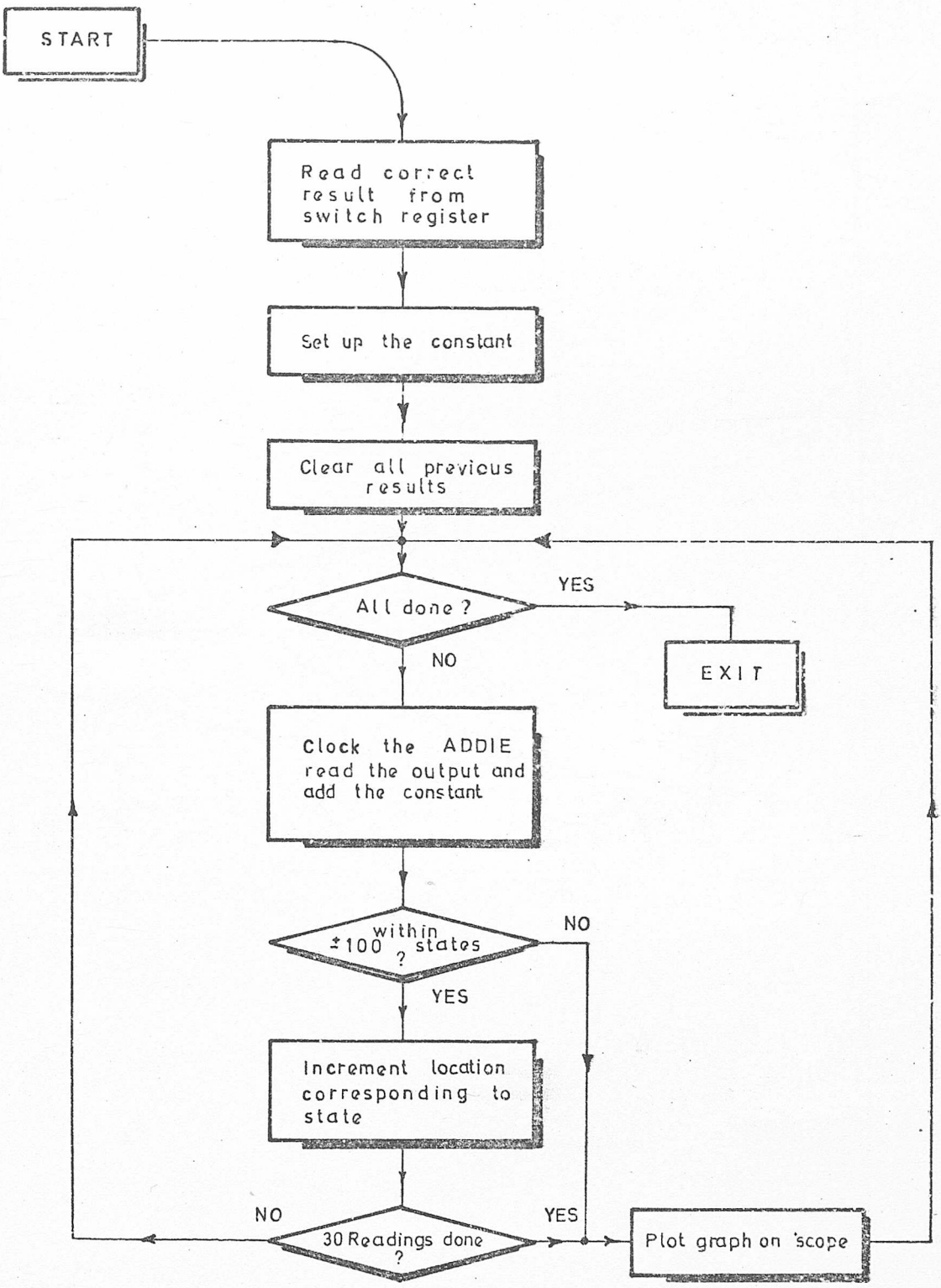


FIGURE 4.12 Distribution curve flowchart.

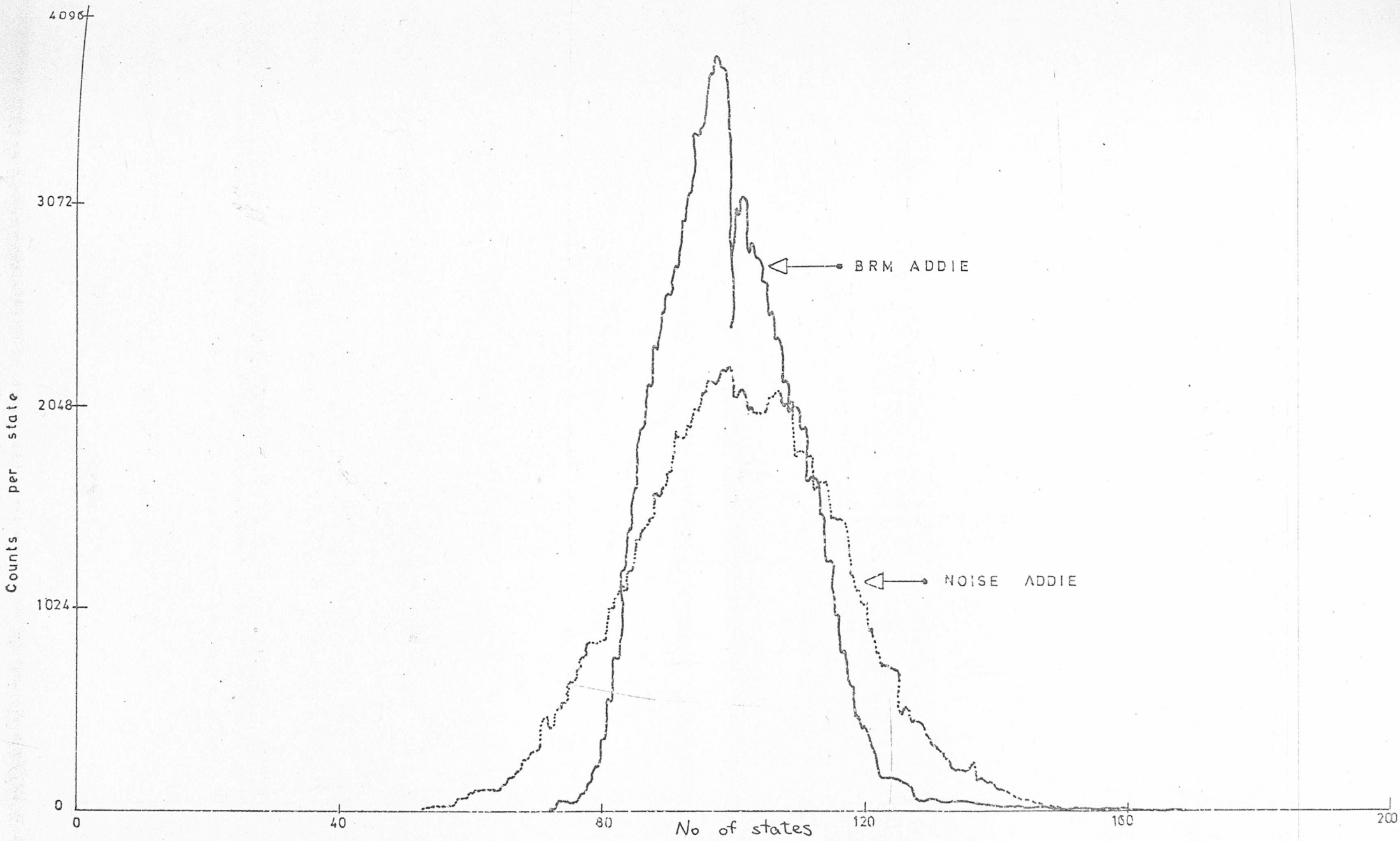


FIGURE 4·13 Distribution curves Input probability $^{1}/_{16}$

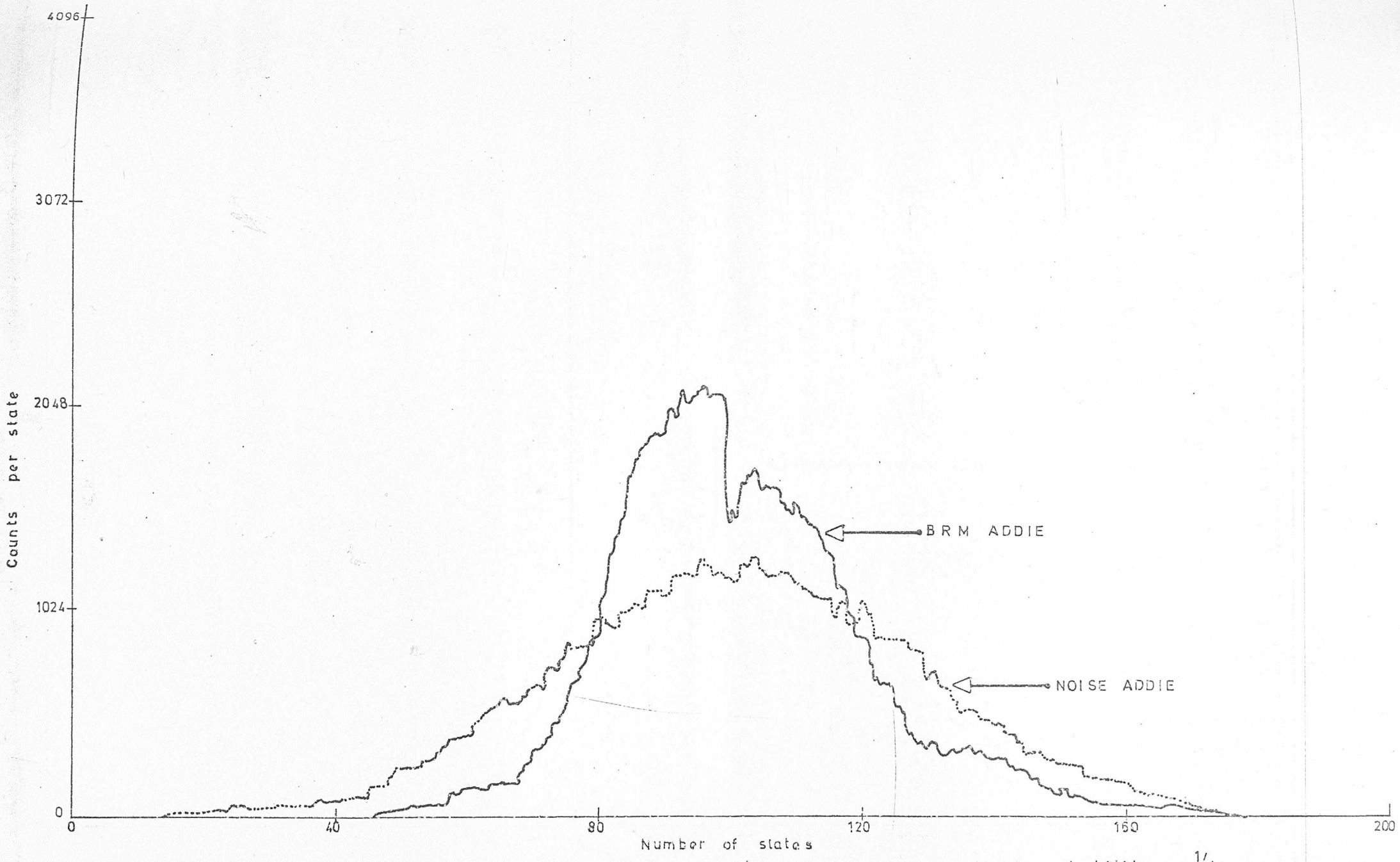


FIGURE 4-14 Distribution curves

Input probability = $^1/_4$

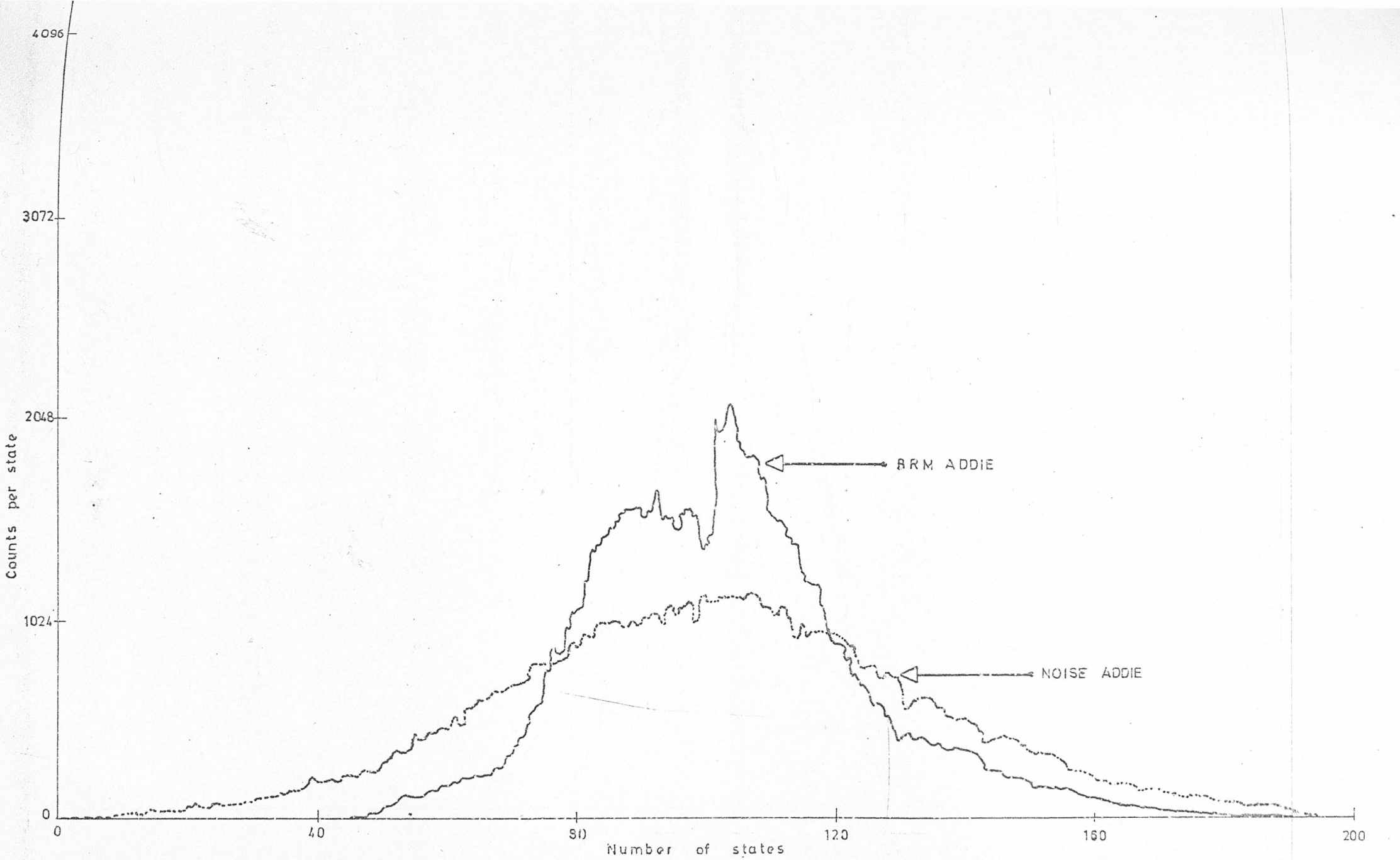


FIGURE 4·15 Distribution curves Input probability $^{1}/_{2}$

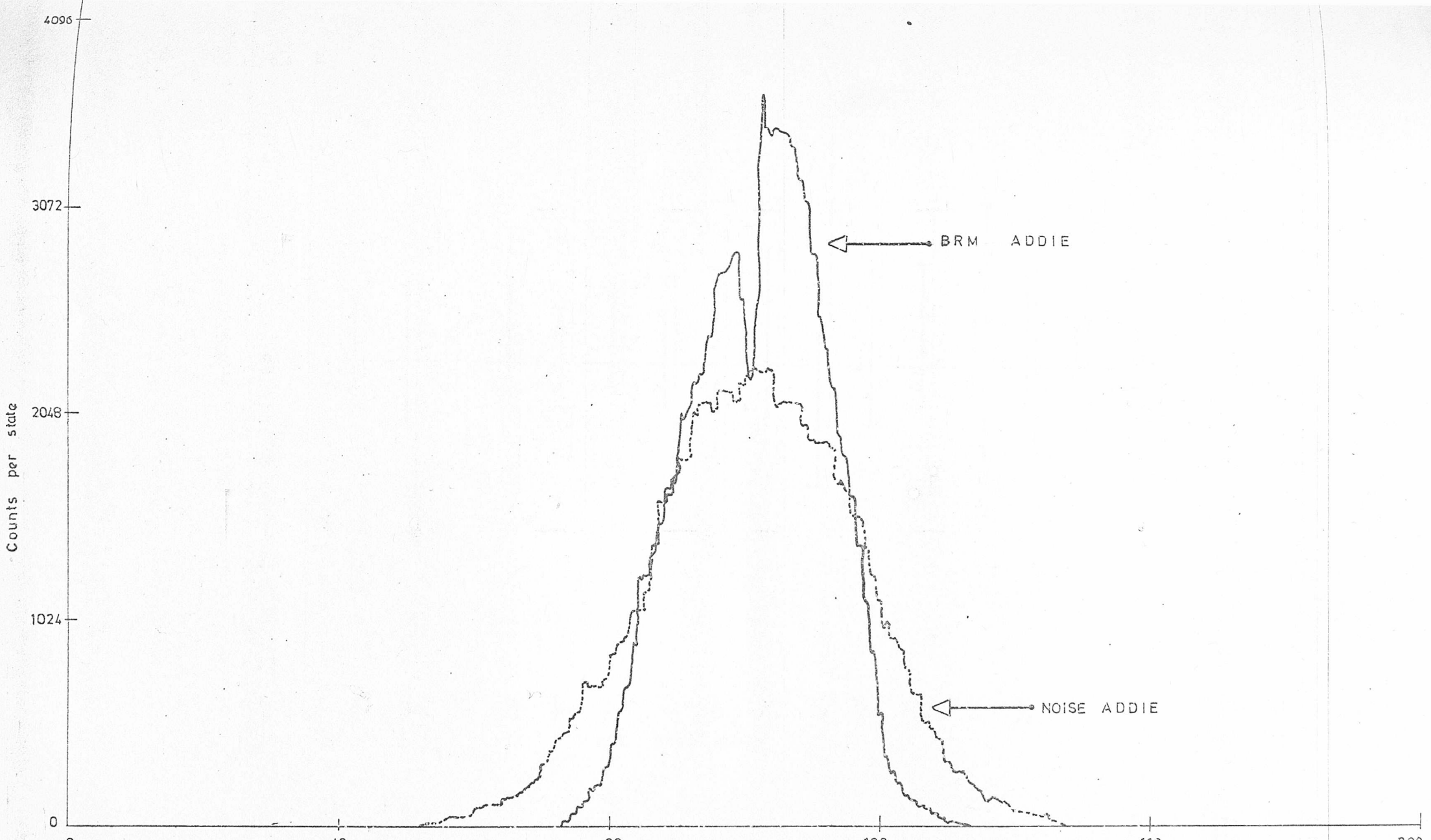


FIGURE 4.16 Distribution curves input probability = $^{15}/_{16}$

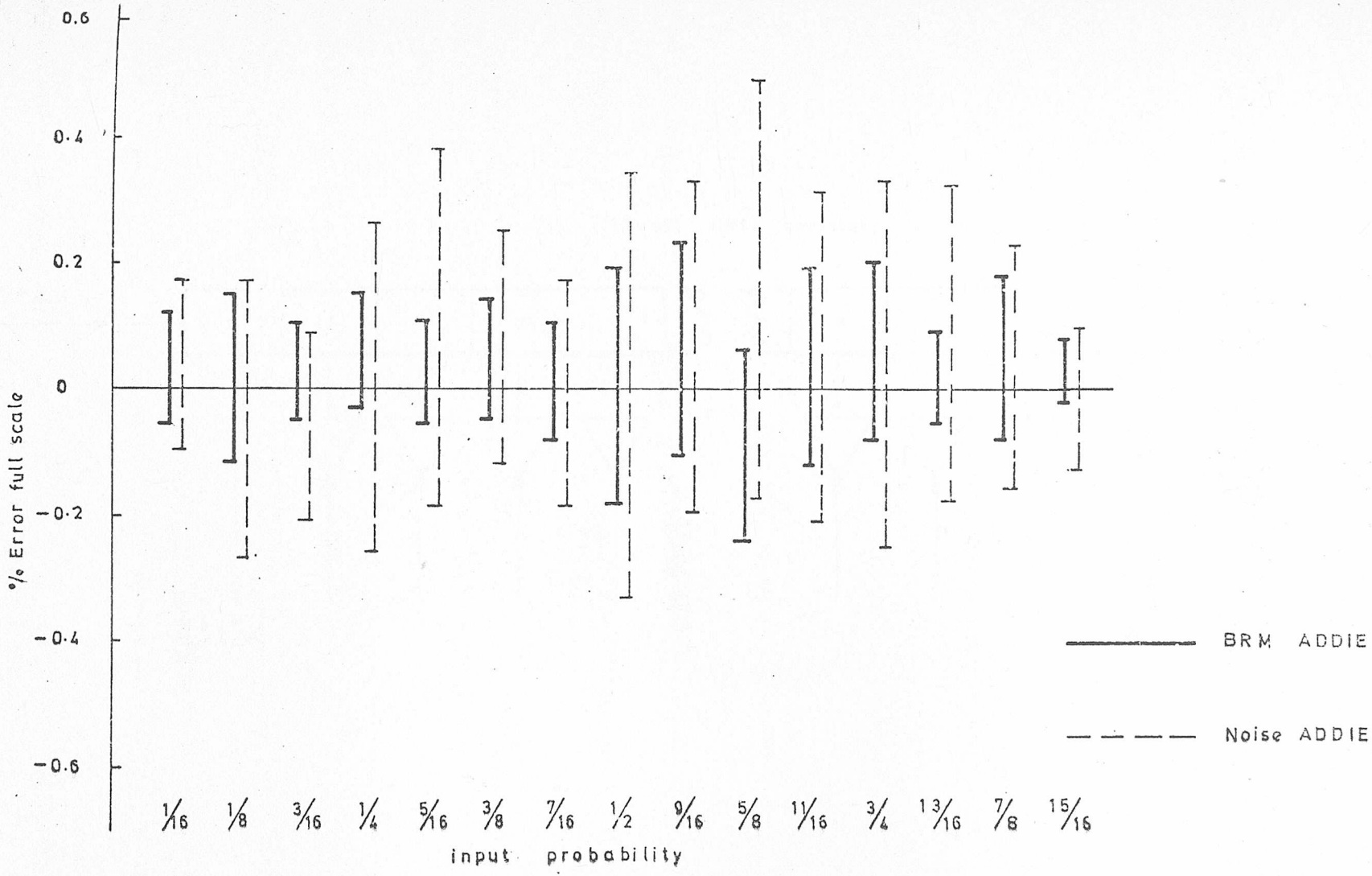


FIGURE 4.17 Errors in BRM and Noise ADDIE structures

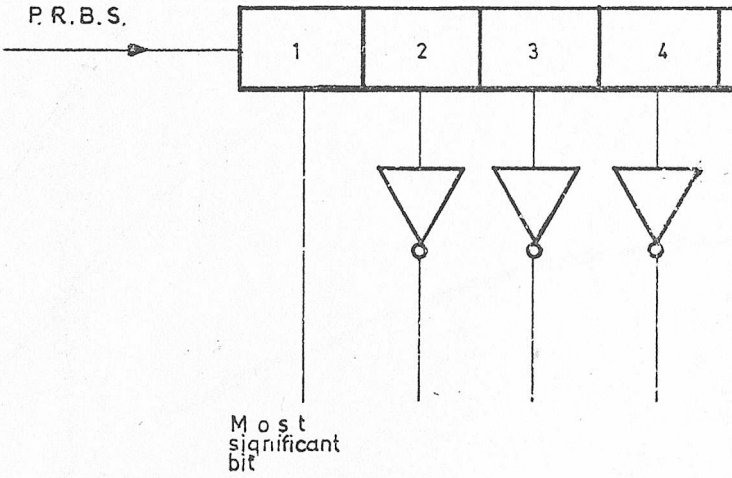
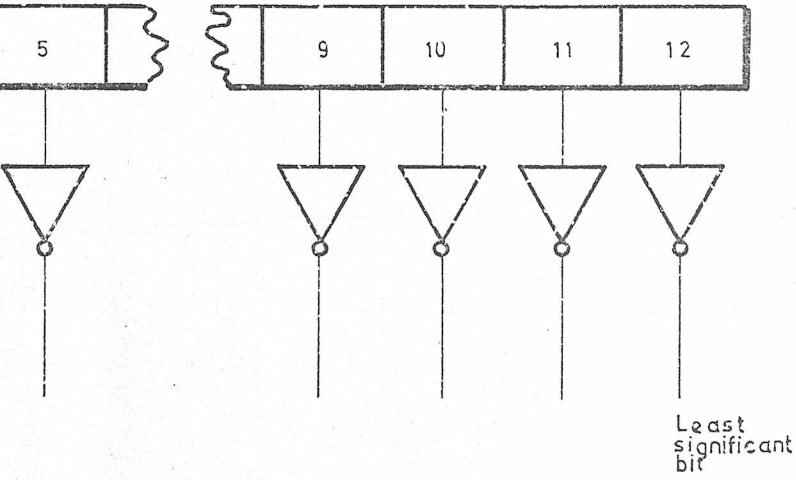


Figure 4.18

12-bit shift register



Generation of 12-bit random numbers

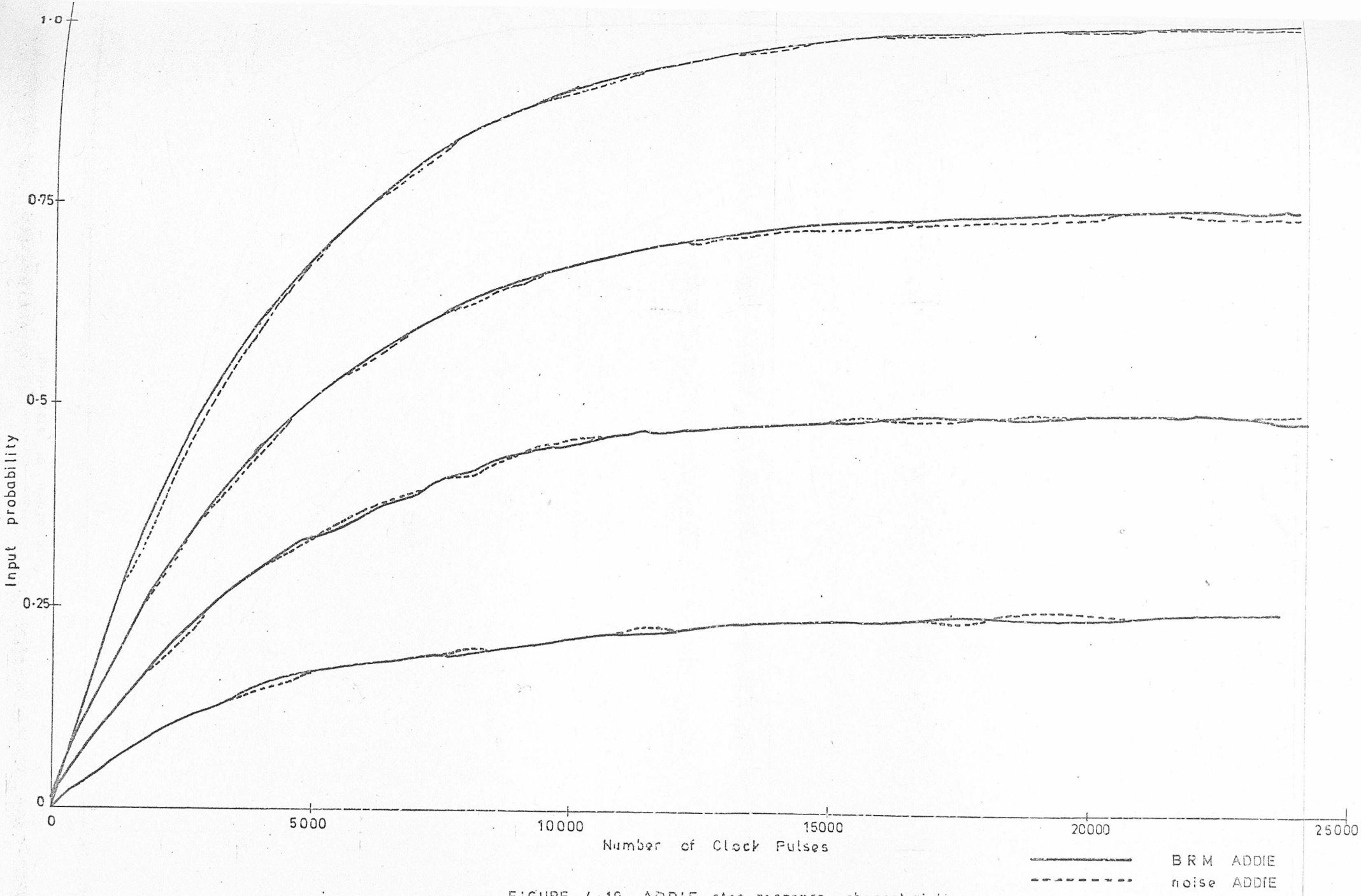


FIGURE 4·19 ADDIE step response characteristics

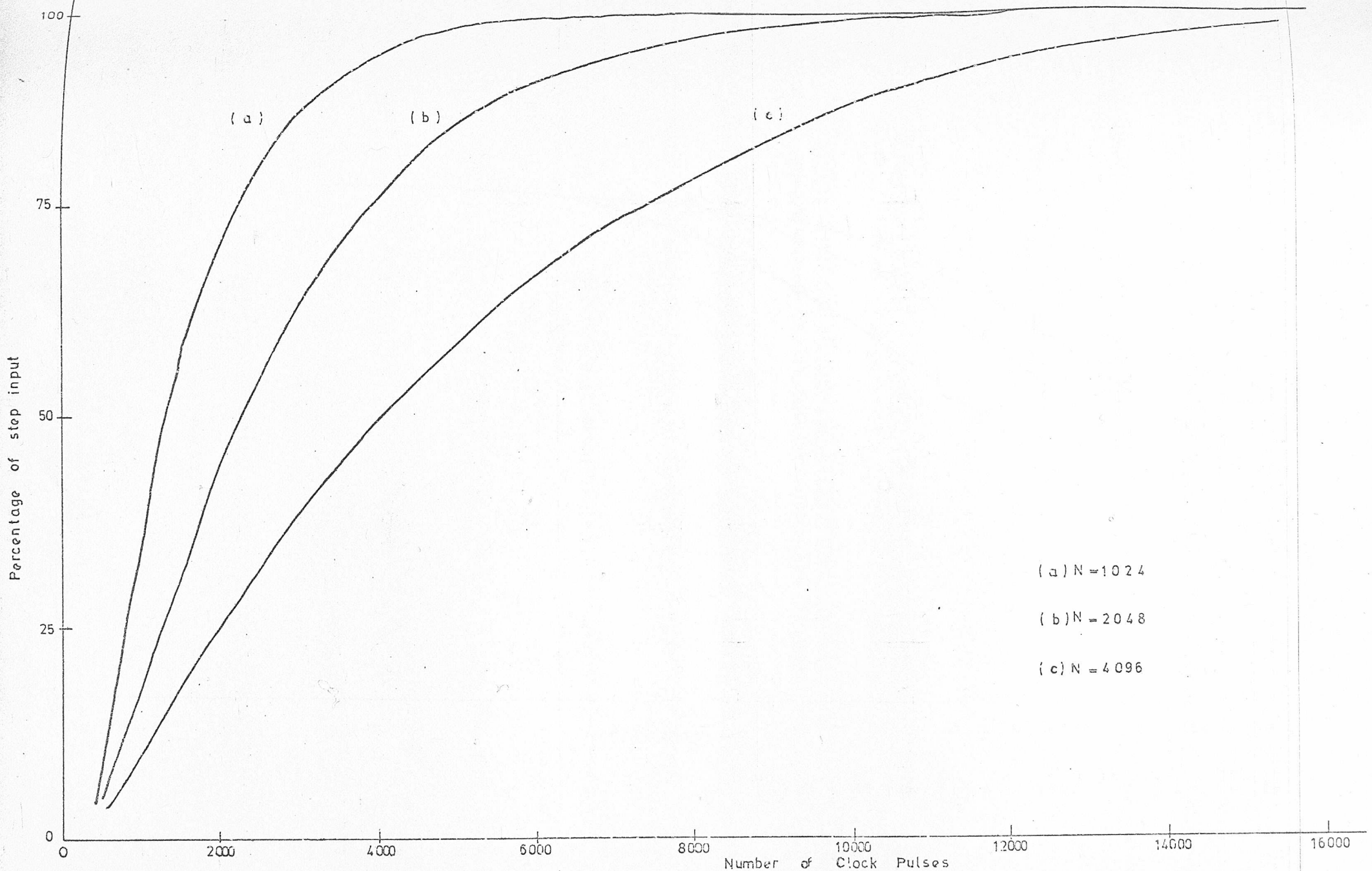


FIGURE 4-20 Noise ADDIE step response characteristics as a function of number of states

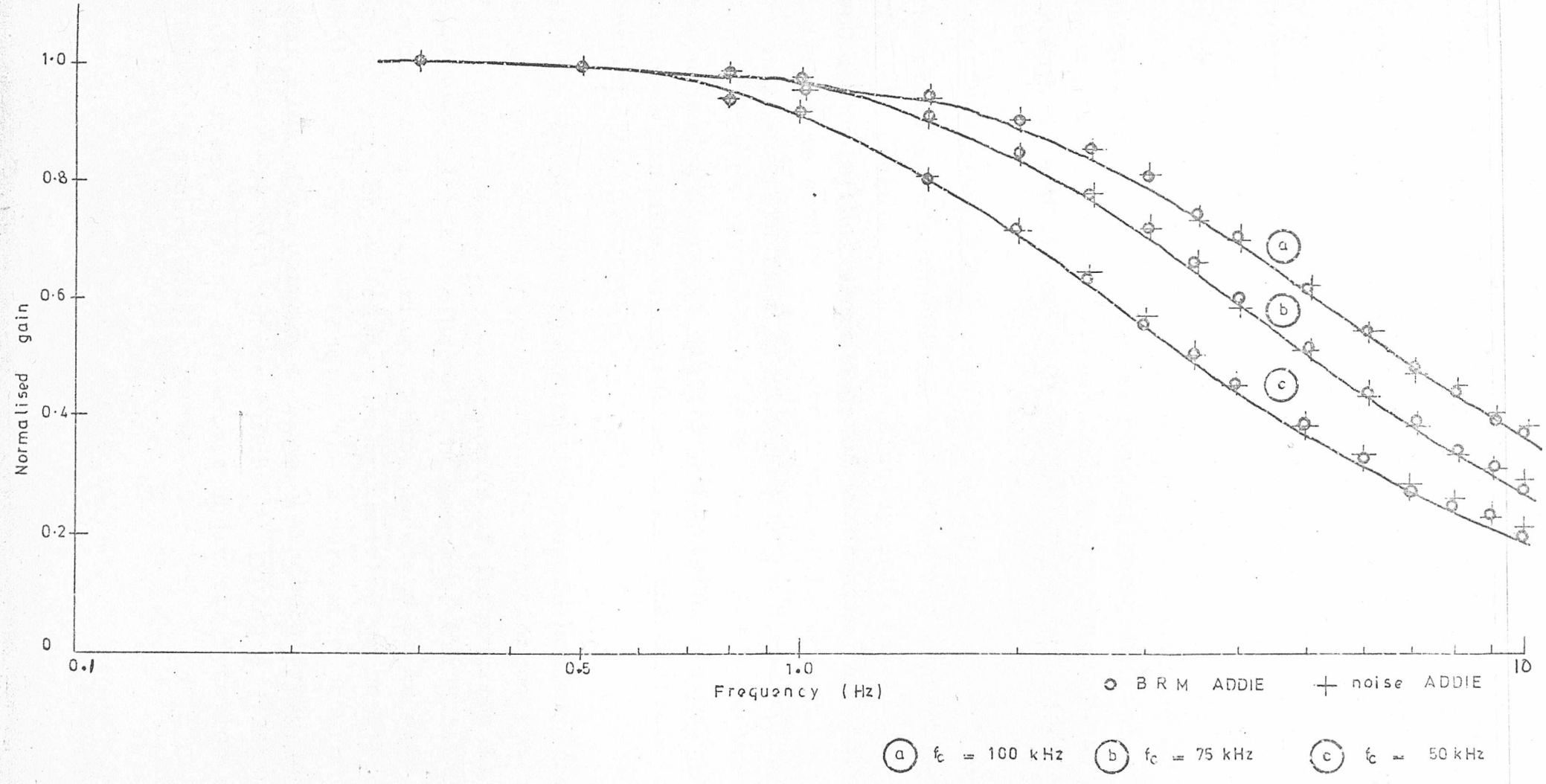


FIGURE 4·21 ADDIE frequency response characteristics

CHAPTER 5

DISCO ORGANISATION

This chapter describes the software designed to perform the various operational functions required by DISCO. Amongst the specific functions involved are the reading of a twelve bit number from the output interface and the loading of a twelve bit number into the input interface. Also described is the software system for the scaling of the integrators and the establishment of their initial conditions. Finally, the visual display unit operation and software is explained.

5.1 Interface with PDP8/E

The stochastic computer is supervised and controlled from the console of a video display unit (VDU). The VDU controls the interchange of information and data between the PDP8/E and DISCO. The minicomputer referred to as the PDP8/E is a Digital Equipment Corporation machine, with 16K of core store and several peripherals including analogue to digital converters, digital to analogue converters, real-time clock, and digital buffered input/output.

All the information typed on the console keyboard is stored in the core store, and only transferred to DISCO upon command. The transfer of information is performed almost exclusively by means of the digital input/output facilities. This consists of 12 bits of information, which may be transmitted or received individually or 'en bloc'. The supervisory program will respond to certain command characters typed on the keyboard, but will ignore others by typing a question mark. These command characters and their function are listed in Figure 3.7.

If /

If one of these commands is typed, then the program will enter the subroutine corresponding to that command. Any information to be transmitted is stored sequentially in memory, to be clocked out on the $ command. This command initiates a series of clock routines for the scaling information, connection information, etc. There are two outputs used for each set of information, one for the data and one for the clock pulse.

The interconnection between the PDP8/E and DISCO is made by a length of flat ribbon cable, terminated at the receiving end by pull-up resistors and Schmitt trigger AND gates (Figure 5.1). Once the information has been entered into DISCO, the inputs may be disabled. This avoids any spurious pulses on the line interfering with the information, and is also a protection against inadvertently entering new data before it is required.

5.2 The Reading of a 12 bit Number

If the command given is R, the program will enter a subroutine to print an average of fifty readings of a 12 bit number. The flowchart for this routine is shown in Figure 5.2. In this case, the digital buffered input on the PDP8/E is used. The number read is the output from an ADDIE or directly from the counter of an integrator. An average of 50 was taken as a reasonable amount to ensure a good representation of the variable. An increase of this figure to 100 samples was found to have no significant effect on the accuracy of the result, while increasing the computing time to a noticeable extent. The process requires the use of the formula

$$E = V(2.p(on) - 1)$$

if the single line bipolar mapping is used.
V is the maximum value represented, p(on) is the 12 bit number read, and E is the value of the variable. Since multiplication is involved, the floating point package supplied by the Digital Equipment Corporation was used. This /

This is a set of mathematical subroutines which may be easily incorporated within the users programs. When fifty readings have been taken, the result is averaged, and then printed in floating point format, eg, 0.734112E02 which is 73.4112.

5.3 The Input to the Comparators

The command I initiates the subroutine to set up the input comparators. The flowchart is shown in Figure 5.3, and the circuit diagram of a comparator is shown in Figure 5.4. Each comparator has a twelve bit number associated with it, corresponding to the required input value. The input subroutine first asks the operator to state the maximum value (V), then the number of comparators being used, and for each comparator, the required input value. These values are converted from the floating point format, in which they are entered, to a twelve bit fraction representing the input probability. This is accomplished using the mapping [3,8]

$$p(on) = \tfrac{1}{2} + \tfrac{1}{2}\,\frac{E}{V}$$

where E is the input value. All the information is stored sequentially in memory and clocked into the comparator shift registers on the $ command. As in the automatic patching input routine, if the data is entered erroneously, then it may be deleted by the rubout key, one character at a time.

5.4 The Scaling of the Integrators

The command S causes the subroutine for scaling of the integrators to be entered. As stated previously in Chapter Four, the scaling requires a four bit code pattern to be entered into each integrator, determining the factor by which the output is to be multiplied. In a similar fashion to the comparators, the information is loaded into a four bit shift register on the integrators. The flowchart for this is shown in Figure 5.5. The subroutine /

subroutine first requests the user to enter the number of integrators in the system, so that the correct number of scaling factors will be requested. The factors themselves are then entered; one per integrator. These are then manipulated to give the correct four bit code, and stored sequentially in memory. Again, the information is only entered into DISCO upon receipt of the $ command.

5.5 The Initial Conditions[9]

In normal analogue computers there is a facility for setting up initial conditions on the integrators. This consists of an initial charge on a capacitor so that the output of an integrator will start at a certain set value. In DISCO, the integrator store is an up/down counter so that any initial condition must take the form of a binary number corresponding to the required starting value being set in that counter.

Normally, this would be accomplished by the use of the preset and clear inputs to the individual bits of the counter which would be set to the required bit pattern. This information would then be entered by a single 'load' pulse. This method is not feasible with the equipment available since it requires twelve lines to each position in DISCO, and a twelve bit output from the PDP8/E. There are not enough pin connections on the verocard slots in DISCO for 12 lines to be connected, and a parallel output of 12 bits from the PDP8/E is not possible since several of the outputs are already being used for other functions.

A possibility would have been to use a 12 bit shift register on each integrator to store the initial conditions. These would have been loaded serially from the PDP8/E in the same manner as the comparator inputs. Although this would have been an excellent solution, the integrator circuitry is so complex that there is no physical space on the verocards to allow a shift /

shift register to be used. It was therefore necessary to store the initial conditions other than in the PDP8/E. Since DISCO may consist of up to a 34 integrator system, the storage required is 34x12 bits. This was realised by the use of MOS integrated circuits. Six 40 bit shift registers with integral recirculate control are contained in one MOS integrated circuit, therefore, two of these are used. This gives the facility for up to a 40 integrator configuration if the DISCO system is expanded at some future date.

The logic for the initial conditions system is shown in schematic form in Figure 5.6 and the machine language program listing is given in Appendix 6.

The system operates by only allowing one integrator at a time to count up to its required initial condition. This is done by having a twelve bit counter in the system, which is clocked at the same time as the integrator counter, and comparing it with the required value. When the required initial condition is reached, the integrator counter is stopped using the hold line and the next integrator enabled. The MOS shift registers are clocked after each integrator has been set, to allow the next initial condition to be compared with the twelve bit counter. This twelve bit counter is also cleared after every comparison.

The initial condition program is entered when the command letter P is typed on the keyboard. The flowchart for this program is shown in Figure 5.7. The first operation is to clear all previous initial conditions from core memory by setting them to zero. The number of integrators in the system and the maximum value V are then requested. The integrators which have to be initialised are identified by the number of the slot in DISCO into which they are inserted. For each integrator in the system, the program requests a slot number and a corresponding initial condition. This information is stored in a series of forty memory locations, one for each integrator in the system. If an

integrator is inserted into a slot, but not identified in the program, it will be set to binary zero, ie -V.

When all the initial conditions are in core store, they must be entered into the MOS shift registers in DISCO. This is done by a specific sequence of pulses. The MOS shift registers may be in one of two modes, write or recirculate, the particular mode being determined by the logic level of an input to the integrated circuit. Initially, this input is set to write and then an initialise command sent. This has the effect of clearing all information from the system, and making it available to accept fresh data. The next step is to enter the initial conditions, in groups of twelve bits, into a serial in, parallel out shift register in the system. The outputs of this shift register are connected to the inputs of the twelve MOS shift registers. A clock pulse is generated automatically at the end of each group of twelve bits, to enter the information into the MOS storage. When all the forty initial conditions have been entered, the write/recirculate input is set to recirculate so that the information will not be lost.

Having entered all the initial conditions, all that remains is to enter a specific time interval after which the DISCO system will be reset. The program requests a 'compute time' in seconds, say n, and at every n seconds, an initialise pulse will be sent to the logic, resetting the integrators to their initial conditions. Any time interval from one to forty seconds may be used. The timing is done by the use of the real-time-clock facility in the PDP8/E. This is a crystal controlled clock which may be set to any specific frequency, from 100 Hz to 1MHz, a twelve bit counter, and a buffer-preset register. The twelve bit counter is allowed to count at the specified frequency, until it overflows, when the contents of the buffer-preset register are loaded into it, and the cycle starts /

starts again. This overflow of the counter may be detected and used to reset the initial conditions system. To allow a specific time interval to pass before the system is reset, the buffer-preset register must be set to minus the number of clock pulses required before the counter overflows. For a time of n seconds, therefore, at a clock frequency of 100Hz the buffer-preset register must be set to -(100 x n).

The overflow is detected by the use of the interrupt system in the PDP8/E, which causes a jump to a specific memory location when an interrupt occurs, eg when the clock counter overflows. When this happens, the real-time clock is examined to determine whether or not the counter had overflowed. If it had, the initial conditions system is reset and the real-time clock restarted. If the interrupt was generated by something other than the real-time clock, eg, a command being typed on the keyboard, then the interrupt is turned off, and the normal program entered.

5.6 The Visual Display Unit

The VDU used with the PDP8/E is shown in Figure 3.5, and consists of a cathode ray tube and keyboard. The characters displayed are generated inside the unit by a read only memory, in the form of a 5x7 dot matrix. This particular VDU does not have its own memory, but uses the core store of the PDP8/E. It may display up to forty rows of sixty-four characters, each character being allocated to one word in memory. An electronic tone may also be generated, equivalent to the normal teletype bell.

The PDP8/E has 16K words of memory, in four fields of 4K words. Field zero is used for the main patching programs and field one is used for the visual display unit program and storage. The flowcharts for this program are shown in Figure 5.8. The characters to be displayed are stored in seven bit ASCII format, the remaining bits of /

of the twelve bit words being used to determine the format of the character. It may be displayed normally, or brighter than normal, or pulsed on and off, or the true and complement forms displayed alternately. All characters are usually displayed in the normal brightness mode, except the last character which is the space character displayed in the pulsed mode, for use as a cursor.

The display uses data break, which is a method whereby the PDP8/E is interrupted by the display, which directly accesses memory for the characters required. This means that the display is automatically refreshed, although the operating speed of the computer is reduced slightly. Memory is accessed sequentially, beginning at a location initially specified by the program. The last of the characters is determined by a special end of screen character which instructs the display to return to the initial location.

When the main patching program is first started, a routine called INIT is entered, which clears the display apart from the flashing cursor. This routine is also entered when the sense switch on the keyboard is depressed. Several special characters have to be sensed before they are displayed since the format they produce is unintelligible. They are the rubout character, the bell character, the carriage return and the special key on the keyboard which signifies that the display is to be moved up one line.

The rubout has to be detected so that the end of screen character and cursor may be moved back one location, which in effect removes the last typed character. The bell character is detected before display, since its seven bit code is the same as that for the letter G. When it is detected, a half second electronic tone is generated within the display. The carriage return character is similar, but does not require any special manipulation, since it is implicit in the line feed character, and so is ignored. Usually, the combination carriage return, line feed is sent, and so only the line feed has an effect, moving /

moving the cursor to the start of the next line. If the special character ↑ is detected, this signifies that the user requires the display to be moved up by one line. This is done by searching the characters in memory sequentially, until a line feed is encountered. When this is found after say x characters have been examined, the remaining characters in the display are moved to the xth previous location in memory. This routine is also entered if the allowable memory storage is full of characters. However, the user is notified of this occurrence by the previously mentioned electronic tone.

The routines that have been described here are the main operating programs of the DISCO software. Each has its own special function, independent of the others, and thus fault finding is made comparatively easy. Each routine was written individually as DISCO expanded and added to the previous section. At the present moment, the system tape consists of all these routines, except the VDU software, plus the floating point package supplied by DEC, and the distribution curve display and graphing routine. It is of course possible to expand these routines to incorporate any additional software, such as to accept analogue inputs and give graphical outputs on the VDU, etc.

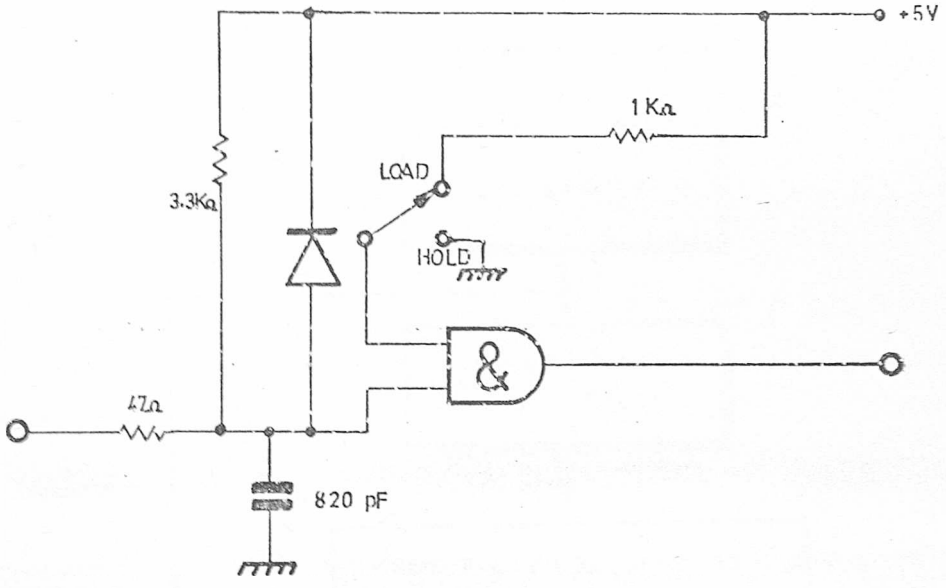


Figure 5.1 Buffer system

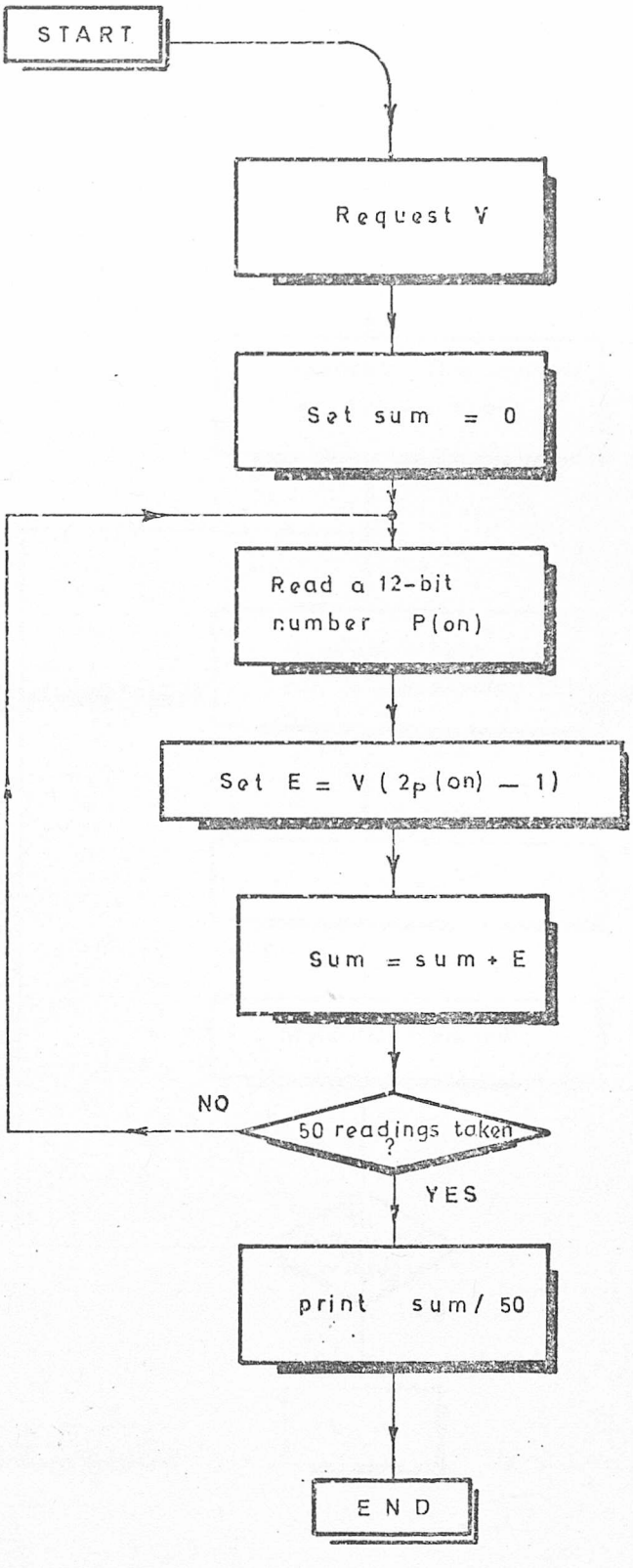


FIGURE 5.2 Reading of a 12-bit number

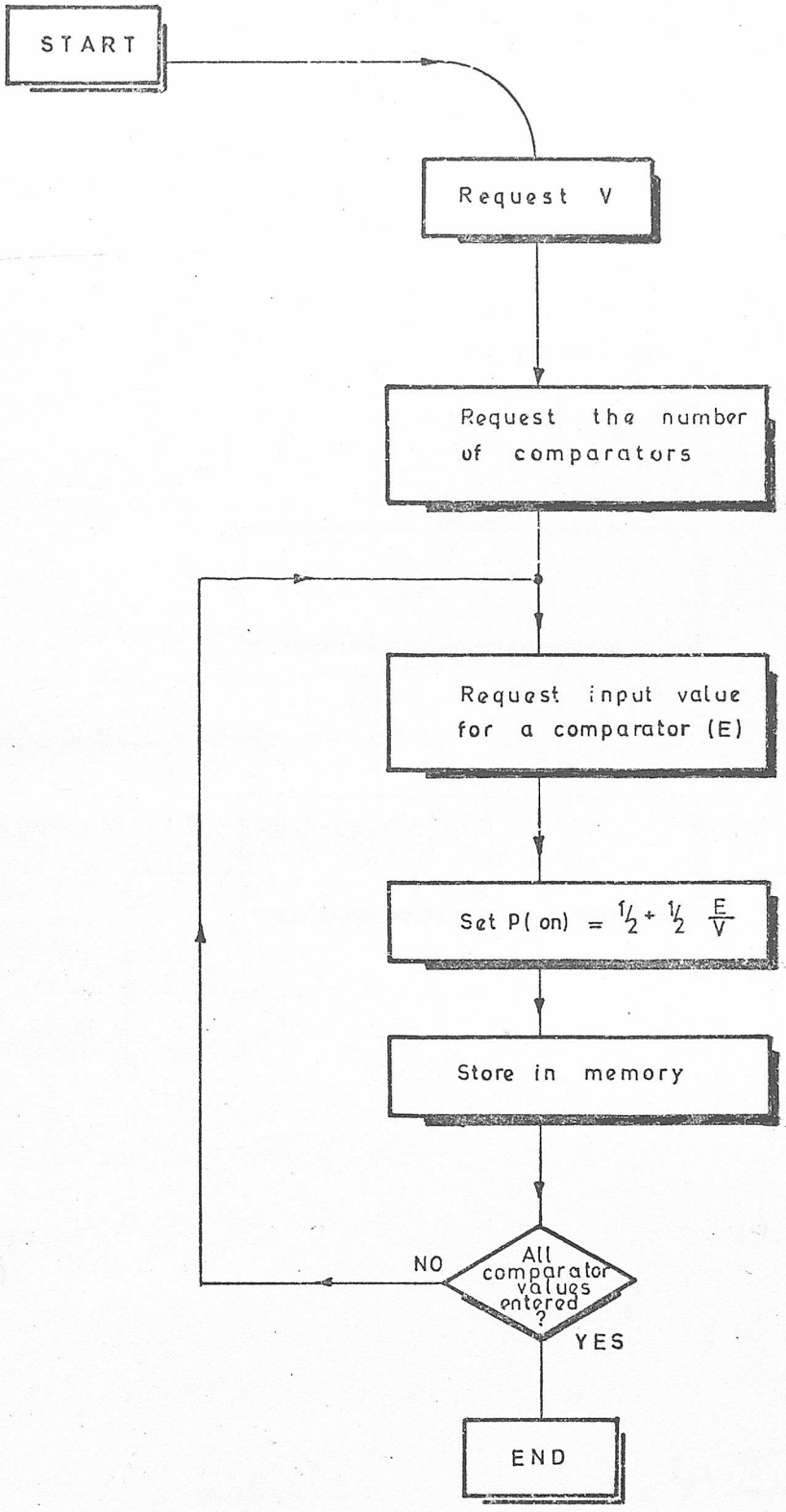


FIGURE 5·3 Setting up of the inputs

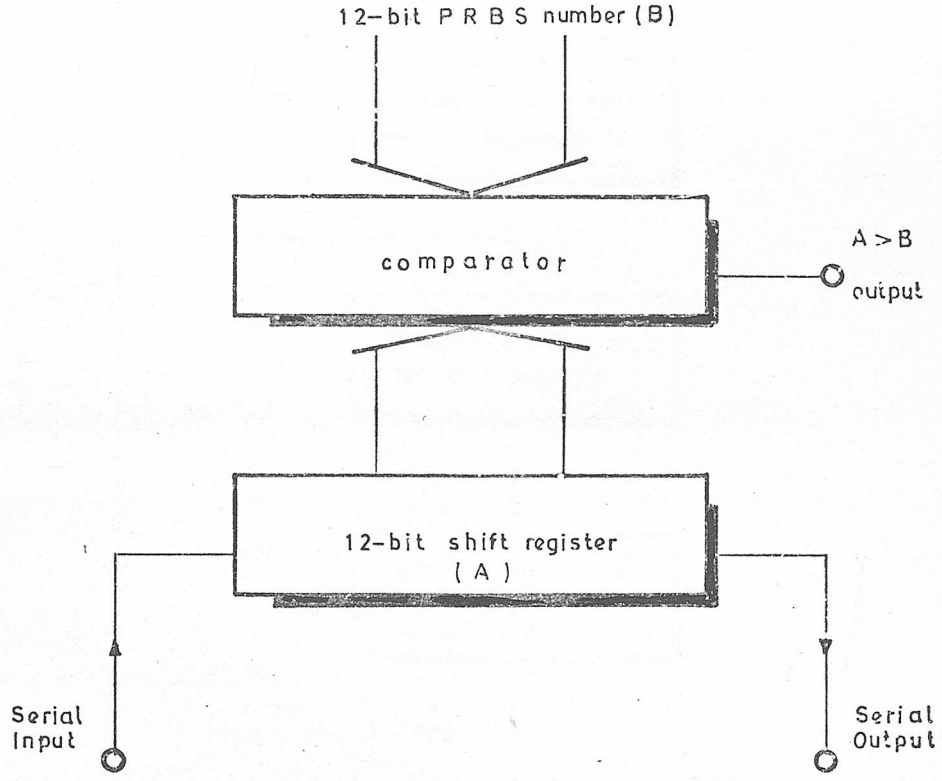


FIGURE 5.4 Input comparator

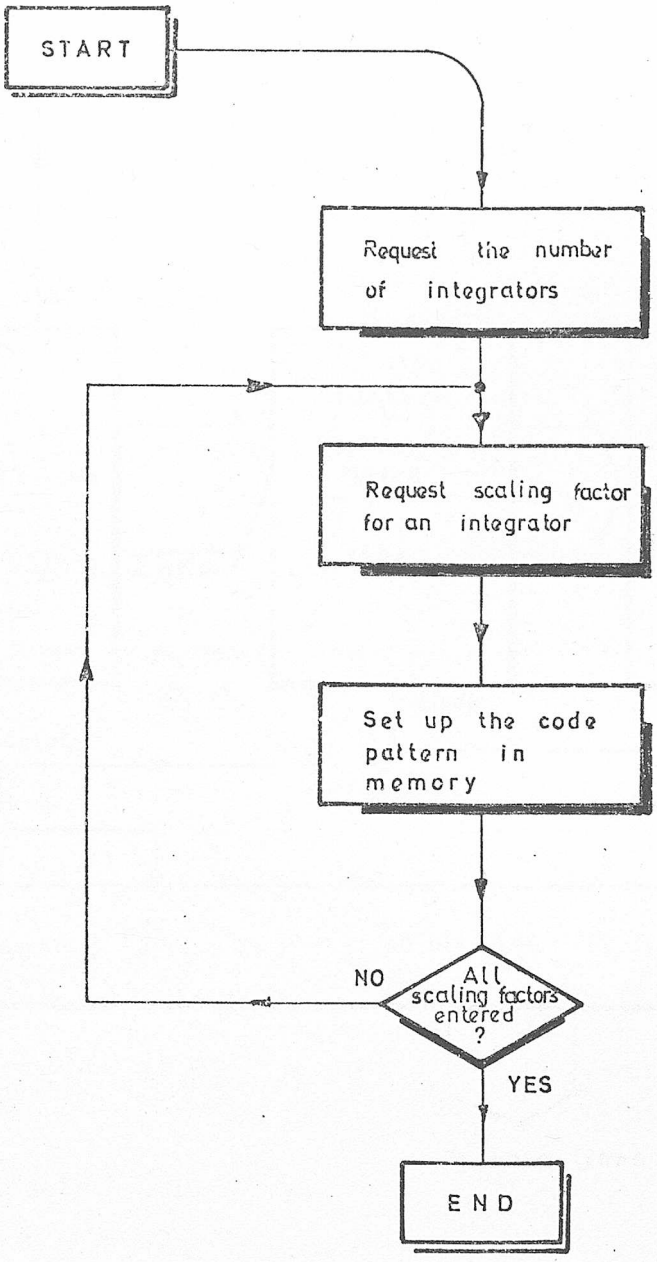


FIGURE 5.5 Scaling of integrators

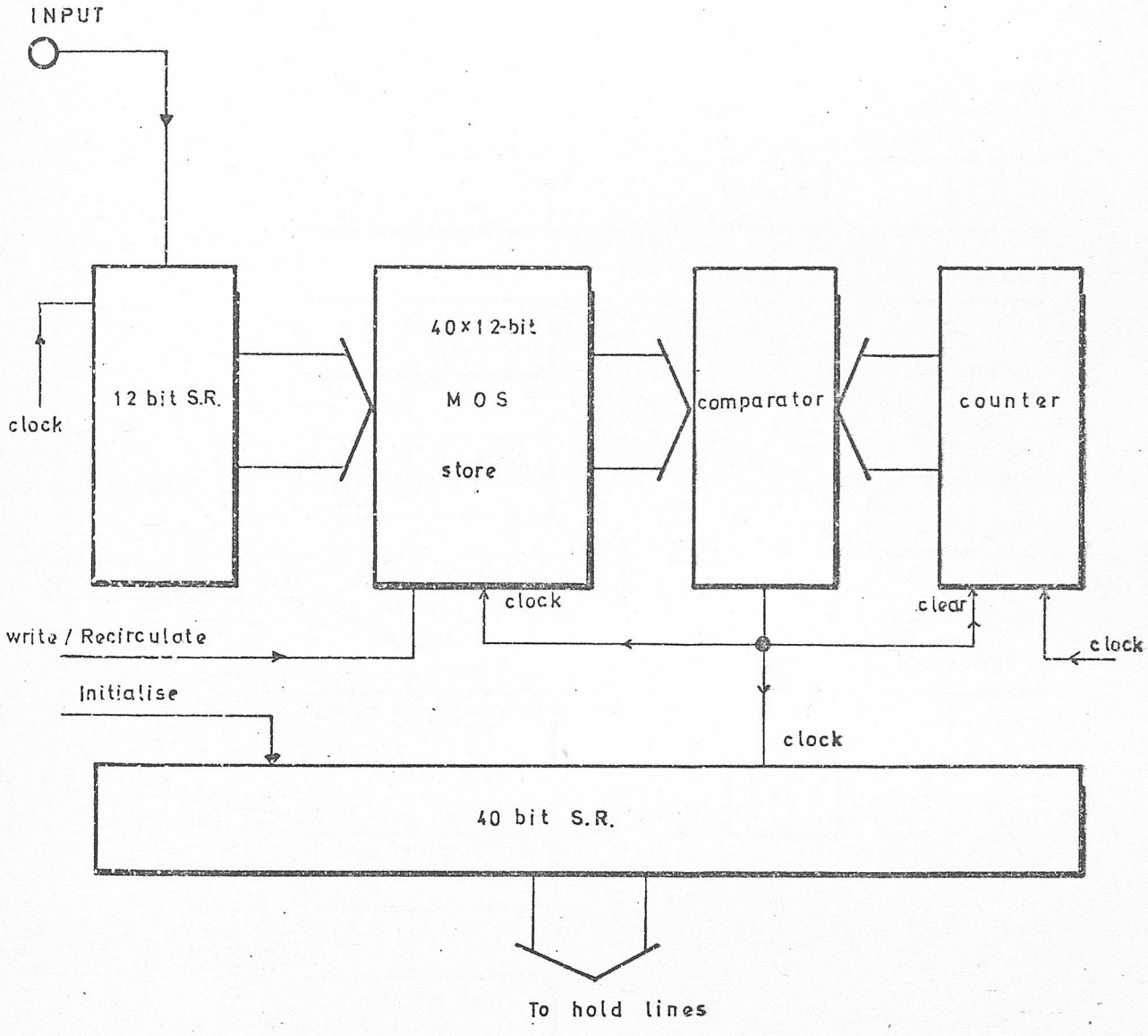


FIGURE 5.6 Initial conditions Logic

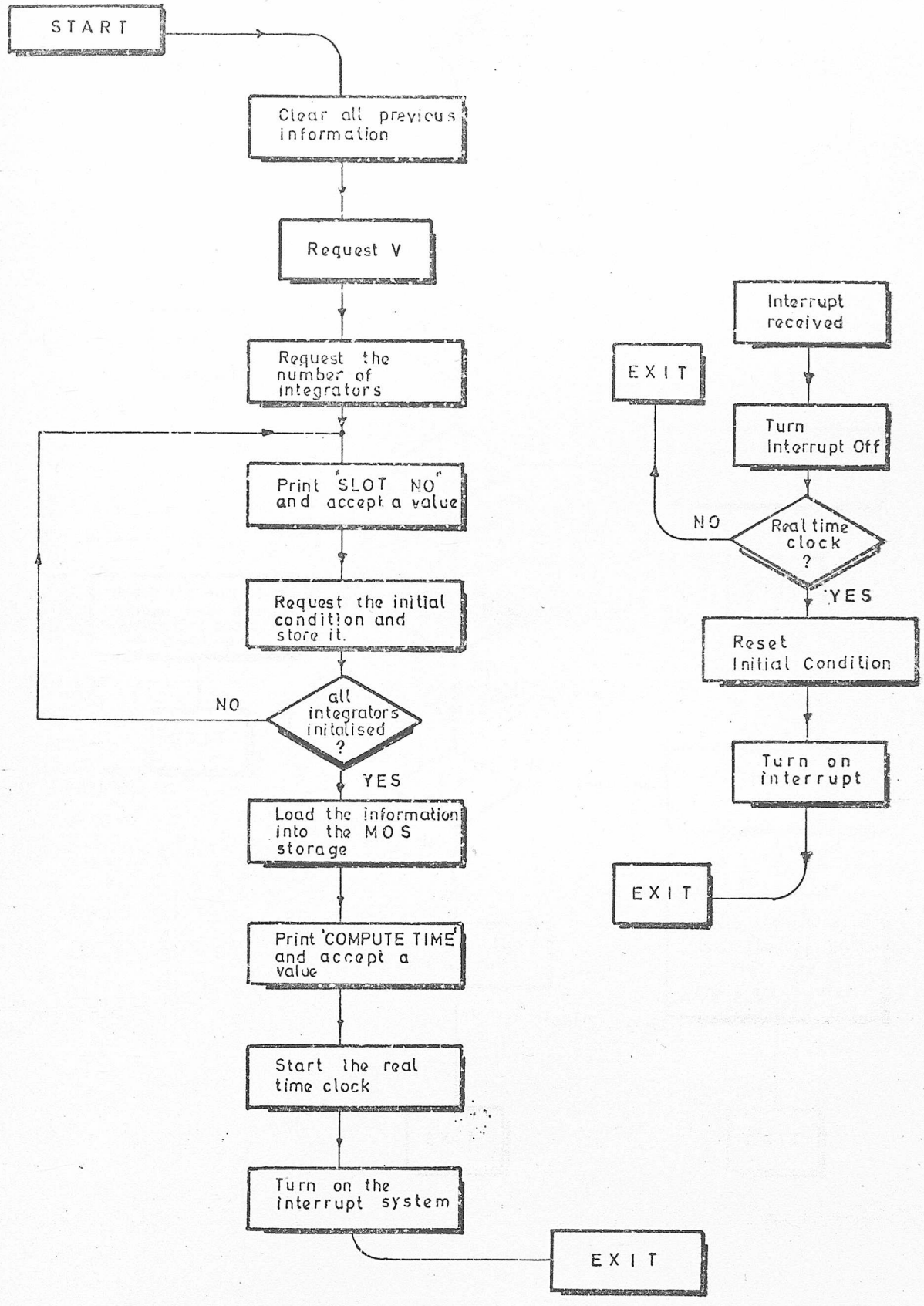


Figure 5.7 Setting up of the Initial conditions

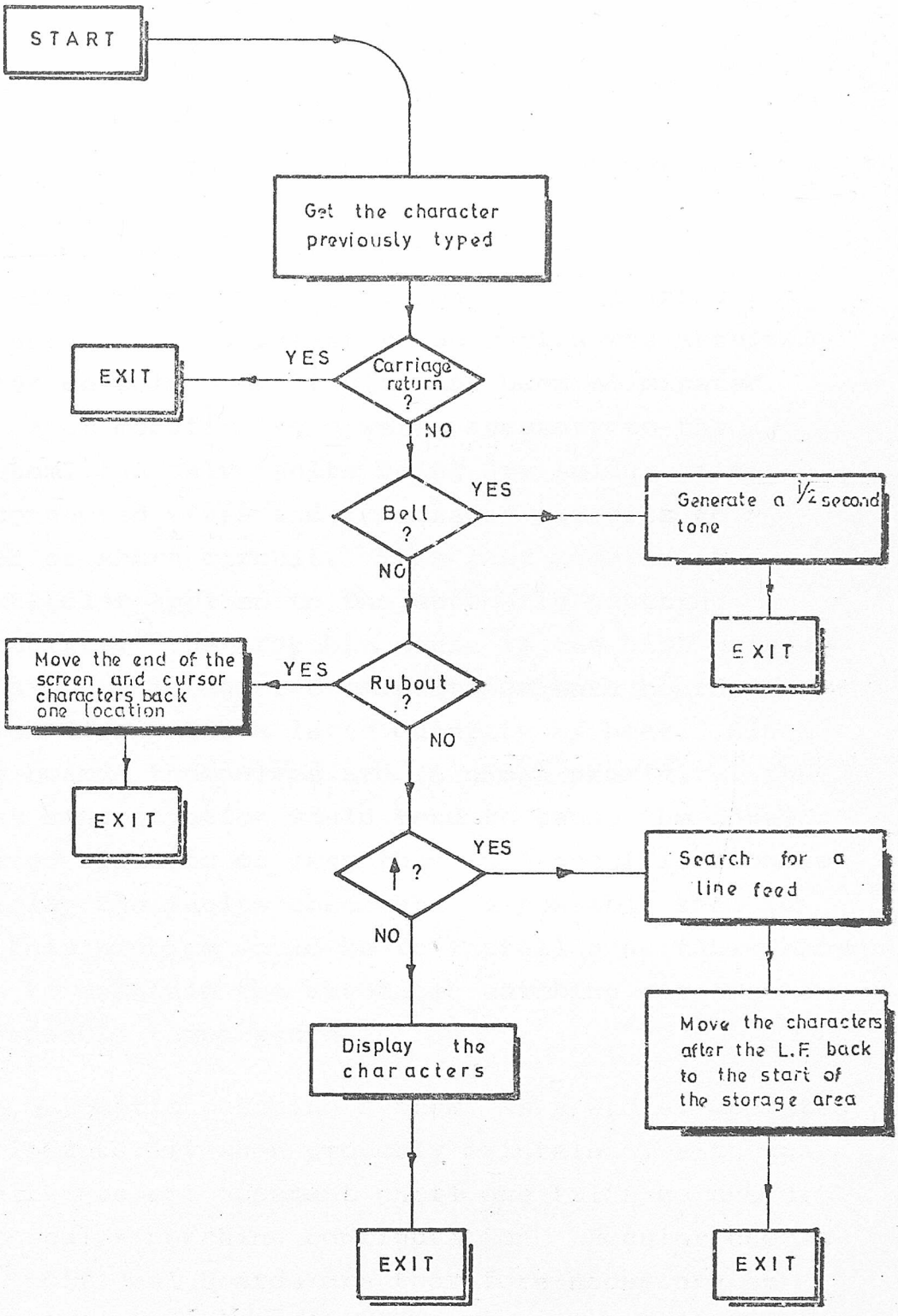


FIGURE 5.8 V. D. U. software

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

6.1 Overall Machine Design

We have seen that the stochastic computer was designed with certain criteria in mind, ie, to be used as a general purpose machine for investigations into non-permanent stochastic structures. Satisfactory operation was attained after certain small faults had been eliminated. No major modifications were necessary to the system, the only faults being dry solder joints, unconnected wires and integrated circuits going open or short circuit. This last problem in particular applied to the automatic patching circuitry.[9] The probable cause is the high packing density of integrated circuits on each board, which dissipates a large quantity of heat. Since the boards themselves are in close proximity, the heat concentration would tend to cause the integrated circuits of less than full specification to display the faults observed. A possible solution to this problem would be to install a suitable blower fan to maintain the automatic patching cards at a reasonable temperature.

The automatic patching system, as a whole, operates satisfactorily when properly maintained, although faults are not apparent until one tries to use a particular patching configuration. Regular checks on individual boards are therefore necessary at monthly intervals using the automatic testing program written specifically for this purpose.

Having /

Having used the computer to study analogue type problems, certain limitations were observed. The most important of these is the scaling of the integrators. This function is performed to multiply the input to the integrators by a factor of 2, 4, 8 or 16, to compensate for the effect of the summers and multipliers. It is achieved by reducing the counter size from 12 bits to 11, 10, 9 or 8. Observations of this procedure showed that it resulted in increased drift for each scaling step, and hence a similar reduction in accuracy. At maximum scaling, the number of states in the integrator is 256. For each clock pulse, the change is 0.39% of the total counter size, and thus results in a coarse approximation to the variable represented. However, if the counter size was increased to 16 bits, and at maximum scaling reduced to 12 bits, the resulting change for one clock pulse would be 0.024%. This would give increased accuracy and less drift, but would reduce the bandwidth.

Another limiting factor in the system is the output interface. At the moment this consists of a stochastic to analogue convertor[9], which is in effect a second order filter, with a damping factor slightly less than one. Unfortunately, the accuracy of this interface depends upon a high clock speed, which means that it is not possible to slow the complete system down to observe certain results.

This interface is also necessarily limited in bandwidth due to its design. It is recommended therefore, that the output interface used be an ADDIE, preferably the BRM ADDIE, in conjunction with a digital to analogue convertor. This will give an increased response time, and also enable the output to be observed at low clock rates.

One /

One further problem arose concerning the main PRBS generator which would start in the all zeros state when power was initially applied. This resulted in no noise sources being available for comparators, integrators, etc. This problem was solved by having a push button on the front panel, which when depressed, loaded several 'ones' into the main PRBS generator shift register.

6.2 The Universal Stochastic Module (USM)

As previously stated, the computer DISCO operates satisfactorily although some improvements have been pointed out. One final improvement would be to have a universal module which contains all the stochastic elements, selectable by a code. This module would contain all the necessary circuitry for use as an input or output interface, integrator, summer or multiplier. It would also have a squarer and invertor, and a PRBS generator which would be programmable to start in one of several states. This would ensure that no correlation of sequences occurred. Selection of the function required would be achieved by means of a three bit code, either entered as three separate digits, or as a serial data string to an on-chip shift register.

This latter method would facilitate cascading of several USMs to form complex systems. A suggested circuit diagram of such a USM is shown in Figure 6.1. In this circuit, the scaling factor for the up/down counter, the ROM address, the function select code, and the input information are all entered into a large serial shift register. It is envisaged that this data will be loaded under control of a mini-computer or microprocessor which will act as supervisor for the complete system. If the length of the counter n is limited to 12 bits, the complete USM could be contained on a 24 pin package. However, if

n /

However, if n is required to be 16 bits, then the next size of package, 40 pin, will be required. In this case, it may be desirable to omit the function select logic, and simply have all functions and their inverses available at separate output pins. This would have the advantage that one USM would provide multiple functions of the two inputs if this was so desired.

The construction of such a module would be economically sound, reducing considerably the overall cost of the stochastic computer and simplifying construction.

6.3 Applications[9]

The stochastic computer and particularly the USM should prove to be a significant step forward in the fields of on-line process control and general instrumentation problems. The limiting factor in process control is not the lack of suitable algorithmic techniques, but the speed of computation. The stochastic computer is faster than the conventional digital computer since the computing operations are performed in parallel and it is also more accurate than the analogue computer. It is envisaged that the main use of the USM will be in hard wired packages performing particular algorithms, which will be plugged into the bus on conventional digital computer systems. The algorithms in which it will be most useful are expected to be linear and dynamic programming and matrix inversion and multiplication.

In many instances, the input to the control system will already be in stochastic form. This gives the stochastic computer an inherent advantage over conventional systems, since no interface problems arise. Because the stochastic computer modules are based /

based on conventional analogue structures, a system of USMs can directly replace these functions. This would result in D/A and A/D convertors being unnecessary, and an increase in computation speed.

6.4 Second Order Simulation [3,9]

As an example of stochastic computation, the well known second order configuration was set up using the facilities of the complete DISCO system. The block diagram of the configuration is shown in Figure 6.2. The interconnections were made using the automatic patching system via the VDU and the output connected to an ultra violet graph plotter through a stochastic to analogue convertor. The results shown in Figure 6.3 were obtained by varying the scaling factors M and N. The initial condition of the output was set to -50, where the maximum was -100, and then the system was activated. The damping factor of this configuration is given by

$$z = \tfrac{1}{2} \cdot \left(\frac{N}{M}\right)^{\frac{1}{2}}$$

where M and N are the scaling factors. In the results shown, the damping factors are 0.25, 0.35, 0.5 and 0.7. This is of course a trivial problem, and is only used to demonstrate the typical method of use of DISCO. Several of the available functions were used, namely the automatic patching, the scaling function, the comparator input, and the initial conditions. The 'read' function was also used when setting up the system although it was converted from reading a 12 bit number, to that of reading an analogue voltage, using the A/D convertor in the PDP8/E.

Another simple system was set up to demonstrate the operation of the system. This was the sine wave generator, the block diagram of which is shown in Figure 2.5. The output of this system was again

monitored /

monitored on the ultra violet plotter and is shown in Figure 6.4. It can be seen that the amplitude of the output varies slightly due to the inherent drift of the integrators. The frequency of oscillation is determined by the scaling factors of the integrators - both of which must be the same, and the clock frequency.

6.5 Specialised Stochastic Systems

Although the stochastic computer is a valuable tool for use in conventional analogue type structures, it is ideally suited for stochastic system simulation. To this end, two specialised systems have been constructed using stochastic computing elements and techniques.[9] These systems are called the Random Walks and Markov chain processes and are contained in a separate rack in the DISCO system. A photograph of the front panel of these systems is shown in Figure 6.5.

The Markov chain process is a stochastic system consisting mainly of repetitive matrix multiplication which is of course ideally suited for use with the DISCO computer. Used in the system are twelve comparators and the automatic patching. The Random Walk circuitry consists of three up/down counters and is connected to the PRBS system in DISCO. The counters are set at some starting points, and allowed to count up and down according to the PRBS input. Eventually the counters will reach a barrier, ie, full up or empty when they may be held, or allowed to reflect from the particular barrier they have reached. The output from the system is taken from the counters and suitably decoded to drive four seven segment displays. Eventually, it is envisaged that the outputs will be converted to analogue /

analogue voltages using BCD to analogue convertors. More detailed explanations and circuit diagrams of these systems are available in two CNAA M.Phil. theses.

6.6 Further Work on Stochastic Systems

It is hoped that stochastic automata will be the next field of research using the stochastic computer. These devices, when used in a random environment can be used as learning models.[16] They may be made to change their operating probabilities due to a change in the environment by a system of punishment and reward. The usefulness of these devices is most obvious in a situation where the characteristics of a process are not known, and there is no mathematical description of the process available. Stochastic automata may be used in such a situation, where the reaction with the environment changes the probabilistic structure of the device so that the optimum result is achieved. The devices would then be known as learning automata. One advantage over conventional optimisation methods such as steepest descent and hill climbing techniques is that the stochastic automata will not lock on to local optima but find the truly global optimum.

Very little work has been done on these learning systems due to the time taken to process any simulation programs. It is hoped that the stochastic computer will result in much faster computation times, as an actual learning system will be constructed, not simulated. It is intended that this will be one of the main fields of investigation undertaken using the completed stochastic computer DISCO.

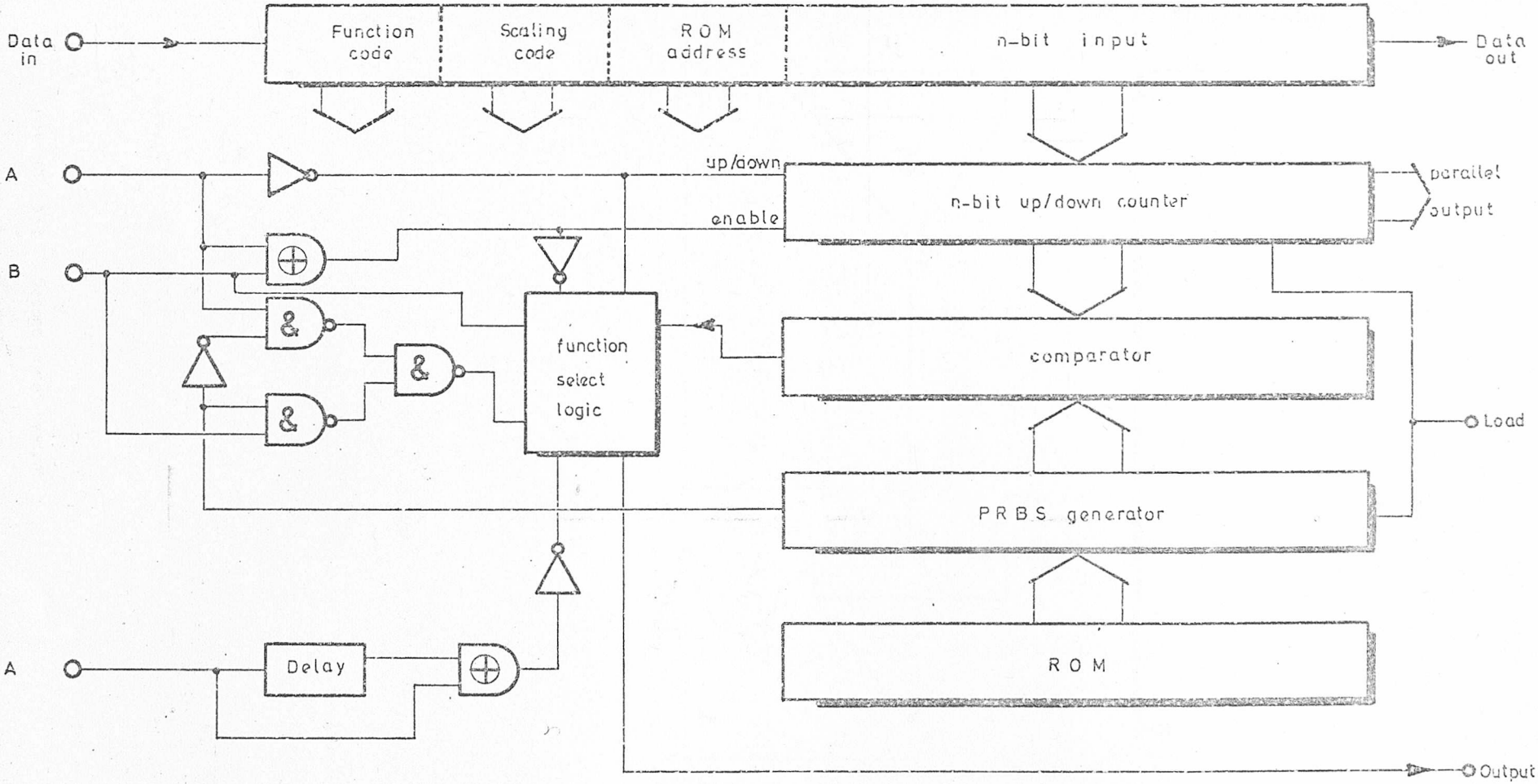


FIGURE 6·1 Universal Stochastic Module

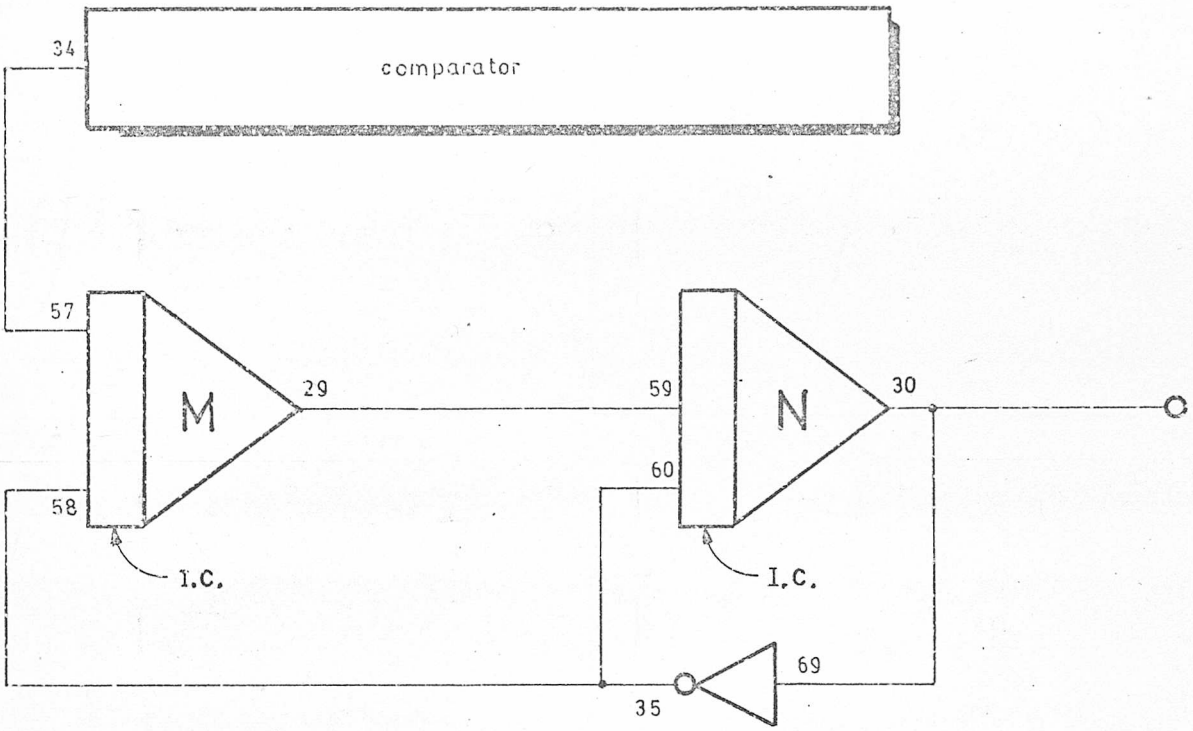


FIGURE 6·2 Second order system

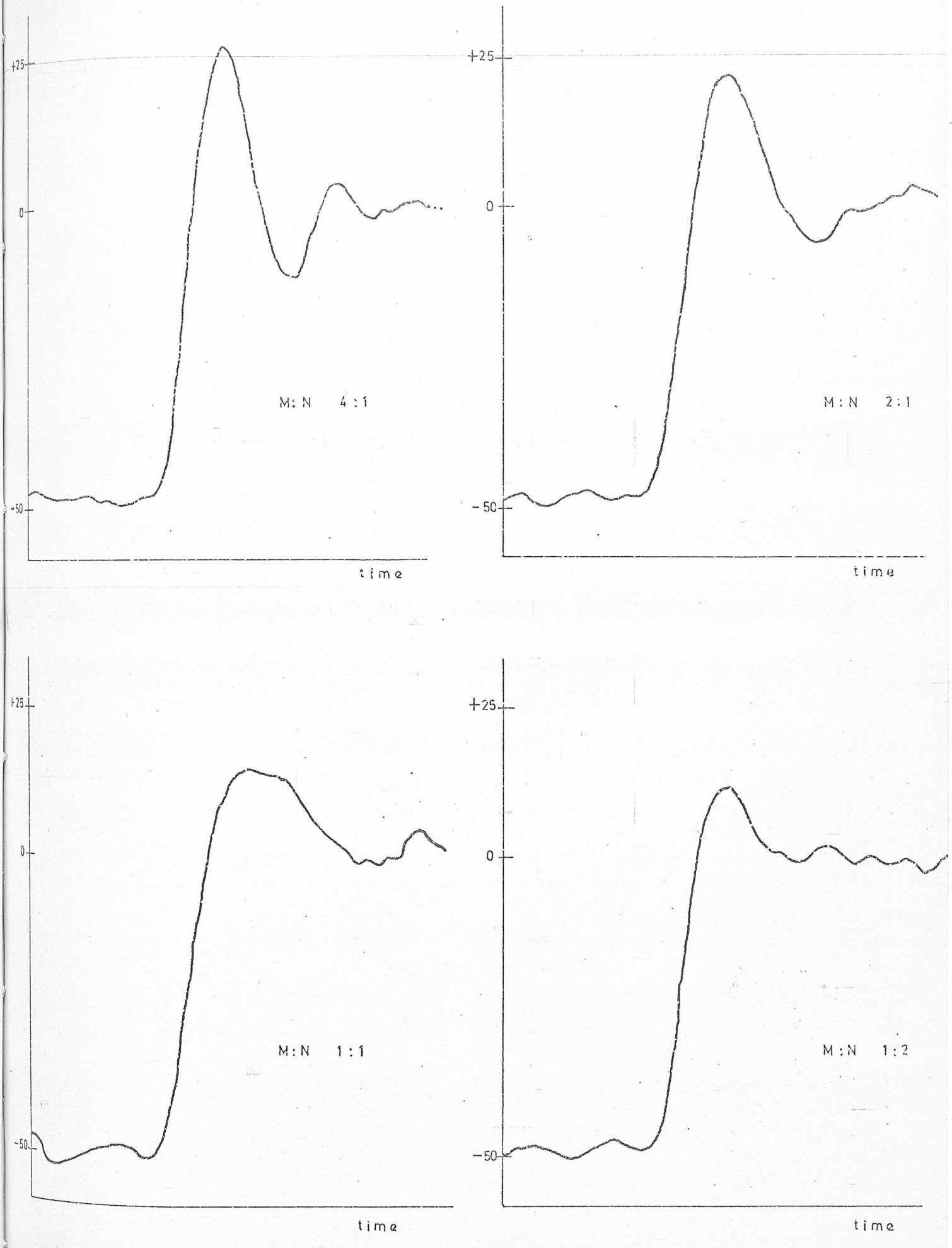
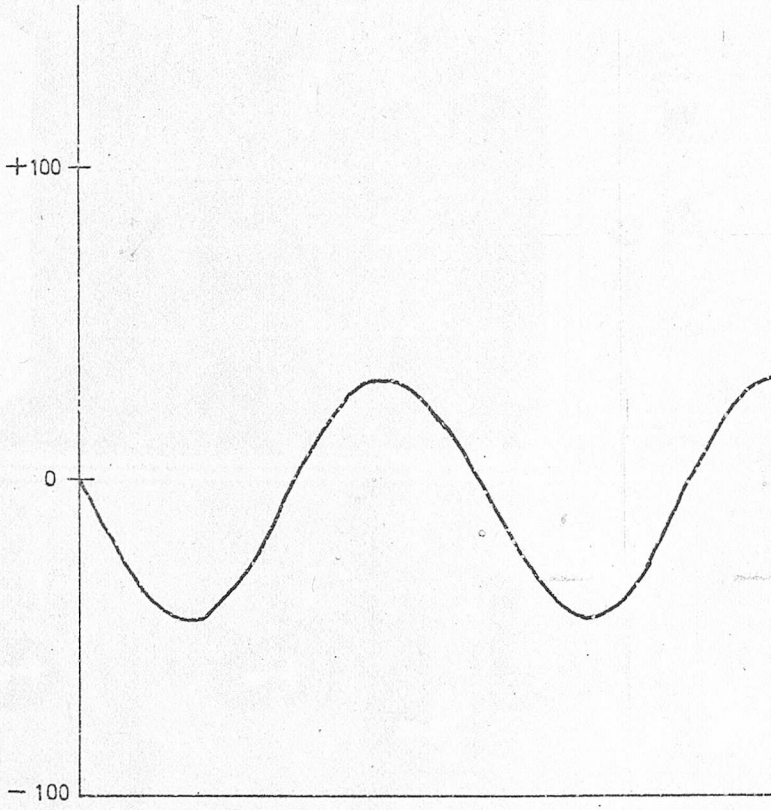


FIGURE 6·3 Second order system response


+100
0
−100

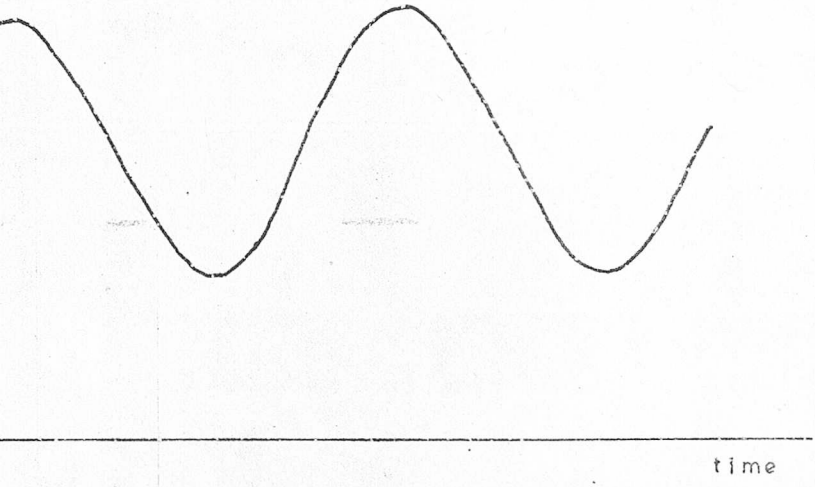


FIGURE 6·4 Stochastic sine wave

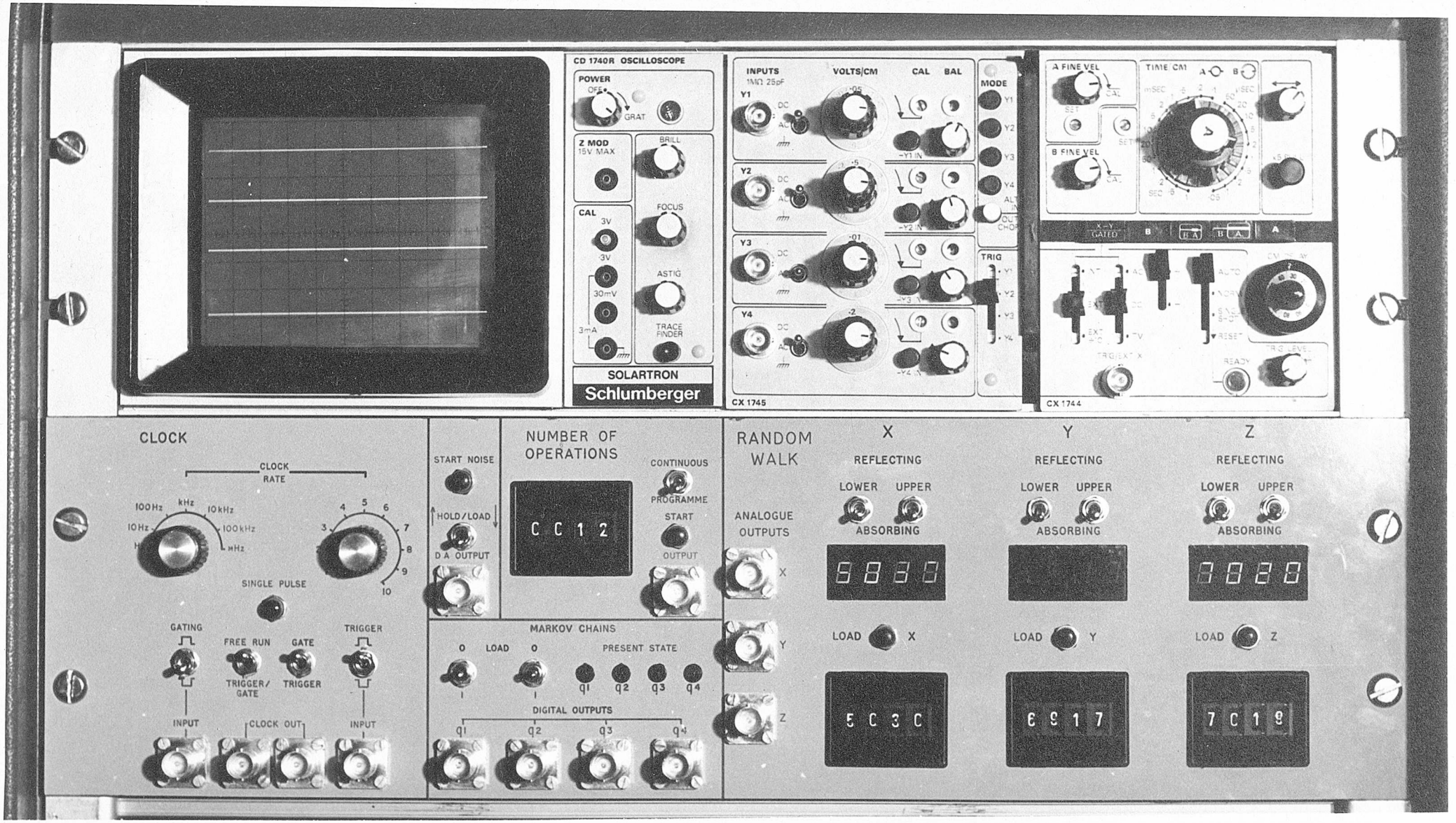


FIGURE 6·5 Specialised Stochastic Systems

REFERENCES

1. B R Gaines, Stochastic Computation,
IEEE Transactions on computers, April 1968, p.410.

2. B R Gaines, Stochastic Computer thrives on noise Electronics
July 1967, pp. 72-79.

3. B R Gaines, Stochastic Computing,
Spring Joint Computer Conference, pp. 149-156.

4. Sergio T Ribeiro, Random Pulse Machines,
IEEE Transactions on Electronic Computers,
June 1967, pp. 261-276.

5. B R Gaines, Computers, Stochastic,
Encyclopaedia of Linguistics Information and Control,
pp. 66-76.

6. W J Poppelbaum, C Afuso, and J W Esch,
Stochastic Computing Elements and Systems,
Fall Joint Computer Conference, 1967, pp. 635-644.

7. S F Reddaway and M French,
Stochastic Digital Computing,
Microelectronics, Oct 1969, pp. 32-37.

8. A J Miller, Digital Stochastic Computation,
PhD Thesis, University of Aberdeen, 1975.

9. T Baxter, Some Design Aspects and Applications
of Digital Stochastic Computers,
MPhil Thesis, CNAA Robert Gordon's Institute of Technology,
1975.

10. A J Miller, A W Brown and P Mars,
Moving Average Output Interface for Digital
Stochastic Computers,
Electronics Letters, 1974, 10, pp. 419-420.

11. A J Miller, A W Brown and P Mars,
Optimum Criteria for Output Interfaces in
Digital Stochastic Computers,
Electronics Letters, 1975, 11, pp. 326-327.

12. A J Miller, A W Brown and P Mars,
Adaptive Logic Circuits for Digital Stochastic Computers,
Electronics Letters, 1973, 9, pp. 500-502.

13. A J Miller, A W Brown, and P Mars,
Error Reduction Techniques in Digital Stochastic Computers,
Submitted for publication to IEEE Trans. on Electronic
Computers.

14. /

14. A J Miller, A W Brown, and P Mars,
A Simple Technique for the Generation of
Delayed Maximum Length Linear Binary Sequences,
Submitted for publication to IEEE Trans. on
Electronic Computers.

15. A J Miller, A W Brown, and P Mars,
A Study of an Output Interface for a Digital
Stochastic Computer,
Int. J. Electronics, 1974, Vol 37, No 5, pp. 637-655.

16. Kumpati S Narendra and M A L Thathachar,
Learning Automata - A Survey,
IEEE Trans on Systems Man and Cybernetics,
Vol SMC-4 No 4, July 1974, pp. 323-334.

APPENDIX 1

DISCO and VDU control

```
                    *20
0020  0000   LISN,   0
0021  6031          KSF
0022  5021          JMP.-1
0023  6036          KRB
0024  3114          DCA CHAR
0025  1114          TAD CHAR
0026  5420          JMP I LISN
0027  0000   TYPE,   0
0030  6041          TSF
0031  5030          JMP.-1
0032  6046          TLS
0033  7200          CLA
0034  5427          JMP I TYPE
0035  0000   QUEST,  0
0036  1125          TAD KQUEST
0037  4027          JMS TYPE
0040  5435          JMP I QUEST
0041  0000   CRLF,   0
0042  1124          TAD K215
0043  4027          JMS TYPE
0044  1123          TAD K212
0045  4027          JMS TYPE
0046  5441          JMP I CRLF
0047  0000   MESG,   0
0050  1546          TAD I POINTR
0051  4027          JMS TYPE
0052  2136          ISZ POINTR
0053  2115          ISZ CHECK
0054  5053          JMP.-4
0055  5447          JMP I MESG
0056  0077   K77,    77
0057  7744   K7704,  7704
0060  0334   KBACK,  334
                    *100
0100  0001   SW1,    1
0101  0000   STORE,  0
0102  0004          0
0103  0001   CHK,    0
0104  7720   K60,    -60
0105  0600   START,  600
0106  0000   POINTR, 0
0107  7475   NG,     -343
0110  0000   CHAR,   0
0111  7455   MS,     -323
0112  7534   MDOL,   -244
0113  7770   M10,    -10
0114  7767   M11,    -11
0115  0000   CHECK,  0
0116  0126   MSG1,   MES1
0117  0000   LOCAD,  0
0120  7777   M1,     -1
0121  0000   LOCHG,  0
0122  0145   MSG2,   MES2
0123  0212   K212,   212
```

```
0124   0215   K215,    215
0125   0277   KOUEST,  277
0126   0000   ST1,     0
0127   7540   MSPACE,  -240
0130   7444   MRUB,    -334
0131   7441   MDEL,    -337
0132   7775   M3,      -3
0133   7520   M260,    -260
0134   7776   M2,      -2
0135   0240   MES1,    240
0136   0240            240
0137   0311            311
0140   0316            316
0141   0320            320
0142   0325            325
0143   0324            324
0144   0240            240
0145   0240   MES2,    240
0146   0240            240
0147   0317            317
0150   0325            325
0151   0324            324
0152   0320            320
0153   0325            325
0154   0324            324
0155   0240            240
                *200
0200   7300   BEGIN,   CLA CLL
0201   3100            DCA SW1
0202   3101            DCA STORE
0203   3102            DCA STORE+1
0204   3103            DCA CHK
0205   6046            TLS
0206   6032            KCC
0207   1104            TAD M60
0210   3103            DCA CHK
0211   1105            TAD START
0212   3136            DCA POINTR
0213   3536            DCA I POINTR
0214   2136            ISZ POINTR
0215   2103            ISZ CHK
0216   5213            JMP .-3
0217   4441   WAIT,    JMS CRLF
0220   4020            JMS LISN
0221   1107            TAD UC
0222   7650            SNA CLA
0223   5236            JMP INPUT
0224   1110            TAD CHAR
0225   1111            TAD MS
0226   7650            SNA CLA
0227   5771            JMP I SCALEX
0230   1110            TAD CHAR
0231   1112            TAD MDEL
0232   7650            SNA CLA
0233   5772            JMP I CLOCKX
0234   4435            JMS OUEST
0235   5217            JMP WAIT
0236   1110   INPUT,   TAD CHAR
0237   4027            JMS TYPE
```

```
0240  1113          TAD M10
0241  3115          DCA CHECK
0242  1116          TAD MSE1
0243  3106          DCA POINTR
0244  4047          JMS MESG
0245  4300          JMS NEXT
0246  4773          JMS I NORM
0247  1117          TAD LOCAD
0250  1120          TAD M1
0251  7110          CLL RAR
0252  1105          TAD START
0253  3121          DCA LOCNO
0254  7004          RAL
0255  3100          DCA SW1
0256  1114          TAD M11
0257  3115          DCA CHECK
0260  1122          TAD MSE2
0261  3106          DCA POINTR
0262  4047          JMS MESG
0263  4300          JMS NEXT
0264  4773          JMS I NORM
0265  1117          TAD LOCAD
0266  1120          TAD M1
0267  3117          DCA LOCAD
0270  1100          TAD SW1
0271  5672          JMP I .+1
0272  0474          WAITX+1
0273  3101          DCA STORE
0274  3102          DCA STORE+1
0275  1103          TAD CHK
0276  5374          JMP NORM+1
0277  5217          JMP WAIT
0300  0000  NEXT,    0
0301  7300          CLA CLL
0302  1126          TAD ST1
0303  3106          DCA POINTR
0304  3103          DCA CHK
0305  4020  XNXT,    JMS LIST
0306  1127          TAD MSPACE
0307  7650          SNA CLA
0310  5700          JMP I NEXT
0311  1110          TAD CHAR
0312  1123          TAD MRUB
0313  7650          SNA CLA
0314  5337          JMP ERASE
0315  1110          TAD CHAR
0316  1131          TAD MDEL
0317  7650          SNA CLA
0320  5364          JMP DELETE
0321  2103          ISZ CHK
0322  1103          TAD CHK
0323  1132          TAD M3
0324  7640          SZA CLA
0325  5330          JMP .+3
0326  4035          JMS CHECK
0327  5217          JMP WAIT
0330  1110          TAD CHAR
0331  4027          JMS TYPE
0332  1110          TAD CHAR
0333  1133          TAD M26
```

```
0334  3506          DCA I POINTR
0335  2106          ISZ POINTR
0336  5305          JMP XNXT
0337  1103  ERASE,  TAD CHK
0340  7650          SNA CLA
0341  5305          JMP XNXT
0342  1103          TAD CHK
0343  1120          TAD M1
0344  7640          SZA CLA
0345  5355          JMP R2
0346  1126          TAD ST1
0347  3106          DCA POINTR
0350  3101          DCA STORE
0351  3103          DCA CHK
0352  1060  XR2,     TAD KBACK
0353  4027          JMS TYPE
0354  5305          JMP XNXT
0355  3102  R2,     DCA STORE+1
0356  1126          TAD ST1
0357  7001          IAC
0360  3106          DCA POINTR
0361  3103          DCA CHK
0362  2103          ISZ CHK
0363  5352          JMP XR2
0364  5273  DELETE, JMP NEXT-5
0365  3103          DCA CHK
0366  1110          TAD CHAR
0367  4027          JMS TYPE
0370  5331          JMP NEXT+1
0371  0371  SCALEX, SCALE
0372  0425  CLOCKX, CLOCK
0373  0400  NORM,   NORMX
0374  7650          SNA CLA
0375  5305          JMP XNXT
0376  5365          JMP DELETE+1
                    *400
0400  0000  NORMX,  0
0401  7300          CLA CLL
0402  1103          TAD CHK
0403  1134          TAD M2
0404  7640          SZA CLA
0405  5220          JMP ONE
0406  1126          TAD ST1
0407  3106          DCA POINTR
0410  1506          TAD I POINTR
0411  7106          CLL RTL
0412  1506          TAD I POINTR
0413  7104          CLL RAL
0414  2106          ISZ POINTR
0415  1506          TAD I POINTR
0416  3117          DCA LOCAD
0417  5224          JMP.+5
0420  1126  ONE,    TAD ST1
0421  3106          DCA POINTR
0422  1506          TAD I POINTR
0423  3117          DCA LOCAD
0424  5600          JMP I NORMX
0425  1110  CLOCK,  TAD CHAR
0426  4027          JMS TYPE
```

```
0427  4041          JMS CRLF
0430  7240          STA
0431  6546          DBS0
0432  7300          CLA CLL
0433  1144          TAD M60
0434  3115          DCA CHECK
0435  1145          TAD START
0436  3117          DCA LOCAD
0437  3101  NLOC,    DCA STORE
0440  1266          TAD M14
0441  3103          DCA CHK
0442  1517          TAD I LOCAD
0443  1101  NBIT,    TAD STORE
0444  7113          CLL RAR
0445  3101          DCA STORE
0446  7004          RAL
0447  6545          DBC0
0450  7200          CLA
0451  1265          TAD K4000
0452  6535          DBC0
0453  6546          DBS0
0454  7240          STA
0455  6546          DBS0
0456  7300          CLA CLL
0457  2103          ISZ CHK
0460  5243          JMP NBIT
0461  2117          ISZ LOCAD
0462  2115          ISZ CHECK
0463  5237          JMP NLOC
0464  5667          JMP I WAITX
0465  4000  K4000,   4000
0466  7764  M14,     -14
0467  0217  WAITX,   WAIT
0470  7650          SNA CLA
0471  5302          JMP .+11
0472  1117          TAD LOCAD
0473  7002          BSW
0474  3117          DCA LOCAD
0475  1521          TAD I LOC10
0476  0356          AND K77
0477  1117          TAD LOCAD
0500  3521          DCA I LOC10
0501  5667          JMP I WAITX
0502  1521          TAD I LOC10
0503  0357          AND K7700
0504  1117          TAD LOCAD
0505  3521          DCA I LOC10
0506  5667          JMP I WAITX
```

```
                FIELD 1
                *200
0200  0000  INIT,   0
0201  1352          TAD BUFM1
0202  3010          DCA 10
0203  1352          TAD BUFM1
0204  3011          DCA 11
0205  1351          TAD BUF
0206  3012          DCA 12
0207  1353          TAD CURSBL
0210  3411          DCA I 11
0211  1354          TAD EOS
0212  3412          DCA I 12
0213  1351          TAD BUF
0214  6050          DPLA
0215  1357          TAD EMF
0216  6051          DPGO
0217  0217          CDI+00
0220  5600          JMP I INIT
0221  0000  ENTER,   0
0222  7421          MQL
0223  7501          MQA
0224  1365          TAD MUP
0225  7650          SNA CLA
0226  5311          JMP MOVE
0227  7501          MQA
0230  1347          TAD MBELL
0231  7650          SNA CLA
0232  5307          JMP TONE
0233  7501          MQA
0234  1350          TAD MCR
0235  7650          SNA CLA
0236  5305          JMP EXIT
0237  7501          MQA
0240  1355          TAD MRUB
0241  7650          SNA CLA
0242  5265          JMP RUB
0243  1354          TAD EOS
0244  3412          DCA I 12
0245  1353          TAD CURSBL
0246  3411          DCA I 11
0247  7501          MQA
0250  0360          AND K177
0251  3410          DCA I 10
0252  1363          TAD MBFRT
0253  1012          TAD 12
0254  7710          SPA CLA
0255  5305          JMP EXIT
0256  6057          6057
0257  6057          6057
0260  1364          TAD MBEND
0261  1012          TAD 12
0262  7650          SNA CLA
0263  5311          JMP MOVE
0264  5305          JMP EXIT
0265  1356  RUB,     TAD MRUF
0266  7041          IAC
0267  1010          TAD 10
```

```
0270  7650          SNA CLA
0271  5305          JMP EXIT
0272  7240          CLA CMA
0273  1010          TAD 10
0274  3010          DCA 10
0275  1010          TAD 10
0276  3011          DCA 11
0277  1353          TAD CURSBL
0300  3411          DCA I 11
0301  1011          TAD 11
0302  3012          DCA 12
0303  1354          TAD EOS
0304  3412          DCA I 12
0305  0217  EXIT,    CDI+00
0306  5621          JMP I ENTER
0307  6057  TONE,    6057
0310  5305          JMP EXIT
0311  1011  MOVE,    TAD 11
0312  1356          TAD MBUF
0313  7650          SNA CLA
0314  5305          JMP EXIT
0315  1352          TAD BUFM1
0316  3013          DCA 13
0317  1413          TAD I 13
0320  1361          TAD MLF
0321  7640          SZA CLA
0322  5317          JMP.-3
0323  1352          TAD BUFM1
0324  3014          DCA 14
0325  1413  MORE,    TAD I 13
0326  1362          TAD MEOS
0327  7450          SNA
0330  5334          JMP EX
0331  1354          TAD EOS
0332  3414          DCA I 14
0333  5325          JMP MORE
0334  1354  EX,      TAD EOS
0335  3414          DCA I 14
0336  1014          TAD 14
0337  3012          DCA 12
0340  7240          STA
0341  1014          TAD 14
0342  3011          DCA 11
0343  7240          STA
0344  1011          TAD 11
0345  3010          DCA 10
0346  5305          JMP EXIT
0347  7571  MBELL,   7571
0350  7563  MCR,     7563
0351  0400  BUF,     400
0352  0377  BUFM1,   377
0353  0640  CURSBL,  640
0354  3000  EOS,     3000
0355  7401  MBUB,    7401
0356  7400  MBUF,    7400
0357  0010  EMF,     10
0360  0177  K177,    177
```

```
0361   7766   MLF,      -12
0362   5000   MEOS,     -3000
0363   6424   MBFPT,    -1354
0364   6400   MBFND,    -1400
0365   7546   MUP,      -232

              DPLA=6050
              DPGO=6051
```

```
                *63
0063  0732  MSE5,   MES5
0064  7762  M16,    -16
0065  0300  XNEXT,  300
0066  0400  NORMX,  400
0067  7764  M14,    -14
0070  7774  M4,     -4
0071  0000  ICOD,   0
0072  0000  MNINT,  0
0073  0000  NINT,   0
0074  0000  PTR,    0
0075  0077  K77,    77
0076  7700  K7700,  7700
0077  0334  KBACK,  334
               *156
0156  1041  PLINT,  END
               *371
0371  0510  SCALE
               *510
0510  1110  SCALE,  TAD CHAR
0511  4027         JMS TYPE
0512  1156         TAD PLINT
0513  3074         DCA PTR
0514  1376         TAD MSE3
0515  3106         DCA POINTR
0516  1064         TAD M16
0517  3115         DCA CHK
0520  4437         JMS I XMSG
0521  4465         JMS I XNEXT
0522  4466         JMS I NORMX
0523  1117         TAD LOCAD
0524  7041         CIA
0525  3072         DCA MNINT
0526  7001         IAC
0527  3073         DCA NINT
0530  4436  NXT,    JMS I XCLF
0531  1377         TAD MSE4
0532  3106         DCA POINTR
0533  1067         TAD M14
0534  3115         DCA CHK
0535  4437         JMS I XMSG
0536  1073         TAD NINT
0537  4775         JMS I XDEC
0540  1063         TAD MSE5
0541  3106         DCA POINTR
0542  1113         TAD M10
0543  3115         DCA CHK
0544  4437         JMS I XMSG
0545  4465         JMS I XNEXT
0546  4466         JMS I NORMX
0547  1070         TAD M4
0550  3115         DCA CHK
0551  1117         TAD LOCAD
0552  7110         CLL RAR
0553  3117         DCA LOCAD
0554  1117  MR,     TAD LOCAD
```

```
0555  7010          RAR
0556  3117          DCA LOCAD
0557  1071          TAD ICOD
0560  7004          RAL
0561  3071          DCA ICOD
0562  2115          ISZ CHK
0563  5354          JMP MR
0564  1071          TAD ICOD
0565  3474          DCA I PTR
0566  3071          DCA ICOD
0567  2073          ISZ NINT
0570  2074          ISZ PTR
0571  2072          ISZ MNINT
0572  5330          JMP NXT
0573  5774          JMP I XWAIT
0574  0217  XWAIT,   WAIT
0575  1000  XDEC,    DECPRT
0576  0700  MSE3,    MES3
0577  0716  MSE4,    MES4
              *700
0700  0240  MES3,    240
0701  0240          240
0702  0316          316
0703  0317          317
0704  0240          240
0705  0317          317
0706  0306          306
0707  0240          240
0710  0311          311
0711  0316          316
0712  0324          324
0713  0247          247
0714  0323          323
0715  0240          240
0716  0240  MES4,    240
0717  0311          311
0720  0316          316
0721  0324          324
0722  0305          305
0723  0307          307
0724  0322          322
0725  0301          301
0726  0324          324
0727  0317          317
0730  0322          322
0731  0240          240
0732  0240  MES5,    240
0733  0240          240
0734  0323          323
0735  0303          303
0736  0301          301
0737  0314          314
0740  0305          305
0741  0240          240
              *1000
1000  0000  DECPRT,  0
1001  3235          DCA VAL
```

```
1002  3236            DCA DIG
1003  1232            TAD CA
1004  3237            DCA CB
1005  1231            TAD AA
1006  3213            DCA AR
1007  7410            SKP
1010  3235            DCA VAL
1011  7100            CLL
1012  1235            TAD VAL
1013  1233  AR,       TAD TPR
1014  7430            SZL
1015  2236            ISZ DIG
1016  7430            SZL
1017  5210            JMP AR-3
1020  7200            CLA
1021  1236            TAD DIG
1022  1240            TAD K260
1023  4027            JMS TYPE
1024  3236            DCA DIG
1025  2213            ISZ AR
1026  2237            ISZ CB
1027  5212            JMP AR-1
1030  5600            JMP I DECPRT
1031  1233  AA,       TAD TPR
1032  7776  CA,       -2
1033  7766  TPR,      -12
1034  7777            -1
1035  0000  VAL,      0
1036  0000  DIG,      0
1037  0000  CB,       0
1040  0260  K260,     260
            END=.
            CHAR=110
            TYPE=27
            POINTR=106
            CHK=115
            M10=113
            LOCAD=117
            XMSG=37
            XCLF=36
            WAIT=217
```

```
                *160
0160    1700    XING,   INTEG
                *234
0234    5777            JMP I 377
                *377
0377    1200            1200
                *776
0776    1262    FIN
                *1200
1200    1332            TAD COMIN
1201    3074            DCA PTR
1202    1110            TAD CHAR
1203    1322            TAD MI
1204    7650            SNA CLA
1205    5210            JMP INP
1206    4435            JMS I QUEST
1207    5660            JMP I WAITX
1210    1110    INP,    TAD CHAR
1211    4027            JMS TYPE
1212    1323            TAD M6
1213    3115            DCA CHECK
1214    1320            TAD MSE6
1215    3106            DCA POINTR
1216    4437            JMS I MESG
1217    4405            JMS I 5
1220    4407            JMS I 7
1221    6333            FPUT V
1222    0000            FEXT
1223    1324            TAD M12
1224    3115            DCA CHECK
1225    1321            TAD MSE7
1226    3106            DCA POINTR
1227    4437            JMS I MESG
1230    4465            JMS I XNEXT
1231    4466            JMS I NORMX
1232    1117            TAD LOCAD
1233    7041            CIA
1234    3325            DCA MCOMP
1235    7001            IAC
1236    3326            DCA NCOMP
1237    4436    DMOR,   JMS I CRLF
1240    1327            TAD KC
1241    4027            JMS TYPE
1242    1326            TAD NCOMP
1243    4661            JMS I XDEC
1244    1330            TAD K240
1245    4027            JMS TYPE
1246    4407            JMS I 7
1247    5333            FGET V
1250    0000            FEXT
1251    4563            JMS I XING
1252    3474            DCA I PTR
1253    2074            ISZ PTR
1254    2326            ISZ NCOMP
1255    2325            ISZ MCOMP
1256    5237            JMP DMOR
1257    5660            JMP I .+1
```

```
1260  0217  WAITX,  WAIT
1261  1000  XDEC,   DECPRT
1262  7240  FIN,    STA
1263  1326          TAD NCOMP
1264  7041          CIA
1265  3115          DCA CHECK
1266  1332          TAD COMIN
1267  3117          DCA LOCAD
1270  3101  NLOC,   DCA STORE
1271  1331          TAD M14
1272  3103          DCA CHK
1273  1517          TAD I LOCAD
1274  1101  NBIT,   TAD STORE
1275  7110          CLL RAR
1276  3101          DCA STORE
1277  7004          RAL
1300  7006          RTL
1301  6505          DBCO
1302  7200          CLA
1303  1317          TAD K1000
1304  6505          DBCO
1305  6506          DBSO
1306  7240          STA
1307  6506          DBSO
1310  7300          CLA CLL
1311  2103          ISZ CHK
1312  5274          JMP NBIT
1313  2117          ISZ LOCAD
1314  2115          ISZ CHECK
1315  5270          JMP NLOC
1316  5660          JMP I WAITX
1317  1000  K1000,  1000
1320  1450  MSE6,   MES6
1321  1456  MSE7,   MES7
1322  7467  MI,     -311
1323  7772  M6,     -6
1324  7766  M12,    -12
1325  0000  MCOMP,  0
1326  0000  NCOMP,  0
1327  0303  KC,     303
1330  0240  K240,   240
1331  7764  M14,    -14
1332  1336  COMIN,  END
1333  0000  V,      0;
1334  0000  0;
1335  0000  0
            END=.
            *1456
1456  0240  MES6,   240
1451  0240          240
1452  0326          326
1453  0277          277
1454  0240          240
1455  0240          240
1456  0240  MES7,   240
1457  0240          240
1460  0303          303
```

```
1461  0317            317
1462  0315            315
1463  0320            320
1464  0323            323
1465  0277            277
1466  0240            240
1467  0240            240

                *1700
1700  0000  INTEG,    0
1701  4407            JMS I 7
1702  6351            FPUT TPS
1703  0000            FEXT
1704  4405            JMS I 5
1705  4407            JMS I 7
1706  4351            FDIV TPS
1707  3346            FMPY HLF
1710  1346            FADD HLF
1711  3343            FMPY KST
1712  0000            FEXT
1713  7200            CLA
1714  1044            TAD 44
1715  7540            SZA SMA
1716  5321            JMP.+3
1717  7200            CLA
1720  5341            JMP DONE+2
1721  1342            TAD M13
1722  7450            SNA
1723  5337            JMP DONE
1724  7500            SMA
1725  7402            HLT
1726  3044            DCA 44
1727  7100  GO,       CLL
1730  1045            TAD 45
1731  7510            SPA
1732  7020            CML
1733  7010            RAR
1734  3045            DCA 45
1735  2044            ISZ 44
1736  5327            JMP GO
1737  1045  DONE,     TAD 45
1740  7004            RAL
1741  5700            JMP I INTEG
1742  7765  M13,      -13
1743  0013  KST,      13
1744  3777            3777
1745  4000            4000
1746  0000  HLF,      0
1747  2000            2000
1750  0000            0
1751  0000  TPS,      0
1752  0000            0
1753  0000            0
                PTR=74
                CHAR=110
                QUEST=35
```

```
TYPE=27
CHECK=115
POINTR=106
MESG=37
XNEXT=65
NORMX=66
LOCAD=117
CRLF=36
STORE=101
CHK=103
WAIT=217
DBCO=6505
DBSO=6506
DECPRT=1000
```

```
                *6
0006    7200    7200
                *1206
1206    5777    JMP I 1377
                *1377
1377    1521    1521
                *1521
1521    1110            TAD CHAR
1522    1353            TAD MR
1523    7650            SNA CLA
1524    5327            JMP READ
1525    4300            JMS QUEST
1526    5754            JMP I BWT
1527    1355    READ,   TAD KV
1530    4027            JMS TYPE
1531    4300            JMS QUEST
1532    1357            TAD K240
1533    4027            JMS TYPE
1534    4405            JMS I 5
1535    4407            JMS I 7
1536    6751            FPUT I VX
1537    5346            FGET ZER
1540    6752            FPUT I PSUMX
1541    0000            FEXT
1542    1356            TAD M62
1543    3103            DCA CHK
1544    5745            JMP I .+1
1545    1600            MORE
1546    0000    ZER,    0;
1547    0000    0;
1550    0000    0
1551    1631    VX,     V
1552    1634    PSUMX,  PSUM
1553    7456    MR,     -322
1554    0217    BWT,    WAIT
1555    0326    KV,     326
1556    7716    M62,    -62
1557    0240    K240,   240
                *1600
1600    6504    MORE,   DBRI
1601    7150            CLL CMA RAR
1602    3045            DCA 45
1603    7010            RAR
1604    3046            DCA 46
1605    3044            DCA 44
1606    4407            JMS I 7
1607    7000            FNOR
1610    3237            FMPY TWO
1611    3231            FMPY V
1612    2231            FSUB V
1613    1234            FADD PSUM
1614    6234            FPUT PSUM
```

```
1615   0000            FEXT
1616   2103            ISZ CHK
1617   5200            JMP MORE
1620   4407            JMS I 7
1621   5234            FGET PSUM
1622   4242            FDIV K50
1623   0000            FEXT
1624   4406            JMS I 6
1625   4630            JMS I XCRLF
1626   5627            JMP I WAITX
1627   0217  WAITX,     WAIT
1630   1504  XCRLF,     CRLF
1631   0000  V,         0;
1632   0000  0;
1633   0000  0
1634   0000  PSUM,      0;
1635   0000  0;
1636   0000  0
1637   0002  TWO,       2
1640   2000            2000
1641   0000            0
1642   0006  K50,       6
1643   3100            3100
1644   0000            0
              CHAR=110
              QUEST=1500
              WAIT=217
              CHK=103
              DBRI=6504
              TYPE=27
              CRLF=1504
```

APPENDIX 2


```
                *20
0020  0000  COD1,    0
0021  0000          0
0022  0000          0
0023  0000          0
0024  0000  SRPT,    0
0025  7700  M100,    -100
0026  0000  CODCT,   0
0027  7774  M4,      -4
0030  0000  NODCT,   0
0031  0020  CODST,   COD1
0032  0000  NODPT,   0
0033  1000  K1000,   1000
0034  0000  CLIN,    0
0035  0000  TALLY,   0
0036  0000  CSR,     0
0037  0000  CODE,    0
0040  0000  CODPT,   0
0041  0000  TEM,     0
0042  0017  K17,     17
0043  4000  K4000,   4000
0044  7772  M6,      -6
0045  0000  CHECK,   0
0046  0000  POINTC,  0
0047  0000  STORE,   0
0050  0000  CHK,     0
0051  0200  K200,    200
0052  0305  E,       305
0053  0330  X,       330
0054  0322  R,       322
0055  0316  N,       316
0056  0304  D,       304
0057  0311  KI,      311
0060  0320  P,       320
0061  0303  C,       303
0062  0000  OP,      0
0063  0000  TEMP,    0
0064  7761  M17,     -17
0065  0400  ERRX,    ERR
0066  7776  M2,      -2
0067  0240  K240,    240
0070  0212  K212,    212
0071  0215  K215,    215
0072  0467  XOUT,    OUTPUT+2
0073  1027  NORM,    TAD M4
0074  0000  TRNIN,   0
0075  1044  ARNORM,  TAD M6
0076  7100          CLL
0077  7106  NRMX,    CLL RTL
                *100
0100  0000  SREL,    0
0101  1034          TAD CLIN
0102  6545          DBCO
0103  1043          TAD K4000
0104  6505          DBCO
0105  7000          NOP
0106  6536          DBSO
0107  7240          STA
```

```
0110  6506          DBSO
0111  7300          CLA CLL
0112  5500          JMP I SRCL
0113  0000  ZER,     0
0114  2113          ISZ ZER
0115  1037          TAD CODE
0116  7104          CLL RAL
0117  3037          DCA CODE
0120  5513          JMP I ZER
0121  0000  ONE,     0
0122  1037          TAD CODE
0123  7104          CLL RAL
0124  7001          IAC
0125  3037          DCA CODE
0126  5521          JMP I ONE
0127  0000  TYPE,    0
0130  6041          TSF
0131  5130          JMP.-1
0132  6046          TLS
0133  7200          CLA
0134  5527          JMP I TYPE
0135  0000  CRLF,    0
0136  1071          TAD K215
0137  4127          JMS TYPE
0140  1070          TAD K212
0141  4127          JMS TYPE
0142  5535          JMP I CRLF
                *200
0200  7300  START,   CLA CLL
0201  6046          TLS
0202  4135          JMS CRLF
0203  3020          DCA COD1
0204  3021          DCA COD1+1
0205  3022          DCA COD1+2
0206  3023          DCA COD1+3
0207  3024          DCA SPBT
0210  1025          TAD M100
0211  3026          DCA CODCT
0212  1027          TAD D4
0213  3030          DCA NODCT
0214  1031          TAD CODST
0215  3032          DCA NODPT
0216  1033          TAD K1200
0217  3034          DCA CLIN
0220  1025          TAD M100
0221  3035          DCA TALLY
0222  4100          JMS SRCL
0223  2035          ISZ TALLY
0224  5222          JMP.-2
0225  3034          DCA CLIN
0226  4100          JMS SRCL
0227  1033          TAD K1200
0230  3034          DCA CLIN
0231  4317  NEWCOD,  JMS CODCL
0232  1025          TAD M100
0233  3036          DCA CSR
0234  7340  SKIP,    CLA CLL
0235  3037          DCA CODE
0236  1031          TAD CODST
```

```
0237  3040          DCA CODPT
0240  1027          TAD M4
0241  3035          DCA TALLY
0242  1024          TAD SRPT
0243  7041          CIA
0244  3041          DCA TEM
0245  1041  MORCOM, TAD TEM
0246  1440          TAD I CODPT
0247  7740          SZA CLA CLL
0250  4113          JMS ZER
0251  4121          JMS ONF
0252  2040          ISZ CODPT
0253  2035          ISZ TALLY
0254  5245          JMP MORCOM
0255  6504          DBRI
0256  7043          CMA
0257  6042          AND K17
0260  3074          DCA TEMIN
0261  1074          TAD TEMIN
0262  7041          CIA
0263  1037          TAD CODE
0264  7743          SZA CLA CLL
0265  4355          JMS XERR
0266  2024          ISZ SRPT
0267  2036          ISZ CSR
0270  7410          SKP
0271  5274          JMP.+3
0272  4100          JMS SRCL
0273  5234          JMP SRUP
0274  3024          DCA SRPT
0275  2026          ISZ CODCT
0276  5306          JMP UPCOD
0277  1025          TAD M100
0300  3026          DCA CODCT.
0301  7000          NOP
0302  2030          ISZ NODCT
0303  5310          JMP UPNOD
0304  7000          NOP
0305  5263          JMP START+3
0306  2432  UPCOD,  ISZ I NODPT
0307  5365          JMP NCOD
0310  2032  UPNOD,  ISZ NODPT
0311  7300          CLA CLL
0312  3020          DCA COD1
0313  3021          DCA COD1+1
0314  3022          DCA COD1+2
0315  3023          DCA COD1+3
0316  5365          JMP NCOD
0317  0000  CODCL,  0
0320  7240          STA
0321  6506          DRSO
0322  7300          CLA CLL
0323  1027          TAD M4
0324  3045          DCA CHECK
0325  1031          TAD CODST
0326  3046          DCA POINTC
0327  3047  NLOC,   DCA STORE
0330  1044          TAD M6
```

```
0331  3050          DCA CHK
0332  1446          TAD I POINTC
0333  1047  NBIT,   TAD STORE
0334  7110          CLL RAR
0335  3047          DCA STORE
0336  7004          RAL
0337  6505          DBCO
0340  1451          TAD K200
0341  6505          DBCO
0342  5372          JMP TIP
0343  6506          DBSO
0344  7240          STA
0345  6506          DBSO
0346  7300          CLA CLL
0347  2050          ISZ CHK
0350  5333          JMP NBIT
0351  2046          ISZ POINTC
0352  2045          ISZ CHECK
0353  5327          JMP NLOC
0354  5717          JMP I CODCL
0355  0000  XERR,   0
0356  1073          TAD NORM
0357  3472          DCA I XOUT
0360  1477          TAD XRMX
0361  3764          DCA I XORE
0362  4465          JMS I XRRX
0363  5755          JMP I XERR
0364  0474  XORE,   NORE-1
0365  3034  NCOD,   DCA CLIN
0366  4100          JMS SPCL
0367  1033          TAD K1200
0370  3034          DCA CLIN
0371  5231          JMP NEWCOD
0372  7200  TIP,    CLA
0373  1051          TAD K240
0374  5343          JMP NBIT+13
              *400
0400  0000  ERR,    0
0401  1052          TAD E
0402  4127          JMS TYPE
0403  1053          TAD X
0404  4127          JMS TYPE
0405  4336          JMS SP
0406  1437          TAD CODE
0407  3362          DCA OP
0410  4265          JMS OUTPUT
0411  4336          JMS SP
0412  1054          TAD P
0413  4127          JMS TYPE
0414  1053          TAD X
0415  4127          JMS TYPE
0416  4336          JMS SP
0417  1374          TAD TEXT
0420  3362          DCA OP
0421  4265          JMS OUTPUT
0422  4336          JMS SP
0423  1055          TAD I
0424  4127          JMS TYPE
0425  1056          TAD O
```

```
0426  4127          JMS TYPE
0427  4306          JMS SP
0430  1032          TAD MODPT
0431  1064          TAD C17
0432  4316          JMS DECPRT
0433  4306          JMS SP
0434  1057          TAD KI
0435  4127          JMS TYPE
0436  1060          TAD P
0437  4127          JMS TYPE
0440  4306          JMS SP
0441  1024          TAD SRPT
0442  7001          IAC
0443  4316          JMS DECPRT
0444  4306          JMS SP
0445  1061          TAD C
0446  4127          JMS TYPE
0447  1056          TAD D
0450  4127          JMS TYPE
0451  5357          JMP ROTOH
0452  3046  BAT,     DCA POINTC
0453  1027          TAD M4
0454  3050          DCA CHK
0455  1446          TAD I POINTC
0456  3062          DCA OP
0457  5362          JMP ROTOH+3
0460  2046          ISZ POINTC
0461  2050          ISZ CHK
0462  5255          JMP.-5
0463  4135          JMS CRLF
0464  5600          JMP I ERR
0465  0000  OUTPUT,  0
0466  7300          CLA CLL
0467  1027          TAD M4
0470  3035          DCA TALLY
0471  3063          DCA TEMP
0472  1062          TAD OP
0473  7002          BSW
0474  7106          CLL RTL
0475  1063  MOPR,    TAD TEMP
0476  7004          RAL
0477  3063          DCA TEMP
0500  7004          RAL
0501  1353          TAD C260
0502  4127          JMS TYPE
0503  2035          ISZ TALLY
0504  5275          JMP MOPR
0505  5665          JMP I OUTPUT
0506  0000  SP,      0
0507  1066          TAD M2
0510  3050          DCA CHK
0511  1067          TAD C240
0512  4127          JMS TYPE
0513  2050          ISZ CHK
0514  5311          JMP .-3
0515  5706          JMP I SP
0516  0000  DECPRT,  0
0517  3354          DCA VALUE
0520  3355          DCA DIGIT
```

```
0521  1350          TAD CNTRZA
0522  3356          DCA CNTRZB
0523  1347          TAD ADDRZA
0524  3331          DCA ARROW
0525  7410          SKP
0526  3354          DCA VALUE
0527  7100          CLL
0530  1354          TAD VALUE
0531  1351  ARROW,  TAD TEMPWR
0532  7430          SZL
0533  2355          ISZ DIGIT
0534  7430          SZL
0535  5326          JMP .ARROW-3
0536  7200          CLA
0537  1355          TAD DIGIT
0540  1353          TAD K260
0541  4127          JMS TYPE
0542  3355          DCA DIGIT
0543  2331          ISZ ARROW
0544  2356          ISZ CNTRZB
0545  5330          JMP ARROW-1
0546  5716          JMP I DECPRT
0547  1351  ADDRZA, TAD TEMPWR
0550  7776  CNTRZA, -2
0551  7766  TEMPWR, -12
0552  7777          -1
0553  0260  K260,   260
0554  0000  VALUE,  0
0555  0000  DIGIT,  0
0556  0000  CNTRZB, 0
0557  4306  BOTON,  JMS SP
0560  1031          TAD CODST
0561  5252          JMP BAT
0562  1075          TAD ABJORM
0563  3267          DCA OUTPUT+2
0564  1076          TAD ABJORM+1
0565  3274          DCA HOPE-1
0566  4265          JMS OUTPUT
0567  5260          JMP OUTPUT-5
```

APPENDIX 3

PRBS Delay Generation (FOCAL)

```
C-FOCAL,  5/69

01.05 C TO FIND DELAYS 0-28 TO BE EX-OR'D FROM ONE INPUT 'Z'
01.10 E
01.20 A ?Z?;S N(1)=Z;S J=1;S V=0
01.25 G 6.05
01.30 F I=1,J;S M(2*I-1)=N(I)-28;S M(2*I)=N(I)-25
01.35 S TA=0
01.40 F I=1,2,2*J;D 3
01.50 F I=1,2*J-1;D 1.6
01.55 G 1.8
01.60 I (M(I)),1.7;F L=I+1,J*2;D 4
01.70 R
01.80 S K=0;F I=1,J*2;D 5
01.90 I (TA-J)1.95,2.1,1.95
01.95 S J=K;G 1.3

02.10 T !;F I=1,J;D 2.2
02.15 T !;0
02.20 T %3,M(I),"  ";S V=V+1;I (V-10)2.4;T !;S V=0;R
02.40 R

03.10 I (M(I))3.15,3.15;I (M(I+1))3.15,3.15;R
03.15 S TA=TA+1
03.20 S M(I)=N((I+1)/2);S M(I+1)=0

04.10 I (M(L)),4.3;I (M(I)-M(L))4.3,4.2,4.3
04.20 S M(I)=0;S M(L)=0
04.30 R

05.10 I (M(I)),5.2;S K=K+1;S N(K)=M(I)
05.20 R

06.05 S SN=1/2
06.10 S SN=2*SN;I (N(1)-2S*SN)6.2,6.2,6.1
06.20 S SN=SN/2;S M(1)=N(1)-SN*28;S M(2)=N(1)-SN*25
06.30 D 1.35;G 1.35
```

APPENDIX 4

PRBS Delay Generation (FORTRAN)

```
C       PROGRAM TO FIND DELAYS TO BE EX-ORED.
C       ANGUS. W. BROWN.
        DIMENSION ST(31),AN(150)
        DO 5 K=1,31
5       ST(K)=0.0
        I=1
        J=0
        READ (1,10) TN
10      FORMAT (F10.2)
30      SN=0.5
15      SN=2.0*SN
        IF (TN-31.0*SN)20,20,15
20      SN=SN/2.0
        B1=TN-31.0*SN
        B2=TN-27.0*SN
        IF (B1-31.0)35,35,25
25      AN(I)=B2
        I=I+1
        TN=B1
        GOTO 30
35      K=B1
        ST(K)=ST(K)+1.0
        IF (B2-31.0)45,45,40
40      TN=B2
        GOTO 30
45      K=B2
        ST(K)=ST(K)+1.0
        I=I-1
        IF (I)50,60,50
50      TN=AN(I)
        GOTO 30
C       PRINTOUT SECTION.
60      DO 70 K=1,31
        L=ST(K)/2.0
        A=ST(K)/2.0
        IF (FLOAT(L)-A)65,70,65
65      WRITE (1,75) K,
75      FORMAT (I10)
        J=J+1
        IF(J-4) 70,70,90
90      J=0
        FINI
70      CONTINUE
        FINI
        STOP
        END
```

APPENDIX 5

ADDIE Distribution Curve

```
                    *0
0000  0000           0
0001  6032           KCC
0002  5403           JMP I 3
0003  0200           200
                    *20
0020  0000  POINTR,   0
0021  0000  KONST,    0
0022  0000  T1,       0
0023  7000  M1000,    -1000
0024  7750  M30,      -30
0025  3777  K3777,    3777
0026  0000  HOTST,    0
0027  0000  SAC,      0
0030  7753  SL,       7753
0031  7470  M310,     -310
0032  1000  K1000,    1000
0033  4000  K4000,    4000
0034  1144  K1144,    1144
0035  0001  K1,       1
0036  6470  M1310,    -1310
0037  0000  LOCADR,   0
0040  0000  XPOS,     0
0041  0000  SOFA,     0
0042  0012  K12,      12
                    *200
0200  7300  START,    CLA CLL
0201  1024            TAD M30
0202  3041            DCA SOFA
0203  1030            TAD SL
0204  3022            DCA T1
0205  6001            ION
0206  7402  HMEAN,    HLT
0207  7604            LAS
0210  7041            CIA
0211  1034            TAD K1144
0212  3021            DCA KONST
0213  4311            JMS CLLOC
0214  2027  BEGIN,    ISZ SAC
0215  5234            JMP GO
0216  2022            ISZ T1
0217  5214            JMP .-3
0220  7300            CLA CLL
0221  1033            TAD K4000
0222  6553            DACS3
0223  7200            CLA
0224  6552            DACS2
0225  7402            HLT
0226  5627            JMP I .+1
0227  2000            2000
0230  3040  GO,       DCA XPOS
0231  1024            TAD M30
0232  3020            DCA POINTR
0233  1033            TAD K4000
0234  6516            DBSO
0235  2020            ISZ POINTR
```

```
0236  5235          JMP.-1
0237  6505          DBC0
0240  7200          CLA
0241  6504          DBRI
0242  7040          CMA
0243  1021          TAD KONST
0244  3037          DCA LOCADR
0245  1037          TAD LOCADR
0246  1023          TAD M1000
0247  7710          SPA CLA
0250  5262          JMP PLOT
0251  1037          TAD LOCADR
0252  1036          TAD M1310
0253  7700          SMA CLA
0254  5262          JMP PLOT
0255  1437  INCLOC, TAD I LOCADR
0256  1035          TAD K1
0257  3437          DCA I LOCADR
0260  2041          ISZ SOFA
0261  5214          JMP BEGIN
0262  1031  PLOT,   TAD M310
0263  3026          DCA NOTST
0264  1024          TAD M30
0265  3041          DCA SOFA
0266  1032          TAD K1000
0267  3037          DCA LOCADR
0270  1033  MORE,   TAD K4000
0271  6553          DACS3
0272  7300          CLA CLL
0273  1437          TAD I LOCADR
0274  6551          DACS1
0275  7300          CLA CLL
0276  1040          TAD XPOS
0277  1042          TAD K12
0300  6552          DACS2
0301  3040          DCA XPOS
0302  1025          TAD K3777
0303  6553          DACS3
0304  7300          CLA CLL
0305  2037          ISZ LOCADR
0306  2026          ISZ NOTST
0307  5270          JMP MORE
0310  5214          JMP BEGIN
0311  0000  CLLOC,  0
0312  1031          TAD M310
0313  3026          DCA NOTST
0314  1032          TAD K1000
0315  3037          DCA LOCADR
0316  1033          TAD K4000
0317  3437          DCA I LOCADR
0320  2037          ISZ LOCADR
0321  2026          ISZ NOTST
0322  5316          JMP.-4
0323  5711          JMP I CLLOC
```

```
                DACS1=6551
                DACS2=6552
                DACS3=6553
                DBC0=6535
                DBS0=6536
                DBRI=6534
                *2000
2000  1032              TAD K1000
2001  3232              DCA STAT
2002  1031              TAD M310
2003  3233              DCA FINISH
2004  3040              DCA XPOS
2005  7300  AGO,        CLA CLL
2006  3234              DCA DELAY1
2007  1231              TAD M60
2010  3235              DCA DELAY2
2011  1040              TAD XPOS
2012  1042              TAD K12
2013  6552              DACS2
2014  3040              DCA XPOS
2015  1632              TAD I STAT
2016  6553              DACS3
2017  2234              ISZ DELAY1
2020  5217              JMP.-1
2021  2235              ISZ DELAY2
2022  5217              JMP.-3
2023  2232              ISZ STAT
2024  2233              ISZ FINISH
2025  5205              JMP AGO
2026  7402              HLT
2027  5633              JMP I .+1
2030  0200              200
2031  7724  M60,        -60
2032  0000  STAT,       0
2033  0000  FINISH,     0
2034  0000  DELAY1,     0
2035  0000  DELAY2,     0
```

APPENDIX 6

Initial Conditions

```
                *1
0001    5402            JMP I 2
0002    2443            SERV
                *1570
1570    5776            JMP I 1576
                *1576
1576    2200            2200
                *2200
2200    1352            TAD IIC
2201    3074            DCA PTR
2202    1110            TAD CHAR
2203    1353            TAD MP
2204    7650            SNA CLA
2205    5210            JMP GO
2206    4435            JMS I QUEST
2207    5756            JMP I RWT
2210    1110    GO,     TAD CHAR
2211    4027            JMS TYPE
2212    1357            TAD M5M
2213    3103            DCA CHK
2214    3474            DCA I PTR
2215    2074            ISZ PTR
2216    2103            ISZ CHK
2217    5214            JMP.-3
2220    1064            TAD M16
2221    3115            DCA CHECK.
2222    1754            TAD I XM3
2223    3106            DCA POINTR
2224    4437            JMS I MESG
2225    4465            JMS I XNEXT
2226    4466            JMS I NORMX
2227    1117            TAD LOCAD
2230    7041            CIA
2231    3355            DCA MNINT
2232    1360            TAD M6
2233    3115            DCA CHECK
2234    1761            TAD I XM6
2235    3106            DCA POINTR
2236    4437            JMS I MESG
2237    4405            JMS I 5
2240    4407            JMS I 7
2241    6362            FPUT V
2242    0000            FEXT
2243    4436    MOR,    JMS I CRLF
2244    1113            TAD M10
2245    3115            DCA CHECK
2246    1365            TAD MSE9
2247    3106            DCA POINTR
2250    4437            JMS I MESG
2251    4465            JMS I XNEXT
2252    4466            JMS I NORMX
2253    7240            STA
2254    1117            TAD LOCAD.
2255    1352            TAD IIC
2256    3074            DCA PTR
2257    1367            TAD K240
2260    4027            JMS TYPE
2261    1374            TAD KE
```

```
2262  4027          JMS TYPE
2263  1367          TAD K240
2264  4027          JMS TYPE
2265  4407          JMS I 7
2266  5362          FGET V
2267  0000          FEXT
2270  4770          JMS I XING
2271  3474          DCA I PTR
2272  2355          ISZ MNINT
2273  5243          JMP MOR
2274  4436          JMS I CRLF
2275  7240          STA
2276  6506          DBSO
2277  7200          CLA
2300  1371          TAD K200
2301  6505          DBCO
2302  7200          CLA
2303  1372          TAD K20
2304  6505          DBCO
2305  7000          NOP;
2306  7000  NOP;
2307  7000  NOP
2310  6506          DBSO
2311  7200          CLA
2312  1352          TAD IIC
2313  3117          DCA LOCAD
2314  1357          TAD M50
2315  3115          DCA CHECK
2316  3101  NLOC,   DCA STORE
2317  1067          TAD M14
2320  3103          DCA CHK
2321  1517          TAD I LOCAD
2322  1101  NBIT,   TAD STORE
2323  7110          CLL RAR
2324  3101          DCA STORE
2325  7006          RTL;
2326  7006  RTL
2327  6505          DBCO
2330  7200          CLA
2331  1066          TAD K400
2332  6505          DBCO
2333  7000          NOP
2334  6506          DBSO
2335  1373          TAD K10
2336  6506          DBSO
2337  7300          CLA CLL
2340  2103          ISZ CHK
2341  5322          JMP NBIT
2342  2117          ISZ LOCAD
2343  2115          ISZ CHECK
2344  5316          JMP NLOC
2345  1371          TAD K200
2346  6506          DBSO
2347  7200          CLA
2350  5751          JMP I .+1
```

```
2351  2400          CTAIM
2352  2510  IIC,    END
2353  7460  MP,     -320
2354  0576  XM3,    576
2355  0000  MNINT,  0
2356  0217  RWT,    217
2357  7730  M50,    -50
2360  7772  M6,     -6
2361  1320  XM6,    1320
2362  0000  V,      0
2363  0000          0
2364  0000          0
2365  2456  MSE9,   MES9
2366  1000  XDEC,   1000
2367  0240  K240,   240
2370  1700  XING,   1700
2371  0200  K200,   200
2372  0020  K20,    20
2373  0010  K10,    10
2374  0305  KE,     305
            *2400
2400  1250  CTAIM,  TAD MSE10
2401  3106          DCA POINTR
2402  1252          TAD M22
2403  3115          DCA CHECK
2404  4437          JMS I MESG
2405  4465          JMS I XNEXT
2406  4466          JMS I NORMX
2407  1117          TAD LOCAD
2410  7106          CLL RTL
2411  1117          TAD LOCAD
2412  7004          RAL
2413  3253          DCA TMP
2414  1253          TAD TMP
2415  7006          RTL
2416  1253          TAD TMP
2417  7004          RAL                 /TIMES 100-NO OF HZ @ 100HZ
2420  7041          CIA
2421  6133          CLAB                /AC-BUFFER PRESET REGISTER
2422  7200          CLA
2423  1255          TAD ENABLE
2424  6132          CLOE                /START CLOCK
2425  7200          CLA
2426  1254          TAD XWT
2427  3000          DCA 0
2430  5241          JMP RET
2431  7200  RESET,  CLA
2432  1251          TAD KK20
2433  6505          DBCO
2434  7000          NOP;
2435  7000  NOP;
2436  7000  NOP
2437  6506          DBSO
2440  7200          CLA
2441  6001  RET,    ION
2442  5400          JMP I 0
```

```
2443  6131  SERV,   CLSK              /CLK INTERRUPT?
2444  5400          JMP I 0           /NO-RETURN WITH INTERRUPT 0
2445  6135          CLSA              /YES-CLEAR CLOCK STATUS
2446  7200          CLA
2447  5231          JMP RESET
2450  2466  MSE10,  MES10
2451  0020  KK20,   20
2452  7756  M22,    -22
2453  0000  TMP,    0
2454  0217  XWT,    217
2455  5210  ENABLE, 5210
2456  0323  MES9,   323
2457  0314          314
2460  0317          317
2461  0324          324
2462  0240          240
2463  0316          316
2464  0317          317
2465  0240          240
2466  0303  MES10,  303
2467  0317          317
2470  0315          315
2471  0320          320
2472  0325          325.
2473  0324          324
2474  0305          305
2475  0240          240
2476  0324          324
2477  0311          311
2500  0315          315
2501  0305          305
2502  0277          277
2503  0240          240
2504  0250          250
2505  0323          323
2506  0251          251
2507  0240          240
      END=.
      PTR=74
      CHAR=110
      QUEST=35
      TYPE=27
      M16=64
      CHECK=115
      POINTR=106
      MESG=37
      XNEXT=65
      NORMX=66
      LOCAD=117
      CRLF=36
      M10=113
      M14=67
      STORE=101
      K400=66
      CLAR=6133
      CLOE=6132
      CLSA=6135
      CLSK=6131
      CHK=103
      DBC0=6505
      DBS0=6506
```

PUBLISHED PAPERS

method introduces inherent lowpass filtering. Thus the high-frequency attenuation provided by the digital filter is increased relative to that of the analogue filter.

Example of a lowpass filter: The 5th-order lowpass Bessel transfer function[6]

$$H_{LP}(s) = \frac{945}{s^5 + 15s^4 + 105s^3 + 420s^2 + 945s + 945}$$

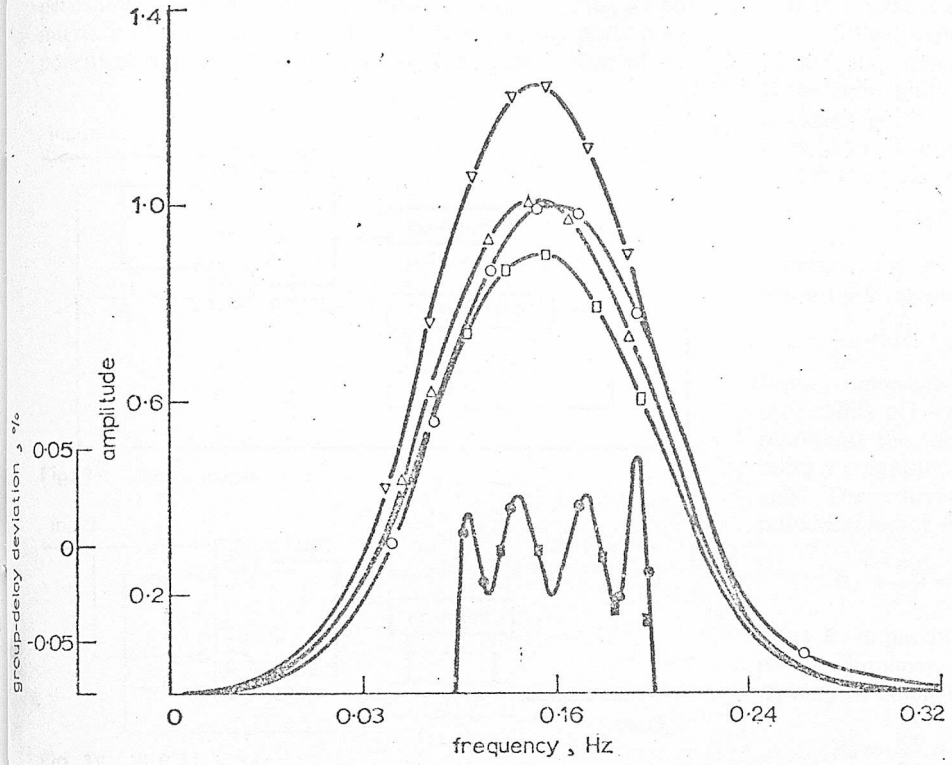


Fig. 2 *Amplitude and group-delay responses for a bandpass filter with an equiripple group delay*

Nominal group delay: 8·6508 s

Amplitude:
○○○ analogue
□□□ $\omega_D = 0{\cdot}7$ rad/s
△△△ $\omega_D = 1{\cdot}0$ rad/s
▽▽▽ $\omega_D = 1{\cdot}3$ rad/s

Group delay:
●●● analogue
■■■ digital

provides a maximally flat group-delay characteristic. This was used to obtain a corresponding lowpass discrete-time transfer function. The invariant-sinusoid method was used with $\omega_D = 1$, 2 and 3 rad/s and with a sampling frequency of 12 rad/s in each case. The amplitude responses and group-delay characteristics are shown in Fig. 1.

Example of a bandpass filter: The bandpass transfer function[7]

$$H_{BP} = 0{\cdot}287063s^2 \times [\{(s+0{\cdot}300384)^2 + 1{\cdot}10927^2\}\{(s+0{\cdot}303271)^2 + 0{\cdot}875795^2\}\{(s+0{\cdot}223361)^2 + 1{\cdot}340451^2\} \times \{(s+0{\cdot}230216)^2 + 0{\cdot}640595^2\}]^{-1}$$

provides an equiripple group-delay characteristic. The invariant-sinusoid method was used with $\omega_D = 0{\cdot}7$, 1·0 and 1·3 rad/s and with a sampling frequency of 4 rad/s in each case. The amplitude responses and group-delay characteristics are shown in Fig. 2.

Figs. 1 and 2 show that the derived digital filters preserve the flat group-delay characteristics of the corresponding analogue filters, as one would expect from eqn. 15. Also, the amplitude response in each digital filter coincides with that of the corresponding analogue filter at the design frequency ω_D, as one would expect from eqn. 16. ω_D is not critical, as long as it is chosen within the passband of the filter.

Conclusions: An approximation method has been described where a digital filter can be derived from an analogue filter that satisfies a set of specifications. The method preserves the phase response of the analogue filter, and, consequently, it is useful for the design of linear-phase or equalised digital filters. The method is less sensitive to frequency folding in comparison with the invariant-impulse method. Furthermore, it can be applied to transfer functions in which the numerator degree is as high as the denominator degree. In the invariant-impulse method, the degree of the denominator must exceed that of the numerator by at least two.

Acknowledgment: The authors are grateful to the National Research Council of Canada for supporting this research under grant no. A–7770.

A. ANTONIOU
C. SHEKHER
Department of Electrical Engineering
Sir George Williams University
Montreal 107, Que., Canada

17th September 1973

References

1 GOLD, B., and RADER, C. M.: 'Digital processing of signals' (McGraw–Hill, 1969)
2 WHITE, S. A.: 'Design and implementation of recursive digital filters'. Autonetics Report X9–359/501, 1969
3 HAYKIN, S. S., and CARNEGIE, R.: 'New method of synthetising linear digital filters based on convolution integral', *Proc. IEE*, 1970, 117, (6), pp. 1063–1072
4 WHITE, S. A.: 'New method of synthetising linear digital filters based on convolution integral', *ibid.*, 1971, 118, (2), p. 348
5 PAPOULIS, A.: 'The Fourier integral and its applications' (McGraw–Hill, 1962), p. 49
6 WEINBERG, L.: 'Network analysis and synthesis' (McGraw–Hill, 1962), p. 499
7 BLINCHIKOFF, H., and SAVETMAN, M.: 'Least squares approximation to wide-band constant delay', *IEEE Trans.*, 1972, CT-19, pp. 387–389

ADAPTIVE LOGIC CIRCUITS FOR DIGITAL STOCHASTIC COMPUTERS

Indexing terms: Computer interfaces, Logic design, Logic elements

A theoretical and experimental investigation of adaptive logic elements suitable for use as an output interface for digital stochastic computers is presented. An adaptive digital element using a binary rate multiplier is demonstrated to provide a significant improvement in accuracy without reduction in bandwidth characteristics.

The basic idea of stochastic computation was originated independently, and almost simultaneously, in the UK and

the USA.[1-3] Digital stochastic computers use probability as an analogue quantity—the probability of switching a digital circuit. It has been demonstrated that randomly switched digital logic circuits with statistically independent inputs may be used to simulate the conventional analogue operations of summation, multiplication, inversion and integration.

The output of a digital stochastic computer will generally be in the form of a nonstationary Bernoulli sequence. Such a sequence can be considered in probabilistic terms as a deterministic signal with superimposed noise. The output interface of the system must be able to reject the noise component and provide a measure of the mean value of the

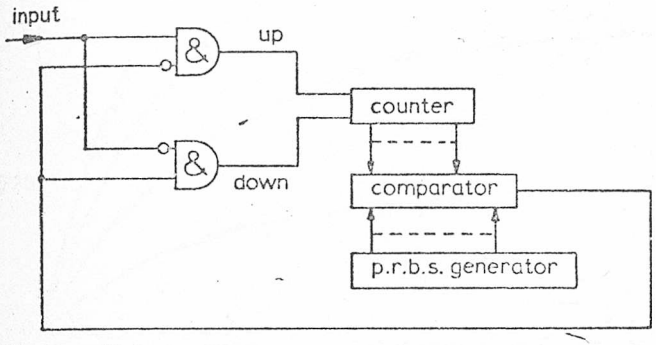


Fig. 1A *Noise* ADDIE

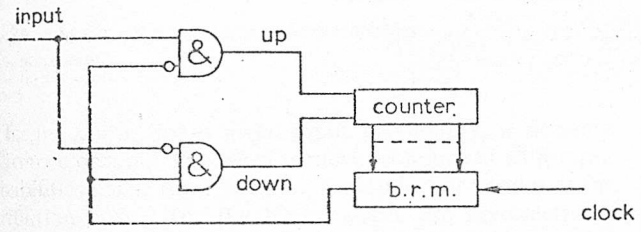


Fig. 1B *B.R.M.* ADDIE

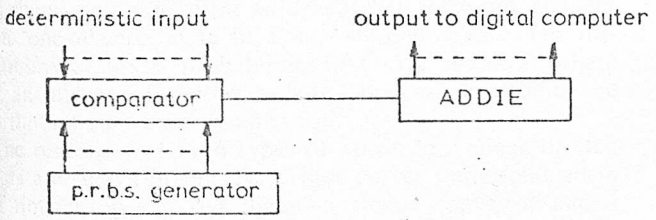


Fig. 1C *Experimental system for measurement of distribution curves*

observed sequence's generating probability. The interface must also be amenable to reasonably simple synthesis with digital logic circuits. If it is assumed that the variations of the original deterministic signal are low compared with the superimposed noise, the signal may be extracted from the noise by conventional lowpass filtering. Of the digital methods available for lowpass filtering, the technique of interest for our investigation is that of exponential smoothing.[4-6]

The digital synthesis in logic form of exponential smoothing is shown in Fig. 1A. This circuit gives an output that is a measure of the generating probability of the stochastic input sequence. The circuit is sometimes called an ADDIE, which is an acronym for adaptive digital element. It is best referred to as a noise-ADDIE structure, since it uses a feedback signal consisting of a Bernoulli sequence. If, alternatively, a deterministic feedback signal is used, the ADDIE will still form an estimate of the input sequence's generating probability. The implicit assumption in stochastic computation of statistical independence between signals is still applicable even if one of the signals is deterministic. The deterministic-feedback signal may be conveniently generated using a binary rate multiplier (b.r.m.) in the feedback loop, as shown in Fig. 1B. To differentiate this structure from the noise ADDIE, we shall refer to it as a b.r.m. ADDIE. The output frequency f_b of the b.r.m. is determined by the current state of the up–down counter and the clock frequency f_c. The generating probability of the feedback sequence is f_b/f_c. Owing to the lack of random fluctuations in the feedback sequence, the distribution function of the ADDIE states will be modified. In fact, as will be shown subsequently, the standard deviation is decreased compared with the noise ADDIE, and this provides improved accuracy without reducing the bandwidth of the interface.

The operation of the b.r.m. ADDIE can be classed as a non-Markovian process, and hence theoretical analysis is extremely difficult. However, it can be shown that a reduction in the standard deviation is to be expected, and an expression that describes the extent of the reduction is developed below.

If the noise ADDIE is in a steady-state condition, the variance of the distribution function is dependent on the variance of the count-up/count-down signals to the up–down counter. If the probability of an input pulse is p, the probability of a feedback pulse is $(1-p)$, owing to the invertor in the feedback loop. The standard deviation of the count-up sequence, σ_n, for the noise ADDIE is thus

$$\sigma_n = [Np(1-p)\{1-p(1-p)\}]^{\frac{1}{2}} \quad . \quad . \quad . \quad . \quad . \quad (1)$$

Similarly, for the b.r.m. ADDIE, the standard deviation of the count-up sequence, σ_b, is given by

$$\sigma_b = \{(1-p)\,Np(1-p)\}^{\frac{1}{2}} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (2)$$

Eqn. 1 represents the standard deviation of a sequence with probability $p(1-p)$ computed over N clock intervals. Eqn. 2 represents the standard deviation of a sequence with probability p computed over a sample size of $(1-p)\,N$ clock intervals. The reduction in standard deviation, as expressed by a reduction factor R_u, is obtained directly from eqns. 1 and 2 as

$$R_u = \frac{\sigma_n}{\sigma_b} = 1 + \frac{p^2}{1-p} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (3)$$

Thus R_u is maximum when $p = 1$ and decreases to unity as $p \to 0$. Similarly, for the count-down line, the reduction ratio R_d may be calculated as

$$R_d = \frac{\sigma_n}{\sigma_b} = \frac{1-p(1-p)}{p} \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (4)$$

This reduction ratio has a maximum at $p = 0$ and decreases to 1 as $p \to 1$. The resultant effect of both reduction ratios has been investigated experimentally for a range of input probabilities.

The basic experimental arrangement is shown in Fig. 1C. A deterministic 12-bit digital signal is compared in parallel form with a 12-bit sequence of random pulses obtained from a p.r.b.s. generator connected so as to generate a maximum-length sequence (m sequence). The resultant serial output from the comparator is a probabilistic representation of the original deterministic signal. This serial stochastic sequence is then fed to the input of an ADDIE that provides an estimate of the generating probability of the stochastic sequence. Typical state-distribution curves for the noise and b.r.m. ADDIE structures, for a given input probability, are shown in Fig. 2A. These distribution curves were plotted by connecting the output of the ADDIE to a digital computer.* Similar

* PDP 8E

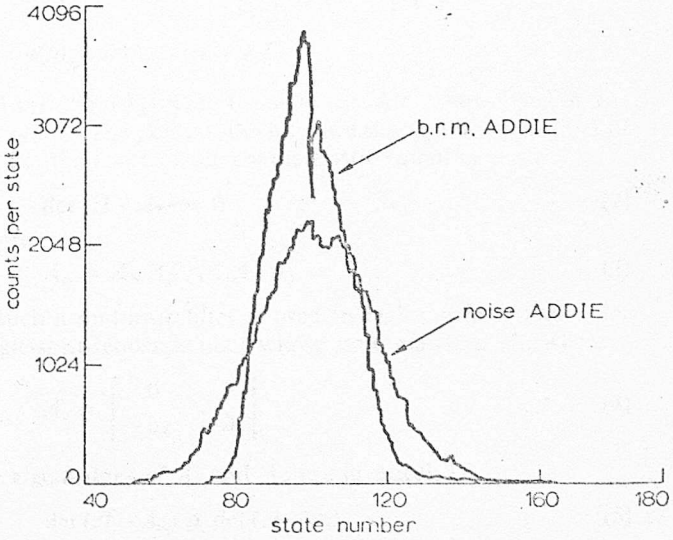


Fig. 2A *Sampled distribution of states*
Input probability = 1/16

distribution curves were obtained for a range of input probabilities.

The distribution curves for the noise ADDIE approximate well to the expected binomial distribution. As predicted for the b.r.m. ADDIE, for a given input probability, a decrease in variance occurs. The effect is most pronounced with input probabilities in the region of 0·5. It may be observed that the distribution curves for the b.r.m. ADDIE are asymmetrical, and a sharp discontinuity occurs at the mean value. This discontinuity is a characteristic of the b.r.m. It is due to a change in phase of the sequence when the b.r.m. changes from 'one-all-zero' state to a 'zero-all-one' state. The discontinuity occurs at probabilities of $1-1/2^n$ and $1/2^n$, where n is an integer. For intermediate values of probability the distribution curves are symmetrical.

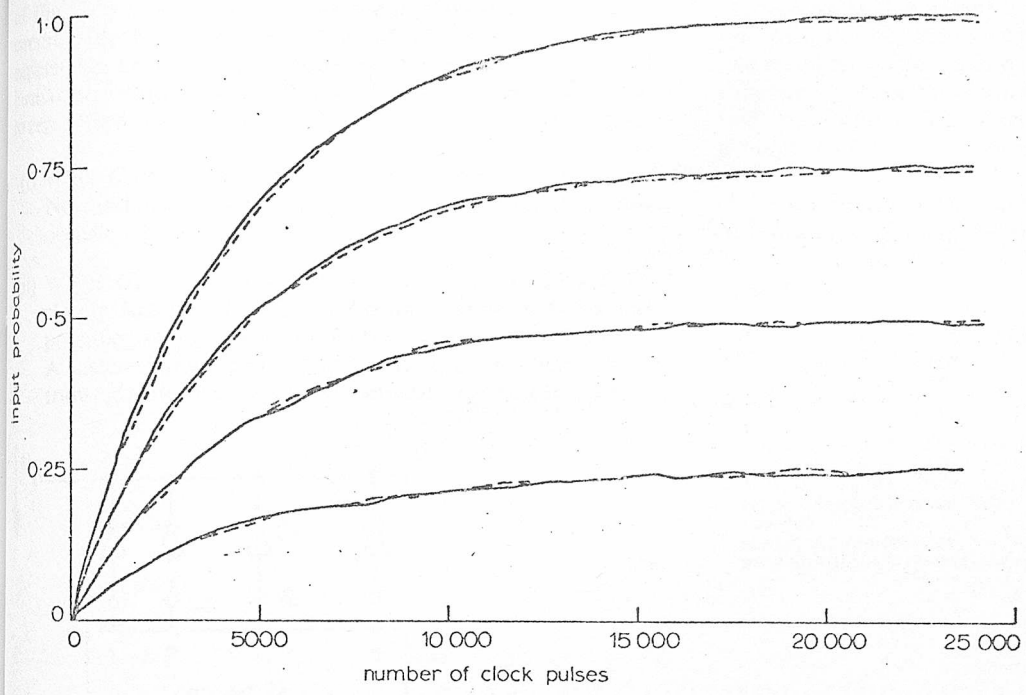


Fig. 2B ADDIE *step-response characteristics*
---- noise ADDIE
—— b.r.m. ADDIE

The responses of both types of ADDIE to a range of step inputs are shown in Fig. 2B. These curves were obtained in real time using a 12-bit digital–analogue convertor and a graph plotter. It may be observed that the response of the b.r.m. ADDIE is almost identical to that of the noise ADDIE.

An experimental and theoretical investigation has been performed of two types of adaptive digital elements suitable for use as an output interface for a digital stochastic computer. Compared with the noise ADDIE, the b.r.m. ADDIE has been demonstrated to provide an improvement in accuracy without reducing the bandwidth characteristics. From an economic viewpoint, the b.r.m. ADDIE is marginally cheaper than the noise ADDIE, using currently available digital integrated circuits. The b.r.m. ADDIE is to be used as the standard output interface in a digital stochastic computer DISCO presently under construction.

A. J. MILLER
A. W. BROWN
P. MARS

School of Electronic & Electrical Engineering
Robert Gordon's Institute of Technology
Aberdeen AB9 1FR, Scotland

19th September 1973

References

1 GAINES, B. R.: 'Stochastic computing', *AFIPS SJCC*, 1967, 30, pp. 149–156
2 POPPELBAUM, W. J., AFUSO, C., and ESCH, J. W.: 'Stochastic computing elements and systems', *AFIPS FJCC*, 1967, 31, pp. 635–644
3 RIBEIRO, S. J.: 'Random pulse machines', *IEEE Trans.*, 1967, EC–16, pp. 261–276
4 BROWN, R. G., and MEYER, R. F.: 'The fundamental theorem of exponential smoothing', Proceedings of 10th national meeting of ORSA, San Francisco, USA, 1956
5 BROWN, R. G.: 'Smoothing, forecasting and prediction of discrete time series' (Prentice-Hall, 1962)
6 ANDREAE, J. H.: 'Learning machines—a unified view' *in* 'Encyclopaedia of information, linguistics and control' (Pergamon Press, 1968)

POLE SENSITIVITY OF A DIGITAL FILTER WITH MULTISHIFT SEQUENCES IN EACH SAMPLING INTERVAL

Indexing terms: Digital filters, Poles and zeros, Sensitivity

The pole sensitivity of a digital filter with multishift sequences in each sampling interval is investigated. A criterion is established to compare the pole sensitivity of a multirate filter with that of a single-rate filter, and the area for less-sensitive poles is plotted.

Fjällbrant[1] has proposed a digital filter with shift registers such that the shifting is continued N times during each pulse-repetition interval, while the filter multiplication coefficients are also allowed to take on different values for the different shift sequences. If a 2nd-order digital filter, realised by the direct configuration (Fig. 1A), has N shift sequences during each sampling interval, while its coefficients are allowed to take on different values every T/N seconds, so that α_{1j} and β_{1j} are the coefficients at nT, α_{2j} and β_{2j} are the coefficients at $\{n+1/N\}\,T$, and α_{ij} and β_{ij} are the coefficients at $\{n+(i-1)/N\}\,T$, then its state equation is given by[2]

$$\begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = A_N A_{N-1} \ldots A_2 \left\{ A_1 \begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} + Bu(n) \right\} \qquad (1)$$

where

$$A_i = \begin{bmatrix} 0 & 1 \\ -\beta_{i2} & -\beta_{i1} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$u(n)$ = input at $t = nT$

and $x_1(n)$ and $x_2(n)$ are the state variables, shown in Fig. 1A, at $t = nT$. The poles of the filter are simply the eigenvalues of A_s, i.e. the roots of the characteristic equation

$$\det(zI - A_m) = 0 \qquad (2)$$

where

$$A_m = A_N A_{N-1} \ldots A_2 A_1 \qquad (3)$$

If such a multirate filter is used to realise a fixed 2nd-order single-shift-sequence filter whose state matrix is A_s, where

$$A_s = \begin{bmatrix} 0 & 1 \\ -b_2 & -b_1 \end{bmatrix} \qquad (4)$$

the eigenvalues of A_s and A_m are identical, i.e.

$$\det(zI - A_m) \equiv \det(zI - A_s) \qquad (5)$$

Let Λ and Λ^* be the eigenvalues of A_s, and, if only complex

on certain nodes during the procedure. If the discharging circuit is nonlinear, which is so in b.b. circuits, this remains unequalised. One must prove that such effects do not occur during the bucket-brigade discharging procedure. The circuit of Fig. 2 is a model of the discharging process in the b.b. circuit. Ideal diodes are connected to each node of the capacitance array. All the anodes are connected to a common busbar, the voltage of which decreases from a positive value to zero. Two statements are valid for this discharging process:

(i) While $U_k < U$, the kth diode is in a nonconducting state. No discharging is possible, and overdischarging cannot occur.

(ii) When $U_k = U$ is reached, the decrease of U will be closely followed by U_k. A slower change in U_k is not possible, owing to the conducting state of the kth diode. A faster change is also impossible, because the voltage-transfer ratio of a capacitive network cannot exceed 1.

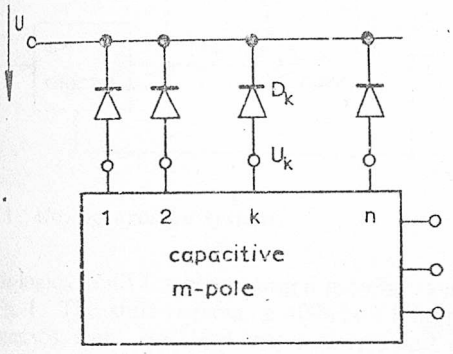


Fig. 2 *Model for discharging process of bucket-brigade arrays*

These two statements prove that the discharge process demonstrated in Fig. 2 forces each array node voltage to decrease just to zero; thus overdischarging cannot occur.

Based on the foregoing, the following conclusion can be reached: the bucket-brigade circuits can be built with an arbitrary capacitive m-pole instead of the simple capacitance array. For open outputs, the charge extracted from an array node is equal to the charge previously filled into the same node and is independent of the charging of the other nodes. Thus the line is nondispersive. The application of the condition $Q_o = 0$ (open outputs) in eqn. 2 yields

$$\Delta U_o = -(C_{oo} - C_{oa}\, C_{aa}{}^{-1}\, C_{ao})^{-1}\, C_{oa}\, C_{aa}{}^{-1}\, Q_a = WQ_a \quad (6)$$

This equation provides the basis for the investigation of the capacitive coupling network. The matrix W contains all the weighting factors that are valid from the array nodes to the output nodes. From the aspect of the design, one should first choose some kind of topology for the capacitive network, and then use the individual capacitance matrix of the same. For example, for the application of the network of Fig. 1*a*, the capacitance matrix is

$$\left[\begin{array}{cccc|c} C_{a1}+C_{w1} & 0 & \dots & 0 & -C_{w1} \\ 0 & C_{a2}+C_{w2} & \dots & 0 & -C_{w2} \\ \vdots & \vdots & & \vdots & \\ 0 & 0 & \dots & C_{an}+C_{wn} & -C_{wn} \\ \hline -C_{w1} & -C_{w2} & & -C_{wn} & C_s+\sum C_w \end{array}\right] \quad (7)$$

If the value of each capacitance C_a is chosen to be $C_a + C_w$ = constant, a very simple design formula can be derived from eqn. 6:

$$U_o = \frac{\sum_{i=1}^{n} C_{wi}\, Q_i}{(C_a+C_w)(C_s+\sum C_w)-\sum C_w{}^2} \quad . \quad . \quad . \quad . \quad (8)$$

i.e. the realised weighting factors are proportional to the capacitances C_w.

Similarly for other networks, only positive weighting factors can be realised by the capacitive-coupling network. It creates no problem because of the possibility of multiple outputs. Two outputs must be created: the first to be in accordance with the positive weighting factors, the second one with the negative weighting factors. The signals of the two outputs will be fed into a differential amplifier. When the capacitive network is equipped with a number of (more than two) output nodes, one single network will be able to produce a number of differently filtered output signals.

Acknowledgments: The author wishes to thank Prof. I. P. Valkó and K. Tarnay for their helpful suggestions.

V. SZÉKELY

Technical University of Budapest
1521 Budapest, Hungary

3rd September 1974

References

1 SANGSTER, F. L. J.: 'The bucket-brigade delay line: a shift register for analogue signals', *Philips Tech. Rev.*, 1970, 31, pp. 97–110
2 BUSS, D. D., BAILEY, W. H., and COLLINS, D. R.: 'Matched filtering using tapped bucket-brigade delay lines', *Electron. Lett.*, 1972, 8, pp. 106–107
3 SZABÓ, Z., and SZÉKELY, V.: 'Analóg léptetöregiszterek és vastagréteg megvalósításuk', *Hiradástechnika*, 25, pp. 167–175

MOVING-AVERAGE OUTPUT INTERFACE FOR DIGITAL STOCHASTIC COMPUTERS*

Indexing term: Computer interfaces

A new logic circuit suitable for use as an output interface for digital stochastic computers is presented. The logic element is based on the theory of moving averages, and is demonstrated to provide a significant improvement in transient-response characteristics, without loss in accuracy.

A previous letter was concerned with the problem of adaptive logic circuits suitable for use as an output interface in digital stochastic computers.[1] The machine variable of a stochastic computer is the probability of an ON logic level occurring at any given clock interval. The logic circuitry required to measure this probability and convert it to a more convenient binary number form must be capable of counting all the observed ON logic levels and performing some type of averaging operation. The general equation that describes all averaging techniques can be written in the form

$$P_N = a_1 A_N + a_2 A_{N-1} + a_3 A_{N-2} + \dots + a_i A_{N-i+1} + a_N A_1 \quad (1)$$

where P_N is the average calculated over N sample points. The coefficients a_i are referred to as weights, and the values of A_i in the binary case are 1 or 0. For the estimate to be unbiased,

$$\sum_{i=0}^{N} a_i = 1 \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (2)$$

If the sample length N is extended to infinity, eqn. 2 may be satisfied by making the coefficients $a_1, a_2, a_3, \dots$ a geometric sequence. The average given by eqn. 1 may then be written as

$$P_N = (1-\alpha)A_N + \alpha(1-\alpha)A_{N-1} + \alpha^2(1-\alpha)A_{N-2} + \dots + \alpha^n(1-\alpha)A_{N-n} = (1-\alpha)A_N + \alpha P_{N-1} \quad (3)$$

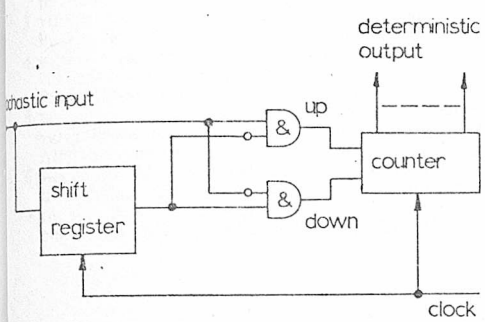
This method of averaging is known as exponential averaging or exponential smoothing. It provides the theoretical basis of the noise ADDIE reported previously.[1] Alternatively, if the

* The authors wish to thank one of the referees for indicating that the use of a moving-average circuit has been previously mentioned in private correspondence by G. V. Pallottino of the Laboratorio Di Ricerca E Tecnologia Per Lo Studio Del Plasma Nello Spazio, Rome. This correspondence was, of course, unknown to the authors

mple length N is finite, eqn. 2 may be satisfied by making l the weights equal to $1/N$. Under this condition, eqn. 1 duces to

$$P_N = \frac{1}{N}\sum_{i=1}^{N-1} A_i + A_N = P_{N-1} + \frac{A_N - A_0}{N} \quad . \quad . \quad (4)$$

e estimate P_N is now referred to as a short-time or moving rage. An apparent disadvantage of the moving-average thod, compared with exponential smoothing, is the uirement, at any time, to store all logic input levels served in the previous N clock intervals. However, since ly the first and last levels are used in the calculation at any e time, a conventional serial-in/serial-out shift register of gth N may be used for a memory.



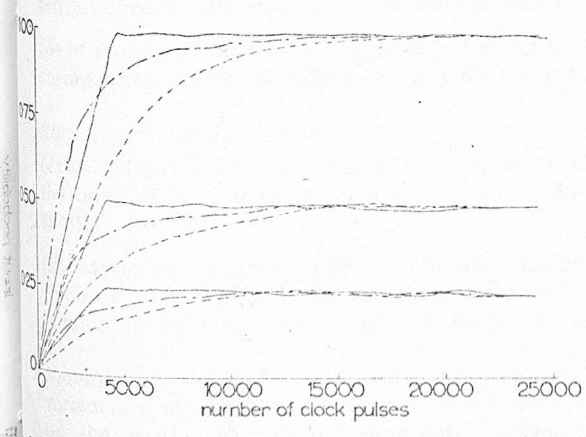
g.1 *Moving-average system*

The logic circuit for generating a moving average is shown Fig. 1. The shift register is 4096 bits long and is simply nstructed from standard m.o.s. integrated circuits. The unter must have the same capacity as the shift register, to ure an unbiased reading by inherent division of the tor N. In operation, with an input Bernoulli sequence of bability $p = 0{\cdot}5$, the standard deviation σ of the output is

$$\sigma = \sqrt{\frac{pq}{N}} = \tfrac{1}{2}\sqrt{\frac{1}{4096}} = 0{\cdot}008$$

a standard deviation of approximately 1% of full scale.

The response of the moving-average circuit to a step input hown in Fig. 2. Also shown are the responses of a noise DIE and a b.r.m. ADDIE of the same accuracy. The improve-nt obtained in transient-response characteristics will be nificant in improving the overall frequency-response aracteristics of stochastic computing systems. It should noted that the moving-average circuit, unlike the expo-ntial smoother, is an open-loop device. To avoid the umulation of errors, it is necessary to set the circuit to the o state before each reading. Apart from stochastic com-ation, the moving-average circuit has been applied erimentally to deterministic pulse-rate measurements. If a stationary Bernoulli sequence is considered as a deter-istic signal with superimposed noise, it is apparent that moving-average circuit may be used for frequency-to-ber conversion. In such applications, it is necessary to provide additional circuitry to synchronise the input with the shift pulses of the shift register. As with the stochastic case, the clock frequency controls the sample length, but the sample must now be thought of in terms of a time period rather than a fixed number of clock intervals. The reading of the counter will depend on the clock frequency as well as the length of the register. The maximum frequency that may be measured is the clock frequency, and the minimum reading is directly dependent on the length of the shift register.



2 *Step-response characteristics*
moving average
noise ADDIE
b.r.m. ADDIE

Acknowledgment: The authors wish to gratefully acknowledge the support of a UK Science Research Council research grant.

A. J. MILLER
A. W. BROWN
P. MARS

School of Electronic & Electrical Engineering
Robert Gordon's Institute of Technology
Aberdeen AB9 1FR, Scotland

4th September 1974

Reference

1 MILLER, A. J., BROWN, A. W., and MARS, P.: 'Adaptive logic circuits for digital stochastic computers', *Electron. Lett.*, 1973, 9, pp. 500–502

MONOLITHIC POLYCRYSTALLINE-SILICON PRESSURE TRANSDUCER

Indexing terms: Monolithic integrated circuits, Pressure transducers

A monolithic pressure transducer using polycrystalline silicon for both the diaphragm material and an integral piezoresistor has been fabricated. The device can be made with good repeatability and with easily varied diaphragm thickness. Electrical-output linearity is very good over a pressure range of 0–11 cm Hg for a 2·4 μm diaphragm having an area of 0·00136 cm^2.

Introduction: A new form of integrated silicon pressure transducer has been fabricated using a thin polycrystalline diaphragm supported by a relatively thick silicon rim. A single piezoresistor is made by ion implantation directly in the polycrystalline silicon. The polycrystalline pressure transducer can be made with fewer processing steps and greater reproducibly than single-crystal monolithic pressure transducers.

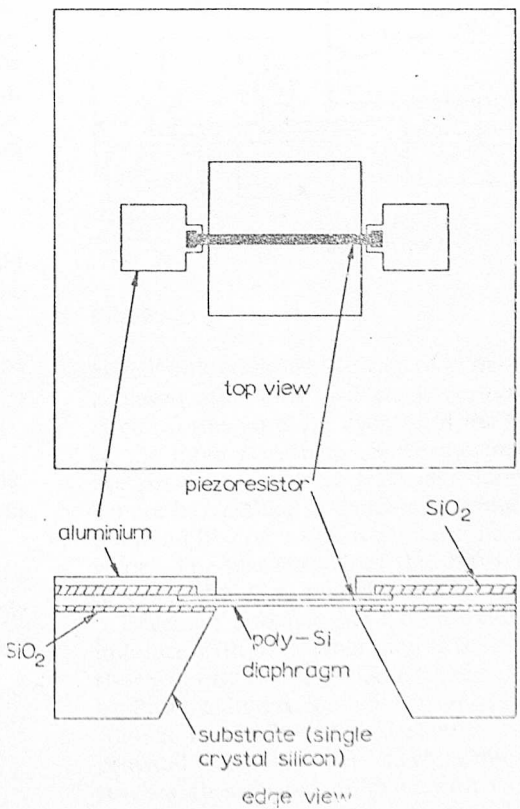


Fig. 1 *Polycrystalline-silicon pressure transducer*

Acknowledgment: The authors wish to express their gratitude to Prof. A. K. Kamal for advice and encouragement.

R. PARTHASARATHY
H. SINGH

Department of Electronics & Communication Engineering
University of Roorkee
Roorkee, India

22nd May 1975

References

1 LAL, M., and SINGH, H.: 'On minimal realization from symmetric transfer function matrix', *Proc. Inst. Elec. Electron. Eng.*, 1972, 60,(1), pp. 139–140
2 PURI, S., and TAKEDA, H.: 'Minimal realization of a symmetric transfer function matrix using moments', *IEEE Trans.*, 1973, AC–18, pp. 305–306
3 YOULA, D., and TISSI, P.: '*n*-port synthesis via reactance extraction. Pt. 1'. IEEE International Convention Record 1966, 14, pp. 183–208
4 KUH, E. S., LAYTON, D. M., and TOW, J.: 'Network analysis and synthesis via state variables' *in* 'Network and switching theory' (Academic Press, 1968), p. 154
5 HO, B. L., and KALMAN, R. E.: 'Effective construction of linear state-variable models from input/output data'. Proceedings of the third Allerton conference on circuit and system theory, 1965, pp. 449–459
6 SHAMASH: 'Minimal realization of a differential system'. Proceedings of international conference on systems and control, 1973
7 DE RUSSO, P. M., ROY, R. J., and CLOSE, C. M.: State variables for engineers' (Wiley, 1967), chap. 4, pp. 252–253

OPTIMUM CRITERIA FOR OUTPUT INTERFACES IN DIGITAL STOCHASTIC COMPUTERS

Indexing terms: Computer interfaces, Logic design, Logic devices

Optimum criteria are discussed for the evaluation of output interface systems in digital stochastic computers. In particular, the ideal probability sensing device is described, as well as a practical realisation of a variable-accuracy averager.

The moving-average output interface[1] is a device for continually estimating the average rate of a random pulse train. This average is computed over a time interval dictated by the circuits' counter length and master-clock frequency. When the input pulse sequence is synchronous with the clock, the moving-average circuit forms a measure of the input sequence's generating probability. Thus the circuit may be used as an output interface for a stochastic computing system.

The properties governing the efficiency of output interfaces are similar to those of conventional statistical estimation, and may be summarised as follows:

Steady-state characteristics:

(*a*) Minimum bias error. Ideally, the output interface should provide an unbiased estimate with the expected value of the output equal to the mean of the measured variable.

(*b*) Minimum variance. The variance of the output should be commensurate with the length of the sample taken.

Step-response characteristics:

(*c*) Minimum response time to minimum bias error. Obviously, the time taken to reach an unbiased estimate should be as short as possible.

(*d*) Minimum response time to minimum variance. The response time should equal the theoretical minimum time required to compute an estimate with the desired variance.

Nonstationary input characteristics: For nonstationary inputs, variants on all the previous properties will exist, depending on the nature of the nonstationarity. One important characteristic is the following:

(*e*) Ability to track nonstationary inputs. Ideally, the output interface should be able to continually track a changing input value without the necessity of resetting the circuit after each reading.

The ideal output interface for a stochastic system is one such that the normalised variance decreases with time and all readings form unbiased estimates of the input probability. Thus the most significant bit of the binary-number estimate would stabilise first, followed by progressive stabilisation of further bits as the normalised variance decreases. An arithmetic operation such as this requires continual division and is beyond the bounds of a simple digital realisation.

The moving-average output interface provides the minimum variance possible after N clock periods and maintains this minimum. However, for the first $N-1$ clock periods, before the sample space is filled, the probability estimates are unusable. Clearly the circuit satisfies (*a*), (*b*), (*d*) and (*e*) of the above criteria.

An output interface recently described by Pratapa Reddy[2] reduces the time taken to achieve an unbiased estimate, i.e. improves property (*c*) and also satisfies property (*d*) by providing a variance proportional to the number of clock pulses counted. However, the claim that the variance associated with counting N trials of a Bernoulli sequence can be achieved after only $N/2$ clock periods is misleading. The circuit forms an N bit average of the first and last $N/2$ trials of the input sequence. Since the first count is never lost, the circuit is unable to track nonstationary inputs and thus cannot be classified as a moving average. To avoid confusion, it should be noted that the circuit diagram[2] contains a slight error. The two outputs of the shift register, of which one is inverted, should be interchanged for correct operation.

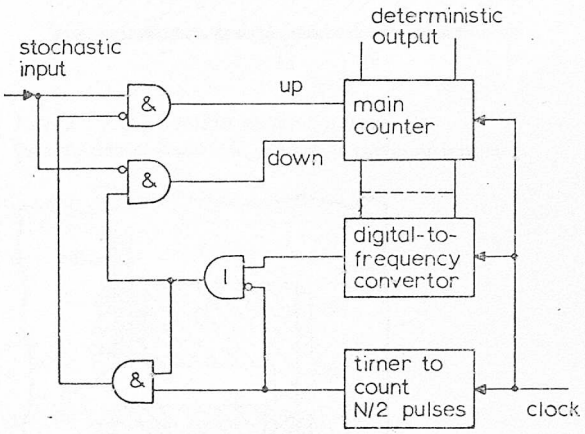


Fig. 1

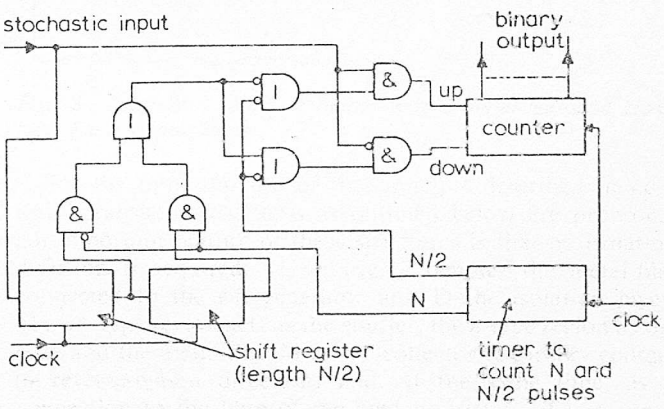


Fig. 2

Recently, attempts have been made to design an output interface with a variable sample length. One such circuit is shown in Fig. 1. The device uses the basic circuit idea proposed by Pratapa Reddy for halving the response time. The main counter is initially set to a state corresponding to an input probability of 0·5. After $N/2$ clock intervals, the main counter reaches an unbiased estimate with variance of the order of $2/N$. At this time, the feedback signal is connected to the input of the integrator, changing the circuit into a standard

ADDIE with deterministic feedback.[3] The accuracy of the estimate then steadily improves, finally giving a distribution function of variance proportional to $1/2N$. It should be noted that the ramp response of the circuit provides the ADDIE with an initial condition, which is itself an acceptable probability estimate. Subsequently, the ADDIE operates on additional data, using its extended memory capacity to decrease the normalised variance.

A more complex output interface structure is shown in Fig. 2. This circuit switches to an N-stage shift register after N clock periods and provides a true moving average. The improvement in the response time to minimum bias error is achieved solely at the cost of additional circuitry.

A. J. MILLER

19th June 1975

GEC Hirst Research Centre
East Lane, Wembley, England

A. W. BROWN
P. MARS

School of Electronic & Electrical Engineering
Robert Gordon's Institute of Technology
Schoolhill, Aberdeen AB9 1FR, Scotland

References

1 MILLER, A. J., BROWN, A. W., and MARS, P.: 'Moving-average output interface for digital stochastic computers', *Electron. Lett.*, 1974, 10, pp. 419–420
2 PRATAPA REDDY, V. C. V.: 'Moving-average output interface for digital stochastic computers', *ibid.*, 1975, 11, pp. 166–167
3 MILLER, A. J., BROWN, A. W., and MARS, P.: 'Adaptive logic circuits for digital stochastic computers', *ibid.*, 1973, 9, pp. 500–502

EXTREMELY HIGH PACKAGING DENSITIES BY OPERATING INTEGRATED CIRCUITS WITH R.F. POWER

Indexing terms: Integrated circuits, Packaging

A binary circuit configuration in which the transistors operate with r.f. power is described. This mode of operation allows the use of very simple circuitry. I.C. configurations in which this principle is exploited to maximum advantage are proposed.

The principle is illustrated with the aid of a circuit configuration implemented with discrete devices (Fig. 1). A sinewave or square-wave r.f. supply voltage of 5 V relative to ground is applied to terminal S. The frequency may be, for example, 15 MHz. The r.f. current flows through the capacitors C_1, C_2 to the transistors. Since the voltage at the transistors is always far below the supply voltage, the capacitors are r.f. current generators with respect to the transistors. The transistors T_1, T_2 form a logical unit cell. When at least one of the two transistors is turned on by a positive drive signal applied to terminals I_1, I_2, the r.f. current flowing through capacitor C_1 is shorted. Since no current is able to flow into the emitter junction of transistor T_3, the latter is turned off. If, however, terminals I_1, I_2 are connected to ground potential, i.e. if transistors T_1, T_2 are off, the positive halfwave of the r.f. current will flow through the emitter

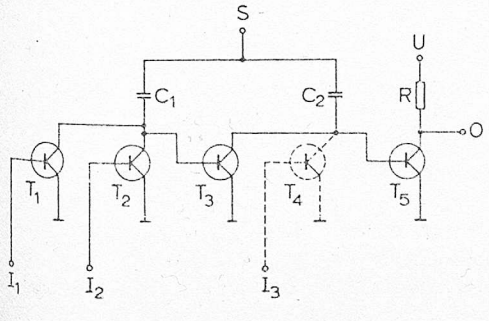


Fig. 1 *Circuit configuration with discrete devices operated with r.f. power*

junction of transistor T_3, which is thereby turned on to short the r.f. current flowing through the capacitor C_2. It is practical to operate the collector of the output coupling stage T_5 with direct voltage, e.g. 5 V. When the transistor T_3 is driven, there is constant operating voltage at the output O. If, however, the transistor T_3 remains turned off, the output voltage will fluctuate between ground potential and the operating voltage U at the rate of the r.f. voltage. Both this r.f. output signal and the mean d.c. value filtered from it can be used to drive further circuits. The transistor T_4 is added to indicate how the configuration can be expanded into a logic network of any required complexity.

It is of decisive interest that capacitors C_1, C_2 need not also be integrated. Instead, either the isolation capacitance may be used or the entire surface of the semiconductor device can be covered with an isolation layer on which is deposited a metal film that is connected to the r.f. generator.

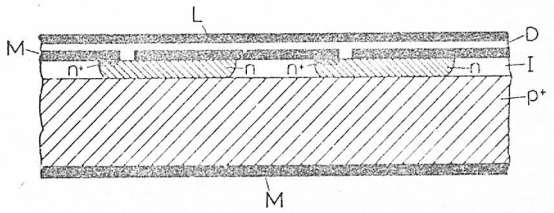


Fig. 2 *I.C. operated with r.f. power*
Schottky contact as collector. No isolation diffusion is required

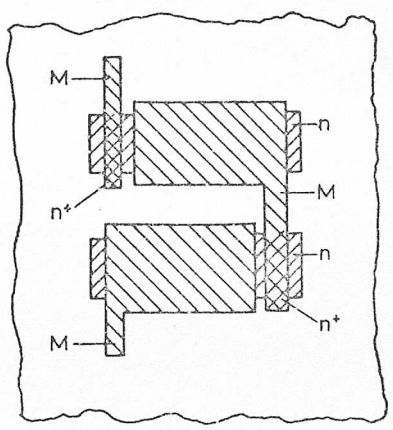


Fig. 3 *Transistor forms at points where metal leads M cross over the n-doped leads*

For the optimum use of the principle described in i.c.s, novel transistor structures as outlined below are proposed. An important feature of these structures is that no isolation diffusion is required. L in Fig. 2 denotes the metal film connected to the r.f. generator and D the isolation layer. The p^+-type region acts as the emitter, the n-type region as the base and the metal film M as the collector (Schottky contact in reverse-biased direction) and, at the same time, as a connection to the base of the next transistor. To ensure a base contact without any barrier-layer effect, the n^+-type region is inserted on one side of the base. The I region ($> 10^2$ cm) isolates the metal film M. No isolation diffusion or oxide isolation is necessary, because the maximum voltage to be isolated is 0·6 V, a large fraction of which is taken up by the potential barrier. The residual isolation current is negligible. It should be noted that the configuration shown in Fig. 2 represents a p–n–p transistor.

Fig. 3 shows how the transistor structure of Fig. 2 can be further simplified. The upper n-doped leads do not terminate at the base, but a transistor is formed at the point where a metal lead M crosses over an n-type lead. The n^+-type terminal is the base connection. If desired, several collector leads can cross over one base lead. The transistor structure in Fig. 3 allows large geometric tolerances, because the various regions are not interdigitated as in conventional transistors. This opens the way to very small transistor structures and, consequently, high packaging densities.

Only the positive halfwave of the r.f. voltage can pass the base of the transistors. To prevent the configuration from

INT. J. ELECTRONICS, 1974, VOL. 37, NO. 5, 637–655

A study of an output interface for a digital stochastic computer

A. J. MILLER†, A. W. BROWN† and P. MARS†

A theoretical and experimental investigation of adaptive digital elements suitable for use as an output interface for digital stochastic computers is presented. An adaptive digital element using a binary rate multiplier is demonstrated to provide a significant improvement in accuracy without reduction in bandwidth characteristics. The experimental results are shown to be in close agreement with the various theoretical predictions.

List of principal symbols

A_i = binary random variable
a_i = constant involved in averaging process
E = analogue input voltage
f_c = clock frequency
f_b = output frequency of BRM
M = mean state of ADDIE
N_s = sample size
N = total number of counter states
$n(t)$ = state of ADDIE at time t
p = generating probability of stochastic sequence
$\hat{p}_{N_s}$ = estimate of p after N_s samples
$q = 1-p$
R_u = standard deviation reduction ratio on count-up line of ADDIE
R_d = standard deviation reduction ratio on count-down line of ADDIE
S_t = estimate of value of p at time t
v_t = voltage at time t
V_o = voltage corresponding to ON logic level
V = maximum value of analogue input voltage
α = constant used in exponential smoothing
$\pi_{ij}(t)$ = probability of changing from state i to state j at time t
$\pi_i(t)$ = probability of being in state i at time t
$\omega_{3\ dB}$ = 3 dB frequency of ADDIE
ϵ = error of ADDIE
σ_n = standard deviation of noise ADDIE
τ = circuit time constant
$\varepsilon\{A(i)\}$ = expected value of A at the ith clock interval
Δt = width of clock pulse

1. Introduction

The basic idea of stochastic computation was originated independently and almost simultaneously in the U.K. and the U.S.A. (Gaines 1967 a, Poppelbaum

Received 6 July 1973.
† Robert Gordon's Institute of Technology, School of Electronic and Electrical Engineering, Schoolhill, Aberdeen AB9 1FR.

1967, Ribeiro 1967). Digital stochastic computers use probability as an analogue quantity—the probability of switching a digital circuit. It has been demonstrated that randomly-switched digital logic circuits with statistically independent inputs may be used to simulate the conventional analogue operations of summation, multiplication, inversion and integration. The stochastic computer is not quite as fast as an analogue computer and it is not as accurate as a digtal computer but it possesses a speed-size-economy combination which cannot be matched by conventional machines (Gaines 1967 b). Recently some potential applications of the stochastic computer have been surveyed (Mars 1972).

Physical quantities within the stochastic computer are represented by the probability that a logic level in a clocked sequence will be ON. Since probability varies from 0 to 1 most physical variables have to be mapped into the allowable range of computer variables. The actual mapping used substantially influences the simplicity of the hardware required for a specific computation. Of the possible mappings which have been investigated (Gaines 1969) the mapping of most interest for the present study is the bipolar representation given by

$$p(\mathrm{ON}) = \tfrac{1}{2} + \tfrac{1}{2}\frac{E}{V} \tag{1}$$

where $-V \leqslant E \leqslant V$. Here $p(\mathrm{ON})$ represents the probability of occurrence of a pulse, E is the analogue input signal and V represents the maximum value of E. For this mapping both positive and negative quantities may be represented on one line. Clearly for maximum positive quantity $E = V$ and $p(\mathrm{ON}) = 1$, and for maximum negative quantity $E = -V$, $p(\mathrm{ON}) = 0$. Zero is represented by a logic level with an equal probability of being ON or OFF. The theoretical basis of stochastic computation is provided by finite state automata theory (Booth 1967, Mars 1968).

The output of a digital stochastic computer will generally be in the form of a non-stationary Bernoulli sequence. Such a sequence can be considered in probabilistic terms as a deterministic signal with superimposed noise. The output interface of the system must be able to reject the noise component and provide a measure of the mean value of the observed sequences generating probability. The interface must also be amenable to reasonably simply synthesis with digital logic circuits. If it is assumed that the variations of the original determinstic signal are low compared with the superimposed noise, then the signal may be extracted from the noise by conventional low-pass filtering. Of the digital methods available for low-pass filtering the technique of most interest for our investigations is that of exponential smoothing (Brown and Meyer 1956, Brown 1962).

2. Theoretical analysis

2.1. *Exponential smoothing*

The process of averaging may be described by the equation

$$\hat{p}_{N_s} = a_1 A_{N_s} + a_2 A_{N_s-1} + \ldots + a_i A_{N_s-1+i} + \ldots + a_{N_s} A_1 \tag{2}$$

Here A_i ($=0$ or 1) is the value of the input sequence at the ith clock pulse. For the estimate $\hat{p}_{N_s}$ to be unbiased

$$\sum_{i=0}^{N_s} a_i = 1 \tag{3}$$

If all the constants a_i in eqn. (2) are equal, then eqn. (2) describes a moving average as given by

$$\sum_{i=1}^{N_s} a_i = N_s a = 1$$

or

$$\hat{p}_{N_s} = \frac{1}{N_s} \sum_{i=1}^{N_s} A_i$$

Any combination of the coefficients a_i which obeys eqn. (3) may be used to form an average of the variables A_i. If N_s is considered to extend to infinity then eqn. (3) may be satisfied by considering the variables a_i in the form of a geometric sequence. Thus we have

$$a_2 = \alpha a_1, \quad a_3 = \alpha^2 a_1, \quad \ldots, \quad a_{N_s} = \alpha^{N_s - 1} a_1$$

In this case

$$\sum_{i=1}^{N_s} a_i = a_1 \cdot \frac{1 - \alpha^{N_s}}{1 - \alpha}$$

and if $\alpha < 1$

$$\underset{N_s \to \infty}{\mathscr{L}t} \left[\sum_{i=1}^{N_s} a_i \right] = \frac{a_1}{1 - \alpha}$$

Clearly if $a_1 = 1 - \alpha$, the original eqn. (3) is satisfied. Under these conditions eqn. (2) reduces to

$$\hat{p}_{N_s} = (1-\alpha)A_{N_s} + \alpha(1-\alpha)A_{N_s-1} + \alpha^2(1-\alpha)A_{N_s-2} + \ldots + \alpha^n(1-\alpha)A_{N_s-n} \tag{4}$$

This method of averaging is known as exponential smoothing. If S_t represents the estimate of the average at time t using exponential smoothing then eqn. (4) may be rewritten as

$$\begin{aligned} S_t &= (1-\alpha)A_t + \alpha[(1-\alpha)A_{t-1} + \alpha(1-\alpha)A_{t-2} + \ldots] \\ &= (1-\alpha)A_t + \alpha S_{t-1} \\ &= A_t + \alpha(S_{t-1} - A_t) \end{aligned} \tag{5}$$

Equation (5) states that the new estimate at time t is equal to the observation at time t plus a correction term multiplied by α. The correction term is the difference between the latest observed value of the input, A_t, and the previously estimated value. The influence that the correction term exerts on the value of S_t depends directly on the value of the parameter α. For $\alpha = 1$, the estimate will be unchanged by the new information, and for $\alpha = 0$ the estimate will simply assume the present value of the observed input.

If a signal with a high noise content is to be averaged, the estimate, S_t, must be made insensitive to small random fluctuations by choosing α to be close to one. If on the other hand the signal has a fairly low noise level, a high value of α can be used, and any change in the input signal will be quickly reflected in the averaged input. Under all circumstances a compromise will be involved between the noise rejection and the bandwidth characteristics of the system.

2.2. *Analogue low-pass filtering*

It is informative at this stage to compare the results obtained for exponential smoothing with the performance of a low-pass analogue filter. The voltage across a capacitor and a resistor connected in parallel depends on the rate of charging the capacitor. If the input to the parallel combination is a pulsed stochastic sequence then the voltage developed across the capacitor forms an estimate of the stochastic sequences generating probability. Each pulse in the stochastic sequence charges the capacitor by a fixed amount. The voltage on the capacitor can therefore only vary up to a certain maximum rate, and this rate is governed by the time constant of the circuit and the frequency of the master clock pulse. Due to the inability of the circuit to respond to high frequency fluctuations, it serves to filter out any high frequency variations in the probability.

It may be easily shown that the output voltage across the capacitor is related to the stochastic input sequence by the equation

$$v_t = v_{t-\Delta t} + \left[1 - \exp\left(-\frac{\Delta t}{RC}\right)\right](A_t V_0 - v_{t-\Delta t}) \tag{6}$$

Here V_0 is the voltage corresponding to an ON logic level, and A_t is the value of the logic level (0 or 1) at time t. Δt represents the width of a pulse. Dividing eqn. (6) by V_0 and rearranging gives

$$\hat{p}_t = \left[\exp\left(-\frac{\Delta t}{RC}\right)\right]\hat{p}_{t-1} + \left[1 - \exp\left(-\frac{\Delta t}{RC}\right)\right]A_t$$

where

$$\hat{p}_t = \frac{v_t}{V_0} \quad \text{and} \quad \hat{p}_{t-1} = \frac{v_{t-\Delta t}}{V_0}$$

Thus

$$\hat{p}_t = \alpha\hat{p}_{t-1} + (1-\alpha)A_t \tag{7}$$

where $\alpha = \exp\{-[\Delta t/(RC)]\}$. Equation (7) is identical to eqn. (5) derived for exponential smoothing and this illustrates the fact that exponential smoothing is equivalent to using a low-pass analogue filter to suppress the high-frequency components of a waveform.

The digital synthesis in logic form of exponential smoothing is shown in Fig. 1. This circuit gives an output which is a measure of the generating probability of the stochastic input sequence. The circuit is sometimes called an ADDIE, which is an acronym for Adaptive Digital Element (Gaines 1967 a, Andreae 1968). The operation of the ADDIE is dependent on both the stochastic input sequence and the probability of a feedback sequence obtained from the current state of the up–down counter. If the probability of the

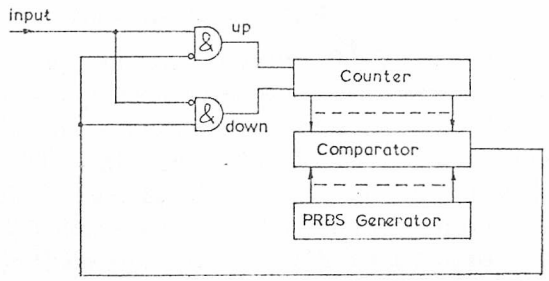


Figure 1. Noise ADDIE.

ADDIE being in state i at time t is $\pi_i(t)$, and the probability of changing from state i to state j at time t is $\pi_{ij}(t)$ then the probability of being in state j at time $t+1$ is

$$\pi_j(t+1) = \sum_{i=1}^{N} \pi_i(t)\pi_{ij}(t), \quad i = 1, 2, \ldots, j, \ldots, N$$

This shows that the operation of the ADDIE can be analysed as a non-stationary Markov process, with $\pi_{ij}(t)$ being a probability matrix with the rows summing to unity. The up–down counter's inability to jump states provides a restriction on the operation of the ADDIE. Thus if the counter is in the initial state i, it can move to states $i-1$ or $i+1$, or it can remain in state i. It follows directly that

$$\pi_{i,\,i-1}(t) + \pi_{i,\,i}(t) + \pi_{i,\,i+1}(t) = 1$$

The inability to the ADDIE to jump states is equivalent to the restriction in the analogue low-pass filter case of the capacitor only being able to change its voltage by a certain fixed amount. It is this constraint which introduces a degree of 'inertia' in the system and provides the filtering action to stochastic sequences.

2.3 *Steady-state analysis of the ADDIE*

The counter of the ADDIE will count up if both inputs are ON and down if both inputs are OFF. When the inputs are different no change of state will occur. Let the probability of the input sequence being ON at any clock interval be p and the probability of it being OFF be q. The ADDIE is a chain-structured automation with $N+1$ ordered states $0, 1, 2, \ldots, n, \ldots, N$. It is shown in Appendix I that if the input signal is a stationary Bernoulli sequence, the states of the ADDIE will randomly fluctuate about a mean value given by

$$M = Np \tag{8}$$

The mean value will form an unbiased estimate of the input sequences generating probability and the distribution function of the ADDIE states will be binomial as given by

$$p_n = p^n q^{N-n} \binom{N}{n} \tag{9}$$

2.4. *Transient response of the ADDIE*

Equations (8) and (9) describe the steady-state characteristics of the ADDIE. These equations may be used to derive an expression for the accuracy of the ADDIE in its measurement of a given stochastic sequences generating probability. The statistics of the input stochastic sequence will not always be constant, and to assess the ADDIE's capability of dealing with non-stationary sequences it is necessary to determine its response to a step input.

Let $n(t)$ be the state of the ADDIE after t steps. It is shown in Appendix II that the expected state of the ADDIE after t steps is given by

$$\varepsilon\{n(t)\} = Np - \{Np - n(0)\}\left(1 - \frac{1}{N}\right)^t \tag{10}$$

Equation (10) indicates that the ADDIE will approach a new level along an exponential path. A time will be reached at which the inherent random fluctuations of the ADDIE states will be greater than the term

$$\{Np - n(0)\}\left(1 - \frac{1}{N}\right)^t$$

At this time meaningful readings may be taken from the ADDIE. Unless t is very large a bias term will exist due to the exponential approach, and this bias term may only be decreased by decreasing the number of ADDIE states N. Although decreasing the number of ADDIE states increases the speed of response of the ADDIE to change, the error due to the random fluctuation of the ADDIE states will also be increased.

The equivalence between the operating characteristics of the ADDIE and a low-pass analogue filter may be illustrated by comparing the equations describing their transient response to a step input.

For the analogue circuit

$$v_t = V_o\left[1 - \exp\left(\frac{-t'}{\tau}\right)\right] \tag{11}$$

where

t' = time in seconds

τ = circuit time constant

$V_o = v(t)$ as $t \to \infty$

For the ADDIE from eqn. (10) with $n(0) = 0$ we have

$$\varepsilon\{n(t)\} = Np\left[1 - \left(1 - \frac{1}{N}\right)^t\right] \tag{12}$$

where t = number of steps $= t' f_c$. Direct comparison of eqn. (11) and (12) gives

$$\tau = -\left\{f_c l_n\left(1 - \frac{1}{N}\right)\right\}^{-1}$$

Equation (12) may thus be rewritten as

$$n(t) = Np\left\{1 - \exp\left[t' f_c l_n\left(1 - \frac{1}{N}\right)\right]\right\} \tag{13}$$

By multiplication of the time-constant term $-\{f_c l_n(1-1/N)\}^{-1}$ the ADDIE's transient response can be equated to that of the analogue filter. However, due to the noise added by the feedback sequence the ADDIE will be less accurate than the analogue filter. A simple calculation given in Appendix III shows that the variance of the analogue filter is given by $(pq)/(2N-1)$. The variance of the ADDIE is obtained directly from eqn. (9) as $(pq)/N$. Thus for the same transient response characteristics the analogue filter is $2-1/N$ times as accurate as the ADDIE structure.

2.5. *Relationship between bandwidth and accuracy*

The frequency limitations of stochastic computing systems may be conveniently stated in terms of the 3 dB point of the ADDIE. The time constant τ of the ADDIE obtained from eqn. (13) is

$$\tau = -\left\{f_c l_n\left(1-\frac{1}{N}\right)\right\}^{-1}$$

Thus the 3 dB frequency is

$$\omega_{3\ \mathrm{dB}} = -f_c l_n\left(1-\frac{1}{N}\right) \tag{14}$$

If the error ϵ of the ADDIE is assumed to be the normalized standard deviation of the state distribution function, then

$$\epsilon = \left(\frac{pq}{N}\right)^{1/2}$$

Clearly the maximum error occurs when $p=q=0{\cdot}5$. Thus

$$\epsilon_{\max} = 0{\cdot}5(N)^{-1/2} \tag{15}$$

The bandwidth may be related to the error by substituting eqn. (14) into eqn. (15) to give

$$\epsilon_{\max} = 0{\cdot}5\left(1-\exp\ -\frac{\omega_{3\ \mathrm{dB}}}{f_c}\right) \tag{16}$$

Figure 2 shows a graph of error ϵ as a function of bandwidth normalized to clock frequency. Clearly the only method to reduce the slope of the graph to provide improved accuracy for a given bandwidth is to increase the value of the clock frequency f_c.

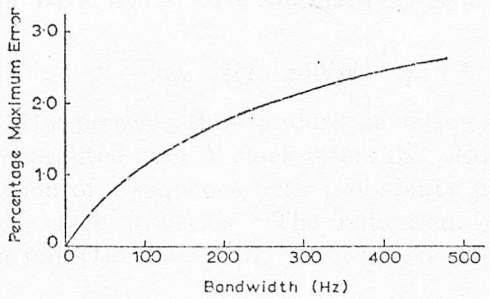


Figure 2. Percentage maximum error as a function of bandwidth. Clock frequency = 1 MHz.

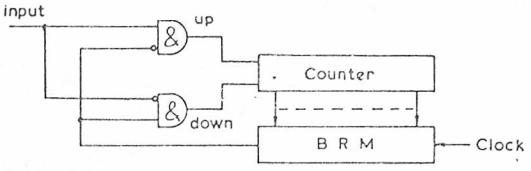


Figure 3. BRM ADDIE.

3. Binary rate multiplier ADDIE

All the analysis so far has considered a noisy ADDIE structure with a feedback signal consisting of a Bernoulli sequence. If alternatively a deterministic feedback signal is used, the ADDIE will still form an estimate of the input sequence's generating probability. The implicit assumption in stochastic computation of statistical independence between signals is still applicable even if one of the signals is deterministic. The deterministic feedback signal may be conveniently generated using a binary rate multiplier (BRM) in the feedback loop as shown in Fig. 3. To differentiate this structure from the noise ADDIE we shall refer to it as a BRM ADDIE. The output frequency, f_b of the BRM is determined by the current state of the up–down counter and the clock frequency f_c. The generating probability of the feedback sequence is f_b/f_c. Due to the lack of random fluctuations in the feedback sequence the distribution function of the ADDIE states will be modified. In fact, as will be shown subsequently, the standard deviation is decreased compared with the noise ADDIE and this provides improved accuracy without reducing the bandwidth of the interface.

The operation of the BRM ADDIE can be classed as a non-Markovian process and hence theoretical analysis is extremely difficult. However it can be shown that a reduction in the standard deviation is to be expected and an expression which describes the extent of the reduction is developed below.

If the noise ADDIE is in a steady-state condition the variance of the distribution function is dependent on the variance of the count-up/count-down signals to the up–down counter. If the probability of an input pulse is p, then the probability of a feedback pulse is $(1-p)$ due to the invertor in the feedback loop. The standard deviation of the count-up sequence, σ_n, for the noise ADDIE is thus

$$\sigma_n = [Np(1-p)[1-p(1-p)]]^{1/2} \tag{17}$$

Similarly for the BRM ADDIE the standard deviation of the count-up sequence σ_b is given by

$$\sigma_b = \{(1-p)Np(1-p)\}^{1/2} \tag{18}$$

Equation (17) represents the standard deviation of a sequence with probability $p(1-p)$ computed over N clock intervals. Equation (18) represents the standard deviation of a sequence with probability p computed over a sample size of $(1-p)N$ clock intervals. The reduction in standard deviation, as expressed by a reduction factor R_u, is obtained directly from eqns. (17) and (18) as

$$R_u = \frac{\sigma_n}{\sigma_b} = 1 + \frac{p^2}{1-p} \tag{19}$$

Thus R_u is maximum when $p=1$ and decreases to unity as p tends to 0. Similarly for the count-down line the reduction ratio R_d, may be calculated as

$$R_d = \frac{\sigma_n}{\sigma_b} = \frac{1-p(1-p)}{p} \tag{20}$$

This reduction ratio has a maximum at $p=0$ and decreases to unity as p tends to unity. The resultant effect of both reduction ratios has been investigated experimentally for a range of input probabilities.

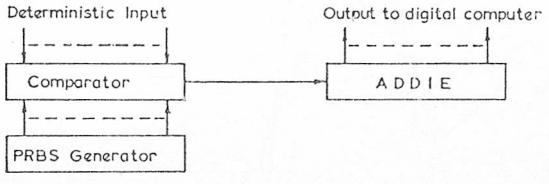


Figure 4. Experimental system for measurement of distribution curves.

4. Experimental results

The experimental work is concerned with the steady-state and transient response characteristics of the two ADDIE structures.

4.1. *Steady-state characteristics*

The basic experimental arrangement is shown in Fig. 4. A deterministic 12-bit digital signal is compared in parallel form with a 12-bit sequence of random pulses obtained from a PRBS generator connected so as to generate a maximum length sequence (m-sequence). The resultant serial output from the comparator is a probabilistic representation of the original deterministic signal. This serial stochastic sequence is then fed to the input of an ADDIE which provides an estimate of the generating probability of the stochastic sequence. Typical state distribution curves for the noise and BRM ADDIE structures for a range of input probabilities are shown in Figs. 5 to 8. These distribution curves were plotted by connecting the output of the ADDIE to a PDP 8E digital computer. A computer programme was devised which interrogated the ADDIE states and stored the results in memory locations directly equivalent to each state of the ADDIE. Each time the ADDIE entered a particular state, the number stored in the equivalent memory location was incremented by one. A digital-to-analogue converter was arranged to provide an output by sweeping across the ordered locations. Due to its chain structure, the ADDIE produces an output sequence which is positively correlated. This correlation tended to obscure the true shape of the distribution function when computed over a short time interval. To avoid this problem the ADDIE was sampled at a low rate compared to the fundamental clock frequency.

The random number generator used for the experiments was a 28-stage shift register with exclusive–OR feedback from stages 3 and 28. Two 12-bit random numbers are required for the experiment ; one for the generation of the input sequence to the ADDIE and one for the generation of ADDIE feedback.

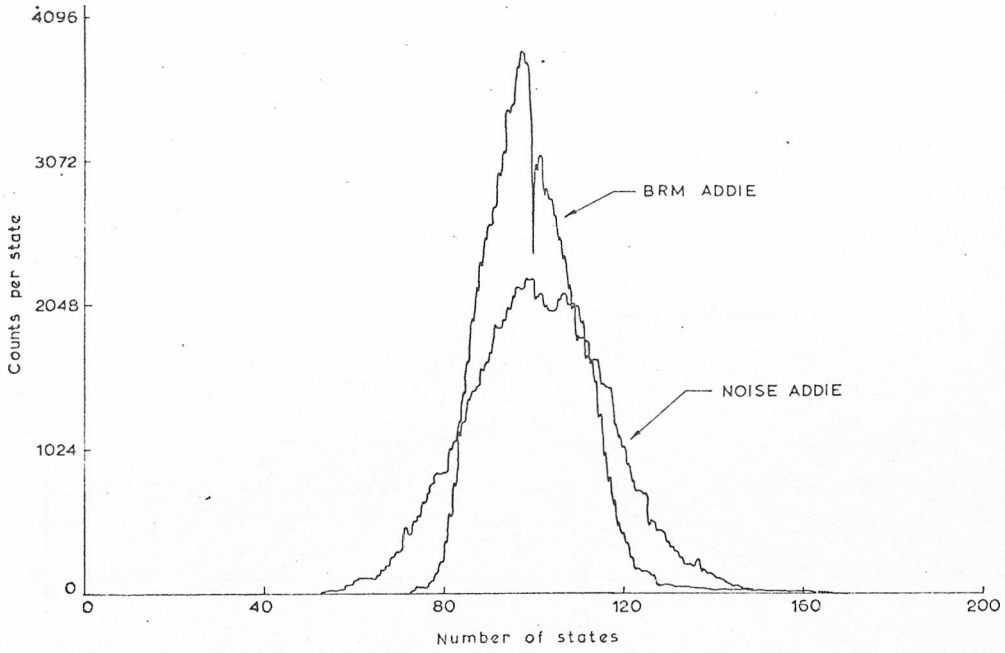


Figure 5. Sampled distribution of states. Input probability = 1/16.

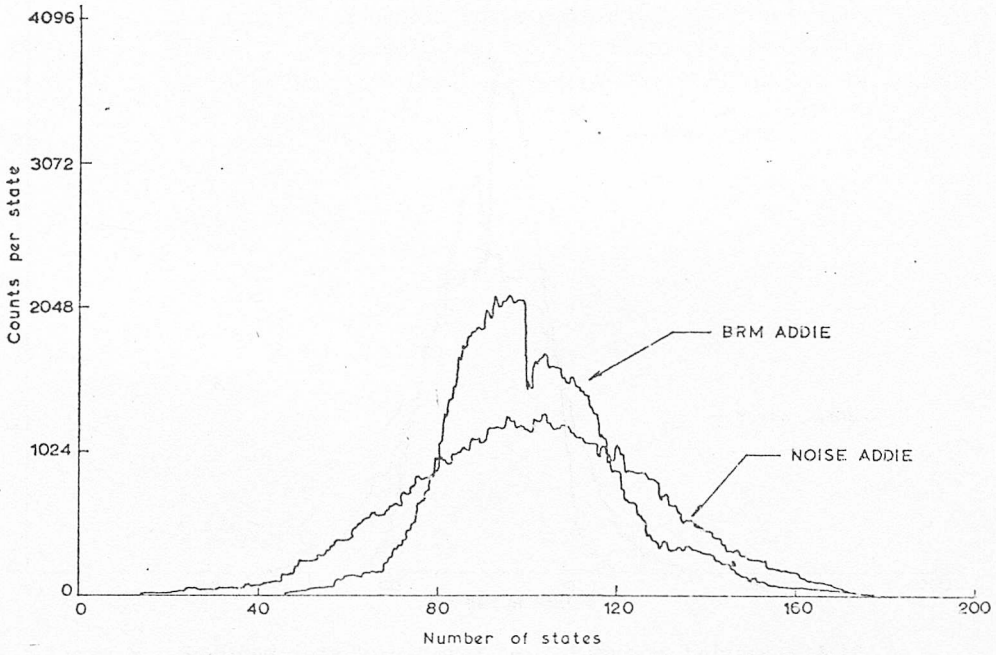


Figure 6. Sampled distribution of states. Input probability = 1/4.

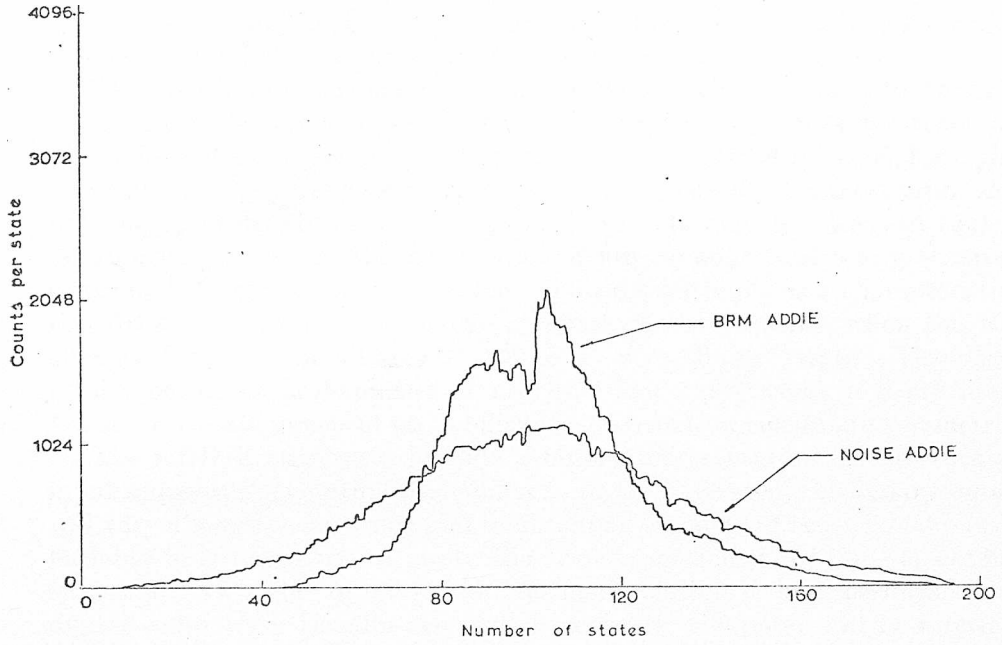


Figure 7. Sampled distribution of states. Input probability = 1/2.

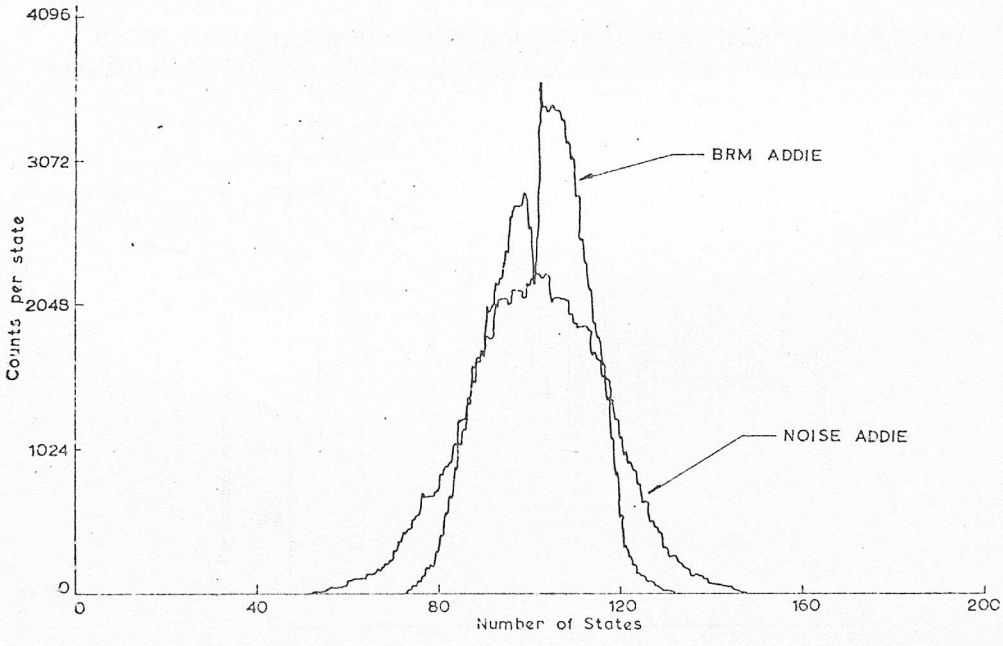


Figure 8. Sampled distribution of states. Input probability = 15/16.

The 24 random bits used were arranged to be 24 consecutive bits of the m-sequence, and the shift register was clocked 32 times between the reading of each random number (Tauseworthe 1965). Thus if the clock frequency of the ADDIE is 100 kHz, the register must be clocked at 3·2 MHz.

The distribution curves for the noise ADDIE shown in Figs. 5 to 8 approximate well to the binomial distribution. This is consistent with the theoretical predictions of § 2.3 (eqn. (9)). As predicted for the BRM ADDIE, for a given input probability a decrease in variance occurs. The effect is most pronounced with input probabilities in the region of 0·5. It may be observed that the distribution curves for the BRM ADDIE are asymmetrical and a sharp discontinuity occurs at the mean value. This discontinuity is a characteristic of the BRM. It is due to a change of phase of the sequence when the BRM changes from a ' one-all-zero ' state to a ' zero-all-one ' state. The discontinuity occurs at probabilities of $1-1/(2^n)$ and $1/(2^n)$ where n is an integer. For intermediate values of probability the distribution curves are symmetrical.

The ADDIE structures provide a binary representation of the stochastic input sequences generating probability. As we have seen, this binary number will vary about a mean value and to obtain an estimate of the original physical variable it is necessary to apply the transformation $E = V(2p-1)$ obtained from eqn. (1). The actual arithmetic manipulation is best performed on a digital computer. Usually the digital stochastic computer will be interfaced to a digital computer and the digital computer may be used to improve the accuracy of the output interface of the stochastic machine by sampling and averaging. If the ADDIE output is sampled N_s times and the average of the samples computed then the accuracy of the output is improved by the factor $1/\sqrt{N_s}$.

Figure 9 shows typical errors for a range of input possibilities for both noise and BRM ADDIES. These results were obtained by sampling the distribution

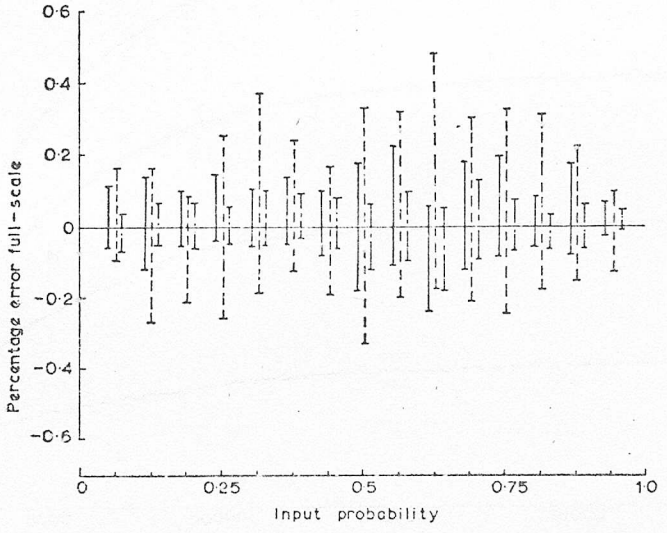


Figure 9. Percentage error full-scale as a function of input probability for two ADDIE structures. —— BRM ADDIE, – – – noise ADDIE, — – — BRM ADDIE (negative correlation). $N = 100$.

of ADDIE states 100 times and evaluating an average using the PDP 8E computer. The estimated value of the physical variable E is expressed as a percentage of the full-scale value V. The sampling technique permits the bandwidth–accuracy relationship to be adapted to a particular problem by initial specification of the number of samples.

The input Bernoulli sequences for the experiment were generated using independent random numbers. It was thought that negatively correlated random numbers might improve the accuracy of the BRM ADDIE since its basic operation is that of addition. This technique is sometimes used for Monte-Carlo simulations. The result of generating a negatively correlated input sequence for the BRM ADDIE is also shown in Fig. 9. The maximum error for the BRM ADDIE is of the order of $\pm 0{\cdot}1\%$, and the improvement in accuracy is clearly evident. The correlated random number was generated by clocking the m-sequence shift register at the same clock frequency as the ADDIE. Negative correlation was achieved by negating all the sequences except the sequence feeding the most significant of the counter.

4.2. *Transient response characteristics*

The responses of both types of ADDIE to a range of step inputs are shown in Fig. 10. These curves were obtained in real time using a 12-bit D/A converter and a graph plotter. It may be observed that the response of the BRM ADDIE is almost identical to that of the noise ADDIE. The curves exhibit a close correspondence to the theoretically predicted curve obtained from eqn. (12). According to eqn. (12) the step response of the ADDIE is related to the

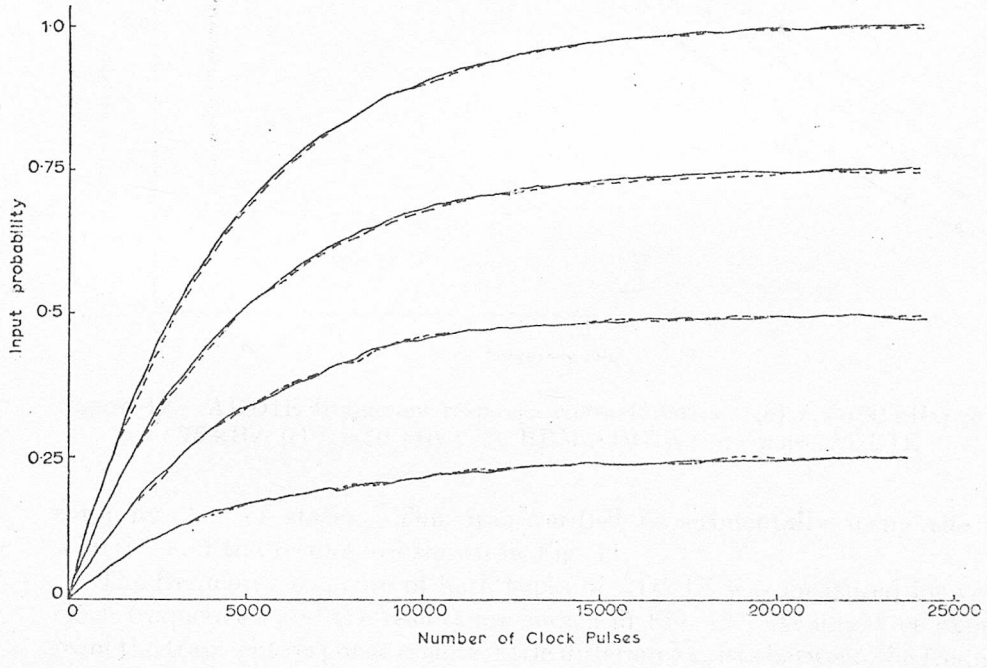


Figure 10. ADDIE step response characteristics. —— BRM ADDIE, – – – noise ADDIE.

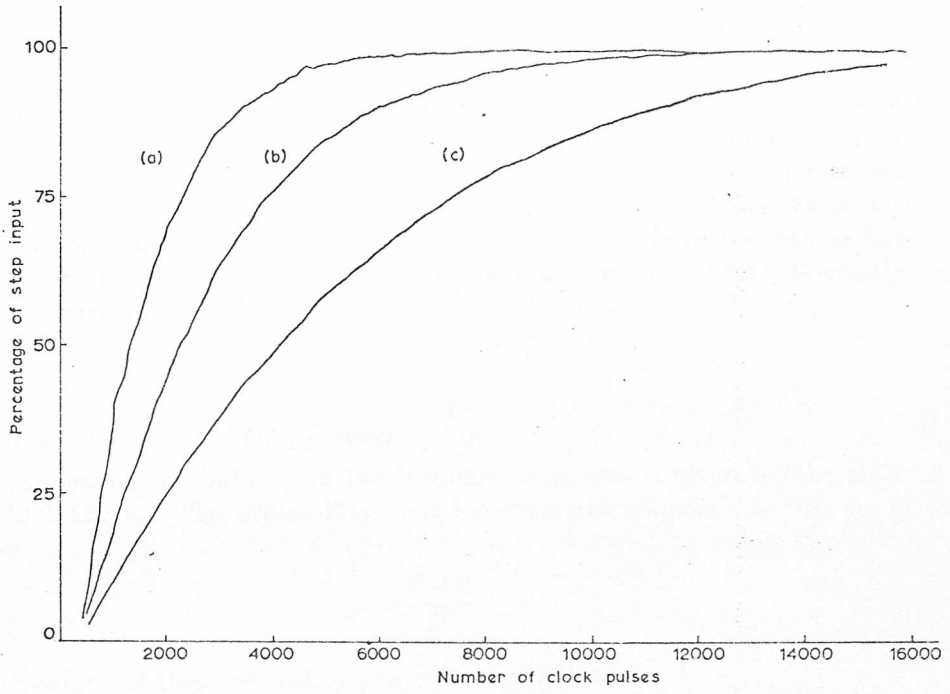


Figure 11. Noise ADDIE step response characteristics as a function of number of states. (*a*) $N=1024$, (*b*) $N=2048$, (*c*) $N=4096$.

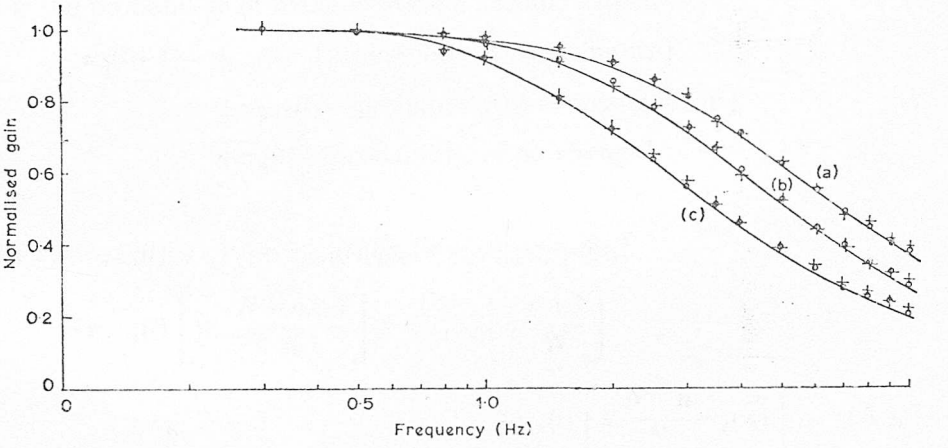


Figure 12. ADDIE frequency response characteristics. (*a*) $f_c=100$ kHz, (*b*) $f_c=75$ kHz, (*c*) $f_c=50$ kHz ; ○, BRM ADDIE ; +, noise ADDIE.

total number of states. This was verified experimentally using the noise ADDIE and the results are shown in Fig. 11.

The frequency response of both types of ADDIE was measured for various clock frequencies and the results are shown in Fig. 12. As might be expected from the transient response results, little difference exists between the frequency response for both ADDIE's and the results show a close correspondence to the theoretical predicted curve obtained from eqn. (14).

5. Conclusions

An experimental and theoretical investigation has been performed of two types of adaptive digital elements suitable for use as an output interface for a digital stochastic computer. Compared with the noise ADDIE the BRM ADDIE has been demonstrated to provide an improvement in accuracy without reducing the bandwidth characteristics. From an economic viewpoint the BRM ADDIE is marginally cheaper than the noise ADDIE using currently available digital integrated circuits. The BRM ADDIE is to be used as the standard output interface in a digital stochastic computer, DISCO, presently under construction.

Appendix 1

Distribution function of ADDIE states

The generating probability of the feedback sequence is given by the state of the ADDIE, n. The probability that the feedback sequence is ON, p_f, is given by

$$p_f = \frac{N-n}{N}$$

The probability of the feedback sequence being OFF is

$$q_f = 1 - p_f = \frac{n}{N}$$

If $\pi_n(t)$ is the probability of being in state n at time t, then

$$\begin{aligned}\pi_n(t+1) = {} & \pi_{n-1}(t) \cdot [\text{probability of counting up}] \\ & + \pi_{n+1}(t) \cdot [\text{probability of counting down}] \\ & + \pi_n(t) \cdot [\text{probability of no change}]\end{aligned}$$

Therefore

$$\begin{aligned}\pi_n(t+1) &= \pi_{n-1}(t)[p \cdot p_f] + \pi_{n+1}(t)[qq_f] + \pi_n(t)[pq_f + qp_f] \\ &= \pi_{n-1}(t)\left[p \cdot \frac{N-n+1}{N}\right] + \pi_{n+1}(t)\left[q\,\frac{n+1}{N}\right] \\ &\quad + \pi_n(t)\left[q\,\frac{N-n}{N} + p\,\frac{n}{N}\right] \qquad \text{(A 1)}\end{aligned}$$

To test that this equation applies in all cases, consider the extreme conditions when the counter is either empty or full up

$$\pi_0(t+1) = \pi_1(t)q\,\frac{1}{N} + \pi_0(t)q$$

Similarly

$$\pi_N(t+1) = \pi_{N-1}(t)p\,\frac{1}{N} + \pi_N(t)p$$

Thus eqn. (A 1) applies for extreme conditions.

Multiplying eqn. (A1) by θ^n and summing from 0 to N gives

$$\sum_0^N \theta^n \pi_n(t+1) = \frac{p}{N} \sum_1^N \theta^n (N-n+1)\pi_{n-1}(t)$$
$$+ \frac{q}{N} \sum_0^{N-1} \theta^n (n+1)\pi_{n+1}(t)$$
$$+ \frac{q}{N} \sum_0^{N-1} \theta^n (N-n)\pi_n(t)$$
$$+ \frac{p}{N} \sum_1^N \theta^n n \pi_n(t)$$

Therefore

$$\sum_0^N \theta^n \pi_n(t+1) = \frac{p}{N} \sum_0^{N-1} \theta^{s+1}(N-s)\pi_s(t) \qquad s = n-1$$
$$+ \frac{q}{N} \sum_1^N \theta^{s-1} s \pi_s(t) \qquad s = n+1$$
$$+ \frac{q}{N} \sum_0^{N-1} (N-n)\theta^n \pi_n(t)$$
$$+ \frac{p}{N} \sum_1^N n\theta^n \pi_n(t)$$

Each summation may now be extended from 0 to N, as each ' missing term ' has a zero coefficient.

In the steady state

$$\pi_n(t+1) = \pi_n(t)$$

Replacing s by n gives

$$\sum_0^N \theta^n \pi_n = \frac{p}{N} \left\{ \sum_0^N \theta^{n+1}(N-n)\pi_n + \sum_0^N n\theta^n \pi_n \right\}$$
$$+ \frac{q}{N} \left\{ \sum_0^N \theta^{n-1} n \pi_n + \sum_0^N (N-n)\theta^n \pi_n \right\}$$
$$= \frac{\theta p}{N} \left\{ \sum_0^N N\theta^n \pi_n - (\theta - 1) \sum_0^N n\theta^{n-1}\pi_n \right\}$$
$$+ \frac{q}{N} \left\{ \sum_0^N N\theta^n \pi_n - (\theta - 1) \sum_0^N n\theta^{n-1}\pi_n \right\}$$
$$= \frac{\theta p + q}{N} \left\{ N \sum_0^N \theta^n \pi_n + (1-\theta) \sum_0^N n\theta^{n-1}\pi_n \right\} \qquad \text{(A 2)}$$

Let

$$G(\theta) = \sum_0^N \theta^n \pi_n$$

Thus . $$G(1)=\sum_{0}^{N}\pi_n=1$$

and $$G'(\theta)=\sum_{0}^{N}n\theta^{n-1}\pi_n$$

Thus directly from eqn. (A 2)

$$G(\theta)=\frac{\theta p+q}{N}\{NG(\theta)+(1-\theta)G'(\theta)\}$$

Therefore $$\frac{G'(\theta)}{G(\theta)}=\frac{Np}{\theta p+q}$$

Since, when $\theta=1$, $G(0)=1$

$$G(\theta)=(p\theta+q)^n$$

Since π_n is the coefficient of θ^n in $G(\theta)$ we have

$$\pi_n=p^nq^{N-n}\left(\frac{N}{n}\right) \qquad \text{(A 3)}$$

The distribution function of states of the ADDIE is thus binomial. It follows that if the input signal is a stationary Bernoulli sequence, then the state of the ADDIE will fluctuate about a mean value given by Np. The mean value will form an unbiased estimate of the input sequences generating probability.

Appendix II

Transient responses of ADDIE

Let $n(t)$ be the state of the ADDIE after t steps. The expected change in state between steps $t-1$ and t is given by

$$E\{n(t)-n(t-1)\}=[\text{probability of counting up}]-[\text{probability of counting down}]$$

$$=p\left\{\frac{N-n(t-1)}{N}\right\}-q\left\{\frac{n(t-1)}{N}\right\}$$

$$=p-\frac{n(t-1)}{N}$$

The average value of $n(t)$ is

$$n(0)+E\{n(1)-n(0)\}+E\{n(2)-n(1)\}+\ \ldots\ +E\{n(t)-n(t-1)\}$$

Thus the expected state of the ADDIE after the first step is

$$E\{n(1)\}=E\{n(0)\}+p-\frac{E\{n(0)\}}{N}=p+\left(1-\frac{1}{N}\right)n(0)$$

$$E\{n(2)\}=E\{n(1)\}+p-\frac{E\{n(1)\}}{N}=p+\left(1-\frac{1}{N}\right)E\{n(1)\}$$

$$=p+\left(1-\frac{1}{N}\right)p+\left(1-\frac{1}{N}\right)^2.n(0)$$

Therefore

$$E\{n(t)\} = p\left[1 + \left(1 - \frac{1}{N}\right) + \ldots + \left(1 - \frac{1}{N}\right)^{t-1}\right] + \left[1 - \frac{1}{N}\right]^{t} . n(0)$$

$$= Np\left\{1 - \left(1 - \frac{1}{N}\right)^{t}\right\} + \left(1 - \frac{1}{N}\right)^{t} n(0)$$

$$= Np - [Np - n(0)]\left(1 - \frac{1}{N}\right)^{t}$$

Appendix III

Variance of analogue filter

Dividing eqn. (5) by V_0 and re-arranging terms gives

$$S_t = \exp\frac{-\Delta t}{RC} S_{t-1} + \left(1 - \exp\frac{-\Delta t}{RC}\right) A_t$$

where

$$S_t = \frac{v_t}{V_0}$$

Replacing $\exp[(-\Delta t)/(RC)]$ by $(1 - 1/N)$ and squaring gives

$$(S_t)^2 = \left(1 - \frac{1}{N}\right)^2 (S_{t-1})^2 + 2\left(1 - \frac{1}{N}\right) A_t S_{t-1} \frac{1}{N} + A_t{}^2 \frac{1}{N^2}$$

By taking expected values and using the relationship

$$\varepsilon(A_t{}^2) = \varepsilon(A_t) = p$$

the above equation becomes

$$\varepsilon(S_t{}^2)\left(2 - \frac{1}{N}\right) = p^2\left(2 - \frac{1}{N}\right) - p^2\frac{1}{N} + \frac{p}{N}$$

Therefore

$$\varepsilon(S_t)^2 - p^2 = \frac{1}{N}(p - p^2)\frac{1}{2 - 1/N}$$

Thus

$$\text{variance} = \frac{p(1-p)}{2N-1} = \frac{pq}{2N-1}$$

ACKNOWLEDGMENTS

The authors wish to thank Professor D. Kerridge of the Department of Statistics, University of Aberdeen, for his interest and for the specific suggestion on the use of negative correlation. One of the authors (A. J. Miller) wishes to acknowledge the support of an S.R.C. studentship.

References

Andreae, J. H., 1968, ' Learning machines—A unified view '. in *Encyclopaedia of Information, Linguistics and Control* (Pergamon Press).

Booth, T. L., 1967, *Sequential Machine and Automata Theory* (John Wiley).

Brown, R. G., 1962, *Smoothing, Forecasting and Prediction of Discrete Time Series* (Prentice-Hall).

Brown, R. G., and Meyer, R. F., 1956, 10th National Meeting of ORSA, San Francisco.

Gaines, B. R., 1967 a, *American Federation of Information Processing Societies, Spring Joint Computer Conference*, **30**, 149 ; 1967 b, *Electronics*, **40**, 72 ; 1969, in *Advances in Information Systems Science*, Vol. 2 (Plenum Press), p. 37.

Mars, P., 1968, *Proc. Instn elect. Engrs*, **115**, 662 ; 1972, UKSC Conference, University of Glasgow, July.

Poppelbaum, W. J., Afuso, C., and Esch, J. W., 1967, *American Federation of Information Processing Societies, Fall Joint Computer Conference*, **31**, 635.

Ribeiro, S. J., 1967, *I.E.E.E. Trans. electronic Comput.*, **16**, 261.

Tausworthe, R. C., 1965, *Maths Comput.*, **19**, 201.