

Computer-Aided Simulation of Relay Logic Circuits
Using Interactive Graphics

Constantine Athanasoulas

A Thesis
in
The Department
of Mechanical
Engineering

Presented in Partial Fulfillment of the Requirements
for the degree of Master of Engineering at
Concordia University
Montréal, Québec, Canada

February 1984

© Constantine Athanasoulas, 1984

ABSTRACT

COMPUTER-AIDED SIMULATION OF RELAY LOGIC CIRCUITS USING INTERACTIVE GRAPHICS

Constantine Athanasoulas

Presently, there is no available, documented, analytical justification for the output state determination techniques used by Programmable Logic Controllers (PLC). Furthermore, no commercially available PLC provides for the simulation of the timing characteristics of CIRCUITS. This Thesis describes two methods for analysing Relay Logic Circuits (CIRCUITS), specified as Ladder Diagrams, in order to determine the output state for the purposes of simulation and programmable logic control. The first method models each device in a CIRCUIT as a resistor. By solving the system of equations describing the model thus obtained, and by establishing the existence, or absence, of a potential difference, across a resistor, which exceeds some threshold value, the output state of a CIRCUIT is determined. The second method determines the output state of a CIRCUIT by modelling it as a "graph" and by systematically "exploring" this graph using a Graph Theoretic method, the CAA. Both methods identify the current-carrying devices in a CIRCUIT, be the paths which include them forward- or backward-directed.

A minicomputer-based CIRCUIT simulator which provides for the interactive, color graphic configuration and

simulation of CIRCUITS and which uses the first of the two methods to determine the output state and to identify the current-carrying devices, as well as to simulate the timing is described. A conceptual design of a microcomputer based PLC, the CPLC, which is for use primarily in teaching the characteristics and operation of PLCs, is also proposed.

ACKNOWLEDGEMENT

The author wishes to convey his gratitude to his supervisors and teachers Dr. R. M. H. Cheng and Dr. S. Lequoc for their interest, guidance, and moral support. He would also like to thank his wife for tirelessly typing the bulk of this manuscript and to Mr. Antonios Georgantas for his care in proof reading it. Thanks are also due to Mr. Vu Ngoc-Lan and Miss Susan Bates for their help in coding the microcomputer version of the CSP. Last, but not least, the author would like to thank his parents and his siblings for their patience and support.

Financial support for the present research was provided by Natural Sciences and Engineering Research Council grants A8662 and A1718.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	xii
LIST OF TABLES	xvii
NOMENCLATURE	xviii

CHAPTER 1

INTRODUCTION

INTRODUCTION	1
--------------------	---

CHAPTER 2

SURVEY OF PROGRAMMABLE LOGIC CONTROLLERS AND REVIEW OF PREVIOUS WORK

2.1 Survey of Programmable Logic Controllers	12
2.1.1 Introduction	12
2.1.2 Part I: General Survey of the State-of-the-Art in PLC Technology	12
2.1.3 Part II: CIRCUIT Configuration and Simulation Facilities	16
2.2 Previous Work	21

CHAPTER 3

MODELLING AND ANALYSIS FOR CIRCUIT SIMULATION

3.1 Introduction	25
3.2 Ladder Diagram Specification of Relay Logic	
Circuits	26
3.2.1 CIRCUIT Devices and Their Ladder Diagram	
Representations	26
3.2.2 Specification of Relay Logic Circuits Using	
Ladder Diagram Symbology	27
3.3 Modelling and Analysis	39
3.3.1 Overview of Simulation Technique	39
3.3.2 Modelling of a Combinational Circuit as a	
Resistive Network	39
3.3.3 Derivation of Model Equations	47
3.3.4 Solution of Model Equations	54
3.3.5 Determination of Internal and Output States ..	58
3.4 Special Case; the Balanced Bridge	62
3.5 Timing Simulation of CIRCUITS	63
3.5.1 Introduction	63
3.5.2 Modelling of the Timing Characteristics of	
CIRCUIT Devices	65
3.5.3 Simulation Clock	67
3.6 Summary	71

CHAPTER 4

MODELLING AND ANALYSIS FOR CIRCUIT SIMULATION
USING GRAPH THEORY

4.1 Introduction	73
4.2 Modelling of a CIRCUIT as a Graph	73
4.3 Analysis of a Graph to Determine the Output State of a CIRCUIT	78
4.4 Identification of Current-Carrying Devices	79
4.5 Description of the DFS	80
4.6 Output Device State Determination and Identification of Current-Carrying devices	91
4.7 Description of the CIRCUIT Analysis Algorithm	86
4.8 Summary	92

CHAPTER 5

DESCRIPTION OF THE IMPLEMENTATION OF THE
CIRCUIT SIMULATION PROGRAM

5.1 Introduction	94
5.2 Development System	95
5.3 Description of the CSP	98
5.3.1 Executive Mode	98
5.3.2 Configuration Mode	100
5.3.3 Simulation Mode	105

5.4 Data Structure	119
5.5 Graphics and Display	122
5.5.1 Introduction	122
5.5.2 Display Element	124
5.6 Implementation of Simulation Program on an 8-bit Microcomputer	128
5.6.1 Memory Usage	131
5.6.2 Coding	131
5.6.3 Graphics	132
5.7 Summary	132

CHAPTER 6

APPLICATION OF CSP TO THE SIMULATION OF TEST CIRCUITS

6.1 Introduction	134
6.2 Case Study 1: Simulation of a Combinational Circuit	134
6.3 Case Study 2: Demonstration of the use of Two Simulation Clocks to Simulate Timing	137
6.4 Case Study 3: Simulation of Race States	148
6.4.1 Use of CSP to Simulate a Simple Transition Path	155
6.4.2 Use of CSP to Simulate Race States	155
6.4.3 Use of CSP to Simulate a Buzzer Circle	161
6.5 Summary	166

CHAPTER 7

CONCEPTUAL DESIGN OF A GENERAL PURPOSE MICROCOMPUTER BASED PROGRAMMABLE LOGIC CONTROLLER

7.1 Introduction	167
7.2 Brief Description of the CPLC	170
7.3 Conceptual Design of the Hardware Configuration of the CPLC	171
7.3.1 Multiplexer	173
7.3.2 Real Time Clock	173
7.3.3 Timer and Delay Relay Duration Counters	174
7.3.4 Input Latch and Buffer	175
7.3.5 Output Latch	175
7.4 Conceptual Design of the CRP	177
7.4.1 Polling Subprogram	177
7.4.2 Modelling and Analysis	177
7.4.3 Set Relay and Counter Contact Pair States ...	177
7.4.4 Setting of Output Device States	179
7.4.5 Executive Subprogram	179
7.5 Setting of Polling Clock Frequency	181
7.6 Summary	183

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions 185

8.2 Future Work 190

REFERENCES 192

APPENDIX A: DESCRIPTION OF A BANDED FORMAT GAUSSIAN

ELIMINATION EQUATION SOLVER 198

APPENDIX B: SHORT GLOSSARY OF THE GRAPH THEORETIC 202

APPENDIX C: STORAGE REQUIREMENTS OF THE CAA 204

APPENDIX D: DATA BASE ATTRIBUTES OF THE CSP 205

LIST OF FIGURES

Fig.1.1	Relay diagram converted to PC-700 or PC-900 Program [4].	5
Fig.3.1	Schematic representations of CIRCUIT devices.	29
Fig.3.2	Schematic representation of a contact pair.	29
Fig.3.3	Series connection of two contact pairs.	31
Fig.3.4	Series connection of n contact pairs.	31
Fig.3.5	Parallel connection of two contact pairs.	33
Fig.3.6	Parallel connection of n contact pairs.	33
Fig.3.7	Series-parallel connection of contact pairs.	34
Fig.3.8	Generalized representation of asynchronous sequential circuits.	36
Fig.3.9	Schematic representation of a transition memory device.	36
Fig.3.10	Typical Ladder Diagram.	38
Fig.3.11	Series connection of more than 7 devices.	40
Fig.3.12	Flowchart of simulation procedure.	40
Fig.3.13	General model of a CIRCUIT.	42
Fig.3.14	Division of a large CIRCUIT into smaller sub-CIRCUITS.	44
Fig.3.15	Typical node n.	50
Fig.3.16	Entries in G_s , \underline{v} and \underline{i} .	50
Fig.3.17	G_s , \underline{v} , and \underline{i} as obtained by inspection from Fig.3.14b.	53
Fig.3.18	Band form storage of G_s .	55
Fig.3.19	Bridge CIRCUIT.	64

Fig.3.20 Inertial Delay Characteristics.	66
Fig.3.21 Pure Delay Characteristics.	66
Fig.4.1 Sample Ladder Diagram.	75
Fig.4.2 Representation showing transmission values of devices.	75
Fig.4.3 Graph of sample Ladder Diagram.	77
Fig.4.4 Node numbering scheme.	77
Fig.4.5 Graph, G.	83
Fig.4.6 Directed spanning tree, G', of G, showing DFN for each vertex.	83
Fig.4.7 Sub-graphs of G'.	85
Fig.4.8 CIRCUIT Analysis Algorithm (CAA).	89
Fig.4.9 Example of tree generated using CAA.	91
Fig.5.1 Configuration of the development system.	96
Fig.5.2 Typical work station.	97
Fig.5.3 Command menu.	99
Fig.5.4 Designation of function keys for Configuration, Mode of CSP.	101
Fig.5.5 Numbered list of devices.	103
Fig.5.6 Sequence of CSP prompts.	106
Fig.5.7 Circuit entered by sequence shown in Fig.5.6. .	107
Fig.5.8 Block diagram of algorithm used to implement configuration facilities.	108
Fig.5.9 Designation of Function keys for Simulation Mode.	109
Fig.5.10 Characteristic of counter device.	112
Fig.5.11 Flowchart for CSP Simulation Mode.	116

Fig.5.12 Data Structure of CSP.	123
Fig.5.13 General contact pair and output device graphic structures.	127
Fig.5.14 Default graphics for output devices.	127
Fig.5.15 Flowchart for graphic element display.	129
Fig.5.16 Sample CSP Simulation display.	130
Fig.6.1 Combinational CIRCUIT.	135
Fig.6.2 Some current path displays for various input states.	138
Fig.6.3 CIRCUIT containing relay, delay relay, timer, and counter.	139
Fig.6.4 Timing chart for test performed on (CIRCUIT shown in Fig.6.3.	140
Fig.6.5 CSP display for CIRCUIT shown in Fig.6.3 at MARK=6.	146
Fig.6.6 Krieger's test CIRCUIT.	149
Fig.6.7 The Y-matrix for Krieger's test CIRCUIT [32]. .	150
Fig.6.8 The Z-matrix for Krieger's test CIRCUIT [32]. .	150
Fig.6.9 A transition path [32].	152
Fig.6.10 A critical rate condition [32].	152
Fig.6.11 A safe race condition [32].	154
Fig.6.12 A "buzzer circle" [32].	154
Fig.6.13 Timing chart for transition from 00/000 for 00 to 10 input transition.	156
Fig.6.14 CSP display used to obtain the column at MARK=3 in Fig.6.13.	157

Fig.6.15	Timing chart for transition from 00/000 for 00 to 01 input transition (DRELY Y2 and DRELY Y3 change simultaneously).	159
Fig.6.16	CSP display used to obtain the column at MARK=3 in Fig.6.15.	160
Fig.6.17	Timing chart for transition from 00/000 for 00 to 01 input transition (DRELY Y3 changes state before DRELY Y2).	162
Fig.6.18	Timing chart for state transition from 00/000 for 00 to 01 input transition (DRELY Y2 changes before DRELY Y3).	162
Fig.6.19	"Forcing" used to set initial, steady state.	163
Fig.6.20	Timing chart for transition from 00/010 for 00 to 10 input transition (DRELY Y1 and DRELY Y3 change simultaneously).	163
Fig.6.21	Timing chart for transition from 00/010 for 00 to 10 input transition (DRELY Y3 changes before DRELY Y3).	164
Fig.6.22	Timing chart for transition from 00/010 for 00 to 10 input transition (DRELY Y1 changes before DRELY Y3).	164
Fig.6.23	Timing chart for simulation of buzzer circle. ..	165
Fig.7.1	Block diagram of typical PLC.	168
Fig.7.2	Block diagram of proposed hardware configuration for CPLC.	172
Fig.7.3	Typical timer or delay relay realization.	176

Fig.7.4	Flowchart of polling subprogram.	178
Fig.7.5	Sequence of operations executed in Automatic Mode.	182
Fig.A.1	Program segment for performing GAUSSIAN elimination on a Matrix stored in Band form [32].	199
Fig.A.2	Active portion of the admittance matrix.	199
Fig.A.3	Modified version of GAUSSIAN Elimination equation solver (GAUSS).	201

LIST OF TABLES

Table 3.1 Input device state assignments.	29
Table 3.2 Transmission across two series-connected contact pairs.	31
Table 3.3 Transmission across two contact pairs connected in parallel.	33
Table 5.1 Failed states of input devices.	121
Table 5.2 Failed states of output devices and memories.	121
Table 5.3 Designation of device names.	121

NOMENCLATURE

A	- Contact pair
A_i	- contact network
B	- bandwidth of G_s
B'	- semi-bandwidth of G_s
C_T	- total number of operations performed by GAUSS
D_C	- duration of delay to be implemented
D_t	- time resolution of simulation
E_g	- maximum number of edges in G
E_i	- event i
G	- graph of a CIRCUIT
G'	- sub-graph of G
G_B	- banded format of G_s
G_{nk}	- admittance of R_{nk}
G_s	- admittance matrix
M	- total number of vertical lines defining the number of columns
M_C	- modulus of counter which implements a timer or delay-relay in the CPLC
MEM_{graph}	- number of bytes required to store working arrays for the implementation of the CAA
MEM_{matrix}	- number of bytes required to store working arrays used by the matrix method
N_B	- number of bytes required to store each variable in G_B
N_T	- total number of rows in Ladder Diagram

NC	- normally closed contact pair
NO	- normally open contact pair
P	- total number of nodes in a model
P_C	- preset value on programmable counter
Q_i	- next internal state
R_i	- resistor connected in first column of i th row
R_{ij}	- resistor connected across nodes i and j
R_T	- total number of resistors in a model
V_{ab}	- actual PD across nodes a and b
V'_{ab}	- numerically calculated PD across nodes a and b
V_g	- maximum number of nodes in G
W	- number of significant digits carried by the computer
c_e	- error constant
e	- maximum numerical error in calculating a nodal voltage as a fraction of
\underline{i}	- Pxl vector of source currents into each node of the model if each node is taken to be at ground potential
q_i	- present internal state
r	- root vertex of the CAA
s	- vertex of a graph
t_{DI}	- inertial delay duration, milliseconds
t_{DP}	- pure delay duration, milliseconds
v	- vertex of a graph
v_a	- potential of node b expressed as a fraction of V
v_b	- potential of node a expressed as a fraction of V

- v_i - potential at node i expressed as a fraction of V .
- \underline{v} - $P \times 1$ vector of voltages (expressed as fractions of V at each node in the model
- x_i - external input excitations
- y_i - external output excitations
- z_{ab} - transmission value across nodes a and b
- z_i - transmission value of the contact network, A_i
- V - source voltage
- Δt - time duration
- Δt_i - time duration between events E_i and E_{i+1}
- Δt_{\min} - minimum time duration between two events
- δ_i - numerical error in calculating v_i

CHAPTER 1

INTRODUCTION

Process or machine control systems, or subsystems, operating exclusively on the basis of binary inputs and outputs and incorporating electromechanical or pneumatic relay devices are often referred to as Relay Logic Circuits (CIRCUITS). CIRCUITS range in complexity from simple safety interlocks of two or three switches, to entire nuclear power plant safety systems comprising hundreds of switches, relays, timers, etc..

In general, CIRCUITS may be asynchronous and sequential. Asynchronous, because the sequence of inputs to a CIRCUIT is governed by a process rather than by a clock. Sequential, because the current states of the outputs are switching functions of the past, as well as the current input states.

Up until the late 1960's, CIRCUITS were implemented almost exclusively by relay or pneumatic logic devices. The introduction of the Programmable Logic Controller (PLC) in 1969, and the phenomenal growth of this technology with the introduction of the microprocessor in 1973, have greatly simplified the processes of specifying, implementing and testing CIRCUITS. There remain however, many applications where only electromechanical relays can be used [1].

Whether a CIRCUIT is to be implemented using relays, a PLC, or a combination of both, the CIRCUIT design process

will involve:

- a) the specification of the switching functions governing the input, output, and internal states.
- b) the configuration of CIRCUIT devices in a manner which realizes the required switching functions.
- c) the implementation and testing of the CIRCUIT to ensure its proper sequential operation.

In order to design a CIRCUIT, the logical relationships between input, output, and internal states must be specified. Such a specification will result from the application of one of several techniques available for the synthesis of sequential circuits [2]. When the number of inputs in the CIRCUIT increases beyond 6 or so however, it becomes difficult to apply these techniques, and the designer's skill and experience are relied upon for the generation of a CIRCUIT specification [3].

Assuming that a CIRCUIT specification can be obtained, the process of designing a CIRCUIT typically involves the description of the interconnections of the CIRCUIT devices using a schematic representation known as

the "Ladder Diagram". In such a schematic representation, the interconnections between devices are defined, and each device is identified.

Interactive CIRCUIT configuration facilities provided by most PLCs greatly reduce the tedium of the traditional "pad and pencil" approach. They allow the designer considerable, though constrained, flexibility in positioning and specifying the devices in a new CIRCUIT, or in editing an existing CIRCUIT, by means of a set of dedicated function keys. Furthermore, the PLC programmer need not undergo lengthy training as most PLCs are programmed using some variation of the familiar Ladder Diagram symbology.

One major drawback of PLC CIRCUIT configuration facilities is that they typically do not allow for the type of documentation one would normally include in a hand drawn Ladder Diagram. The device labelled "PRESS 107" on a manually drawn diagram, for example, would be labelled "I107" on a typical PLC display. Also with regard to documentation, annunciation messages, which indicate the function of a device on the Ladder Diagram, can be accommodated on the display of only a small number of commercially available PLCs.

Another shortcoming of the programming facilities provided by PLCs is that the Ladder Diagram format they use is constrained. Because the CIRCUIT analysis used to determine the output state operates on the information contained in the Ladder Diagram, the format must be kept simple in order

to keep the processing time to a minimum. Most notable of these constraints is the necessity that all current paths be forward-directed.

Consider the CIRCUIT, shown in Fig.1:1, which is used to illustrate the use of the "reference ladder diagram" format for programming the NUMA LOGIC PC-900 and PC-700 series PLCs [4]. Although this configuration can be entered on either of these PLCs, neither is capable of correctly determining the output state when backward-directed current paths are involved [5]. This exemplifies situations where the inadvertent creation of a backward-directed current path would lead to the erroneous determination of the output state. In applications where the safety of personnel or equipment is concerned, such errors cannot be tolerated.

Even if the logic operation of a CIRCUIT is correct, race states in a CIRCUIT may cause it to malfunction, depending on the timing characteristics of the devices in the CIRCUIT. Hence, both the logic and the timing of a CIRCUIT must be thoroughly tested. Only after such testing has been performed can a CIRCUIT design be considered successfully implemented.

One method for testing a CIRCUIT is to assume input and internal states, and to analyse its characteristics by tracing the paths of logical transmission on the Ladder Diagram. This, however, is extremely tedious and unreliable when large CIRCUITS are involved. The alternatives are to either physically implement the CIRCUIT and to test it under.

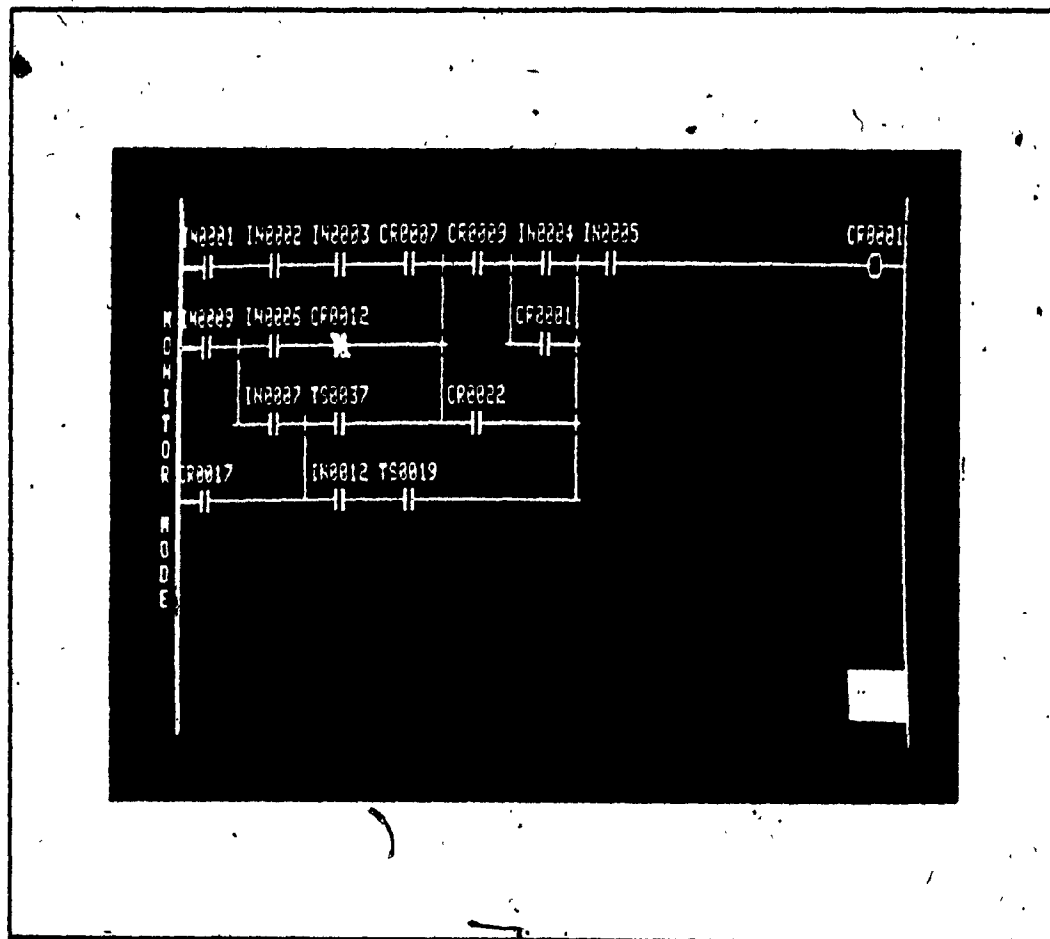


Fig.1.1 Relay diagram converted to PC-700 or PC-900 program [4].

all realistic modes of operation, or to otherwise simulate its operation.

CIRCUITS implemented using electromechanical relays are the most difficult to test since no automatic method for completely simulating their operation exists. Hence, such CIRCUITS must be physically constructed. In the event that a fault is detected, corrections to the wiring as well as to the CIRCUIT drawings and documentation would be required.

Many PLCs provide facilities for the simulation of the CIRCUITS they are used to implement. These facilities are based on the substitution of switches and indicator lamps for process inputs and outputs respectively. With the capability of advancing the operation of a CIRCUIT one "step"* at a time, and of displaying the input, output, and internal states on the graphics screen, it is possible to observe the transient, as well as the steady state response of a CIRCUIT.

There are several difficulties associated with the use of PLC CIRCUIT simulation facilities which are presently available for industrial use. First, the simulator and the PLC are one in the same device and can only operate exclusively of one another. Second, the "single stepping" capabilities, although of some use in simulating the timing

* Where a "step" is defined as the completion of a series of computations to determine the output state of a CIRCUIT for a given input, and present internal state.

characteristics of CIRCUITS, do not easily allow for the simulation of devices with inertial delay characteristics, such as electromechanical relays. Hence, they are not convenient for simulating the timing of CIRCUITS comprising such devices. Third, the simulation facilities of PLCs do not provide on-line documentation of the CIRCUIT being simulated. This makes it difficult for the operator to keep track of the operation of each device in a large CIRCUIT. Furthermore, a much simpler and more effective user interaction can be achieved by using interactive graphics, rather than mechanical switches, to set device states during simulation.

Finally, the most important shortcoming of PLC simulation facilities is that, to the best of this author's knowledge, there is no publicly available, documented analytical justification for any of the methods used by PLCs to determine the state of a CIRCUIT. Without such a basis, the validity of simulations performed using PLC simulation facilities becomes questionable. In applications where the safety of personnel or equipment is involved, such simulations are unacceptable.

It is the purpose of this Thesis to report on two distinct methods for analysing any CIRCUIT which is specified in the form of a Ladder Diagram and which conforms to certain constraints to be described. The first method models the CIRCUIT as a resistive network and analyses this model to determine the output state. The second method

models the CIRCUIT as a graph and explores the graph in a systematic manner in order to determine the output state. Furthermore, both methods are capable of identifying the current-carrying devices in a CIRCUIT so that they may be displayed on a graphics screen in order to show the current paths, be these forward- or backward-directed.

The first of the two methods of analysis described above forms the basis for a CIRCUIT Simulation Program (CSP) which, in addition to ensuring the correct determination of the output state, provides a wide range of CIRCUIT simulation capabilities which equal or surpass those provided by any commercial PLC. These include:

- interactive graphic CIRCUIT configuration
- ability to analyse CIRCUITS containing backward-directed current paths
- color graphic display of current-carrying devices
- color graphic display of output states
- interactive graphic setting of excitations on input devices
- on-line documentation including:
 - user-defined CIRCUIT device identification
 - annunciation message for every device state
 - hard copy output

- capability of tailoring simulation program to meet specific requirements in the areas of
 - file management
 - documentation
 - other functions (process control, motion control).

With these capabilities, the operator can completely specify and test a CIRCUIT interactively on a graphics terminal. There is no need to connect an I/O simulator, nor to consider the constraints imposed by the PLC. The ability to simulate the timing of a CIRCUIT makes it possible to completely simulate the operation of a CIRCUIT, whether it is to be realized on a PLC, or using electromechanical or pneumatic relays.

This Thesis comprises eight chapters. The first chapter introduces Relay Logic Circuits and outlines the process by which these are designed. The advantages and shortcomings of PLC simulation and CIRCUIT configuration facilities are also discussed.

In the second chapter, a survey of PLC technology is presented, along with a review of previous work. Although the use of PLCs provides an alternative method for the implementation of CIRCUITS that invariably makes the design process more productive, it is evident that the design support provided is weak, and that a general purpose

computer with graphics capabilities could provide more powerful configuration, documentation, and simulation facilities.

In the third chapter, a novel method for the analysis of CIRCUITS is described. The method is based on the generation of a resistive network to model the CIRCUIT. The subsequent analysis of the system of equations which describe the model, provides sufficient information to determine the output state, as well as to identify the current-carrying devices. The basis for a simulation of the timing of a CIRCUIT is also developed.

In the fourth chapter, a method for analysing a CIRCUIT by modelling it as a graph, and by exploring this graph in a systematic manner is described. A known method for exploring a graph, the depth-first search, is modified and presented as an algorithm for future implementation. Once implemented, this method would provide for the determination of the output state, as well as for the identification of the current-carrying devices in a CIRCUIT.

The fifth chapter describes the computer program for the interactive, color graphic configuration and simulation of CIRCUITS. The software architecture, data base, and interactive graphic system are described. It is shown how the network analysis and timing analysis methods described in Chapter 3 are used as the bases of the logic and timing simulation programs respectively.

In the sixth chapter, the CSP is applied to several

simple CIRCUITS which have been contrived for the purpose of demonstration. The results of these "experiments" illustrate the capabilities of the program for free format CIRCUIT configuration, logic simulation with display of current carrying devices, and for timing simulation.

A conceptual design for a general purpose, microcomputer-based PLC comprising an 8-bit microcomputer, a real time clock, and an I/O interface is described in Chapter 7. Used in conjunction with the CSP, this system provides almost all of the features of the most powerful industrial PLC, at a fraction of the cost. In addition, it provides access to the system software so that the user may "personalize" the system by adding to, or by modifying the code. The determination of the output states can be performed on the basis of either the analysis method described in Chapter 3, or the novel graph theoretical method described in Chapter 4.

The final chapter summarizes the present work and advances suggestions for future work in this area.

CHAPTER 2

SURVEY OF PROGRAMMABLE LOGIC CONTROLLERS AND REVIEW OF PREVIOUS WORK

2.1 Survey of Programmable Logic Controllers

2.1.1 Introduction

The PLC was born of General Motors' efforts to replace electromechanical relays in its production equipment with more reliable, solid state devices, in the late 1960's [6]. At that time, 3 manufacturers were involved. Today, internationally, there are more than 70 manufacturers of PLC's, each offering a number of products.

The first part of this two-part survey is an overview of the state-of-the-art in PLC technology in general. The second part focuses on the features provided by the PLCs which provide for CIRCUIT configuration and simulation.

2.1.2 Part I: General Survey of the State-of-the-Art in PLC Technology

Several comprehensive surveys of current PLC technology have been conducted by various authors. Morris 1983, [7] surveys 37 manufacturers and tabulates some of the features of 101 different PLCs. In a staff report, "Canadian Controls and Instruments" 1980, [8] presents a

survey of 17 manufacturers of 32 different PLCs. Gele and Mansion 1979, [9] surveys 78 manufacturers of 152 PLCs in all, to provide the most complete such survey available. Although this last survey is several years old, the author has found that most of the PLCs available at that time are still in production, and most of the data presented, still relevant. Taking into account the instances of PLCs included in more than one of these surveys, there are in excess of 160 different PLCs available today with new models being introduced each year.

In related work, Mennie 1979, [10] reviews the memory technology currently being used by 12 PLC manufacturers, and comments on industry trends towards the use of non-volatile semi-conductor memories. Laduzinsky 1983, [11] surveys a number of manufacturers of peripheral equipment which can be used in conjunction with some PLCs to provide special features. Some examples of this type of equipment are: specialized I/O modules; encoders; displays; operator interfaces; clocks; program loaders; and documentation systems. Miller 1983, [12], surveys several new entries in the PLC market which are designed to meet the growing trend towards using small PLCs in distributed networks. Coughlin 1981, [13] surveys the microprocessor and memory architecture used by some major PLC manufacturers.

To supplement the information in the above studies, the author has reviewed literature supplied by the manufacturers of several PLCs in an effort to obtain a

better appreciation of the facilities available. From the compilation and analysis of this large body of information, the following generalizations can be made to give an overview of the state-of-the-art in PLC technology.

2.1.2.1 Central Processing Unit (CPU)

As PLCs become more sophisticated, so do the demands on the CPU being used. Most PLCs presently use 8 bit microprocessors. The PLCs at the "high end" however, i.e. the more powerful PLCs, use 16 bit microprocessors. Gould Inc., uses bit slice technology to achieve high computing efficiency in its MODICON 584 PC. Texas Instruments Inc., uses dual, 16 bit microprocessors, in its PM550 PC in order that both logic and PID control can be performed simultaneously. Other manufacturers are exploring the use of parallel processing in efforts to reduce scan time [14].

2.1.2.2 Scan Rate

The speed with which a PLC can respond to a change in the input state is an important factor in deciding whether that PLC is suitable for a particular application. This speed will depend upon the time required to read the input state and on the time required to determine the output state.

Typically, only the "scan rate" of a PLC is given;

this, in terms of the time required to read 1k words of memory, and includes input and output devices as well as internal memory. The scan rate can vary anywhere from 5 milliseconds to 80 milliseconds, depending on the PLC [9]. These values can, however, be misleading since the time required to determine the output state is not included nor is it available in the technical literature. Hence, the "...scan rate data is, at most, an estimate of typical performance." [15]

2.1.2.3 System Input/Output

PLCs are available with as few as 12 I/Os, to as many as 8192. Most can accommodate several types of I/O, i.e. DC, AC, and analog.

Although many manufacturers provide PLCs which can accompany large numbers of I/Os, several authors [6,7,12,14] have commented that distributed systems are the prevailing trend. That is, rather than using a single PLC with a large number of I/Os, one smaller PLC can be used as a "master" over several other "slave" PLCs, by means of a communication link between them. The advantage of such a configuration is that both the documentation and the control strategy are greatly simplified [16]. Consequently, most PLCs provide for communication links often referred to in the literature as "data highways".

2.1.2.4 PID Capability

Many PLCs at the "high-end" of the spectrum, are capable of handling up to 3 proportional-integral-differential, (PID), control loops. By combining this capability with the simplicity and flexibility of programmable control, these PLCs have the potential of cost effectively replacing industrial control systems comprising pneumatic PID controllers and relay logic.

2.1.3 Part II: CIRCUIT Configuration and Simulation Facilities

2.1.3.1 Introduction

Consider two programmable controllers. One is fully programmable interactively, using Ladder Diagram symbology, and provides for comprehensive CIRCUIT documentation as well as for simulation. The other is programmable using a low level mnemonic language and is neither capable of documentation nor of simulation. Both are PLCs in all respects. The difference between them however, is that the first PLC is capable of assisting the user in specifying and testing a CIRCUIT while the second only implements the necessary control functions. The capabilities of the former type of PLC can be classified under 3 headings: configuration, documentation, and simulation.

2.1.3.2 CIRCUIT Configuration Capabilities of PLCs

Interactive graphic CIRCUIT configuration facilities available with some PLCs, allow for the visualization of the CIRCUIT entry procedure. In general, these facilities consist of the interactive placement and specification of CIRCUIT devices.

User inputs are made through a dedicated keypad on which function keys for the various commands are identified. Command prompting, which guides the user through the CIRCUIT entry procedure, is presently offered by only two PLC manufacturers, Texas Instruments and Allen-Bradley, but does not seem to be gaining popularity. This is mainly due to the demands for memory and computing that the addition of this feature would place on the microprocessor.

PLCs are programmable in a number of different languages, ranging from simple mnemonic codes, to high level graphic languages. The most popular of the latter type are the languages based on the Ladder Diagram symbology. The extensive use of the relay Ladder Diagram format puts the power of these CIRCUIT configuration facilities in the hands of both the technician, as well as the experienced designer.

One major limitation of PLC CIRCUIT configuration facilities is due to the requirement for a simple state determination technique. For a PLC, the time required to perform a state determination will depend upon the

complexity of the CIRCUIT. By restricting the input format, as all commercial PLCs do, to allow the configuration of forward-directed current paths only, this time can be reduced considerably. Hence PLCs require that the operator replace any backward-directed paths in a CIRCUIT with equivalent, forward-directed current paths.

2.1.3.3 Documentation Capabilities of PLCs

The usefulness of a PLC is greatly enhanced if it provides the facilities for producing "documentation" of CIRCUITS implemented on it. Ideally, documentation includes device function descriptions, device names and identifications, annunciation messages, and cross-referencing of coils and of their respective contact pairs. Preferably, this data would be made available in hard copy as well as on the graphics screen accompanying the graphics symbols in a Ladder Diagram.

Although many PLCs claim to provide documentation facilities, most provide nothing more than a hardcopy of a Ladder Diagram. Hence, it can be concluded that PLC designers have sacrificed complete documentation capabilities in order to obtain fast response while using as little memory as possible.

2.1.3.4 Simulation Capabilities of PLCs

Facilities for the simulation of CIRCUITS implemented on PLCs consist basically of arrays of switches and indicator lamps which can be substituted for process inputs and outputs respectively. Some PLCs have the capability to graphically display input, output, and internal states on the same CRT used for CIRCUIT configuration.

All PLCs can be used to simulate a CIRCUIT independently of a machine or process. Some PLCs (Allen-Bradley PLC-2; Westinghouse Numa Logic; SQUARE D CLASS 8881) which use CRT programming units provide for the display of CIRCUIT device states by highlighting the appropriate graphic symbols. Only one PLC, (Westinghouse Numa Logic), has the capability of displaying forward-directed current paths on a CRT screen.

The above capabilities are useful in testing the logic of a CIRCUIT. They do not however, give any indication as to whether the CIRCUIT will function properly under actual operating conditions. Often, race conditions will cause a CIRCUIT to malfunction. In order to detect these, provisions must be made for simulating timing.

Several PLCs provide for single stepping through the control program. This facility allows the user to observe the state of the CIRCUIT after each "scan" of the logic. Although this facility can be used to simulate the timing

characteristics of a CIRCUIT, there are no facilities available for the measurement of the amount of simulated time elapsed between events.

Facilities for monitoring the time duration between specified events in real time as provided by one PLC, (SQUARE D CLASS 8881), is certainly a powerful tool for detecting races in a CIRCUIT once it has been implemented on the process or machine. They are not however, suitable for use in simulation since they require real time inputs.

Although PLCs provide for process- or machine-independent, logic simulation of CIRCUITS, they offer little in the form of diagnostics. To the best of the author's knowledge, no PLC

- provides documented analytical justification for the methods used to determine the output state.
- is capable of displaying both backward- and forward-directed current paths.
- is capable of simulating inertial delays of electromechanical relays, and the timing of CIRCUITS in general.
- provides for the setting of input states by means of the programming console using

interactive graphics.

- can provide on-line documentation during the course of a simulation.
- uses color graphics to highlight simulation data.

2.2 Previous Work

Several authors have indicated that simulation and documentation are important aspects of any CIRCUIT design. Penz 1982, [17] suggests a procedure for the efficient development of PLC realizable CIRCUITS. Documentation and off-line "check-out", are an integral part of this procedure. In a case study of a CIRCUIT design, Baker and Rahoi 1981, [18] explain how CIRCUIT simulation can be used to test and improve designs. According to Gele and Mansion 1975, [9] the use of a PLC as a design tool, can cut CIRCUIT development time to half of that required for a CIRCUIT designed by conventional techniques.

Gardner 1975, [19] finds that the benefits obtained from a simulation are well worth the expense of setting it up. Schalach 1981, [20], however, points out that the advanced features for documentation, simulation and interactive CIRCUIT configuration are not standard on most

PLCs. Hence, the cost of these features may be prohibitive to users with modest applications. Considering the importance of simulation and documentation, it is surprising that so few manufacturers have addressed themselves to the improvement of these facilities, as provided by their respective products.

The trend towards the use of PLCs to implement CIRCUITS raises questions as to which PLC programming language is most effective. Fox 1978, [21] proposes the use of two languages, one using a Ladder Diagram format, and the other using a flowchart format. By applying the language best suited for the execution of a particular portion of the control function, greater programming efficiency would be gained. Struger and Christensen 1982, [22] survey 6 different programming languages, two of which are experimental. Attempts to standardize PLC programming languages are also discussed. The conclusion of this work is that ladder diagrams will continue to be the principal PLC programming language. The conclusions of the PLC survey of the previous section substantiate these findings.

Pluhar 1981, [1] describes the advantages of electromechanical relays over solid state relays in various applications. He predicts that the increased popularity of PLCs will not stop the growth in the use of electromechanical relays. To provide a general CIRCUIT simulation facility, the present Thesis addresses any CIRCUIT specified in the form of a Ladder Diagram,

regardless of the implementation.

With regard to the analysis of CIRCUITS for the purposes of simulation and programmable control, most work on the subject in recent years has been done by PLC manufacturers. After extensive library searches, the author has been unable to locate any published material describing the techniques used by PLCs, to determine output states. The analysis techniques described by Unger [23] are general, and not particularly well suited for computer implementation, as they become cumbersome when the number of state variables increases beyond 5 or 6.

The simulation of CIRCUITS has been the subject of several publications. Cheng 1978, [3] describes a computer program for the simulation of fluid logic CIRCUITS. Although the methods used could be adapted for application to the simulation of CIRCUITS, they would not allow for the determination of paths, nor for the simulation of timing. Kelley, et.al. 1981, [24] describes a simulation program using a simple topological analysis. This simulation however is neither capable of analysing backward-directed current paths, nor the timing of a CIRCUIT. Furthermore, the CIRCUIT input format used is cumbersome and user-interaction is slow.

Kapps 1983, [25] develops a simulator for electronic logic CIRCUITS. The programming language requires that the logic relationships between input and output states be stated explicitly. No provision is made for the simulation

of timing, nor for the identification of the paths of logic transmission. McDermott 1983, [26] proposes an electronic logic simulator for synchronous CIRCUITS. He uses a fixed format CIRCUIT description based on the configuration of the logic gates in the circuit to be simulated. It is a batch mode simulator requiring the complete specification of the timing and logic levels of the inputs, and provides a timing diagram of the outputs. It does not however provide for the display of the paths of logic transmission through the CIRCUIT, nor can it accomodate the simulation of inertial delays, i.e. relays, timers etc.

In a recent publication, Cheng, et.al. 1983, [27] describes a computer program for the simulation of CIRCUITS based on one of the analysis methods described in this Thesis.

CHAPTER 3

MODELLING AND ANALYSIS FOR CIRCUIT SIMULATION

3.1 Introduction

A computer program for simulating the logic and timing characteristics of CIRCUITS has been developed. It employs interactive computer graphics to allow the operator to specify a CIRCUIT, as well as to allow him to change the excitation on the input devices and observe the response to these changes on a graphics screen.

In order to simulate a CIRCUIT, successive analyses must provide the following:

- capability to analyse logic.
- capability to analyse timing.
- results amenable to graphic display.
- correct determination of output state for any orientation of current paths, be they forward- or backward-directed.
- identification of current-carrying devices for use in diagnosing faults in a CIRCUIT.

The methods for analysing the logic and the timing characteristics of CIRCUITS, which are described in this chapter, provide all the above features and form the basis of the CIRCUIT Simulation Program (CSP).

3.2 Ladder Diagram Specification of Relay Logic Circuits

3.2.1 CIRCUIT Devices and Their Ladder Diagram Representations

Three classes of CIRCUIT devices are represented in a Ladder Diagram; input, output, and memory. Input devices can be classified as external input devices, or as connectors. External input devices are contact pairs to which an excitation is applied from the process or machine being controlled. Examples of such devices are pushbuttons, and various other types of switches, (i.e., limit, level, pressure, temperature, and hand switches, etc.). There are two types of external input devices: normally open, (NO); and normally closed, (NC). In their quiescent states, NO type inputs exhibit an open circuit, and NC type inputs, a closed circuit, across their respective terminals. When an excitation is applied, NO inputs, become closed, and NC inputs become open. Table 3.1 summarises these state assignments. Finally, connectors, both vertical and horizontal, are input devices having a constant, closed

state across their terminals.

Output devices apply excitations from the CIRCUIT, onto the machine or process being controlled. Devices of this type are lamps, motors, solenoids, valves, buzzers, etc..

Memory devices include relays, timers, delay relays, and counters. Typically, they comprise two components: a coil, and a set of contact pairs which may be of the NO or of the NC type. The coil of a memory device provides an excitation on the contact pairs associated with it.

The schematic representations of the various CIRCUIT devices are shown in Fig.3.1. In addition to these, some method for identifying each device in the CIRCUIT must be established. One such method is to let each device in the CIRCUIT be identified by the excitation associated with it. Hence, multiple contact pairs of a single, physical input device, as are often encountered in CIRCUITS, can be easily dealt with as every contact pair belonging to a physical device will be identified by a single excitation.

3.2.2 Specification of Relay Logic Circuits Using Ladder Diagram Symbology

The method of specifying a switching function using Ladder Diagram symbology can be explained using the concept of "transmission". The capability of the Ladder Diagram to represent switching functions, as well as memory, makes it

possible to use this symbology to specify CIRCUITS.

Consider the schematic representation of a contact pair, A, as shown in Fig.3.2. Assume that a potential difference, v_a is applied between terminal "a" and ground, and let v_b denote the potential difference between node "b" and ground. Also assume that when there is a closed CIRCUIT across the terminals of A, the resistance across them is negligible. Then, the states of A can be expressed in terms of v_a and v_b as follows:

$$\begin{aligned} \text{if } v_b &= 0, & A \text{ is open} & & (3.1) \\ \text{if } v_b &= v_a, & A \text{ is closed} & \end{aligned}$$

where "open" and "closed" refer to the state assignment summarized in Table 3.1. From this relation, the switching variable z_{ab} denoting transmission is defined as:

$$z_{ab} = v_b / v_a \quad (3.2)$$

such that,

$$\begin{aligned} z_{ab} &= 0 & \text{if } A \text{ is open} & & (3.3) \\ z_{ab} &= 1 & \text{if } A \text{ is closed} & \end{aligned}$$

where, z_{ab} describes the state of electrical transmission across terminals a and b and applies to any contact pair.

The concept of transmission can be extended to apply

INPUT DEVICE TYPE	EXCITATION	DEVICE STATE
NO	QUIESCENT	OPEN
NO	EXCITED	CLOSED
NC	QUIESCENT	CLOSED
NC	EXCITED	OPEN

Table 3.1 Input device state assignments.





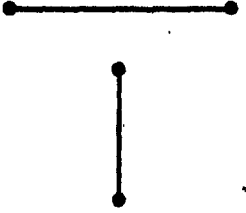
INPUT DEVICE REPRESENTATIONS	MEMORY AND OUTPUT DEVICES	CONNECTORS
 NO  NC	 MEMORY  OUTPUT	

Fig.3.1 Schematic representations of CIRCUIT devices.

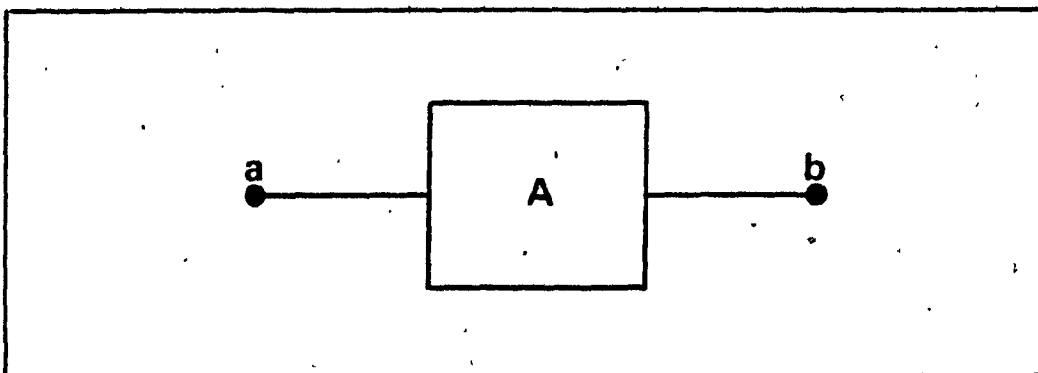


Fig.3.2 Schematic representation of a contact pair.

to contact networks. Consider first the series connection of two contact networks A_1 and A_2 , as shown in Fig.3.3. The tabulation of the possible transmission values for each device and the corresponding transmission value for their series connection is given in Table 3.2. From this table, the switching function of the transmission value across terminals a and c can be written:

$$z_{ac} = z_{ab} \cdot z_{bc} \quad (3.4)$$

where " \cdot " denotes logical multiplication. It can be shown that for n contact pairs connected in series across terminals a and b, as shown in Fig.3.4, that:

$$z_{ab} = z_1 \cdot z_2 \cdot \dots \cdot z_n = \prod_{i=1}^n z_i \quad (3.5)$$

where z_i represents the transmission value across the terminals of the i th input device, and " \prod " denotes the logical product.

Next, consider the parallel connection of two contact pairs A_1 and A_2 as shown in Fig.3.5, with individual transmission values z_1 and z_2 across their respective terminals. The transmission value for the parallel connection of these devices is tabulated in Table 3.3, along with z_1 and z_2 . From Table 3.3, the transmission value across the terminals a, and b can be written:

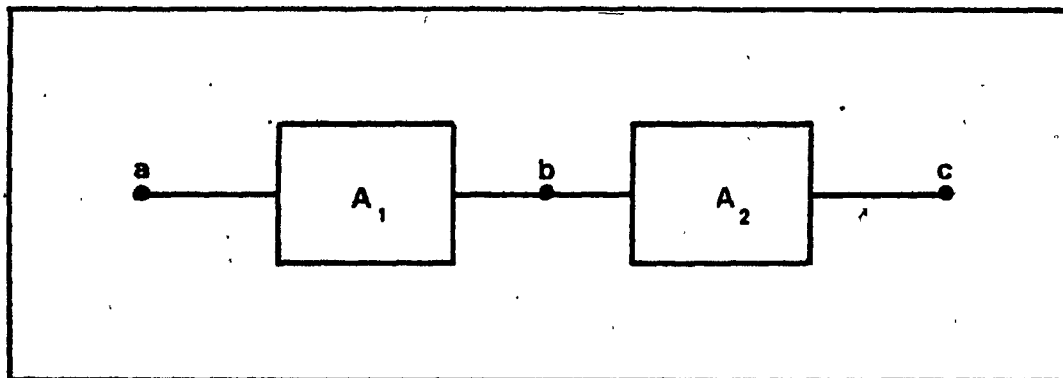


Fig.3.3 Series connection of two contact pairs.

z_{ab}	z_{ab}	z_{ac}
0	0	0
0	1	0
1	0	0
1	1	1

Table 3.2 Transmission across two series-connected contact pairs.

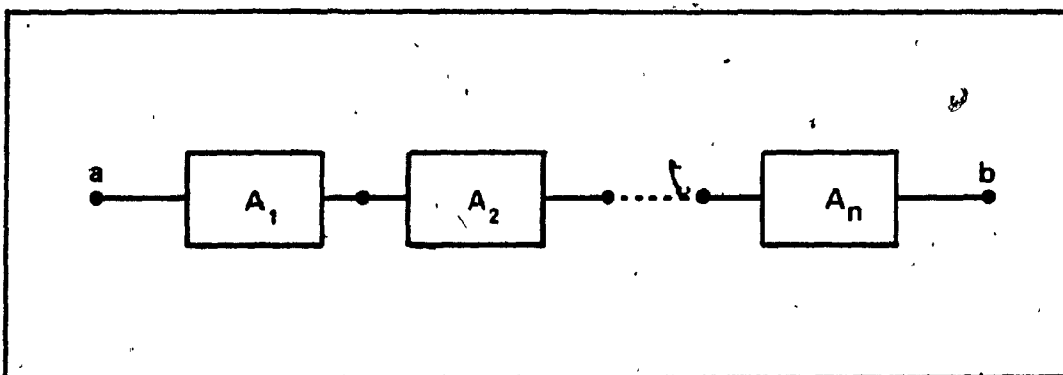


Fig.3.4 Series connection of n contact pairs.

$$z_{ac} = z_1 + z_2 \quad (3.6)$$

where "+" denotes logical summation. It can be shown that for n contact pairs connected in parallel across the terminals a and b, as illustrated in Fig.3.6, the transmission value across these terminals is given by

$$z_{ab} = z_1 + z_2 + \dots + z_n = \sum_{i=1}^n z_i \quad (3.7)$$

where each z_i is the transmission value across the terminals of the i^{th} contact pair and " \sum " denotes logical summation.

Now, consider the series, parallel connection of contact pairs shown in Fig.3.7. The transmission value across terminals a and c can be determined by applying (3.4) to obtain the expression.

$$z_{ac} = z_{ab} \cdot z_3 \quad (3.8)$$

From (3.6) however,

$$z_{ab} = z_1 + z_2 \quad (3.9)$$

Combining (3.8) and (3.9),

$$z_{ac} = (z_1 + z_2) \cdot z_3. \quad (3.10)$$

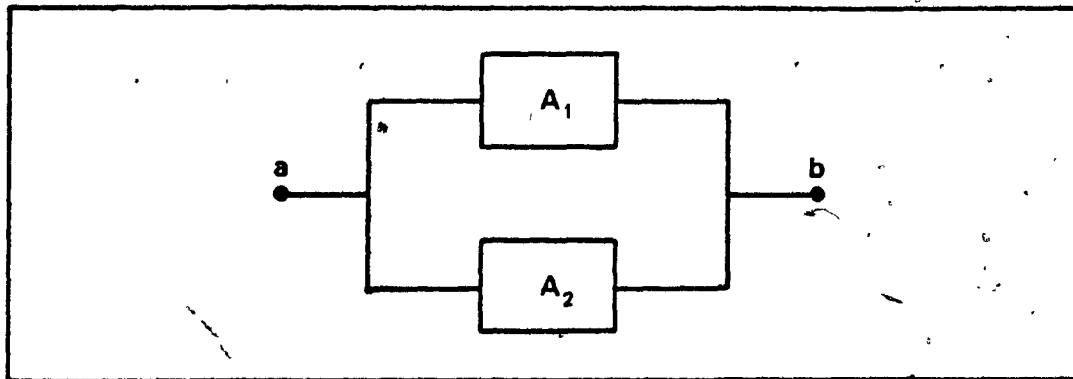


Fig.3.5 Parallel connection of two contact pairs.

z_1	z_2	z_{ab}
0	0	0
0	1	1
1	0	1
1	1	1

Table 3.3 Transmission across two contact pairs connected in parallel.

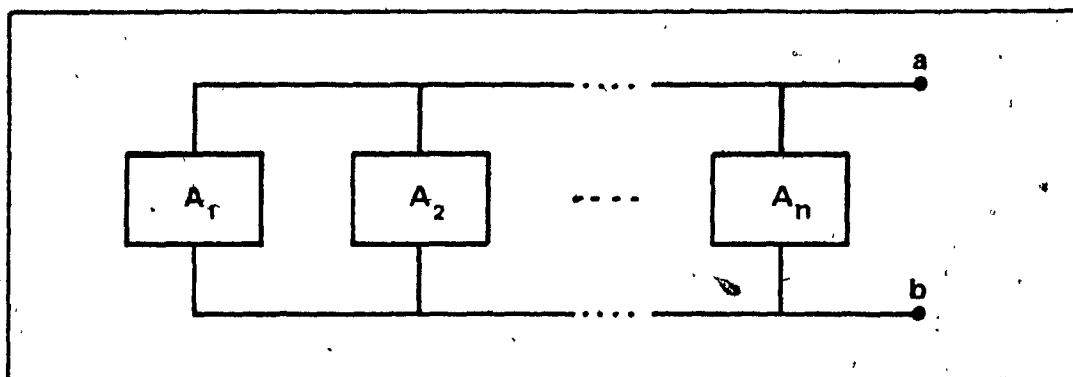


Fig.3.6 Parallel connection of n contact pairs.

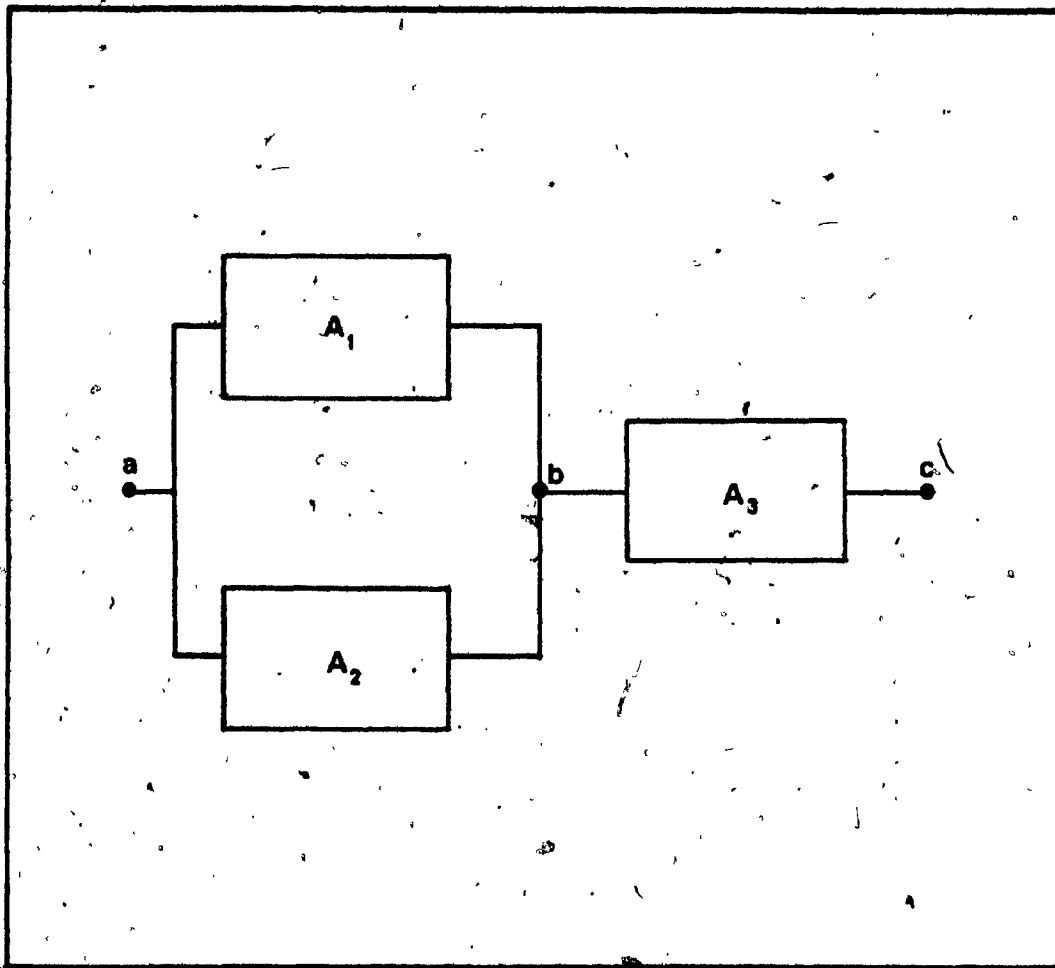


Fig.3.7 Series-parallel connection of contact pairs.

The combination of input devices in a series, parallel fashion, yields a functionally complete set of operators with which any logical function can be realized [28]. It remains to be shown, however, how a CIRCUIT, which is typically sequential, can be specified, by Ladder Diagram symbology.

For any CIRCUIT, the output state will depend upon the input, as well as the internal states. Hence, it can be represented as a combinational circuit with a memory [29], as shown in Fig.3.8, where:

x_1, x_2, \dots, x_p represent the p external input excitations which define the input state,

y_1, y_2, \dots, y_m represent the m external output excitations which define the output state,

q_1, q_2, \dots, q_k represent the k , internal excitations defining the present internal state, and

Q_1, Q_2, \dots, Q_k represent the k , internal excitations defining the next internal state.

A transition memory device, such as the one illustrated schematically in Fig.3.9, possesses the characteristic that for a given input value, Q_i , the output, q_i , will take on the value of Q_i , after some time Δt .

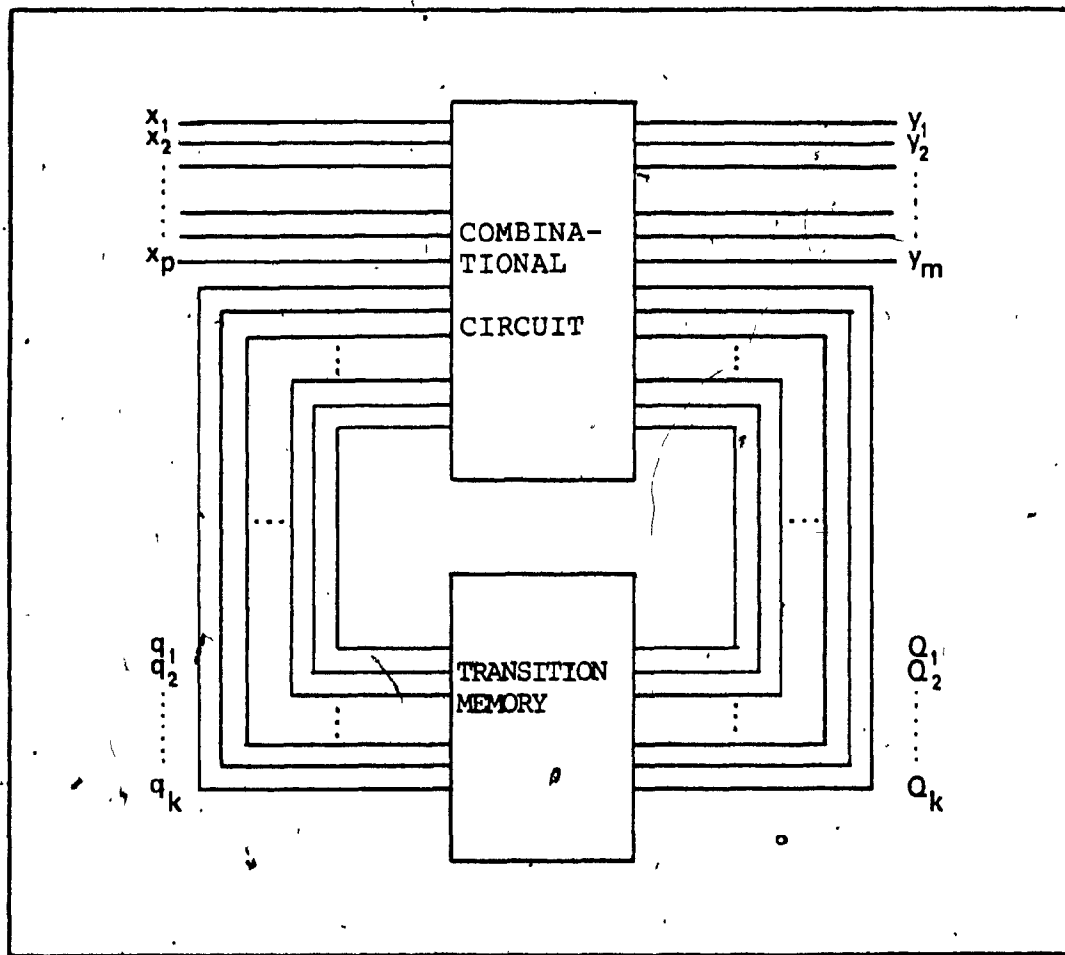


Fig.3.8 Generalized representation of asynchronous sequential circuits.

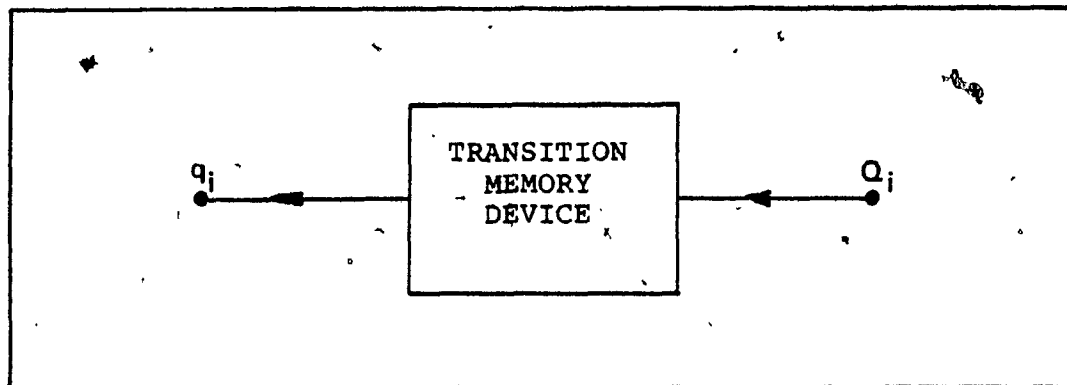


Fig.3.9 Schematic representation of a transition memory device.

Hence, any device capable of logical transmission can be used as a memory since all physical devices exhibit a finite time delay. Typically, relays, timers, and delay relays, are used as memory devices in CIRCUITS. Counters are also used as memory devices. Counters however, are independent of time and require that Q_i undergo a preset number of off-to-on transitions before $q_i = Q_i$.

Figure 3.10 shows a typical Ladder Diagram. There are N_T rows of $(M+1)$ columns of CIRCUIT devices. Only vertical connectors can be used to connect one row to another. Input devices can only be placed in the first M columns, while output devices can only occupy positions in the $(M+1)^{th}$ column. Each CIRCUIT device, other than a connector, is identified by a name and a number.

The vertical lines at the extreme east and west of the Ladder Diagram represent the negative and positive terminals of the voltage source of the CIRCUIT. A typical CIRCUIT will have 5 to 30 rows, and 5 to 6 columns. CIRCUITS of more than 100 rows and 10 columns are also possible. The CSP can accommodate CIRCUITS of up to 100 rows and 7 columns. The capability of the CSP to analyse backward-directed current paths however, makes it possible to extend the number of columns by connecting one row to the next, as shown in Fig.3.11.

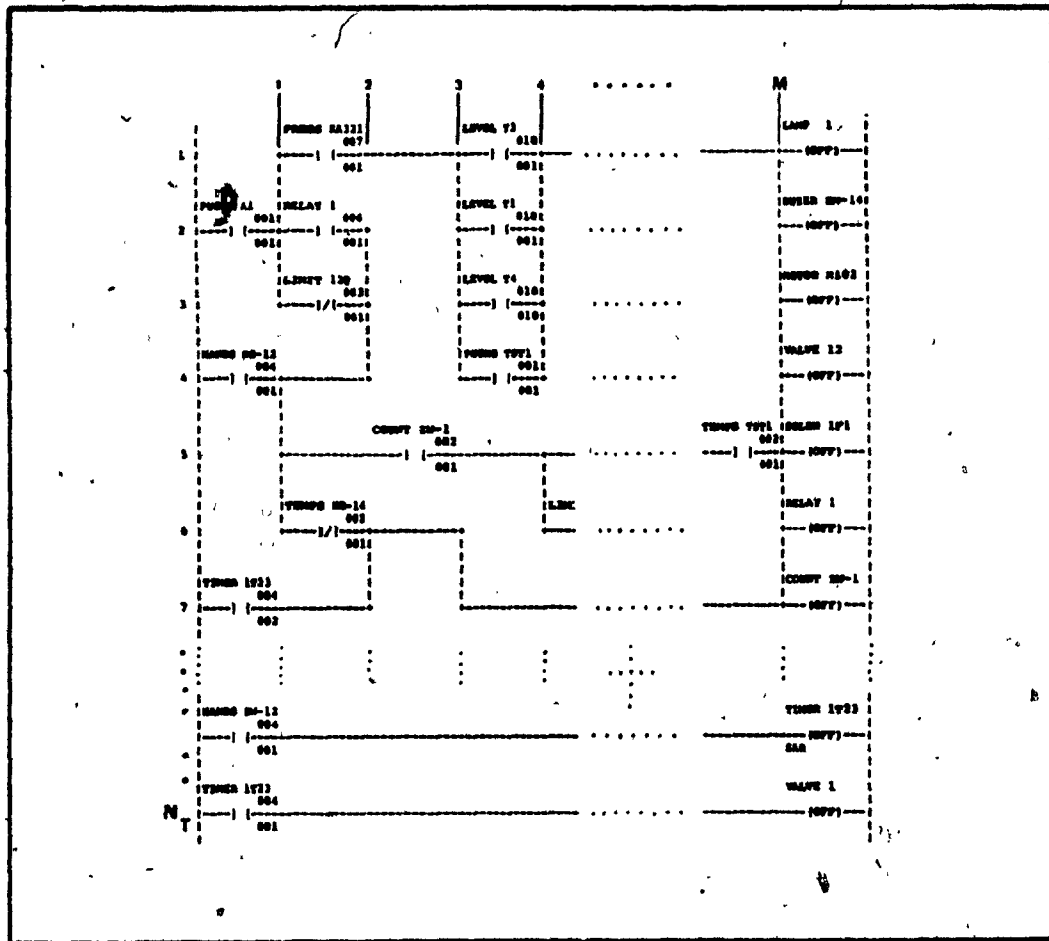


Fig.3.10 Typical Ladder Diagram.

3.3. Modelling and Analysis

3.3.1 Overview of Simulation Technique

The method for simulating CIRCUITS models the combinational CIRCUIT for a given set of internal and external input states as a resistive network. A description of the simulation procedure is given in the form of a flowchart in Fig.3.12.

In this section, the method for modelling and analysing the CIRCUIT to determine the output state and to identify the current-carrying devices in order to display the current paths will be described. The implementation of this method is described in Chapter 5.

It is important to note that only the logic and timing characteristics of CIRCUITS are to be simulated. No inferences are made about the electrical characteristics of the CIRCUITS to be simulated.

3.3.2 Modelling of a Combinational Circuit as a Resistive Network

Any combinational CIRCUIT specified in the form of a Ladder Diagram can be modelled as an electrical network, comprising only resistors and a voltage source. Such a network will be referred to as a "model", and its

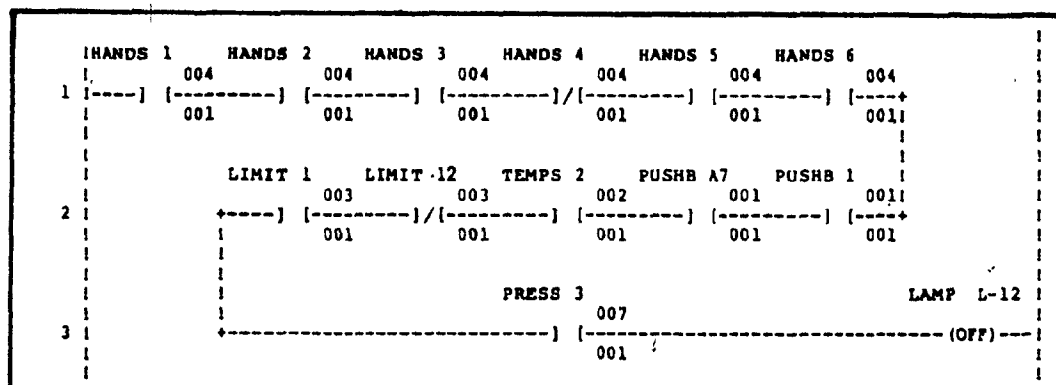


Fig.3.11 Series connection of more than 7 devices.

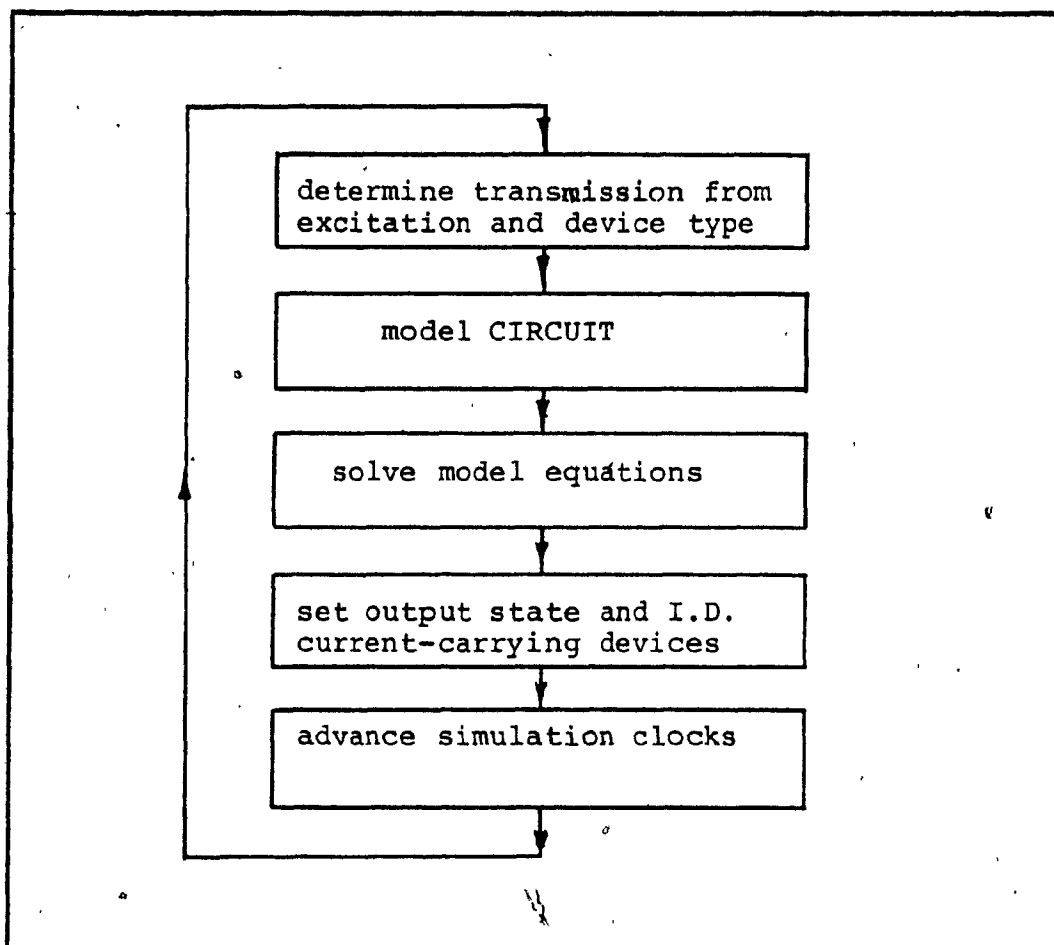


Fig.3.12 Flowchart of simulation procedure.

constituents, other than the voltage source, as resistors.

A model can be constructed directly from a Ladder Diagram on the basis of a one-to-one correspondence between device positions in the Ladder Diagram, and resistor positions in the model. For the purposes of simulation, input devices with current transmission values of 0 and 1, are modelled as infinite-valued, and unit-valued resistors, respectively. Voids (blank positions), in the Ladder Diagram, are modelled as infinite-valued resistors, and output devices are modelled as unit-valued resistors.

In the general model shown in Fig.3.13, there are N rows of resistors arranged in M+1 columns. In addition, there are Mx(N-1) resistors for modelling vertical connectors, giving a total of

$$R_T = N + 2MN - M \quad (3.11)$$

resistors in a model. For practical purposes, there should be at least 2 columns and 2 rows in a model, ie., $M > 2$, and $N > 2$.

In order to reduce the amount of memory required to store the model equations, the CIRCUIT is analysed in a piecemeal fashion. The CIRCUIT can be looked upon as comprising numerous sub-CIRCUITS, where each sub-CIRCUIT is a set of consecutive rows, connected to one another. A sub-CIRCUIT may consist of up to N rows. For practical purposes, N is assigned a value of 7. As a result,

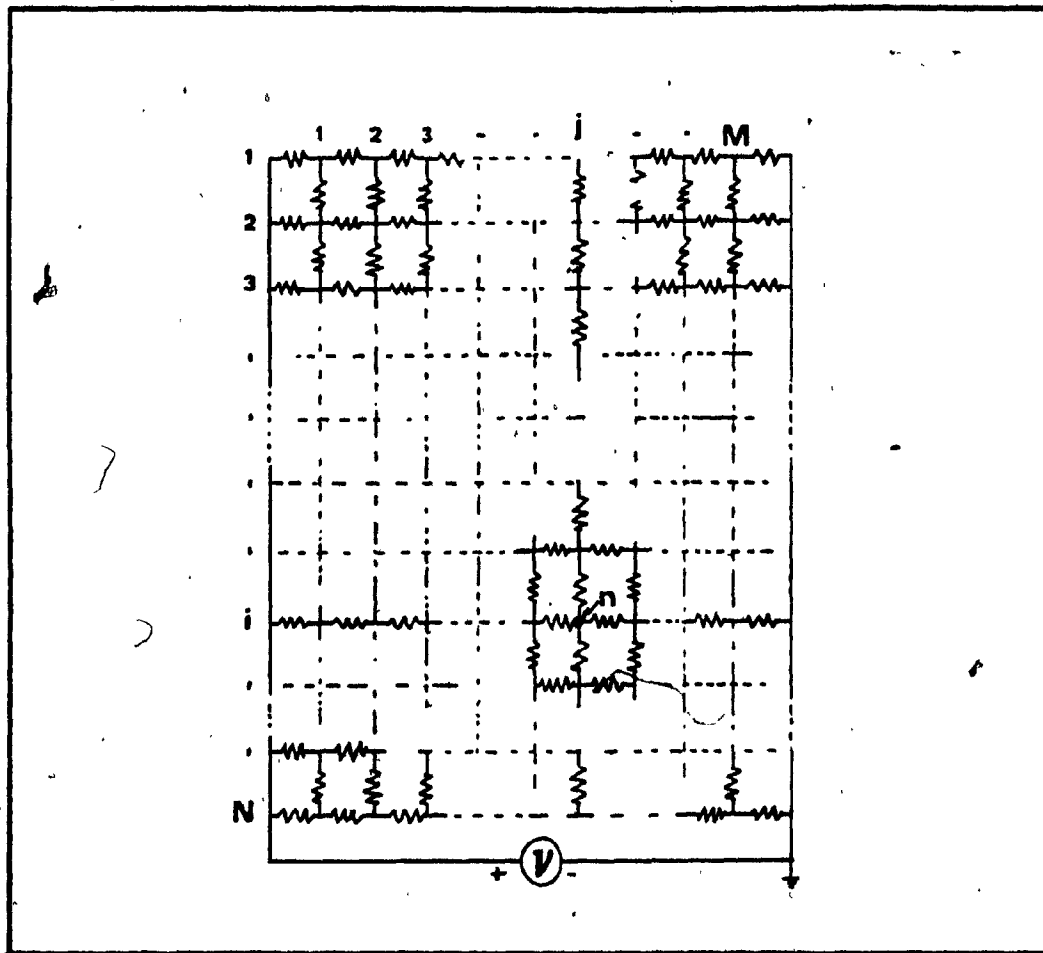


Fig.3.13 General model of a CIRCUIT.

CIRCUITS with more than 7 consecutive, interconnected rows cannot be simulated without some modification.

Consider the CIRCUIT shown in Fig.3.14a. By introducing an additional contact pair, the CIRCUIT can be divided into manageable sub-CIRCUITS as shown in Fig.3.14b. The simulation of CIRCUITS which cannot be broken into sub-CIRCUITS with $N \leq 7$ is beyond the limitations of an executable CSP program. By allocating additional memory and by increasing N , such CIRCUITS can be accommodated. This however, would require that the CSP be recompiled.

A node is defined at each junction of two or more resistors. In all, there are

$$P = M \times N \quad (3.12)$$

nodes, which are numbered sequentially from left to right, and top to bottom of the model, and the node index, n , is given by

$$n = M(i-1) + j \quad (3.13)$$

where i and j are the row and column indices of the model, respectively.

Resistors in the model are identified by the symbol

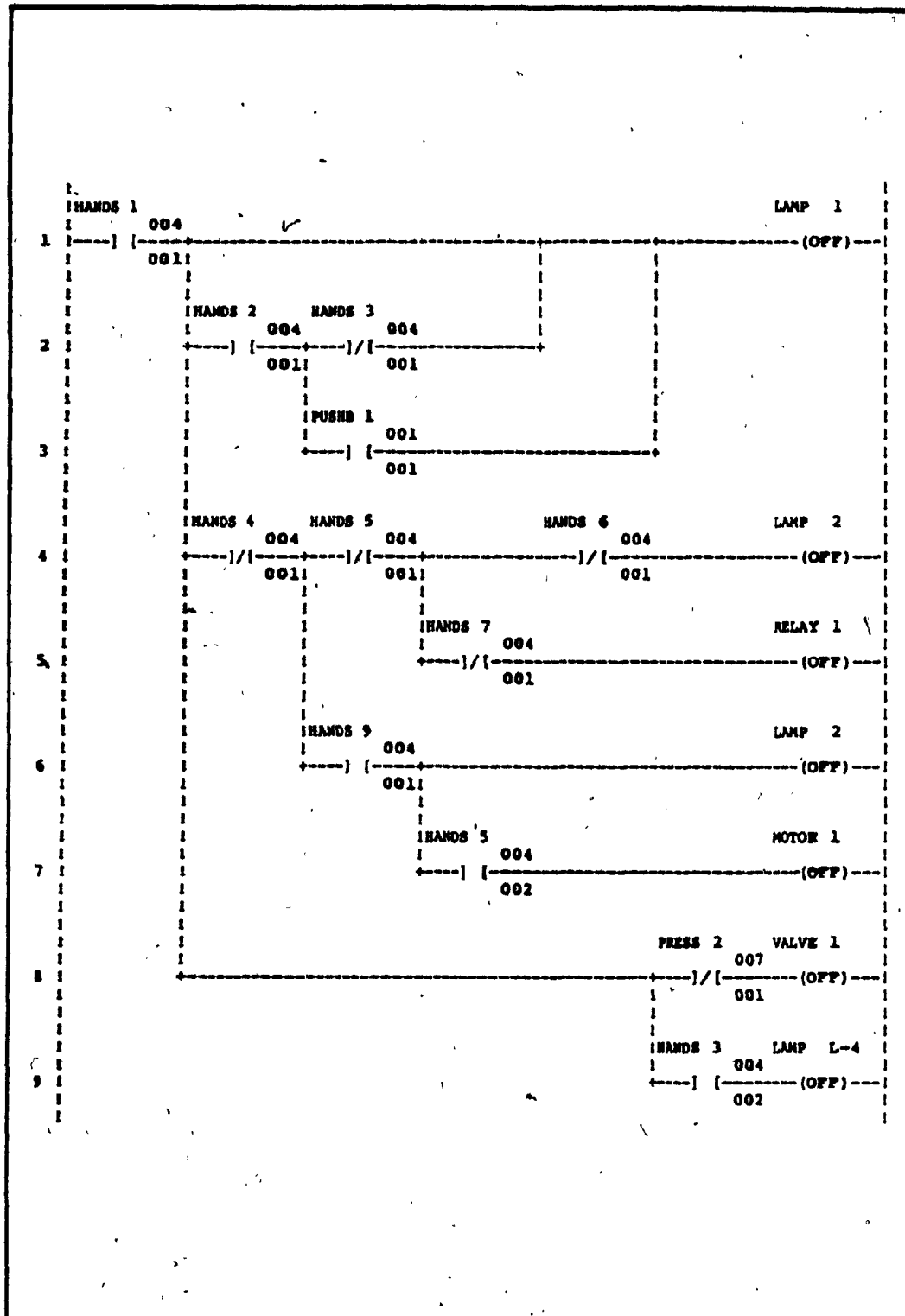


Fig. 3.14a Division of a large CIRCUIT into smaller sub-CIRCUITS.

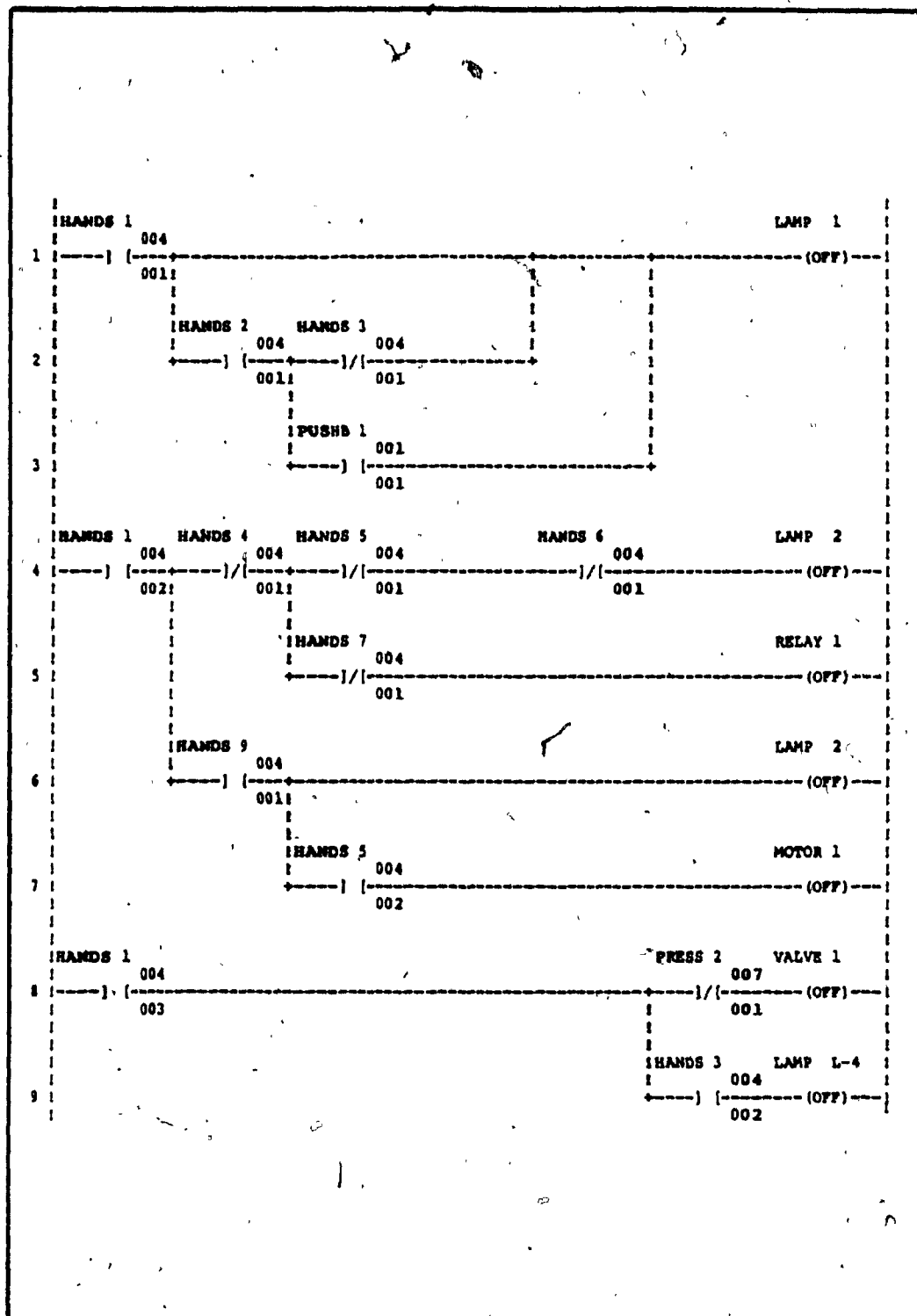


Fig. 3.14b Division of a large CIRCUIT into smaller sub-CIRCUITS.

R_{nk} , where n is the node index, and where k takes on a value of 1 and 2, denoting the resistor connected to the east and south of node n , respectively, as illustrated in Fig.3.16. In general, up to 4 resistors may be connected to a node. There are 2M boundary cases however, where only 3 resistors can be connected to a node, (see Fig.3.13 for $i=1$, and $i=N$).

The above numbering scheme does not identify the resistors in the first column*. Let these resistors be denoted R_i where i is the row index of the general model. Hence, for these N resistors,

$$R_{n-1,1} = R_i \quad (3.14)$$

where, n is given by (3.13), $i = 1, 2, \dots, N$, and $j = 1$.

To complete the model, the positive terminal of a voltage source of V units is connected to each resistor in the first column, and the negative terminal, to each resistor in the $(M+1)^{th}$ column, as shown in Fig.3.13.

Given a model of the type described in this section, it is possible to write a system of linear algebraic

*-----
In order to be able to use the same notation for resistors in the first column, as for the other resistors in the general model, it would have been necessary to define N additional nodes. These would have served no other useful purpose and would have made the numbering scheme dissimilar to that of the Ladder Diagram.

equations which can be solved for the voltage of each and every one of the P nodes, based on the current values of the resistors.

3.3.3 Derivation of Model Equations

The admittance across a resistor is defined as:

$$G_{nk} = 1 / R_{nk} \quad (3.15)$$

for $n = 1, 2, \dots, P$, and $k = 1, 2$. Then, for the typical node, n , shown in Fig.3.15, the application of Kirchhoff's current law yields the relationship

$$\begin{aligned} G_{n,1}(v_n - v_{n+1}) + G_{n,2}(v_n - v_{n+M}) + \\ + G_{n-1,1}(v_n - v_{n-1}) + \\ + G_{n-M,2}(v_n - v_{n-M}) = 0 \end{aligned} \quad (3.16)$$

where, v_n denotes the voltage at node n , etc..

There are four boundary cases which must also be considered. First, when $i=1$, the general model does not allow for the connection of a resistor to the row above, hence:

$$\begin{aligned} G_{n,1}(v_n - v_{n+1}) + G_{n,2}(v_n - v_{n+M}) + \\ + G_{n-1,1}(v_n - v_{n-1}) = 0 \end{aligned} \quad (3.17)$$

Similarly, when $i=N$, $R_{n,2}$ cannot be connected, hence:

$$G_{n,1}(v_n - v_{n+1}) + G_{n-1,1}(v_n - v_{n-1}) + G_{n-M,2}(v_n - v_{n-M}) = 0 \quad (3.18)$$

The two remaining boundary cases can be stated as follows;

$$v_{n-1} = v \quad (3.19)$$

for n as given by (3.13), $i = 1, 2, \dots, N$, and $j = 1$, and

$$v_{n+1} = 0 \quad (3.20)$$

for n as given by (3.13), $i = 1, 2, \dots, N$, and $j = M$. The special case of an unconnected node must also be considered.

When the node n is not connected,

$$v_n = 0. \quad (3.21)$$

By rearranging equations (3.16), (3.17), and (3.18), the following equations are obtained respectively:

$$\begin{aligned} v_n(G_{n,1} + G_{n,2} + G_{n-1,1} + G_{n-M,2}) + \\ - v_{n+1}G_{n,1} - v_{n+M}G_{n,2} - v_{n-1}G_{n-1,1} + \\ - v_{n-M}G_{n-M,2} = 0 \end{aligned} \quad (3.22)$$

$$\begin{aligned} v_n(G_{n,1} + G_{n,2} + G_{n-1,1}) - v_{n+1}G_{n,1} + \\ - v_{n+M}G_{n,2} - v_{n-1}G_{n-1,1} = 0 \end{aligned} \quad (3.23)$$

$$\begin{aligned} v_n(G_{n,1} + G_{n-1,1} + G_{n-M,2}) - v_{n+1}G_{n,1} + \\ - v_{n-1}G_{n-1,1} - v_{n-M}G_{n-M,2} = 0 \end{aligned} \quad (3.24)$$

Based on the above equations, and considering the boundary conditions where appropriate, at each of the P nodes in the general model, the matrix equation of the form,

$$G_s y = i \quad (3.25)$$

where,

G_s is the $P \times P$ admittance matrix

y is the $P \times 1$ vector of voltages at each node

i is the $P \times 1$ vector of the source currents into each node if each node is taken to be at ground potential,

can be written. Fig.3.16 shows the entries in G_s and y required to specify the value of the current, i_n , where i_n is the n^{th} element of i . Due to the node numbering scheme used, and to the particular construction of the model, the matrices in (3.25) possess some unique characteristics. These are:

- 1) From (3.22), (3.23), (3.24), and (3.25), it is

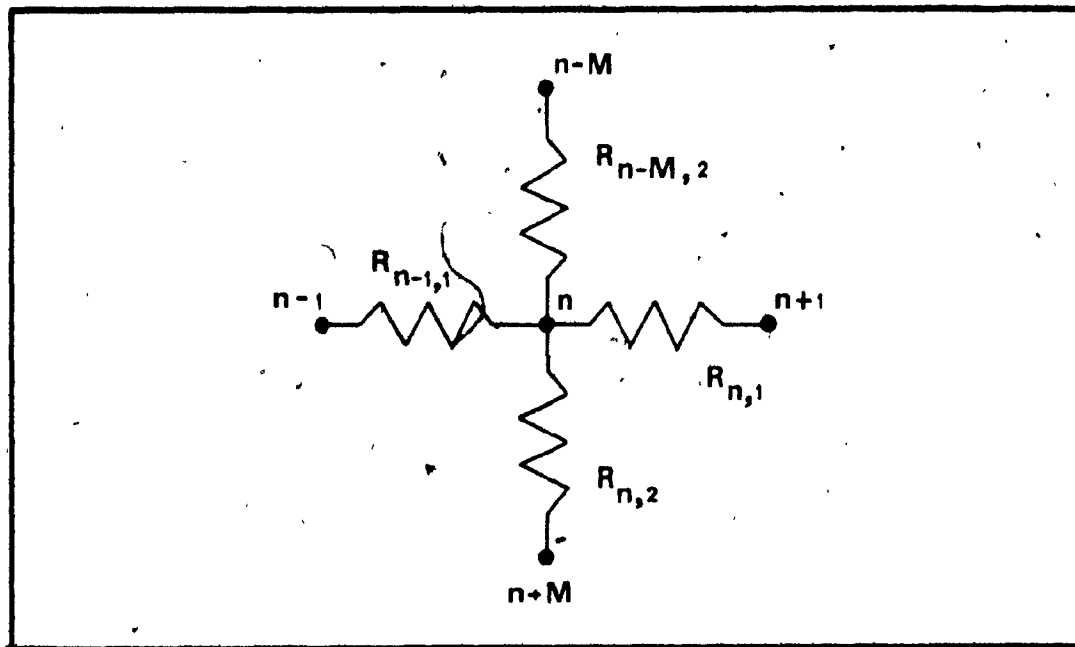


Fig.3.15 Node numbering scheme at the general node, n .

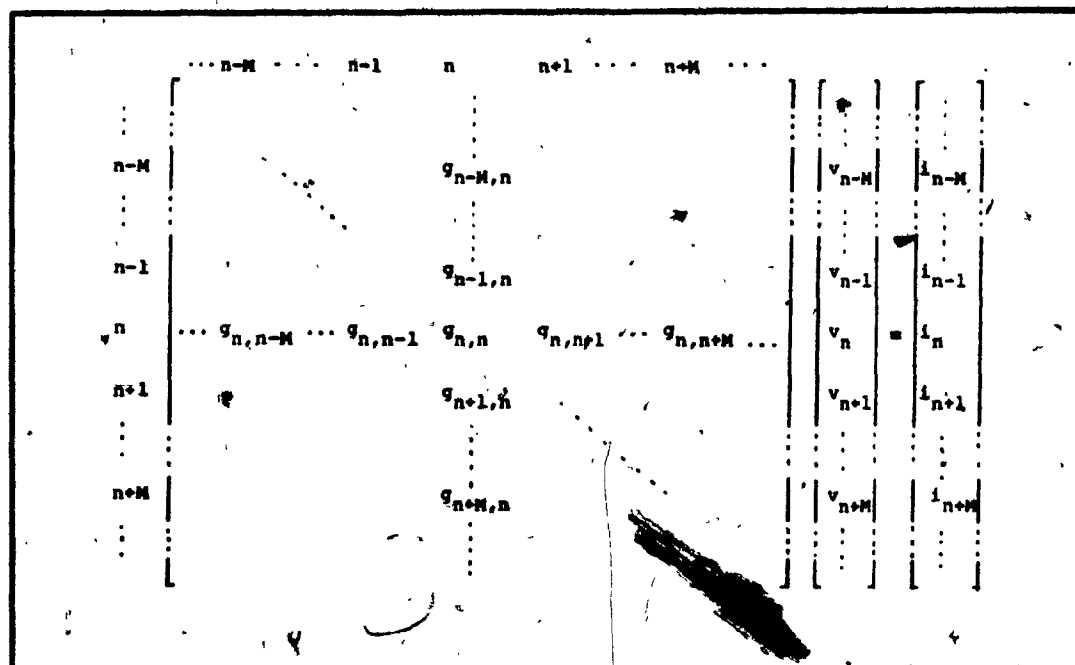


Fig.3.16 Entries in G , v , and i .

evident that the diagonal entry, g_{nn} , in G_s , is the coefficient of the nodal voltage v_n , in the equation for the value of i_n . In general, g_{nn} for $n = 1, 2, \dots, P$, is the sum of the admittances connected to node n .

- 2) The off-diagonal entries in G_s , are denoted g_{nl} , $n = 1, 2, \dots, P$, and $l = 1, 2, \dots, P$, and $n \neq l$. Again from the same set of equations as used in (1), it is evident that the off-diagonal elements in row n , i.e. $g_{n,n-M}$, $g_{n,n-1}$, $g_{n,n+1}$, and $g_{n,n+M}$, are the coefficients of v_{n-M} , v_{n-1} , v_{n+1} , v_{n+M} , respectively, in the equation for i_n . Furthermore, the values of these entries can only be 0, or -1.
- 3) The matrix G_s , will always be symmetric. This is based on reciprocity; if node n is connected to node $n+1$, then node $n+1$ is connected to node n .
- 4) Each row in G_s will contain at most 5 non-zero entries corresponding to the coefficients of v_n , v_{n-M} , v_{n-1} , v_{n+1} , and v_{n+M} .
- 5) The matrix G_s will always be banded and the

bandwidth will depend upon the node numbering scheme. The value of the bandwidth is determined by the number of nodes between node n and the most distant node to which it can be connected. For the numbering scheme used in the general model, $B = 2M+1$. (Had the nodes been numbered from top to bottom, and left to right, the value of B would be given by $B = 2N+1$. Since, a smaller bandwidth is desirable, and since typically, $M < N$, the numbering scheme selected provides a minimum bandwidth).

- 6) There will be at most N , non-zero entries in i_1 , (one for each unit-valued resistor in the first column). These resistors connect the voltage sources to the rest of the CIRCUIT. Each i_n has the value of the current flowing into the node n , from the source node, if the node n is taken to be at ground potential, i.e., $i_n = G_n V$.

Given the above characteristics, the system of equations describing any model can be written in the form given in (3.25) by inspection from the Ladder Diagram. This is illus-

trated by Fig.3.17 which shows the matrices G_s , \underline{v} , and \underline{i} as obtained by inspection of the first 3 lines of the sub-CIRCUIT shown in the ladder diagram of Fig.3.14b, if it is assumed that all the contact pairs have a transmission value of 1.

$$\begin{bmatrix}
 3 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 -1 & 0 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & -1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 2 & -1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 2 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}
 \begin{bmatrix}
 v_1 \\
 v_2 \\
 v_3 \\
 v_4 \\
 v_5 \\
 v_6 \\
 v_7 \\
 v_8 \\
 v_9 \\
 v_{10} \\
 v_{11} \\
 v_{12} \\
 v_{13} \\
 v_{14} \\
 v_{15} \\
 v_{16} \\
 v_{17} \\
 v_{18}
 \end{bmatrix}
 =
 \begin{bmatrix}
 \frac{v_R}{R} \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

G_s
 \underline{v}
 \underline{i}

Fig.3.17 G_s , \underline{v} , and \underline{i} as obtained by inspection from Fig.3.14b.

3.3.4 Solution of Model Equations

In order to simulate a CIRCUIT, the model equations must be generated and solved each time there is a change in the input or internal state. Since numerous solutions are to be executed in succession, and since an interactive simulation is desired, the speed with which the solution is obtained is a primary concern.

There are several well established methods available for the solution of systems of linear equations. These can be classified as either direct or iterative. A direct method was selected for use in solving the model equations for two reasons. First, the method is compact in terms of storage requirements, as well as in terms of the amount of computation it involves. Second, a solution will always be obtained quickly. Although convergence would be guaranteed by an iterative method, the rate of convergence would vary with the structure of the model [30]. A more predictable solution rate is preferable in this particular application.

The banded symmetric structure of the admittance matrix is exploited in order to increase the processing speed, as well as to reduce memory requirements. To do this, a modified version of a Gaussian elimination equation solver is used. Hence the amount of labor required to solve a system of the form given in (3.25) is almost halved [31]. Figure 3.18 shows how a banded symmetric matrix with

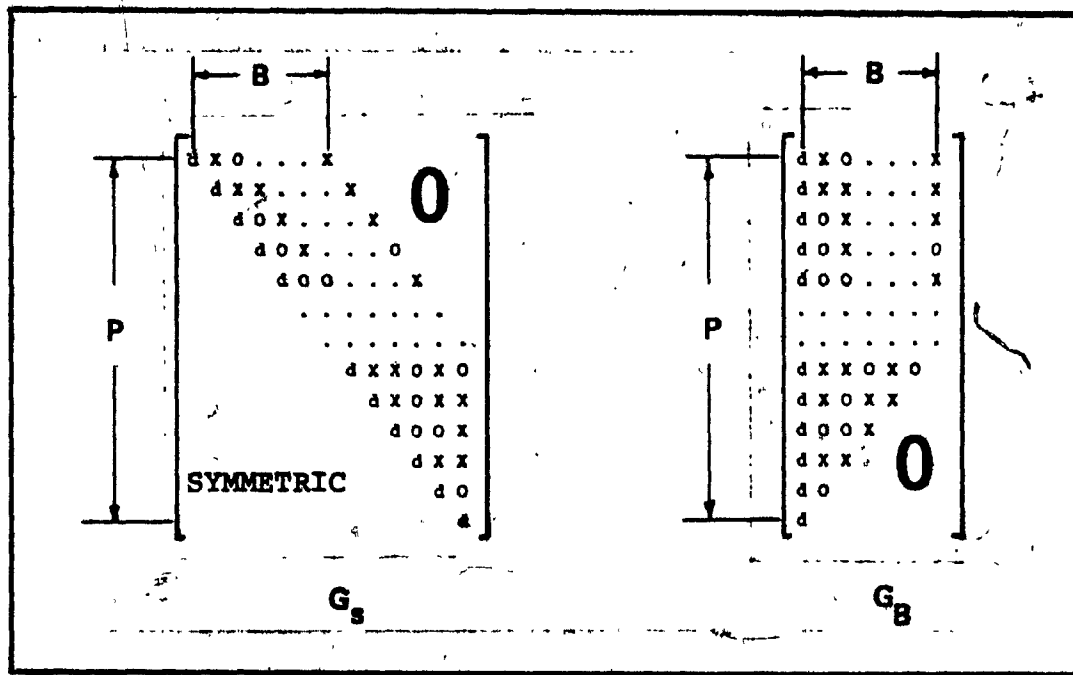


Fig.3.18 Band form storage of G_s .

semi-bandwidth,

$$B' = \frac{B - 1}{2} + 1 \quad (3.26)$$

can be stored in a banded form. In this storage format, all the information contained in the original matrix is retained. The storage requirement for G_B , the band form matrix, is reduced however as only the semi-band of elements is stored. Only $P \times B'$ storage locations are required to store G_B as compared to the $N(N+1)/2$ locations required to store the upper or lower triangle of the $P \times P$ symmetric matrix, G_S . Also considering the storage requirements for \underline{i} , the total memory required is given by

$$MEM_{matrix} = N_T(M^2 + M + 1) \times N_B \quad (3.27)$$

where N_B is the number of bytes required to store each variable. If single precision arithmetic is used, $N_B = 4$; if double precision is used, $N_B = 8$.

A FORTRAN coded Gaussian elimination equation solver based on the band form storage scheme is provided by Cook [32]. A modified version for the solution of the model equations is given in Appendix A, and will hereafter be referred to as GAUSS. A detailed description of GAUSS is also given in Appendix A.

3.3.4.1 Operation Count and Error Analysis for the Equation Solver

In the worst case, each floating point computation, or "operation", performed by GAUSS will contribute to the total error in calculating the first entry in the solution vector, \underline{v} . Let v_i be the true solution for the i th entry in \underline{v} , and let v_i' be the numerically calculated solution for the same entry. The error, δ_i is given by the expression

$$\delta_i = v_i - v_i'. \quad (3.28)$$

GAUSS has two distinct segments. The first performs a forward reduction of both G_s and $\underline{1}$. The second implements back substitution. The error in calculating every v_i will accumulate to give, in the worst case, a maximum error in the calculation of v_1 , (v_2, v_3, \dots, v_p will be used to determine v_1 during back substitution). Therefore, in order to determine the maximum error as a fraction of the source voltage, e , the number of operations is required. If it is assumed that all operations, contribute the same amount of error, then, the value of e is given by;

$$e = \frac{C_T (c_e \times 10^{-W})}{V} \quad (3.29)$$

where W is the number of significant digits stored by the

computer, C_T is the total number of operations, and

$$c_e = \begin{cases} 1.0 & \text{if error is due to truncation} \\ 0.5 & \text{if error is due to summetric rounding} \end{cases} \quad (3.30)$$

For a model with $N = 7$ and $M = 6$, the value of C_T as determined from (A.2) is $C_T = 2225$ operations.

3.3.5 Determination of Internal and Output States and Identification of Current-Carrying Devices

An output device is said to be "energised" if the contact network connected between that device and the voltage source in the Ladder Diagram has a transmission value of unity. It is said to be "off" otherwise.

The primary objective of the simulation is to determine the output state of a CIRCUIT given its internal and input states. The voltages at the nodes of the model are used to determine the output state of a CIRCUIT.

The secondary objective of the simulation is to determine which devices carry current, and to highlight them on the graphic display of the Ladder Diagram. This provides the operator with a picture of the current paths in a CIRCUIT, with the exception of a class of redundant current paths to be described later in this chapter.

In general, every output device is modelled as a unit-valued resistor with one terminal connected to a

resistive network, and the other terminal connected to ground, (Fig.3.13). If it can be established that there exists a potential difference, (PD), across a resistor modelling an output device, it can then be inferred that:

- 1) The resistive network to which the resistor is connected must comprise at least one set of serially connected resistors connecting the voltage source to this resistor (which models an output device).
- 2) Since each unit-valued resistor contained in this path models a CIRCUIT device with a transmission value of 1, from (3.5), the transmission value between the output device and the voltage source is unity.

Therefore, the output device modelled by the resistor (in the $(M+1)^{th}$ column, across which a PD is found to exist) is energised. Similarly, the current-carrying devices can also be identified. If a PD exists across a resistor, and if the value of the resistor is unity, it can be inferred that:

- 1) there is a flow of current through the resistor,
- 2) there is at least one path of series

connected, unit-valued resistors connecting one terminal of the current-carrying resistor to the voltage source, and hence, the transmission between this terminal and the voltage source is unity,

- 3) there is at least one path of series connected, unit-valued resistors connecting the other terminal of the current-carrying resistor to ground and therefore the transmission between this, other, terminal and ground is 1, and
- 4) an output device must be included in each and every path to ground since only output devices can be placed in the $(M+1)^{th}$ column of the Ladder Diagram, (Section 3.2.2).

Hence, every resistor found to be carrying current, models a device which is a member of a path of unity transmission between the voltage source of the Ladder Diagram and the ground.

Both the determination of the output state and the identification of the current-carrying devices in a CIRCUIT, require that the existence of a PD, or absence thereof, be established. To do this let

$$V_{ab} = v_a - v_b \quad (3.31)$$

where V_{ab} is the "true" value of the PD across the nodes a and b, and where v_a and v_b are the "true" values of the potentials, with respect to ground, of nodes a and b respectively. Then, the numerically calculated PD between nodes a and b is given by:

$$V_{ab}' = (v_a \pm \delta_a) - (v_b \pm \delta_b) \quad (3.32)$$

where δ_a and δ_b are the values of the numerical errors in calculating the nodal voltages at nodes a and b, respectively. Let e denote the maximum value of δ_m for $m = 1, 2, \dots, P$, for given values of M and N . Then, in the worst case,

$$V_{ab}' = V_{ab} \pm 2e \quad (3.33)$$

If $|V_{ab}'| > 2e$, the range of V_{ab} does not include 0. Therefore, it can be concluded that there exists a PD across the nodes a and b. If, on the other hand, $|V_{ab}'| \leq 2e$, the range of V_{ab} (i.e. $|V_{ab}| \leq 4e$) includes 0 and consequently the existence, or absence, of a PD cannot be immediately established.

Based on this author's extensive experience with the present method, and on the basis of a great many simulations, the minimum, non-zero, value of the PD across a

unit-valued resistor is 10^{-4} . Hence, by employing double precision arithmetic (i.e. $O(e) = 10^{-12}$) to calculate the nodal voltages as fraction of the source voltage, it can be concluded that, when $|V_{ab}| \leq 2e$, $V_{ab} = 0$ and hence there is no PD across nodes a and b.

Thus, the existence, or absence, of a PD across every unit-valued resistor in a model can be established and used to determine the output state of a CIRCUIT and to identify the current-carrying devices therein.

3.4 Special Case; the Balanced Bridge

The method for modelling and analysing CIRCUITS presented thus far guarantees two things:

- 1) all the output states of the CIRCUIT will be simulated correctly.
- 2) the unordered set of current-carrying devices, when presented on the graphics screen, will indicate the current paths in the physical CIRCUIT.

It was shown in Section 3.3.4, that all the current paths in a CIRCUIT are paths of logical transmission from the source of the "high" logic signal, (voltage source),

to an output device.

Consider the schematic of a model shown in Fig.3.19. This configuration is a balanced bridge if $R_1 = R_2 = R_3 = R_4$. In such a model, no current will flow through R_5 . This resistor however, models a device which is in fact, a member of a valid, redundant, path of logical transmission from the source to the output device.

As a result, in the case where the model of a CIRCUIT contains a balanced bridge, not all the paths of logical transmission will be represented by the current paths. Practically however, this is of little consequence since it can be easily shown that the path not shown is redundant. Furthermore, it is difficult to configure a CIRCUIT, the model of which, will contain a balanced bridge.

3.5 Timing Simulation of CIRCUITS

3.5.1 Introduction

The timing characteristics of a CIRCUIT ultimately depend upon two factors which must be considered in order to establish a basis for a CIRCUIT simulation. First, is the timing of the inputs to a CIRCUIT. Second, the timing characteristics of the devices it comprises.

For a physical CIRCUIT, the inputs are determined entirely by the process or machine being controlled. In a

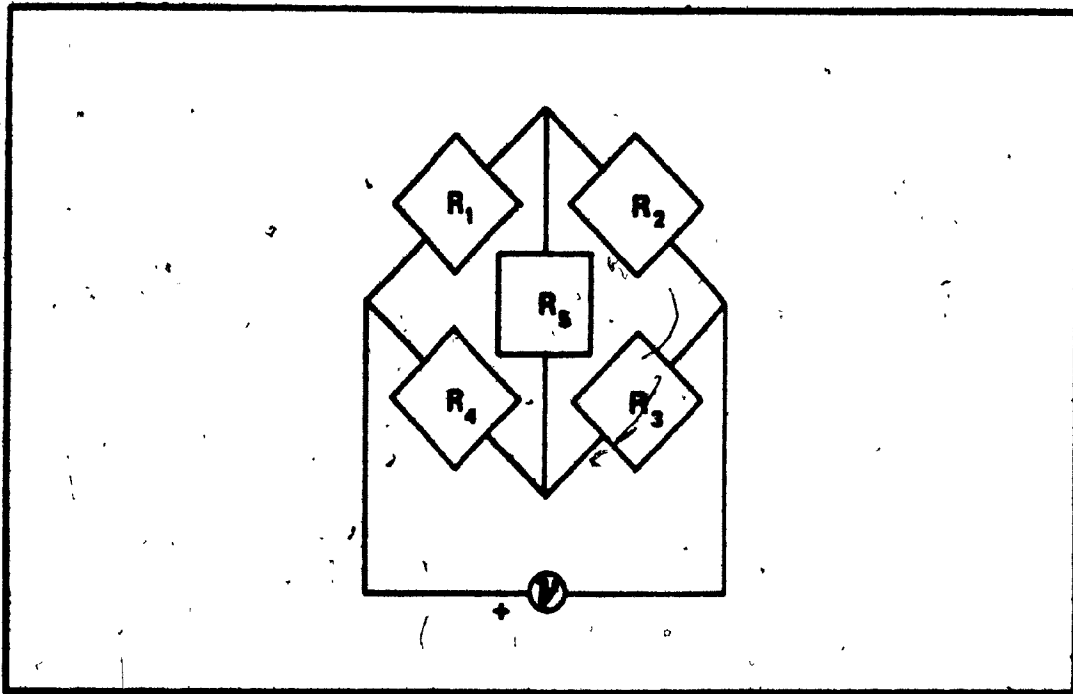


Fig.3.19 Bridge CIRCUIT.

simulation however, these inputs must be entered explicitly. In order to simulate the timing of inputs to, and outputs from, a CIRCUIT, as well as the delay characteristics of the devices it comprises, these must first be modelled, and a simulation time base must be defined.

3.5.2 Modelling of the Timing Characteristics of CIRCUIT Devices

Two types of delay must be considered when analysing the timing characteristics of a CIRCUIT; pure and inertial. An ideal inertial delay, is illustrated in Fig.3.20. The output will only respond to a change in input if the input change persists for at least t_{DI} time units, the inertial delay duration [34]. A pure delay acts only to transform a signal, $f(t)$, into $f(t - t_{DP})$, where t is the time variable, and t_{DP} is the delay duration, (Fig.3.21).

Relays, whether solid state or electromechanical, can be modelled as ideal inertial delays [35]. Two inertial delays can be associated with each relay device. The first, often referred to as the pull-in speed, is the period between the time the coil of a relay is energised and the time its contacts take on the appropriate state for that particular coil state. The second, referred to as the drop-out speed is similarly defined for the opposite coil state, and is usually about 60% of the pull-in speed of the same device [36]. The values of the delays range anywhere

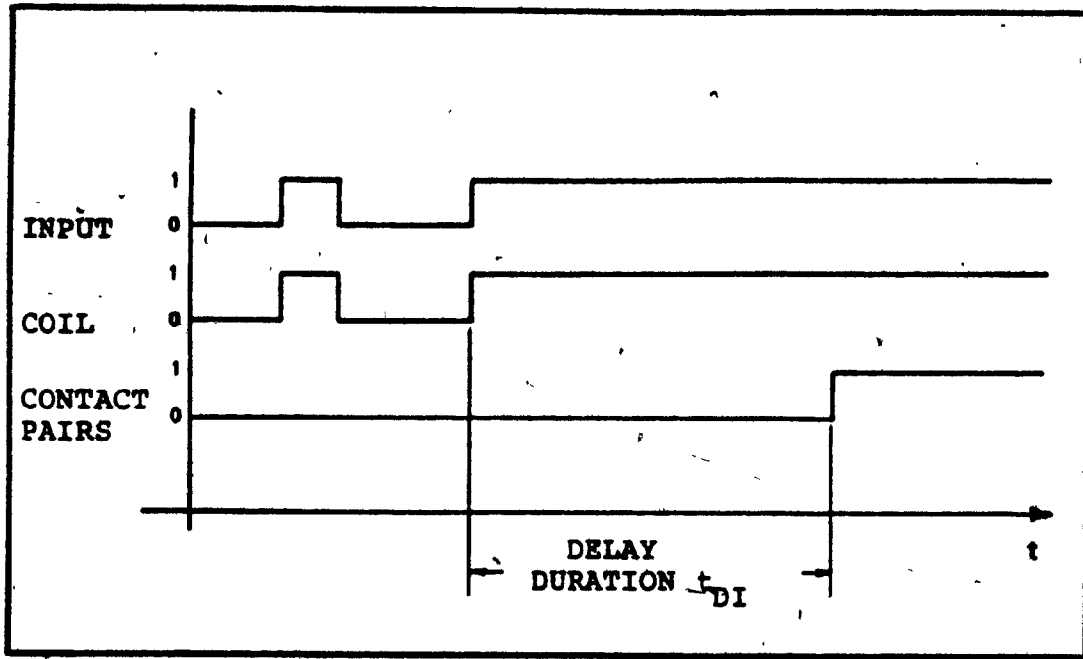


Fig. 3.20 Inertial Delay Characteristics.

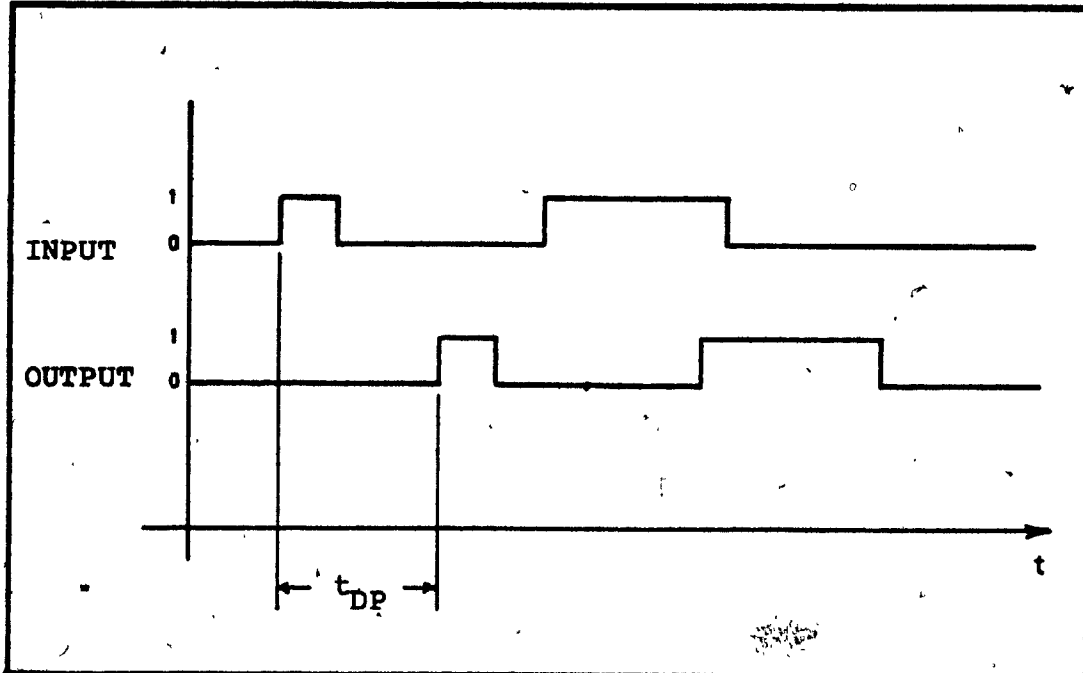


Fig. 3.21 Pure Delay Characteristics.

from 1 millisecond for solid state relays, to 30 milliseconds or more, for special, delay relays.

Timers are also modelled as ideal inertial delays. The durations of their delays however can vary anywhere from 1 second to many hours. Three types of timer are available: "slow activate"; "slow release"; and "slow activate, and slow release". The timing diagram shown in Fig.3.22 illustrates their respective characteristics.

The final class of CIRCUIT device to be considered are the external input and output devices. For these devices, the propagation speeds of the signals through them can be modelled as pure delays, and are in the order of magnitude of the speed of light in duration. Since inertial delays are in the order of several milliseconds or more, the values of pure delays can be assumed to be negligible.

3.5.3 Simulation Clock

In a CIRCUIT, the input states are determined by occurrences of changes in the states of control variables, or of changes in the internal state. Such occurrences are termed "events". The analysis presented thus far, provides a means of determining the output state of a CIRCUIT for a given input state.

The operation of a CIRCUIT can be simulated by successive analyses. On the completion of each analysis, the CIRCUIT will be found to be in one of four possible

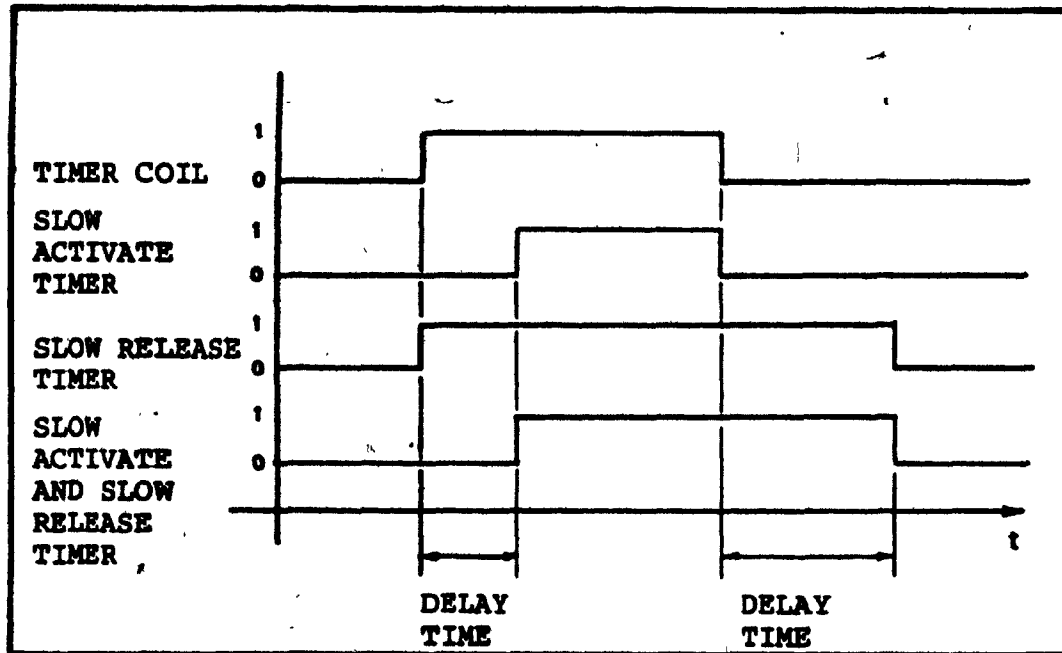


Fig.3.22 Relay Device types.

states which can be described in terms of the states of the relays and timers therein. A relay or timer device in a CIRCUIT is said to be in a "transient" state if the contact pairs associated with it are in transition from one state to another. This definition also includes timers and delay relays, the coils of which have been energised and the contact pairs of which have not assumed the appropriate state. It is said to be at a "steady" state otherwise. A CIRCUIT is said to be in a steady state if there are no timers or relays in transient states at a particular instant. It is said to be in a transient state if it comprises at least one device which is in a transient state.

In order for a CIRCUIT model to advance from one transient state to another, the passing of time must be simulated. A counter will be used to count time steps of duration D_t and will serve as a simulation clock. To simulate relay and timer delays, the coils for each of these will have a delay counter associated with them. Each incrementation of the simulation clock will cause the delay counter for each coil which is in a transient state to be incremented by the same amount. When the delay counter for a particular device has counted a preset number of steps, the delay duration will have been simulated, and the states of the contact pairs for that coil will be set.

This scheme can be applied to obtain either an absolute or a relative time simulation. Only the latter will be considered however, in order to avoid the necessity

for the simulation of steady states of long duration. Hence, the timing of a CIRCUIT will be simulated from the instant it enters a transient state, until it reaches a steady state, at which time, the simulation clock will be reset. When only "direct" type CIRCUIT devices are involved in the operation of a CIRCUIT, there is no need to simulate the timing as the operation of these devices has been assumed to be instantaneous.

The value of D_t will define the time resolution of the simulation. Consider the n possible events for a particular CIRCUIT, E_1, E_2, \dots, E_n , occurring at time t_1, t_2, \dots, t_n respectively such that $t_{i+1} > t_i$, for $i=1, 2, \dots, n-1$, and where $\Delta t_i = t_{i+1} - t_i$ expresses the duration between events. If $\Delta t_i = 0$, E_i and E_{i+1} are said to be simultaneous. Otherwise, they are said to be successive, where the minimum duration between successive events is given by:

$$\Delta t_{\min} = \min \{ \Delta t_i \mid \Delta t_i \neq 0 \} \quad (3.34)$$

for $i = 1, 2, \dots, n-1$.

In order that all possible events can be simulated, the condition,

$$D_t \leq \Delta t_{\min} \quad (3.35)$$

must be satisfied. In general however, the value of Δt_{\min}

is not known. Therefore, the value of D_t can be selected on the basis of a compromise between the desired resolution and the number of CIRCUIT analyses necessary to perform a simulation. Then, if $\Delta t_i \leq D_t$, E_i and E_{i+1} will be simulated as simultaneous events. If $\Delta t_i > D_t$, then E_i and E_{i+1} will be simulated as successive events. For the purpose of this Thesis, D_t will be taken to be 1 millisecond, which is sufficient to simulate almost any CIRCUIT. Further discussion on the details of how the simulation clock is advanced is deferred to Chapter 5.

3.6 Summary

In this chapter, a novel technique for the analysis of CIRCUITS is presented. Given a Ladder Diagram of a CIRCUIT, a resistive network model is generated. The system of linear, algebraic equations which describe this model are subsequently solved to generate information with which, the output state can be determined. Furthermore, the current-carrying resistors in the model can be identified. These represent the current-carrying devices in the physical CIRCUIT which constitute the paths of logical transmission from the voltage source to the output device. The analysis technique guarantees the correct determination of the output state of any CIRCUIT containing sub-CIRCUITS of no more than N rows, where N is the allowable number of rows in a sub-CIRCUIT, be the current paths forward- or backward-directed.

This chapter also describes the basis for the simulation of the timing characteristics of CIRCUITS. Two independent simulation clocks are used in order to simulate the inertial delays of relays and delay relays, (which have delay durations in the order of a few milliseconds), as well as of timers (which have delays in the order of seconds, minutes, or hours).

CHAPTER 4
MODELLING AND ANALYSIS FOR CIRCUIT SIMULATION
USING GRAPH THEORY

4.1 Introduction

In the previous chapter, a method which models and analyses a CIRCUIT as a resistive network in order to determine its output state, as well as to identify the current-carrying devices in it, was described. One shortcoming of this method is that it requires double precision, floating-point arithmetic to solve the relatively large system of equations that describe the model of a CIRCUIT. In order to reduce the amount of memory required, the CIRCUIT is divided into sub-CIRCUITS and is analysed in a piecemeal fashion. Even with this modification however, the method is both memory- and time-consuming. Another shortcoming is that there are instances where devices that carry current in redundant paths of logical transmission in the actual CIRCUIT, will not be identified by the analysis. This was shown in Section 3.4.

This chapter describes an alternative method which models the CIRCUIT as a "graph", and uses a variation of an established Graph Theory technique known as the "depth-first search" (DFS) to analyse the model and to obtain the required output states and current paths. Unlike the previous method, which divides the CIRCUIT into

sub-CIRCUITS, the present method operates on the entire CIRCUIT and performs the analysis using only logical operations. Although the method is described only in concept (in the form of an algorithm), if implemented, it would require less memory and would analyse a given CIRCUIT more quickly than does the previous method.

The Graph Theory terminology to be used in the following discussion is defined in most text books on the subject [36,37]. A short glossary is given in Appendix B for the reader's convenience.

4.2 Modelling of a CIRCUIT as a Graph

A CIRCUIT can be modelled as a graph directly from its Ladder Diagram representation and from its input and internal state. A graph is obtained by modelling each device with a transmission value of 1 (Section 3.2.2) as an "edge". The edges of the graph are arranged in a grid, identical to that for the Ladder Diagram, on the basis of a one-to-one correspondence between their respective grid positions, and the positions of the CIRCUIT devices they model in the Ladder Diagram.

Consider the Ladder Diagram shown in Fig.4.1. Given the input and internal state of the CIRCUIT, a representation of the type shown in Fig.4.2 can be drawn to show the transmission value of each device in the Ladder Diagram. The graph of this CIRCUIT can be obtained by

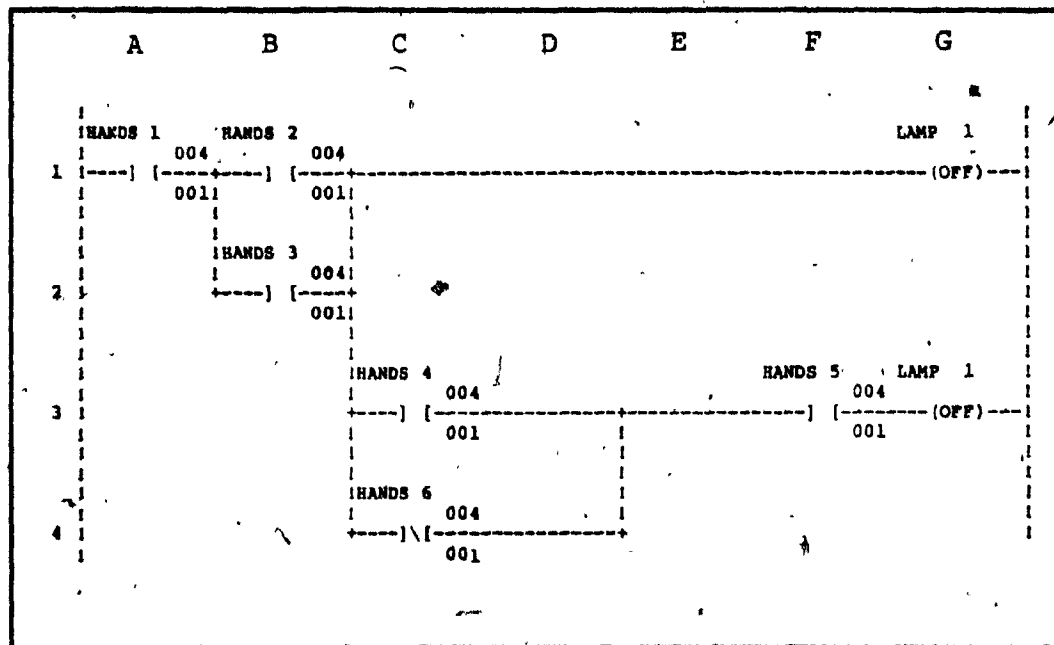


Fig. 4.1 Sample Ladder Diagram.

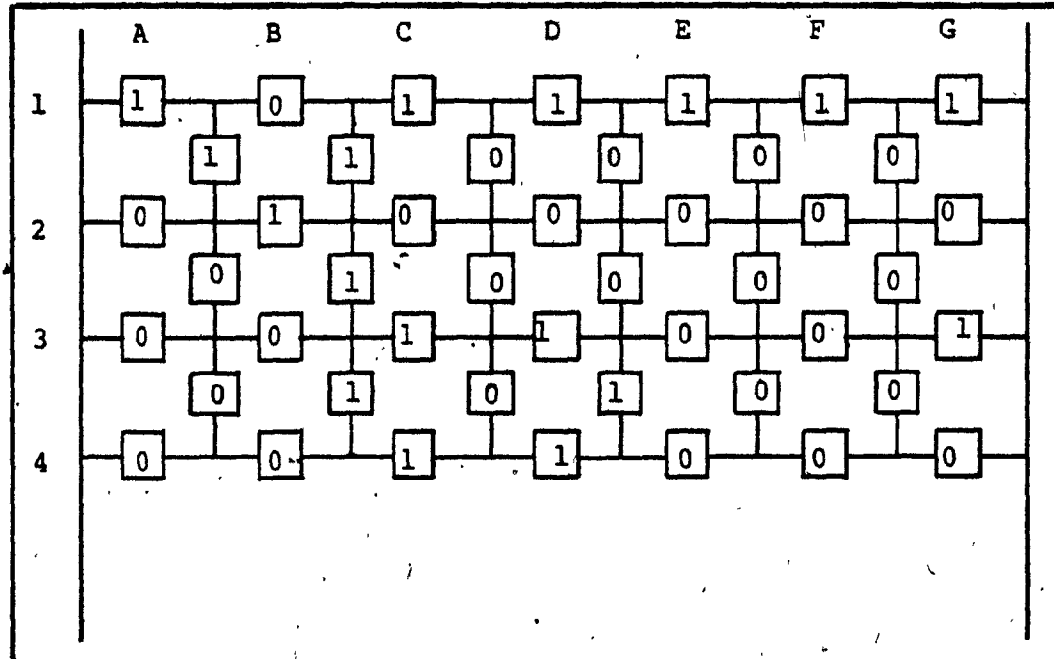


Fig. 4.2 Representation showing transmission values of devices.

drawing an edge at each position where there is a device with a transmission value of 1, and by leaving a blank at the positions in the grid where the device has a transmission value of 0, as shown in Fig. 4.3. In general, the graph of a CIRCUIT will be unconnected.

For a CIRCUIT represented by a Ladder Diagram of N_T rows and $M+1$ columns, the model is constructed on a grid of N_T rows and $M+2$ columns. It comprises a maximum of E_g edges, and up to V_g vertices, where:

$$E_g = (2M+1)N_T - M \quad (4.1)$$

and

$$V_g = (M+2)N_T \quad (4.2)$$

Vertices, which are numbered from left to right, and top to bottom, as shown in Fig. 4.4. The serial identification of the vertex in the i^{th} row and j^{th} column for $i = 1, 2, \dots, N_T$ and $j = 0, 1, 2, \dots, M+1$, is given by:

$$v = (i-1)(M+2) + (j+1). \quad (4.3)$$

The vertices at $j = 0$ will hereafter be referred to as source vertices, and those at $j = M+1$ will be referred to as ground vertices.

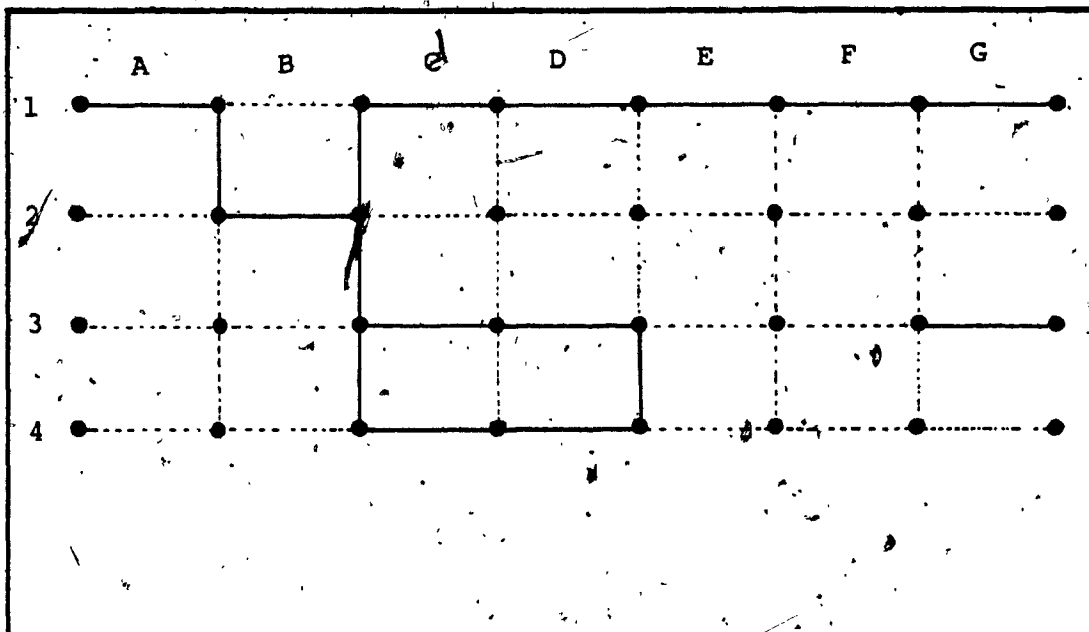


Fig. 4.3 Graph for sample Ladder Diagram.

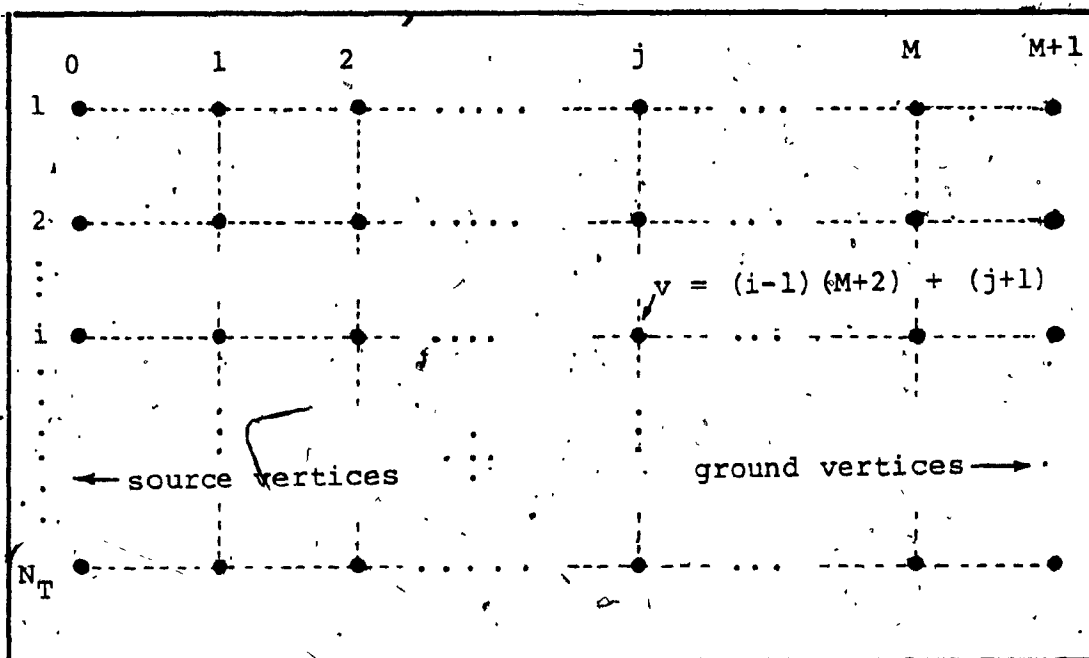


Fig. 4.4 Vertex numbering scheme.

4.3 Analysis of a Graph to Determine the Output State of a CIRCUIT

In general, the state of every output device is defined by the transmission value of the contact network connected between it, and the voltage source of the Ladder Diagram (Section 3.3.4). Given the graph, G , of a CIRCUIT, the Graph Theory can be applied to determine this transmission value for every output device in that CIRCUIT, and to subsequently determine the output state.

The techniques used, are based on the premise that for every output (or memory) device in the CIRCUIT which is energised, there must exist at least one set of serially connected devices, each with a transmission value of 1, which connects that output (or memory) device to the voltage source. Every such set in G will be modelled by an elementary path with an initial, source vertex, and a terminal, ground vertex and constitutes a connected subgraph of G . Hence, by enumerating all the connected subgraphs which contain at least one source vertex and at least one ground vertex, the output state of the CIRCUIT is determined.

In order to determine if such a path, or such a set of paths, exist(s), all the spanning trees which include source vertices of G are enumerated. Considering each such spanning tree that contains one or more ground vertices separately, it can be concluded that there exists an

elementary path from a source vertex, to every ground vertex contained by that spanning tree. This conclusion can be drawn since, for a connected, undirected graph, such as the spanning tree, every vertex is connected to every other vertex by an elementary path [38]. Hence, every ground vertex in a spanning tree containing at least one source vertex has, incident on it, an edge which models an energised output device (or memory).

4.4 Identification of Current-Carrying Devices

Every edge belonging to a path from a source vertex to a ground vertex models a CIRCUIT device which is carrying current. A method is required which will be capable of identifying these devices so that they may be highlighted on the graphic display of the CIRCUIT in order to indicate the current paths in it.

Although the method of enumerating the spanning trees of a graph can determine whether or not there are any elementary paths from a source vertex to a ground vertex, it is not capable of enumerating these paths. This is because, by definition, a tree does not include closed paths*. Hence, although there may be several elementary paths from a

* These paths are referred to, in the literature, as "circuits" but the term is not used in this Thesis, to avoid confusion.

source vertex to a ground vertex, the spanning tree will include only one of the elementary paths.

To solve this problem, a backtracking method for exploring a graph, known as the depth-first search (DFS), has been used as the basis for a CIRCUIT analysis algorithm. This algorithm is capable of both determining the state of the output devices, and of identifying the current-carrying devices in a CIRCUIT.

4.5 Description of the DFS

The DFS is a systematic method for exploring any graph which does not contain "self loops." If the graph to be explored is connected, the DFS need only be applied once. If it is unconnected, the DFS must be applied to each "part". The following description of the DFS is based on [39] and [40].

The application of the DFS begins at a specified starting vertex, or "root vertex", r . In general however, the DFS can be described with reference to a general vertex, s .

From s , an edge (s,v) , which is incident on s , is selected and traversed in order to "visit" v . Once v is reached, the edge (s,v) is said to have been "explored". If v has not been visited previously, then, when (s,v) is explored, a direction from s towards v is imposed by the DFS, with s being called the "father" of v , and v , the "son"

of s . Furthermore, as v is visited for the first time it is assigned an integer value, the "depth-first number" (DFN), which indicates the order in which it was visited. If, in exploring (s, v) , v has already been visited, (i.e., $DFN(v) < DFN(s)$) (s, v) is directed from v towards s , and (s, v) is called a "back edge".

Having visited v , the DFS proceeds to explore another edge incident on v (if one exists), and will continue to do so until it reaches a vertex which has been "completely scanned"; that is, a vertex which has no unexplored edges incident on it. The DFS will then "backtrack" to the most recently visited vertex having unexplored edges incident on it and continue the search from that vertex. When all the vertices have been visited and all edges have been explored (i.e. when search returns to r), the DFS is terminated. The result is a directed spanning tree of the graph, or subgraph. Examples illustrating the DFS, and algorithms describing its general implementation are given in both [39], and [40].

4.6 Output Device State Determination and Identification of Current-Carrying Devices Using the DFS

Consider the graph, G , which models a CIRCUIT. By applying the DFS using every unvisited source vertex in G as a root, a directed spanning tree can be enumerated for each connected subgraph, which contains at least one source

vertex. Furthermore, for each ground vertex contained in such a spanning tree there must be at least one elementary path from the source vertex to that ground vertex (Section 4.3). Hence, every edge incident on a ground vertex which is contained in a directed spanning tree rooted at a source vertex, models an output device (or memory) which is energised.

The graph, G , for a CIRCUIT is given in Fig. 4.5. By applying the DFS using vertex r as the root, the directed spanning tree, G' shown in Fig. 4.6 can be obtained. (G' is not unique. It is possible to obtain a directed spanning tree with a different ordering by exploring the edges in G in a different order.) The graph G' contains 2 source vertices, (1) and (23), and one ground vertex, (15). Hence, the output (or memory) device modelled by the edge (14,15) will be energised.

With reference again to Fig. 4.6 and by ignoring direction, it can be seen that G' contains all paths from each of the source vertices, to the ground vertices. Also recall that an elementary path is defined as a sequence of edges in which all the vertices are distinct. Now, consider the undirected version of the subgraph of G' shown in Fig. 4.7a, where the edges have been labelled to simplify the notation to be used. It is obvious that any path from 1 to 3 other than a , b , will involve a path which is not elementary. Similar reasoning can be used for the undirected versions of the subgraphs of G' shown in Fig. 4.7b

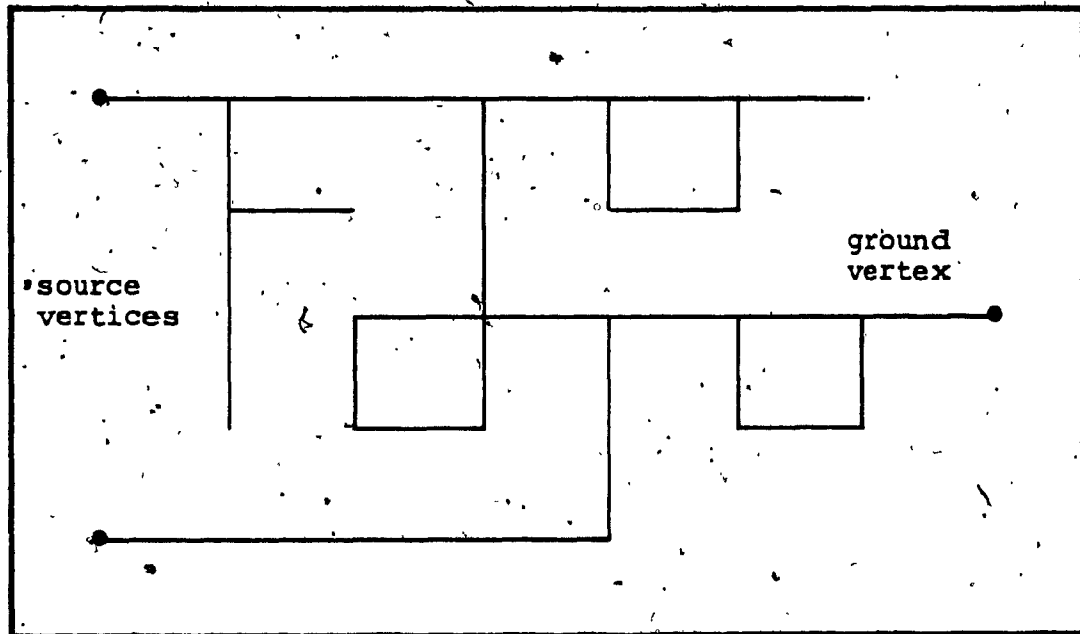


Fig.4.5 Graph, G .

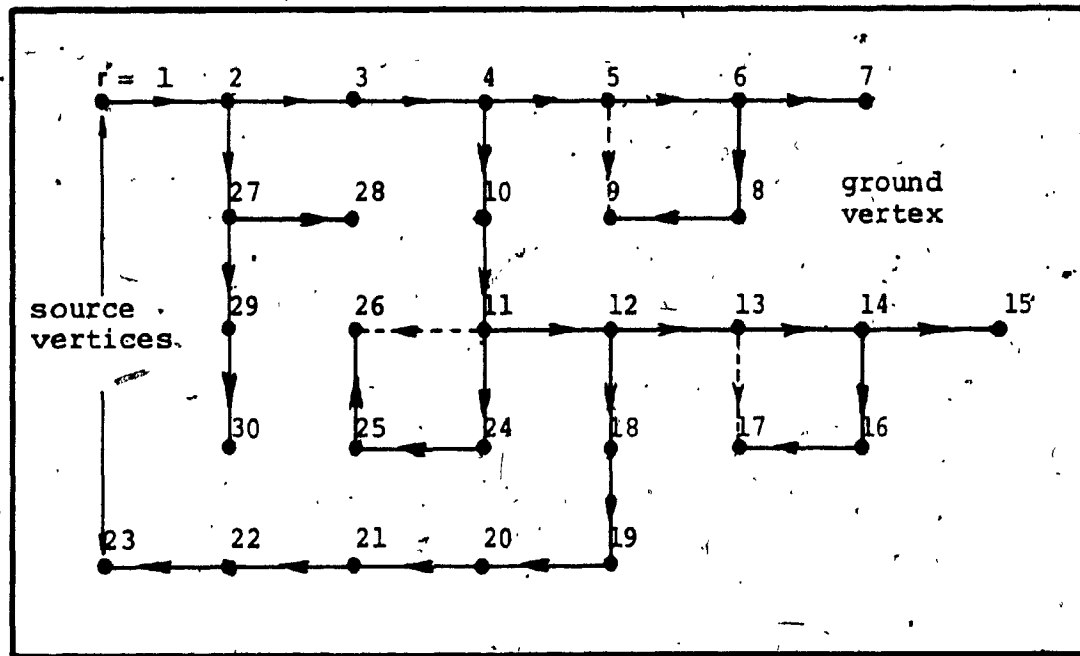


Fig.4.6 Directed spanning tree, G' , of G , showing DFN for each vertex.

and Fig.4.7c. In these figures, the only elementary path from 10 to 12 and 3 to 10, respectively, is a,b.

For the purposes of this analysis it is necessary to modify the DFS so that elementary paths can be distinguished from those which are not elementary. More precisely, subtrees of the DFS tree which either contain no vertices, or only 1 vertex of an elementary path, cannot comprise edges which belong to elementary paths and must hence be rejected. The subtrees of the DFS tree which must be rejected can be classified into two groups: those which comprise back-edges, and those which do not.

The author has yet to develop a systematic method for rejecting subtrees belonging to the first group. For subtrees belonging to the second group however, this author has developed a method which has been developed which can easily be incorporated into the DFS.

Any edge of the DFS tree which is incident on a vertex having only one explored edge incident on it, source and ground vertices excluded, cannot be part of an elementary path for the reasons previously stated. Hence, if an edge satisfies this description it is stricken. Consequently, the number of explored edges incident on the other vertex on which the stricken edge was incident will be reduced by 1. In this manner, all the non-elementary paths belonging to the second group can be eliminated. Spurious paths generated by the method can be rejected by inspection.

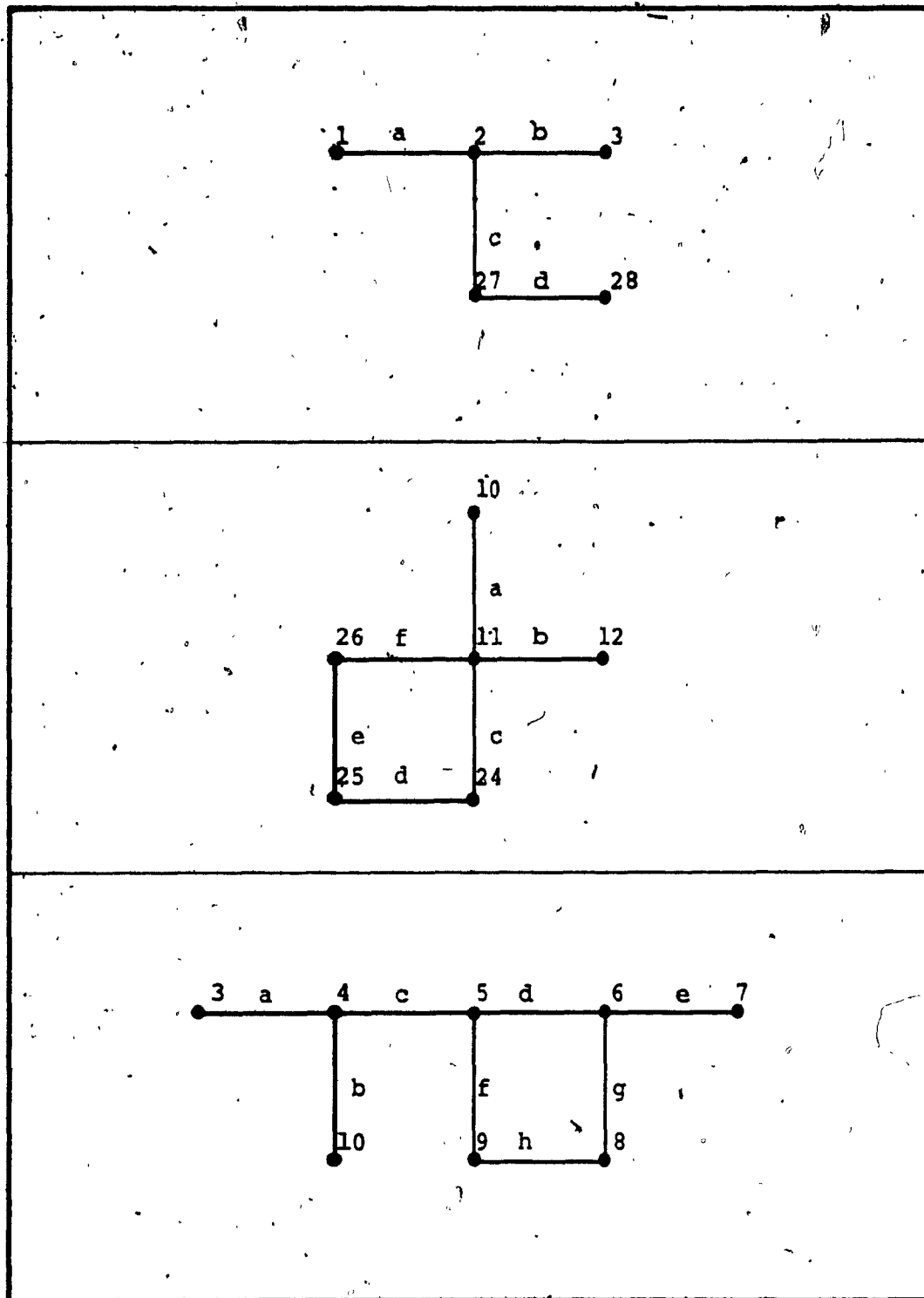


Fig.4.7 Sub-graphs of G' .

Noting the limitations of the method, the CIRCUIT Analysis Algorithm (CAA) has been developed on the basis of the DFS Algorithm as given in [40]. Applied to the graph of a CIRCUIT, the CAA will directly determine the output state as well as identify the current-carrying devices, regardless of the orientation of the current paths to which they belong.

4.7 Description of the CIRCUIT Analysis Algorithm

Given the Ladder Diagram of a CIRCUIT and the input and internal states, a graph, G , can be generated in the manner described in Section 4.1. In the actual implementation of the method, G could be generated as part of the search algorithm. This would involve a systematic process of identifying the transmission value of the device in each grid location of the Ladder Diagram. In order not to unnecessarily complicate the algorithm however, it will be assumed that G is available in a suitable form.

The CAA requires several working arrays for its operation. As was indicated previously, one array, $DFN(v)$, is required to store the DFN of every vertex. Another array, $FATHER(v)$, is required to store the location of the father of each vertex when it is visited for the first time.

As the edges of G are explored, the CAA graph is generated. The CAA graph possesses many of the characteristics of the DFS tree. The main difference is

that it contains only those edges which model devices that will be shown to carry current in the display of the Ladder Diagram of the particular CIRCUIT being analysed. This includes the spurious paths described previously but does not include the edges which are rejected.

Starting at each unvisited source vertex of G, the CAA proceeds to generate a number of generally unconnected, directed sub-graphs. Each sub-graph is assigned a graph number (GRAPH_NUM) and as each edge in that sub-graph is explored, it is tagged with that number and is assigned a direction. An edge can be stricken from a graph by setting its tag equal to 0. In the event that a sub-graph contains no ground vertices, it is necessary to strike all the edges of that sub-graph. Rather than exploring the graph again in order to strike these edges, a storage area (GRAPH_STACK) is created to store the graph numbers of the graphs which contain ground vertices. When the CAA graph has been completely explored, only those edges which are tagged by graph numbers which are contained in GRAPH_STACK will be considered to be edges of the CAA graph.

The memory required to store the working arrays is given by:

$$MEM_{\text{graph}} = N_T(8M + 10) - 2M + 100 \quad (4.4)$$

where N_T is the number of rows in the Ladder Diagram,

M is the number of columns,
and MEM_{graph} is the number of bytes required to
store DFN, FATHER, GRAPH_NUM, AND
GRAPH_STACK.

The method by which MEM_{graph} is obtained is described in Appendix C. With regard to the memory requirements of the method described in Chapter 3, it can be seen that the storage requirements increase linearly with the number of rows for both methods. For a fixed value of columns, M, however, the requirements for the previous method (Chapter 3) will always be greater when an equal number of rows are considered.

A formal description of the CAA is given in Fig.4.8. Figure 4.9 shows the graph of a CIRCUIT which was chosen to illustrate the CAA. All output devices modelled by edges incident on ground vertices having DFNs not equal to 0 will be energised. In Fig.4.9, the numbers in square brackets indicate the value of GRAPH_NUM for each edge and the numbers in parentheses indicate the DFN number for each vertex. The arrows on each edge indicate the direction assigned during the search and edges drawn as dashed lines indicate back edges.

Notice that the edges (6,7), (2,10), (10,11), (10,18), (18,26), (44,55), and (45,46) have, 0 as the value for GRAPH_NUM to indicate that they do not belong to elementary paths and will therefore not represent current-

S1. For every vertex, $v = 1, 2, \dots, V_g$ in G , set $FATHER(v) = 0$ and $DFN(v) = 0$. Also initialize $TREE_STACK$ by setting all entries in it to 0.

For every edge, $w = 1, 2, \dots, E_g$, in G , set $TREE_NUM = 0$.

S2. Locate next root vertex, r , such that $DFN(r) = 0$ which has one edge incident on it. Let i be equal to the row index in G where r is found.

If no such vertex can be found, go to S8.

S3. $j = 1$

S4. $DFN(r) = j$

$v = r$

S5. If all edges incident on v have already been explored, go to S7. Otherwise, select an edge (v, v') which has not yet been explored and go to S6.

S6. IF $DFN(v') = 0$, then:

set $j = j+1$

$DFN(v') = j$

$TREE_NUM((v, v')) = i$

Fig.4.8 CIRCUIT Analysis Algorithm...cont'd

DIRECTION((v,v')) = from v towards v'

FATHER(v') = v

v = v'

IF v' is a ground vertex, then:

set TREE_STACK = TREE_STACK U i

ELSE, continue

ELSE, set TREE_NUM((v,v')) = i

DIRECTION((v,v')) = from v' towards v

go to S5

S7. IF FATHER(v) ≠ 0, then:

IF there is only one edge, w, having

TREE_NUM(w) ≠ 0 incident on v, then:

set TREE_NUM((FATHER(v)),v) = 0

ELSE,

v = FATHER(v)

go to S5

ELSE, go to S2

S8. HALT

Fig.4.8 cont'd ... CIRCUIT Analysis Algorithm

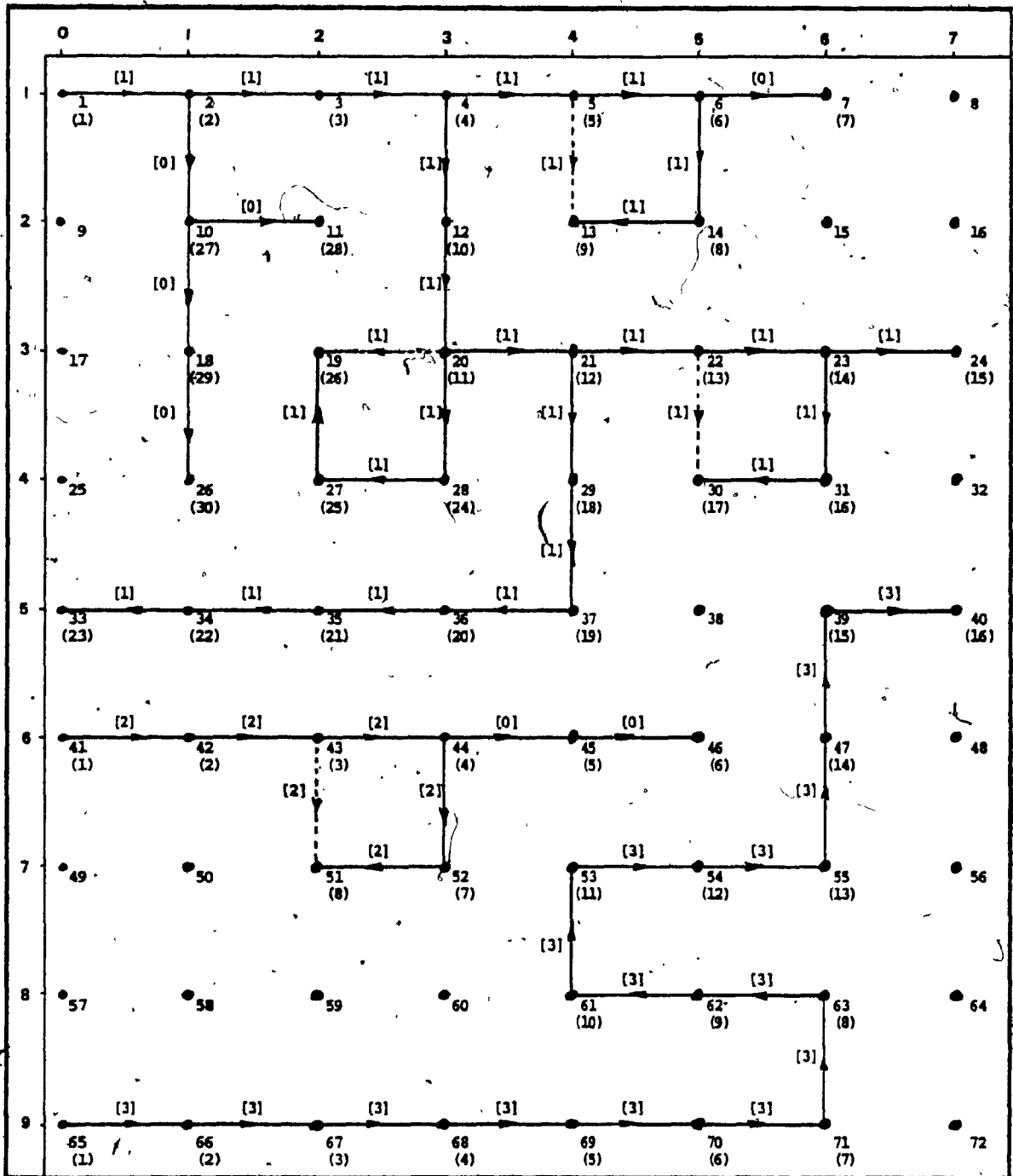


Fig.4.9 CIRCUIT Analysis Algorithm (CAA).

carrying devices. Also notice the spurious paths {4,5,6,14,13}, {4,5,13}, {20,28,27,19}, and {20,19}. Finally, since the directed spanning tree rooted at vertex 41 does not contain a ground vertex, edges for which `GRAPH_NUM = 2` will not be considered to model current-carrying devices in the CIRCUIT. All other edges will be taken to represent current-carrying devices.

4.8 Summary

In this chapter, the CAA, a method for modelling and analysing a CIRCUIT as a Graph has been described, and an algorithm for its implementation is given. The CAA is based on a systematic technique for exploring a graph known as the depth-first search. By exploring the graph of a CIRCUIT and by identifying the elementary paths it comprises, the CAA identifies the devices which will be carrying current in the physical CIRCUIT. Furthermore, it can correctly determine the output state for a CIRCUIT, regardless of the orientation of the current paths in it, be they forward- or backward-directed.

The storage requirement for the implementation of this method is less than that of the method described in Chapter 3. Furthermore, since only logical operations are required, this method would also be considerably faster.

One important shortcoming of the CAA is that it cannot distinguish between elementary paths and a special

class of non-elementary paths. Hence, the display of the current-carrying devices, which depends on the correct identification of the elementary paths, will at times show spurious paths. Although this is not desirable, it is not a serious problem since the output state termination, which is of primary interest, will always be correct. Furthermore, at no time will an actual current path not be shown.

CHAPTER 5

DESCRIPTION OF THE IMPLEMENTATION OF THE CIRCUIT SIMULATION PROGRAM

5.1 Introduction

The CSP allows the user to enter CIRCUITS using Ladder Diagram symbology and to simulate these CIRCUITS interactively, on a color graphics screen. Based on the logic and timing analysis methods described in Chapter 3, the CSP provides the following features:

- an analytical basis for determination of the output state.
- the identification of current-carrying devices.
- simulation of timing during transient states of the CIRCUIT.
- capability for analysing backward-directed current paths.
- capability for simulating both PLC and relay CIRCUITS, independently of the physical realization (general purpose computer based simulation).

Using dedicated function keys and input prompting, the CSP makes it possible for anyone familiar with Ladder Diagram symbology to specify and simulate complex CIRCUITS. This chapter details the various aspects of the implementation of the CSP.

5.2 Development System

The computer system on which the CSP was developed comprises:

- A DIGITAL VAX 11/780 minicomputer with a 3 mega-byte memory that uses the VAX VMS 3.3 operating system.
- A DIGITAL VT52 data terminal.
- A NORPAK VDP-3 Visual Data Processor, equipped with 3 memory banks of 4 bit planes. This is a raster graphics system with a 512x512 pixel resolution.
- A DIGITAL LA120 terminal used as a line printer.

A schematic of the system configuration is shown in Fig. 5.1, and a typical work station is shown in Fig. 5.2.

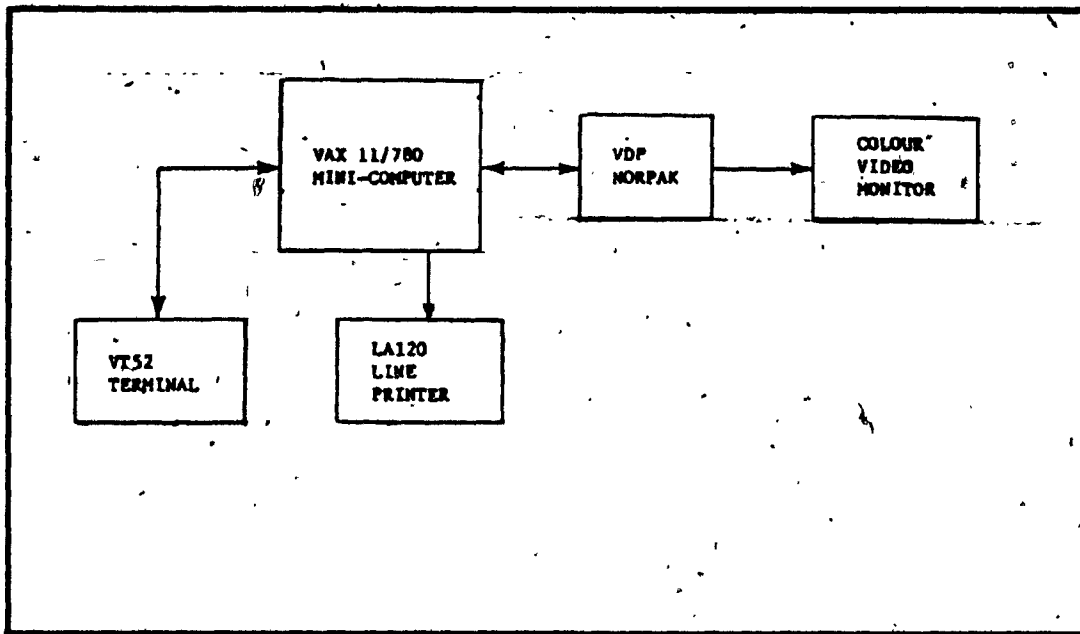


Fig.5.1 Configuration of development system.

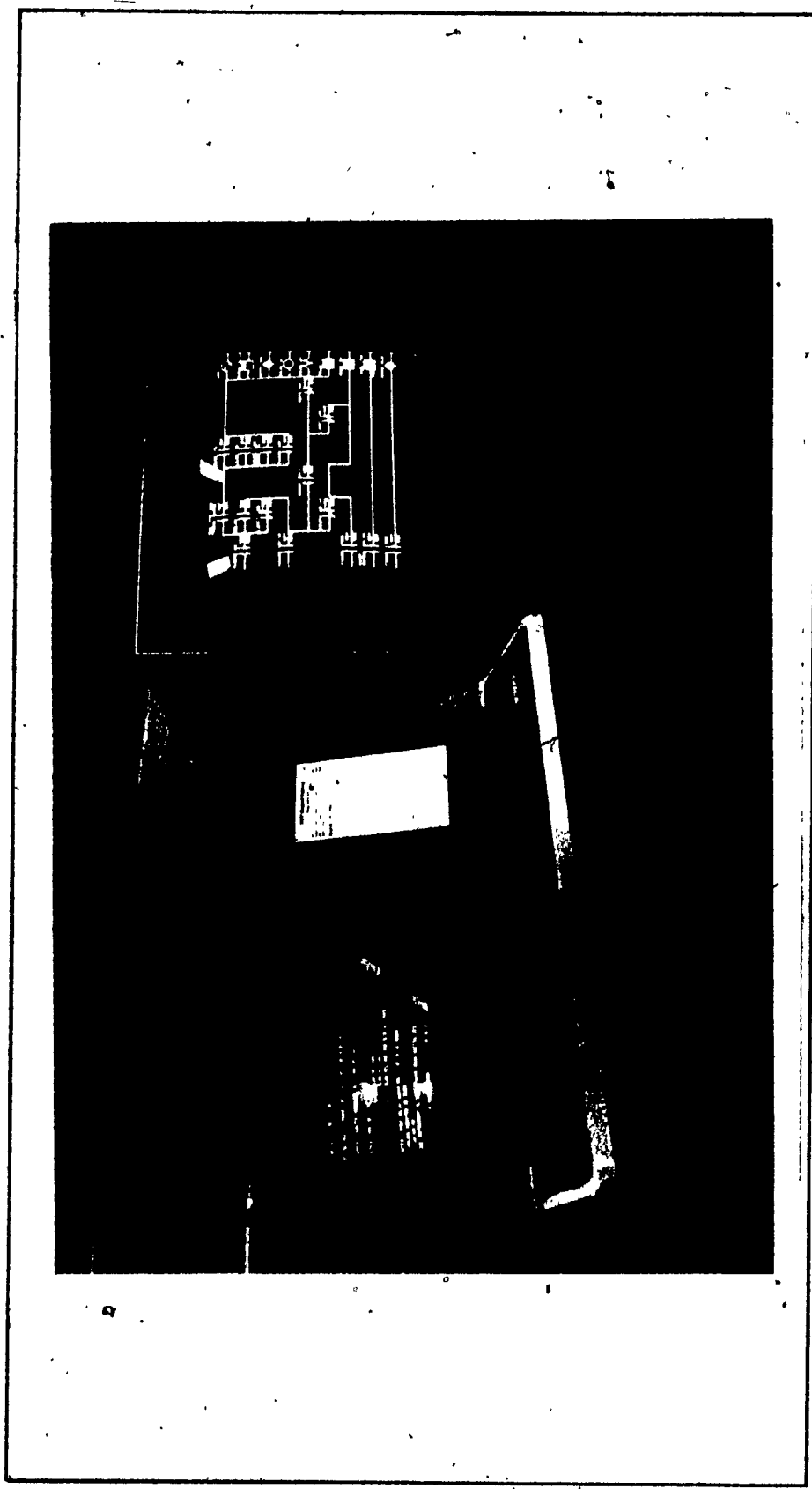


Fig.5.2 Typical work station.

The CSP⁰ is written almost entirely in VAX VMS FORTRAN, with the exception of a group of routines for console input which were written in VAX Assembly Language [41]. In order that the source code of the CSP be compatible with other versions of FORTRAN, many of the advanced features of the available compiler were not used. Hence, much of the source code can be transferred to any microcomputer capable of supporting FORTRAN 80 with only minor changes. Segments of the CSP which handle inputs, outputs, and graphics, however, are not easily transferrable from one machine to another as these often require machine- and operating system-dependent coding.

5.3 Description of the CSP

5.3.1 Executive Mode

The Executive Mode allows the operator to prepare a CIRCUIT for simulation. This includes: editing a Ladder Diagram; storing a Ladder Diagram on mass storage; loading a Ladder Diagram from mass storage; and specifying the default values of the pull-in, and drop-out, speeds of the relay devices. The Executive Mode also enables the operator to create a Ladder Diagram and to submit for simulation, the Ladder Diagram currently in the memory. The command menu used in the Executive Mode is given in Fig.5.3.

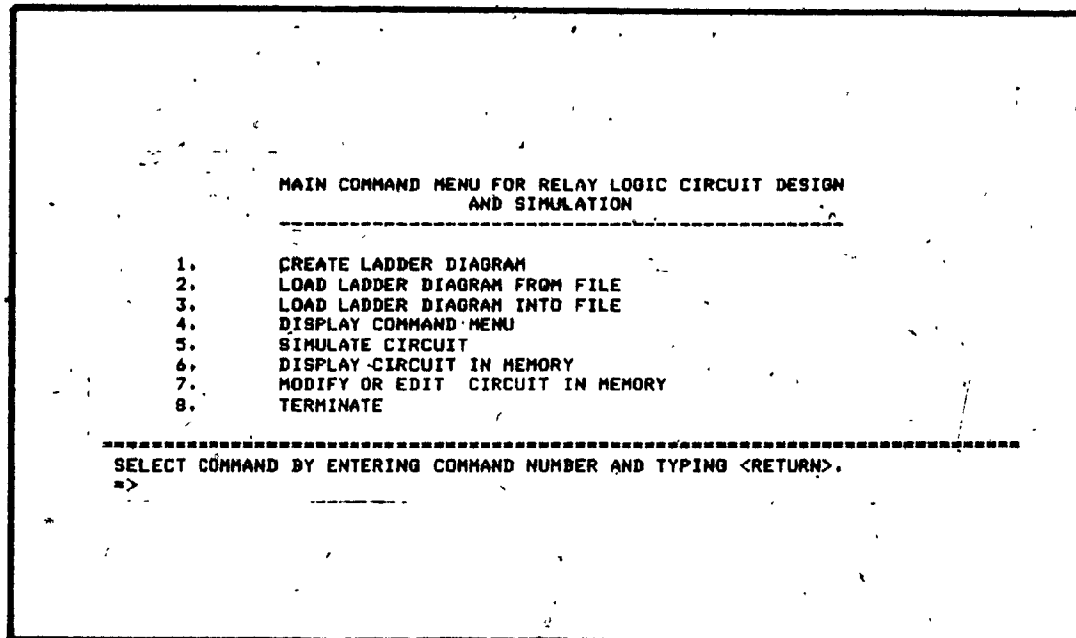


Fig.5.3 Command menu.

5.3.2 Configuration Mode

The Configuration Mode of the CSP makes use of interactive computer graphics to allow the user to enter a CIRCUIT in the form of a Ladder Diagram. The size and resolution of the graphics screen require that the width of Ladder Diagrams implemented by the CSP be limited to 7 columns. The amount of memory available limits the length of these Ladder Diagrams to 100 rows.

Output devices (lamps, buzzers, valves, motors, solenoids, etc.), and memory device coils (relays, delay relays, timers and counters) can only be placed in column number 7 of the Ladder Diagram. Input devices (hand, pressure, temperature, limit, and level switch contact pairs), memory device contact pairs, and vertical and horizontal connectors can only be placed in the first 6 columns. Only vertical connectors can be used to connect one row to the next, and no more than 7 consecutive rows can be connected to one another. This last constraint is a direct consequence of the limitation of the size of the model to 49 nodes (Section 3.3.2).

A total of 12 dedicated function keys let the operator select the desired function. Prompting, displayed on the data terminal (non-graphic terminal) from the CSP guides the operator through the appropriate input sequence. Figure 5.4 shows the designation of the function keys. The functions these keys provide can be divided into two






SHIFT			PAGE UP
			UP
	DELETE VERTICAL		PAGE DOWN
			DOWN
			RIGHT
			LEFT
		EXIT MODE	
			DEVICE DELETE

Fig.5.4 Designation of function keys for Configuration Mode of CSP.

categories: cursor movement; and device specification.

Four cursor movement keys allow the operator to move the cursor, one step at a time, in any direction; up, down, left, and right. It is also possible for the operator to move the cursor up or down, one "page" at a time by depressing the "SHIFT" key before the "UP" or "DOWN" keys.

As it is not possible to display the entire length of the Ladder Diagram on the graphics screen, an arrangement whereby the screen is used as a "window" which the operator can position over any 9 consecutive rows is used. As the cursor moves up or down, so does the window. Furthermore, the cursor is kept in the middle third of the window so that the operator may have a better idea of where the cursor is situated in the Ladder Diagram.

There is a total of 6 device specification function keys which the operator can use to enter or delete a device from the current cursor position. When the function key to enter a device other than a connector is depressed, a numbered list of input devices and memory contact pairs, or output devices and memory coils, is displayed on the data terminal, as shown in Fig.5.5. The operator can select the desired device by typing the number identifying it on the list.

Once a device has been selected, a series of prompts guides the user to complete the input sequence. The first prompt is for the entry of the "Device Identification Code". This can be any 5 character, alphanumeric string and is used

LIST OF AVAILABLE INPUT ELEMENTS

- | | | | |
|----|-------|-----|-------|
| 1. | HANDS | 2. | PUSHB |
| 3. | TEMP9 | 4. | LEVEL |
| 5. | LIMIT | 6. | PRESS |
| 7. | TIMER | 8. | COUNT |
| 9. | RELAY | 10. | DRELY |

ENTER NUMBER OF ELEMENT TYPE DESIRED.
=>

LIST OF AVAILABLE OUTPUT ELEMENTS

- | | | | |
|----|-------|----|-------|
| 1. | LAMP | 2. | BUZZR |
| 3. | MOTOR | 4. | VALVE |
| 5. | SOLE | 6. | TIMER |
| 7. | COUNT | 8. | RELAY |
| 9. | DRELY | | |

ENTER NUMBER OF ELEMENT TYPE DESIRED.
=>

Fig.5.5 Numbered lists of devices.

to distinguish between devices having the same name. By the same token, memory coils and the contact pairs associated with them will be identified by the same name and identification number, as will the various contact pairs of a given switch.

If an output device or memory device coil is entered, the entries described above complete the input sequence. For an input device or memory contact pair however, the contact pair number must also be entered. The prompt for the contact pair number will specify the maximum number of contact pairs that devices having the same name can accommodate. Once a valid entry has been made, the input sequence is terminated and the cursor is automatically positioned at the next device position.

Horizontal and vertical connectors are drawn on the graphic screen as the appropriate function key is depressed. When a vertical connector is drawn however, the cursor is not moved in order to allow for the possible subsequent entry of a device at the same cursor position. (Vertical connectors are drawn upward from the west-most side of the cursor to join the row above.)

Devices can be deleted from the Ladder Diagram simply by positioning the cursor over the device to be deleted and depressing the "DEVICE DELETE" function key. To delete a vertical connector, a separate key, "DEL VERT", is provided in order to provide for deleting a vertical connector without disturbing the device at that cursor position.

Figure 5.6 shows the sequence of CSP prompts and operator inputs required to create the simple Ladder Diagram shown in Fig.5.7. A block diagram of the algorithm used to implement the configuration facilities is given in Fig.5.8.

5.3.3 Simulation Mode

In the Simulation Mode, the operator is permitted to interactively position the cursor over any input device, toggle its state, and immediately observe the response of the CIRCUIT on the graphics screen. Furthermore, it is possible to simulate the timing characteristics of a CIRCUIT containing memory devices.

As with the Configuration Mode, the dedicated keypad permits operator to enter commands to the CSP. For the Simulation Mode, the keys are designated as shown in Fig.5.9. There are 13 keys providing a total of 19 functions. The keys for positioning the cursor, and exiting the Simulation Mode are the same as for the Configuration Mode. The remainder of the keys select specific simulation functions. The main functions of the Simulation Mode are:

- interactive specification of input states
- advancing the CIRCUIT in time
- specification of fault conditions

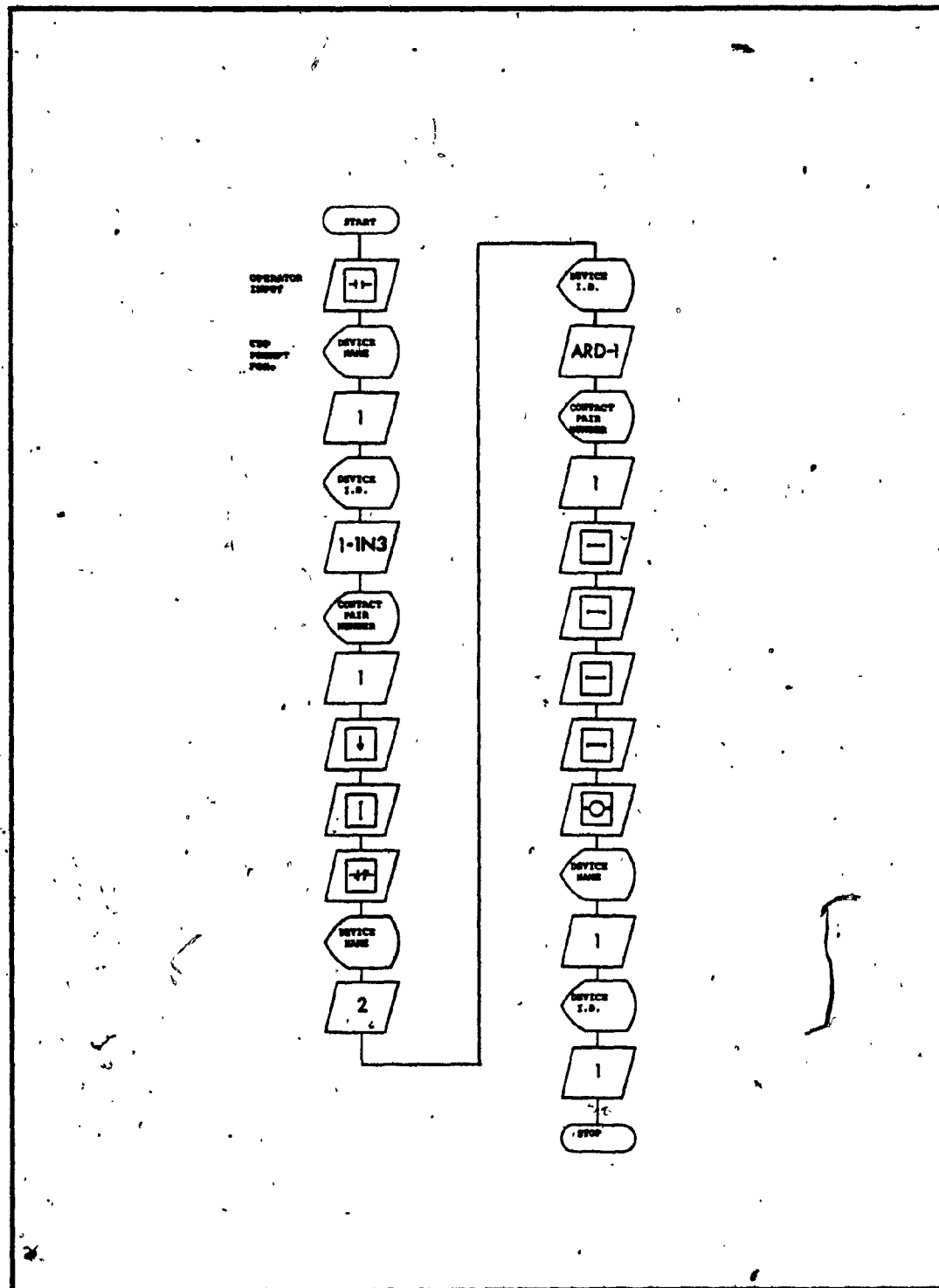


Fig.5.6 Sequence of CSP prompts.

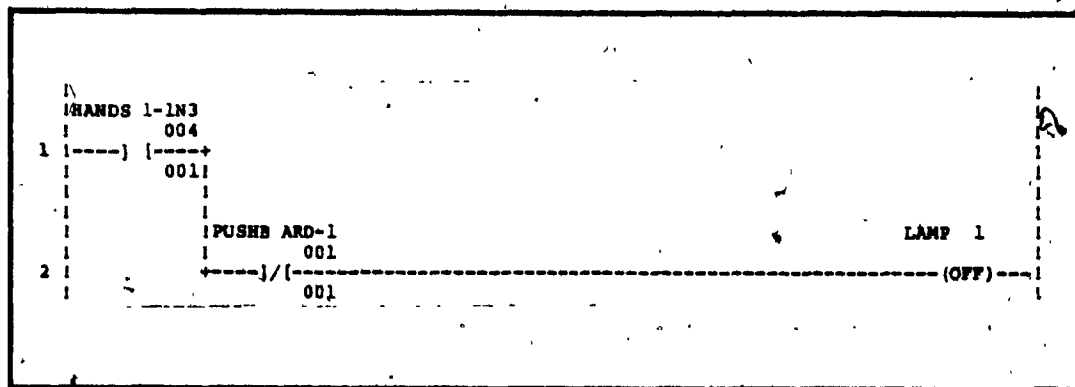


Fig.5.7 CIRCUIT entered by sequence shown in Fig.5.6.

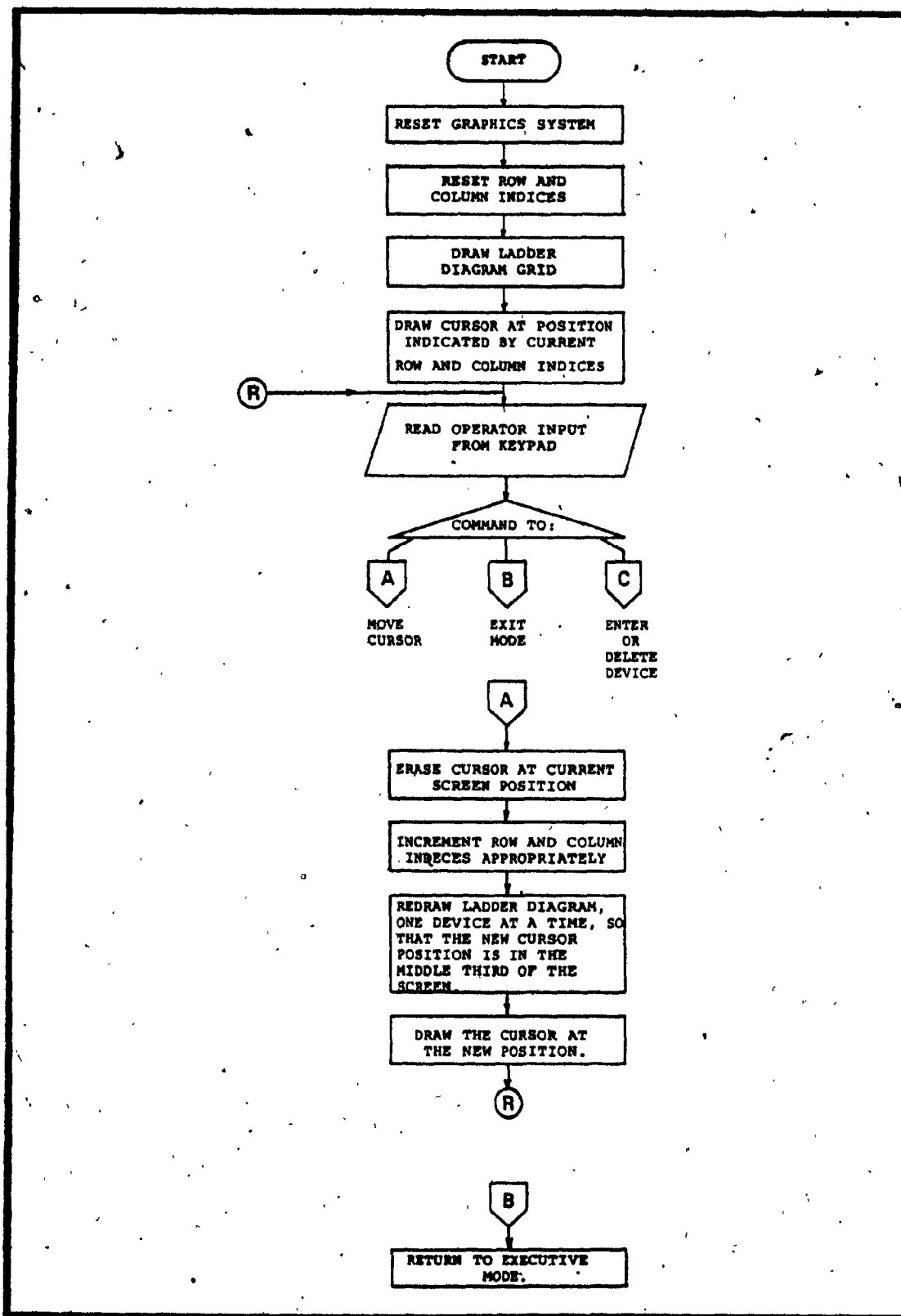


Fig.5.8 Block diagram of algorithm used to implement configuration facilities ... (cont'd).

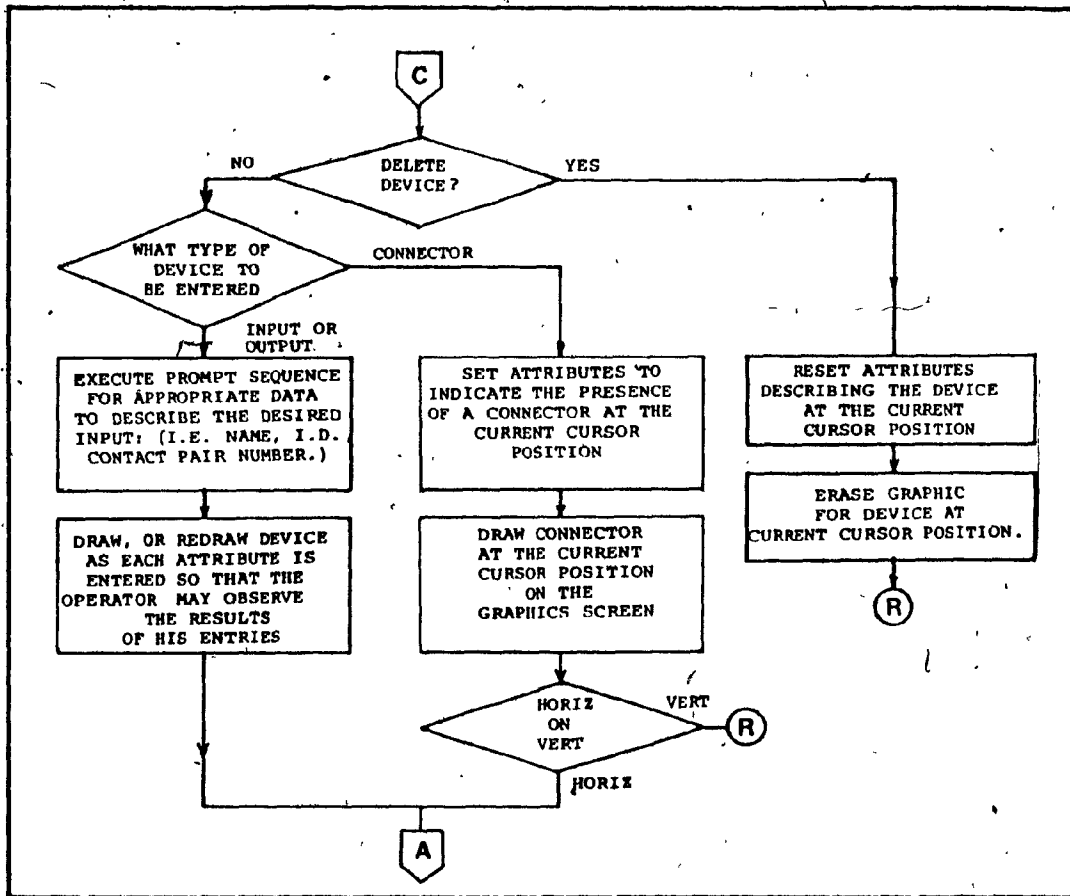


Fig.5.8 ... (cont'd) Block diagram of algorithm used to implement configuration facilities.

SHIFT	RELAY CLOCK ADVANCE		PAGE UP
			UP
STEADY TRANSIENT		HARD COPY OUTPUT	PAGE DOWN
			DOWN
			RIGHT
DELETE ANNUN. MESSAGES ENTER ANNUN. MESSAGES		RESET FAULT	
		SET FAULT	LEFT
TIMER CLOCK ADVANCE		EXIT MODE	
TOGGLE		RESET COUNTER	

Fig.5.9 Designation of Function keys for Simulation Mode.

- specification of annunciation messages.

When the CSP is placed in the Simulation Mode, all input devices and memory device contact pairs are reset to their quiescent states. Once this process has been completed, the operator can set the state of any input device. To do this, he must position the cursor over the device and depress the "TOGGLE DEVICE" key. Hence, this and every other device having the same name and identification number, will go from a "quiescent" to an "energised" state, or vice-versa. The change in state is accompanied by a change in the graphic representations of these devices.

5.3.3.1 Simulation of Timing

For each change in the input or internal state, the CIRCUIT is analysed using the method described in Chapter 3. In this manner, the output state is determined, the current-carrying devices are identified, and a color graphic display is generated. Also in Chapter 3, a method for simulating the timing of a CIRCUIT using 2 simulation clocks was described. In order to implement this method, it was necessary to define and classify the transient states of a CIRCUIT.

For a given set of input and internal states, each of the memory devices in a CIRCUIT will be found to be in either a "transient", or in a "steady" state. A memory

device is said to be in a transient state if the excitation on the coil is different from that on the contact pairs associated with it. There are two types of memory devices: those which require the passage of time in order to reach a steady state (i.e. relays, delay relays, and timers); and those which count events (i.e. counters). For relays, the duration of the "pull-in" and the "drop-out" speeds are given default values of 5 and 3 milliseconds respectively, by the CSP. These can be changed by the operator before a simulation is initiated. The types ("SLOW ACTIVATE", "SLOW RELEASE" and "SLOW ACTIVATE AND SLOW RELEASE") and durations of the delay relays and timers, as well as the counter moduli, are entered in response to prompts from the CSP when the simulation mode is entered.

The counter is a device which is capable of counting the number of off-to-on transitions experienced by the counter coil. When this count becomes equal to the preset modulus, the contact pairs associated with that coil, (i.e. contact pairs having the same name and identification number), will become excited. This characteristic is exhibited by the CIRCUIT in Fig. 5.10.

Counters, as defined for the purposes of the simulation, possess the peculiar characteristic that their operation is independent of simulated time. While the simulation clocks remain fixed in time, the counter experiences a number of off-to-on transitions which are taken to be simultaneous. After the preset number of

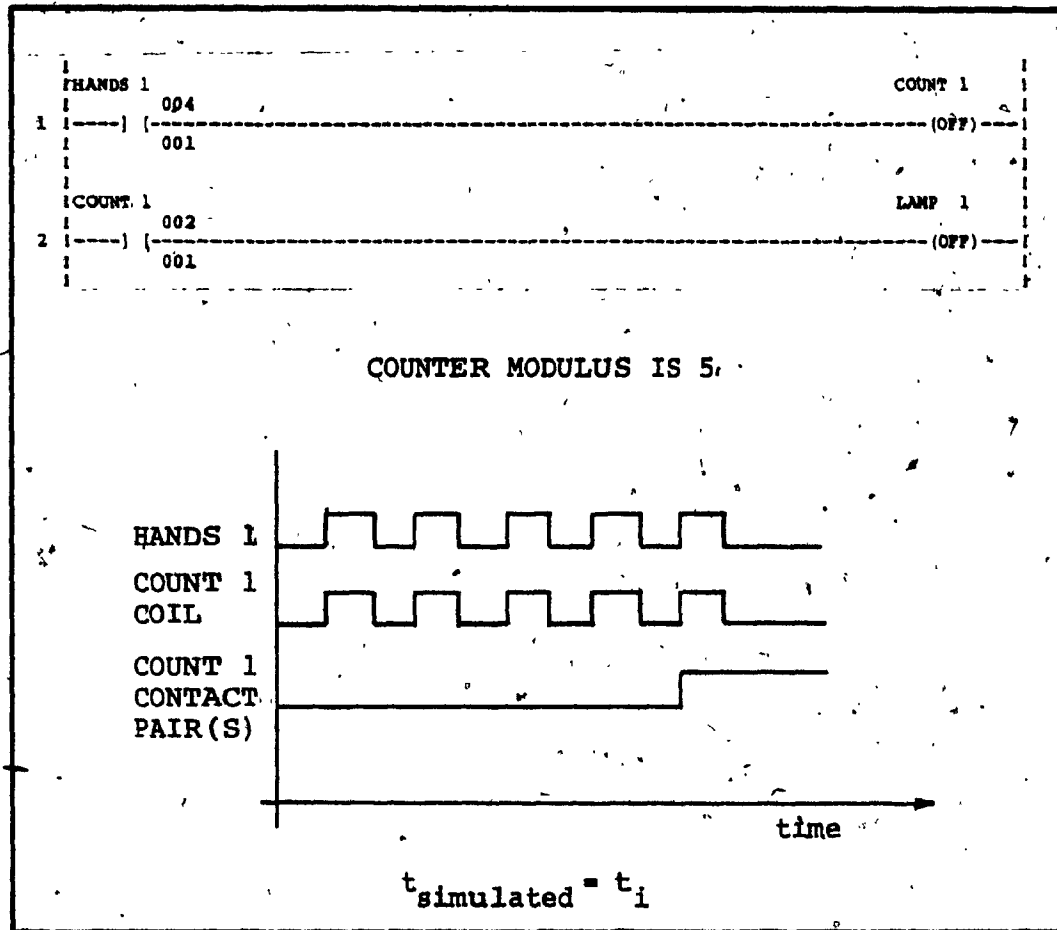


Fig.5.10 Characteristic of counter device.

off-to-on transitions on the counter coil have been counted, the preset count is satisfied and the counter reaches a steady state; this occurring simultaneously, in simulated time, with the transitions on the counter coil. Obviously, this is not possible on a real process. It is provided however, to allow the operator to quickly increment the counter without disturbing the other transient states. The counter does not involve any time dependent characteristics and therefore will not be considered in the following discussion.

When a CIRCUIT comprises memory devices, other than counters, which are in transient states, the state of the CIRCUIT will also be a transient state. The transient states of CIRCUITS can be categorized as: "timer transient", "relay transient", and "timer and relay transient", and "steady". A CIRCUIT is said to be in a timer transient state if it contains at least one timer which is in a transient state, and when all relays and delay relays in it are in a steady state. It is said to be in a relay transient state if it comprises at least one relay, or one delay relay which is in a transient state with all timers in it being in a steady state. Finally, a CIRCUIT is said to be in a timer and relay transient state if it comprises at least one timer and one relay or delay relay which are in transient states. A CIRCUIT is said to be in a steady state otherwise.

In order to simulate the timing of a CIRCUIT, two

independent simulation clocks are defined. One is incremented in steps of 1 millisecond and is hereafter referred to as the Relay Clock. The second clock, or Timer Clock, can be incremented by specifying the amount of time advance in terms of hours, minutes, and seconds.

By depressing the "RELAY CLOCK ADVANCE" function key, the operator can advance the Relay Clock, and the time step counter of every relay and delay relay in a transient state. In similar fashion, the Timer Clock can be advanced and every timer in a transient state can be incremented using the "TIMER CLOCK ADVANCE", function key.

When the CIRCUIT is in a steady state, both the Timer Clock as well as the Relay Clock are reset. Upon entry into one of the transient states from a steady state, one, or both simulation clocks (depending on the type of transient state) will be initialized. Thereafter, the operator must explicitly advance the CIRCUIT in time in the manner described above. To prevent the operator from advancing the Timer Clock when relays are in transient states, the "TIMER ADVANCE" feature is disabled whenever the CIRCUIT is in a timer and relay transient state. Furthermore, the Timer Clock can only be advanced by an amount less than or equal to the time required for a timer device to change from a transient to a steady state.

Using the "STEADY/TRANSIENT" function key, the operator may select between two types of timing simulation. The "STEADY" option automatically advances the Relay Clock

until all relay and delay relays in the CIRCUIT have reached a steady state. The "TRANSIENT" option requires that the operator manually advance the Relay Clock, while the CIRCUIT and simulation diagnostics are displayed for each transient state. At any time during the simulation, the operator may toggle from one option to the other by depressing the "STEADY/TRANSIENT" function key. Figure 5.11 shows a flowchart for the operation of the CSP in Simulation Mode.

5.3.3.2 Documentation

An important documentation feature of the CSP is that it provides for the display of user defined annunciation messages during the course of the simulation. These messages convey noteworthy information about a particular device directly on the Ladder Diagram. For example, the contact pairs of a pressure switch may have the messages "PRESS HIGH" "PRESS. LOW" and "FAULT. PRESS" associated with them. Only the appropriate message for a particular state of the device will be displayed at any given time.

An annunciation message can be entered at any time during the simulation by placing the cursor over the device for which a message is to be entered, and by depressing the "ENTER ANNUN. MESSAGE" function key. If the device is an input device, the CSP will prompt the operator for messages to accompany the device for each possible operating condition; energised, quiescent, or fault. For output

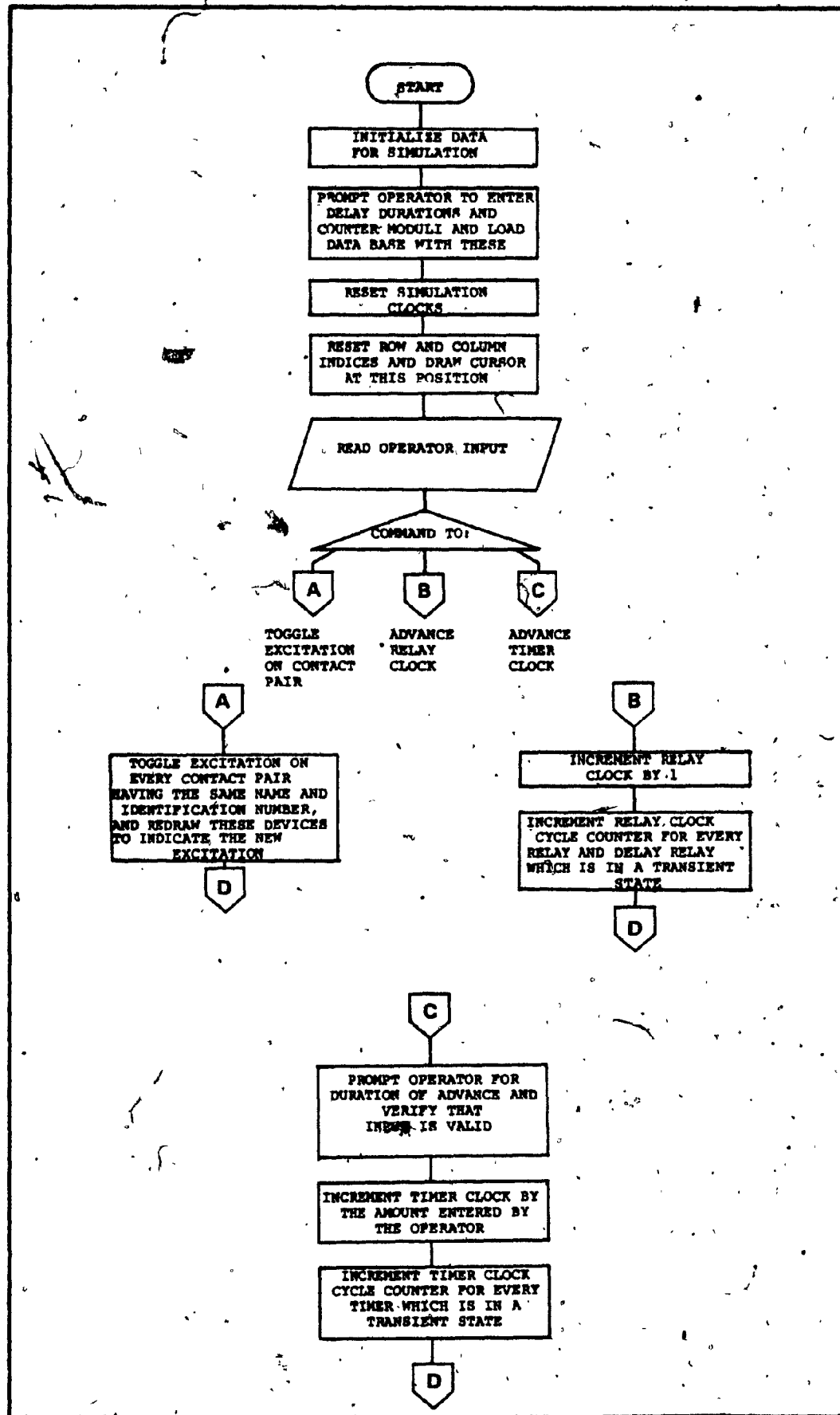


Fig.5.11 Flowchart for CSP Simulation Mode ... (cont'd)

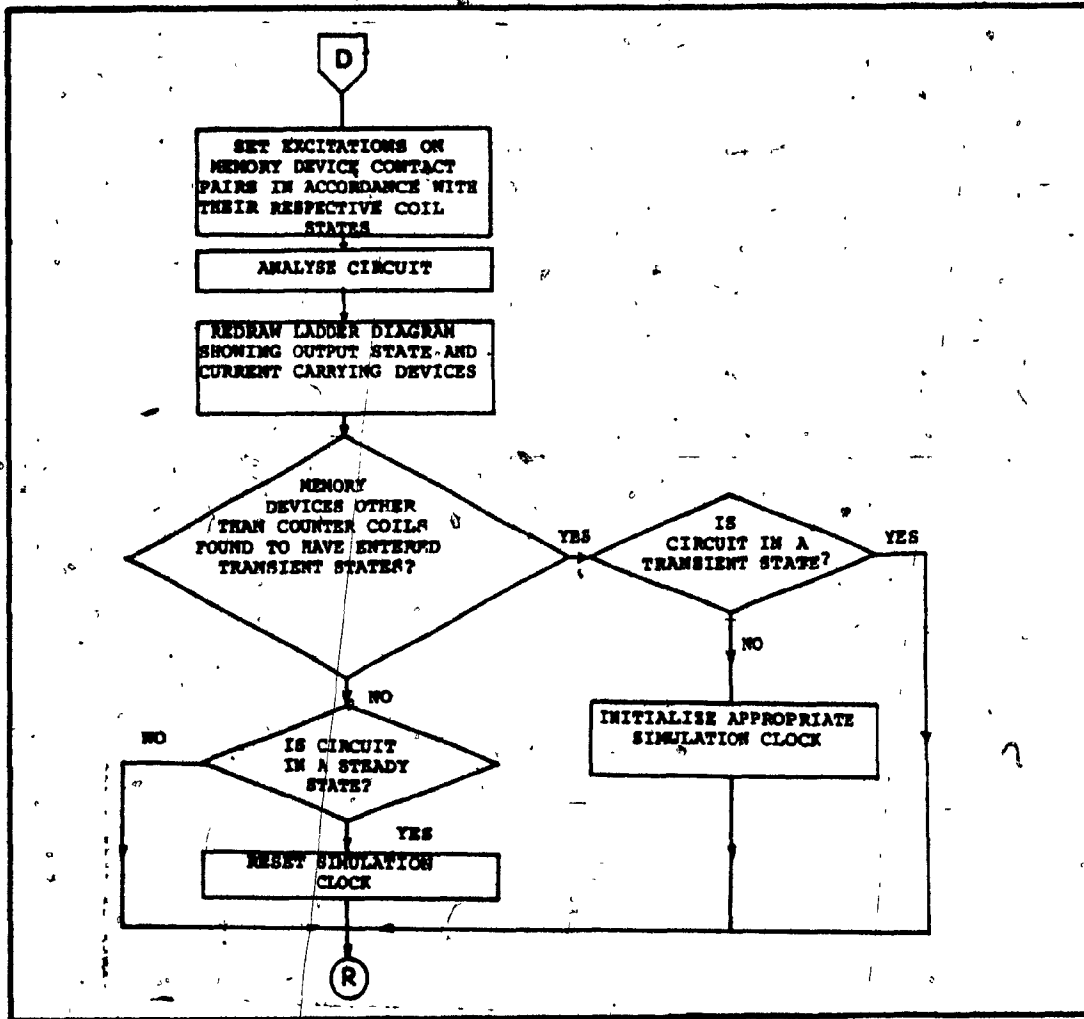


Fig.5.11 (...CONT'D) Flowchart for CSP Simulation Mode.

devices, there are four messages; one each for energised, off, fault, and transient operating conditions. Once a set of annunciation messages has been entered for a device, the same set of messages will automatically be associated with every other device with the same name and identification code.

5.3.3.3 Fault Conditions

Often, the result of the failure of a device, or number of devices, on the operation of the CIRCUIT must be considered during testing. To allow the operator to simulate the failure of a device, or to "force" a device to a desired state, the CSP provides for the interactive assignment of fault conditions to any device, at any time during the simulation. Each class of devices has its own distinct failure modes.

Input devices, be they direct inputs or contact pairs of memory devices, can fail in one of two modes; "fail excited", or "fail quiescent". Table 5.1 shows that a failed input device, or memory contact pair, will have its transmission value set by the specified fault condition rather than by the excitation acting upon it.

Output devices also have two failure modes; "fail off" and "fail energised". Table 5.2 shows that the state of a failed output device will depend on the fault mode, rather than on the transmission value of the contact network

connected between the voltage source and that device.

Memory devices have three failure modes; "fail off", "fail energised", and "fail transient". The first two failure modes are much the same as those for output devices, except that they define the excitations of contact pairs. The third failure mode will prevent the contact pairs associated with a faulty device from changing when the conditions for such a change are satisfied.

To specify a fault, the operator simply positions the cursor over a device and depresses the "SET FAULT" key on the keypad. In response, the CSP will prompt the operator for the entry of the desired fault mode. Once the entry is completed, the display is revised to indicate that there is a fault, and the CIRCUIT is analysed and displayed to indicate the input, output, and internal state, as well as the current-carrying devices of the CIRCUIT.

5.4 Data Structure

The data base of the CSP can be divided into four data groups: General Data; Element Data; Look-up Data; and Graphics Data.

The General Data group comprises two data files. One file contains the Command Menu which is displayed when the CSP is operating in the Executive Mode. Since this file is displayed in its entirety only occasionally, it is both convenient and economical (in terms of memory usage) to

store this data in a sequential access disk file. The second file contains the error messages for the CSP. By storing these in a direct access disk file, a considerable (about 1K bytes) saving in memory is achieved.

Data describing each device in the Ladder Diagram is contained in the Element Data group. Each device position is allotted either 9, or 7 bytes of core memory, depending on whether it is in the 7th column or not.

The data stored defines a number of "attributes" for each device. Each attribute can be classified as being either two-valued or coded. Two-valued attributes require only a single bit in memory for each device. Coded attributes require several bits of memory each and must be coded for storage and decoded for use by the CSP. The various device attributes are tabulated in Appendix D. Of these, the annunciation messages form an excessively large block of data (33 bytes for each input device, and 44 bytes for each output device), which is stored in a direct access disk file in order to conserve memory.

The 5-bit coded attribute designated the "Device Name Code", in the Element Data group, identifies a device as being one of the devices named in Table 5.3. Each name has, associated with it, certain attributes which are common to every device having that name, that is, maximum number of contact pairs, and the address of the graphics data base, irrespective of its position in the Ladder Diagram. The Look-up Data group stores this data in an ordered fashion so

TYPE	FAILURE MODE	TRANSMISSION
NO	QUIESCENT	0
NO	EXCITED	1
NC	QUIESCENT	1
NC	EXCITED	0

Table 5.1 Failed states of input devices.

FAILURE MODE	DEVICE STATE
ENERGISED	ENERGISED
OFF	OFF
TRANSIENT	TRANSIENT

Table 5.2 Failed states of output devices and memories.

INPUT			
HAND SWITCH	HANDS	TEMPERATURE SWITCH	TEMPS
LEVEL SWITCH	LEVEL	PUSHBUTTON	PUSHB
LIMIT SWITCH	LIMIT	PRESSURE SWITCH	PRESS
OUTPUT			
LAMPS	LAMP	BUZZER	BUZZER
VALVE	VALVE	MOTOR	MOTOR
SOLENOID	SOLENOID		
MEMORY			
TIMER	TIMER	RELAY	RELAY
COUNTER	COUNT	DELAY RELAY	DELAY

Table 5.3 Designation of device names.

that the "Device Name Code" will address the memory location where this information is stored for each device name.

Finally, the Graphics Data group comprises the graphics data bases used to draw the symbols for the devices in the Ladder Diagram. The graphics data base address stored in the Look-up Data group points to the data required to draw that device. Fig.5.12 illustrates the data structure for the CSP.

5.5 Graphics and Display

5.5.1 Introduction

One of the most important features of the CSP is its use of interactive, color, computer graphics to display information. Where PLCs typically use alphanumeric characters, or miniature graphic symbols, the CSP presents the graphic symbol for each device within a rectangular portion of the screen 2.54 cm high and 4.5 cm wide. These oversized graphics allow the operator to work at a comfortable distance from the graphics screen. Furthermore, they allow for the presentation of simulation diagnostics and documentation on the graphics screen, accompanying each device graphic.

In addition to the information displayed on the graphics screen, the data terminal is used to echo some operator inputs, to display prompts from the CSP, and to

indicate the elapsed time on the two simulation clocks. With this arrangement, the operator has visual feedback whether his input causes a change in the graphic display, or is simply a response to a CSP prompt. Hence, eye strain is reduced, and entries can be easily verified.

In the Configuration Mode, only the Ladder Diagram and the information identifying each device is monochromatically displayed on the graphics screen. This same information is also displayed in the Simulation Mode. In the latter mode however, color is used extensively to highlight simulation data pertaining to current-carrying devices, as well as fault, and output states.

5.5.2 Display Element

Irrespective of the operating mode of the CSP, the graphic display of the Ladder Diagram is created one device at a time. The display for each device is contained within a rectangular display area, or display element. There are 63 such display elements per page, one for each Ladder Diagram grid location. (There are 9 rows of 7 columns per page.) Any display element can be constructed on the basis of one of four basic structures: blank; contact pair; output; and memory coil.

As the name implies, the first of these structures serves to indicate the absence of a device. It contains no documentation, and is represented by a rectangle drawn in

the colour of the background (black).

The second structure is used to represent contact pairs of switches or of memory devices. The general contact pair structure is shown in Fig.5.13a. The top of the display element is reserved for the 11 character annunciation message which corresponds to the device state. The area directly below is designated to hold the device name and identification number. The balance of the display element contains the graphic for the device, and is interspersed with documentation pertaining to the contact pair number and the fault conditions (when they apply).

A contact pair graphic drawn in red indicates that the device is carrying current. Blue indicates that it is not carrying current, and purple indicates that a fault has been imposed. At the upper left of the graphic, a 3-character code describes the nature of the fault, (FNO for fail quiescent, and FNN for fail excited). The maximum number of contact pairs which can belong to the same device is indicated at the upper right of the graphic, and the contact pair number of the device is given at the lower right.

Output devices are drawn on the basis of the display element structure shown in Fig.5.13b. As with the contact pair structure, the annunciation message, device name, and identification code are displayed in the upper portion of the display element. The graphic representing the device is drawn in the lower portion, along with the 3-character code

describing the fault conditions (FOF for fail off; FEN for fail energised).

Unlike the graphic symbols for contact pairs and memory device coils which are fixed by the CSP, the default graphics for output devices shown in Fig.5.14 can be modified, or completely changed by the user. This can be done by modifying the data file containing the graphics data base.

In the display of an output device, color is used to indicate its state; red when energised, blue when off, and purple when a fault is imposed.

The last structure to be considered, is that which forms the basis for the display of memory coils. These are set apart from other output devices because they are capable of entering transient states which must also be represented on the graphic screen. An indication of how near the device is to a steady state, in terms of simulated time for relays and timers and in terms of registered count for counters, is also required.

As with the two previous structures, the annunciation message, device name, and identification code is placed in the upper portion. The graphic symbol for a memory coil is displayed in the lower portion. Three fault conditions are also displayed: the elapsed time or registered count, in terms of the percentage of the delay duration or counter modulus, and a bar graph display of this quantity. To the two fault conditions for output devices, FPR for fail

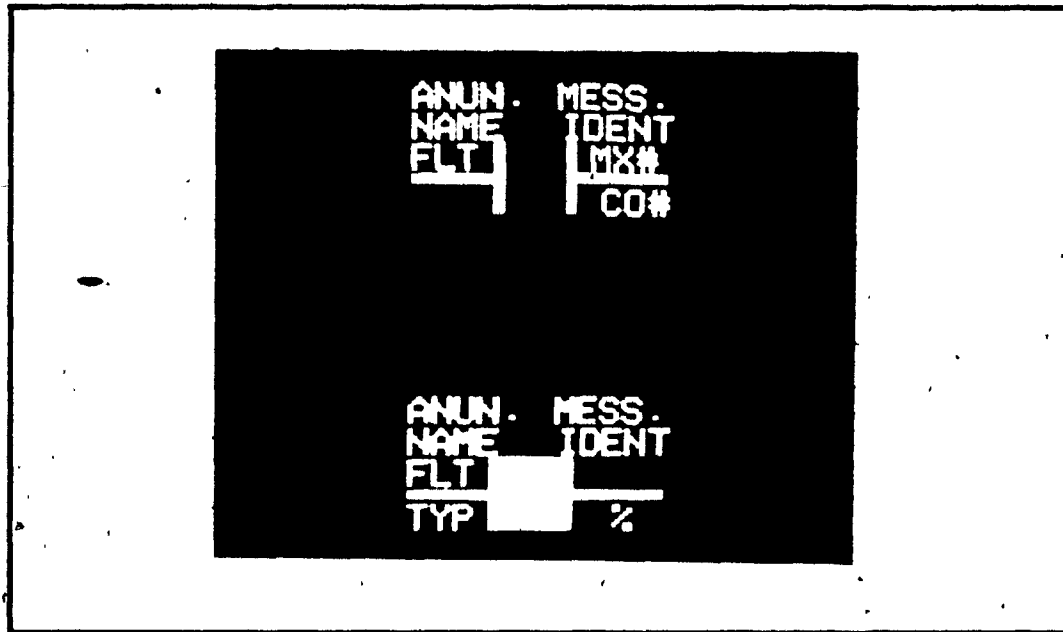


Fig.5.13 (a)General contact pair graphic structure (upper).
(b)General output device graphic structure (lower).

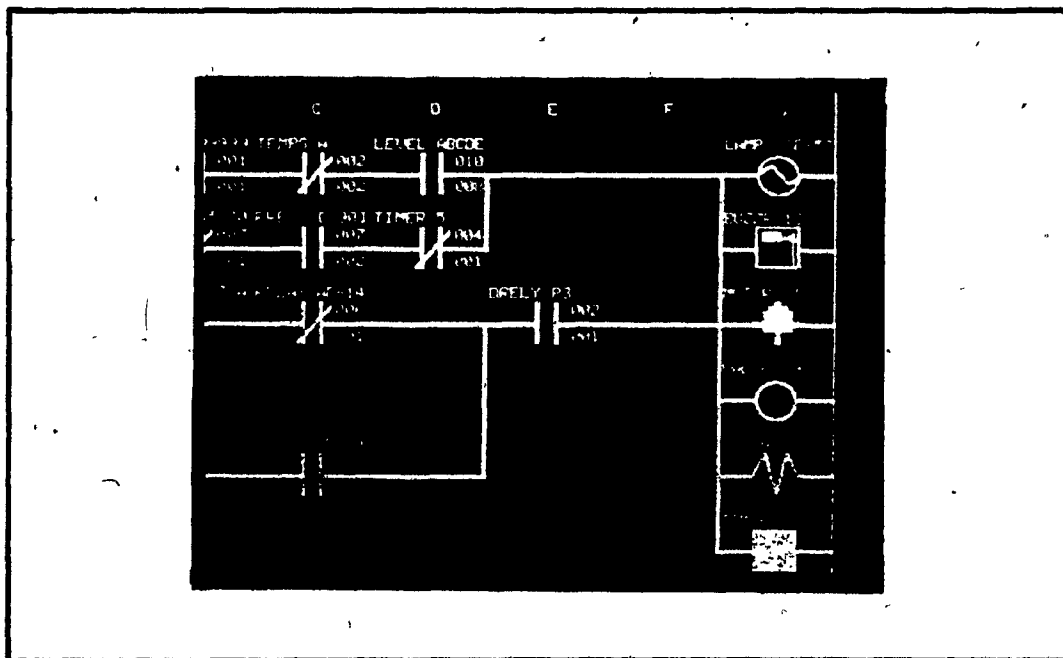


Fig.5.14 Default graphics for output devices (column G).

transient is added.

Red, blue, and purple indicate energised, off, and fault states of the memory device, respectively. A memory device drawn in a dark red colour is used to indicate a transient state. Within the body of the graphic, a rectangular window is filled by a bright red bar as the device approaches a steady state. The ratio of the window area filled, to the total window area, represents the percentage of time elapsed or count registered.

The algorithm for producing each display element involves decoding the appropriate data and displaying it on the graphics screen. This process is flowcharted in Fig.5.15. In the Configuration Mode, specific segments of this procedure are executed to display data as it is entered. Both the Configuration Mode, and the Simulation Mode, call upon this procedure repeatedly to either refresh, or to produce anew, the screen display. A sample, Simulation Mode display is shown in Fig.5.16.

5.6 Implementation of Simulation Program on an 8-bit Microcomputer

Although the simulation program has been developed with the microcomputer implementation in mind, several modifications were required in order to obtain a similar simulation program which would run on an 8-bit microcomputer with a 64k byte memory. These modifications can be

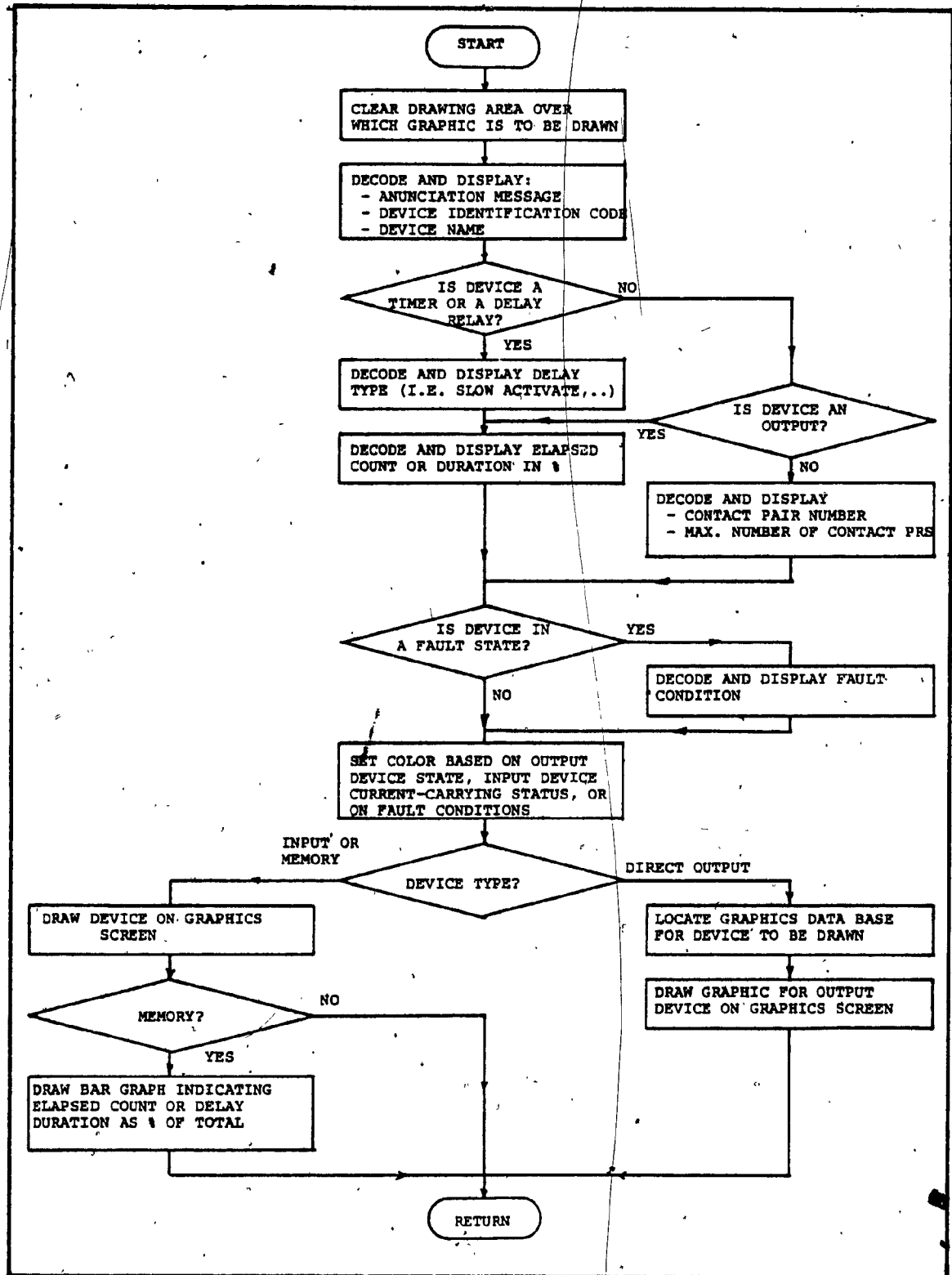


Fig.5.15 Flowchart for graphic element display.

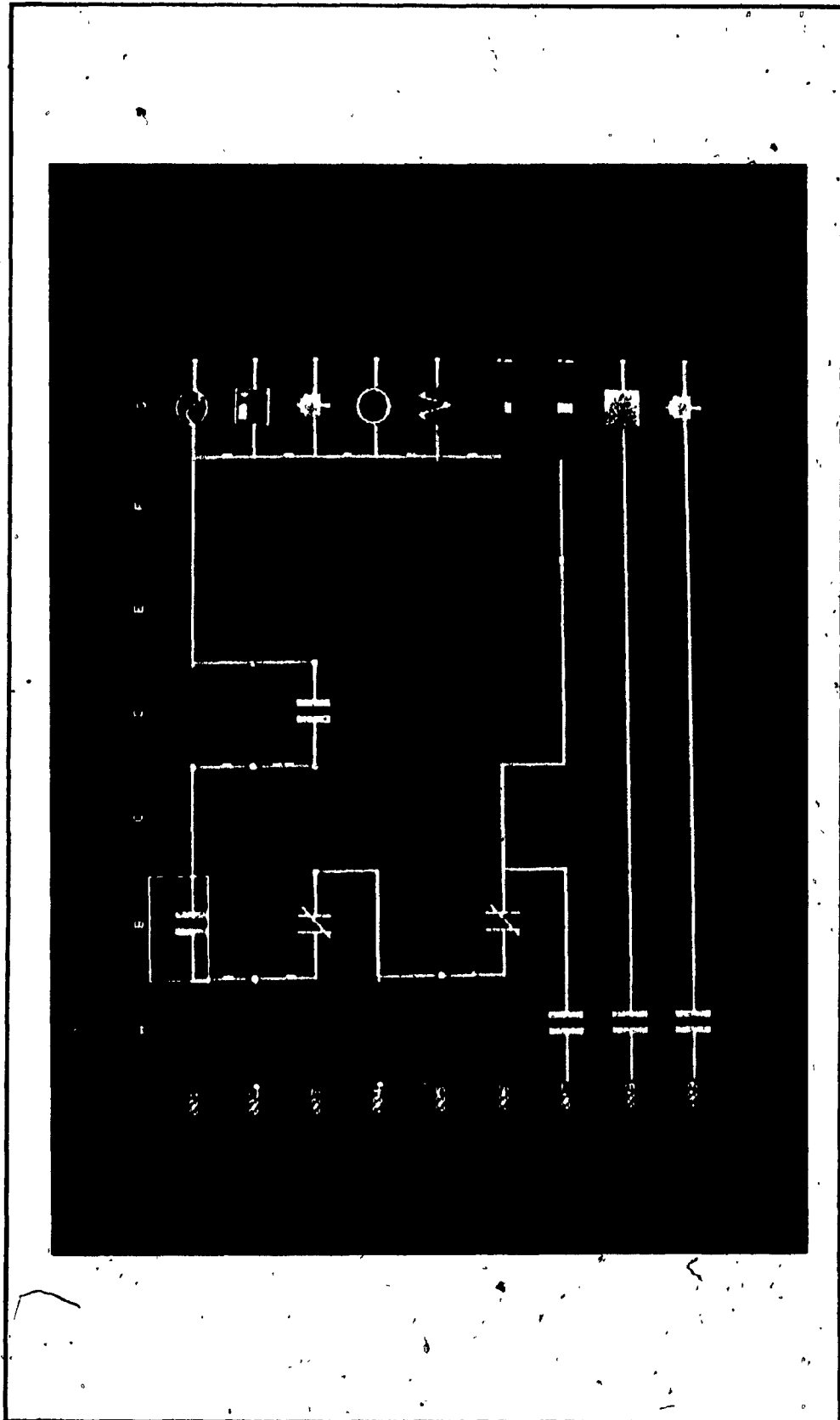


Fig.5.16 Sample CSP Simulation Mode Display of Ladder Diagram.

classified under three headings: memory usage, coding, and graphics.

5.6.1 Memory Usage

The greatest difficulty in implementing the simulation program on a microcomputer was that the executable code requires approximately 80k bytes of memory; about twice as much as is available. To overcome this difficulty, the program was divided into smaller, independent segments which are "chained" to one another as the operation of the simulation program requires it. Each chaining operation takes about 20 seconds. In the course of a typical simulation, a chaining operation has to be executed each time a CIRCUIT is analyzed. Hence, an undesirably slow operation is the best that can be obtained. The use of a machine having a larger memory, 128k bytes for example, would alleviate this problem.

5.6.2 Coding

As was mentioned previously, considerable care was taken in coding the CSP using only the features of FORTRAN which are common to both the F80 as well as the VMS versions of this language. The same is not possible in the case of the graphics routines which are specific to the particular hardware on which they are to be implemented.

In transferring the CSP software from the VAX to the microcomputer, all the I/O routines had to be modified. Furthermore, since the graphics display screen on the microcomputer is monochromatic, the graphic display of the simulation diagnostics had to be modified.

5.6.3 Graphics

A NORTHSTAR ADVANTAGE microcomputer, was used as the development system for the microcomputer version of the CSP. The GRAPHICS CP/M operating system supplied with the machine provides a number of graphic subroutines written in Z80 assembly language. A library of FORTRAN callable graphics subroutines was written to provide the basic graphics primitives required to generate the display for the simulation program.

In order to display the same information on a monochromatic screen as is displayed on the color graphics screen, the area fill pattern generator provided by the GRAPHIC CP/M was used extensively.

5.7 Summary

The CSP allows the user to enter a CIRCUIT in the form of a Ladder Diagram and to simulate its logic and timing characteristics interactively, on a color graphics screen. By analysing the CIRCUIT using the method described

in Chapter 3, the CSP ensures the correct determination of the output state.

Interactive color graphics are used extensively to display the various simulation outputs including the output state, current paths, and documentation. During the course of either interactive CIRCUIT input or simulation, menus and descriptive prompts guide the experienced or the novice user, offering a user-friendly operating environment.

Work is in progress on the implementation of the features of the minicomputer version of the CSP onto an 8-bit microcomputer. The necessity for machine-dependent coding required by the graphics, and the memory constraints imposed by the 8-bit microcomputer however, have made this a difficult task.

CHAPTER 6

APPLICATION OF CSP TO THE SIMULATION OF TEST CIRCUITS

6.1 Introduction

A completely operational version of the CSP, as described in Chapter 5, is presently available on the VAX 11/780 of the Computer Research and Interactive Graphics Laboratory (CRIGL) at Concordia University. It provides all the features necessary for completely simulating the logic and timing characteristics of CIRCUITS interactively on a color graphics screen.

To demonstrate the use of the CSP, this chapter documents the simulation of several test CIRCUITS. The first test demonstrates the capabilities of the CSP for simulating purely combinational switching circuits and for displaying current paths. The second test uses a simple CIRCUIT to illustrate the method for advancing the simulation clocks and for incrementing a counter. Finally, the third test illustrates the use of the CSP in simulating the timing characteristics of a CIRCUIT, including the simulation of race states.

6.2 Case Study 1: Simulation of a Combinational Circuit

The circuit shown in Fig. 6.1 is designed specifically

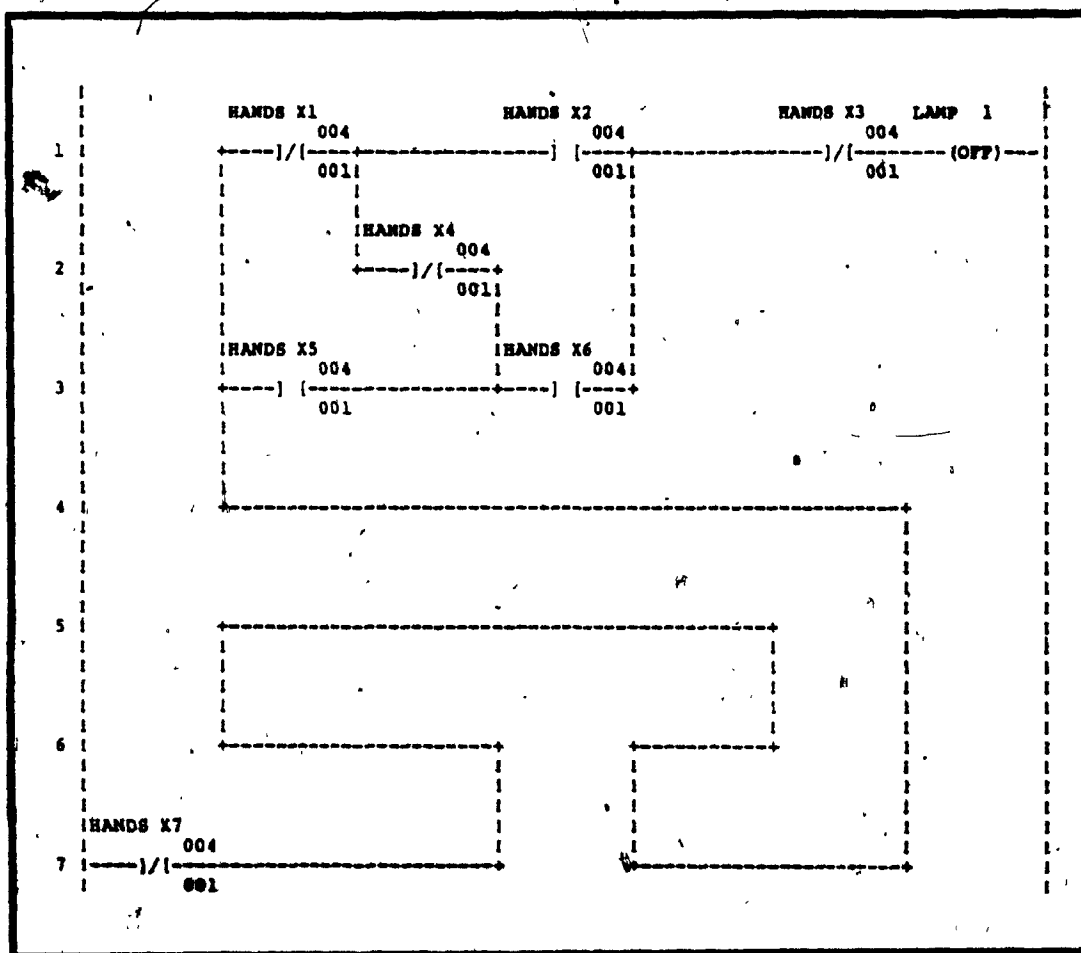


Fig.6.1 Combinational circuit.

for the purpose of illustrating the capabilities of the CSP for displaying the current paths and output state for a given input state. In all, it comprises 7 input devices (HANDS), 1 output device (LAMP), and a number of horizontal and vertical connectors.

The device, HANDS X7, in position 7A, is connected to HANDS X5 in position 3B, through a winding configuration of connectors. It is obvious that these connectors could easily have been removed by connecting HANDS X7 in either position 1A, 2A, or 3A. They are included however, to demonstrate the ability of the CSP to identify complex, backward-directed, current paths.

The CSP could be applied in several ways. One method is to set all the possible input states (of which there are 2^7 for the circuit shown in Fig.6.1), one at a time, and to record the output state corresponding to each of these, in order to obtain a truth table. This method is not preferred however, as it is tedious to apply.

Another method for applying the CSP, is to use it to determine the output state and to identify the current paths for a particular input state of interest. It is often the case, when analysing complex circuits, that the operator is interested in observing the response of the circuit to some change in the input state. The CSP is particularly well suited for this type of application as it allows the operator to interactively set the desired input state and to immediately observe the output state and current paths.

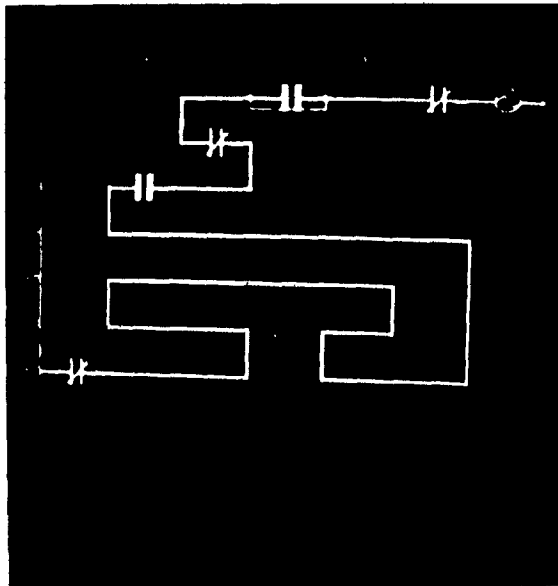
Figure 6.2 shows the current paths and output states, as determined by the CSP and verified by calculation, for several input states of the circuit shown in Fig.6.1. Since this circuit comprises no memory devices, the CSP clocks are not used at all during the simulation. The outputs and the current paths change simultaneously with the input state.

6.3 Case Study 2: Demonstration of the use of Two Simulation Clocks to Simulate Timing

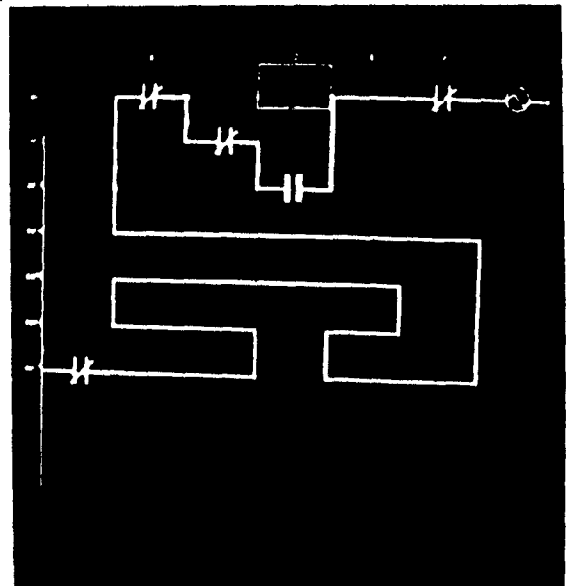
The characteristics of timers, relays, and counters are discussed briefly in Chapters 3 and 5, where it is shown how these devices are used to provide a CIRCUIT with memory. In this section, a CIRCUIT comprising one of each type of memory device is simulated in order to illustrate how the two simulation clocks of the CSP can be used to simulate the timing of a CIRCUIT.

For the purpose of documenting a simulation, the state of a CIRCUIT can be completely specified in the form of a timing chart. Consider the basic structure of the timing chart shown in Fig.6.4 which documents a simulation of the CIRCUIT shown in Fig.6.3. In this representation, each column completely describes the state of the CIRCUIT at some simulated time as indicated by the Relay Clock (row 1) and the Timer Clock (row 2).

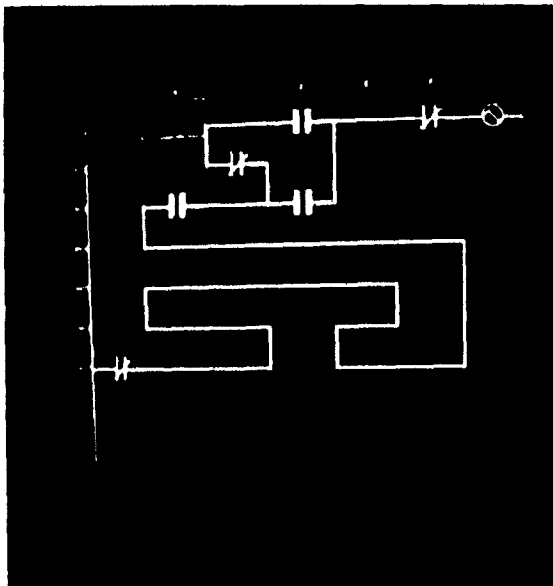
Every output and memory coil in the CIRCUIT is assigned a row in the timing chart. Input and memory



$$X_1 X_2 X_3 X_4 X_5 X_6 X_7 = 1100100$$



$$X_1 X_2 X_3 X_4 X_5 X_6 X_7 = 1100110$$



$$X_1 X_2 X_3 X_4 X_5 X_6 X_7 = 0000010$$

Fig.6.2 Some current path displays for various input states.

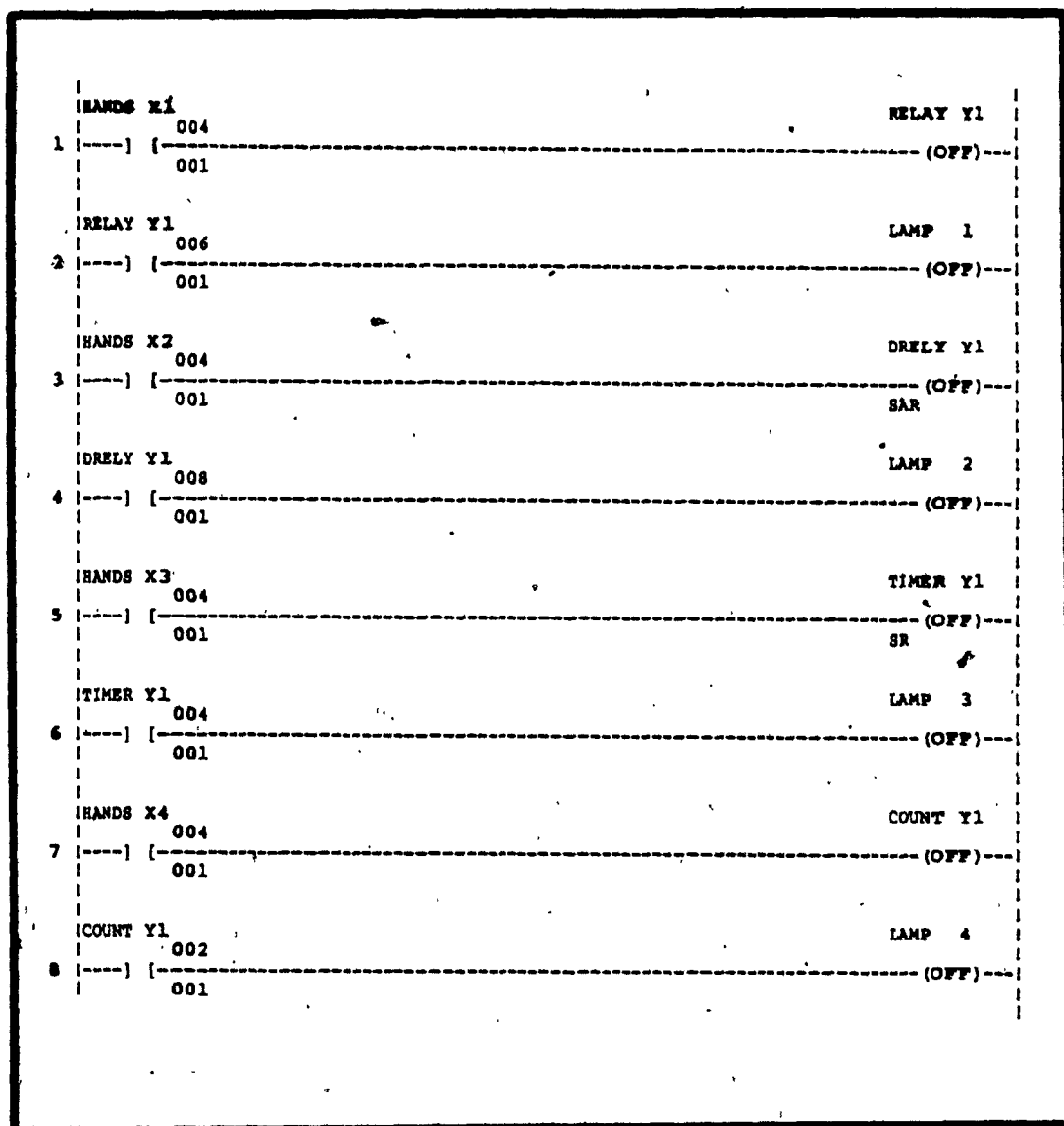


Fig.6.3 CIRCUIT containing relay, delay relay, timer, and counter.

[illegible]

Fig. 6.4 Timing chart for test performed on CIRCUIT shown in Fig. 6.3.3.

contact pairs of the CIRCUIT are also represented in the timing chart. These however, are assigned only one row for each physical device (all contact pairs having the same name and identification number belong to the same physical device) since all the contact pairs belonging to that device will be acted upon by the same excitation.

The rows of the timing chart are grouped with respect to the type of the device. There are four such groups: Input Device; Memory Device Contact Pair; Output Device; and Memory Device Coil.

For the rows of the Input Device group, the entry in each column represents the excitation acting on the contact pairs of the physical input device named in that row. An "X" indicates that the contact pairs of a device are "excited", and a "-" indicates they are "quiescent". To indicate that an excitation has changed a number of times, an integer value indicating the number of times an excitation has been toggled, preceded by the value of the initial state of the excitation, can be entered (i.e. "-4" indicates that an excitation has undergone 4 transitions, beginning with a quiescent value).

Entries in the rows representing devices in the second group consist of either an "X" to indicate that the contact pairs associated with the memory device named are excited, or a "-" to indicate that they are quiescent. The same notation is used to indicate the states of devices in the third group, where an "X" indicates that an output

device is "energised", and a "-" indicates that it is "off".

Finally, for Memory Device Coils, the same notation that is used for devices of the third group also applies. In addition however, an integer value in a column is used to indicate the amount of time that has elapsed, in units of the preset delay duration (in the case of a counter coil, the registered count is given). The delay duration or counter modulus accompanies the device name in the timing chart.

At the bottom of the timing chart, a row labelled "MARK", numbers the CIRCUIT states shown in timing chart sequentially so that they may be easily referred to in the following discussion. MARK has no other significance. It is important to note that in a timing chart, it is not necessary to show all the transient states observed during the simulation.

Also at the bottom of the timing chart, a row labeled "TCAE" (Timer Clock Advance Enable) is included to indicate the times when the operator is able to advance the Timer Clock. These instances are indicated by an "X". Although this information is never explicitly displayed during a simulation, any attempt by the operator to advance the Timer Clock when TCAE="-" will result in the output of an error message.

A timing chart can be obtained directly from the displays produced by the CSP. The entries in each column represent the states of the devices as they are displayed on

the color graphics screen immediately prior to the input of a valid command to advance either simulation clock. The states of the devices are obtained from the graphics screen, while the values on the simulation clocks can be obtained from the data terminal.

Consider the CIRCUIT shown in Fig. 6.3. RELAY Y1, is automatically assigned the default delay duration of 5 millisecond pull-in speed and 3 millisecond drop-out speed. A "SLOW ACTIVATE AND SLOW RELEASE" type delay relay with a 4 millisecond delay duration and a "SLOW RELEASE" type timer with a 5 second delay duration, are selected for RELAY Y1, and TIMER Y1, respectively. Finally, the counter, COUNT Y1, is assigned a modulus of 10.

The purpose of the simulation illustrated by Fig. 6.4, and described below, is to demonstrate the use of the CSP to simulate timing. There are several items which should be noted:

- 1) Several inputs may be changed simultaneously, as illustrated when HANDS X1, HANDS X2, and HANDS X3 are excited at MARK=1.

- 2) Advancing the Relay Clock does not affect the reading on the Timer Clock, as illustrated by advancing the Relay Clock from MARK=7 to MARK=8.

3) Incrementing the counter, COUNT Y1, does not affect either simulation clock, as illustrated at MARK=2, MARK=5, MARK=7 and MARK=10.

4) The Relay Clock is reset when all relays and delay relays are in "steady" states, as illustrated at MARK=0, MARK=4, MARK=11, etc..

The simulation begins with all outputs and memory devices "off", and with all input devices and memory contact pairs in their quiescent states. At MARK=1, the operator excites the inputs: HANDS X1, HANDS X2, and HANDS X3, thus causing the CIRCUIT to enter a "relay transient" state, and the timer coil, TIMER Y1, to become energised. Because the timer is of the "SLOW RELEASE" type, the contact pair, associated with it will become excited simultaneously, causing LAMP 3 to also become energised.

Observing the state of the CIRCUIT, as well as the current paths, the operator advances the Relay Clock by 2 milliseconds, noting only a change in the elapsed durations for RELAY Y1, and DRELY Y1.

With the Relay Clock showing 2 milliseconds (MARK=2), the operator toggles the state of HANDS X3 four times, causing 2 off-to-on transitions of the counter coil (COUNT Y1). The registered count of the device is incremented accordingly. Note that the Relay Clock is not advanced.

Further incrementing the Relay Clock by 2 milliseconds, the operator brings the CIRCUIT to MARK=3, with the Relay Clock showing 4 milliseconds. At this point in the simulation, DRELY Y1, having become energised, is in a steady state. The resulting excitation applied to the contact pair DRELY Y1 causes LAMP 2 to become energised. Also at MARK=3, the excitation is removed from HANDS X3, causing the CIRCUIT to go into a "timer and relay transient" state, since both TIMER Y1, and RELAY Y1 are in transient states. The Timer Clock cannot be advanced however, until the CIRCUIT enters a "timer transient" state.

Advancing the Relay Clock by 1 more millisecond to MARK=4, causes RELAY Y1 and consequently LAMP 1 to become energised. With RELAY Y1 in a steady state, the CIRCUIT enters a "timer-transient" state. This is indicated to the operator when the Relay Clock is reset to 0.

At MARK=5, the operator explicitly advances the Timer Clock by 2 seconds. In addition, he toggles the state of HANDS X4 eight times, to bring the registered count of COUNT Y1 to 6.

The excitation on HANDS X1 is removed at MARK=6 causing the CIRCUIT to once again enter a "relay and timer transient" state. The CSP display for the Ladder Diagram from which the column at MARK=6 in Fig.6.4 was obtained is given in Fig.6.5. Advancing the relay clock twice to read 2 milliseconds brings the CIRCUIT to MARK=7, where HANDS X4 is toggled 4 times to bring the total registered count for

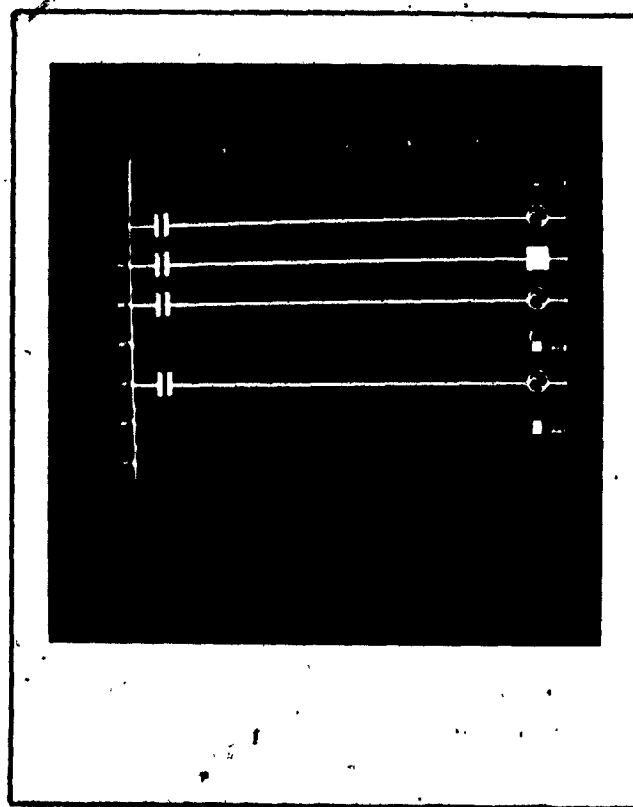


Fig.6.5 CSP display for CIRCUIT shown in Fig.6.3 at MARK = 6.

COUNT Y1 to 8 (80% of MODULUS).

Advancing the Relay Clock to 3 milliseconds (MARK=8), causes RELAY Y1 to reach a steady state, and its associated contact pair to become quiescent, and thus causing LAMP 1 to go "off". Further advancing the Relay Clock two times, brings DRELY Y1 to a "steady" state at MARK=9, and causes LAMP 2 to go "off". The CIRCUIT is now in a "timer transient" state, with the Timer Clock still reading 2 seconds.

At MARK=10, the Timer Clock is advanced by 2 seconds to show that 4 seconds have elapsed. Also, HANDS X4 is toggled 4 times to bring the counter to a steady state, and to cause LAMP 4 to become energised.

Finally, at MARK=11, the Timer Clock is advanced by 1 second to bring the CIRCUIT into a "Steady State", and to cause LAMP 3 to go "off". In the "steady" state, both the relay and Timer Clocks are reset to 0.

The foregoing simulation illustrates several important features of the CSP simulation. First, it shows how the counter can be advanced several times without advancing the CIRCUIT in time. Second, it shows how advancing the Relay Clock does not advance the Timer Clock. Finally, it shows how relay transients can be simulated accurately.

6.4 Case Study 3: Simulation of Race States

The CIRCUIT shown in Fig.6.6 is used by Krieger [32] to illustrate a tabular method for analysing CIRCUITS. This method uses two matrices to store the information describing the CIRCUIT. The Y-matrix, describes the internal state of the CIRCUIT, while the Z-matrix describes the output states. The Y-and Z-matrices for the CIRCUIT shown in Fig.6.6 are given in Fig.6.7 and Fig.6.8 respectively.

Consider the present state of the memory device contact pairs DRELY Y1, DRELY Y2, and DRELY Y3, which is denoted $y_1y_2y_3$. Under the influence of the input state x_1x_2 , which represents the states of HANDS X1 and HANDS X2 respectively, the state of the relay coils is given by each of the entries of this row. That is, if $Y_i=1$, then y_i will become (or remain) excited after some time interval Δt which represents the inertial delay of the device. Similarly, if $Y_i = 0$, then y_i will become (or remain) quiescent. Whenever there arises a situation where $y_1y_2y_3 = Y_1Y_2Y_3$, that entry is said to represent a "stable" state. Stable states are shown as cross-hatched rectangles in Fig.6.7. Otherwise, an entry is said to represent an "unstable" state.

Krieger makes extensive use of the Y-matrix to show the intermediate, unstable states of a CIRCUIT, as it goes from one stable state to another. He also uses it to demonstrate three types of timing hazard: critical

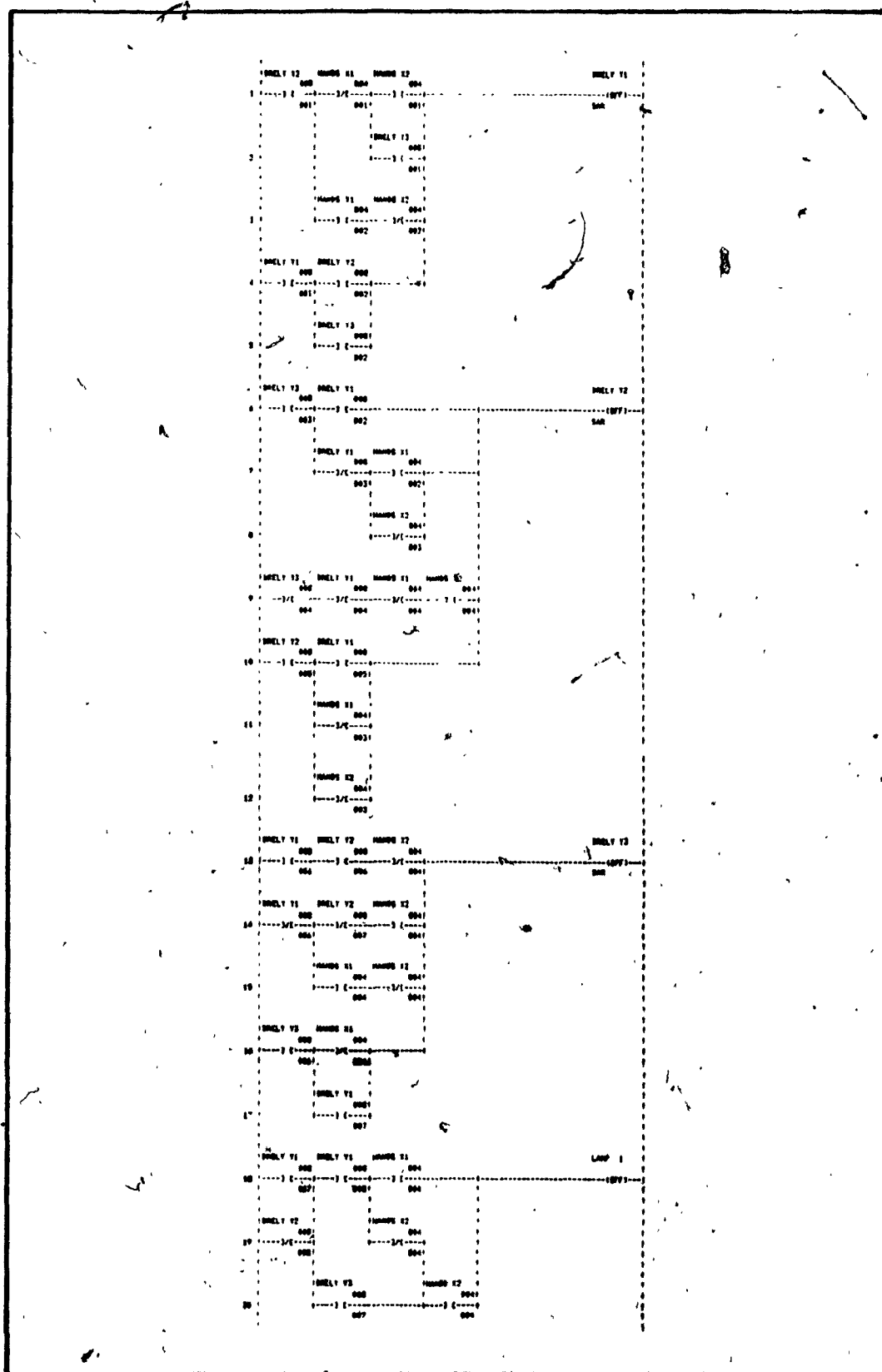


Fig. 6.6 Krieger's test CIRCUIT.

		x_2			
		x_1			
		00	01	11	10
y_1	000	000	011	001	001
	001	011	001	011	011
	011	111	111	010	111
	010	010	110	000	111
	110	111	110	110	111
	111	111	111	111	111
	101	111	111	111	111
	100	000	000	000	000

y_2 { 000, 001, 011, 010, 110, 111, 101, 100 }

y_3 { 000, 001, 011, 010, 110, 111, 101, 100 }

$y_1 y_2 y_3$

Fig.6.7 The Y-matrix for Krieger's test CIRCUIT [32].

		$x_1 x_2$			
		00	01	11	10
$x_1 x_2 y_3$	000	0	0	0	0
	001	0	1	1	1
	011	0	0	0	0
	010	0	0	0	0
	110	0	0	1	1
	111	0	1	1	1
	101	0	1	1	1
	100	0	0	1	1

z_1

Fig.6.8 The Z-matrix for Krieger's test CIRCUIT [32].

race, non-critical or safe race, and buzzer circle.

First, to illustrate the transition from one stable state to another, consider the stable state for $x_1x_2=0$ and $Y_1Y_2Y_3=0$, which will be written as 00/000 for convenience. If the input is changed to $x_1x_2=10$, $Y_1Y_2Y_3$ takes on the value 001. After a time delay, t , the state of the CIRCUIT will be given by 10/001, for which the resulting excitation, $Y_1Y_2Y_3$, is 011. After another delay, the CIRCUIT will be found to be in the state 10/011, which in turn leads to the steady state 10/111 after yet another time delay. This "transition path" is illustrated in Fig.6.9.

To demonstrate a critical race, consider the same initial stable state as in the previous case, i.e., 00/000. If the input state is changed to 01, the excitation becomes 011. Because two memory devices have to change states however, the relative values of their respective inertial delays will determine the path of the subsequent state transitions. From the Y-Matrix, it would be expected that the CIRCUIT would reach the stable state 01/111. If Y_3 changes before Y_2 however, the stable state, 01/001, will be attained. If Y_2 changes first, the CIRCUIT will reach the stable state 01/110, through the unstable state, 01/010. Finally if both Y_2 and Y_3 change simultaneously, the unstable state 01/011 will lead to the stable state 01/111. Hence, the race state 01/011 leads to a critical race as illustrated by Fig.6.10.

Figure 6.11 illustrates a state transition which also

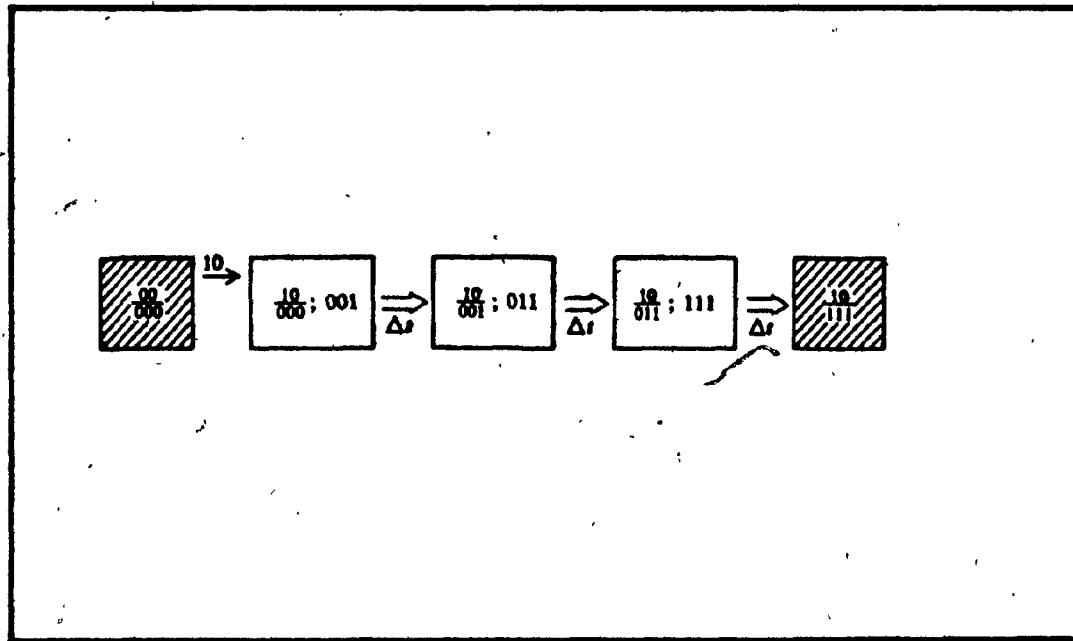


Fig.6.9 A transition path [32] .

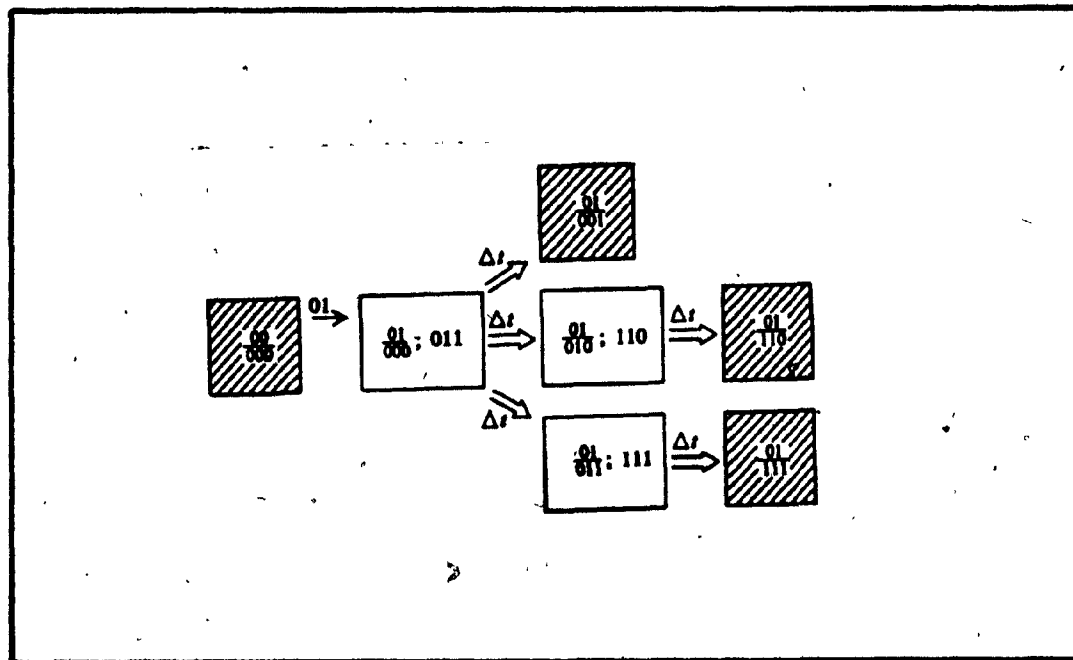


Fig.6.10 A critical race condition [32] .

exhibits a race condition. In this case however, the same stable state will be attained regardless which internal state changes first. Such a race condition is referred to as a non-critical or safe race. It is also possible for a CIRCUIT in transition from a stable state, to enter a path of unstable states which it cannot leave. Such a transition sequence is called a "buzzer circle" and is illustrated by Fig.6.12.

The above studies are discussed in somewhat greater detail in Krieger [32]. The reason they are discussed here is to show that, given the Y- and Z-matrices for a CIRCUIT, it is a relatively simple matter to analyse the operation of a small, but fairly complex CIRCUIT. The greatest difficulty with the tabular method for analysing a CIRCUIT is that the Y- and Z-matrices are difficult to obtain when a large number of input devices and memory devices are involved.

The CSP is capable of simulating a CIRCUIT given an initial stable state and a change in the input state that initiates the state transition. In order to bring the CIRCUIT to a desired, initial stable state, the operator has the option of either following the necessary sequence of state transitions to achieve that state, or to use the "FAULT CONDITION" facility of the CSP to force the states of the memory devices. When the desired stable state has been reached, the forcing may be removed. Then the operator can observe the transition path as he advances the simulation

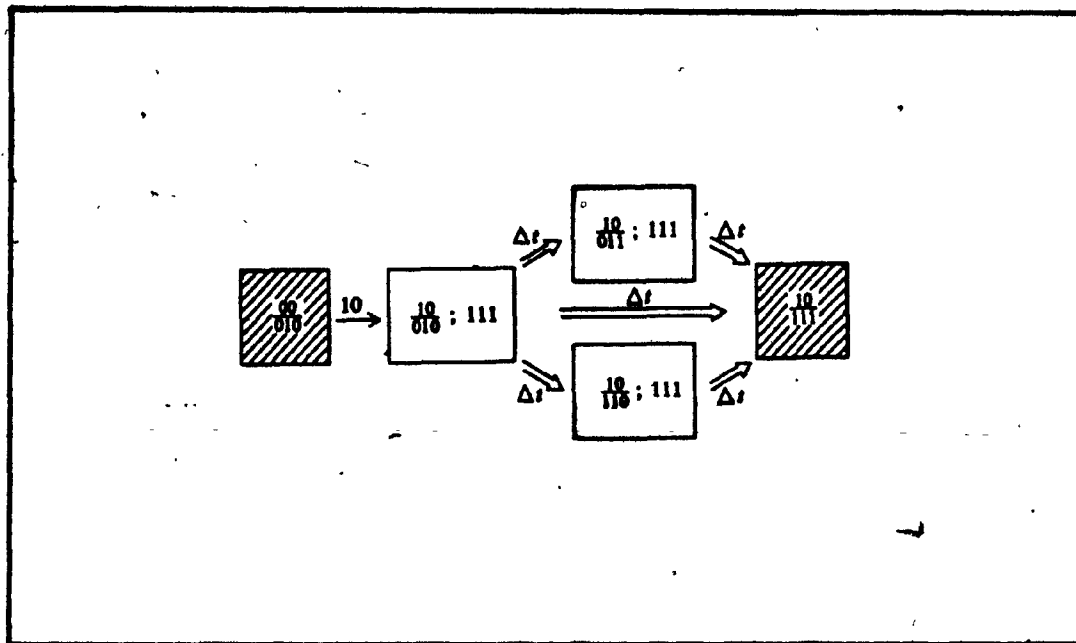


Fig.6.11 A safe race condition [32] .

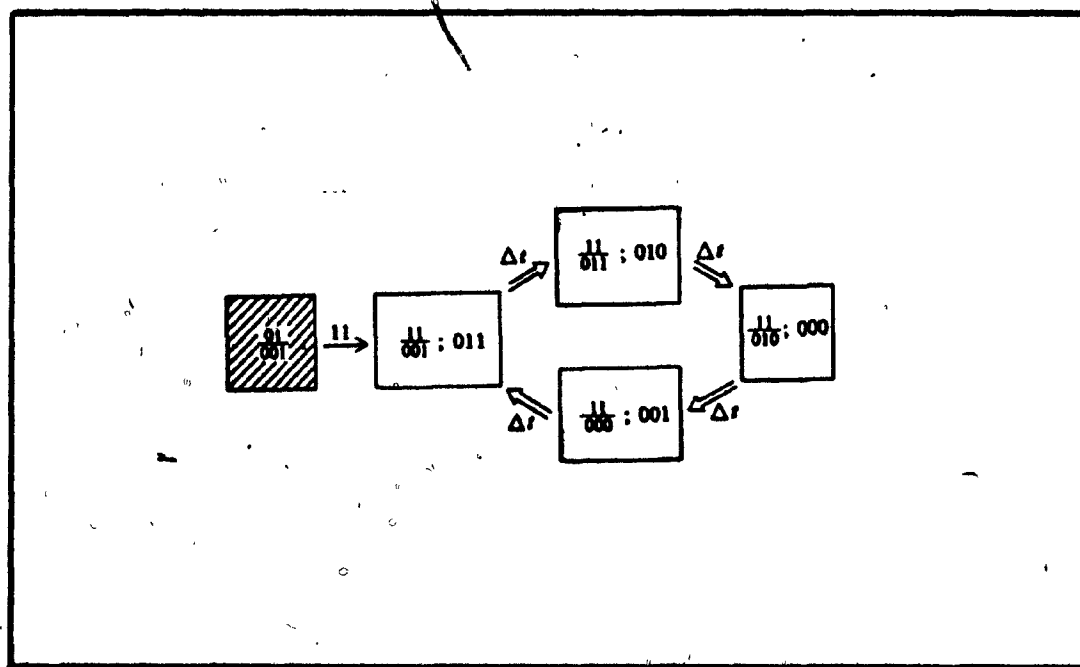


Fig.6.12 A "buzzer circle" [32] .

clocks.

6.4.1 Use of CSP to Simulate a Simple Transition Path

For the purpose of the simulation, all delay relays are selected to be of the "SLOW ACTIVATE AND SLOW RELEASE" type with a 2 millisecond delay duration. Figure 6.13 shows the timing chart for the simulation.

At MARK=1, an excitation is applied to HANDS X1 which causes the CIRCUIT to enter a "relay transient" state. As the CIRCUIT is advanced in time, the transition sequence illustrated by Fig.6.9 is observed. This timing chart is generated in the manner described in Section 6.3. The Timer Clock and the TCAE are not shown since they are not required. The CSP display for the Ladder Diagram from which the column at MARK=3 in Fig.6.13 was obtained, is given in Fig.6.14.

6.4.2 Use of CSP to Simulate Race States

With the CIRCUIT initially in the stable state 00/000, an excitation is applied to HANDS X2. Depending on the value of the delay durations of the relay devices, a different final state will be achieved. In the following discussion, the simulations are documented using timing charts which are obtained in the manner described in Section 6.3.

		DURATION OF DELAY									
	RELAY CLOCK		0	0	1	2	3	4	5	0	
INPUTS	HANDS 1		-	X	X	X	X	X	X	X	
	HANDS 2		-	-	-	-	-	-	-	-	
MEMORY CONTACT	DRELY 1		-	-	-	-	-	-	-	X	
	DRELY 2		-	-	-	-	-	X	X	X	
	DRELY 3		-	-	-	X	X	X	X	X	
MEMORY COILS	DRELY 1	2	-	-	-	-	-	0	1	X	
	DRELY 2	2	-	-	-	0	1	X	X	X	
	DRELY 3	2	-	0	1	X	X	X	X	X	
OUTPUT	LAMP 1		-	-	-	X	X	-	-	X	
	MARK		0	1	2	3	4	5	6	7	

Fig.6.13 Timing chart for transition from 00/000 for
00 to 10 input transition.

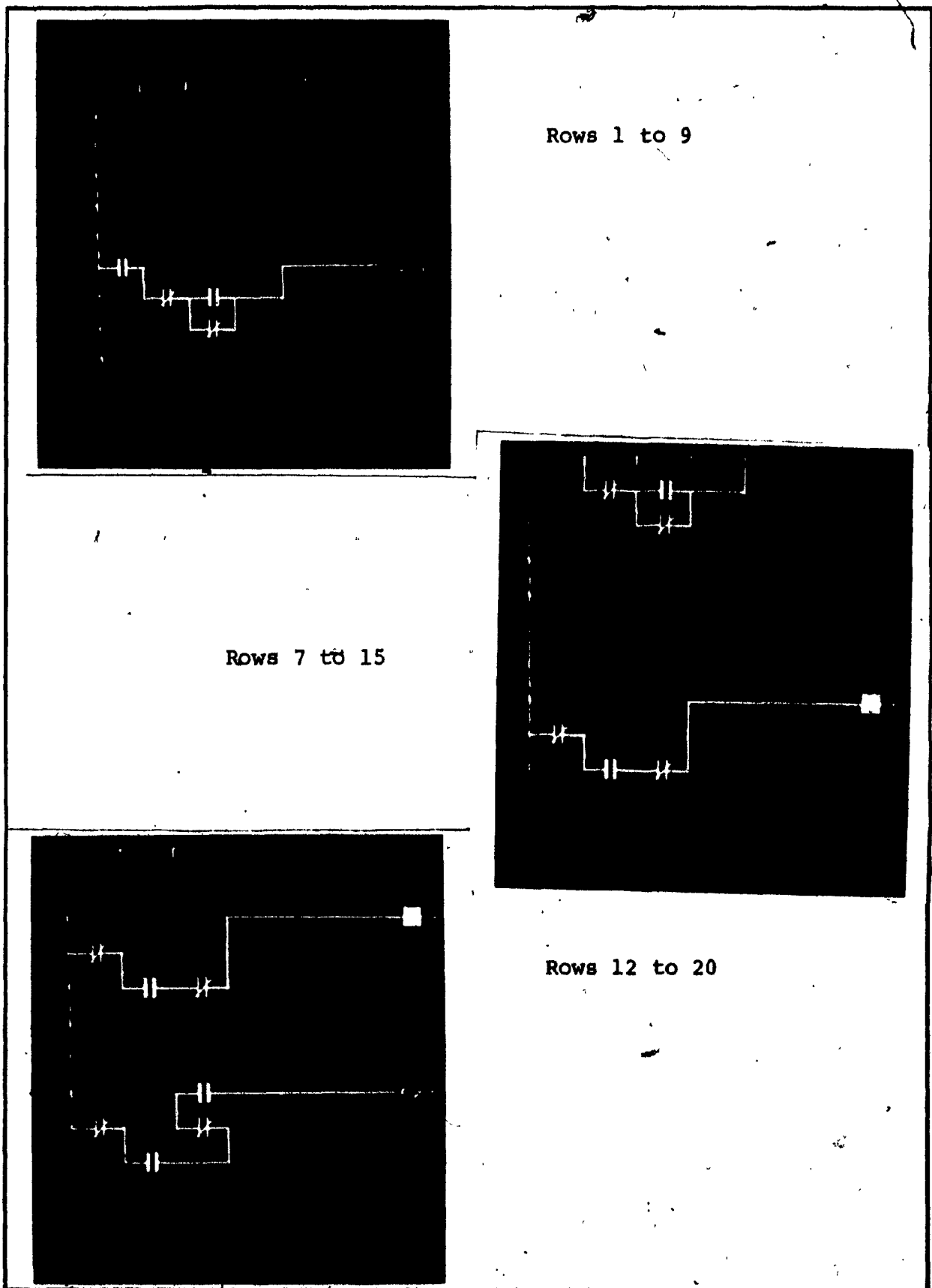


Fig.6.14 CSP display used to obtain column at MARK=3 in Fig.6.13.

6.4.2.1 Case I: DRELY Y2 and DRELY Y3 Change Simultaneously

If the inertial delay durations of all 3 delay relays are identical, say 2 milliseconds, then, the CIRCUIT will undergo a transition as illustrated by the timing chart of Fig.6.15.

At MARK=1, the excitation applied to HANDS X2 causes both DRELY Y2 and DRELY Y3 to go into transient states. The graphic display from which the state at MARK=2 was obtained is given in Fig.6.16. One millisecond later, at MARK=3, the internal state becomes 01/011 causing DRELY Y1 to enter a transient state. Advancing the Relay Clock two more milliseconds brings the CIRCUIT to the final steady state 10/111, for which LAMP 1 becomes energised, (MARK=4).

6.4.2.2 Case II: DRELY Y3 Changes State Before DRELY Y2

Beginning with the same initial state as in the previous case, with the delay duration of DRELY Y3 being 1 millisecond and that for DRELY Y2 and DRELY Y1 being 2 milliseconds, the CIRCUIT will go into the steady state 01/001, 1 millisecond after the excitation is applied to HANDS X2, as shown in Fig.6.17.

		DURATION OF DELAY						
	RELAY CLOCK		0	0	1	2	3	0
INPUTS	HANDS X1		-	-	-	-	-	-
	HANDS X2		-	X	X	X	X	X
MEMORY CONTACT	DRELY Y1		-	-	-	-	-	X
	DRELY Y2		-	-	-	X	X	X
	DRELY Y3		-	-	-	X	X	X
MEMORY COILS	DRELY Y1	2	-	-	-	0	1	X
	DRELY Y2	2	-	0	1	X	X	X
	DRELY Y3	2	-	0	1	X	X	X
OUTPUT	LAMP 1		-	-	-	-	-	X
	MARK		0	1	2	3	4	5

Fig.6.15 Timing chart for transition from 00/000 for 00 to 01 input transition (DRELY Y2 and DRELY Y3 change simultaneously).

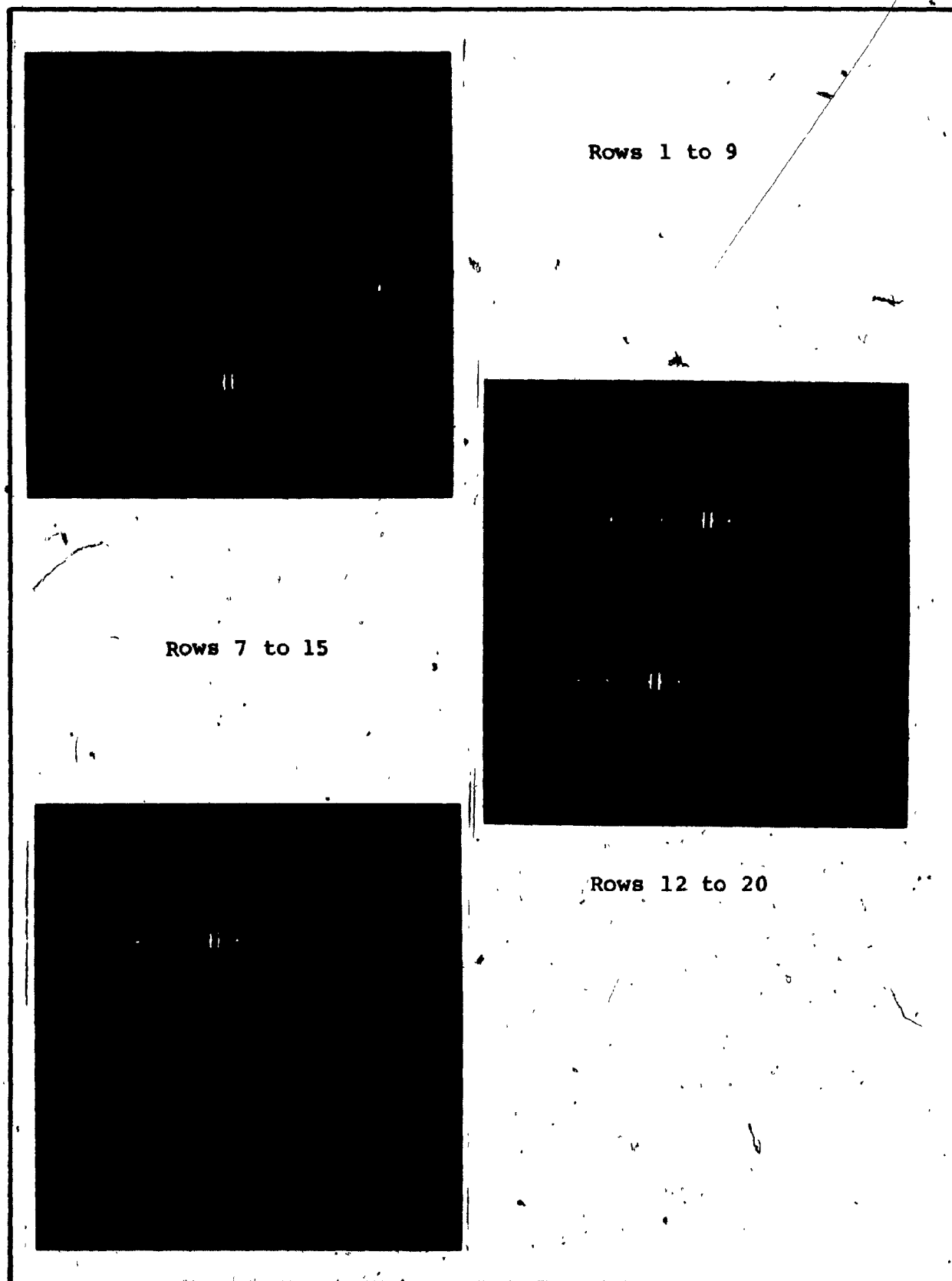


Fig.6.16 CSP display used to obtain column at MARK=2 in Fig.6.15.

6.4.2.3 Case III: DRELY Y2 Changes Before DRELY Y3

The timing chart for this state transition is given in Fig. 6.18.

6.4.2 Use of CSP to Simulate Safe Race

The timing chart given in Fig. 6.19 illustrates how the CIRCUIT is brought to the initial steady state 00/010. The timing charts in Fig. 6.20, Fig. 6.21, and Fig. 6.22 illustrate how, from this initial steady state, the transition path due to the input state transition from 00 to 10 will always lead to the 10/111 final steady state.

6.4.3 Use of CSP to Simulate a Buzzer Circle

A state transition diagram illustrating the buzzer circle was given in Fig. 6.12. In order to simulate this state transition, it is necessary to first bring the CIRCUIT to the steady state. The simplest method to do this is to first force the coil of DRELY Y3 to become energised thus exciting the associated contact pairs, and then to set the input excitation to 01. Subsequent removal of the forcing on the coil will leave the CIRCUIT in the desired steady state. This sequence is illustrated in Fig. 6.23 for MARK=1, MARK=2, and MARK=3.

		DURATION OF DELAY					
	RELAY CLOCK		0	0	2	4	7
INPUTS	HANDS X1		-	-	-	-	-
	HANDS X2		-	X	X	X	X
MEMORY CONTACT	DRELY Y1		-	-	-	-	X
	DRELY Y2		-	-	X	X	X
	DRELY Y3		-	-	-	-	-
MEMORY COILS	DRELY Y1	3	-	-	-	0	X
	DRELY Y2	2	-	0	X	X	X
	DRELY Y3	3	-	0	2	-	-
OUTPUT	LAMP 1		-	-	-	-	-
	MARK		0	1	2	3	4

Fig.6.17 Timing chart for state transition from 00/000 for 00 to 01 input transition (DRELY Y3 changes before DRELY Y2).

		DURATION OF DELAY			
	RELAY CLOCK		0	0	0
INPUTS	HANDS X1		-	-	-
	HANDS X2		-	X	X
MEMORY CONTACT	DRELY Y1		-	-	-
	DRELY Y2		-	-	-
	DRELY Y3		-	-	X
MEMORY COILS	DRELY Y1	2	-	-	-
	DRELY Y2	2	-	0	-
	DRELY Y3	1	-	0	X
OUTPUT	LAMP 1		-	-	X
	MARK		0	1	2

Fig.6.18 Timing chart for state transition from 00/000 for 00 to 01 input transition (DRELY Y2 changes before DRELY Y3).

		DURATION OF DELAY					
	RELAY CLOCK		0	0	0	1	0
INPUTS	HANDS X1		-	-	-	-	-
	HANDS X2		-	-	-	-	-
MEMORY CONTACT	DRELY Y1		-	-	-	-	-
	DRELY Y2		-	X	X	X	X
	DRELY Y3		-	-	-	-	-
MEMORY COILS	DRELY Y1	2	-	-	-	-	-
	DRELY Y2	2	-	FX	0	1	X
	DRELY Y3	2	-	-	-	-	-
OUTPUT	LAMP 1		-	-	-	-	-
	MARK		0	1	2	3	4

Fig.6.19 "Forcing" used to set initial, steady state.

		DURATION OF DELAY					
	RELAY CLOCK		0	0	1	0	
INPUTS	HANDS X1		-	X	X	X	
	HANDS X2		-	-	-	-	
MEMORY CONTACT	DRELY Y1		-	-	-	X	
	DRELY Y2		X	X	X	X	
	DRELY Y3		-	-	-	X	
MEMORY COILS	DRELY Y1	2	-	0	1	X	
	DRELY Y2	2	X	X	X	X	
	DRELY Y3	2	-	0	1	X	
OUTPUT	LAMP 1		-	-	-	X	
	MARK		0	1	2	3	

Fig.6.20 Timing chart for transition from 00/010, for 00 to 10 input transition (DRELY Y1 and DRELY Y3 change simultaneously).

		DURATION OF DELAY				
	RELAY CLOCK		0	0	1	0
INPUTS	HANDS X1		-	X	X	X
	HANDS X2		-	-	-	-
MEMORY CONTACT	DRELY Y1		-	-	-	X
	DRELY Y2		X	X	X	X
	DRELY Y3		-	-	X	X
MEMORY COILS	DRELY Y1	2	-	0	1	X
	DRELY Y2	2	X	X	X	X
	DRELY Y3	1	-	0	X	X
OUTPUT	LAMP 1		-	-	-	X
	MARK		0	1	2	3

Fig.6.21 Timing chart for transition from 00/010 for 00 to 10 input transition (DRELY Y3 change before DRELY Y1).

		DURATION OF DELAY				
	RELAY CLOCK		0	0	1	0
INPUTS	HANDS X1		-	X	X	X
	HANDS X2		-	-	-	-
MEMORY CONTACT	DRELY Y1		-	-	X	X
	DRELY Y2		X	X	X	X
	DRELY Y3		-	-	-	X
MEMORY COILS	DRELY Y1	1	-	0	X	X
	DRELY Y2	2	X	X	X	X
	DRELY Y3	2	-	0	1	X
OUTPUT	LAMP 1		-	-	-	X
	MARK		0	1	2	3

Fig.6.22 Timing chart for transition from 00/010 for 00 to 10 input transition (DRELY Y1 change before DRELY Y3).

	DURATION OF DELAY	RELAY CLOCK																
		0	0	1	0	0	1	2	3	4	5	6	7	8	9	10	11	12
INPUTS	HANDS X1	-	-	-	-	X	X	X	X	X	X	X	X	X	X	X	X	X
	HANDS X2	-	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
MEMORY CONTACT	DRELY Y1-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	DRELY Y2	-	-	-	-	-	X	X	-	X	X	-	-	X	X	-	-	-
	DRELY Y3	-	FX	X	X	X	X	-	-	X	X	-	-	X	X	-	-	X
MEMORY COILS	DRELY Y1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	DRELY Y2	-	-	-	-	0	X	0	-	0	X	0	-	0	X	0	-	0
	DRELY Y3	-	FX	0	X	X	0	-	0	X	0	-	0	X	0	-	0	X
OUTPUT	LAMP 1	-	X	X	X	X	-	-	-	X	-	-	-	X	-	-	-	X
	MARK	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Fig. 6.23 Timing chart for simulation of a Buzzer Circle.

By changing the input excitation from 01 to 11, the CIRCUIIT is made to enter a sequence of transient states which it cannot leave. This is illustrated in Fig.6.23, where the delay duration of the delay relays is 1 millisecond.

6.5 Summary

In this chapter, it was shown by illustrative examples, how the CSP can be used to quickly and accurately analyse a complex CIRCUIIT, without the need for a truth table, or similar specification. This includes the determination of the output states, the identification of current paths, and the simulation of race states. Although the CSP is applied only to fairly small CIRCUIITs, it is clear that the principles of its operation are solidly founded and that the CSP can be applied with confidence to any CIRCUIIT which conforms to the constraints described in Section 3.2.2.

CHAPTER 7

CONCEPTUAL DESIGN OF A GENERAL PURPOSE MICROCOMPUTER BASED PROGRAMMABLE LOGIC CONTROLLER

7.1 Introduction

Programmable Logic Controllers have been widely accepted in industry as a reliable, cost effective alternative to hardwired Relay Logic Circuits. They provide facilities for interactively specifying, modifying and implementing CIRCUITS, and are constructed to withstand the often harsh, industrial environment.

Figure 7.1 shows a block diagram of a typical PLC. The PLC comprises 3 basic components: the central processing unit and memory (CPU); the programming unit; and the I/O modules. The programming unit provides an interface between the operator and the CPU while the I/O modules provide the interface between the CPU and the process or machine being controlled. It is the CPU, which is typically an 8- or a 16-bit microprocessor, that is charged with the task of managing and performing all communications and analyses. For a PLC, the software used to perform these functions is stored in a "Read Only Memory" (ROM), and strictly defines the capabilities of the device.

A conceptual design of a novel PLC, the Concordia Programmable Logic Controller (CPLC), is presented herein. A general purpose microcomputer, rather than a dedicated

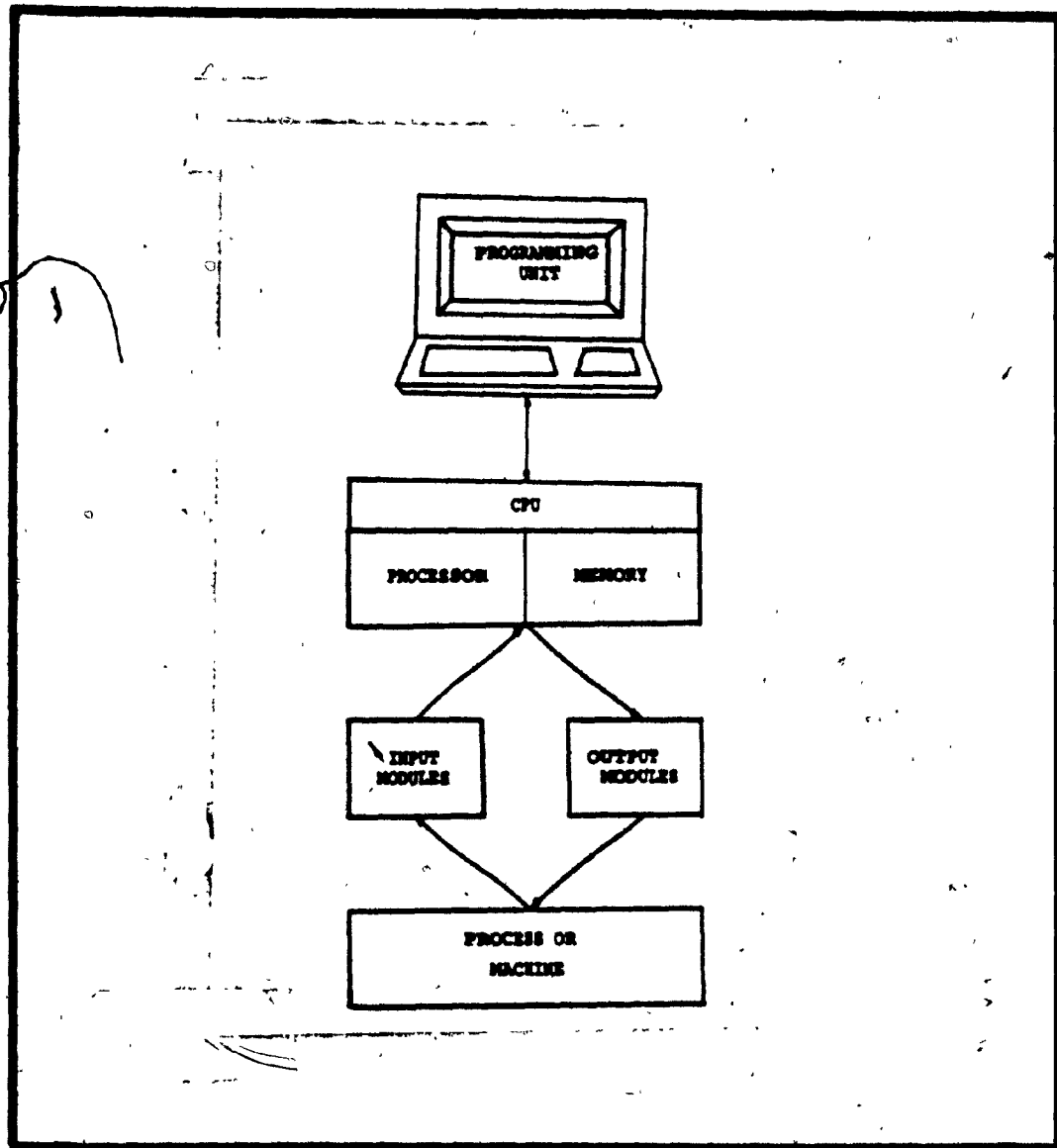


Fig.7.1 Block diagram of typical PLC.

microprocessor, is used to perform the functions of the CPU. The necessary software is loaded into memory from a floppy diskette, and an I/O interface is connected to the parallel input and output ports of the microcomputer, to allow the CPU to read the input state, and to drive the outputs.

In some applications, the CPLC can provide several advantages over commercial PLCs. Foremost of these advantages is its relatively low cost. Commercial PLCs are constructed to withstand a harsh industrial environment and must meet stringent industrial standards. Although the satisfaction of these is clearly necessary when the PLC is to be used in industry, it is completely unnecessary when the PLC is destined for use in teaching technical personnel how to program and use such equipment. Thus, for teaching applications, the CPLC could emulate the features offered by the more powerful PLCs at a small fraction of the cost (i.e. the cost of the software and the I/O interface, assuming that a microcomputer is available).

Another advantage of the CPLC is that the software is written almost entirely in a high level language (FORTRAN). This being the case, it is possible for the user to modify the CPLC to meet his requirements. Examples of such modifications include the development of new CIRCUIT analysis techniques to improve scan speed and the addition of PID control capabilities.

An additional advantage of the CPLC is that it can

provide facilities for programming commercial PLCs by downloading CIRCUIT configurations to them. This, of course, would require the cooperation of the PLC manufacturers in supplying the coding formats for their respective PLCs.

The only foreseeable disadvantage of the CPLC is that its software is written in a high level language and, as a result, its operation will be slower than if it were written in a more efficient, Assembly or Machine Language.

7.2 Brief Description of the CPLC

The CPLC software consists of two programs: the microcomputer version of the CSP, and the CIRCUIT Realization Program (CRP). The microcomputer version of the CSP was described in detail in Chapter 5, and is used to specify and simulate CIRCUITS, interactively on a graphics screen. The CRP operates in real time and reads the input state, analyses the CIRCUIT, and sets the output state through the I/O module designed conceptually in this chapter.

The CIRCUIT to be realized by the CPLC is first entered and stored on file using the CSP. When the CRP is executed, the data base for this CIRCUIT is loaded into the memory and an input or output port is assigned to each device. Every device in the process or machine to be controlled, is physically connected to its designated port

on the multiplexer, through which, the state of any input device can be selectively entered into the data base, and consequently, the state of any output device can be selectively set. In the course of the operation of the CRP, the input device states are polled and read into the data base at regular intervals.

When all the inputs have been polled, the CIRCUIT model is analysed, and the states of the output devices are determined. When timers or delay relays in transient states are present in a CIRCUIT, a real time clock is used to measure the elapsed time for each such device, and to set the states of their respective contact pairs automatically. Relay and counter devices are implemented entirely by software.

Once the output device states have been determined, the appropriate control signal is sent to each output device through the multiplexer, and the states of the relay and counter contact pairs are updated. This entire process is repeated each time the polling of the inputs is initiated.

7.3 Conceptual Design of the Hardware Configuration of the CPLC

The hardware configuration of the CPLC is shown in Fig.7.2. It can be seen that, in addition to an 8-bit microcomputer, it comprises an I/O multiplexer, a real time clock, counters, input latches, and output buffers and

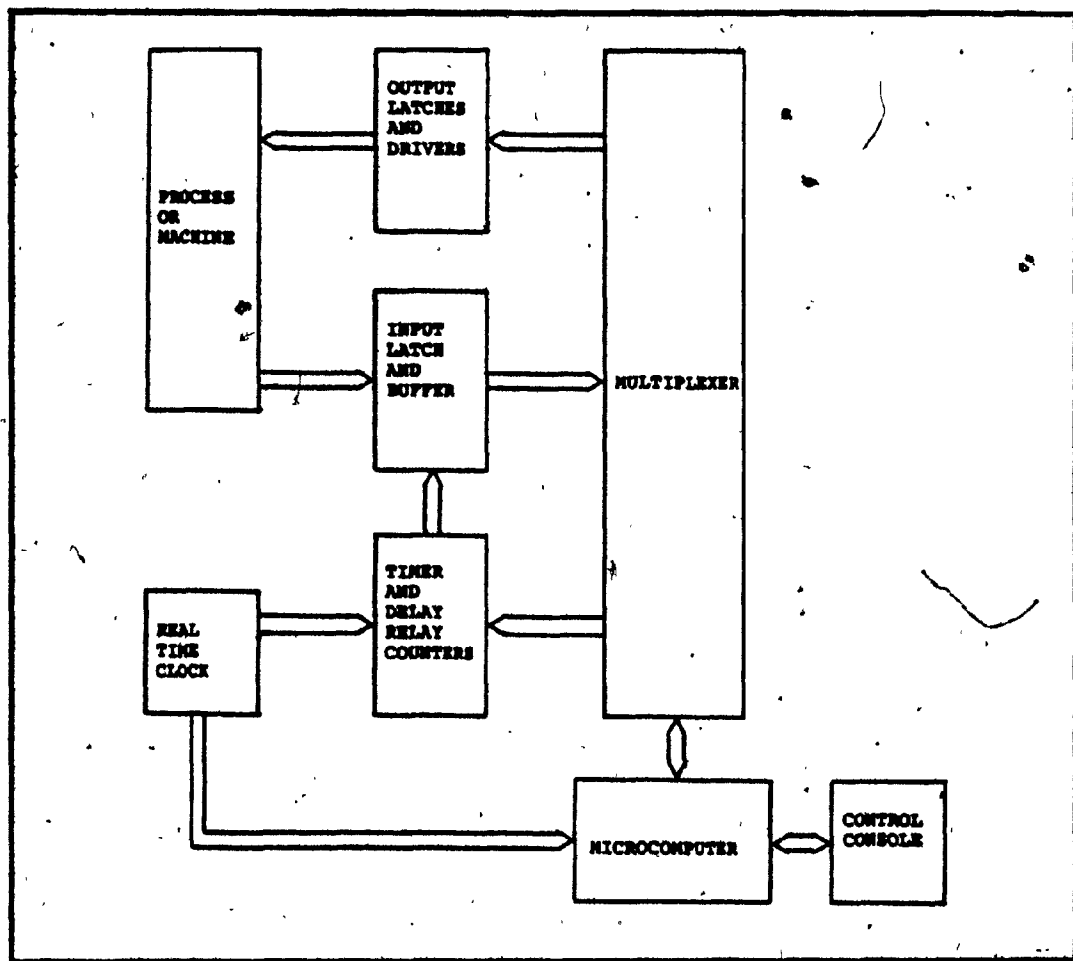


Fig.7.2 Block diagram of proposed hardware configuration for CPLC.

drivers.

7.3.1 Multiplexer

The numbers of input and output devices which the CPLC can accomodate is limited by the ability of the microcomputer to address these devices through the available input and output ports. An 8-bit microcomputer will normally provide parallel input and output ports, each with 8 data lines, and a number of control lines. Depending upon the complexity of the hardware design of the multiplexer, a wide range of I/O devices can be accomodated (anywhere from seyerall, to more than two thousand).

Without detailing a multiplexer design, it is sufficient to note that as the number of I/Os increases, so will the complexity of the hardware and software, required to address them. Since, only 8 data lines are available, it can be expected that in order to use these to both address devices, and to send data to these devices through the same 8 lines, the hardware design will be rather complex when more than 16 outputs are involved.

7.3.2 Real Time Clock

An external, real time clock provides the three different clock frequencies required by the CPLC. A 1000 Hz clock, is required to measure delay relay delays, a 1 Hz

clock to measure timer delays, and one additional clock to initiate successive pollings of the input and the internal states. The determination of the frequency of this clock, hereafter referred to as the polling clock, is discussed in Section 7.5.2.

7.3.3 Timer and Delay Relay Duration Counters

Timers and delay relays require that the timer delay durations they represent be measured in real time. The CPLC dedicates a certain number of hardwired inputs and outputs to the realization of these memory devices. Hence, the number of timers or delay relays which can be accommodated by the CPLC is limited to the number of I/O lines dedicated to this function.

Each timer and delay relay coil is realized by a hardwired, presetable, programmable, modulo- M_C counter. When the coil of such a device is found to have changed states by the CIRCUIT analysis software, a signal is sent through the multiplexer which presets the programmable counter of that device to a value P_C , such that

$$P_C = M_C - D_C \quad (7.1)$$

where, D_C is the duration of the delay to be realized. At the same time, the counter is enabled to count clock pulses from the appropriate real time clock. When a number of

clock pulses has been counted which is equal to D_c , the counter output will be M_c . When this condition is satisfied a logic signal is set, or reset, in order to indicate the state (excited or quiescent) of the contact pairs of that device. This signal is in turn connected to the multiplexer as an input to the CPLC. Figure 7.3 shows how timers and delay relay delays are implemented. During each polling of the input states therefore, the states of timer and delay relay contact pairs are available in real time, and are accurate to within 1 polling clock cycle.

7.3.4 Input Latch and Buffer

From the time the polling process is initiated, until the time when the last input has been read into the data base, it is possible that some inputs will have changed states. During the analysis however, all inputs entered during a polling sequence are treated as if they occurred simultaneously. In order to ensure that all the inputs are sampled at the same instant, their states are latched with a single pulse when polling begins.

7.3.5 Output Latch

When a signal to set an output device state is sent through the multiplexer, it persists only momentarily. In order to latch this signal, and to maintain the output

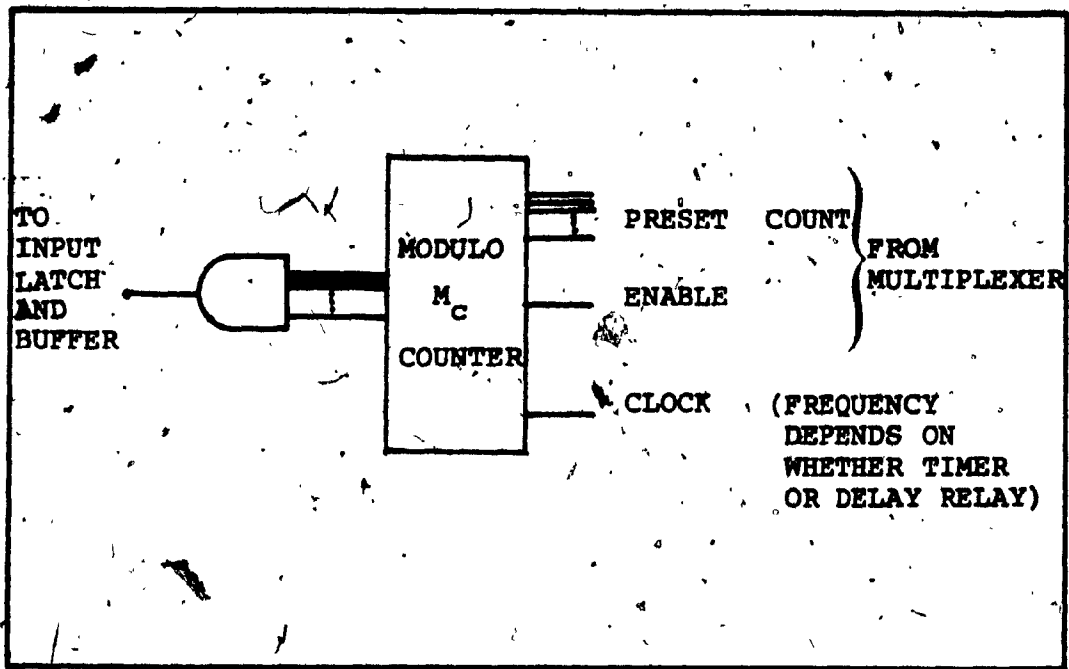


Fig.7.3 Typical timer or delay relay realization.

states until the next state determination has been completed, an output buffer must be provided.

7.4 Conceptual Design of the CRP

7.4.1 Polling Subprogram

Each time the polling subroutine is initiated by the polling clock, a pulse is sent to the strobe input of the latch for each input device in the CIRCUIT. Subsequently, each such device is individually addressed, and its state is read and loaded into the data base. When the polling has been completed, the CIRCUIT is submitted for modelling and analysis. Figure 7.4 shows a flowchart for the polling subprogram.

7.4.2 Modelling and Analysis

Either the method described in detail in Chapter 3, or the Graph Theory method described in Chapter 4, could be used as the basis for the analysis of CIRCUITS used by the CRP. Although only a conceptual design has been given, it is this author's opinion that the method based on the Graph Theory method would be preferable over the method which models a CIRCUIT as a resistive network since its implementation promises to use less memory and to be faster.

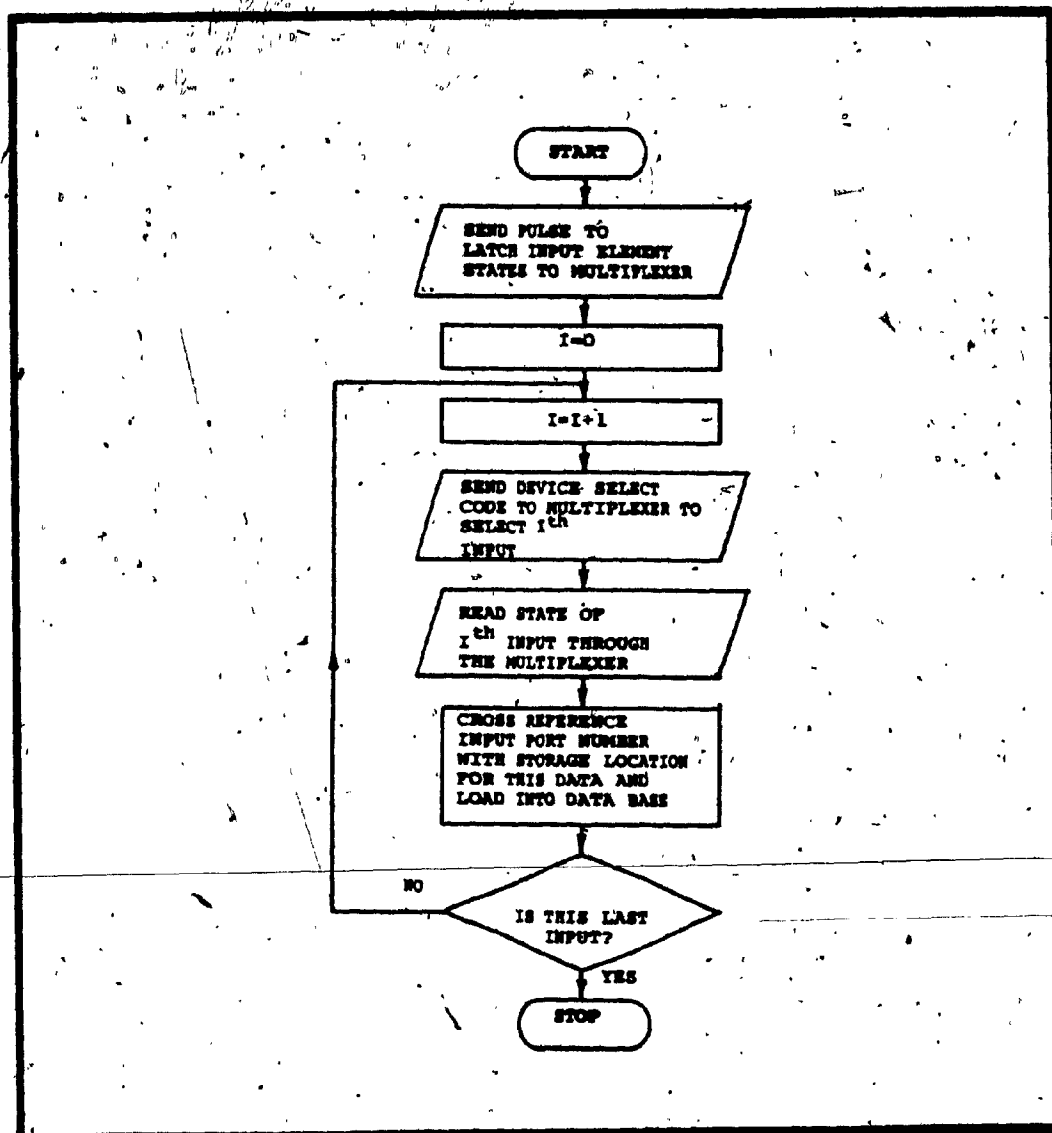


Fig.7.4 Flowchart of polling subprogram.

7.4.3 Set Relay and Counter Contact Pair States

Unlike timers and delay relays, relay and counter elements are realized entirely by software. As a result, given the states of the relay and counter coils in a CIRCUIT, the contact pairs associated with each coil must be set by a software routine.

In the CSP, the relay devices were assigned a default inertial delay value in order to simulate the characteristics of electromechanical devices. A relay realized by software will also display an inertial delay. The duration of this delay will be, at most, 1 polling clock period. This is because a change in the state of a memory device during 1 clock cycle can only be detected during the next polling clock cycle.

7.4.4 Setting of Output Device States

Once the output state has been determined, the states of the individual output devices and memory device coils can be sent through the multiplexer to their respective output buffers, and output drivers.

7.4.5 Executive Subprogram

The Executive subprogram supervises the operation of the CRP and, from it, a CIRCUIT can be interactively loaded,

initialized and submitted for implementation. A facility for the interactive manual manipulation of the output devices is also provided.

7.4.5.1 Loading of a Ladder Diagram from a Disk File

CIRCUITS can be configured by the interactive CIRCUIT configuration program described in Chapter 5 and stored on a disk file. The Executive program can be used to load a stored CIRCUIT into the CPLC data base, which is identical to that used for the CSP.

Once a CIRCUIT configuration has been loaded, each device is automatically assigned an input or an output port. Thereafter, the multiplexer is used to address devices using only these I/O port numbers rather than the device names and identifications as shown on the Ladder Diagram of a CIRCUIT.

When the device assignment has been completed, the delay durations of the timers and delay relays, which are used to preset the hardware counters, are entered interactively by the operator.

7.4.5.2 Manual Mode

The Executive Program allows the operator to interactively set the state of any output device, without consideration of the input or internal states. This facility provides for the testing of the operation of the

outputs, and for the setting of the initial conditions for the process or machine being controlled.

7.4.5.3 Automatic Operation

When the Executive Program calls the Automatic Mode, the CPLC is given complete control of the process or machine. The "STOP" function is the only allowable operator input through the keyboard. This causes all the outputs to be reset and gives control back to the Executive Program.

The polling clock synchronizes the operation of the CPLC. Every clock pulse initiates a polling of the input device states. When polling has been completed, the CIRCUIT is modelled and analysed before the newly determined output device states are set. Figure 7.5 shows the sequence of operations executed in Automatic Mode.

7.5 Setting of the Polling Clock Frequency

The polling clock initiates the sequence of operations described in Section 7.4.5.3. The proper operation of the CPLC requires that this sequence be executed completely before the next one is initiated. Hence, the frequency of the polling clock will be determined by the amount of the time required to execute that sequence.

This sequence of operations involves several program subroutines as well as outputs and inputs to and from the

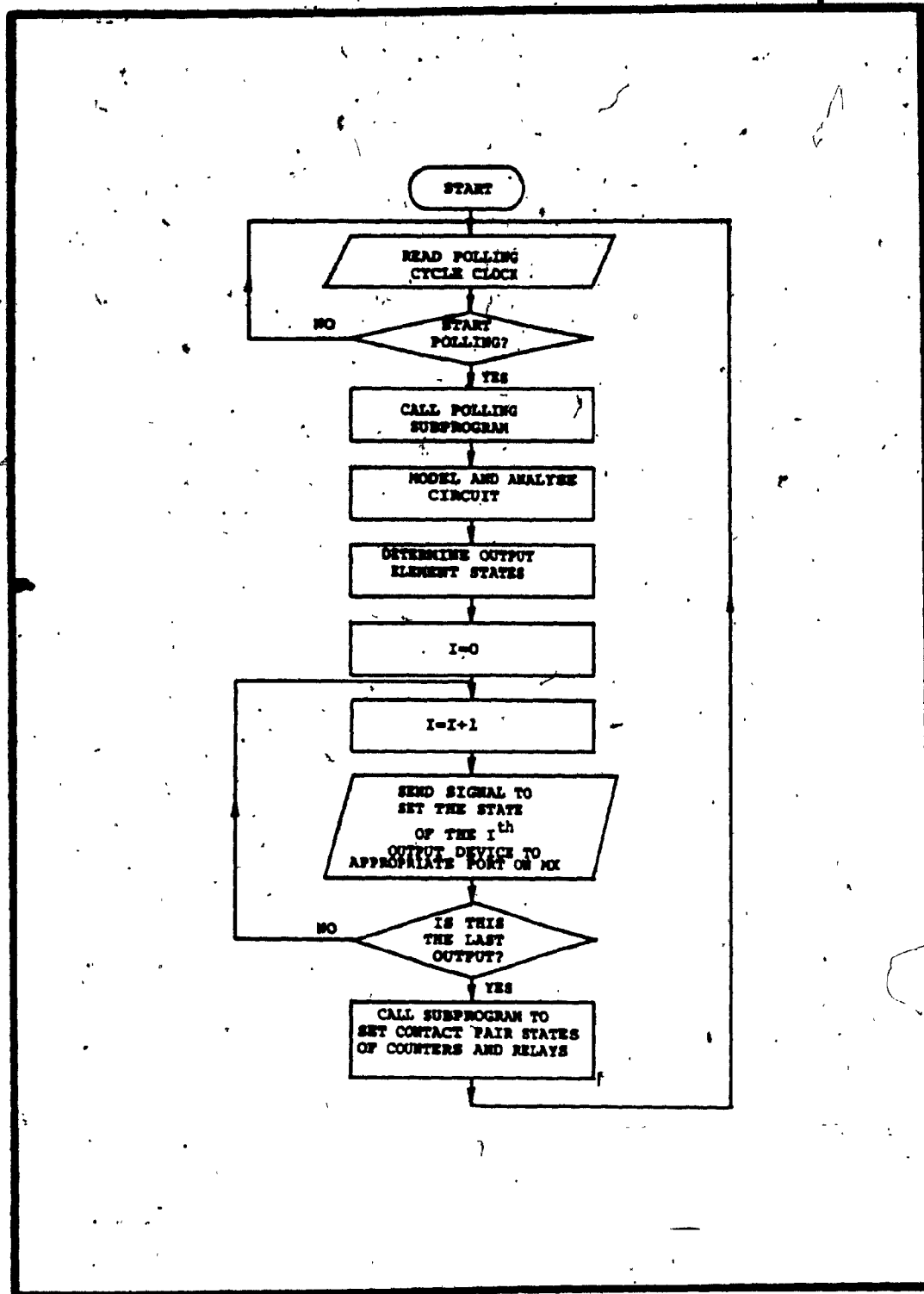


Fig.7.5 Sequence of operations executed in automatic mode.

process through the multiplexer.

The period of the polling clock, denoted T_p , must satisfy the condition:

$$T_p \geq E_T \quad (7.2)$$

where E_T is the maximum amount of time required to scan the inputs, analyse the CIRCUIT, and set the output state of a CIRCUIT. Otherwise, the next determination of the output device states will be based on data which may not be valid. Given a value of E_T , the value of T_p can be selected, and the hardware required to implement the polling clock, designed.

Once the frequency of the polling clock has been established, the performance characteristics of the CPLC can be expressed. First, the minimum persistence of a change in state of an input is T_p . Any input state change of a lesser duration may not be recognized as it may not be present when the signal to latch the input states is sent. Second, the minimum time duration between successive events must be T_p . Otherwise, two events will be input during the same polling clock cycle and will be looked upon as if they occurred simultaneously.

7.6 Summary

A conceptual design for a microcomputer based PLC has

been proposed. This design calls for the use of either of the analysis methods described in Chapters 3 and 4, as well as for the design of a multiplexer and of an array of hardware clock cycle counters.

The performance of the CPLC will depend, to a great extent, upon the hardware configuration of the multiplexer. As the number of ports increases, the complexity of the software also increases, resulting in a subsequent decrease in the time resolution of inputs to the CPLC.

Judging from the performance of the CSP, it can be inferred that a microcomputer based programmable controller of the type described above, can be used to realize any CIRCUIT configuration which conforms to the constraints given in Chapter 3. The implementation, and use, of the Graph Theory method described in Chapter 4 is likely to result in an even faster operation of the CRP. Due to the largely software implementation however, the frequency of the inputs of the processes which the CPLC can be used to control will be limited to about 3 Hz.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

Whether a Relay Logic Circuit (CIRCUIT) is implemented using electromechanical relays or a programmable logic controller (PLC), its commissioning will inevitably involve thorough testing. Only after the correct operation of a CIRCUIT has been verified for all realistic operating conditions can the implementation of a design be considered complete.

A CIRCUIT implemented using electromechanical relays must be built and tested either on-site, or by otherwise simulating inputs to, and outputs from it. This is both costly and time consuming. The testing of a CIRCUIT implemented by a PLC, on the other hand, is somewhat simpler because many PLCs provide facilities for simulating the logic operation of a CIRCUIT. Typically, this involves having the operator set the input state using mechanical switches. For every input state specified, an array of indicator lamps displays the output state.

In spite of the advantages they provide, PLC simulators have several shortcomings. First, they do not provide for the simulation of the timing (a facility which is essential for identifying race states). Second, and more important, there is no documented, analytical justification

supporting the validity of the methods used to determine the output state of a CIRCUIT. Without such a basis, the results of a simulation conducted using a PLC are doubtful, particularly in applications where the safety of personnel or equipment is involved.

In this Thesis, two distinct methods for analysing CIRCUITS have been described. Both of these methods correctly determine the output state. Furthermore, they identify the devices in the CIRCUIT which carry current. This information can be used to produce a display showing the current paths in a CIRCUIT (both forward- and backward-directed) so as to provide useful CIRCUIT diagnostics.

The first method models a CIRCUIT as a resistive network and a voltage source. The network comprises unit-valued and infinite-valued resistors which model devices having transmission values of 1 and 0 respectively. The output state of the CIRCUIT can be determined by establishing whether or not there is a potential difference (PD) across each resistor that models an output device (or memory). If there is, the output device (or memory) is "energised". Otherwise, it is "off".

In order to establish the existence of a PD across a resistor, the voltage at each "node" in the model is evaluated as a fraction of the value of the voltage source of the model. By calculating the difference between the values found across the terminals of each unit-valued

resistor and by comparing this difference to some threshold value, the existence, or absence, of a PD is established.

Having obtained the values of the voltages at each node in the model, it is possible to also identify the other current-carrying devices in a CIRCUIT. Any unit-valued resistor across which there is found to exist a PD will be carrying current in the CIRCUIT. For CIRCUIT models containing balanced "bridges" however, this method will fail to identify some devices contained in redundant paths of logical transmission. This shortcoming is of little consequence however, because it does not, under any circumstances, lead to the erroneous determination of the output state. Furthermore, such a situation will rarely arise.

The second method described in this Thesis models each device having a transmission value of 1 as an "edge" of a "graph". By systematically "exploring" such a graph, it can be determined whether there is a path between the voltage source of the CIRCUIT and ground. Since such a path must include an output device, that device will be "energised". Furthermore, if the "elementary" paths from the voltage source to ground (of the CIRCUIT) can be enumerated, then it can be concluded that the devices modeled by the edges in these paths will be carrying current.

In order to explore the graph, the CIRCUIT Analysis Algorithm (CAA) is used. This is a modified version of the

"depth-first search" method for exploring graphs. By applying the CAA using each of the "source vertices" of the graph as "roots", all the elementary paths containing a source vertex can be enumerated. As a result, the output state can be determined.

Although the second method identifies all the elementary paths, it also includes some spurious paths which cannot be distinguished from the elementary paths. This shortcoming however, will never cause the incorrect determination of the output state and is therefore inconsequential.

A computer program for the (interactive color graphic configuration and simulation of CIRCUITS (CSP) has been developed on the basis of the first of the two methods described above (the development system is described in Section 5.2). It allows the operator to specify an input state by means of dedicated keys on a data terminal and to immediately observe the output state on the graphics screen. Important information, such as the identification of current-carrying devices and energised outputs, is highlighted on the display of the Ladder Diagram. Furthermore, the CSP is capable of simulating the timing characteristics of CIRCUITS. Using these facilities the operator can identify race states as well as locate hazards.

The application of the CSP to the simulation of several test CIRCUITS shows how the CSP allows the operator

to initiate and observe any state transition and to accurately simulate its timing characteristics. This facility provides a tremendous advantage over traditional, tabular methods, because it can be used to analyse CIRCUITS with a large number of inputs and outputs.

Once a CIRCUIT has been specified using the configuration facilities of the CSP, all that is required to perform a simulation is that the operator has in mind the transitions he wishes to observe. By bringing the CIRCUIT to the desired "steady" state and by specifying an input change which will initiate the desired state transition, the operator can advance the CIRCUIT in simulated time and observe the various transient states.

Either one of the methods described in this Thesis can be used to design a general purpose, microcomputer-based PLC. A conceptual design of such a PLC, the CPLC, has been presented. The CPLC is intended primarily for use as a tool for teaching the use and the operating characteristics of PLCs. Since there is no need for industrial "packaging" in such an application, and assuming the availability of a microcomputer, the CPLC can be implemented simply by loading the CPLC computer programs from mass storage and by connecting the I/O interface. In spite of its relatively low projected cost, the CPLC would provide all the advanced features of the more powerful PLCs and would, in addition, provide access to the CPLC programs for user modification.

The two analysis methods presented in this Thesis

provide, for the first time, a documented, analytical foundation for the determination of the output state of a CIRCUIT. Moreover, either method can be used to identify the current-carrying devices in a CIRCUIT in order to generate a display of the current paths.

8.2 Future Work

The analysis methods presented in this Thesis are both capable of identifying the current-carrying devices in the general CIRCUIT. They both have their shortcomings however. The first method uses double precision arithmetic and will omit some current-carrying devices. The second method creates spurious paths comprising devices which do not carry current.

With regard to the method for analysing a CIRCUIT as a resistive network, if the minimum possible PD across a resistor can be evaluated, or if its order of magnitude can be analytically established, it would be possible to determine whether the use of double precision arithmetic is necessary. If not, the memory requirements for the implementation of this method would be halved. Furthermore, the analysis could be performed more quickly.

The CAA method, on the other hand, requires more work to be done to modify it in order to enable it to reject the spurious paths it generates. If such an improvement could be made, it would be possible to completely specify all the

current paths, and therefore all the paths of logical transmission in a CIRCUIT.

In an internal report, Waddington and Wild [42] indicate that there is a need for new methods to generate fault trees. By employing the CSP for determining the paths of logical transmission in a CIRCUIT, and by combining this with a reliability analysis, it may be possible to develop an interactive fault tree analysis system which would operate much along the same lines as the CSP.

LIST OF REFERENCES

- [1] Pluhar, K., "Electromechanical Relays - They'll be with us for a long time", Control Engineering, Vol. 28, No. 11, October 1981, pp. 84-86.
- [2] Krieger, M., "Basic Switching Circuit Theory", Macmillan, New York, 1967, pp. 96-164.
- [3] Cheng, R.M.H., "Computer Simulation of Fluid Control Circuits", Fluidics Quarterly, Vol. 10, No. 2., April 1978, pp. 57-70.
- [4] Anon., "Descriptive Bulletin 16-354(E): NUMA-LOGIC 700 and 900 Series Programmable Controllers", p. 10., Sept. 1982. Westinghouse Canada, Inc.
- [5] Masur, Mike, Private Communication, June 1983. (Westinghouse Canada Ltd.)
- [6] Anon., "Programmable Controllers-Microprocessors in Overalls", Mechanical Engineering, Vol. 104, No. 2., February 1981, pp. 26-31.
- [7] Morris, H.M., "Fast Growing PC Market Encourages Wide Range Of Product Offerings", Control Engineering, Vol. 30, No. 1, January 1983, pp. 57-61.

- [8] Anon. "Who Makes What in PCs - Programmable Controllers Bring Advanced Technology To Industrial Users" Canadian Controls and Instruments, Feb. 1980, pp.16-20, 22, 24, 26.
- [9] Gele, C., Mansion, D., "Panaorama Des Automates Programmables: 150 Modeles Presentes", Le Nouvele Automatisme, (2 Parts) No. 9, Nov. 1979, pp.29-38 No.10, Dec. 1979, pp.29-45.
- [10] Mennie, D., "Programmable Controllers Put Microprocessors, IC Memories To Work In Industry", Electronic Design, Vol. 27, No. 14, July 1979, pp.38-41.
- [11] Laduzinsky, A.J., "Peripherals Enhance PLC Performance", Control Engineering, Vol. 30, No. 1, January 1983, pp.63-66.
- [12] Miller, T.J., "New PLCs Exemplify Trends To Smaller Size And Distributed Network Compatibility", Control Engineering, Vol. 30, No. 1, January 1983, pp.49-51.
- [13] Coughlin, V., "Selecting An Intelligent PC", Design Engineering, April 1981, pp.69-72.
- [14] Rusch, B.R., "The Future Of Programmable Controllers",

Actuator Systems, August 1981, p.14.

[15] Op.cit., [7], p.57.

[16] Ibid.

[17] Penz, D.A., "Organizing a PC Software Development",
Instruments And Control Systems, Part I in Vol 55, No.
2, Feb. 1982, pp.53-57, Part II in Vol. 55, No 3,
Mar. 1982 pp.73-76.

[18] Baker, A.T., Raho, R.J., "Programmable Controllers:
Systems Of Choice For Power Plant Wastewater
Treatment", In Tech, Vol. 28, No. 9, Sept. 1981,
pp.51-54.

[19] Gardner, R.E., "User Software For The Programmable
Logic Controller", Instrumentation Technology, May
1975, pp.33-36.

[20] Schalach, M.K., "Operator Interfaces For Programmable
Controllers", In Tech, Vol. 28, No. 9, Sept. 1981,
pp.63-65.

[21] Fox, P.J., "The Choice Of Programming Language Applied
To Programmable Controllers", Automation, Vol 12.. No.
11-12 Nov-Dec 1978., pp.7-12.

[22] Struger, O.J., Christensen, J.H., "Languages For Programmable Controllers", Conference Record Of The Industry Applications Society, IEEE-IAS-1982, Annual Meeting San Francisco, CA., U.S.A., 4-7 Oct. 1982 (NY, U.S.A. IEEE, 1982.)

[23] Unger, S.H., "Asynchronous Sequential Switching Circuits", John Wiley & Sons, New York, 1969, pp.64-113.

[24] Kelly, M., Lequoc, S., Cheng, R.M.H., "A Simulation Program For Logic Relay Circuits In Process Control", ISA/81 Industry-Oriented Conference And Exhibit, St. Louis, Missouri, April 6-9, 1981.

[25] Kapps, C., "Development Of A Logic Simulator For A Small Computer System", Ninth Semi-Annual ATE Seminar/Exhibit, Automated Testing For Electronics Manufacturing Proceedings, Boston Mass., U.S.A, 8-11 June 1981, Bewill Publishing 1981.

[26] McDermott, R., "Simulation Of Simple Digital Logic Through A Computer Aided Design System", Byte, Vol. 8, No. 1, January 1983: pp.396, 398, 402, 404, 406, 408, 410, 412, 414.

[27] Cheng, R.M.H., Lequoc, S., Athanasoulas, D.,

"Simulation Of Logic Circuits In Process Control Using Interactive Computer Graphics", Proceedings of the American Control Conference, San Francisco, California, June 1983, pp.18-21.

[28] Op.Cit., [2], p.90.

[29] Op.Cit., [2], p.96.

[30] Johnson, L.W., Reiss, R.D., "Numerical Analysis", Addison Wesley Publishing Company, Reading Mass., 1977., pp.50-61, 108.

[31] Fox, L., "An Introduction To Numerical Linear Algebra" Clarendon Press, Oxford, England, 1964, p.79.

[32] Cook, R.D., "Concepts and Applications of Finite Elements Analysis", John Wiley & Sons, New York, 1974, p.45.

[33] Op.Cit., [22], p.20.

[34] Op.Cit., [22], p.20.

[35] Anon., "Control Relays", G & W Eagle Signal Industrial Controls, Relay Catalogue, March 1981.

[36] Johnson, D. E., Johnson, J. R., "Graph Theory with Engineering Applications", The Ronald Press Co., New York, 1972.

[37] Mayeda, Wataru, "Graph Theory", John Wiley & Sons, New York, 1972.

[38] Op.cit., [36], p.24.

[39] Swamy, M. N. S., Thulisiraman, K., "Graphs, Networks, and Algorithms", John Wiley & Sons, New York, 1981, pp.449-453.

[40] Tarjan, R., "Depth-first Search and Linear Graph Algorithms", SIAM Journal Comput., Vol.1, No.2, June 1972, pp.146-160.

[41] Boom, H., Private Communication, June 1982.

[42] Waddington, J. G., Wild, A., "The Fault Tree as a Tool in Safety Analysis in Nuclear Power Plants", Atomic Energy Control Board, Ottawa, Canada, June 10, 1981, INFO-0036.

APPENDIX A
DESCRIPTION OF A BANDED FORMAT GAUSSIAN
ELIMINATION EQUATION SOLVER

Figure A.1 shows the program segment for performing GAUSSIAN elimination on a system of equations stored in band form, as given by Cook [32].

Referring to Fig.A.1, array S contains the band form of the admittance matrix and the array R stores the input vector. As the entries of the solution vector are calculated by back-substitution, they replace the corresponding row entries of R. The DO 790 and DO 830 loops treat each equation (i.e. row of S). The DO 780 and DO 820 loops substitute the equation of the current row into the subsequent rows in the active portion of the admittance matrix, as shown in Fig.A.2. Finally, the DO 750 loop treats each term in the subsequent row.

This program segment is intended for use in applications where the admittance matrix remains fixed while the solution is obtained for various inputs (stored in R). In solving the model equations (Chapter 3), the admittance matrix will change every time there is a change in the input state or internal state. Hence, the DO 830 and DO 820 loops of Fig.A.1 can be incorporated into the DO 780 loop. By implementing this modification, the program length is reduced slightly.

The modified version of the Gaussian elimination

```

C FORWARD REDUCTION OF MATRIX (GAUSS ELIMINATION)
700 DO 790 N=1,NSIZE
DO 780 L=2,MBAND
IF (S(N,L) .EQ. 0.) GO TO 780
I = N + L - 1
C = S(N,L)/S(N,I)
J = 0
DO 750 K=L,MBAND
J = J + 1
750 S(I,J) = S(I,J) - C*S(N,K)
S(N,L) = C
780 CONTINUE
790 CONTINUE
C FORWARD REDUCTION OF CONSTANTS (GAUSS ELIMINATION)
800 DO 830 N=1,NSIZE
DO 820 L=2,MBAND
IF (S(N,L) .EQ. 0.) GO TO 820
I = N + L - 1
R(I) = R(I) - S(N,L)*R(N)
820 CONTINUE
830 R(N) = R(N)/S(N,I)
C SOLVE FOR UNKNOWN BY BACK-SUBSTITUTION
DO 860 M=2,NSIZE
N = NSIZE + 1 - M
DO 850 L=2,MBAND
IF (S(N,L) .EQ. 0.) GO TO 850
K = N + L - 1
R(N) = R(N) - S(N,L)*R(K)
850 CONTINUE
860 CONTINUE

```

Fig.A.1 Program segment for performing Gaussian Elimination on a matrix stored in band form [32].

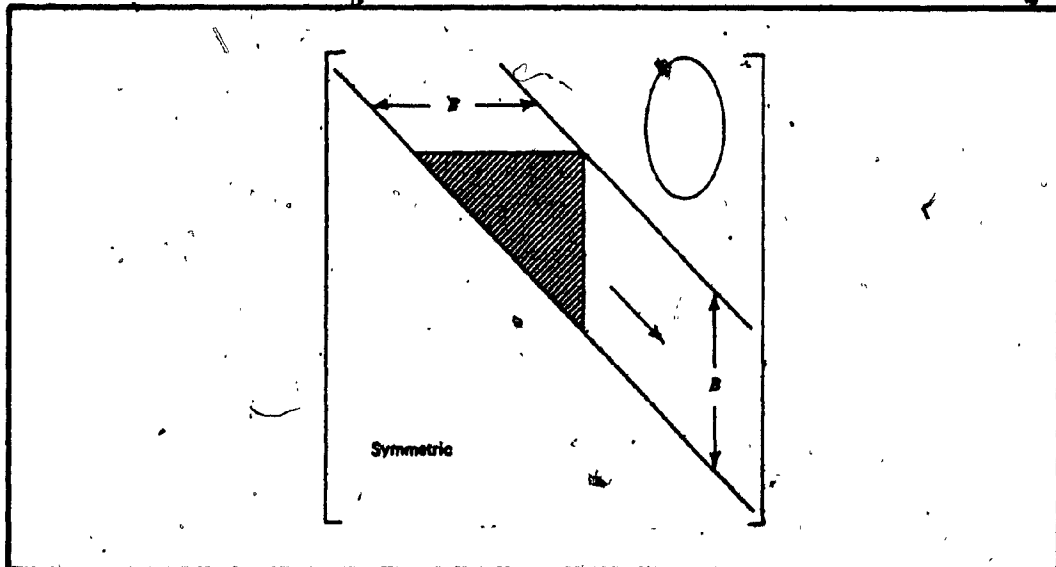


Fig. A.2 Active portion of the admittance matrix.

equation solver (GAUSS) is given in Fig.A.3. Another modification from the original program is the decision to go to the next row if the "diagonal" entry in the admittance matrix is 0. (line 2 of GAUSS). This is necessary since, if the diagonal entry of G_B is 0, then entries in the row and column containing that diagonal entry will also be 0..

The number of "operations" is given by COOK [32] to be:

$$C_{op} = (NB'^2) / 2 \quad (A.1)$$

Because "operation" represents one multiplication or division plus one addition or subtraction, the total number of operations is given by:

$$C_T = NB'^2 \quad (A.2)$$

where, N is the number of rows in the admittance matrix and where B' is the semi-bandwidth.

```
      IMPLICIT INTEGER*2 (A-Z)
      REAL*8 S(50,7), R(50), C
      DO 3 N = 1, NSIZE
      IF( S(N,1) .EQ. 0.0 ) GO TO 3
      DO 2 L = 2, MBAND
      IF( S(N,L) .EQ. 0.0 ) GO TO 2
      I = N + L - 1
      C = S(N,L) / S(N,1)
      J = 0
      DO 1 K = L, MBAND
      J = J + 1
1      S(I,J) = S(I,J) - C * S(N,K)
      S(N,L) = C
      R(I) = R(I) - S(N,L) * R(N)
2      CONTINUE
      R(N) = R(N) / S(N,1)
3      CONTINUE
      DO 7 M = 2, NSIZE
      N = NSIZE + 1 - M
      DO 6 L = 2, MBAND
      IF( S(N,L) .EQ. 0.0 ) GO TO 6
      K = N + L - 1
      R(N) = R(N) - S(N,L) * R(K)
6      CONTINUE
7      CONTINUE
```

Fig.A.3 Modified version of Gaussian Elimination
equation solver (GAUSS).

APPENDIX B
SHORT GLOSSARY OF THE GRAPH THEORETIC

BRANCH - An edge of a tree is called a "branch".

CIRCUIT - A "path", the initial vertex of which, v_0 , is the same as its terminal vertex v_n (ie., $v_0 = v_n$), is called a "circuit".

CLOSED PATH - Circuit.

CONNECTED - An undirected graph is said to be "connected" if every edge is connected to every other edge by a path.

ELEMENTARY PATH - A path is said to be "elementary" if all its vertices are distinct (all its edges must also be distinct).

GRAPH - A graph $G = (V, E)$ is a set V of "vertices" together with a set E of "edges", which are pairs of V . G is null if V and E are empty.

INCIDENT - An edge $e = (a, b)$ is said to be "incident" with each of its vertices and vice-versa.

PATH - A "path" is a finite sequence of (not necessarily distinct) edges e_1, e_2, \dots, e_n where $e_i = (v_{i-1}, v_i)$, $i = 1, 2, \dots, n$ and where each v_i is a vertex. The vertex v_0 is called the "initial" vertex and v_n is called the "terminal" vertex.

SELF LOOP - A "self loop" is an edge, $e = (a, b)$ where $a = b$.

SPANNING TREE - A "spanning tree" is a tree which contains all the vertices of a graph.

TREE - A tree is a connected graph which has at least two vertices and contains no circuits.

APPENDIX C

STORAGE REQUIREMENTS OF THE CAA

Since the CAA has not been coded, it is difficult to predict the amount of memory it would require. It is possible however, to determine precisely the amount of memory required to store the working arrays.

The first working variable, DFN, stores one integer number for each vertex in the graph. An INTEGER*2 variable type is sufficient to store any DFN number which may be generated. FATHER requires two bytes for each vertex; one to store the row index, and another to store the column index of the vertex.

The variable, GRAPH_NUM, is stored for each edge in the graph and requires 1 byte/edge. Finally, the direction assigned during the DFS to each edge also requires 1 byte/edge. In addition to these requirements, an array of 100 bytes is required for Graph_stack.

Hence the total memory requirement (number of bytes) for storing the working variables of the CAA is given by:

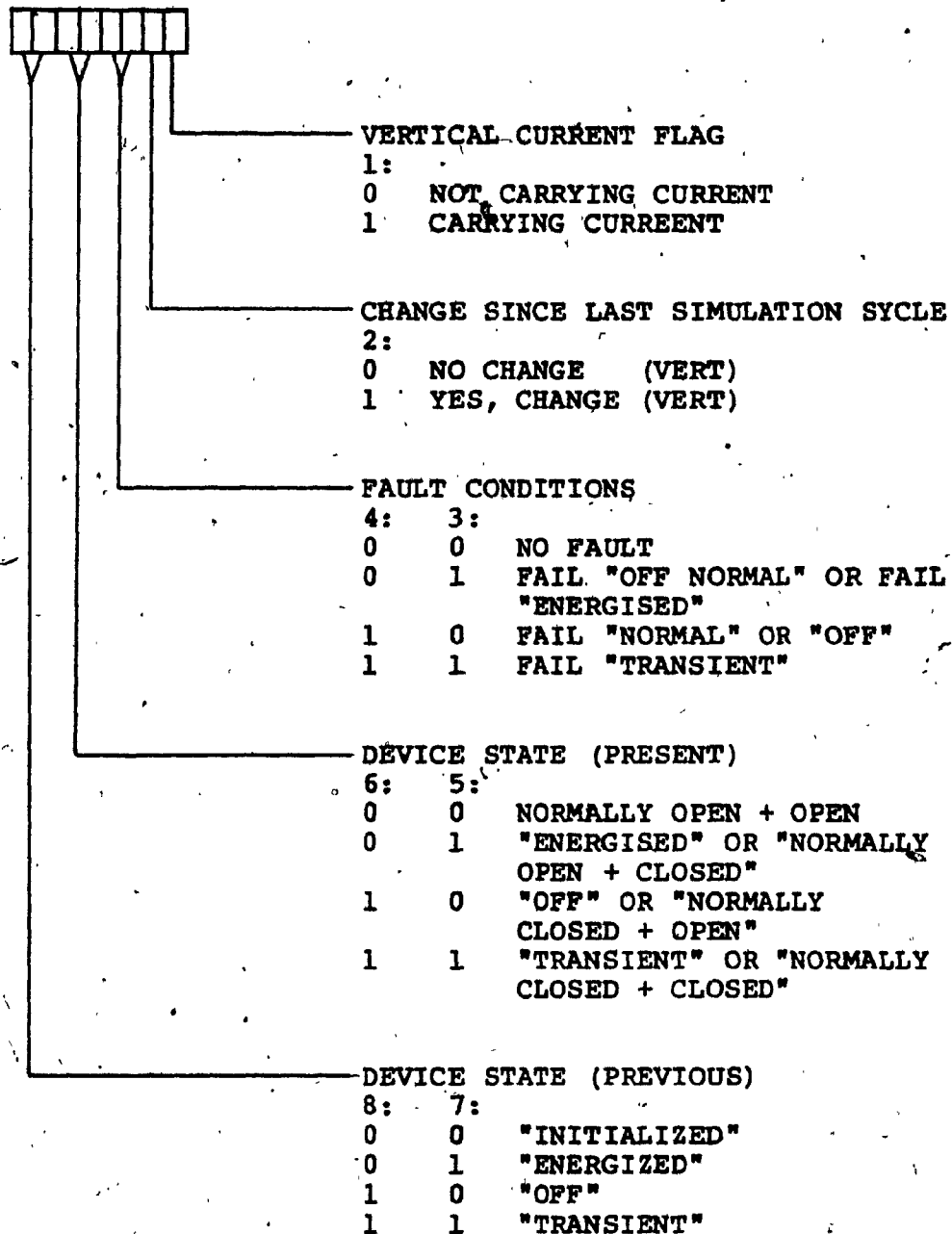
$$\begin{aligned} \text{MEM}_{\text{graph}} &= 2(\text{number of edges}) + \dots \\ &\dots + 4(\text{number of vertices}) + 100 \end{aligned} \quad (\text{C.1})$$

From (4.1) and (4.2),

$$\text{MEM}_{\text{graph}} = N_T(8M+10) - 2M + 100 \quad (\text{C.2})$$

APPENDIX D DATA BASE ATTRIBUTES OF THE CSP

DATA STRUCTURE FOR "AW"



DATA STRUCTURE FOR "BW"



DEVICE NAME CODE

5: 4: 3: 2: 1:

X X X X X

CODE REFERS TO ARRAY
CONTAINING DEVICE

NAMES OF VARIOUS DEVICES ALONG WITH
GRAPHICS DATA BASE ADDRESS AND
MAXIMUM NUMBER OF CONTACTS (FOR
INPUT DEVICES)

CHANGE OF STATE SINCE LAST SIMULATION
CYCLE

6:

0 NO CHANGE OF STATE SINCE SIMULATION
CYCLE

1 DEVICE HAS CHANGED STATE SINCE LAST
SIMULATION CYCLE

VERTICAL CONNECTOR

7:

0 NO VERTICAL CONNECTOR

1 VERTICAL CONNECTOR

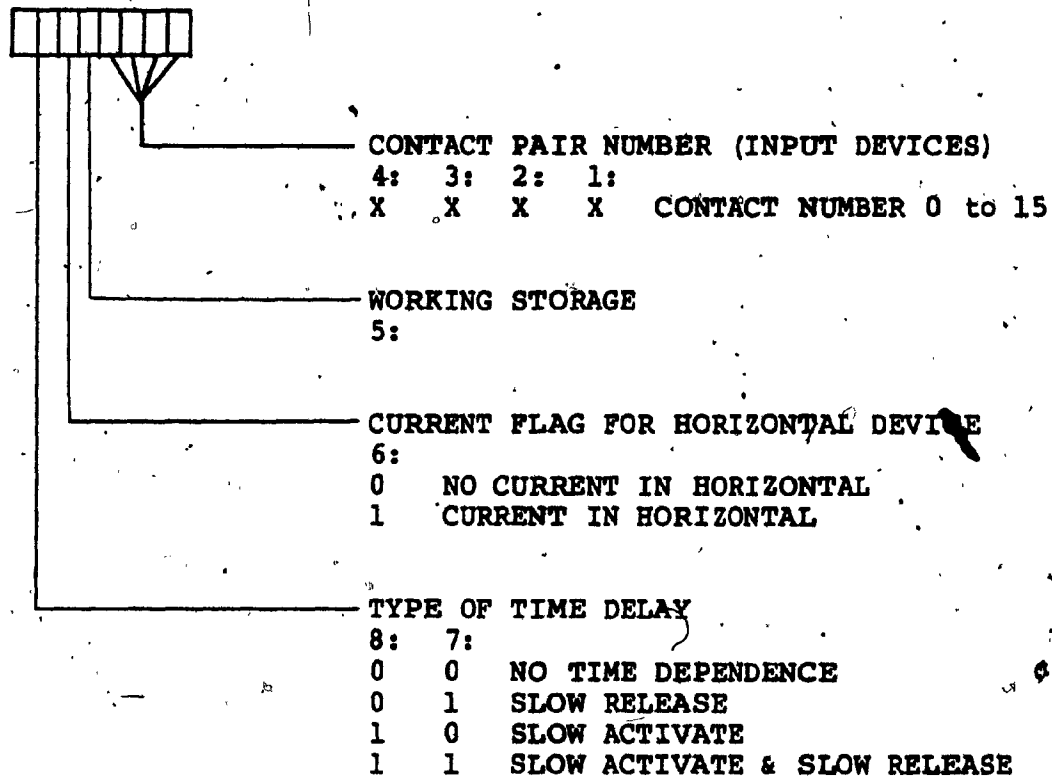
ANNUNCIATION MESSAGE

8:

0 NO ANNUNCIATION

1 ANNUNCIATION

DATA STRUCTURE FOR "CW"

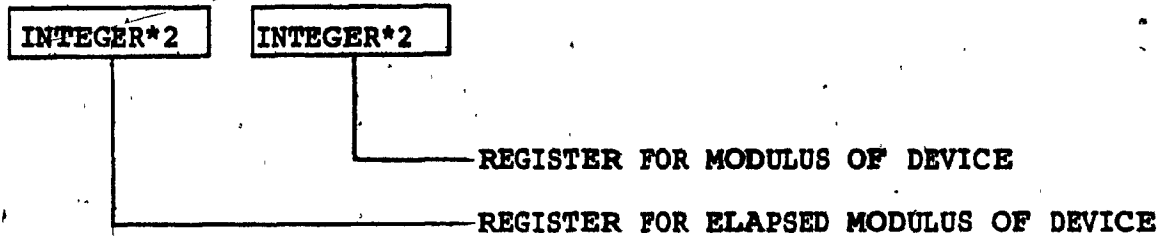


DATA STRUCTURE FOR "EW"



NOTE: THE PROGRAM WILL ACCEPT ANY 5-CHARACTER STRING WHICH DOES NOT INCLUDE LOWER CASE ALPHA CHARACTERS. ON INPUT, THIS STRING IS ENCODED AS AN INTEGER AND IS STORED IN ARRAY EWORD. EWORD CAN BE DECODED FOR OUTPUT WHENEVER NECESSARY.

DATA STRUCTURE FOR "FW"



DATA FILE FOR ANNUNCIATION MESSAGES

"ANUN.DAT"

THIS DATA FILE IS TO CONTAIN 4, 30 CHARACTER ANNUNCIATION MESSAGES FOR EACH CIRCUIT ELEMENT. SPACE IS RESERVED FOR THESE MESSAGES WHETHER OR NOT A MESSAGE IS GIVEN FOR A PARTICULAR ELEMENT. THE MESSAGES ARE STORED AS FOLLOWS:

- 1) SWITCH OPEN MESSAGE
- 2) SWITCH CLOSED MESSAGE
- 3) FAULT MESSAGE
- 4) PROCEEDED MESSAGE

IN ORDER TO ACCESS THE SET OF MESSAGES FOR THE ELEMENT IN (ROW, COLUMN)

$$\text{COUNT} = (\text{ROW}-1) * 7 + \text{COL}$$

RECORD "COUNT" IN ANUN.DAT WILL ACCESS THE FIRST MESSAGE,
COUNT +1, +2, +3 WILL ACCESS THE FIRST MESSAGE. COUNT +1,
+2, +3 WILL ACCESS THE 2ND, 3RD & 4TH MESSAGES RESPECTIVELY.

