A Study of Non-interactive Computer Methods for Microcircuit Layout

P F A Reilly

Thesis submitted for the degree of Doctor of Philosophy
of the University of Edinburgh in the Faculty of Science

March 1974

ABSTRACT

The conception, creation and development of a non-interactive layout aid for microcircuit design is described. The system is based on a review of existing layout techniques and the requirements of low cost and size for mounting on a timesharing bureau, using remote teletype and graph plotter for input and output; the programs are written wholly in FORTRAN IV. It is designed for general application throughout the range of microcircuit technologies, with a slight bias towards MOS integrated circuits. A degree of implicit manual control is built into the programs which allows the production of a range of trial layouts with little additional effort. Resulting layouts, obtained in several technologies, are presented and these form the basis of the study.

The system was found to perform adequately and to provide substantial assistance in microcircuit layout. The results demonstrate that effective layout aids do not necessarily require expensive hardware or a dedicated computer. Emphasis is placed on critical aspects of converting a manual layout process in order to incorporate computer aided design. An alternative method of approach to the design of layout aids is outlined, which offers advantages over present techniques.

ACKNOWLEDGMENT

The research work for this thesis was carried out in the Wolfson Microelectronics Liaison Unit, supported by a grant from the Wolfson Foundation. I am indebted to my supervisors, Jim Murray, who created the opportunity for this work and provided invaluable advice and guidance, Professor W.E.J. Farvis, who provided encouragement and criticism at crucial stages of the work, and John Gray for many valuable discussions. I would also like to thank many colleagues in the Department of Electrical Engineering and the Computer Aided Design Project of the Department of Computer Science for assistance, advice and criticisms. Special thanks are due to Miss Jean Clarke who typed and corrected the manuscript.

I am grateful to my parents, who provided encouragement, sympathy and support. Finally, Sharyn, assisted by Tamsyn, who encouraged, cajoled, commisserated, criticised and proof-read in addition to providing essential help and without whom none of this would have been possible.

TABLE OF CONTENTS

CHAPTER 0 INTRODUCTION

0.0 Microcircuits 1

0.1 Circuit Fabrication 1

0.2.0 Circuit Layout 4

0.2.1 Layout Aims 5

0.3 The Thesis 7

CHAPTER I LAYOUT AIDS - PHILOSOPHY AND BASIS 8

1.0.0 Manual Aids 8

1.0.1 The Computer and Layout 8

1.0.2 Computer Layout Aids 9

1.1.0 GAELIC - a non-automatic aid 10

1.1.1 Errors 11

1.2.0 Interaction and the Graphics Screen 13

1.2.1 Data Structures 13

1.2.2 Ring Data Structures 15

1.2.3 The Gaelic Data Structure 17

1.3 Interactive Layout 22

1.4 Automatic Layout 23

CHAPTER II REVIEW OF EXISTING AUTOMATIC LAYOUT METHODS 24

2.0.0 Introduction 24

2.0.1 Program Specifications 25

2.0.2 Interaction 26

2.1 Criteria of Optimality 28

2.2.0 Placement Techniques 30

2.2.1 Unconstrained Placement 30

2.2.2 Constrained Placement 33

2.3.0 Routing Techniques 42

2.3.1 Maze Running Algorithms 42

2.4.0 Topographical Techniques 55

2.4.1 Circuit Representations 55

2.4.2 Planarity Algorithms 61

2.4.3 Graph Synthesis and Analysis Techniques 63

2.5 Conclusion 76

CHAPTER III STUDY STRATEGY AND THE TOPOLOGICAL ANALYSIS 79

3.0 Statement of Initial Aims 79

3.1.0 Program Aims and Capabilities 80

3.1.1 Circuit Input 81

3.1.2 Program Specifications 84

3.2 Method of Approach 86

3.3.0 Description 88

3.3.1 Initial Permutation Construction 93

3.3.2 Optimisation 95

3.4 Worked Example 97

3.5 Discussion 100

3.6.0 Applicability to Electrical Circuits 102

3.6.1 Component to Link Mapping 103

3.6.2 Component to Node Mapping 105

3.6.3 Net Representation 108

3.7.0 Implementation 113

3.7.1 Graph Input 116

3.7.2 Initial Permutation Creation 118

3.7.3 Permutation Optimisation 123

3.8 Modifications 125

.3.9.0 Results 128

3.9.1 Complete Graphs 128

3.9.2 Symmetric Utility Graphs 131

3.9.3 Random Networks 132

3.10 Net Representation Results 133

3.11 Conclusions 141

CHAPTER IV PLACEMENT OF COMPONENTS 142

4.0 Layout Realisation 142

4.1.0 Layout Execution 143

4.1.1 Partitioned Placement 144

4.1.2 Placement based on the Permutation Description 144

4.2.0 Node-line Considerations 146

4.2.1 Polygonal Deformation 149

4.2.2 Spiral Deformation 149

4.2.3 Folded Configurations 152

4.2.4 Concentric Polygons 154

4.2.5 The Inadequacies of Predetermined Configurations 157

4.3 General Placement 157

4.4 The Placement Algorithm 159

4.5.0 Implementation - Data Requirements 167

4.5.1 Perimeter Calculations 172

4.5.2 •Segment Slection for Placement 177

4.5.3 Calculation of Separations 180

4.5.4 Final Positioning 183

4.6.0 Initial Results 184

4.6.1 Segment Selection Modifications 184

4.7.0 Results 189

4.7.1 Effects of Control Parameters 189

4.7.2 Conclusions 191

CHAPTER V ROUTING CONNECTIONS 192

5.0.0 Introduction 192

5.0.1 Optimisation 192

5.1.0 Problems of Routing 194

5.1.1 Contact Holes 198

5.1.2 Multi-component Nets 198

5.2.0 A Routing Algorithm 200

5.2.1 Link Sequencing 203

5.2.2 Sequencing Procedure 205

5.3 Implementation 207

5.4 Results 207

CHAPTER VI RESULTS 212

6.0.0 Evaluation Requirements 212

6.0.1 Evaluation Technique 213

6.1.0 Circuit (i) - MOS Shift Register Stage 214

6.1.1 Comparison 220

6.2.0 Circuit (ii) - MOS Clock Driver Cell 220

6.2.1 Comparison 227

6.3.0 Circuit (iii) - MOS D-type Bistable 227

6.3.1 Comparison 234

6.4.0 Circuit (iv) - Thin-film Multivibrator 234

6.4.1 Comparison 241

6.5.0 Circuit (v) - Printed Circuit Board 241

6.5.1 Comparison 249

6.6 Summary of Results 249

CHAPTER VII SYSTEM ANALYSIS AND DISCUSSION OF IMPROVEMENTS 251

7.0 Introduction 251

7.1.0 Software Limitations 251

7.1.1 The Topological Analysis 252

7.1.2 The Placement Algorithm 256

7.1.3 The Routing Algorithm 261

7.1.4 The Programs - A Summary 261

7.2.0 System Modifications 263

7.2.1 Circuit Elements - Omissions 263

7.2.2 Elemental Interactions - Omissions 266

7.3 System Adaptability 267

7.4 Economic Considerations 269

7.5 Conclusions 270

CHAPTER VIII CONCLUSIONS 271

8.0.0 Layout Aids - Summary 271

8.0.1 The Automatic Aid 271

8.0.2 The Interactive Aid 272

8.0.3 The Third Alternative 272

8.1 A Comprehensive Layout System 274

8.2 Summary 276

APPENDIX I 278

APPENDIX II 283

APPENDIX III 286

APPENDIX IV 289

GLOSSARY OF TERMS 293

REFERENCES 298

CHAPTER O

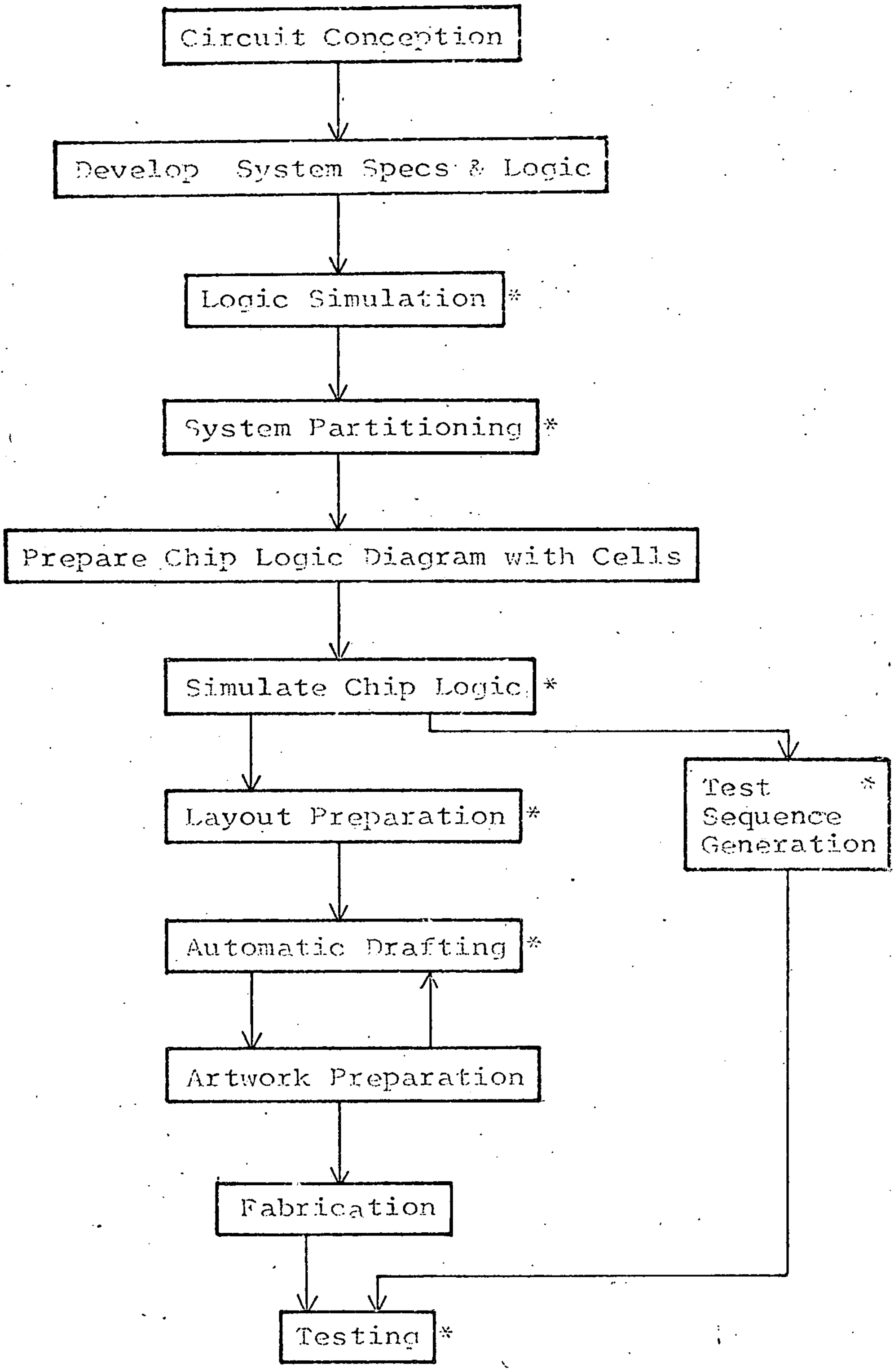
INTRODUCTION

O.O MICROCIRCUITS

Microcircuits were originally developed to gain technical and economic advantages through miniaturising equipment and so reducing the size and weight of installations. Improved reliability and cheaper production have led to the widespread acceptance and use of microelectronics, which has had considerable impact in the field of electronic equipment. Figure O.1 shows a flowchart of a typical microcircuit design process culminating in the translation of a circuit schematic into a layout suitable for fabrication. The layout preparation is the most time-consuming, and therefore expensive, of the sub-processes and is the major cause of bottlenecks in microcircuit production; for this reason computer-aided design (CAD) is of interest to many manufacturing firms as a means of reducing layout turnaround time (typically by about 80%) and thus, decreasing circuit development costs. Moreover, since computers are far less error-prone than humans, CAD also offers a more accurate layout method, which is a consideration of overriding importance to all microcircuit manufacturers.

O.1. CIRCUIT FABRICATION

In order to appreciate the significance of layout it is essential to have a knowledge of the processes and problems associated with microcircuit production. There are two



* ⇒ process can be computer aided.

Figure 0.1 Flowchart of Microcircuit Design.

principle classes of microelectronic circuitry, film and monolithic circuits.

Thick-film circuits are constructed by printing conductors and resistor tracks, using a silkscreen process, on to ceramic substrates. Thin-film circuits are fabricated by the vacuum deposition of passive components onto a finely polished glass, oxidised silicon or ceramic substrate; the required geometries are obtained by using metal deposition masks or photoetching complete films. Both thick and thin-film circuits generally require the manual addition of active components. Production is frequently based on the use of standard sized substrates, which means that a circuit layout is constrained by the substrate sizes available.

Monolithic or integrated circuits (ICs) are constructed in silicon substrates by diffusing dopants to control the type of filamentary volumes; surface insulation is normally achieved by oxidation. Inter-device connections and bonding pads are made with vacuum-deposited metal conductor tracks, which require etched contact holes in the insulation so as to make contact with the doped areas. Each process uses a photographic mask which defines the areas to be processed and the circuit function is defined by the inter-relationships of the patterns on the various masks. Up to 1000 circuits, or die, are simultaneously manufactured on a single silicon slice of fixed diameter; the die size is determined by the layout dimensions which therefore affects the production cost per circuit. In addition, random crystal defects affect a fixed number of circuits per slice and so layout size also controls the yield. Because ICs are so small, there are frequently undesired effects, caused by interactions between components, which detract from the circuit's performance. These parasitic effects can be minimised through judicious physical layout. However, this is a more complicated matter than simply physically isolating components because this expands the layout size which, in turn, increases cost.

Printed circuit boards (PCBs), although not micro-circuits, present layout problems similar to film circuits. They consist of metal conductor tracks, deposited or etched onto a thin insulating backboard, forming the conductor patterns for discrete components which are soldered onto the board. These circuits are included for consideration in this work to allow comparison and discussion of techniques which have been applied to PCB layout.

0.2.0 CIRCUIT LAYOUT

Layout is the process which converts a circuit schematic or logic diagram into the patterns used in circuit production. It may be viewed as three related subprocesses, the definition of individual cell geometries, the placement of cells on a substrate and the positioning of intercell and outside world connections. It is a lengthy process; for example, it can take many man-months, costing up to £50K, for a large integrated circuit, and many efforts have been made to decrease development costs through computer aids to layout. Two major divisions have occurred within these aids, resulting in the classifications interactive and automatic. Interactive layout aids assist the designer by employing a computer for housekeeping and rule checking functions, thus freeing the man for the creative design efforts. The flowchart shown in figure 0.2 shows a typical sequence of events in using an interactive layout system. Automatic aids differ from interactive ones in that they must replace the human factor rather than just assist it. This is clearly a much more demanding program since the computer must carry out pseudocreative tasks. In recent years there has been a rapid expansion of commercial, stand-alone interactive systems complete with software (Beardsley, 1971), while practically no commercial automatic aids have appeared. The reason for this is most probably the difference in programming problems, however several techniques of use in automatic layout have been developed and these are presented in chapter 2.

0.2.1 LAYOUT AIMS

Circuit layout deals mainly with the generation, modification and manipulation of patterns and the main aim is to fit all the required components and connections onto a minimal substrate area. The circuit manufacturer is chiefly concerned with circuit performance and cost which are inversely related, that is, a performance/price trade-off exists. A circuit designer must balance the two during layout; he must minimise parasitic and stray effects in addition to circuit area. This balance is achieved through largely intuitive techniques which cannot be translated into algorithmic procedures; as these are the basis of any computer program the implication is that a computer cannot hope to simulate a human designer's reasoning in layout.

It is accepted that the ideal microcircuit is planar

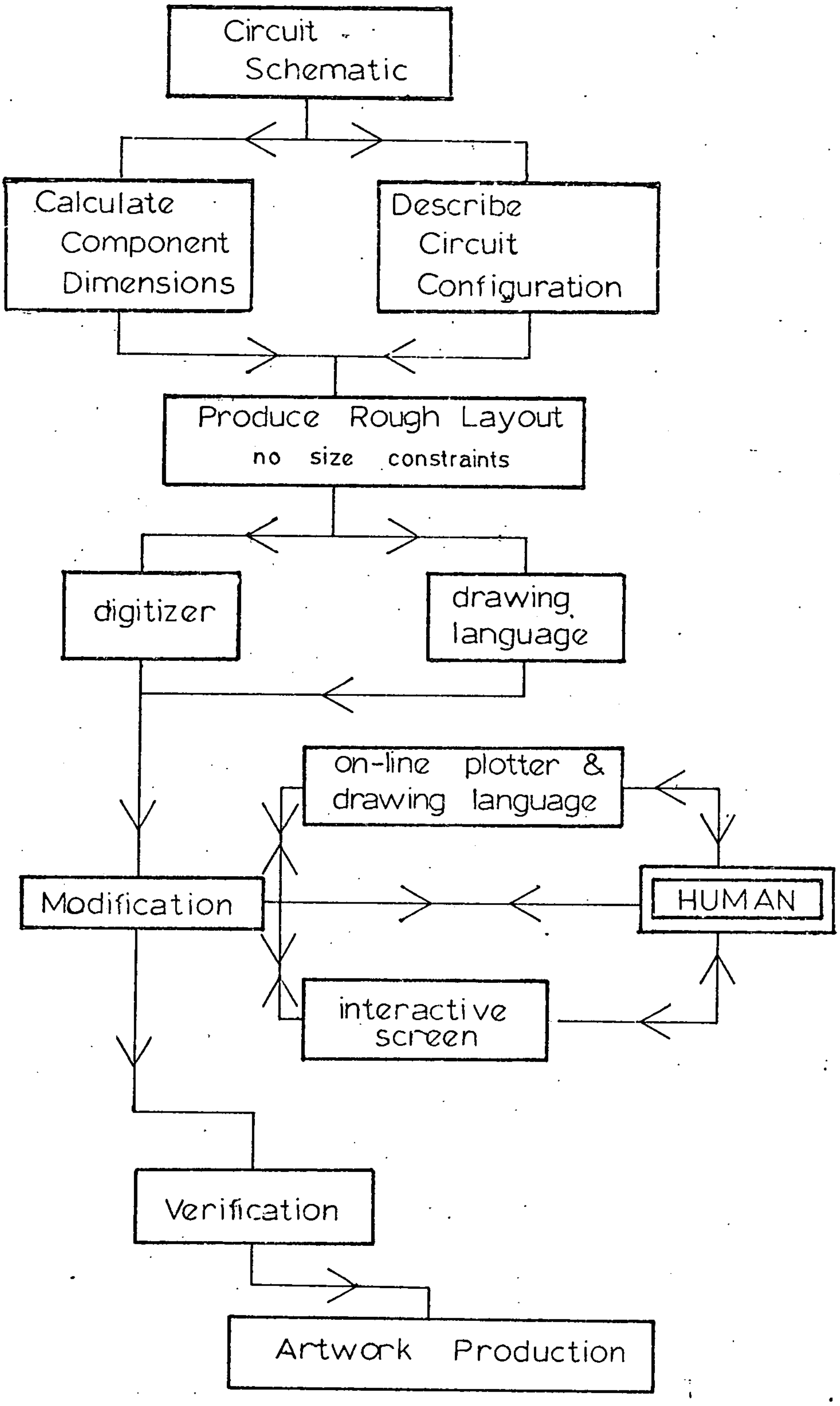


Figure 0.2 Flowchart of Microcircuit Layout using CAD.

since crossovers between conducting materials are a major source of parasitic effects. Although there are various techniques for incorporating crossovers so as to reduce deleterious factors, the best solution is undoubtedly to eliminate crossovers. However, the ideal, planar circuit is a rarity and part of the layout problem is to arrange a minimum number of crossovers in an optimal configuration. This is one area where CAD can be applied. However, the ideal layout program should be capable of much more than purely minimising crossovers.

0.3 THE THESIS

This work comprises an evaluation of automatic layout aids from the point of view of the small microcircuit manufacturer. There are two reasons for adopting this stance; first, it is a widely held opinion that CAD is becoming necessary if a firm is to remain competitive (Stone & Dietz, 1968; Hazlett, 1969; Mays, 1971; Beardsley, 1971; BAC report R84-C1, 1971); secondly, it is felt that effective CAD will require,in the near future,extensive capital commitment in excess of £100K. Obviously, small firms cannot readily afford such expenditure and so it is of importance to evaluate whether effective automatic layout software can be supported by inexpensive equipment.

CHAPTER I

LAYOUT AIDS - PHILOSOPHY & BASIS

1.0.0 MANUAL AIDS

Manual layouts are constructed on squared paper using pencil and ruler, this method requires erasure and redrawing of portions of the circuit when more space is required in a critical area. This is obviously time-consuming and tedious, especially when designing an integrated circuit which involves working with several different colours to represent the different masks. Cut-out shapes of basic components and, more recently, basic cells, were the first attempts at reducing the layout tedium; these can be moved without a massive redrawing effort since only the connection pattern has to be altered. However, working with pre-defined cells cuts down on the designer's freedom and many prefer to work from scratch rather than be restricted to creating space for cells of specified dimensions. Cutting masks manually from such a drawing is also slow and error prone so that manual methods have come to be regarded as unsatisfactory for modern design standards and the computer has offered the best alternative.

1.0.1 THE COMPUTER AND LAYOUT

The computer's power is a function of the sophistication of its circuitry and so there is a positive feedback system within CAD of microcircuits since computer aids produce better circuits which produces more powerful machines which, in turn, facilitate better aids. The first machine-based layout aids were confined to the improvement of the artwork generation process and have been so successful that, today, most IC artwork is automatically prepared. The problems associated with the construction of layouts are much more complex and it is only recently that machines capable of dealing with the sheer quantity and complexity of layout data have been developed. However, in the past decade the computer has come to take an increasingly important role in the field of microcircuit layout.

1.0.2 COMPUTER LAYOUT AIDS

The interactive graphics screen has provided a moveable, variable window through which a designer can draw, modify and delete parts of the layout with little effort. This has provided an alternative to the drawing board, ruler and pencil used in manual construction; the computer can easily simulate the operations of layout acting on instructions via a light-pen or keyboard and so eliminating much of the tedium of layout. However, interactive graphics requires expensive equipment and entails different techniques so that there are disadvantages to its use. A commercial interactive layout system is described in appendix I, together with a discussion of users' comments which gives an idea of the difficulties any manufacturing firm is faced with when installing an interactive layout system.

The CAD alternative to interaction is automation. This is still a young field and, before becoming too deeply involved in the minutiae of automatic layout, it is advisable to study the non-automatic aid in some depth to gain some insight into the basic problems associated with the application of digital computers to the field of micro-circuit layout.

1.1.0 GAELIC - A NON-AUTOMATIC AID

GAELIC (Graphics Aided Engineering Layout of ICs)(Eades, 1973) provides an ideal means for studying the philosophy and tools of layout aids. It began as a series of programs aimed at providing a more precise artwork preparation process. The system translates a circuit layout description, taken from a drawing, into a variety of outputs which can be used for preparing composite plots, micro-plots or photographic masks. It is designed for use over a remote teletype connected to a timesharing bureau which entails a minimum of expense.

The layout descriptions are input in one of two forms, a manual input language based on the CAMP language (J. Wood et alia, 1969) or a numerically coded tape prepared on a digitiser. The circuit is described in terms of shapes and layers corresponding to the patterns on the individual masks. This data is processed to produce a dump-code file which is a sequential numerical file adhering to the form of the original description,but suitable for rapid machine processing. As this file is created, syntax checks are made on the input data to identify any errors or anomalies which could cause confusion during latter stages. Next, the dump code is input to the compiler which produces a coordinate file which is in a form suitable for mask production; group calls are replaced by shape descriptions and information is associated by mask rather than area. From this coordinate file, various post processors are used to provide plots, for checking purposes, and drive-tapes for automatic mask-cutting machines. A flow chart of the system is shown in figure 1.1. The main drawback of this system is the lack of an on-line error correction process; the alteration of the input files is a slow and tedious procedure and greatly detracts from the efficiency of GAELIC as a layout aid.

1.1.1 ERRORS

Error detection and correction is an important part of all artwork preparation systems; the machine must be instructed to anticipate, and reject, any mistake which would cause confusion. Simple errors, such as mis-spelling key words or omitting key figures, are relatively easy to allow for, however errors involving violation of the design rules or data errors cannot be efficiently dealt with by the computer and require manual detection on an output plot. Once an error of the latter class is identified it must be corrected; this involves comparing the original drawing, identifying the exact nature of the mistake, its location in the input file and, finally, correction via editing. Thus current GAELIC error-control techniques are inefficient since they require skilled personnel to spend much time, utilising a fraction of their expertise, in checking drawings, altering input files and checking more drawings.

Obviously, it would be advantageous to use the computer to assist in this area by providing on-line error detection and correction facilities. Such a system would require a display of the layout in a form which could be understood by both man and machine, and a method of manually

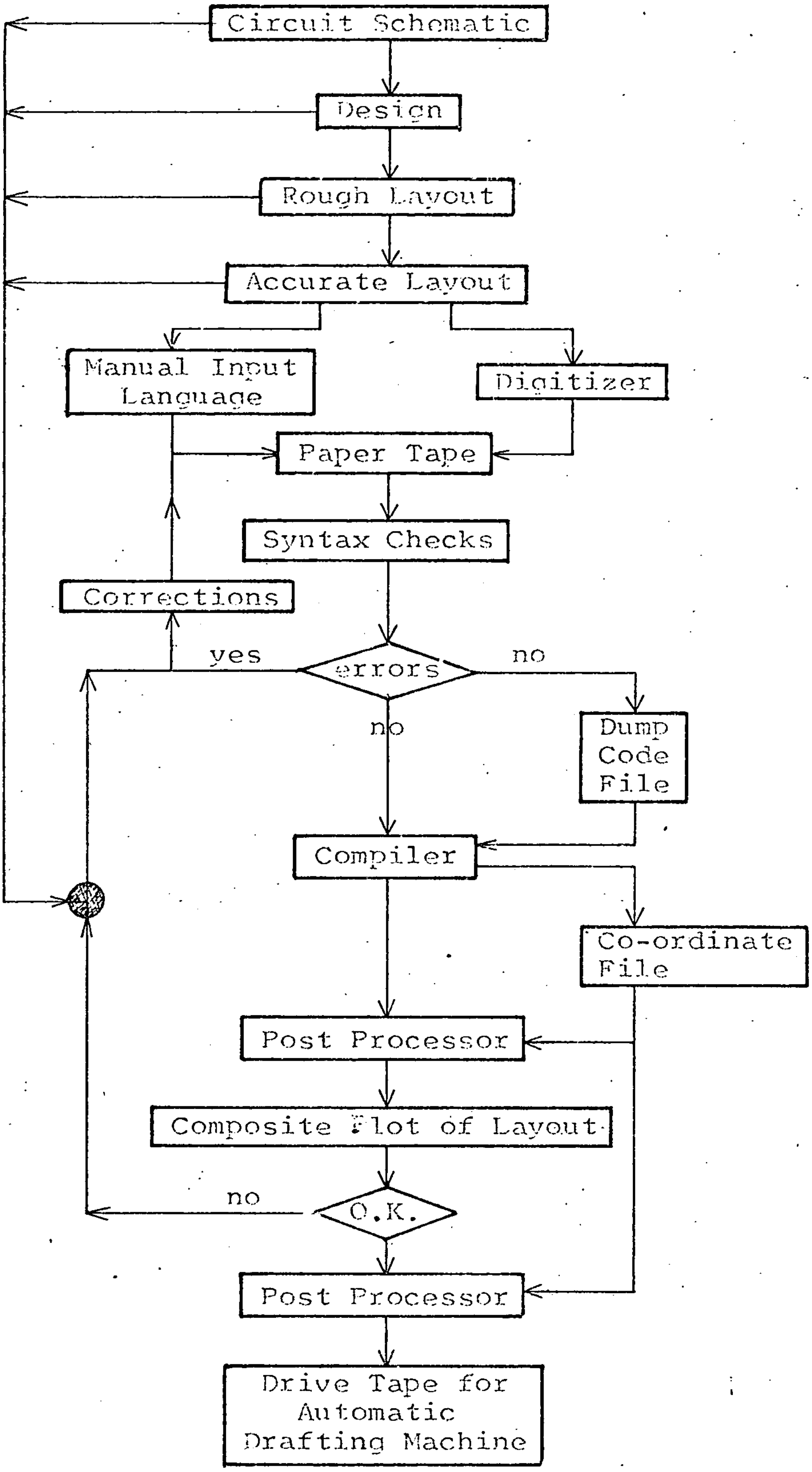


Figure. 1.1 Flowchart of GAELIC system for Integrated Circuit Artwork Preparation.

modifying the layout quickly in a permanent or temporary fashion, together with a fast displayed feedback on the effects of any such modifications. These specifications are met only by an interactive graphics screen with a light pen or equivalent facility. It is hoped that GAELIC will be expanded to make use of such a display for error-control purposes and,ultimately, dynamic design; the data handling problems associated with such a modification are discussed below.

1.2.0 INTERACTION AND THE GRAPHICS SCREEN

An interactive system must be fast; if the machine takes too long to access, modify and display data then the design process loses continuity as the operator is forced to wait while the display 'catches up'. Such delays destroy concentration and consequently decrease the human's, and hence the system's, effectiveness. To avoid this it is necessary to store the layout data in a form which is wholly and readily accessible to the machine, is easily processed and does not require long, time-consuming searches to locate a specific datum. The sequential files used by GAELIC do not satisfy these criteria, instead it is necessary to use a data structure to provide speed of processing, compactness and simplicity. It is convenient here to present a brief discussion of data structures since they have become a fundamental aspect of computer graphics.

1.2.1 DATA STRUCTURES

In recent years the data input to computers has considerably increased in size and complexity and this has lead to an emphasis on efficient data storage as well as data processing. Initially most on-core data were stored in sequential arrays reserved exclusively for that purpose. This became inefficient since each array used must have sufficient space reserved for the maximum requirement although, in many cases, this would rarely be required. For example, a circuit analysis program might be designed to handle circuits of up to 60 resistors, 30 transistors, 50 capacitors, 40 diodes, etc., owing to the exponential increase in execution time with the number of circuit components, it might never be applied to circuits of over 90 components. This means that over half the reserved space would never be used in a single application, therefore pre-defined arrays are clearly an inefficient data storage technique for variable data applications.

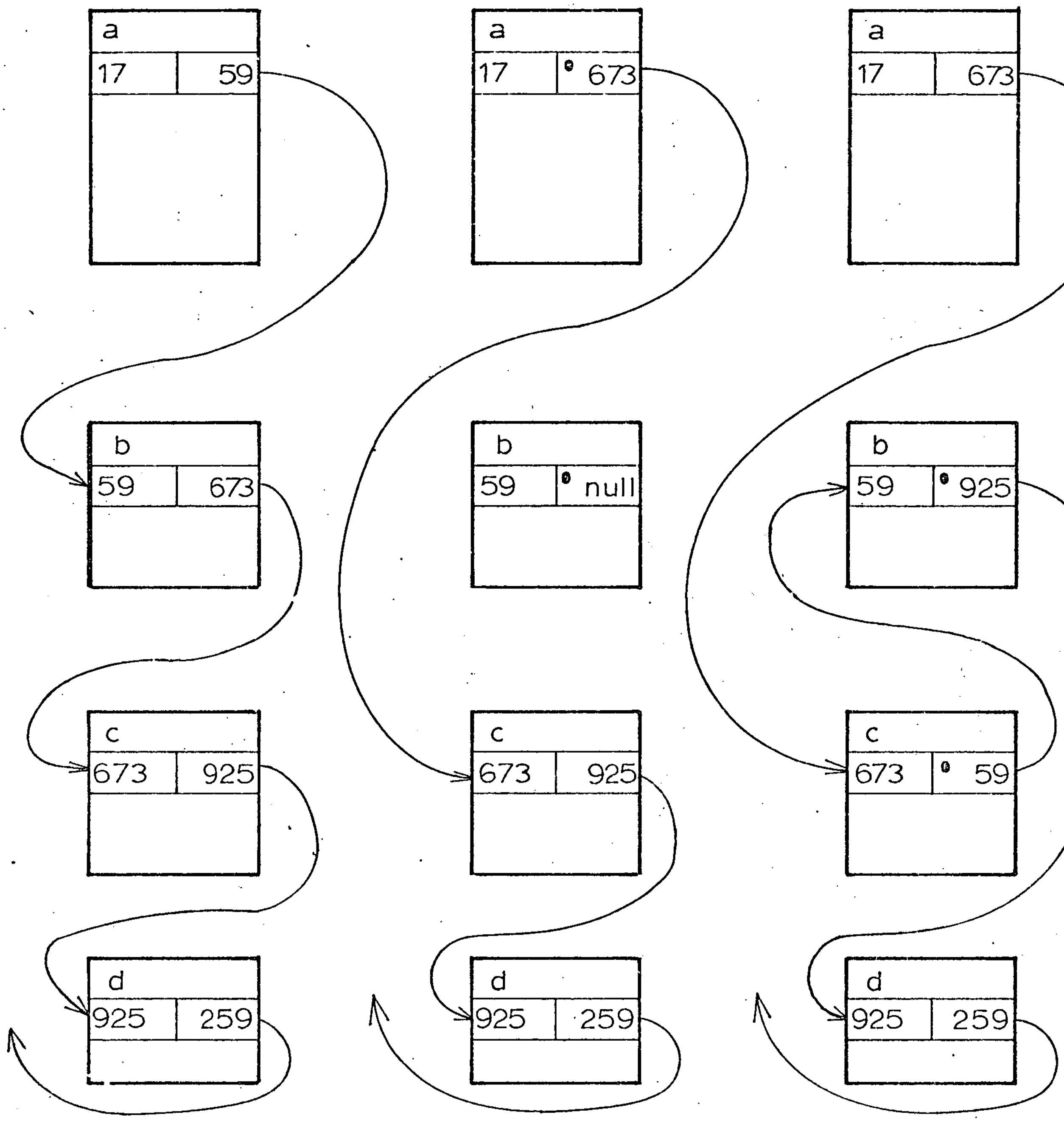
Sequential arrays are extremely inefficient in dynamic applications because simple operations, such as additions and deletions, require massive amounts of data manipulation to create and absorb vacant spaces. These operations are a fundamental part of interactive systems as the operator must be allowed to dynamically modify the data in as unconstrained a fashion as possible. List structures provide an answer to these problems by avoiding multiple array type storage and simplifying data modification operations. The data is segmented into small lots of associated information which are stored in consecutive array locations. An additional word, called a pointer, is placed at the beginning and this is used to indicate the location of the subsequent lot of information. These information units are called beads and are 'strung' on the list by the pointers at their head.

The pointers serve two purposes which overcome the basic disadvantages of sequential storage mentioned above. First, because list members are only accessible through the pointer sequence which is independant of physical storage locations, several different lists may be stored in a single array, with space reserved for the overall maximum rather than individual maxima requirements. Secondly, data modification involves a minimum of processing since only the pointer values need be manipulated instead of the data itself. Figure 1.2 shows the sequence of operations for altering a list sequence without physically moving any data.

The list structure has a major disadvantage in that it is necessary to begin all searches at the list head as any bead can only access subsequent beads of the same list. Therefore, if data has to be accessed in a variety of ways, a variety of lists are required which implies a duplication of data which is inefficient. This problem has been surmounted by placing several pointers within a bead so that the data appears once but can be located on any of the lists on which it is strung. This technique has evolved into the modern data structure described below.

1.2.2 RING DATA STRUCTURES

Ring data structures differ from simple list structures in two ways: first, they allow a bead to appear on several lists, or rings, and secondly they eliminate the necessity of starting all searches at the beginning of a list by replacing the final null pointer with one that points to the start or head of the ring. This means that a bead can be



(a) abcd
original sequence

(b) acd
bead b deleted

(c) acbd
b added after bead c

● ⇒ pointer alteration

Figure 1.2 Modification of List Sequence by Pointer Manipulation.

accessed from any other bead which is a member of an incident ring. Thus fast, complex structures incorporating levels of nested rings can be constructed and, in applications involving large amounts of data, the program speed is often a function of the efficiency of the data structure.

There is one disadvantage with ring data structures in that search procedures can become involved if there are several pointers per bead. Beads are usually structured as shown in figure 1.3. It is therefore necessary, during a search, to unpack the pointer, locate the head word, unpack it and thus identify the bead before being able to decide whether it is of current interest. A typical ring search procedure is flowcharted in figure 1.4;despite the apparent length and complexity of such procedures they do not constitute a significant handicap relative to the ease of processing gained through the use of a ring data structure. The proposed GAELIC data structure provides a good example of how a suitable structure is evolved from a consideration of the program's requirements.

1.2.3 THE GAELIC DATA STRUCTURE

The first specification for a Graphics system is usually discontinuity; to allow the man to work at peak efficiency it is necessary to provide a facility for transferring the data structure on to backing store so that work at the screen may be suspended when necessary and resumed when convenient. This implies that the entire structure must be uniquely accessible from one point only

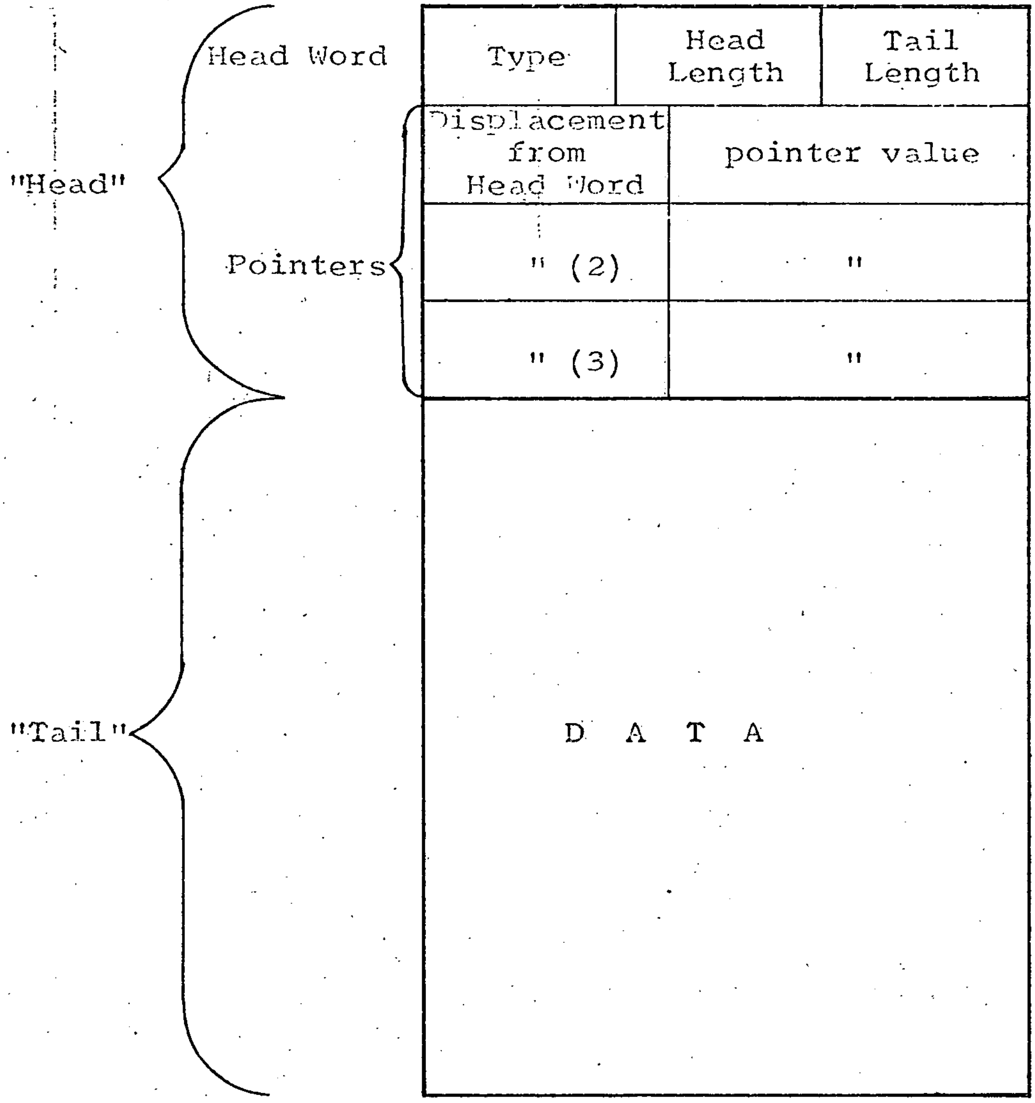


Figure I.3 Typical Bead Formation used in Ring Data Structures.

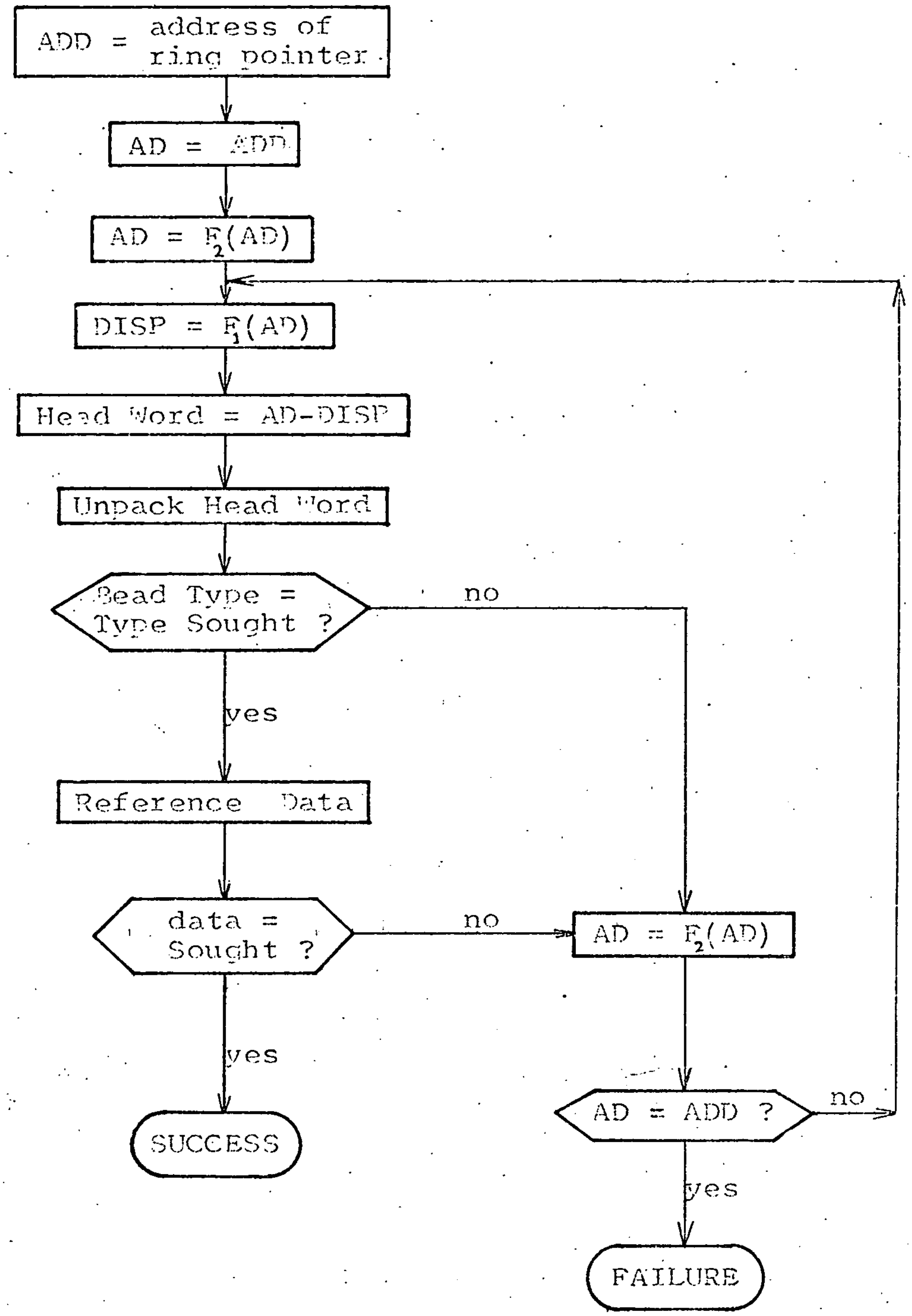


Figure 1.4 Flowchart of Ring Search Procedure.

ADD = location of search start

AD = current value of ring pointer

DISP = pointer displacement (from Head Word)

F_1 = pointer displacement calcualation

F_2 = pointer value calcualation

to allow simplification of the structure storing and retrieval processes as well as a basic transportability for the data structure itself. For this purpose, the first 6 words of the array are reserved for specific purposes dealing with garbage collection, head bead identification, two ring head pointers and a word used for layout identification.

Shape data for GAELIC is classified by mask and so each mask has a ring along which are strung beads which describe shapes appearing on that mask. In addition, the mask ring head beads are strung on a ring which commences at the head bead. Each group definition is a sub-structure identical to the main structure in format. The two structures are related in two ways; first a group definition ring starts at the layout head bead and connects all group definition head beads. Secondly, each occurrence of a specific group is included by a group call bead occurring on the appropriate main mask rings and all group call beads are strung on a group incidence ring with its head in the appropriate group definition head bead. The structure is shown in figure 1.5. To avoid the problem of packing and unpacking pointer displacements and values a scheme was developed where each pointer of each specific ring had a fixed displacement. Another reason for avoiding a displacement index is that the structure can be used on machines of small word length. This data structure was programmed on the ERCC IBM 360 and tests were run which indicated that it was suitable for use in an interactive environment being compact, fast and transportable.

I wish to point out that the proposed GAELIC Data Structure was designed in conjunction with John Eades.

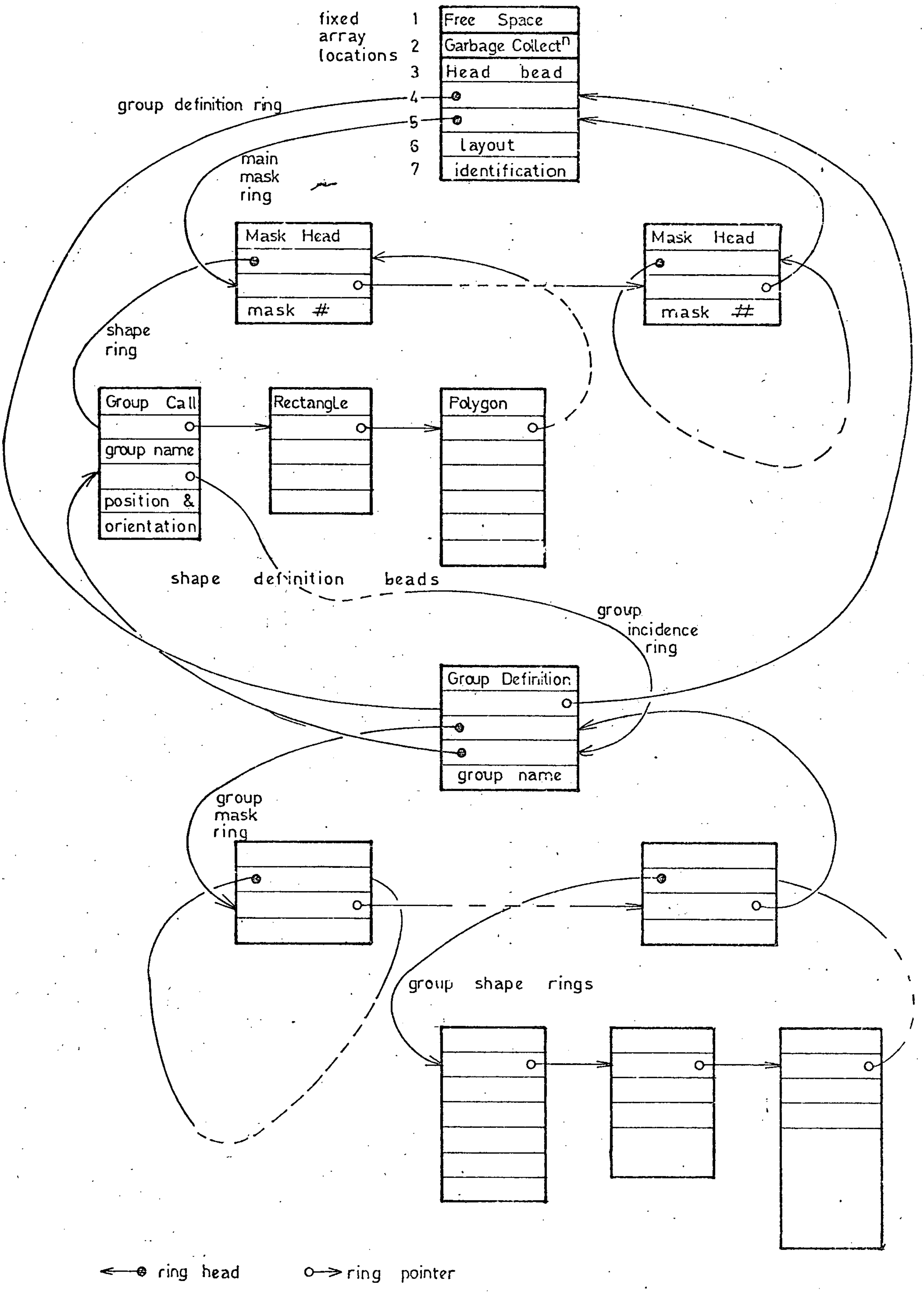


Figure 1.5 Proposed Data Structure for GAELIC.

1.3 INTERACTIVE LAYOUT

Microcircuit layout is a process of manipulating shapes to satisfy a set of design rules and optimize certain features which affect the quality of the resulting circuit. The optimizing procedure is guided by the criteria of optimality which are usually non-specific and are incorporated in manual layout at an intuitive level. These criteria are therefore very difficult to interpret to a machine and so interaction has become the usual CAD tool for circuit layout because the man can control the layout optimization without requiring the computer to become involved with the complex reasoning. Each can perform the task most suited to his or its capabilities, that is,the man makes the decisions and the machine performs the laborious calculations and accurate drawing, thus providing an efficient, balanced combination of the two providing the system is designed intelligently. Such a system requires a computer, suitable software and hardware for displaying the layout, and an experienced operator so that the system may be used efficiently.

There are severe disadvantages associated with interactive design however; the first concerns operator-training and is discussed in more detail in appendix I. The main drawback, however, is the cost of acquiring a suitable system. There are two factors which affect the system's price, hardware and software. Hardware costs are considerable since interactive graphic display equipment is expensive. Software can be either purchased or developed in-house, however this latter approach is expensive in terms of time and little return can be expected until the system is completed. Many small firms cannot consider interaction purely on financial grounds and although in recent years complete systems, such as Applicon's Design Assistant, have become commercially available, these too are out of the range of many firms' CAD budgets.

1.4 AUTOMATIC LAYOUT

The alternative to interaction is automation, however a completely automatic design system is clearly unfeasible with today's facilities as the machines available are not sufficiently powerful to accommodate the decision-making processes of layout. However, automatic layout aids are a potential source of easing the burden on the designer and producing circuits more effectively. The main features which differentiate automatic aids from interactive ones are that only access to a machine is required, the input and output can be arranged in a variety of ways so that time-sharing bureaux may be used and so eliminating the high cost of interactive hardware. No human, on-line assistance is necessary thus providing a more comfortable situation for the designer who does not have to face the machine directly.

The balance between hardware and software costs reflects the trade-off between interaction and automation. It is the purpose of this project to examine the effectiveness and usefulness of non-interactive computer aids to layout in terms of a limited CAD budget, that is, does effective computer aided layout necessarily involve high costs? The next chapter shall discuss some of the techniques and algorithms which have been applied to circuit layout as a basis for this work.

CHAPTER II

REVIEW OF EXISTING AUTOMATIC LAYOUT METHODS

2.0.0 INTRODUCTION

The layout phase of circuit design converts a logic diagram, or circuit schematic, into the information required to fabricate the circuit, be it a computer-produced drawing, a set of photographic masks or a drive tape for an automatic milling machine. The core of the layout procedure is the detailed positioning of the components and their interconnections, termed placement and routing, which are common to all microcircuit and printed circuit layouts.

Circuits which are too large to fit on one board, or one substrate, necessitate the allocation of their components to individual boards, or substrates, each of which becomes a separate layout problem. The placement and routing of these modules on mother boards is also a layout problem which may, or may not, be related to the process of laying out a daughter board. Logic circuits which are constructed from integrated circuit packages require selection of logic modules from a library and allocation of the circuit gates to the individual modules. Although much effort has been devoted to producing automatic and semi-automatic programmed aids for placement and routing, few have achieved wide application apart from automatic 'drawing board' programs which are dependant on the presence of a circuit designer at a graphics terminal; these programs merely automate the processes which would otherwise be done on a drawing board, using pencil and paper instead of a graphics screen. This apparent deficiency is a direct result of the number of parameters affecting circuit production and the lack of uniformity within the industrial production of circuits.

2.0.1 PROGRAM SPECIFICATIONS

Each manufacturing organisation produces a set of design rules on layout, determined by that firm's consideration of the most important features of a circuit and the tolerances of its production process. Standards will differ between firms, either in the emphasis placed on certain attributes of the final design or in the tolerances on physical dimensions. As an illustration of the difficulty of producing an automatic layout program for wide use, consider the decisions which must be made for a program to layout printed circuits: before any programming can be done, it is necessary to specify which printed circuit boards the program should be capable of handling, how many layers of signal wiring, will power and ground planes be present, what types of components are to be mounted on them and will they appear in constrained positions or otherwise? Each of the alternatives to the above questions will require a different technique to make full use of the features associated with it. This will generally result in a different program as a single program embodying the necessary different techniques would be large, slow and, consequently, quite inefficient. Furthermore, within a single program additional specifications must also be made: are different track widths to be allowed and, if so, how may they be specified and by whom, how many tracks to be allowed between component pins, what separations are necessary, whether there should be a range, what types of edge connector will be allowed, who can specify their whereabouts on the board and are the boards to be of a standard or variable size? These decisions will, in some cases, require a large amount of extra cost and effort in coding and necessitate, in others, entirely differing approaches to the layout problem if it is to be a generally applicable program.

Finally, the criteria of optimality must be selected and sequenced: are jumper wires to be allowed, if so where and when and how, are track lengths to be limited, if so in which cases, how many plated through holes are to be allowed and how important are size and shape of the board? These decisions will affect the performance and quality of the layouts produced and they are rarely available in a form which is readily adaptable to computer usage.

All the above fundamental decisions demonstrate the unlikelihood that a program to automatically layout printed circuit boards created by one firm could be effectively used by another.

2.0.2 INTERACTION

The advent of interactive graphics has permitted an alternative, more general approach to the layout problem. The machine can present on a video display unit (VDU), within seconds, a two-dimensional picture to a man at the terminal, who in turn can use it to alter the layout by means of a light pen and receive instant feedback on the results of his modifications. This has two major advantages. First, the design steps can be allocated to the man and the machine so that the best features of each may be combined: the designer can incorporate his experience, intuition and pattern recognition abilities, while the laborious calculations and monotonous redrawing can be left to the computer. Secondly, many of the above specifications can be left to the discretion of the operator at the console and need not be incorporated in the program, thus replacing the more complicated sections of programming. The resulting programs, therefore, often contain very few automatic design features, but are composed of routines which allow the screen to imitate a drawing board with as little inconvenience to the operator as possible and, furthermore, have a wide range of application; appendix I contains a brief description of such a system and comments on its performance from a user's point of view.

There is a severe disadvantage associated with interactive graphics in that the hardware is very expensive and few small firms can afford the initial outlay. This is compounded by the fact that it takes a considerable time for a designer to gain the expertise required to make full, efficient use of the system. Interactive programs require many safeguards against the possibility of operator mistakes since every conceivable action, valid or otherwise, must be allowed for if the program is to be reliable and, consequently, the programming takes longer and requires extensive testing before it is ready for use. These factors all cause increased expense and it has yet to be determined whether the interactive or the completely automatic design mode is the more cost-effective, computer-aided design method.

2.1 CRITERIA OF OPTIMALITY

Automatic layout aids, as has been mentioned, all require some criteria of optimality; they are necessary as a guiding strategy for the decision making processes and enable the machine to decide which of two possible alternative layouts is the better. The most widely used criteria are functions of size and shape and wiring patterns. The computer requires a very simple and unambiguous statement of these criteria, which are usually stated in a form similar to the following:

(i) minimal rectangular area

(ii) minimal total wiring length

(iii) no crossovers

Although these criteria are inapplicable in some cases, they embody the general aims of layout programs. Layout methods assign different priorities to these items and, in so doing, solve one part of the layout problem while ignoring the other. For example, attaching high priority to minimisation of dimension while ignoring crossovers results in an approach which is suitable for component placement, but does little to solve the routing problem. The consideration of one aspect at the expense of another has resulted in techniques falling into one of three categories: placement algorithms, routing algorithms and topographical methods. Placement algorithms attempt to minimise circuit area and wiring lengths; routing algorithms minimise track lengths and other wiring penalties such as the numbers of plated-through holes, crossovers or wiring bends; topographical methods attempt to relate the problems of placement and routing through a graph representation of the circuit generally with the initial aim of minimising crossovers.

The remainder of this chapter is devoted to a discussion of layout techniques which contain automatic features, that is,some proportion of the layout decisions are made by the algorithms or programs, within the three categories described above.

2.2.0 PLACEMENT TECHNIQUES

Placement is that part of the layout phase which positions the circuit components. Two cases require separate treatment. The first is the unconstrained case where the components are free to appear anywhere on the board, provided that no two components occupy the same space and that they all appear completely within the boundary of the layout. The second, or constrained case, requires that the components appear only at specified points, usually a rectangular array of positions, and have a uniform orientation; components for this type of board generally have the same dimensions. An example of the former case is the discrete component printed circuit board, whereas a circuit composed of integrated circuit packages of a uniform configuration usually belongs to the latter case.

2.2.1 UNCONSTRAINED PLACEMENT

Current unconstrained placement techniques are all similar and are called force placement, rubber-band placement or centre-of-gravity collapse; despite the variety in nomenclature the basic concepts are nearly identical. Capocaccia and Frisiani (1970) have developed a force placement technique for the placement of large scale integrated circuit components, that is based upon two characteristics of LSI technology. The first is that there will be at least two signal wiring layers, the second recognises that the proportion of the chip's area occupied by conductor tracks rapidly increases with circuit complexity, which will affect size and, consequently, yield. The procedure was thus designed to ignore the crossover problem while minimising a weighted function of the interconnection length.

Elements are treated as circles with equivalent radii dependant upon their dimensions and number of connections to them. Initially elements are placed arbitrarily; they are then subjected to an iterative shifting procedure which results in incremental moves governed by the resultant force vector, which is obtained from summation of the forces acting on the element. The movement of elements is divided into three phases. First, the field size is defined larger than the desired final dimensions and the process proceeds until the resultant vectors become insignificant; concurrently the field boundary is decreased to the limit required. Finally, the elements are moved within this limit until equilibrium is achieved as the incremental moves fall beneath a significance limit.

The forces are derived in a straightforward manner through a consideration of the placement objectives. Terminal pads are either fixed, in which case they cannot be moved, or free to move, when they will be considered as additional elements placed along the boundary of the chip. All connected elements are considered to exert a mutual attractive force proportional to their separation and a weighing factor. To avoid superimposition at the field's centre each element is considered to exert a repulsive force on every other element in inverse proportion to their separation and this force is increased during the compression stage of the algorithm so as to balance the increase in attraction resulting from the smaller boundary.

In addition the boundaries exert a repulsive force on all free elements to prevent their escaping, while exerting attractive forces on those elements connected to the outside world, that is, the terminal pads, so that they move towards the boundaries.

The main disadvantage of this procedure is its inability to cope with irregularly-shaped components as they are represented by equivalent circles. Any departures from a roughly square outline will produce problems when the circular areas are translated back to their real outlines. Other applications of force placement techniques employ different component representations, such as rectangles, to overcome this drawback. Another disadvantage is that the spatial requirements of the tracks are not considered. When the components are placed there is no guarantee that the areas available for interconnections will be sufficient. This problem is not so severe in some technologies, such as thin film, which allow under-component wiring because the tracks are not confined to the areas between components.

Atiyah and Wall (1970) attempted to improve placements obtained from a force technique by examining interchanges of elements in order to decrease the length of the longest connections, but they do not explain how elements of different sizes or shapes may be exchanged without a major reshuffling of elements. Fisk et alia (1967) have improved the force placements by introducing a moment of rotation so that elements are oriented in such a manner as to further reduce interconnection length.

2.2.2 CONSTRAINED PLACEMENT

Force placement is unsuitable for constrained layouts because incremental moves are not permitted. Elements must appear either in one position or another and nowhere in between, which invalidates the slow settling down process of the force method.

Pomentale (1965) has produced an algorithm for placing N modules represented by (E_1, ... E_N) in a regular array of a x b positions where $a.b \geqslant N$. Null or dummy elements are added to occupy the vacant positions in the matrix and the set of elements thus becomes (E_1, .. E_N, E_{N+1} .. E_{ab}). A function F is defined, which calculates the advantage of one placement over another. Pomentale uses the number of angles in the manhattan connection of the elements which corresponds to the number of plated through holes required for a two-sided PC board. The value of F is calculated from the position matrix P_{ij} ($i = 1, a$: $j = 1,b$) and the connection matrix C. The algorithm examines interchanges of elements by assessing the alteration in F, $\triangle F_{is}$, produced by exchanging elements E_i and E_s. The most advantageous exchange, that which results in the greatest decrease in the number of corners, is carried out by altering the appropriate elements of P; the exchange of null elements is prohibited as it would obviously produce no change in F. The algorithm proceeds until no further reduction in F is possible through positional exchanges, that is, a local minimum has been reached. Although Pomentale presents results for three different schemes of selecting the interchanges, he does not give any indication of time requirements for the program.

Pomentale's method has been adapted by James (1968) to layout dual-in-line integrated circuits (DILIC) on a two-sided PC board using two placement algorithms. The first employs the number of angles in component to component connections for its backboard function and it is applied until no reduction is possible. The second algorithm uses a function which gives the total inter-connection length of the nets of the circuits, calculated on a minimum tree basis using Abruca's algorithm (Abruca, 1964).

Steinberg (1961) has developed an algorithm for the placement of elements in a specified set of positions which minimises a metric function, derived from the component placement, by examining alterations in the positions of unconnected sets of elements. This metric function gives the total length of connections based on the metric matrix D where d_{ab} is the distance between position A and B. After an initial, arbitrary placement an unconnected set of elements is constructed as follows: an element is chosen and placed in the set and then an element which has no connections to any member of the set is added to it on an iterative basis until no more can be added, that is, the set is maximal. This set of elements is then removed from the board creating additional vacant positions and, since they are unconnected, their relative positions do not affect the wire length. Each element in the set is tried in each of the vacant positions and the best replacement of the set is calculated using any one of the available algorithms for the optimal assignment of n objects to n locations, for example Kuhn (1955), Munkres (1957) or Ford and Fulkerson (1956). Then a different unconnected set is removed and examined in the same way. The algorithm terminates when no further decrease in the wiring length is possible.

Rutman (1964) has improved this algorithm by treating nets and point-to-point connections individually. He uses a calculation of the wire length based on Manhattan distances and divides the algorithm into two phases; phase I attempts to reduce the lengths of the longest wires only, which tends to produce a layout requiring some fine adjustment for optimality, and this is supplied by phase II which minimises the total wire length of the layout. All unconnected sets of elements are calculated at the beginning of the program and used cyclicly in successive iterations. He made further improvements by adding an examination of interchanges between connected elements to avoid inefficient sublayouts of tightly-connected sets of elements, such as that shown in figure 2.1 where elements A and B would not be interchanged by the algorithm because they are connected and, therefore, one of the pair will always be present on the board. This examination of possible exchanges is interspersed with the replacement algorithm at a fixed interval (for example, after every 10 iterations).

Two boards were placed using the program on an IBM 7090; the first, 70 elements in 77 locations, was completed in 4 minutes with a 19.4% improvement over the initial manual placement by a skilled designer and the second board consisting of 516 elements in 540 locations, was

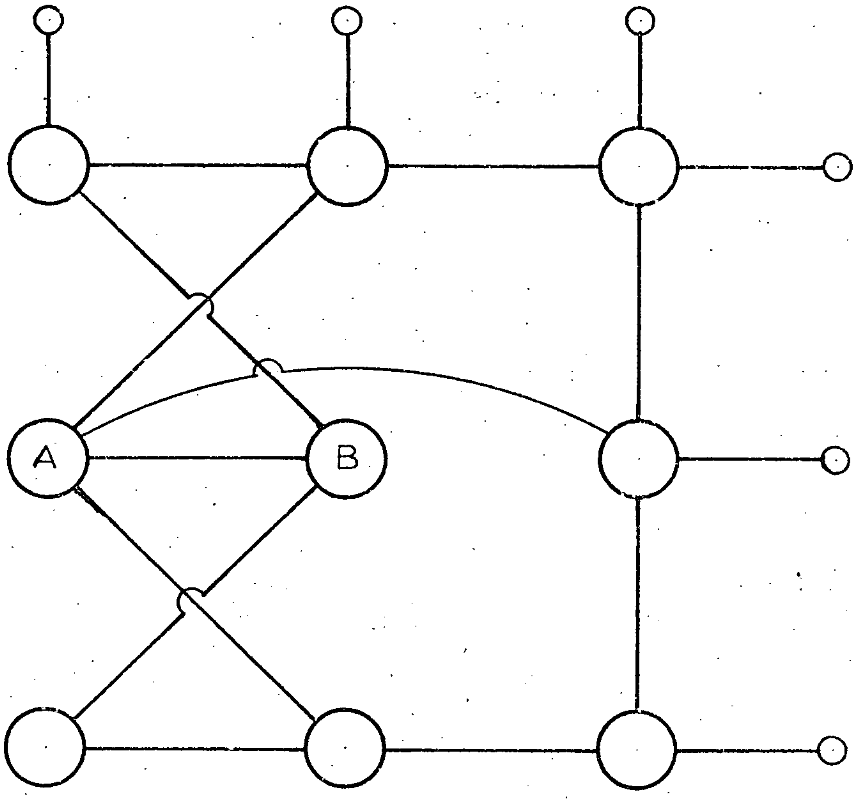


Figure 2.1 An Instance where Steinberg's Algorithm would be inneffective.

terminated after 4 hours, 20 minutes,having made a 39% improvement over the manual placement. It would appear, therefore, that Steinberg's algorithm,as modified by Rutman,is feasible for use with circuits of around 200 elements.

Garside and Nicholson (1968) have developed a permutation procedure for minimising the wiring length of an array of elements. The positions of the elements within the array may be described by a permutation P of the integers 1 to N (where N is the number of positions available), using dummy elements to occupy surplus positions. An initial permutation is created by placing the element with most connections centrally on the board; thereafter the next element placed is that with the most connections to those already placed and its position is selected to minimise the increase in total wire length. If a second element already occupies this position, it is shifted to a nearby vacant location and the increase in wire length recorded. This is repeated until all elements have been placed; alterations to the permutation which decrease the wire length are sought by selecting the element with the highest total of wires connected to it for tentative exchange with all others. The exchange producing the largest non-zero reduction is made; if there is none, the permutation is not altered and the element is added to the list of elements selected since the last interchange. The procedure is stopped when the list contains all the non-dummy elements, which indicates that a local minimum has been reached. A procedure for exchanging three elements in cyclical fashion is also proposed as a further improvement, however the results given demonstrate that it rarely makes a significant improvement on the pair exchange procedure. Furthermore, the computing times given imply that even the binary interchange procedure is too slow for realistic use.

Crockford, Maller & Carnell (1967) have produced a system for the layout of naked ICs mounted on a thin-film. The elements are placed in a regular array of M x N cells, where M x N is the number of elements to be placed. Dummy elements are included for the 2(M+N) pin positions around the periphery. At first the elements are placed arbritrarily and nets are described by strings of electrically common points. Measures of central location (the cell nearest the centroid of the net members) and distribution (the sum of the distances of members from the central location) are compiled for each string. Elements are then moved to adjacent cells in the direction of the centroid by neighbouring exchanges and the effect on all other string distributions is calculated; if the overall distribution is not increased the exchange is made permanent. Repeated scannings of the input string list and consequent exchanges continue until a predetermined number of consecutive scansions fails to produce any improvement, when the process is stopped. This procedure rapidly achieves a placement which is a good approximation of minimum wire length and this is considered to be sufficient as subsequent routing difficulties may require positional alterations for their solution.

A system for placing computer modules, that is, small cards 'plugged' into a regular array of sockets on a mother board, has been developed by Houghton (1969). The program operates in three phases and allows modules to be preplaced and to occupy more than one board position. The first phase consists of clustering, which groups together highly-connected sets of boards so as to minimise the sum of the squares of the point-to-point distances; the use of the square of the distance means that long connections will contribute a higher proportion to the distance function. The process is partitioned into X and Y directions; first, only horizontal distances are used and board movement is confined to the rows and, secondly, vertical distances and columnar movements are used; these two subprocesses are iterated until movements become insignificant. The second and third phases of the algorithm are transpositional, both use the same mechanics, but different measures of optimality to examine the gains from interchanging each element with its 8 neighbours; any exchange which yields a reduction in the wiring length is fixed. Phase two uses the sum of squares of point-to-point distances, (whilst phase three calculates the string lengths of nets using a procedure which takes time proportional to the cube of the number of pins in the net) as its criterion of optimality; in addition, the number of twisted links is significant but Houghton does not clarify this criterion. One example quoted reduced an initial layout with a total (calculated) wiring length of 1788 feet with 913 twisted links, to 1224 feet and 178 twisted links. The three phases took 1, 4 and 3$\frac{1}{2}$ hours respectively on an ICL 1903A, which is very expensive in terms of time compared with the manual layout of 974 feet and 188 twisted links.

Mamelak (1966) has developed another procedure for the placement of elements in a fixed-array on a two-sided board, with horizontal tracks on one side, vertical on the other; it considers the presence of edge connectors and the requirements of minimising pulse delay-times along certain connections. The program has three objectives; first, the minimisation of intersections between conductors, secondly, the satisfaction of the proximity conditions (for minimum delay times) and thirdly, the minimisation of the total wire length. The connection matrix is examined, and a list of 'chains' compiled; a 'chain' is a set of elements at least two of which are connected to the remaining elements of the set, ignoring power and ground plane connections (see figure 2.2). For a circuit of n logic elements Mamelak found that there are about $n^2/3$ 'chains' and each element belong to about n/3 chains. The 'chains' are allocated to rows of board positions and permuted so as to minimise the interchain connection distances; pairs of 'chains' are folded into rows to form composite 'chains' in order to compact the layout. The permutation of chains followed by folding is then repeated until no more folding is possible; this process determines the abscissae. The allocation procedure then considers the placement as viewed from the underside of the board and uses 'chains' whose elements are farthest apart in the y direction, thus the ordinates of each element are determined. Results given, using a 20K RCA 301, indicate that the quality of layout achieved is comparable to that produced by a skilled designer. The time requirements for circuits of 50 and 100 elements were respectively 20 and 60 minutes.

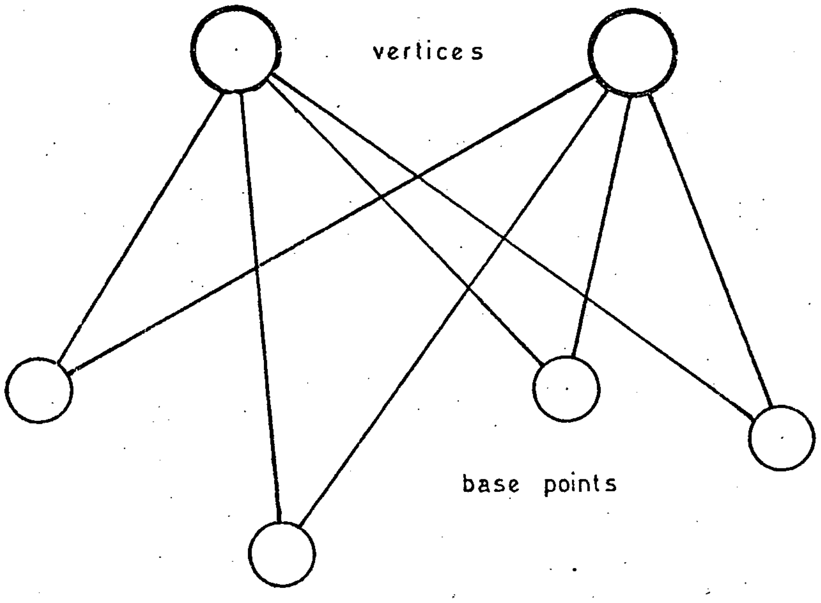


Figure 2.2 A "Chain" of Order 2.

The order of a "chain" = number of vertices.

2.3.0 ROUTING TECHNIQUES

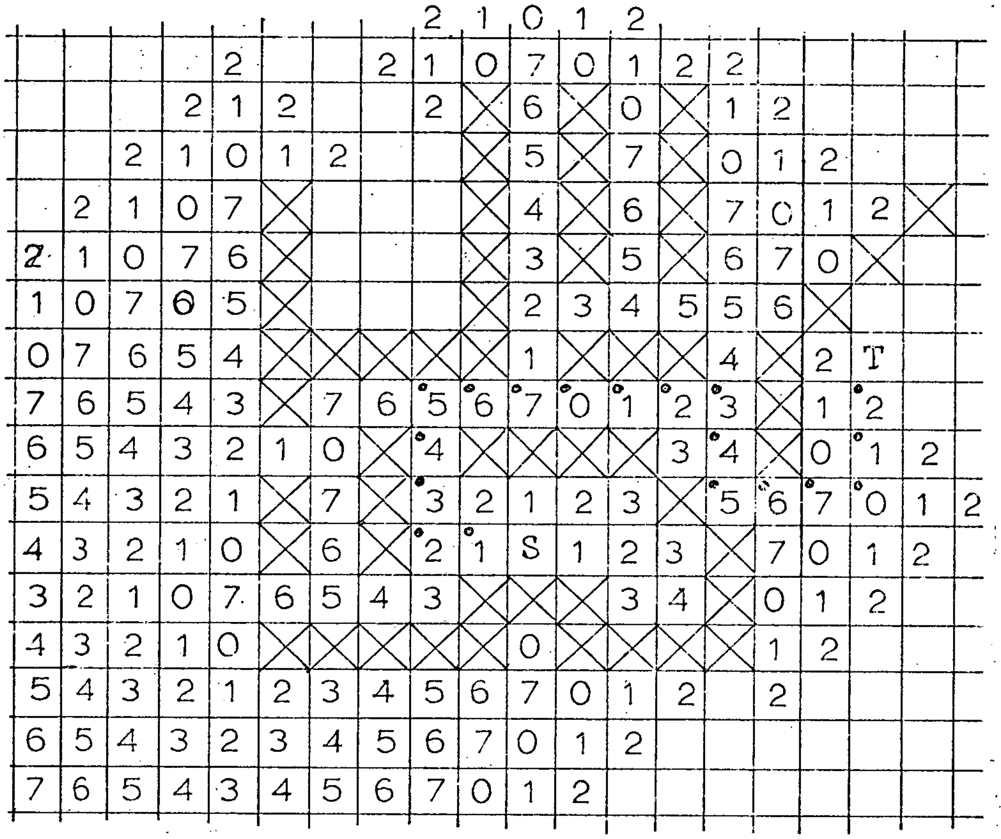
The routing of a layout consists of fixing the positions of the interconnections on the board or substrate. The conductors are bound by two constraints; first, they must not crossover each other on the same wiring layer, unless electrically equivalent, otherwise a short circuit would result and, secondly, they must maintain a minimum separation between insulated conductors to avoid leakage. Other effects, such as stray capacitance, resistances in long integrated circuit connections, propagation delays and parasitic devices, also place constraints upon the routing; however, these are generally acknowledged to be too subtle for an automatic program to take fully into consideration.

2.3.1 MAZE RUNNING ALGORITHMS

The principle routing algorithm is that of Lee (1961) which is a maze-solving algorithm that will find the shortest path inside a specified area between any two points. The area is divided into a regular, usually rectangular, array of cells and the target cells, of which there may be any number, are given unique flags and any cells through which a route may not pass are given an appropriate flag. The starting cell is assigned the value O and the iterative search procedure commences. In each iteration unmarked cells, that is, those without an assigned flag or value, are scanned and their adjacent cells examined; if any of a cell's neighbours contain a value then that cell is assigned a value of one plus the value of the neighbour. However, this incrementing is usually done on a modulo basis to avoid very large values. The iterations cease when a target cell is reached, or when no new values are assigned during an iteration, which implies that no route exists; a route can be traced back to the start through adjacent cells of descending value. All cells which receive a value during a single iteration are equidistant from the search origin. This means that the first route found is also the shortest. An example of the search process using a modulus of 8 is shown in figure 2.3, demonstrating that its behaviour may be considered analagous to a wave spreading from the origin. This is a very powerful algorithm which guarantees to find the shortest path, if one exists, between any pair of cells, but it requires a cellular description of the board which is very costly in storage, as 57K arrays are required for the resolution of some boards. Lee does not give information on time requirements.

Ginsberg, Maurer & Whitley (1969) have developed a technique based on Lee's algorithm for routing multilayer boards. The nets of the circuit are sequenced into an order of increasing length and are assigned length constraints prior to routing, and then they are classified according to type, size, number of connections and slope class. The slope class is derived from the boundary rectangles' dimensions which results in a horizontal, vertical or diagonal classification. When all the information is compiled the nets are assigned to wiring layers for the routing algorithm.

The routing is done in the following manner: the list of points or pins which are to be connected by a net is supplied, the first is designated the starting point and



								2	1	0	1	2							
				2			2	1	0	7	0	1	2	2					
			2	1	2			2	☒	6	☒	0	☒	1	2				
		2	1	0	1	2			☒	5	☒	7	☒	0	1	2			
	2	1	0	7	☒				☒	4	☒	6	☒	7	0	1	2	☒	
2	1	0	7	6	☒				☒	3	☒	5	☒	6	7	0	☒		
1	0	7	6	5	☒				☒	2	3	4	5	5	6	☒			
0	7	6	5	4	☒	☒	☒	☒	☒	1	☒	☒	☒	4	☒	2	T		
7	6	5	4	3	☒	7	6	•5	•6	•7	•0	•1	•2	•3	☒	1	•2		
6	5	4	3	2	1	0	☒	•4	☒	☒	☒	☒	3	•4	☒	0	•1	2	
5	4	3	2	1	☒	7	☒	•3	2	1	2	3	☒	•5	•6	•7	•0	1	2
4	3	2	1	0	☒	6	☒	•2	•1	S	1	2	3	☒	7	0	1	2	
3	2	1	0	7	6	5	4	3	☒	☒	☒	3	4	☒	0	1	2		
4	3	2	1	0	☒	☒	☒	☒	☒	0	☒	☒	☒	☒	1	2			
5	4	3	2	1	2	3	4	5	6	7	0	1	2		2				
6	5	4	3	2	3	4	5	6	7	0	1	2							
7	6	5	4	3	4	5	6	7	0	1	2								

☒ = occupied

S = start

T = target

0–7 = values assigned by algorithm, modulo 8.

Successful path, found after 18 iterations, shown by dotted cells.

Figure 2.3 Application of Lee's Algorithm.

the others target points; Lee's algorithm is then applied until a target point is reached, when the path is identified and flagged as a subnet. An unconnected home point is then designated as the new start and the search applied until another target point or subnet is reached. The former result generates another subnet while the latter extends an already existing subnet; this procedure continues until all points belong to a subnet (if there is only one subnet the net is complete). In order to connect distinct subnets the component cells of one are considered as starting points and all other subnet cells as target points and an iterative application of Lee's algorithm, similar to the above procedure for connecting pins to subnets, is used to complete the net, which is then finalised on the grid. If the length constraint is violated, the search procedure is transferred to an adjacent wiring layer with the length limit reduced by the lengths of any completed subnets.

The use of length constraints is a valuable adjunct to the algorithm as it prevents the search procedure producing inefficient connection layouts by generating long and complex paths. It is also an excellent means of conserving the time used for the search procedure as the constraints are specified in terms of cells, which is equivalent to placing a limit on the number of iterations of the algorithm.

Hightower (1969) has produced a modification of Lee's algorithm, which avoids the cellular grid board description and applies it instead to the continuous plane; the paths are restricted to Manhattan geometry which gives rise to the assumptions on which the modifications are based. The board is described in terms of line segments and it is noted that to connect any two points each must be at an end of a vertical or horizontal line segment; if a line segment is bounded by another which it cannot cross then it must be followed, in the sequence of line segments which form a path, by a segment perpendicular to it.

Hightower has produced a search procedure applicable to one and two-sided boards, which is similar to Lee's except that line segments are used instead of adjacent cells. Routes between two points are constructed by starting the segment construction at both points and allowing them to spread simultaneously until they connect, since a point target would be hard to hit if the search was carried out from one end only. The connection path is identified by tracing the line segments back to each of the points; an example of the search procedure is demonstrated in figure 2.4.

The advantages of this technique are two-fold; first, the precision of definition is not limited by the cell size, that is, a more accurate description does not cause a vast increase in storage requirements, and secondly, this search procedure is much faster than the cellular technique; moreover, 33 DILICs, with 2 edge connectors on a two-sided board were wired in 33 seconds, which is an impressive figure.

Brown and James (1972) have produced another modification of Lee's algorithm for use on two-sided boards with segregated wiring connecting a regular array of DILICs,

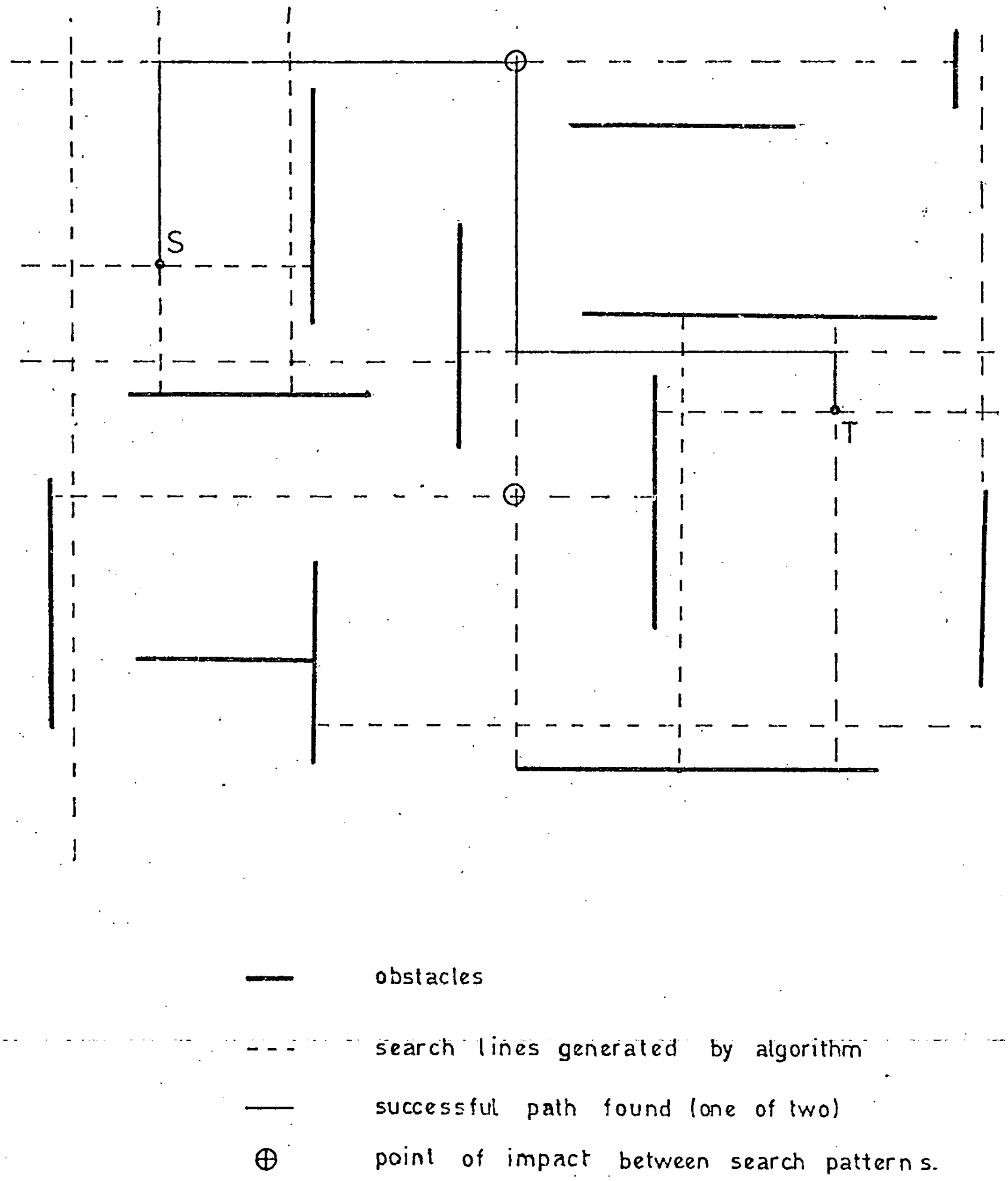

S
T
obstacles
--- search lines generated by algorithm
successful path found (one of two)
⊕ point of impact between search pattern s.

Figure 2.4 Application of Hightower's Algorithm.

which have been placed as described above. The algorithm is partitioned by restricting scanning to the directions in which the wiring runs, on the top or horizontal side, only the squares to the left and right are examined and, similarly, on the underneath, only the squares above and below are considered. The algorithm also examines the corresponding square on the other side of the board in each case to allow for the inclusion of plated through holes.

The nets are sequenced on a minimum tree basis so that routes with little choice are completed first; the nets are constructed by first forming a subnet of two pins and then connecting the remaining pins to it, at each stage extending the subnet to include another pin. Each connection between two points is limited to a maximum of 5 segments confined to the boundary rectangle of the net to avoid devious and inefficient connections. The cells comprising the route are then examined and each is given one of five symbols, signifying left, right, up, down or through, which indicates the location of the successor cell. Although no details of performance are given, the algorithm will be more efficient than the original algorithm as only a subset of cells is examined at each stage producing a faster solution, however this advantage will be offset by the storage necessary to hold the descriptions of both sides of the board simultaneously.

A programme for producing automatic trial layouts of thin film microcircuits has been developed by Cullyer et alia (1969). The programme requires a schematic of the circuit, which has crossovers only where the technology permits; this is done manually and constitutes a large portion of the layout effort. A numerical coding of the schematic is processed by the computer and the sizes of components are converted to representative rectangles, which are placed on an integer grid in such a way that the relative positioning of the schematic is preserved. After deducing the terminal pin positions, connections are routed with Lee's algorithm. This process is limited to circuits with about 20 components, requiring around 5 minutes processing time. There is no optimising procedure and the layouts are not produced within a specified boundary, and they will therefore require manual condensing to fit on the substrate area. These limitations are a severe shortcoming and would greatly limit the value of the program. In fact, the program does little more than plot a layout, as defined by the schematic, replacing symbols by rectangles.

Péze (1969) describes another example of the use of Lee's algorithm; this program is confined to semi-automatically tracing printed wiring connections for a specified component placement. The routes are completed in an order which is determined from their length and direction. The problem of a 'block' forming when a connection is rejected and the limitation of the Lee-type-routing to tracing only one path at a time have been examined and the following solution proposed: an interactive technique which allows a designer to control, affect and help the machine by intuitively steering it around troublesome points which arise from the above shortcomings. The designer is permitted to control, via a teletype, the operations of the routing in the program and to delete or re-route any paths which are creating a problem. The implementation of the program was not complete at publication and as no other reference to its use has been found, nothing can be said of the pros and cons of operator assistance for Lee's algorithm.

The problem of routing the constrained position board with any number of wiring layers has been tackled by Hitchcock (1969), who has developed a modelling technique based on board configuration. The board is considered to have a regular array of vias which are used for component pin insertion and interlayer connection. The modelling scheme partitions the overall routing problem into sub-structures, each of which is the smallest portion required for analysis but is, at the same time, large enough to contain a non-trivial sub-problem which can aid the gross solution. Each via is conceptualised as a diamond and by connecting adjacent corners of neighbouring diamonds with horizontal and vertical lines an array of octagons is produced. Any conductor route must, therefore, start in a diamond, pass through octagons only and terminate in a diamond, thus the board's description and analysis may be compartmentalised into the descriptions of the octagons. A set of capacity parameters is associated with each octagon and defines the amount of printed wiring allowed to cross each side of the octagon and the two main diagonals.

The routing is done with an adaptation of Lee's maze-running algorithm which uses the octagonal pattern for searching for the target diamond. A route passing through an octagon divides its perimeter in two, or as it appears to the routing algorithm in subsequent searches, into an accessible and inaccessible region; this means that the search algorithm starts at a diamond and spreads, crossing only those octagon boundaries which have an accessible portion available. A path is described with three entities viz., a list of the cells passed through, which sides are crossed and the relative position of the path as it crosses each edge. When the routing is finished these path descriptions can easily be translated into the actual coordinates required for board production, the capacity sets having assured this possibility in terms of permissable wiring density. This appears to be a very powerful modelling technique and, used in conjunction with Lee's algorithm, possibly would be amongst the most effective routing algorithms available; its efficiency should be high since the storage required would be vastly decreased by the versatile cellular description and the time requirements would similarly be diminished over the conventional grid representation. However, as Hitchcock gives no details of its use in a wiring program it is impossible to compare, other than speculatively, its cost-effectiveness with that of other methods.

Fletcher (1970) has described a modification of Lee's algorithm, designed to reduce the storage requirements of the cellular circuit description, for use in bipolar IC layout. The search is confined to channels, which are the areas between isolation regions. An integer grid is used and the mechanics of the algorithm are identical to Lee's with the exception that only that area of the channel in the vicinity of the wave front is transferred to the grid description, thus vastly reducing the storage requirement. Fletcher does not describe an implementation of this method and so nothing definite can be said about its time requirements; however, the usual space/time tradeoff would undoubtedly apply and it is doubtful if it would significantly increase the algorithm's overall efficiency.

Fisk, Caskey and West (1967) have incorporated Lee's algorithm in ACCEL (Automated Circuit Card Etching Layout) for laying out two-sided boards with a unique topographic twist to produce an even wiring density. The paths are routed simultaneously from all member pins (it is not clear how this is done, as no description is included), all failures are than reattempted, simultaneously. Any routes which are still un-routed are tried separately, starting from only one end. If this fails, the other end is used as the starting point (it is unclear how Lee's algorithm could fail from one end but succeed from the other). The topographic simulation is now used to take maximum advantage of the available space; terminal pin locations are assigned an altitude of 0, and the card perimeter an altitude of 1 and all neighbouring squares are assigned an altitude 1 larger, in the same way that Lee's algorithm assigns values, up to a maximum of 7. The paths are examined sequentially, starting with the shortest, in an attempt to produce a new route of lower altitude by tracing backwards and forming a new path by altering the positions of the corners in the route. The corners are shifted to permit the path to keep to the squares of lower altitude wherever possible, whilst making the same directional changes as the old. As a path is finalised, its member squares are assigned an altitude of 1 and the board's new topography is generated. This system has been used with a standard force placement to produce layouts in the form of board negatives, hole drilling and parts lists. It is one of the first working layout systems, however, as Brown and James (1972) have commented, 'it seems established that this program is inefficient and uneconomic'.

Crockford, Maller and Carnell (1967) employ a maze-running algorithm in IC layout procedure, in which the connections are made by a structure of multi-layer thin film conductors. The routing aim is to find and reserve the shortest connection path, including a corner penalty, which connects all nodes of the net. The nets are sequenced so that the paths formed start at one corner of the substrate and spread outwards, like a wave, to increase the chances of success; it is not explained why this should result. The algorithm is initially confined to the boundary rectangle of the net, which is progressively enlarged until the specified limit is reached. Nets are formed by subnet construction and extension and any failures produce a message identifying the net and these are manually inserted later; alternatively, if there are a large number of failures, elements may be replaced and the routing redone after an inspection of the layout. The layout of a 66 element substrate was completed in about 30 minutes on a medium speed processor.

An heuristic algorithm for interconnection of electronic components on a standard multilayer board with ground and power planes has been developed by Evans and Gribble (1969). The components have been placed in a regular array and the routing is accomplished by following a set of rules designed for programming ease.

The first rule defines the order of interconnection which is as follows:

(i) critical paths requiring minimum pulse delay times

(ii) largest nets

(iii) collinear pairs of pins, that is, pins in the same row or column

(iv) pairs situated at the opposite corners of large rectangles

(v) points within small squares (.5" - .5")

(vi) points within medium size rectangles (.5" - 1.5" long)

The remaining rules are concerned with the method of routing viz., the top or left hand track available must be used and tracks are confined to the boundary rectangle of the net. The order of attempted routes is one corner routes (two sides of a rectangle) followed by all possible two corner routes. If this fails to produce a path, routes composed of one and two corner routes are examined.

The algorithm connects a net by first joining the nearest pair of pins using a route which has a corner nearest an unconnected pin. The corners produced are now treated as pins and the process iterated until the net is complete. This technique was found to produce complete wiring of logic circuits containing 30 DILICs on a .9" x .6" spacing; this is made possible by using a double pair of track planes which allows packing density to be limited only by physical dimensions.

2.4.0 TOPOGRAPHICAL TECHNIQUES

Topographical techniques of circuit layout use a graph representation of the circuit to redefine the layout problem as a graph analysis or synthesis problem. Analysis refers to those methods which re-arrange a graph in order to improve the circuit layout, whilst synthesis is applied to those methods which build up a graph representation of an optimised layout. Many of the techniques are concerned with constructing or deriving a planar graph.

2.4.1 CIRCUIT REPRESENTATIONS

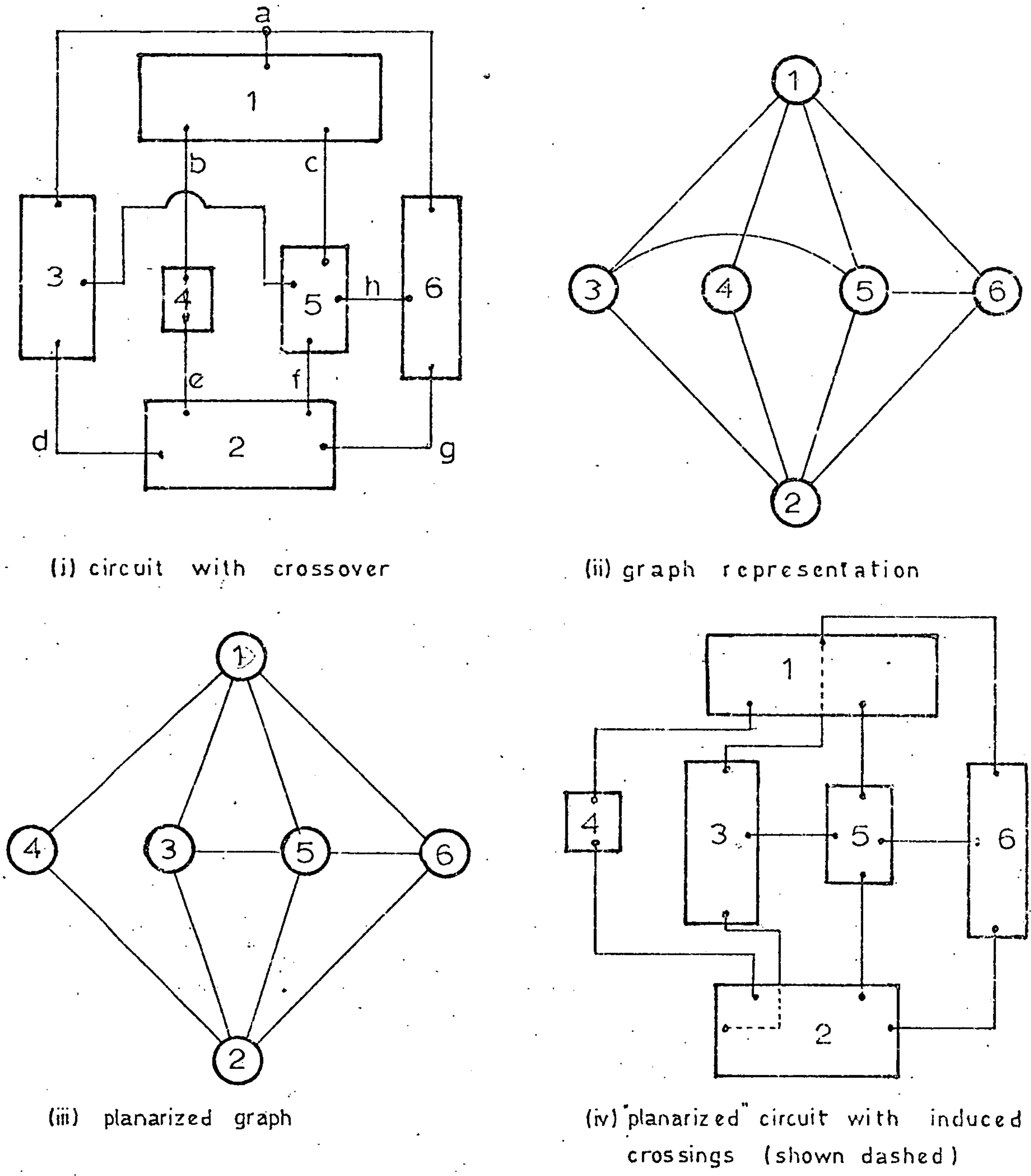
Goldstein and Schweikert (1973) have presented a discussion of the conventional graph representations of circuits. These are as follows:

(i) components to nodes, nets to edges

(ii) components to edges, nets to nodes

(iii) components and nets to nodes

The first representation treats the components as points and the nets as complete graphs on the nodes belonging to them or a set of two node nets, or branches, whose union forms a spanning tree of the net's constituent nodes. This representation has two main drawbacks: first, the terminal pin ordering of the nodes is not considered and therefore the presence of induced crossings (or intracomponent crossings) as in figure 2.5 (a), cannot be detected when planarity is checked (a simple cyclic check of edges would however correct this); secondly, the possible net representations are inaccurate and either over-complicate



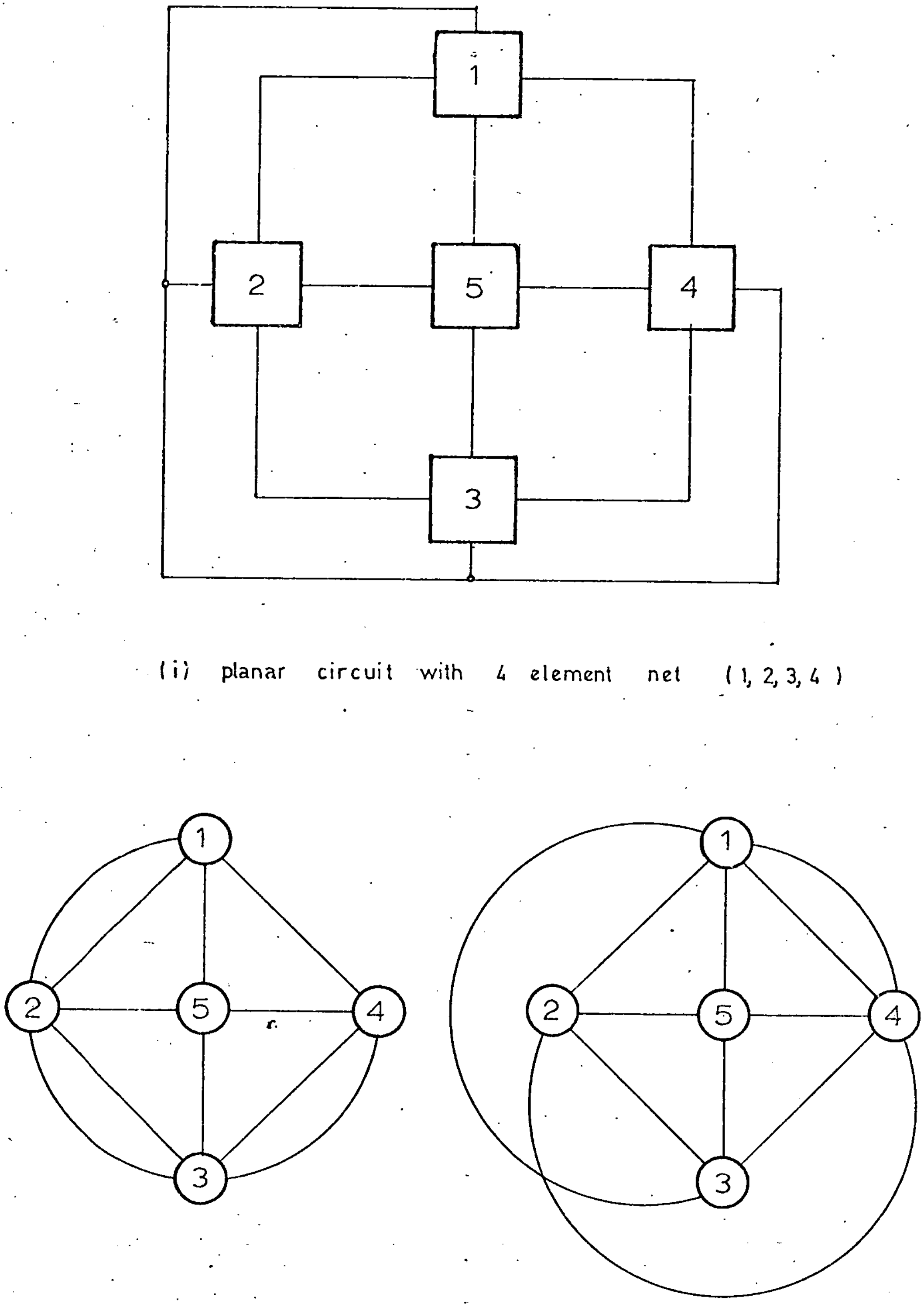
(i) circuit with crossover

(ii) graph representation

(iii) planarized graph

(iv) "planarized" circuit with induced crossings (shown dashed)

Figure 2.5 (a) Planarizing through Induced Crossings. in a Component to node and Net to link mapping.



(i) planar circuit with 4 element net (1, 2, 3, 4)

(ii) net represented as the chain (1,2,3,4) gives planar graph.

(iii) net represented as the chain (2413) gives non-planar graph.

Figure 2.5 (b) Inadequacy of chain (and cycle) as a means of representing Nets in a Net to link mapping.

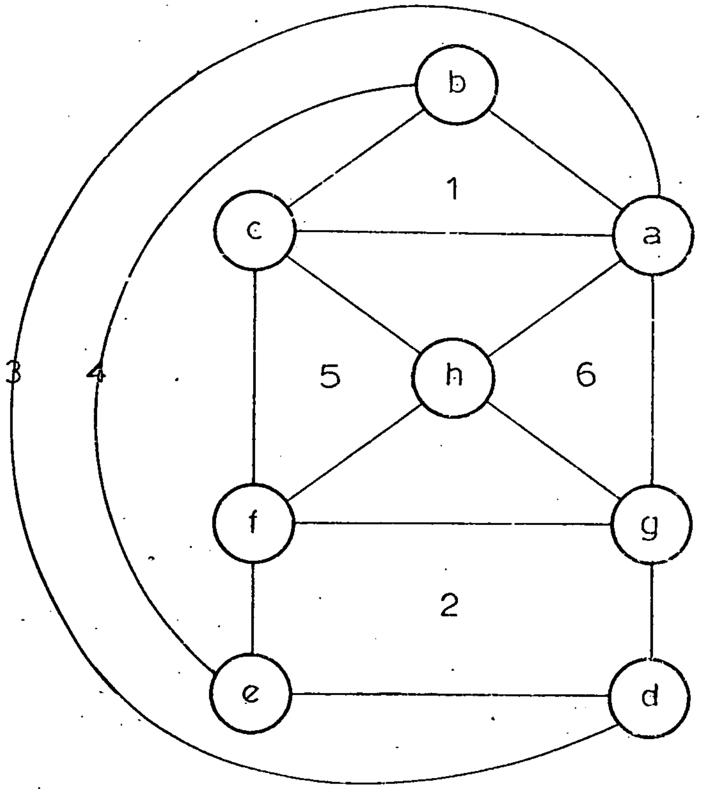


Figure 2.5(c) Component to link and Net to node mapping of circuit 2.5(a)(i) omitting non-planar connection (3,5).

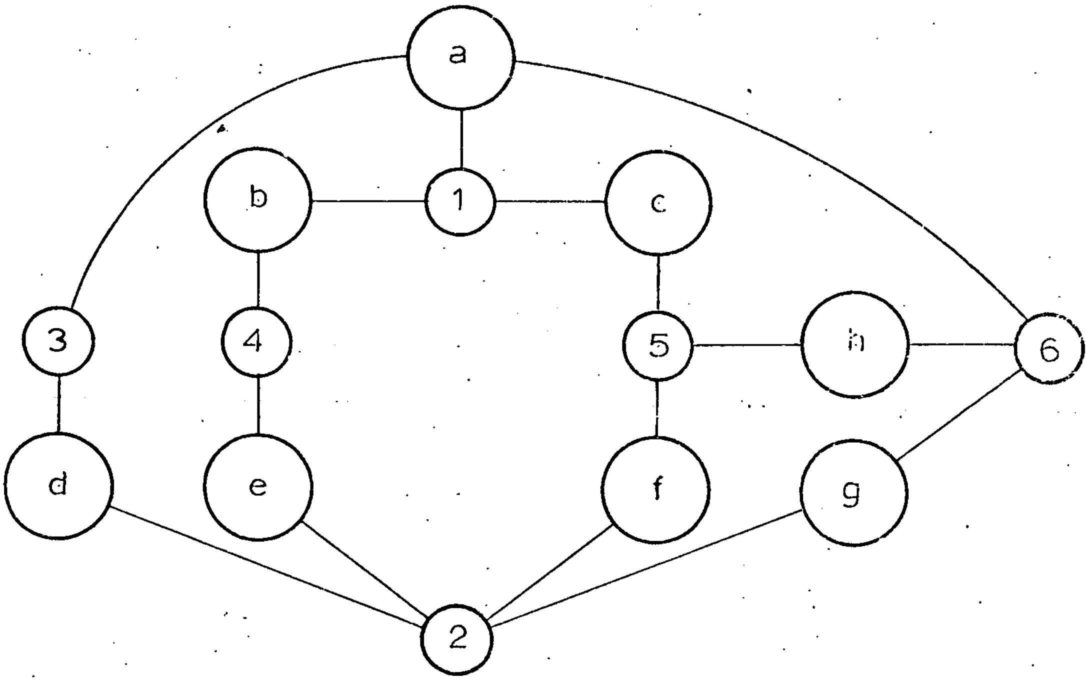
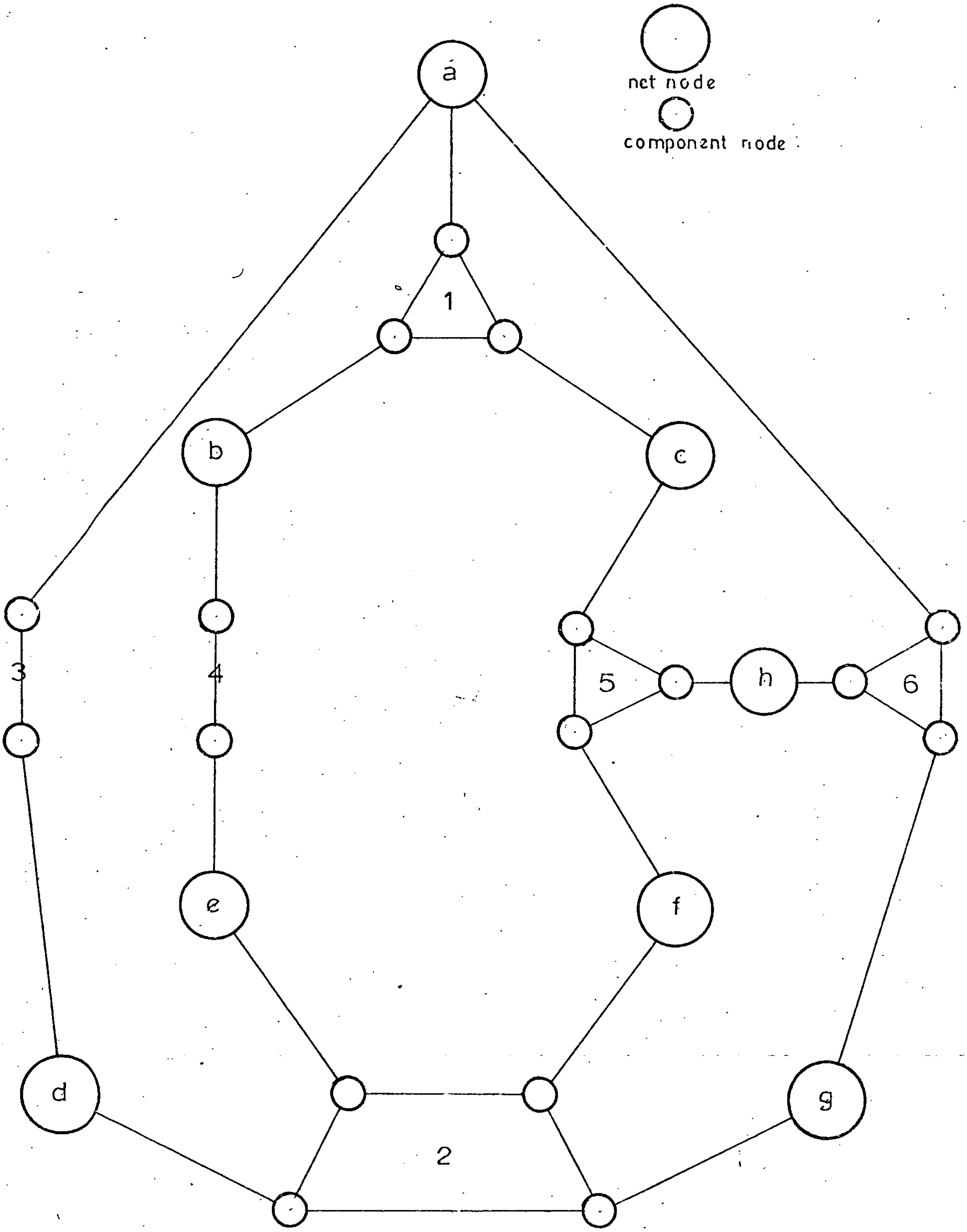


Figure 2.5 (d) Component and Net to node mapping of circuit 2.5 (a) (i) omitting non-planar connection (3,5).



Figur 2.5 (e) Component to subgraph and Net to node mapping of circuit 2.5(a)(i) omitting non-planar connection (3,5).

or over-simplify the circuit, the former making it impossible by Kuratowski's theorem (Kuratowski, 1930) to connect a complete graph for a 5 node net without crossovers, while the latter creates hazards as illustrated on figure 2.5 (b) where different partitionings of the net (1,2,3,4) produce different results.

The second representation uses a node for a net and components are represented by a subgraph containing the nodes corresponding to the nets incident on the component (figure 2.5(c)), thus creating the auxiliary of the earliest representation.

The subgraphs used to represent components, as with the net representation earlier, can be either complete graphs, circuits or an intermediate representation, however since the circuit or chain representation is a fairly accurate picture of the real world it causes few problems and none have been found using a complete graph. This representation has the disadvantage of being very difficult to conceptualise.

The third circuit representation uses nodes for both components and nets, and a net incident on a component is represented by an edge connecting the corresponding component and net nodes (figure 2.5(d)). This is a more accurate representation of a net if the incident edges are conceptualised as the conductors of the net. However, the lack of consideration of terminal ordering with the consequent drawback relating to the first representation also applicable to this one.

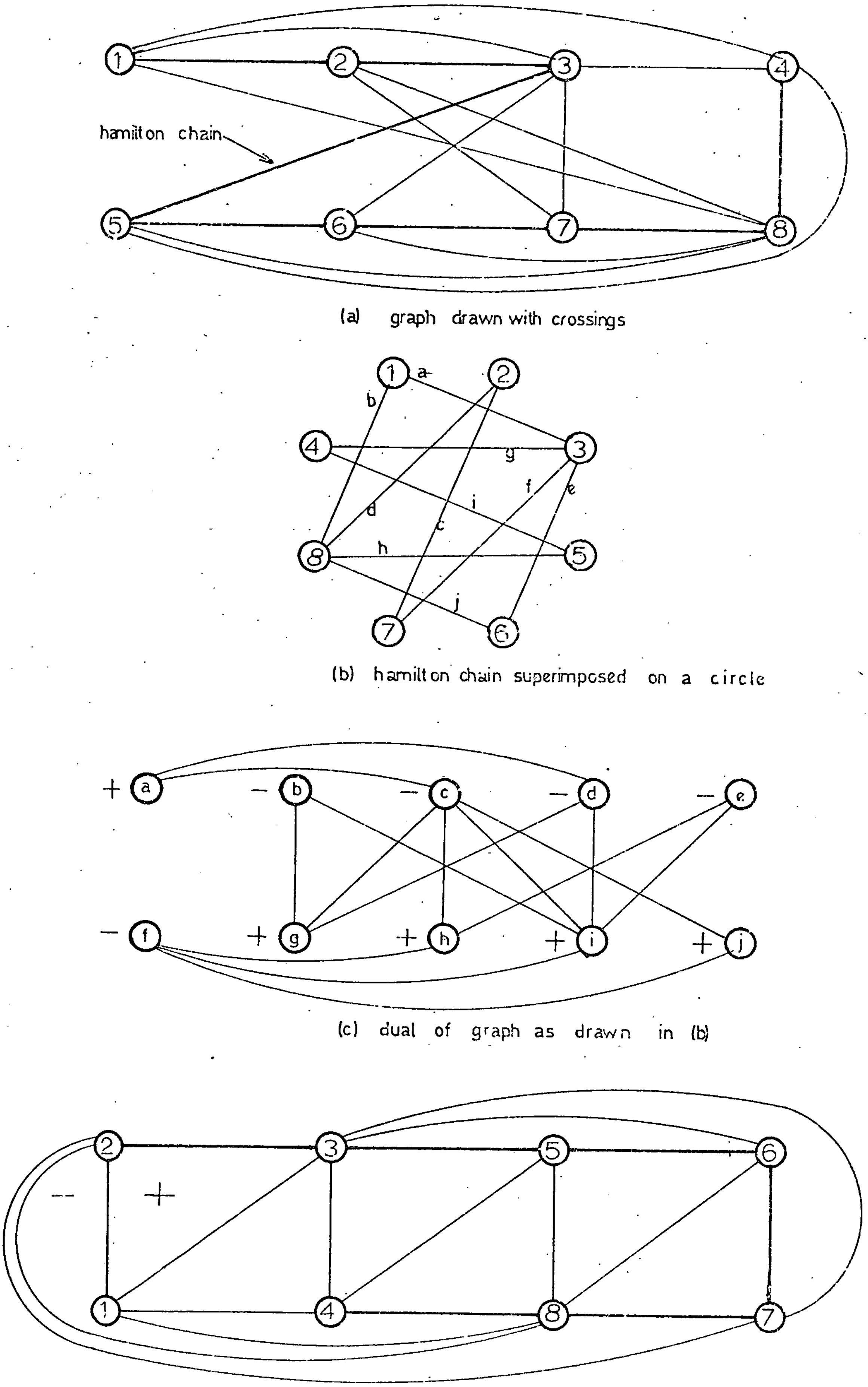
A fourth representation exists (figure 2.5(e)), on which

Goldstein and Schweikert make no comment; it combines the third's net with the second's component mappings. It has been used by Rose (1970) and Fletcher (1970) and appears ideally suited to portray a circuit graphically, as it combines the best features of both the second and third representations and provides the most accurate mapping of all four.

2.4.2 PLANARITY ALGORITHMS

Bader (1964) has produced an algorithm for determining whether a given graph is planar or not. First, a circuit of the graph containing as many nodes as possible is found, preferably a Hamilton circuit (1, 2, 3, 5, 6, 7, 8, 4 in the example in figure 2.6 (a); the graph is then redrawn with this circuit as a circle and branches are drawn as straight lines (figure 2.6(b)), excluding those edges which belong to the circuit. The dual of this graph is drawn and one node is arbitrarily assigned a positive sign and all nodes connected to it are then given a negative sign; the procedure continues assigning opposite signs to nodes connected to signed nodes until a conflict is reached, that is, two nodes of like signs are connected, which indicates that the original graph is non planar. If all nodes are given signs without a conflict arising (figure 2.6(c)) the graph is planar and can be redrawn as follows:

The circuit is drawn, with branches, as the perimeter of a simple area. Branches whose nodes in the dual graph have a positive sign are drawn as straight lines inside the area, whilst negative branches are drawn external to the area (figure 2.6(d)). This method has been developed by



(a) graph drawn with crossings

(b) hamilton chain superimposed on a circle

(c) dual of graph as drawn in (b)

(d) graph redrawn without crossings

Figure 2.6 Bader's Planarity Algorithm.

Fisher and Wing (1966) who have described the procedure in terms of matrix operations on the incidence matrix of the graph. They have also extended it to identify a set of branches which, when removed, produce a maximally planar subgraph of a non-planar graph; this would be of use in printed circuit applications to identify a minimum set of tracks to be replaced by jumper wires. Furthermore, the identification of conflicting branches and subsequent partitioning could be applied to a two-sided board by dividing the branches between the top and bottom instead of an internal and external division. The programmed version of this algorithm was limited to 50 nodes and 160 branches and a graph containing these numbers of elements required 2.3 minutes on an IBM 1620.

2.4.3 GRAPH SYNTHESIS AND ANALYSIS TECHNIQUES

Kodres (1962) describes a technique applicable to multilayer boards with interlayer connections made via component pins. The components are represented by nodes and the nets by complete graphs on the component nodes of the net; edge connection pins are represented by a circuit of nodes, drawn as a circle and defining the perimeter of the board. The component nodes are positioned using a centre of gravity method which takes into account the number of branches in the nets and minimum length spanning trees of each net are determined with the edge length proportional to the number of crossings, including induced crossings caused by the terminal ordering of a node. The dual of this graph is then formed and coloured with as many colours as there are signal conductor layers in the board; if necessary, a minimum number of edges is removed to permit colouring and the last stage attempts to insert the removed edges by using the spaces between the component pins. The planar graph is now partitioned into the layers available by splitting it into planar subgraphs representing the connections on each plane; a piecewise linear transformation is used to convert these subgraphs into a layout which places each component on a grid point belonging to the array of permitted component positions. It is not clear how the detailed routing of conductors is accomplished following the transformation to the component grid; moreover, as no examples of a computer implementation have been cited, its efficiency as an algorithm cannot be assessed.

Rose (1970) has developed a program for laying out single sided, discrete component PC boards in which multi-pin components are represented by subgraphs consisting of a circuit of nodes and pseudo-branches and the two pin elements are included as component branches. A planar graph is constructed by commencing with the edge connector and a circuit of pseudo-branches representing the board's perimeter and then circuit elements are inserted sequentially on a connectivity basis, whilst preventing crossings by removing any branches which generate them. This provides a planar subgraph and a list of removed branches which are re-inserted by a path search method similar to Lee's algorithm; a component branch search proceeds through regions who share conductor branches until the wire capacity of the component is exceeded and conductor paths are sought through regions sharing component or pseudo-branches. Both these searches are designed to make use of the capability of under-component wiring.

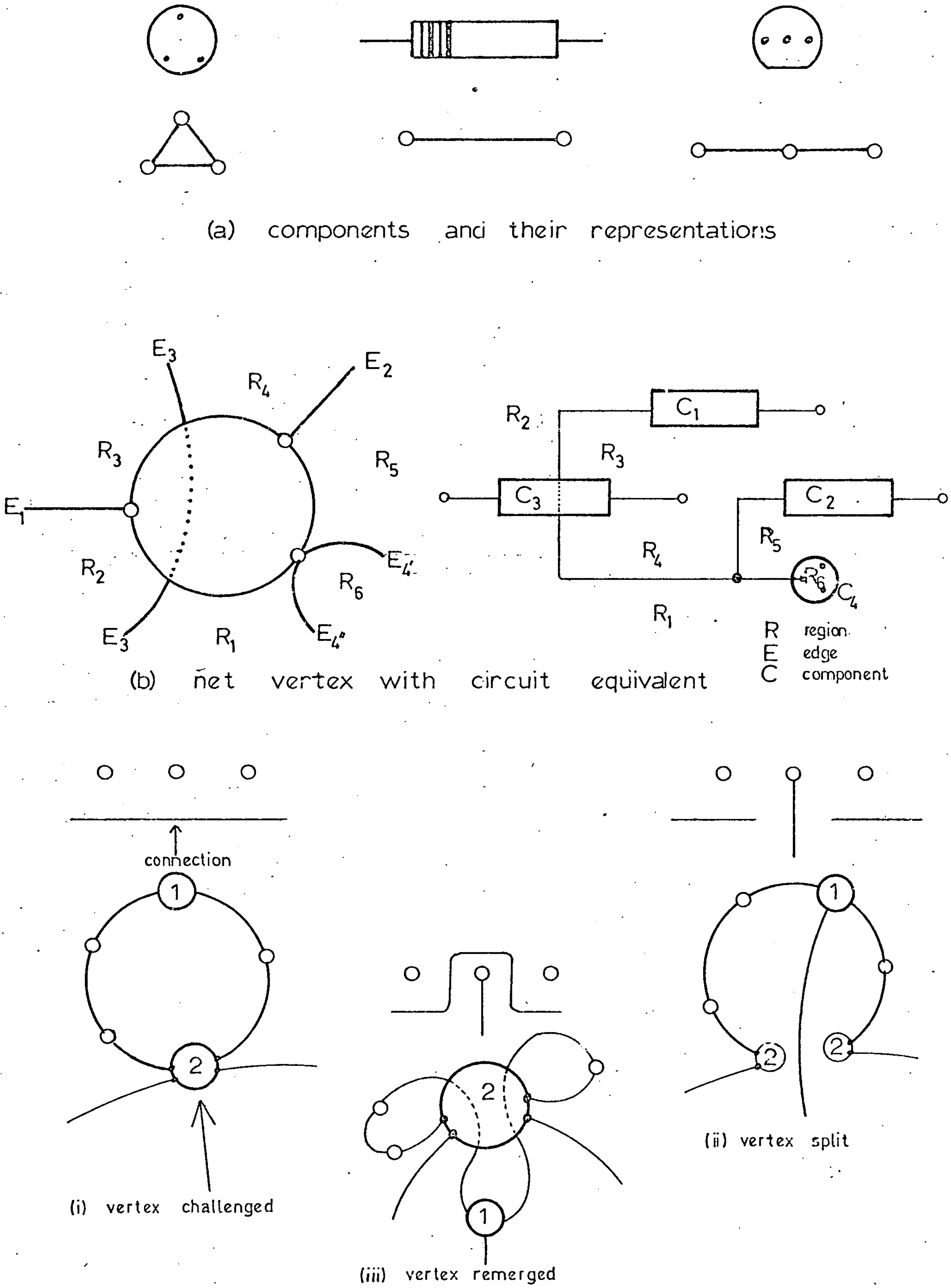
The components are laid out on the board by building up from the edge connector, fitting the components into available spaces so as to avoid overlaps, whilst wiring is generated automatically from the planar graph and placement information. Alterations are possible through interaction via a graphics screen, which allows movement of components and variations in routing. This program produces layouts from a circuit description in about 20 seconds for a 20 component network.

A bipolar integrated circuit layout program has been described by Fletcher (1970), which uses a force technique for the initial placement, followed by a detailed packing of elements such that each is positioned optimally with respect to its neighbours. The links are then considered as point to point connections and a planar representation achieved by deleting links or bending them around the ends of links which they cross. Alternatively, crossover sites on components may be used to remove crossings. However, this technique is of little use in integrated circuits as the deletion of links cannot be considered a valid planarising operation; integrated circuits cannot be intra-connected with jumper wires and so a link removed remains removed until it is re-inserted in the layout.

Fletcher then discusses a topological method, using a graph synthesis technique but he presented no evidence on its performance in real layouts. Finally he presents some details of a proposed method which are difficult to interpret and little can be said about the techniques, described or proposed, owing to the lack of implementation.

Loberman and Weinberger (1957) have developed a procedure for routing nets with a minimum wire length. All n(n - 1)/2 possible branches of an n-node net are sorted by length, with the shortest placed first. The minimum spanning tree of the net is constructed by choosing branches from the head of the sorted branch list. Branches with their two nodes already in the sub-tree are discarded, branches with neither node present are incorporated by forming a new sub-tree, whilst branches with one node appearing in a sub-tree are added. Sub-trees are then connected when a branch having a node in each sub-tree appears at the head of the list. This procedure is computationally efficient to construct nets in isolation, however it would be of little use in a practical routing problem which involves avoiding obstacles.

Basden and Nichols (1973) have developed a synthesis method for laying out printed circuits by producing a planar graph representation which is also applicable, theoretically, to integrated circuits. Components are represented by a vertex-edge chain or circuit (figure 2.7(a)) with an edge capacity assigned to each edge to represent the number of conductors allowed to pass between the pins corresponding to the vertices of the edge, nets are



(a) components and their representations

(b) net vertex with circuit equivalent

(i) vertex challenged

(iii) vertex remerged

(ii) vertex split

(c) routing round pin to prevent crossover

Figure 2.7 Basden and Nichols' circuit representation and Synthesis techniques.

represented by a vertex which is incident on those vertices representing the pins belonging to a net (figure 2.7(b)) and edges representing the perimeter of the board have a capacity of zero, which constrains the conductors to lie within the area of the board. Initially, the graph consists of a circuit, representing the edge connectors and board perimeter, and then components are added by inserting their representative subgraphs according to a set of elementary layout rules and inter-pin conductor routing so as to avoid crossovers. The routing of a conductor between a component's terminal pins is achieved by moving the vertex, representing the conductor, so that it lies on the edge between the vertices, representing the pins between which it passes; this is illustrated in figure 2.7(b) where the conductor passes under the component C_3 represented by edge E_3, thus reducing the edge capacity of that component edge by one.

The insertion procedure is carried out in terms of the regions of the graph, a subgraph being inserted in a region and connections made by moving vertices. If it is impossible to route a connection (figure 2.7(c)(i)), a conductor vertex can be split into two vertices (figure 2.7(c)(ii)), which is equivalent to breaking a conductor. When the components have been connected as far as possible, any split vertices must be remerged, and this is achieved by moving one half of the split pair onto the component edges to allow remerging, which is equivalent to routing a connection between the broken ends, that is, between conductor pins (figure 2.7(c)(iii)). Any conductors which cannot be routed, or any split vertices which cannot be remerged, are listed and presumably would require jumper wires or re-assignment to another wiring layer.

The technique, as presented, gives no indication of how the layout would be realised from the graph, that is, how the components would be placed; indeed the only example cited is the connection of two components and a 4 terminal edge connector from which it is impossible to make any real estimate of the program's capability to deal with realistically-sized circuits.

Engl and Mlynski (1972) have developed an algorithm which treats the circuit's placement and routing as a coupled surface layer problem and they claim it is applicable to all technologies. The method relies on two requirements; first, the graph must be uniquely assigned to the circuit and, secondly, the planarity of the graph is necessary and sufficient for the existence of a planar layout of the circuit.

The components are represented by nodes and the nets by 'spiders' of edges. The graph is planarised by the application of two planarity operators, termed extension and reduction, which split a node in two and delete a spider edge, respectively. The relevance of these operations to the layout of components and tracks appears to be in routing a track through a component and deleting a connection. Although the analagous operations necessary in the various technologies are not discussed, a table of means of circumventing crossovers is included. Repeated application of these operations can reduce any non-planar graph to a planar graph, but it would most likely be a subgraph, with connections absent and no means to re-inserting removed edges is discussed. A further point of weakness in the algorithm is that it depends on human assistance, via an interactive graphics terminal to apply the planarising operators, which is almost equivalent to an interactive program where a human can re-route conductors through certain nodes, dependant on the technology involved, and delete other conductors to produce a planar layout. No mention is made of the means by which the resulting planar graph is transformed into a circuit layout, thus little can be said about the algorithm's utility.

Narraway (1971) has developed a method for establishing the planarity of a graph which can be used to generate a wiring pattern for a given placement. The components are represented by sub-graphs with one node per terminal; this representation is subsequently contracted to a single node per component during the analysis. No mention is made of the representation of a net but Narraway probably uses a sequence of n - 1 branches to represent an n-node net. The program requires a Hamilton chain of the circuit as a starting point, which is obtained by matrix operations performed on the connection matrix of the contracted graph; if a Hamilton chain does not exist, dummy branches are inserted to create one. The Hamilton chain is then drawn in a vertical line and branches are drawn as right π, 2π or left π loci (figure 2.8), with the order of connection

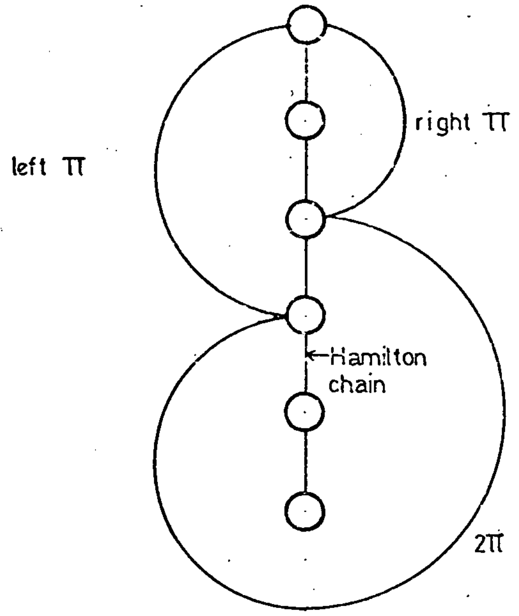


Figure 2.8 Narraway's Loci Classification.

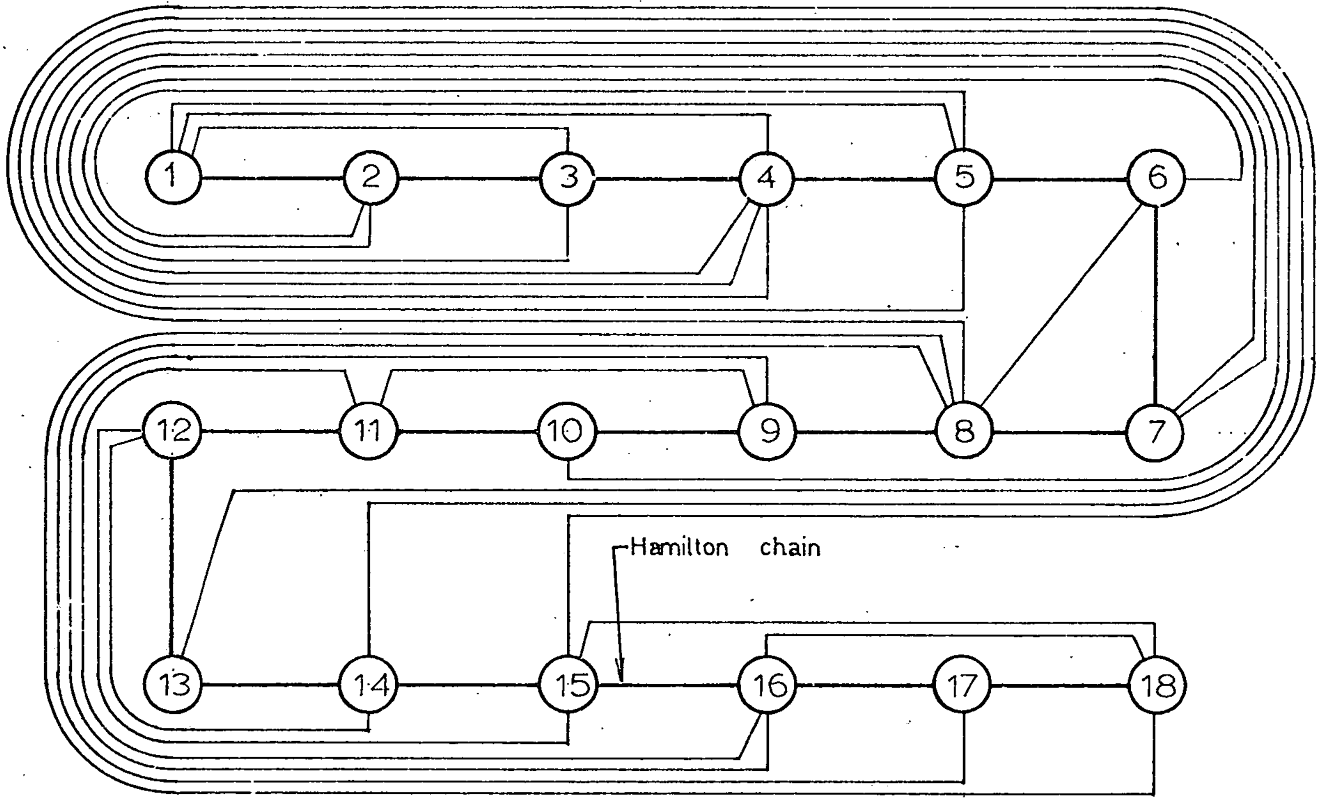
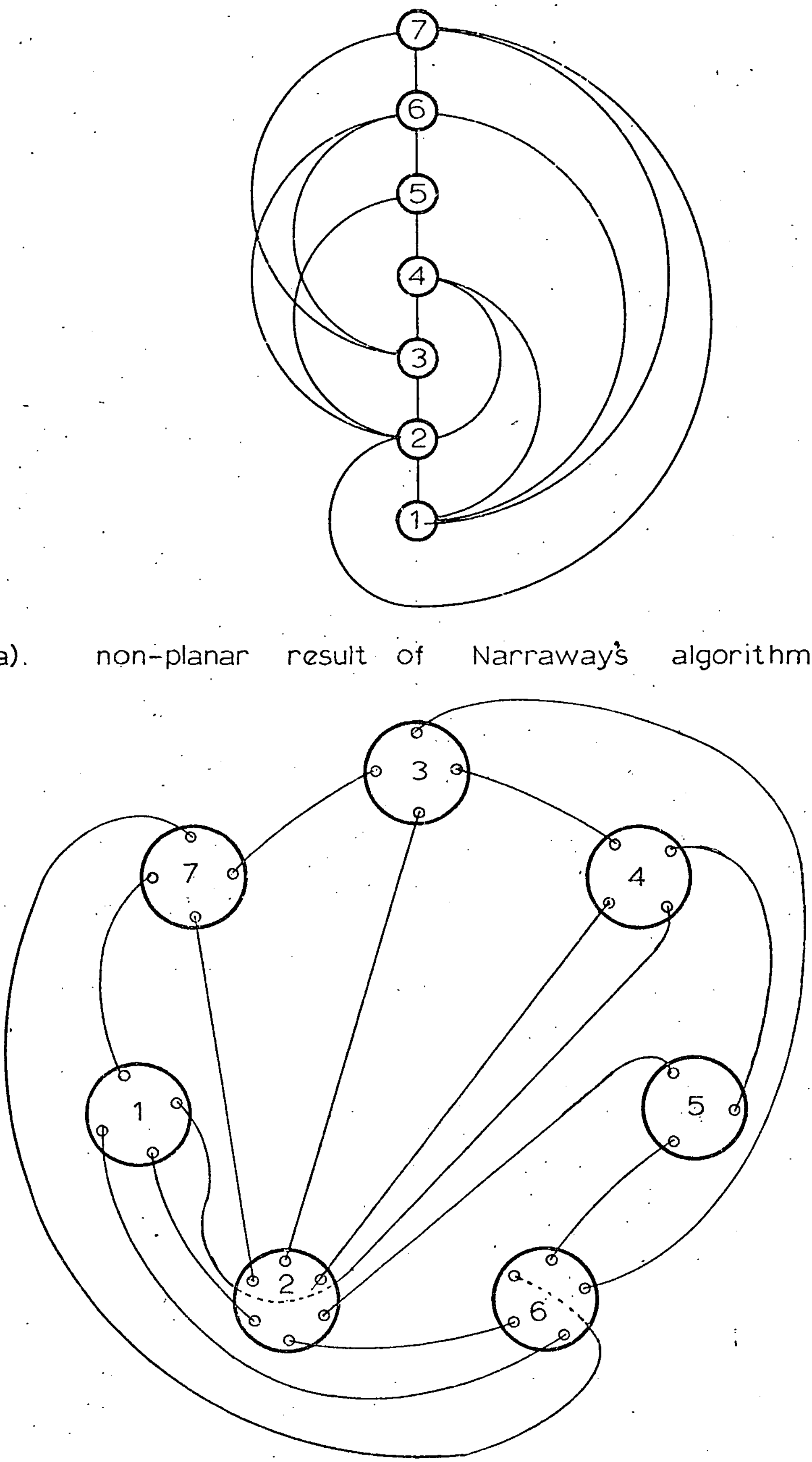


Figure 2.9 A 6x3 board routed Topographically using Narraway's algorithm.

from the top node to the bottom and drawing the branches joining each node to nodes below it in the chain. Branches are drawn by trying the loci in the order mentioned above, using the configuration of the first planar locus tried. If all would create a crossing then a right π locus is used and any loci it crosses are converted into left π loci; Narraway states that this procedure will establish the planarity of a graph while producing a planar drawing, if possible, of it. The nodes can then be expanded to the subgraphs without inducing crossings as Narraway proves that any n points on a circle can be connected to a set of n points inside the circle without crossings.

When applying this system to a PC board layout, Narraway generates a Hamilton chain as a simple curve, drawn through the components on the board, having assumed a prior placement, and then dummy branches are inserted where the curve does not coincide with an existing branch. The graph produced by the above method is then imposed on the board by drawing the loci in the same relationship as they appear on the graph, which Nicholson (1968) has proved feasible. There are several problems encountered when applying this theoretical approach to a real problem; for instance, the layout shown in figure 2.9 derived by this method is clearly inefficient and would not be acceptable because there are no standard boards available which allow 12 tracks between adjacent rows of components. Although it is theoretically possible to represent the component as it appears on the board by expanding a node to a subgraph, the physical limitations on the number of tracks allowed between component pins are not considered, nor does the graph take into account track width and available space. Furthermore, the circuit drawn in figure 2.10(a), although non-planar according to Narraway's algorithm, does have a planar realisation as shown in figure 2.10(b). It would therefore appear that this technique, although of theoretical interest, has little practical value for printed circuit boards.

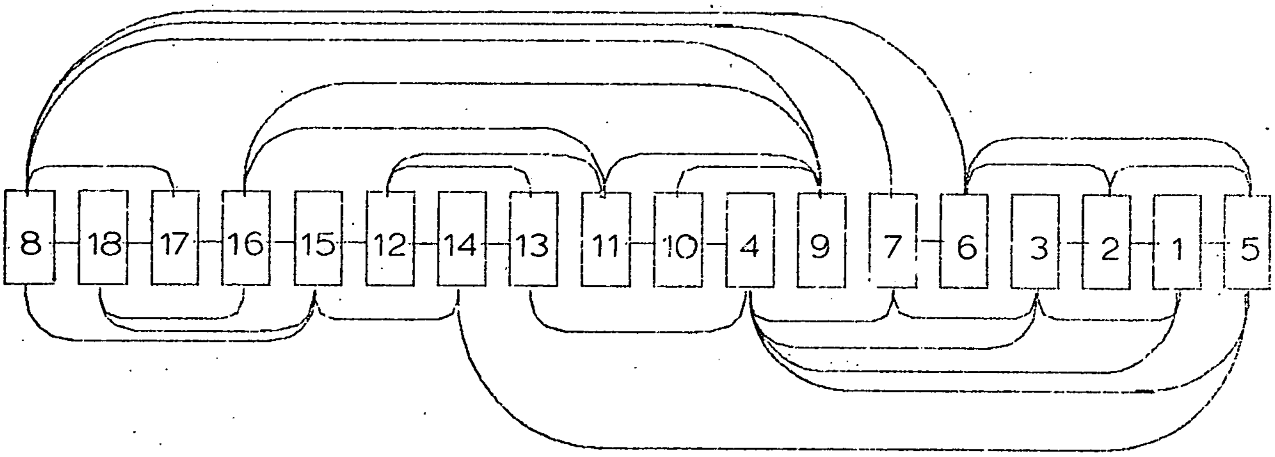
An algorithm for minimising the number of crossings in a graph, developed by Nicholson (1968), is superficially similar to Narraway's and although it is purely theoretical it is mentioned here for comparison. A node line is used instead of a Hamilton chain; the node line consists of a simple curve drawn through the graph's nodes together with the nodes themselves, and Nicholson describes it as a permutation of the nodes. An initial permutation is created by commencing with one node in the node line and the one with the most branches incident on it is selected. Nodes are then injected into the node line, with each one selected on the basis of the number of connections it has to nodes already in the node line and positioned in the node line so that the increase in the number of crossovers is minimised. Branches are drawn as a composition of left and right π loci as shown in figure 2.11, which can be compared with Narraway's layout of the same network in figure 2.9. Finally, minimisation of crossovers is attempted by altering the permutation (nodal order); nodes with the most crossovers are examined in all other positions and the number of crossovers calculated and if the change decreases the



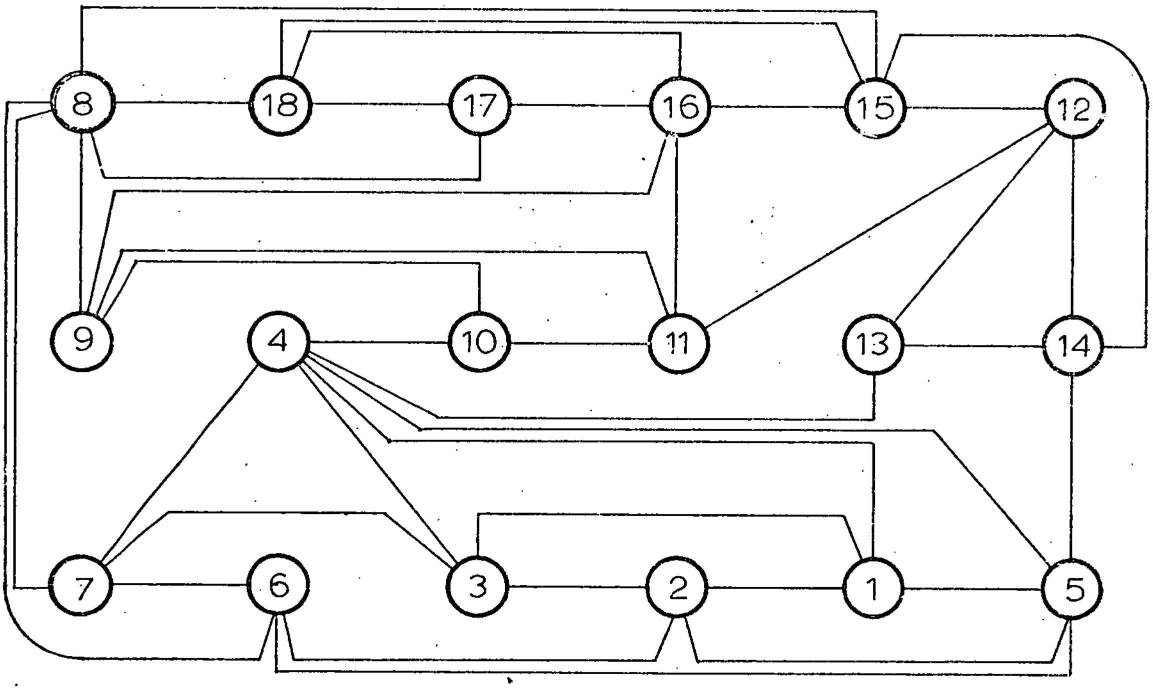
(a). non-planar result of Narraway's algorithm

(b) pseudo-planar circuit realization

Figure 2.10 Narraway's algorithm does not allow for Under-component Wiring.



(a) the optimized permutation



(b) corresponding layout

Figure 2.11 Solution to circuit of figure 2.9 obtained using Nicholson's Algorithm.

crossovers it is made permanent. The algorithm terminates when no further decrease is possible through altering the position of one node. This procedure will be discussed fully in chapter 3.

2.5 CONCLUSION

The placement part of the layout problem has been partitioned into two cases, the constrained and the unconstrained; the constrained case is that where the components of the circuit are confined to a predefined, usually regular, array of positions on the board. Algorithms either commence with a manual or arbitrary placement, with or without dummy elements to occupy any vacant positions, or generate an initial placement, based on the sequential addition of components determined by connectivity. Optimisation of this positioning is attempted via interchange of groups of elements so as to minimise a function of the length of wiring.

The unconstrained placement algorithms so far developed employ the force technique which aims to minimise potential wiring length, resulting in a minimum of spatial requirement for wiring, although areas with potential high wiring density have not, as yet, received any special consideration.

The routing portion of the layout problem is divided between positioning one layer of wiring and cases where several wiring layers must be routed. The former requires a planar layout as crossovers will produce short circuits, whilst the latter case has been tackled two ways; two layers, such as two-sided PC boards are routed using segmented wiring and when more than two wiring layers are present the routing problem is largely a matter of assigning conflicting connections to different layers, once the connection positioning is finalised.

The advent of Lee's algorithm has provided a powerful, but expensive, means of laying wiring paths; its wide usage and many modifications have proved its popularity, but it is limited by the amount of storage available in the computer. When incorporated it is used, with the one exception of Fisk (1971), to route individual paths with little regard for creating bottlenecks or producing an even wiring density, which affects circuit performance. Many route sequencing schemes have been proposed to avoid the deterioration in length as the routing proceeds; they generally leave paths with many possible, equi-distant alternatives till last on the assumption they will be the easiest to route.

Layout programs which separate the problem into placement and routing and treat them as two sub-problems separately, have a common deficiency. They lack an overall guiding scheme, that is, a procedure for solving one sub-problem which bears in mind general criteria to facilitate an easier solution of the other; as many authors have observed, the two are highly inter-related and consequently, algorithms for their solutions should be likewise related. At the very least, a feedback mechanism should be incorporated so than an exchange of information between the two segments is possible, and this is the main reason why a successful layout program cannot be produced by amalgamating a placement technique with a routing algorithm. Graph theoretic or topological techniques attempt to overcome this deficiency; the analysis or synthesis of a graph enables the two sub-problems to be considered simultaneously, by constructing a representation of a suitable wiring placement which can be consulted as the component placement is executed. This allows space to be left where space is required and the close-packing of elements in areas of few tracks, and thus the problem is then one of realising a layout from a suitable graph representation; Rose (1970) has produced such a technique. It has been found necessary, however, to augment it with interactive facilities to produce realistic layouts, which are of a similar standard to manual products. The majority of the graph theoretic techniques discussed above have not, yet, reached the stage of implementation and of those that have, none have been used, to my knowledge, for actual production purposes.

CHAPTER III

STUDY STRATEGY AND THE TOPOLOGICAL ANALYSIS

3.0 STATEMENT OF INITIAL AIMS

Discussions at the start of this project with people associated with firms who were interested in the GAELIC system indicated that many felt computer aids to layout were of great importance. A common opinion held was that CAD capability was necessary to maintain competitive standards, by providing a faster and more efficient production process. Many firms with little CAD experience were 'looking around' for layout aids without any clear idea of the features which they required. This is a hazardous business as the effectiveness of an aid within a firm can only be judged through use, often several months use, which is necessarily subsequent to purchase.

The review of literature on automatic layout techniques has demonstrated the proliferation of layout techniques aimed at assisting microcircuit design by transferring some of the work load to a computer. The programs are all semi-automatic in the sense that none produce layouts which are fed directly into the circuit production stage. Some provide a complete automation of the layout process, but the results of these programs are qualitatively inferior to existing manual layouts and therefore do not warrant production.

There is a strong case for a program producing trial layouts: the designer can obtain an inexpensive layout which could provide the basis for a final layout or, at worst, highlight the problems associated with laying out a particular circuit. If designed with this aim in mind, such a program would not require the quantity of constraints needed for a final layout and would therefore be a faster and less expensive aid. The individual standards need not be considered because they only become significant in the final layout stage and therefore the program has potentially wide-ranging application within industry. However, the main advantage is that a rough layout circumvents much of the initial, time-consuming experimenting of the layout phase; thus the designer would be required to rearrange, not construct.

Cullyer et al (1968) gives the only example of a procedure to provide trial layouts, rather than realistic layouts; however, the program has severely limited capability as it has a maximum circuit size of 20 components and the layout must be contracted manually to the required size. Despite this the program has more value than others of greater power, because it acknowledges that the results are of a trial nature and not meant to be final.

3.1.0 PROGRAM AIMS AND CAPABILITIES

On the basis of the opinions expressed by those involved with industrial microcircuit production, the scope of the work was confined to industrially feasible layout techniques. Consequently it was decided to carry out the study by producing a layout aid whose use was confined to equipment widely available to industry. This limited the hardware to a remote, teletype terminal, accessing a time sharing bureau. The teletype is confined to a transmission speed of 120 baud (10 characters per second) using modems to interface with the GPO telephone lines; this limited rate of information exchange means that transmission of large quantities of input/output data, such as required by interaction, becomes tedious, time-consuming and consequently expensive, therefore an additional constraint was placed on the program's design, namely that the program would have to be capable of working largely without human assistance which, coupled with the hardware limitations, ruled out interaction.

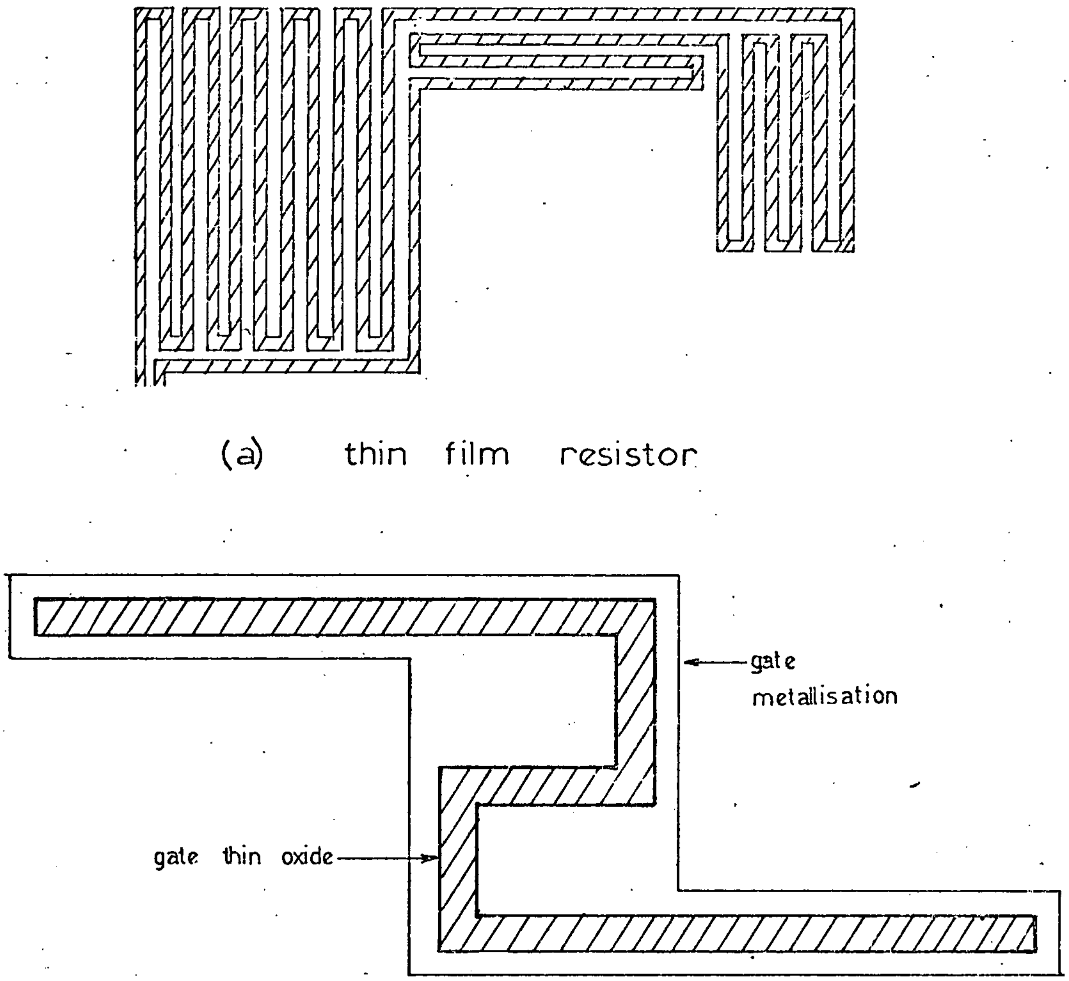
Having defined these limitations, it was necessary to decide the scope of the output, that is, the level of sophistication. Trial layouts were chosen on the basis of two reasons; first, the review of literature indicated that this was an area which lacked basic research, and secondly, if the basis of the study was the production of a working layout aid it would be both time-consuming and unproductive to study the layout procedure in depth. The overall aim of the work became, therefore, to evaluate the effectiveness of automatic microcircuit layout aids, rather than to produce a marketable program, that is, the program was to be the means to a realistic evaluation of non-interactive methods.

3.1.1 CIRCUIT INPUT

The program's application was originally confined to metal oxide silicate integrated circuits, but restriction to a subset of ICs would be of little use in providing an overall evaluation because the technique applicable to one technology might prove to be of little use in another. It was decided, therefore, to view the layout problem as one of positioning components, treated as rectangular areas, with a single layer of connections, thus providing an aid appropriate for use within a wide range of microcircuits and therefore in keeping with the basic aim of generality.

It was decided to confine the wiring to a single layer of connections both for simplicity and because the majority of integrated circuits manufactured at present are restricted to one metallization layer by technological difficulties. The rectangular component representation is common, almost to the point of convention, and is sufficiently accurate for most microcircuit components; non-rectangular component description was considered unnecessary as it merely complicates the placement with extra calculation, such as procedures for checking overlap of components.

The main advantage of this simple circuit description, apart from computational efficiency, is its generality. Both the integrated circuit cells as well as most discrete components mounted on backboards and substrates are designed on a rectangular basis. Diffused resistors and deposited resistors, such as in bipolar ICs and film circuits, together with integrated transistors which have a very small aspect ratio, are often folded to provide a more compact shape. These need not necessarily be confined to a rectangular outline in practice, (see fig. 3.1), since their overall length is the most significant dimension, however a suitably-sized, rectangular area is a sufficiently accurate representation.



(a) thin film resistor

(b) MOS transistor

Figure 3.1 Examples of Components which are not confined to a Specific Outline.

Again to reduce the computational requirements, connections would be confined to a uniform width. This is not quite accurate, especially in integrated circuits where diffused tracks may be used for connections and the minimum widths of metal tracks and diffused tracks are generally different. Also, printed circuits often have different track widths for connections which conduct different current levels and, in general, connections are made as wide as possible to lower the resistance of the link. However, these considerations are once again of major significance only at the final layout stage and can be overlooked in trial layouts. Alternatively, all tracks may assume the maximum width specified which will produce a feasible layout, but one with a higher proportion of wiring area than necessary.

Finally, there are generally two scales in the design of large circuits; the circuit is partitioned into sub-circuits, each of which is laid out independently, and these cells are then laid out together as an additional process. This corresponds to the process used by Case et al (1964), or to the design of a large integrated circuit using standard cells, such as adder units, flip flops and clock drivers, each of which has been designed independently. The two stages of design are very similar except in scale and it was hoped to produce a method which would be applicable to both scales.

3.1.2 PROGRAM SPECIFICATIONS

An automatic layout program must contain internal indications as to which is the better of any available choices; these are called the criteria of optimality and have been discussed briefly in section 2.1. They are concerned with the minimisation of area and wiring length and the removal of crossovers. The minimisation of area is of greatest importance because it directly affects yield in integrated circuit production; in technologies which use standard-size substrates or backboards, the circuit's dimensions are clearly limited by the standard size and minimisation of area is therefore necessary to fit the circuit into the required space. As crossovers produce short circuits it is of little use to produce a minimum-area layout which contains crossovers, however, as the results are not final but are rather intended as a guide, the removal of crossovers is best left to a designer who can use the features of the particular technology to eliminate them, thus the program should only minimise but not eradicate them. Although conductor lengths obviously should be kept as short as possible, this is the least important of the criteria as it does not directly affect the circuit's production. The circuit's performance is affected by the connection lengths, especially when propagation delay times are significant, however, an experienced designer can best decide which links are critical and their subsequent, detailed positioning.

There are two additional constraints which must be included for accuracy, namely, components must not overlap and connections must be confined to the areas between components. Component overlap is a general constraint in microcircuit layout except in some isolated cases, for example, thin film substrates allow deposited resistors underneath added components. Nevertheless, the restriction is sufficiently universal to be an inevitable inclusion. Component wiring is far less uniform; printed circuits, thick and thin film substrates, all facilitate under-component wiring as the components do not appear in the plane of the board and some integrated circuits allow over-component wiring, such as metal tracks crossing bipolar, diffused resistors. An integrated circuit component, whether it be a solitary device or a predesigned cell, cannot have conductor tracks appearing over it as the elements contain metallisation themselves, and so this restriction is a necessary feature of any method applicable to integrated circuits.

The final specifications for the layout program, therefore, fall into two categories; the fundamental restrictions:

(i) no component overlap

(ii) no under-component wiring

and the criteria of optimality in order of priority:

(i) the minimisation of circuit area

(ii) the minimisation (but not necessarily elimination) of crossovers

(iii) the minimisation of connection lengths.

3.2 METHOD OF APPROACH

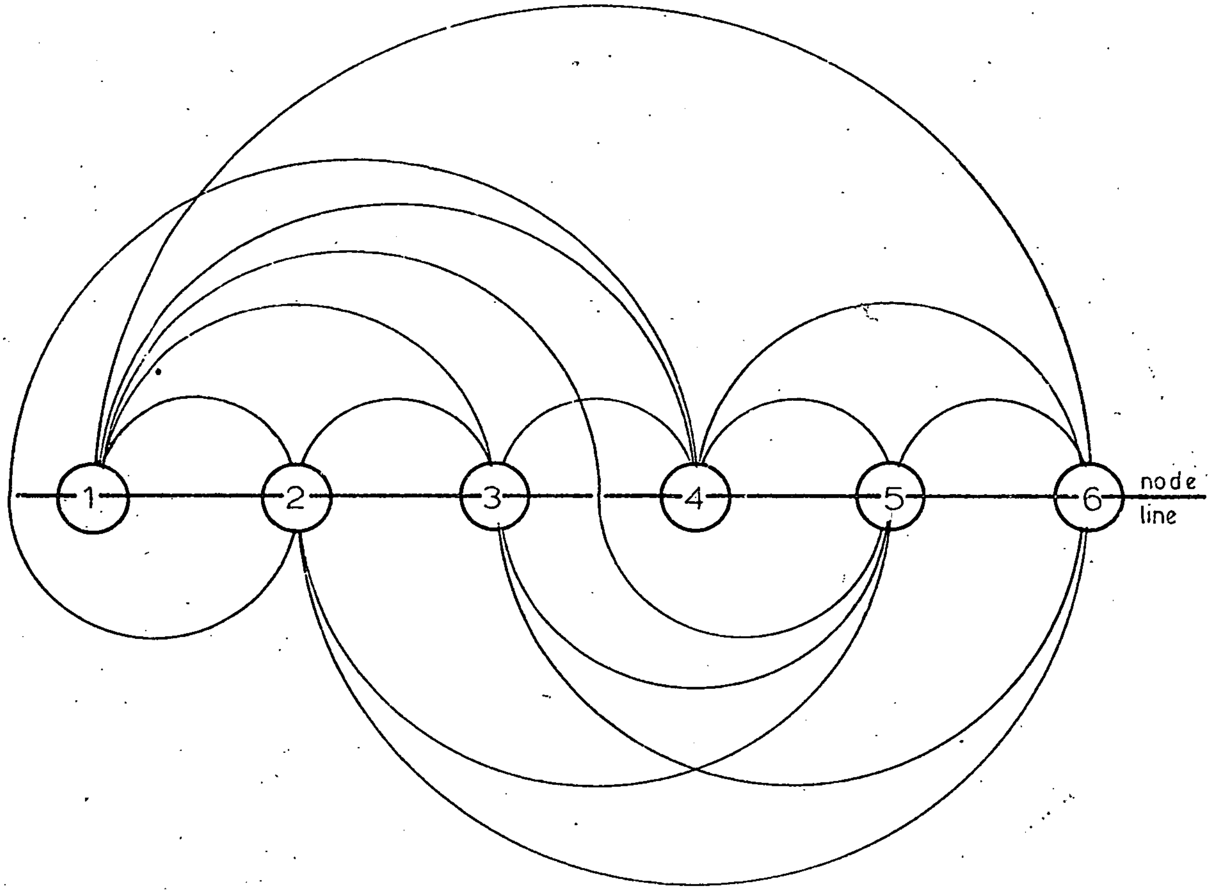
Layout of microcircuits consists of two inter-related sub-problems; the placement of components and the routing of connections (see chapter 2). Since a connection cannot be routed until both the components it connects have been placed, the most computationally efficient method is to place the components and then execute the routing. This requires the placement algorithm to reserve space between components for the wiring, so that there is sufficient space available for the routing, as well as providing information on the areas that must be avoided. Moreover, the placement algorithm should have information on the connection pattern such that the required area is left in the appropriate places. The obvious means of providing the information on the relationship between components and wiring is a topological algorithm. A graph representation does not consider dimensions, but permits analysis of the inter-relationship between the position of components and connections, thus providing detailed information on the spatial routing requirements in relation to the components.

At this stage the criteria of optimality should be considered, specifically the minimisation of crossovers. Providing the results can be incorporated at the layout stage, a topological analysis technique is also ideally suited to executing this process. Such a scheme readily allows the program to incorporate the optimality criteria in a similar order of priority as specified above. Once the crossovers are minimised topologically, placement and routing can proceed such that each optimises the layout with respect to area and conductor length respectively.

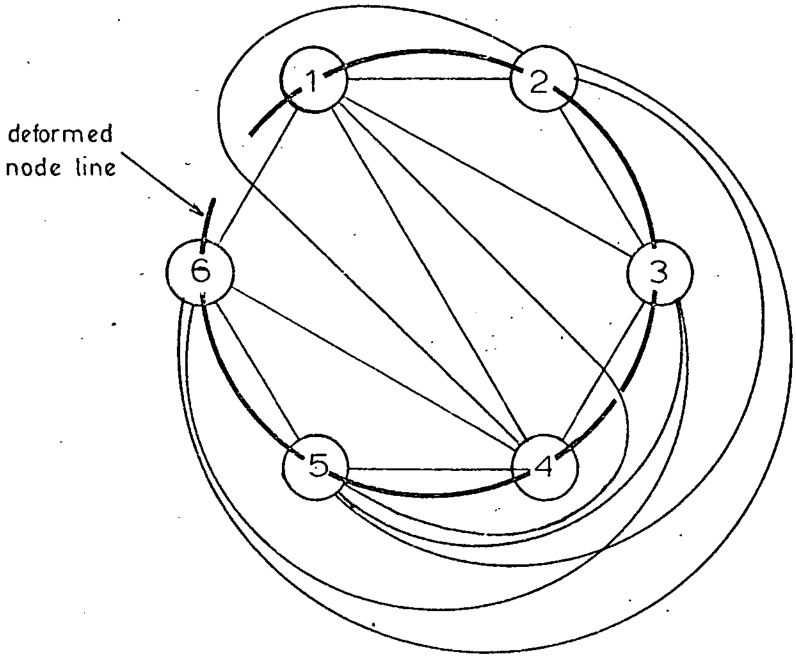
Having defined these conditions it appeared that Nicholson's algorithm "A permutation procedure for minimising the number of crossings in a network" (Nicholson, 1968) contained some of the desired features for the topographic analysis. On closer examination, the algorithm appeared capable not only of providing a sufficiently detailed account of the relationship between components and connections, but also of minimising cross-overs. The remainder of this chapter is devoted to a detailed description and discussion of the algorithm, together with a presentation of results which indicate its capabilities of analysing a range of networks.

3.3.0 DESCRIPTION

The algorithm will be described in terms of a network of nodes and links, each link connecting exactly two nodes (a discussion of representing an electrical circuit in these terms is presented in section 3.6). The network is described as a permutation of the node and link positions. The nodes are positioned at equal intervals on a straight line, called the node-line. Links are drawn as semi-circles above or below the node line, or in some cases a series of semi-circles alternating about the node line, with the extreme ends incident on those nodes which the link connects. Figure 3.2(a) shows the permutation description of the complete graph on six nodes and figure 3.2(b) shows the same configuration except that the nodes have been arranged to lie on the circumference of a circle. This representation is achieved by a continuous deformation of the node-line into a circle, with the links being similarly affected, so that in this case the interior of the circle is equivalent to the region above the node-line. All the links are single semi-circles except (2,4) and (1,5) which are both represented as pairs of semi-circles, one above and one below the node-line.



(a) Permutation Description



(b) Circular Node Line Deformation of above

Figure 3.2 Representation of the Complete Graph on 6 nodes.

In order to justify this representation, Nicholson proves that any general network may be redrawn as an equivalent network, as described above, having the same crossings structure (see appendix III for the proof). A corollary of this proof is that, if a network contains a Hamilton chain on which no crossings occur then it can be redrawn, as above, with the links all represented as single semi-circles. This is significant as, if link representation is restricted to single semi-circles, the algorithm is considerably simplified, as will be demonstrated later.

The network, consisting of N nodes numbered from 1 to N, and M links described as a set of node pairs, can be described by an ordered list of the integers 1 to N. This permutation (P) is defined as

$(P) = (P_i \;/\; i = 1,N)$

P_i = node occupying ith position on the node line

The links are ordered by first node, then second node, that is, link set $1 = \left\{(n_i, n_j) \;/\; n_i \neq n_j,\ Cij \neq \emptyset\right\}$ where

Cij refers to the connection matrix

and if $l_k > l_m$ then $n_{i(k)} > n_{i(m)}$ or

$n_{i(k)} = n_{i(m)}$ and $n_{j(k)} > n_{j(m)}$

This procedure uniquely defines the linkage order as that obtained by a row by row scanning of the upper, right-hand diagonal of the connection matrix. The link routing may now be defined by a list of m indices which indicate whether the corresponding link lies above, or below, the node line, that is,

$(\Omega) = (q_i \;/\; i = 1, m)$

$q_i = +1$ if ith link above node line

$q_i = -1$ if ith link below node line

This description is possible only by constraining the links to appear as single semi-circles. If sequences of alternating semi-circles were permitted, the linkage description would be much more complicated as information on the number of semi-circles and their positioning relative to the nodes would have to be included.

The network may now be described by concatenating the permutation P which describes the node positions, with the ordered list Q, which describes the link positions. For example, consider the graph in figure 3.3(a) with its permutation in figure 3.3(b); the nodes are numbered 1 to 6 and the ordered link set is (1,2), (1,4), (1,6), (2,3), (2,4), (2,5), (3,4), (3,5), (3,6), (4,5), (5,6). The permutation of the nodes (P) as shown is (3,5,6,1,4,2), the linkage structure is described by the sequence (Q) which is (1,1,1,1,1, -1,-1,1,1,-1,1) and the network description (P:Q) is then (3,5,6,1,4,2:1,1,1,1,1,-1,-1,1,1,-1,1).

The procedure for calculating the number of crossings in a permutation (P,Q) relies on the connection matrix C = (Cij). This is used to define two auxiliary matrices, each of the same size, called the upper and lower connection matrices, A and B respectively.

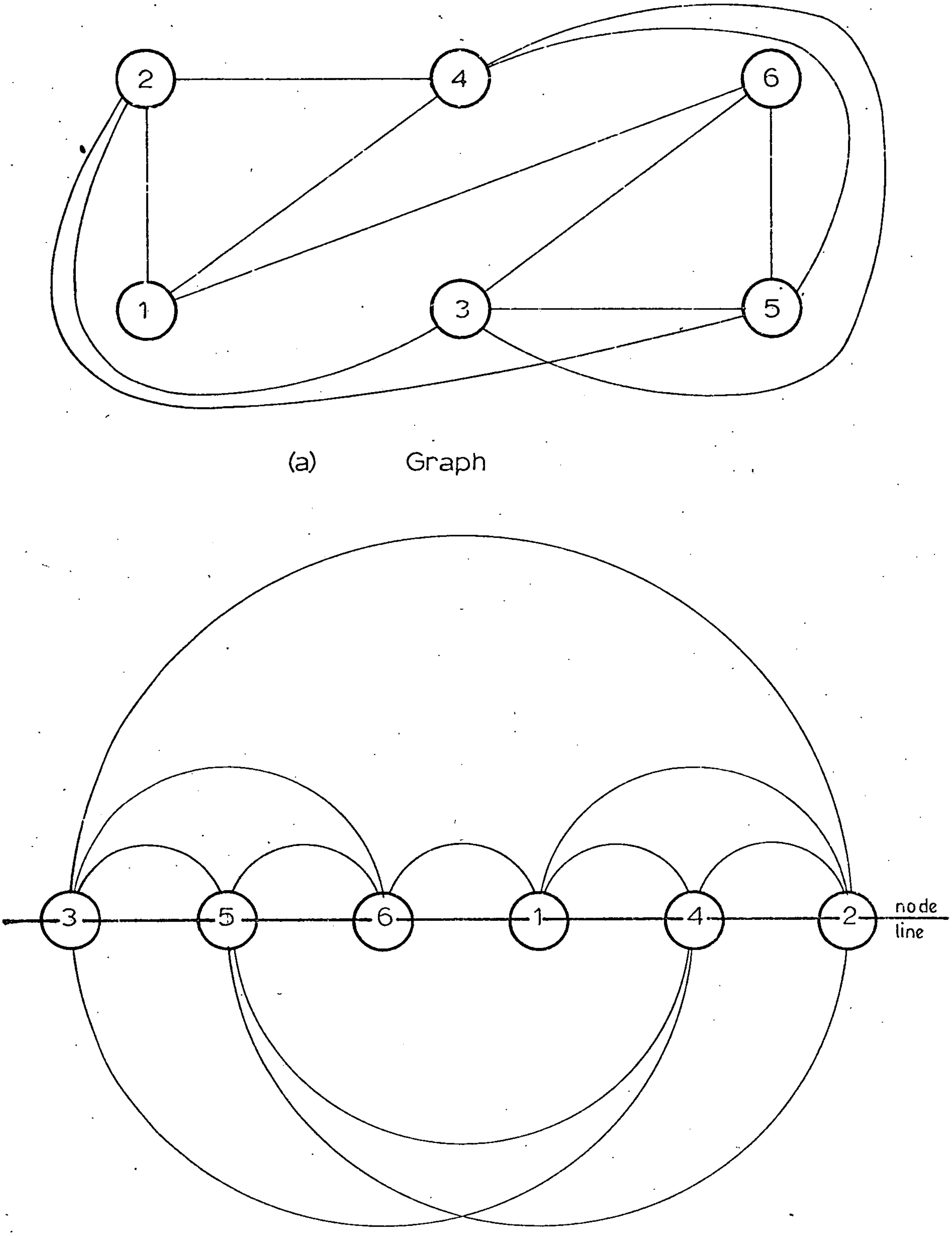
$A = (a_{ij}) \quad i = 1,N; \; j = 1,N$

$a_{ij} = 1$ iff Cij = 1 and q = 1 where q ≡(ni,nj)

$B = (b_{ij}) \quad i = 1,N; \; j = 1,N$

$b_{ij} = 1$ iff Cij = 1 and q = -1 where q ≡(ni,nj)

These partition the connection matrix into the connections above the node line (A) and the connections below the node line (B). The formula for the number of crossings can then



(a) Graph

(b) Permutation

Figure 3.3 Permutation Description of a Graph.

be obtained by inspection of figure 3.4 and the following observations. Links can only cross if they are on the same side of the node line and a link (i,j) can only cross link (k,1) with $p(i) < p(j)$ and $p(k) < p(1)$ and $p(i) < p(k)$ if $p(j) > p(k)$ and $p(j) < p(1)$; that is, if one link has one end point between the end points of the other link and its other end point outside the end points of the other link. The formula for the total number of crossings is then

$$F[PQ] = \sum_{i=1}^{N-3} \sum_{j=i+2}^{N} \left(a_{ij} \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{N} a_{kl} + b_{ij} \sum_{k=i+1}^{j-1} \sum_{l=j+1}^{N} b_{kl} \right)$$

The procedure for minimising the number of crossovers is carried out in two phases; first an initial permutation is constructed and then this is optimised by alteration of the node sequence and link re-routing.

3.3.1 INITIAL PERMUTATION CONSTRUCTION

The initial permutation is constructed by sequentially introducing nodes onto the node line, with the order of introduction being decided on a connectivity basis. The first node introduced is the one with the highest number of links associated with it; subsequent choices are made on the basis of the highest number of connections to those nodes already on the node line and nodes are placed in the position offering the minimum number of crossings. This procedure is continued until all the nodes are present on the node line. Nicholson has formalised this procedure as follows:

after t nodes have been introduced into the permutation let $p_t(i)$ denote the node in position i and $q_t(j)$ denote

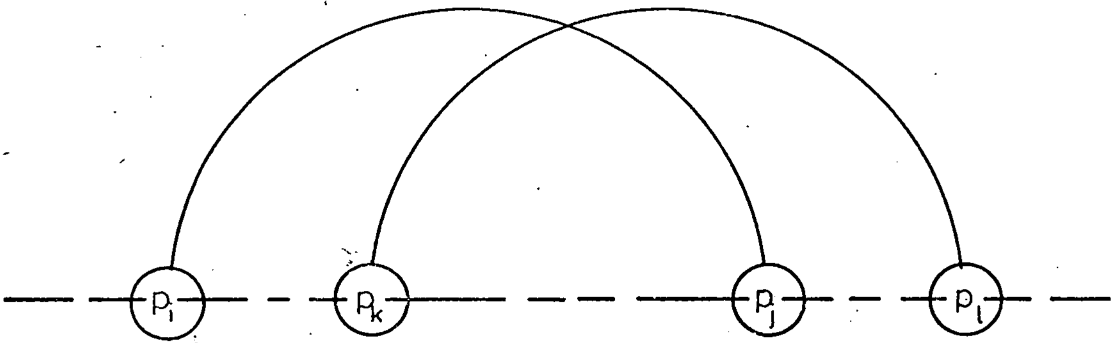


Figure 3.4 Crossing Configuration of the Permutation.

the route of the j^{th} link. This gives the partial permutation description $(P_t : Q_t)$. Initially $p_o(i) = \emptyset$ and $q_o(j) = 0$ $(i = 1,N;\ j = 1,M)$ to indicate the elements not yet included on the permutation; $f_t(i,j)$ denotes the total number of crossings if node i is placed in position j at the t^{th} iteration, i.e. introduction of the t^{th} node. The node i_t is selected at the t^{th} iteration and placed in position j_t determined by the following equations:

$$\sum_{k=1}^{t-1}\left(c_{i_t i_k} + c_{i_k i_t}\right) = \max_{\substack{1 \leqslant i \leqslant n \\ i = i_1 \cdots\cdots i_{t-1}}} \sum_{k=1}^{t-1}\left(c_{i\,i_k} + c_{i_k i}\right)$$

$$f_t(i_t,j) = \min_{1 \leqslant j \leqslant t-1} f_t(i_t,j)$$

After the node i_t has been introduced to the permutation, i_t and j_t have been determined. The inclusion is finalised by setting:

$$\begin{array}{ll} p_t(k) = p_{t-1}(k) & \text{for } k = 1, j_t - 1 \\ p_t(j_t) = i_t & \\ p_t(k) = p_{t-1}(k-1) & \text{for } k = j_t + 1,\ t \end{array}$$

3.3.2 OPTIMISATION

Optimisation of the permutation is not feasible by an exhaustive search as there are $(N-1)!/2$ different node-line permutations, each having 2^m different linkage configurations; in the simple graph in figure 3.3. this would be $(5!/2).2^{11} = 122,880$, which implies networks of 10 or more nodes and would take too long to optimise by exhaustive search. Thus, the optimisation procedure operates by altering the initial permutation to reduce the number of crossings. The alterations are made by repositioning a node and then optimising the routing to that node in its new position. This is called level-one optimisation, as only one node at a time is repositioned, and it continues until no further reduction of the number of crossings is possible and the permutation is said to be optimal at the first level. Higher levels of optimisation are achieved by examining cyclic alterations of 2 or more nodal positions.

A permutation P is altered to a related permutation P_{ij} obtained from P by placing node p_i in position j. This has associated with it $Q' = (q'_1 \ldots q'_m)$ which defines the optimal routing of the links such that F(Pij:Q') cannot be reduced by altering any of the qi of any of the links connected to node p_i. A permutation (P,Q) is said to be optimal at the first level if, for all i and j

$$F\ (P,Q) \leqslant F\ (Pij:Q')$$

The nodes are selected for trial repositioning in decreasing order of the number of associated crossings. Furthermore, a node is only chosen once between permutation alterations to avoid duplication, by repeating the same procedure with the same configuration. The procedure terminates when all nodes have been tried unsuccessfully since the last alteration: that is, a local minimum has been achieved with respect to the number of crossings.

Nicholson formalises this procedure as follows: Let $(P_t:Q_t)$ be the permutation obtained after t iterations of the optimising procedure, and $\ddot{U}_t$ denote the set of nodes which have been selected for trial movement since the last alteration. At iteration t node i_t is moved to position j_t as determined by the following equations:

$$\sum_{k=1}^{n}\left(c_{p(i_t)p(k)} + c_{p(k)p(i_t)}\right) = \max_{\substack{1\le i\le n \\ i\notin U_t}} \sum_{k=1}^{n}\left(c_{p(i)\,p(k)} + c_{p(k)\,p(i)}\right)$$

$$F(P_{t-1}\, i_t\, j_t : Q') = \min_{1\le j\le n} F(P_{t-1}\, i_t\, j : Q')$$

A successful alteration is accomplished by the following operations:

$(P_t : Q_t) = (P_{t-1}\ i_t\ j_t : Q^1)$

and

$U_t = \emptyset$

If the t^{th} iteration does not produce an alteration, then

$(P_t : Q_t) = (P_{t-1} : Q_{t-1})$

$U_t = U_{t-1} + i_t$

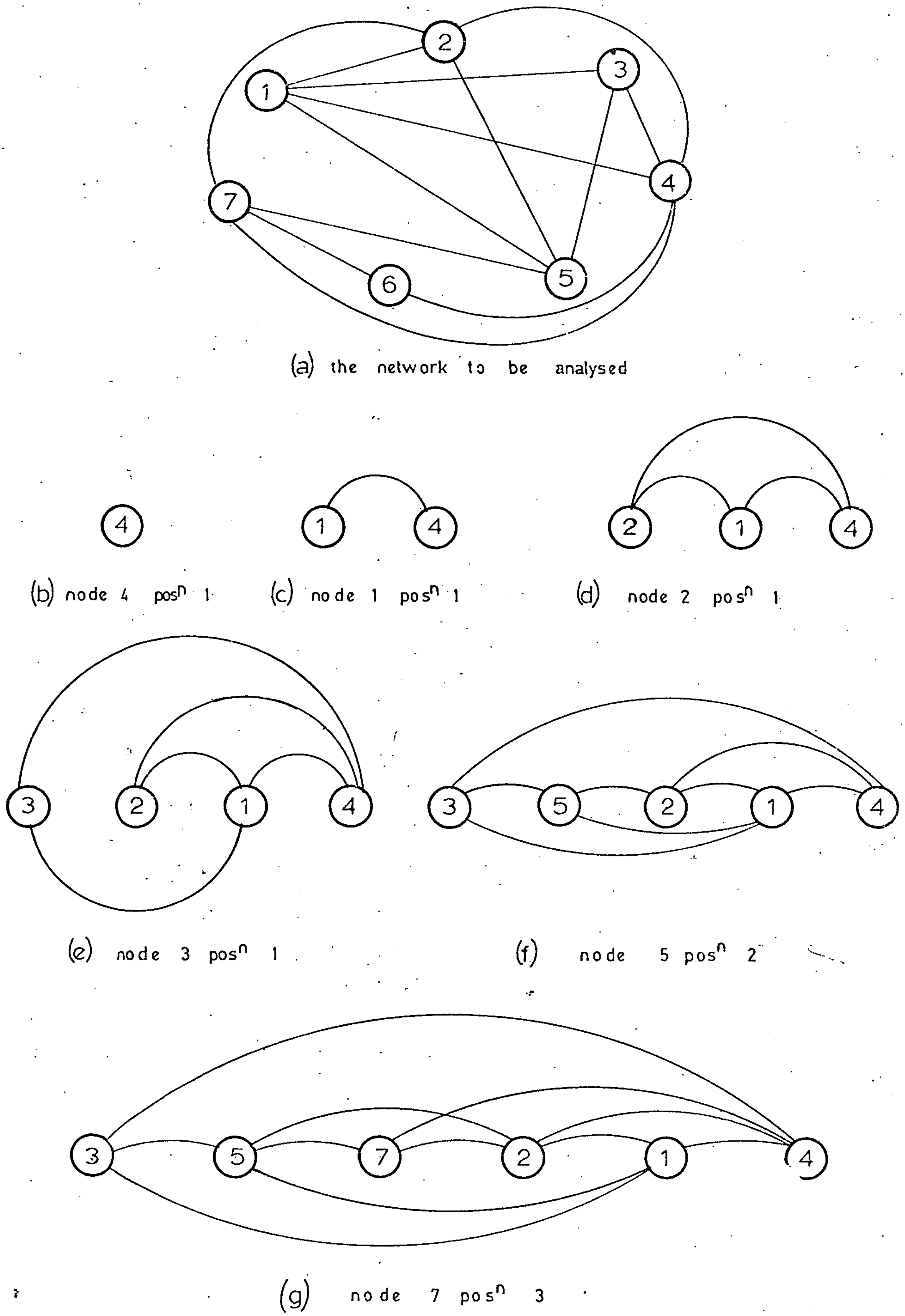
When $i \in U_t$ Vi = 1,N, $(P_t : Q_t)$ is the optimal permutation at the first level.

3.4 WORKED EXAMPLE

A worked example is shown in figures 3.5(b) to 3.5(i). Figure 3.5(a) shows the network with crossings, while a planar drawing of the same network is shown in figure 3.5(j) as obtained from the algorithm. The network contains 7 nodes and 13 links. The links are as follows:

(1,2) (1,3) (1,4) (1,5) (2,4) (2,5) (2,7) (3,4)
(3,5) (4,6) (4,7) (5,7) (6;7)

The most connected node is 4, which is first inserted into position 1. Next, node 1 is selected from 1,2,3,6,7 as all are connected to 4 and inserted in position 1. 4 is pushed into position 2 (figure 3.5(c)). Nodes 2 and 3 are



(a) the network to be analysed

(b) node 4 pos^n 1

(c) node 1 pos^n 1

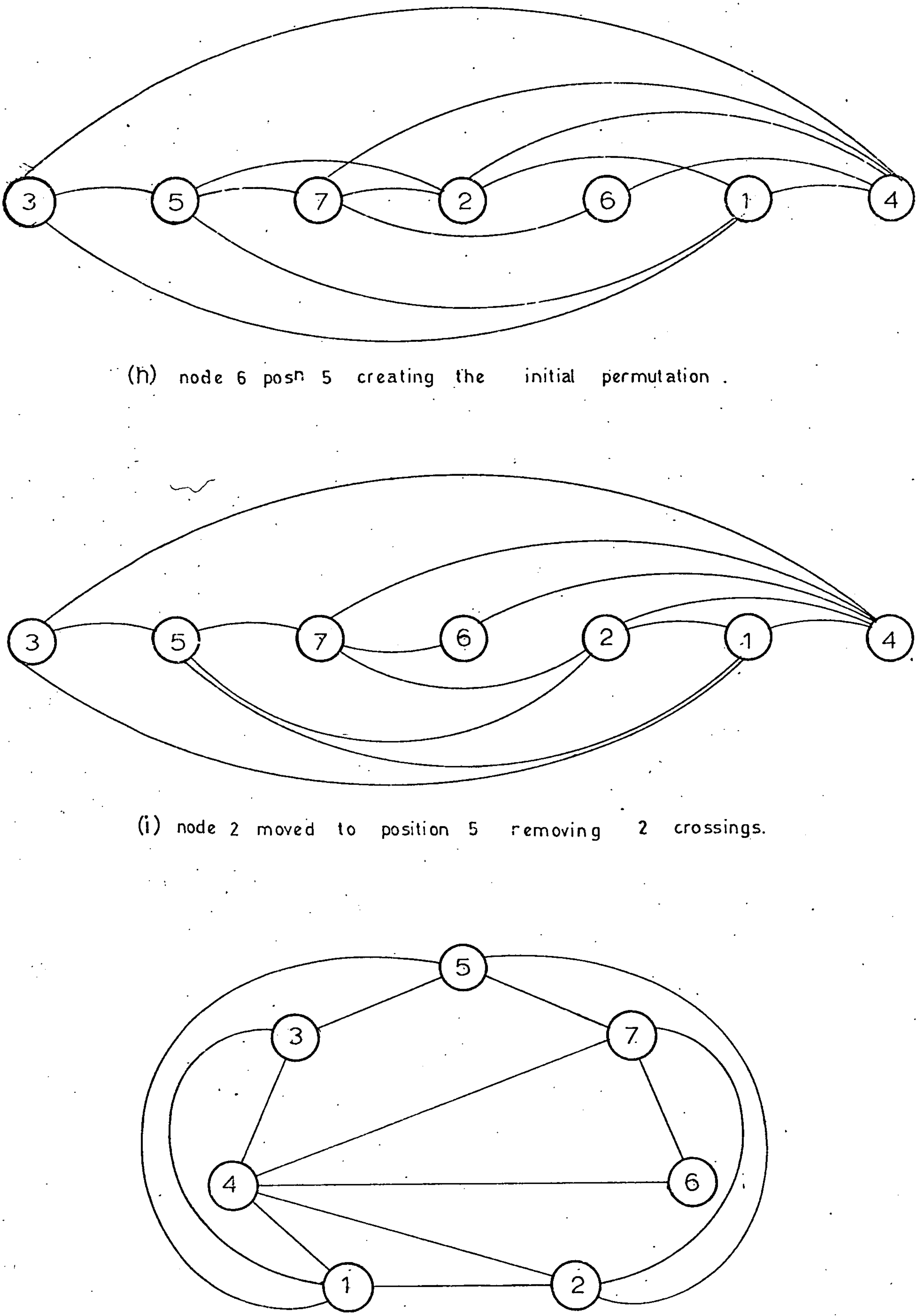
(d) node 2 pos^n 1

(e) node 3 pos^n 1

(f) node 5 pos^n 2

(g) node 7 pos^n 3

Figure 3.5 Nicholson's Algorithm — a worked example



(h) node 6 posn 5 creating the initial permutation .

(i) node 2 moved to position 5 removing 2 crossings.

(j) resulting planar drawing.

Figure 3.5 Nicholson's Algorithm— a worked example.

the next insertions both at position 1 (figures 3.5(a) and (e)). Node 5 is now inserted in position 2 as it results in the least crossovers. Similarly, nodes 7 and 6 are inserted in positions 3 and 5 respectively, each insertion generating one crossover. This leaves the initial permutation with two crossings:

(2,5) x (4,7)

(1,2) x (4,6)

Nodes 2 and 4 each have two associated crossings and are therefore the first nodes to be examined for trial repositioning. Node 2 is shifted to positions 1,2,3,4 and 5 generating 2,2,1,2 and Ø crossings respectively. It is therefore positioned between nodes 6 and 1 (figure 3.5(i)) and removes all the crossings from the network. There is no need for further optimisation as the permutation is planar and can be redrawn as in figure 3.5(j).

3.5 DISCUSSION

The algorithm does not guarantee to find a global minimum with respect to the number of crossovers. This is verified by the results Nicholson obtained on a symmetric utility graph of 8 nodes, for which the minimum number of crossings is known to be 4, whereas the algorithm produced a first level optimum of 5. It will derive a local minimum, which may coincide with the global minimum as was found to be the case in analysing complete graphs, but it would be necessary to alter the optimisation procedure to improve the probability of achieving a global minimum.

The obvious improvement would be to raise the level of optimality by examining interchanges of pairs of nodes.

This would substantially increase the time requirements of the algorithm as the optimisation procedure would require two node positional changes for each trial. The other alternative would be to include multi-semicircular link representation which would, as we have seen, complicate the procedure considerably as it requires much more stored data to describe the linkage structure.

The results obtained from complete and symmetric utility graphs indicate that the local minimum produced by a first level optimisation frequently coincides with the global minimum. As the complete removal of crossovers is desirable, but not fundamental to the program's aims, it was decided that the first level optimisation produced results of a sufficient standard.

The deformation of a general network into a permutation description can be reversed just as easily, so that a node-line may be drawn as a simple curve to avoid crossings on the node line, with the same linkage structure (see appendix III for proof). This is of great value when the nodes require placement during the subsequent layout phase and will be discussed in section 4.2.

Another advantage of Nicholson's algorithm is the ease with which the relationships between the nodes and links can be determined. The number of links and their directions, associated with each node, together with the numbers passing above and below, can easily be calculated from the permutation description. This is of great use (see section 3.1) in the layout phase.

Perhaps the greatest advantage, however, is associated with the permutation representation; this is based on an easily conceptualised layout of the network, which once analysed can be drawn readily. The ability to draw the minimum crossing configuration clearly and easily is necessary as an algorithm which gives a minimum of cross-overs without any details on how to position and connect to achieve that minimum is of little use in layout problems.

3.6.0 APPLICABILITY TO ELECTRICAL CIRCUITS

The first consideration in using any graph-theoretic analysis algorithm, such as Nicholson's permutation procedure, in conjunction with electrical circuits is the method of converting the electrical network into a graph network. The graph consists of point nodes and links, each of which connects exactly two nodes and thus components and connections must be mapped into nodes and links by a transformation which is essentially simple and reversible. Reversibility is necessary to facilitate the mapping of the output from the graph analysis algorithm onto the electrical circuit.

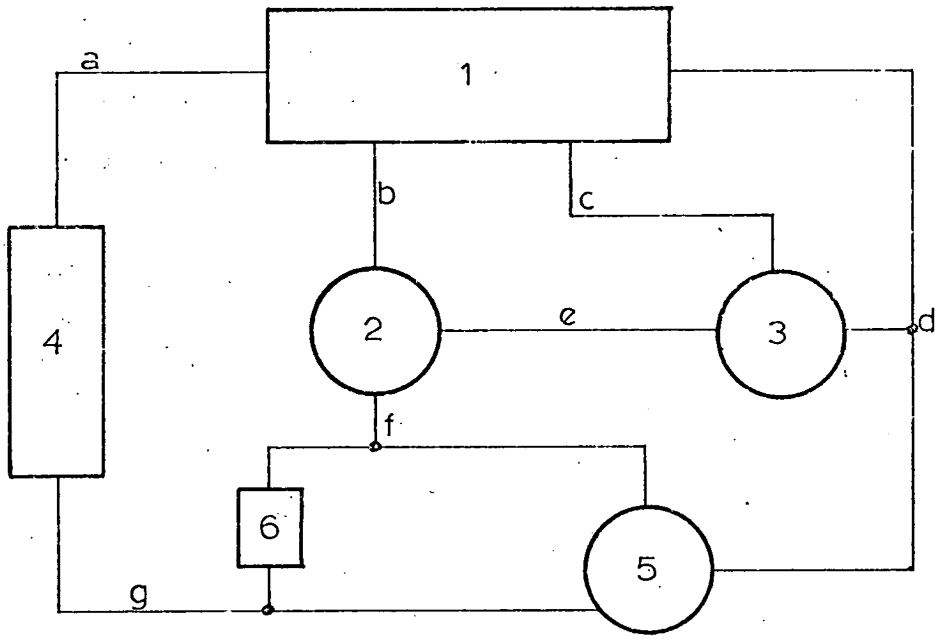
A simple mapping of components onto nodes and connections onto links encounters two major problems; first, if a component has three or more connections to it, these will appear around the perimeter of the component in a specific order, the terminal pin order, and if this component is mapped onto a node the terminal pin order is lost since the node, as considered in graph theory, has no perimeter. Secondly, the electrical connections are often in the form of nets, that is, groups of electrically common pins connected by a single conductor track and they cannot be mapped directly on to a link if there are more than two components associated with the net.

Goldstein and Schweikert (1973) have published a paper on graph modelling of electrical circuits which has been briefly discussed in section 2.5. A discussion of methods of circuit representation by graphs on which the representation used for Nicholson's algorithm is founded, is given below.

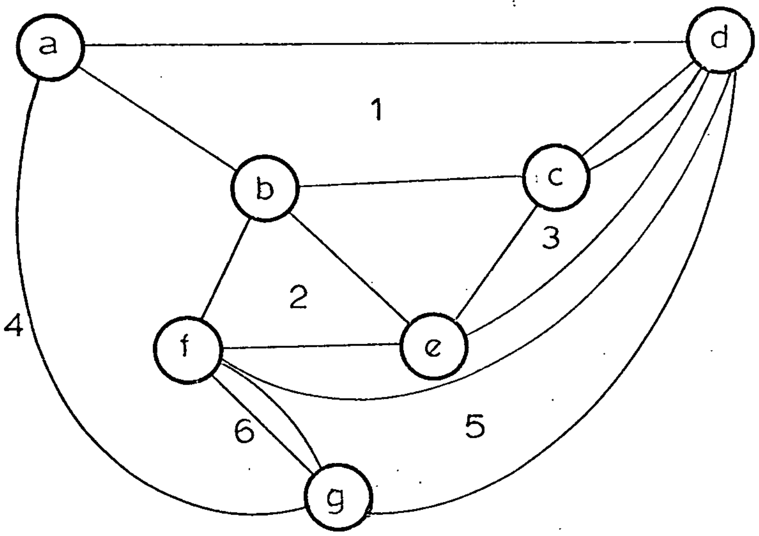
3.6.1 COMPONENT TO LINK MAPPING

The first mapping considered is a transformation of nets onto nodes and components onto links. Each set of electrically common pins is incorporated as a separate node, components are then included as a cycle containing those nodes representing nets incident on one of the component's terminal pins; if the component has only two terminals it is represented as a single link; figure 3.6(b) is such a representation of the circuit shown in figure 3.6(a). It can be seen that a multi-terminal component corresponds to a region of the graph.

This mapping is difficult to grasp, although it maintains terminal pin ordering and it is an accurate mapping of the nets. It is unsuited for use with Nicholson's algorithm as the links, corresponding to one component, would have to be constrained to define an elemental area with no other links inside it to avoid overlap of the components. This is illustrated by adding a two terminal component to figure 3.6(a) between nets b and d, which is represented in figure 3.6(b) as a link between nodes b and d. However,



(a) circuit



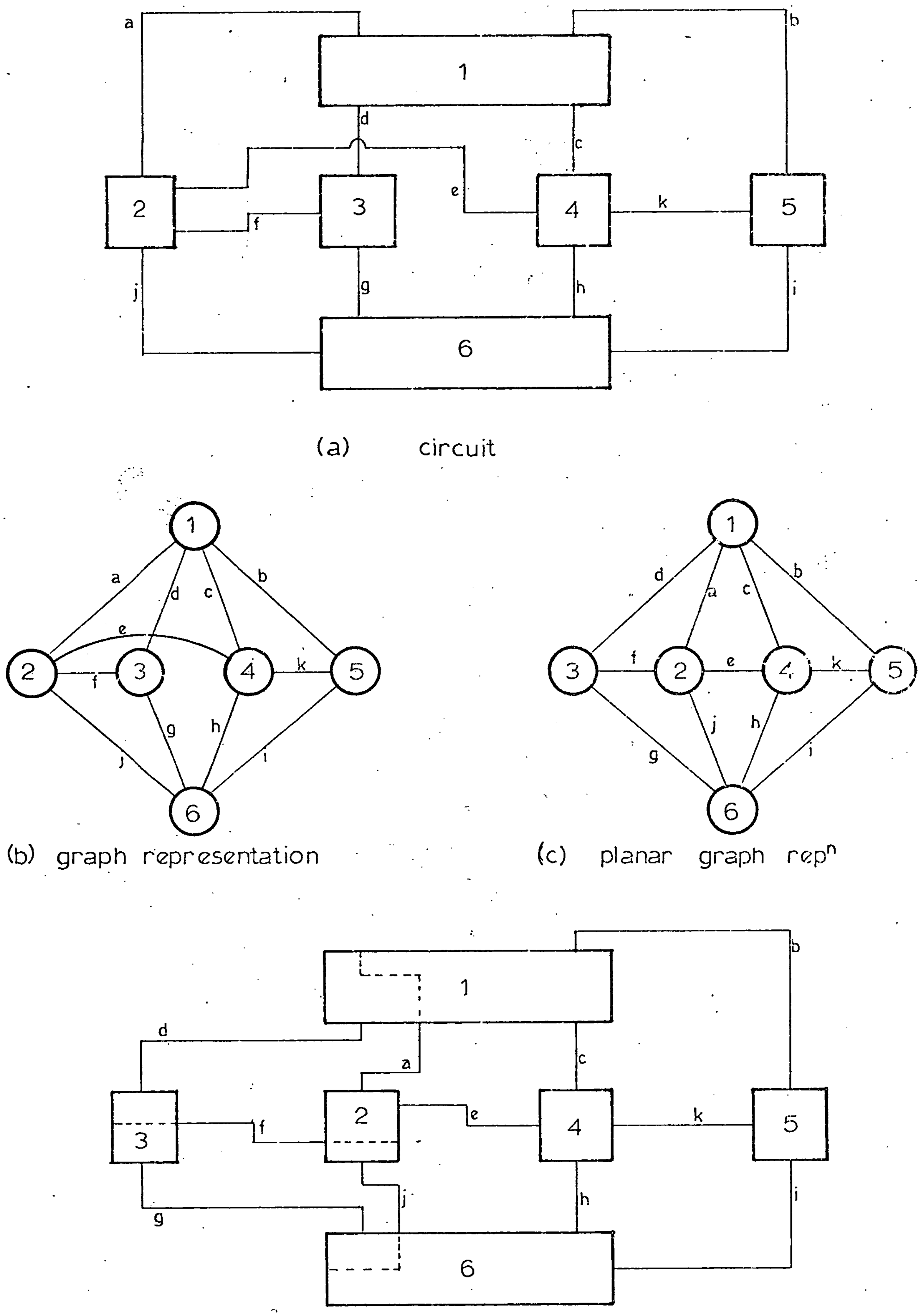
(b) graph representation.

Figure 3.6 Component to Link Mapping.

this would correspond to an overlapping of the component with 1, which is obviously a non-planar layout although the graph representation is planar, but the constraint required to avoid this inaccuracy would decrease the speed and efficiency of the planarizing algorithm.

3.6.2 COMPONENT TO NODE MAPPING

The other basic mapping, components to nodes and nets to links, also displays the disadvantages mentioned in section 3.6.0. The terminal pin ordering is significant as is revealed by the circuit of figure 3.7(a), represented this way in figure 3.7(b) as being non-planar. Exchanging the positions of nodes 2 and 3 produces a planar representation as in figure 3.7(c). In laying out this planar representation we find that it is necessary to route net a under component 1, net f under 2,and 3, and net j under 6, as shown in figure 3.7(d). This situation is called an induced or module crossing as it is only apparent at the layout realisation stage. Alternative component representations were examined in an attempt to elude this difficulty. The most common of these is subgraph representation, in which each component is described as a subgraph with the same number of nodes as the component has terminal pins. Links between the subgraph nodes are drawn so as to preserve both the terminal ordering and the basic component structure, thus removing the need for under-component wiring to make the connections when the layout is realised. Some examples of components and subgraph representations are illustrated in figure 3.8.



(a) circuit

(b) graph representation

(c) planar graph rep^n

(d) circuit realization of planar graph.
(induced crossings shown dashed)

Figure 3.7 "Planarization" via Induced Crossings.

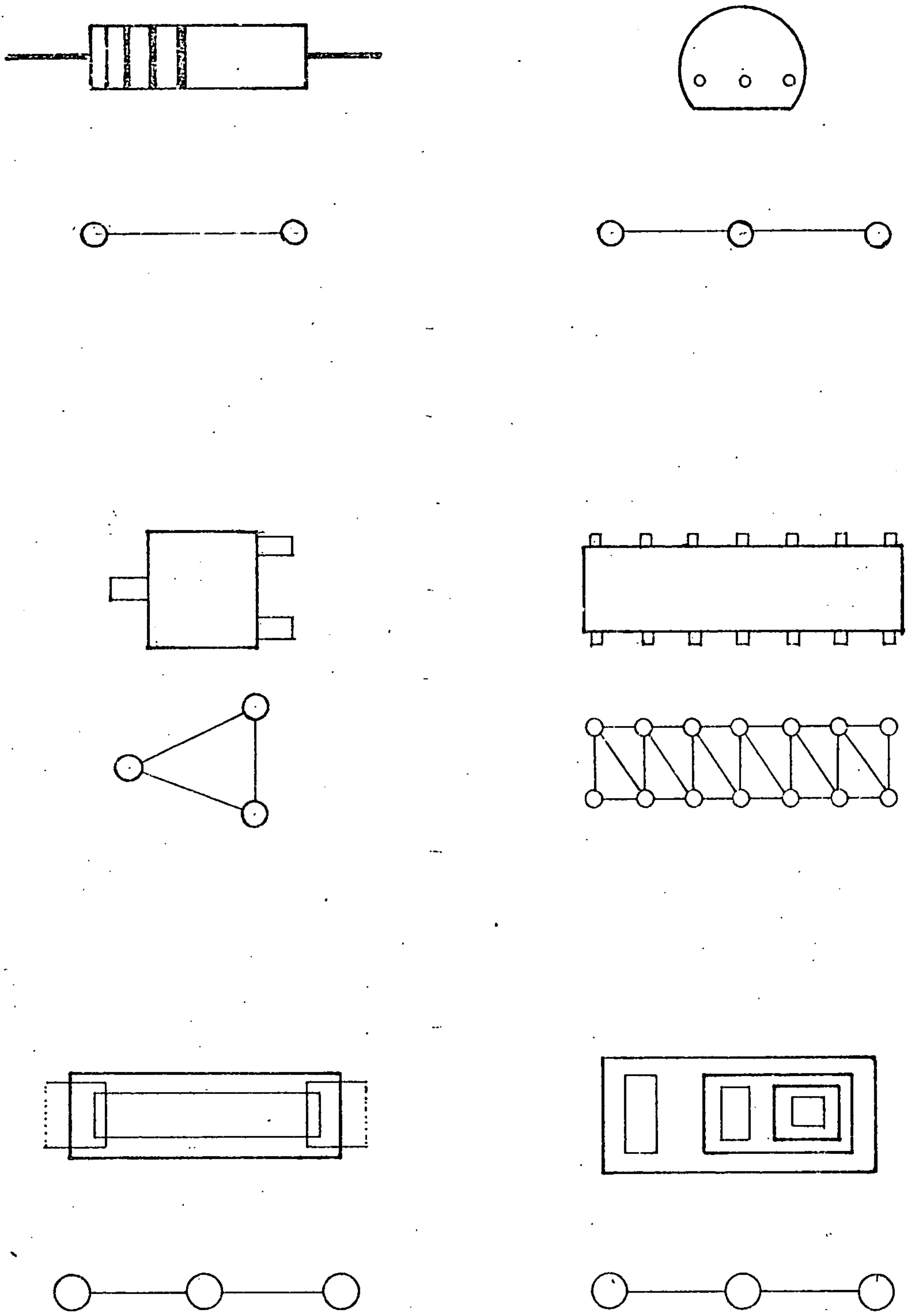


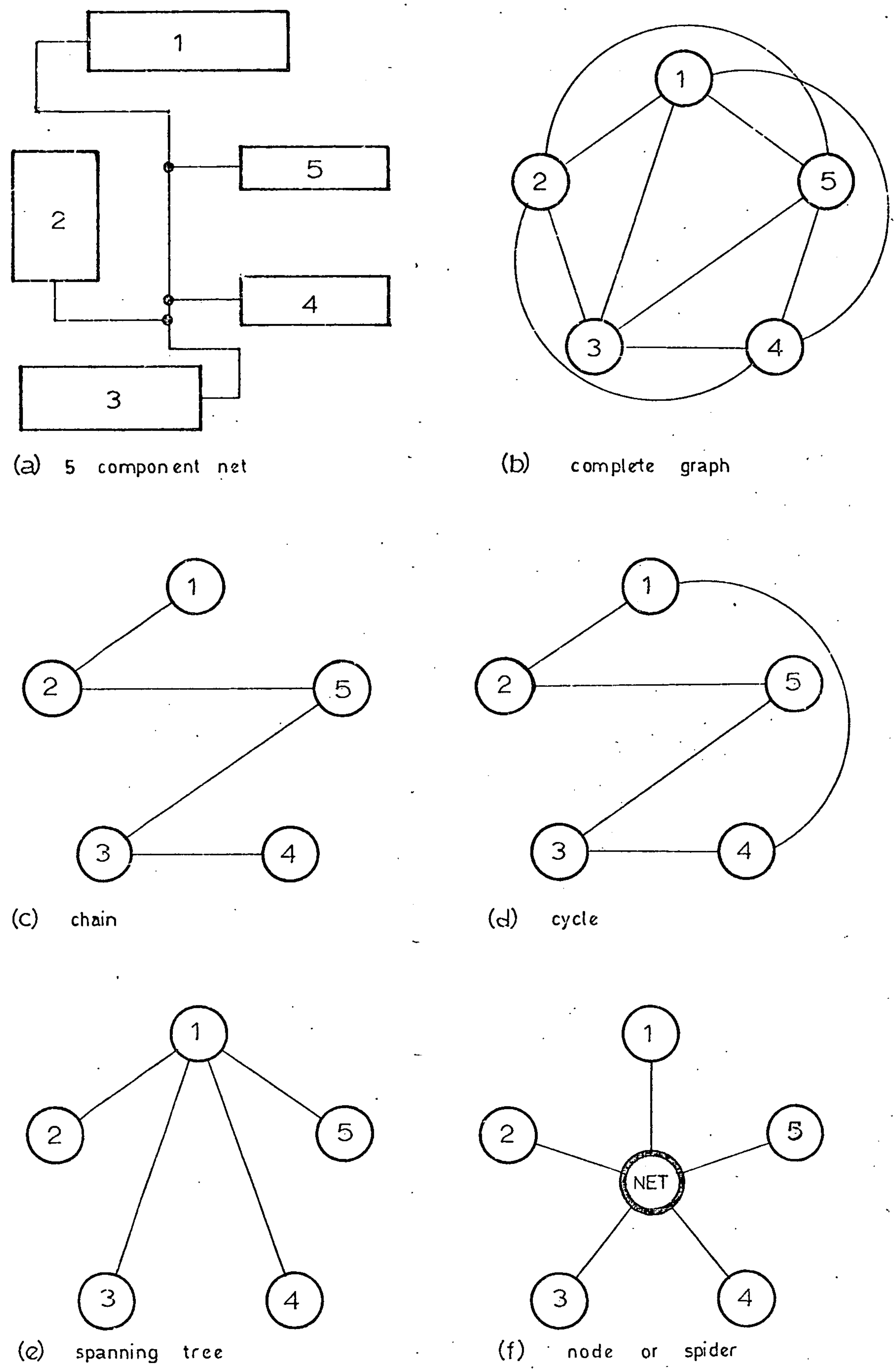
Figure 3.8 Some examples of Subgraph Component representations

In order to use subgraph representation in Nicholson's algorithm complicating constraints similar to the component to link mapping, except applied to the nodes, would be necessary. It would also mean a circuit of 10 components could have around 50 nodes, which is discouraging.

3.6.3 NET REPRESENTATION

A net of more than two electrically common points must be represented as a set of links; there are various ways in which this may be accomplished. The first method is by means of a complete graph on the nodes of the net, as in figure 3.9(b). This corresponds to connecting each node on which the net is incident to every other net member with a separate link. This becomes unwieldy if there are more than three nodes in the net and, if a circuit contains several multi-node nets, the link concentration increases out of proportion and the algorithm's processing time suffers. Therefore, this type of net representation is considered computationally inefficient.

A second method is to use a chain or a cycle containing all the net members as in figure 3.9(c) and (d). The sequence of the nodes in the chain or cycle can give rise to anomalies. For example, the net (1,2,3,4) of figure 3.10(a) is represented by the cycle (1,2,3,4,1) in figure 3.10(b), which is planar and by (1,3,4,2,1) in figure 3.10(c), which is non-planar, although the links which cross are members of the same net. It is impossible to say at the outset which chain is the best in terms of net representation. The only certain method is to try them all, which is exhaustive and requires (n-1)!/2 runs for each n-node net. Consequently, this technique is considered



(a) 5 component net

(b) complete graph

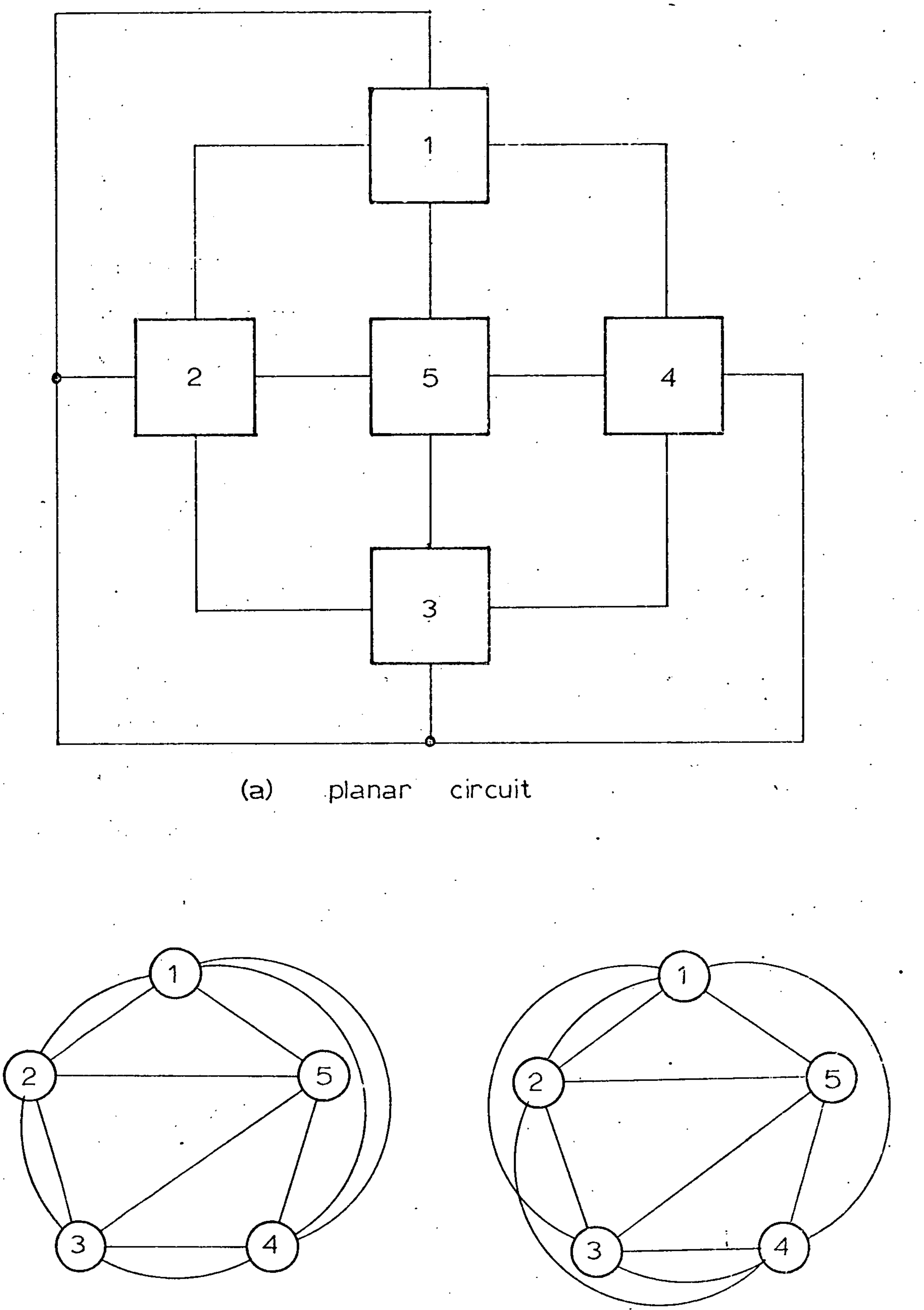
(c) chain

(d) cycle

(e) spanning tree

(f) node or spider

Figure 3.9 Net representations..



(a) planar circuit

(b) planar rep^n: net = {1,2,3,4,1}

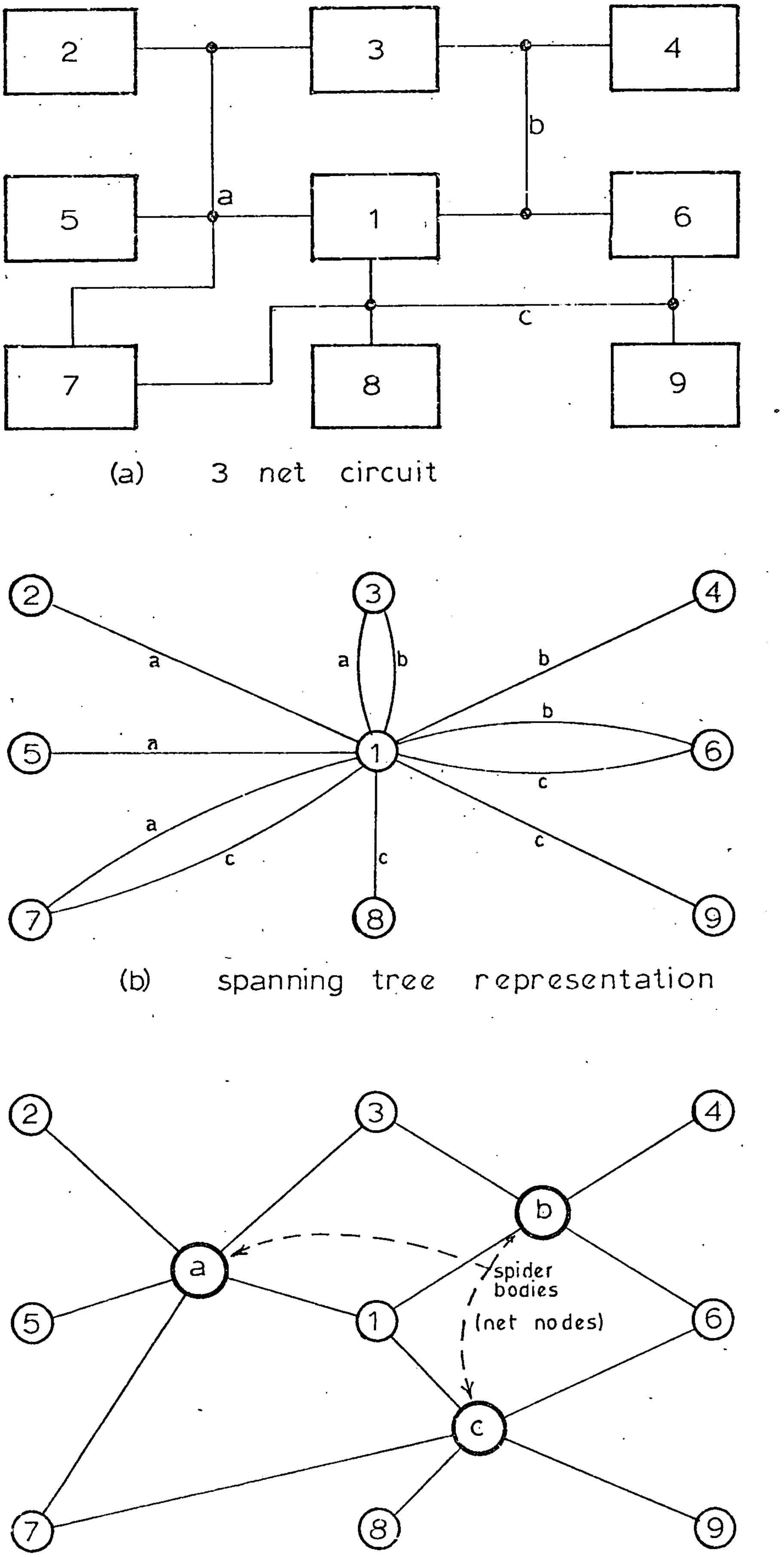
(c) non planar rep^n: {net = {1,3,4,2,1}

Figure 3.10 An Anomaly arising from Cyclic Net representation.

unsuitable as different results arise from different net decompositions.

The third technique of net representation is to connect one node to all the other net members (figure 3.9(e)); this forms a spanning tree of the nodes belonging to the net. Alternatively, an additional node may be introduced to represent the net (figure 3.9(f)) which is equivalent to describing the net as a spider, with the net members at the ends of the spider's legs. These two methods are very similar as the former representation is obtained by merging the spider's 'body' with one of the connected nodes.

The spanning tree and the spider are the most conceptually accurate of the different methods and do not suffer the drawbacks associated with the others. The spanning tree is the most economic representation in terms of numbers of nodes and links. As can be seen in figure 3.8, however, the central node does have many more links incident upon it than corresponding connections in the circuit and the situation in figure 3.11(b) is undesirable where node 1, with only 3 electrical connections, has 11 attached links because it was chosen for the trunk node for the three nets to which it belongs. This can be avoided by using the spider representation which maintains a strict 1 to 1 relationship between component connections and node/link incidence (figure 3.11(c)). The increase in the number of graph components is a trade-off with isomorphism and it is difficult to state which is better in this case.



(a) 3 net circuit

(b) spanning tree representation

(c) spider representation

Figure 3.11 Tree and Spider Net Mappings.

The representation used here is components to nodes and nets to spanning trees for two and three pin nets, and spiders for nets of four or more pins. A more detailed study of the results produced by the various net representations is undertaken in section 3.10.

3.7.0 IMPLEMENTATION

Nicholson uses a connection matrix to store the network link structure, which is partitioned into an upper and lower connection matrix to evaluate crossovers. Each of the matrices is N by N and, as we have seen in section 1.2.1, space must be reserved for the maximum case. Consequently, the storage requirements of the link structure alone is $3N^2$ words where N represents the maximum number of nodes the program will be required to process. For a network maximum of 50 nodes this would be equivalent to 7.5K, which would severely limit the space available for the program on timesharing systems.

The maximum number of links for a 50 node network in the case of a complete graph is 1225; however, a more realistic figure would be 250 (obtained by using a connectivity index of 1Ø). In this case, 30 words are reserved per link on average, a rather extravagant ratio! Finally, the connection matrix of a non-directed graph, and all electrical circuit representations are non-directed, is symmetric and so half the contents are redundant and could be excluded if an array-type format was avoided.

Computational speed favours array-type storage as it permits rapid accessing of elements without a lot of computation in comparison, for example,with a data structure where pointers have to be traced, words accessed and unpacked and lists examined. Again we are reminded of the computational trade-off between speed and size (see section 1.2.3).

As the objective of this program was its use on a time-sharing system which can have a 10k limit on program size, the advantage lies with compact storage. With such a low limit it is obviously much better to have a slow program which will fit on the system, rather than a very fast program which requires more storage than allowed. This consideration led to the choice of a ring data-structure (figure 3.12).

There is one bead per node of the network, which contains information on the links incident on that node. Each link is described by the node which it connects so that there is one node bead word per link incident on the node. There are two 'housekeeping' words per bead, the head word and a pointer; the head word contains the node number and the number of links attached to the node in packed form and the pointer is used to string the bead on to the ring. The head bead of the structure contains the ring head pointer, the number of nodes and links in the network and a word used for network identification. In addition, there are two words prior to the head word, one is used to point to the free space and the other is a pointer for garbage collection. The free space indicator is necessary when the data-structure is being created as it shows where space is available for placing the next bead to be added. Garbage collection is of doubtful use

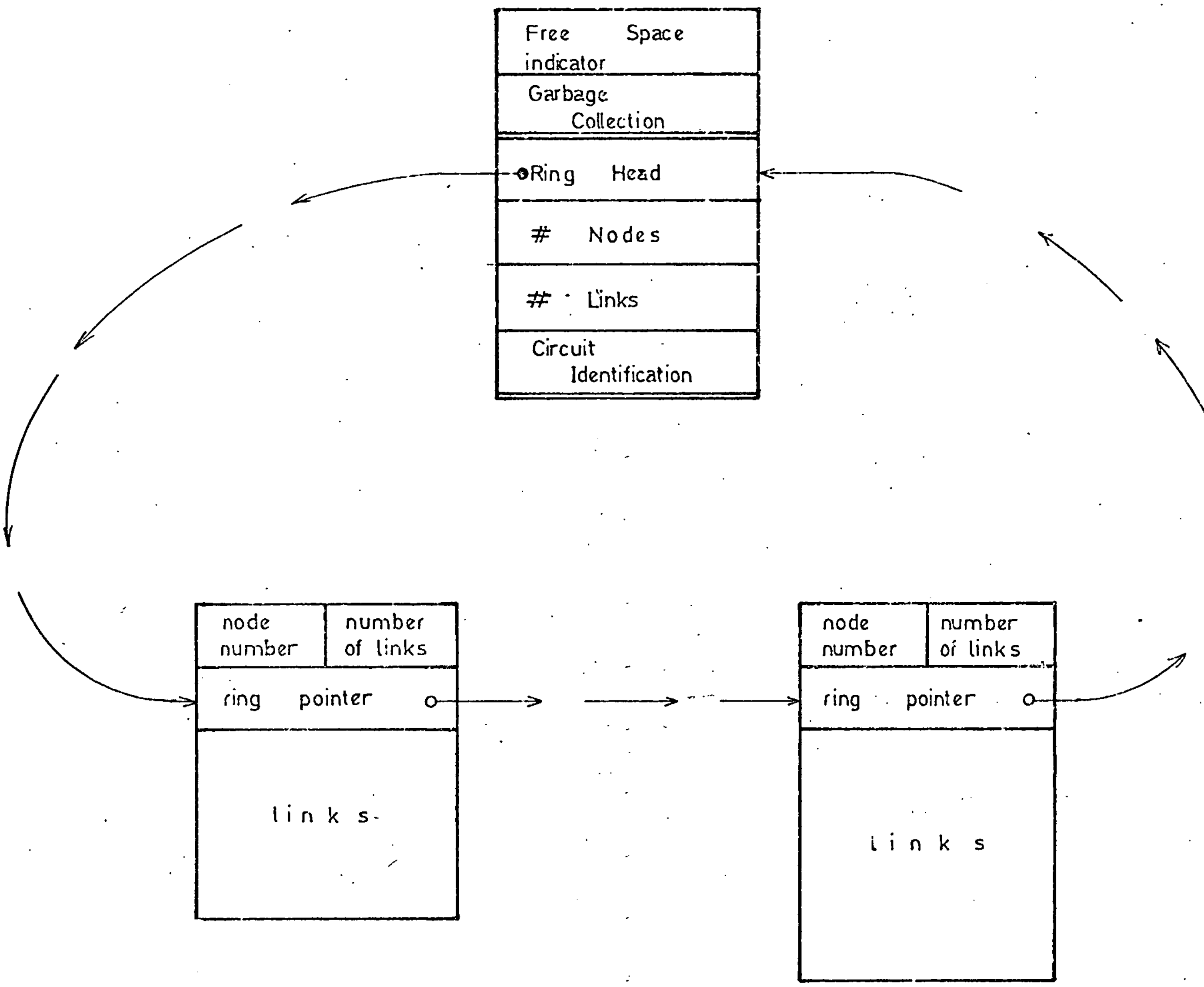


Figure 3.12 Data Structure for Permutation Procedure

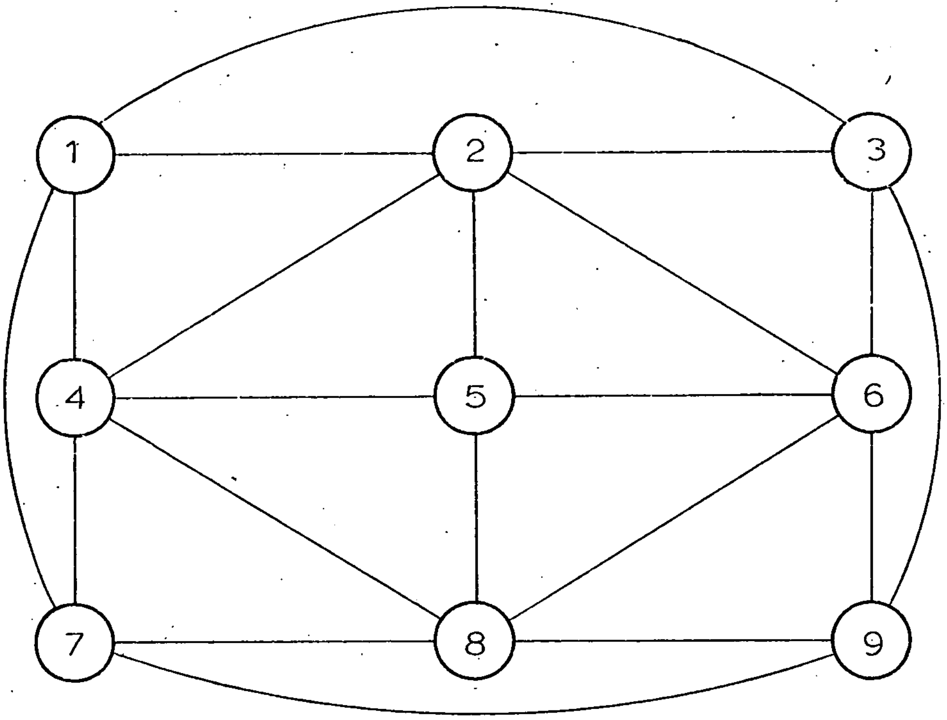
as the dynamic addition and deletion of beads is not a necessary facility for Nicholson's algorithm because the beads, once present, remain. However, it was thought it might be of use in the layout phase of the program and so was included initially to avoid the possibility of rearranging the structure, if garbage collection was found to be advantageous.

The circuit description requires, for a network of N nodes and M links 6 + 2N + 2M words, equal to 606 for N=50 and M=250, which is a 75% reduction on the connection matrix and a 92% reduction on the three connection matrices that Nicholson uses.

3.7.1 GRAPH INPUT

The circuit graph input is the first step in the program. FORTRAN does not efficiently process character input and so the description is confined to numerical lists. The format used is one input line per node; the node number is the first integer, followed by the list of nodes to which it is connected by links. The final line contains a negative node number to signify that the input is complete, for example see figure 3.13. As the network description is input, the data structure is built up; one new bead is created for each line of input and is strung on to the ring.

Once the input is complete, certain validity checks are made to ensure no elementary mistakes have been made with the input data. The nodes must be numbered 1 to N inclusive and each node must be described, i.e. appear at the head of an input line, exactly once. The links are then scanned, to see that each is corroborated by a link



(a) network

```
1   2   3   4   7
2   1   3   4   5   6
3   1   2   6   9
4   1   2   5   7   8
5   2   4   6   8
6   2   3   5   8   9
7   1   4   8   9
8   4   5   6   7   9
9   3   6   7   8
-1
```

(b) input description

Figure 3.13 Format for Coding Networks

in the'opposite direction'. If node i contains a link entry for node j, then clearly node j must contain a corresponding link entry for node i, otherwise a mistake is present. Although it would be easier to code the network by the list of links described by node pairs, this would be difficult to examine for coding errors. The above description means that mistakes will not escape detection unless the same mistake is made twice in different places.

3.7.2 INITIAL PERMUTATION CREATION

When the validity of the network has been established the initial permutation must be created. Since the connection matrix is not used, the mathematical equations used by Nicholson are inapplicable. A heuristic procedure, which is described below, was developed to replace the matrix based formulations. It was found that this procedure involved a large amount of bead examination based on the order of the nodes in the permutation. To speed the process of locating the bead following another or before another in the permutation sequence, two ordered rings were added to the basic structure. These rings, forward and reverse, pass through the node beads in the same order as the nodes appear on the node-line (left to right order is conventionally called forwards). The amended data structure is illustrated in figure 3.14. It should be noted that there is a storage increase of (2N + 2) words for the additional rings but this is far outweighed by the resulting increase in computational efficiency.

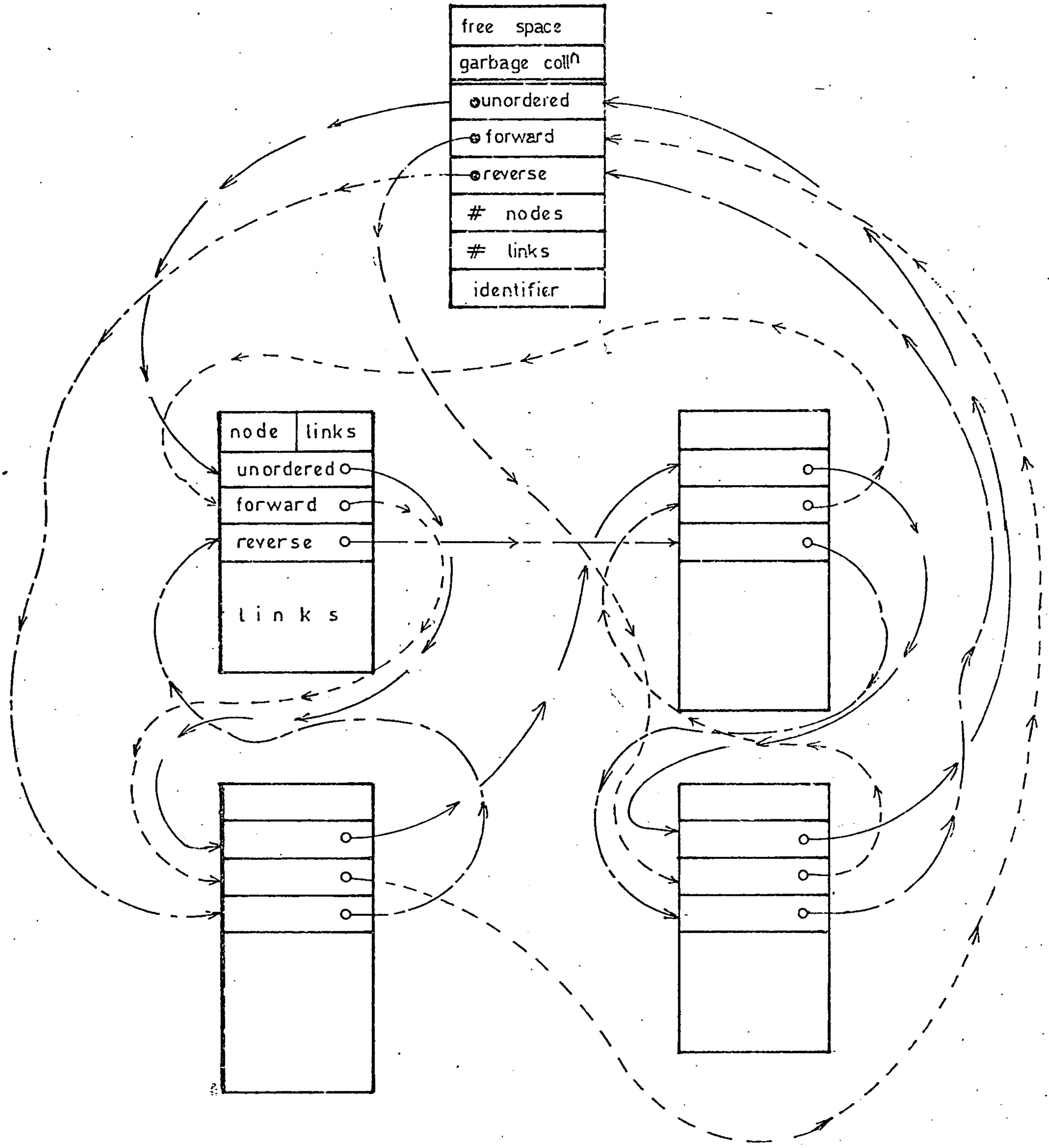


Figure 3.14 Ammended Data Structure

Beads are only added to these rings when their node is introduced into the permutation, so that the bead ordered ring pointers are initially null. The permutation is described by means of a one dimensional array, the n^{th} entry of which contains the current position of node n, corresponding to the inverse of the P used by Nicholson. The node beads are added to the ordered rings as soon as a node is introduced into the permutation. They are then deleted and re-added in the appropriate position for each trial location of the node in the permutation. This constant alteration of pointers is necessary to allow the crossover calculation routine to operate via the data structure.

The initial permutation creation commences by introducing the node with the highest number of links which is obtained by a simple search around the node ring. Subsequently, nodes are selected for introduction to the permutation on the basis of the highest number of connections to already present nodes. The selected node is placed at the end (right hand) of the permutation and incrementally moved forward, whilst the permutation array P and the ordered ring pointers are updated at each step. In each position, the minimum number of crossovers associated with optimal routing of the nodes' links is calculated; if it is zero, the node is fixed in that position, its bead added to the ordered rings in the corresponding location and the routing finalised. If there is a positive number of associated crossovers at that position, the incremental movement is repeated. When a node reaches the start (left hand end) of the permutation without a planar location being found, it is returned to the location with the minimum number of associated crossovers and is added to the ordered rings there. The links are routed optimally, i.e. to minimise the crossovers for that position. When all the nodes are present a check is made to see that all the links have been routed from both ends.

The crossovers are calculated with a simple heuristic procedure, which operates via the ordered rings and calculates the number of crossings for each link routed above or below the node-line. The routine is supplied with the node numbers connected by the link. It locates the bead following the first node of the pair and scans it for links passing beyond either of the end points of the link. If any such links are found the number of potential crossovers above or below is incremented, depending on the routing of the link found. The routine follows the forward ordered ring round, repeating the examination for each bead found, until it reaches the second node of the pair, when it stops. The validity of this procedure is best illustrated in figure 3.15.

The node-line is shown containing nodes a to f in their correct positions and omitting intermediate nodes, the link from b to e (1 or 2) is being examined for potential crossovers. There are four configurations of other links, stated in terms of the relative positions of their end nodes:

i both are between b and e (3,4)

ii one is between, the other is coincident with b or e (5,6,7)

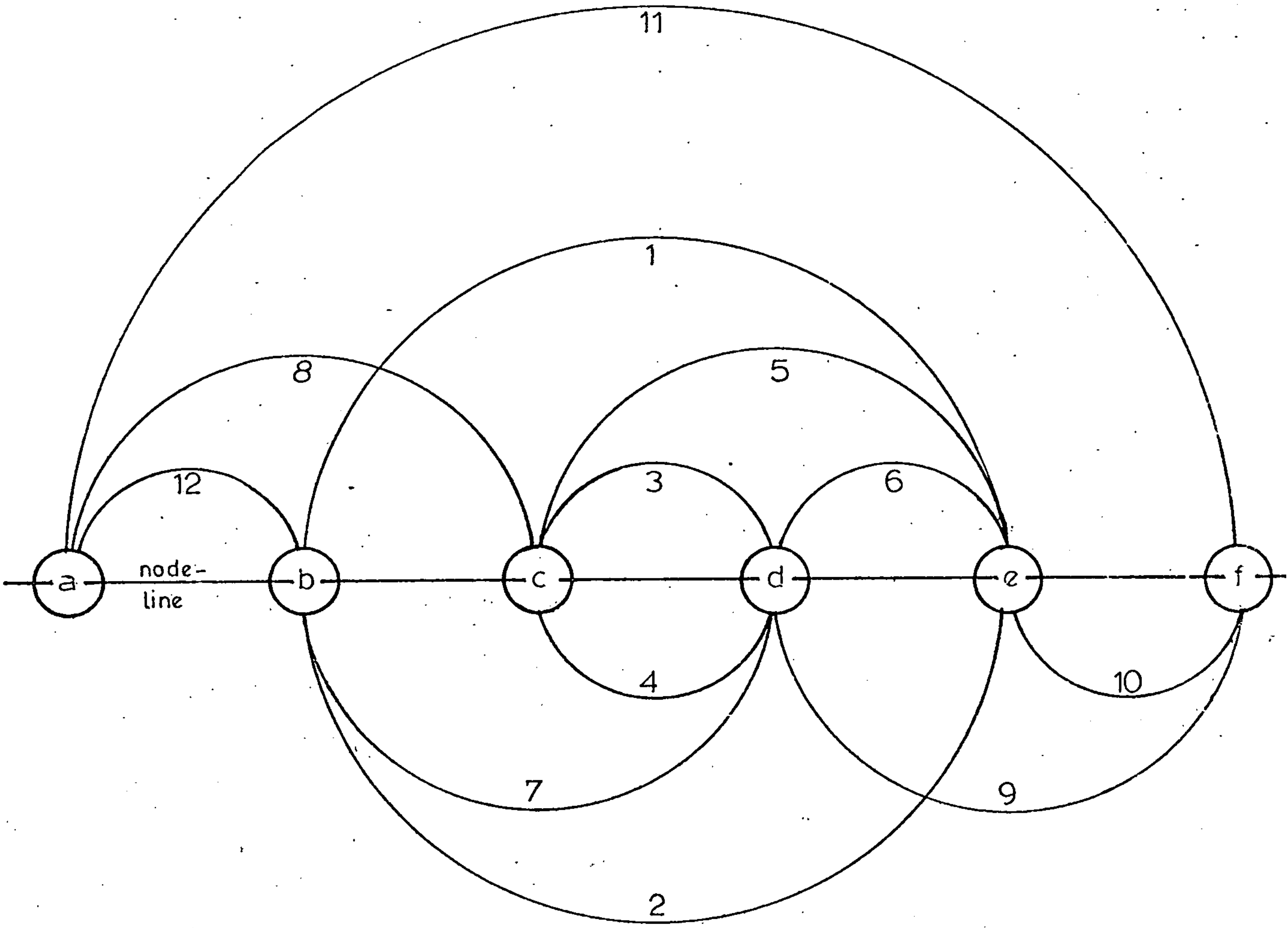


Figure 3.15 . Heuristic Crossover Evaluation.

iii one is between, the other outside (8,9)

iv neither are between (10,11,12)

It is apparent that only the third configuration causes a potential crossover. The routine is designed to locate such configurations and increment the appropriate (above or below) crossing total when it finds them. This routine is applied to each link which can be routed of the node in question. The result is the minimum number of crossovers indicating the associated optimal routing.

Links are routed above the node line unless it is more advantageous to route them below. The routing is accomplished by flagging the link entries as positive if routed above and negative if routed below. To avoid confusion between positively routed links and unrouted links, which arise from one node being absent from the node line, all links' entries are initially incremented by 128, which constitutes an 'unrouted' flag in bit 7 of the word. This flag is removed when both nodes are present in the permutation. When the initial permutation has been created, the optimisation phase is entered.

3.7.3 PERMUTATION OPTIMISATION

Single level optimisation is accomplished by shifting nodes one at a time to new positions on the permutation which give a maximal reduction on the number of crossovers. The node with the highest number of associated crossovers is selected for trial repositioning; the method is identical to locating the best position for a node in the initial permutation creation, that is, the node is gradually moved forward through the permutation until a position giving zero associated crossovers is found. Failing this, the position offering the maximal decrease in crossovers is used. If no decrease is possible, the original position is selected. The node is then fixed in the selected position and routing is optimised.

As the permutation is altered only by changes in node position, with associated routing optimisation, it is obvious that once a node has been selected unsuccessfully it is pointless to re-select it if no permutation alteration has since occurred. Furthermore, a node with no associated crossovers cannot be repositioned to advantage in a first level optimisation as no reduction from zero is possible (such repositionings would only be of possible use in a second level optimisation, which is not considered). These observations are useful in determining when the optimisation is maximal.

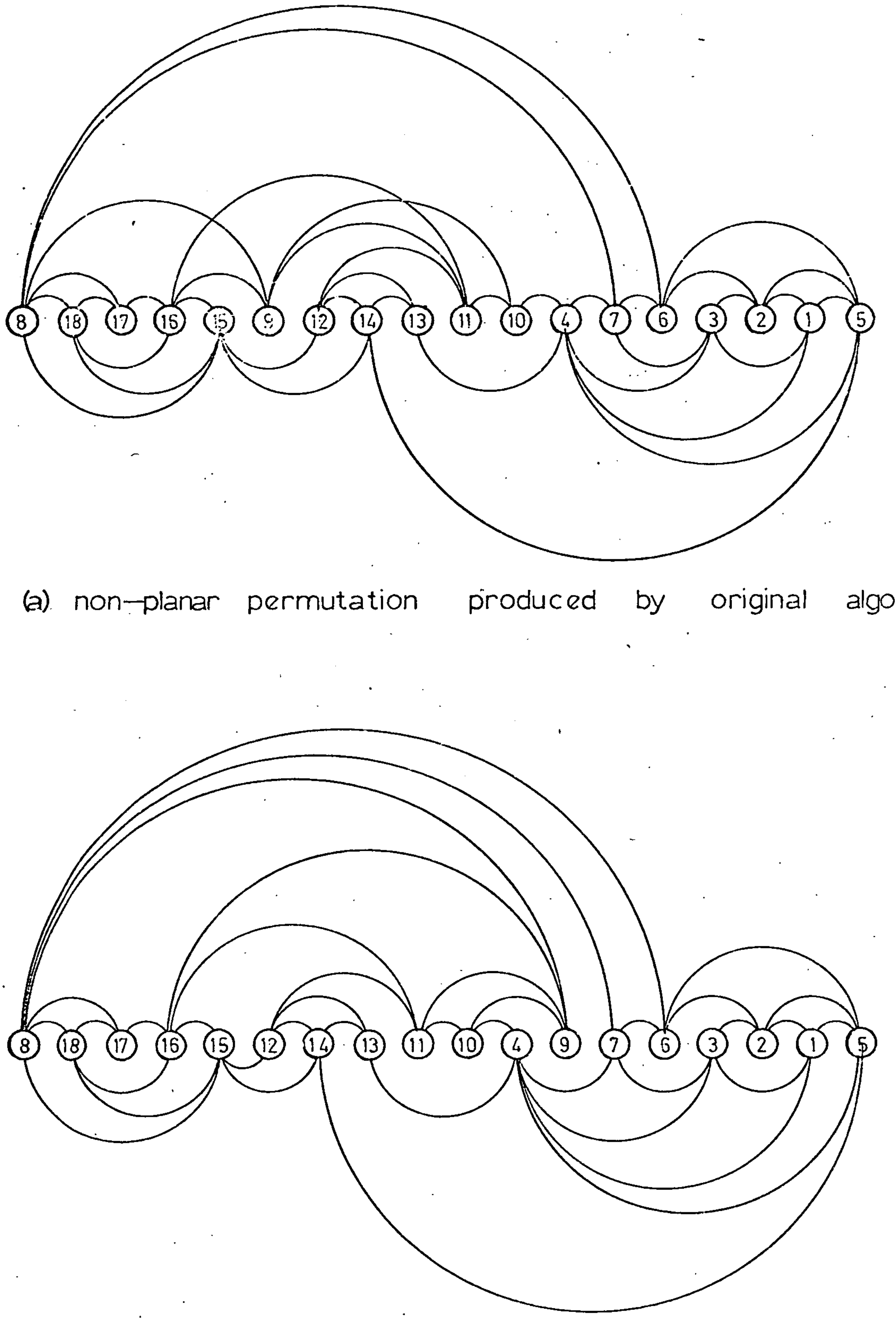
The nodes are selected in decreasing order of associated crossovers and, at most, each is selected once after each permutation alteration.

There are two situations which indicate that a local minimum has been reached. The first is when the total number of crossovers is reduced to zero and the second occurs when all non-planar nodes have been subjected to trial repositioning without success. When either of these two situations arise, the permutation has reached a local minimum with respect to first level optimisation and the procedure is terminated.

3.8 MODIFICATIONS

The first complete graph results indicated that the procedure was optimising the permutation as expected by changing node positions. However, as all nodes in a complete graph are equivalent in that each is connected to every other, the node positions are irrelevent and optimisation should be based solely on routing since this is a much faster method. The algorithm was consequently modified so that the linkage structure, prior to the optimisation stage, could be examined for intermediate crossing reductions by link re-routing. This is done very simply by travelling round the forward order ring and examining the links as they occur in the beads and, when a link is found to have fewer crossings if routed on the other side of the node-line, it is altered accordingly. This procedure is repeated until a complete trip around the ring produces no decrease in crossings, when the routing is said to be optimal with respect to the current node permutation. This modification was found to take, on average, the same amount of time as one optimising iteration and consequently is considered well worth executing.

The algorithm was found to have another serious fault as illustrated in figure 3.16(a), which is an optimised permutation according to Nicholson's algorithm. Links (16,11) crosses links (8,9) and (9,10) yet there are two obvious ways of removing these crossings. The first is to route link (16,11) around node 9, involving a three-semicircle routing which the algorithm cannot consider.



(a) non–planar permutation produced by original algorithm

(b) planar permutation acheived with modified version

Figure 3.16 Effect of Modification to Algorithm.

The second way is to reposition node 9 at position 4,11 or 12 and to re-route the appropriate link ((7,16), (10,4) or (4,7) respectively) to appear below the node line: this is a direct shortcoming of the algorithm.

The inability of the original algorithm to deal with this situation is caused by two factors. Connections between adjacent nodes can always be re-routed without affecting the number of crossovers, and routing of such links is therefore of no significance and they may be drawn as straight lines. However, when a node is interspersed between two adjacent connected nodes, the routing of the connecting link becomes significant. Secondly, this fact cannot easily be taken into consideration by the matrix-based optimisation as it is not readily capable of allowing a different consideration of links connecting adjacently positioned nodes.

This oversight was corrected by permitting the optimising procedure to re-route certain links not connected to the node currently on trial. If a node is positioned during a trial, so that its immediate neighbours are connected, the possible advantages of re-routing that connecting link are considered. This modification produces negligible time increase, while returning the permutation of figure 3.16(b), which is an optimal layout.

These two modifications have been incorporated and the algorithm has been qualitatively tested. Results are given in the succeeding section to allow a comparison between the modified heuristic and the matrix mathematical techniques.

3.9.0 RESULTS

The program was originally written for running on Systemshare Timesharing Bureau, however it became necessary to transfer it to a timeshared PDP10 in the Computer Science Department. The results listed here have been obtained on the PDP10 using a remote teletype coupled by MODEM and GPO telephone lines. The results are output on the following forms:

- i a teletype printout as in figure 3.17(b)
- ii a plotted drawing using a Calcomp 30" 563 (shown in figure 3.17(a))
- iii a binary file

Forms i and ii are for use in network analysis and the results presented in a format designed for comprehensability, whilst iii is for use in the subsequent layout phase discussed in chapters 4 and 5.

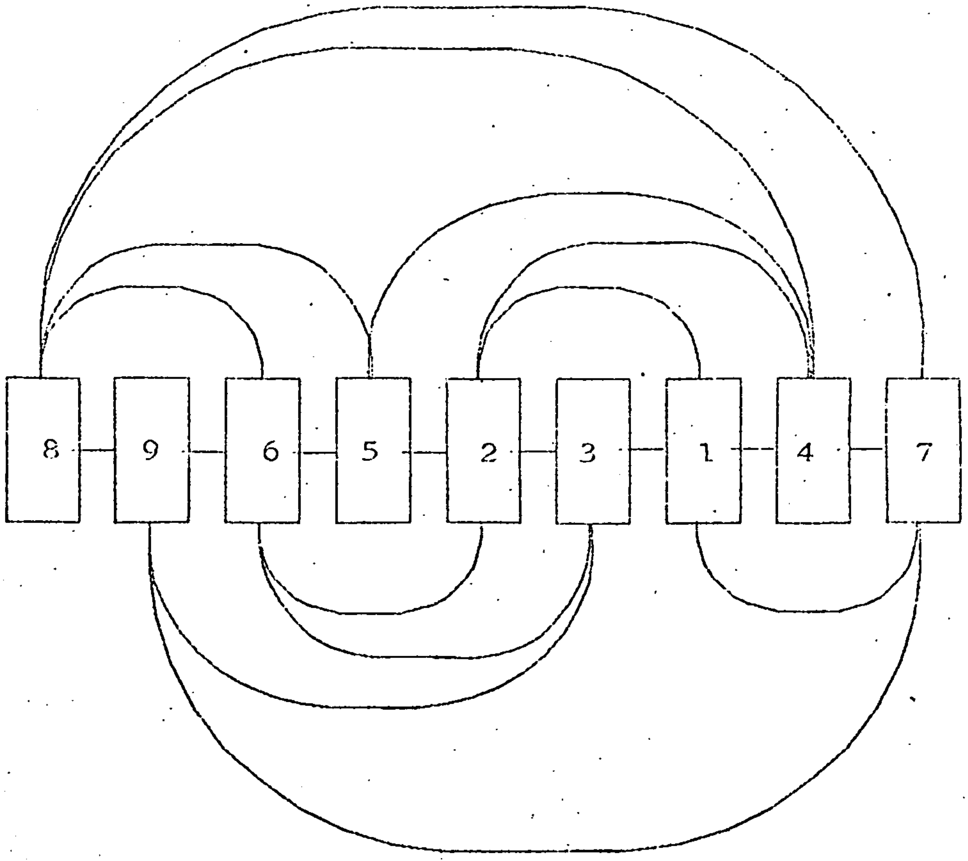
The qualitative testing was carried out on complete graphs and symmetric utility graphs (SUG) as the minimum numbers of crossovers have been determined for both by Saaty (1964) and Zarankiewicz (1954) respectively. These also allow a comparison between the original algorithm as programmed by Nichsolson on an IBM 7030 and the modified algorithm which was programmed as described above.

3.9.1 COMPLETE GRAPHS

Saaty (1964) formulated an upper bound for the minimum number of crossings in a complete graph representation as

$$\frac{N(N-2)^2\ (N-4)}{64}\ (N\ \text{even})$$

$$\frac{(N-1)^2\ (N-3)^2}{64}\ (N\ \text{odd})$$



(a) plotted output on CALCOMP 563

```
NODE POSN    CONNECTIONS
   8    1  :   4,   5,   6,   7,   9,
   9    2  :  -3,   6,  -7,   8,
   6    3  :  -2,  -3,   5,   8,   9,
   5    4  :   2,   4,   6,   8,
   2    5  :   1,   3,   5,   4,  -6,
   3    6  :   1,   2,  -6,  -9,
   1    7  :   2,   3,   4,  -7,
   4    8  :   1,   2,   5,  -7,   8,
   7    9  :  -1,  -4,   8,  -9,
```

(b) teletype output

Figure 3.17 Output from Topological Analysis.

Table 3.1 shows the results for the two algorithms applied to complete graphs of between 5 and 14 nodes. Additional results for 15 to 20 nodes are included for the modified algorithm, but Nicholson did not include comparative figures in his published results.

Table 3.1: Results for complete graphs

	Number of Crossings							
No.of	initial		final		min.	Time (secs)		
Nodes	orig	mod	orig	mod	poss	init	final	links
5	1	1	1	1	1	0.20	0.58	10
6	3	3	3	3	3	0.41	0.85	15
7	9	9	9	9	9	0.84	1.26	21
8	20	20	18	18	18	1.32	3.30	28
9	36	36	36	36	36	1.94	3.68	36
10	62	60	60	60	60	2.52	6.86	45
11	102	100	100	100	100	3.76	14.22	55
12	157	150	150	150	150	5.10	16.32	66
13	231	225*	225	225	225	7.32	24.46	78
14	325	315	315	315	315	8.30	45.16	91
15		441*		441	441	13.06	51.06	105
16		588		588	588	14.78	72.82	120
17		784		784	784	19.48	120.04	136
18		1008		1008	1008	25.70	154.50	153
19		1296*		1296	1296	39.58	187.02	171
20		1620		1620	1620	40.94	364.30	190

*achieved through routing optimisation

The results indicate that the routing optimisation is well worth the inclusion as the algorithm did not fail to produce an initial permutation with minimised crossovers, whereas the original algorithm required optimising alterations which are unnecessary in 6 out of 14 cases, which includes every graph with more than 10 nodes.

3.9.2 SYMMETRIC UTILITY GRAPHS

The formulae for the minimum crossings, established by Zarankiewicz (1954) for a symmetric utility graph on 2N nodes are:

$$\frac{N^2\ (N-2)^2}{16} \qquad \text{(N even)}$$

$$\frac{(N-1)^4}{16} \qquad \text{(N odd)}$$

The results are tabulated below, together with comparative figures from Nicholson where available.

Table 3.2: Results for Symmetric Utility Graphs

nodes	links	Number of Crossings				minimum possible	time for modified
		initial		final			
		orig	mod	orig	mod		
6	9	1	1	1	1	1	0.80
8	16	5	5	5	4	4	2.44
10	25	16	16	16	16	16	4.42
12	36	39	39	36	36	36	11.62
14	49	81	81	81	81	81	18.42
16	64	151	149	144	144	144	44.64
18	81		256		256	256	57.18
20	100		408		400	400	149.86
30	225		2401		2401	2401	671.60

The second modification is probably responsible for the difference in results between the two algorithms as applied, to the 8 node SUG. Additionally, in only one out of six comparable cases the routing optimisation produces less initial crossings, but this is to be expected as only three of these cases, as created by Nicholson's algorithm, have non-optimal initial permutations.

3.9.3 RANDOM NETWORKS

To provide a temporal comparison, it was necessary to process a number of randomly generated graphs, as these were the only timed results given by Nicholson. A program was written to produce these graphs, which generated a series of discrete random links until the number of nodes and links specified had been included. The results are shown in table 3.3 below with timed results for comparable networks, as provided by Nicholson.

Table 3.3: Results obtained from Randomly Generated Graphs

		Number of Crossings				Total Program Time	
		Initial		Final			
Nodes	Links	orig	mod	orig	mod	orig	mod
10	15	0	0	0	-	.9	0.94
10	15	1	0	1	-	4.1	0.89
10	16	0	0	0	-	.8	0.95
10	30	18	19	12	12	10.2	11.68
10	30	16	12	9	9	5.2	8.66
10	30	16	18	14	11	4.2	17.96
15	23	2	2	0	1	15.2	4.52
15	23	0	0	0	0	4.3	1.38
15	23	1	2	0	1	8.2	4.48
15	45	38	36	26	26	40.2	47.82
15	45	31	33	26	29	31.1	48.28
15	45	41	36	19	26	105.2	63.26
15	68	126	115	101	108	57.6	60.00
15	68	121	120	105	111	61.3	108.54
15	68	132	123	101	99	61.0	146.34
20	30	10	10	3	3	194.6	20.92
20	30		5		1		17.08
20	60	59	72	57	54	151.7	157.70
20	60		72		68		108.20
20	90	233	207	183	179	474.8	255.82
20	90		201		164		293.04

These results are not directly comparable as the networks used for the two algorithms are not identical, but have the same numbers of nodes and links each. The times indicate that the heuristic procedure is as fast as the matrix-based procedure, which is quite surprising in view of the extra processing required to access information held in the data structure. However, there is a wide range of differences which are probably caused by individual differences between the networks processed, as well as the machines used. Overall, the implication is that the cost of saving storage space is not at all expensive in time. Figure 3.18 shows the storage requirements of the alternative storage structures for a range of network sizes.

3.10 NET REPRESENTATION RESULTS

Once the program was producing results, a closer study of net representation was undertaken. The underlying rationale has been discussed in section 3.6, however the representation was not yet practically proven. The test was carried out on a relatively highly-connected graph, (figure 3.19). The network consists of 9 nodes and 17 nets, 7 of which connect more than 2 nodes. The nets are as follows:

(1,2) (1,4) (2,3) (3,5) (4,5) (5,6) (5,7) (6,9)(7,8)
(8,9) (1,2,3,) (1,2,3,4,5,) (3,5,6,9) (1,4,5,7)
(5,6,7,8,9) (7,8,9) (1,3,7,9)

The results have been tabulated in table 3.4 to indicate the degree of success achieved with each of the representations with respect to real and induced crossings produced in the final drawing.

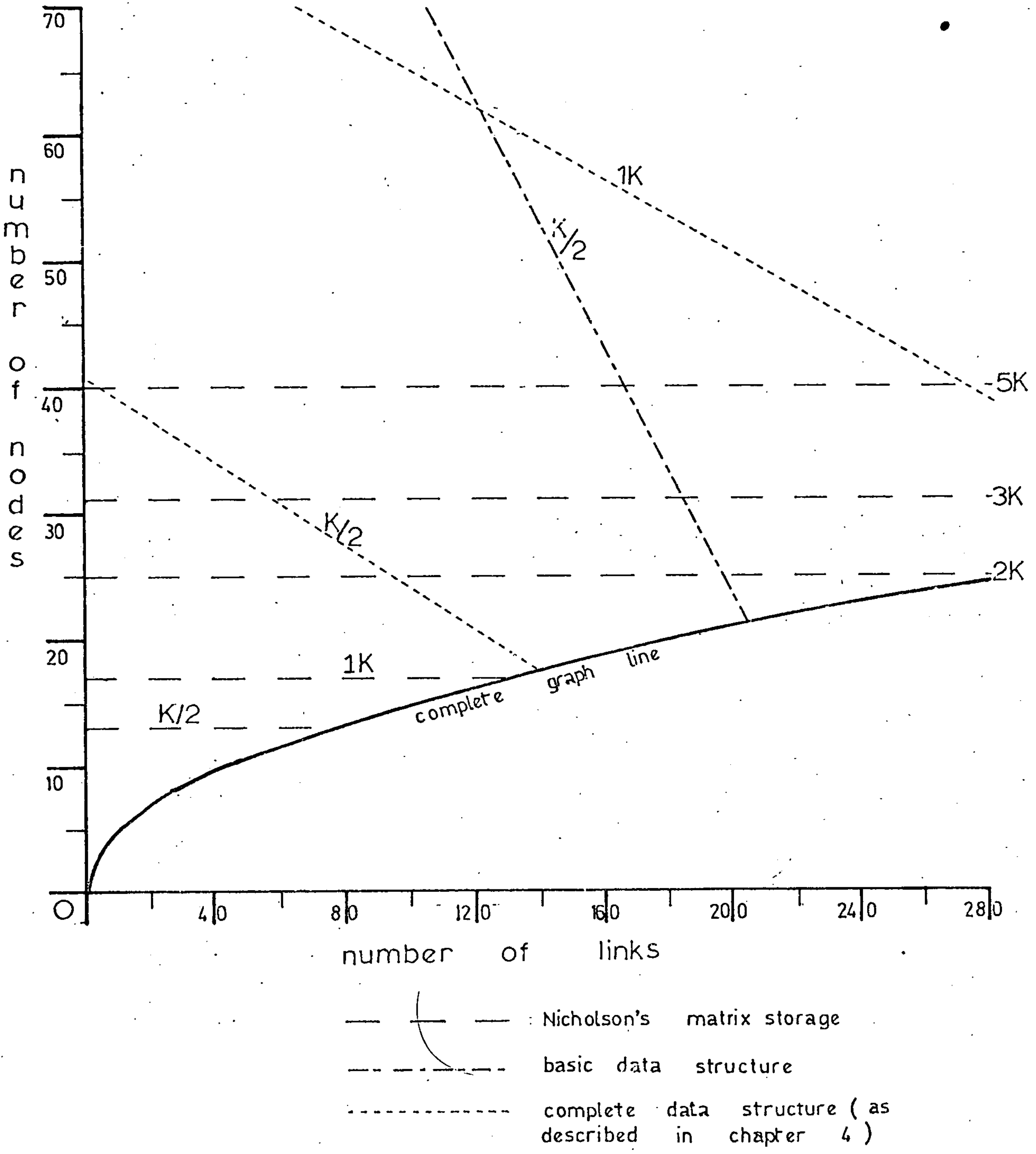
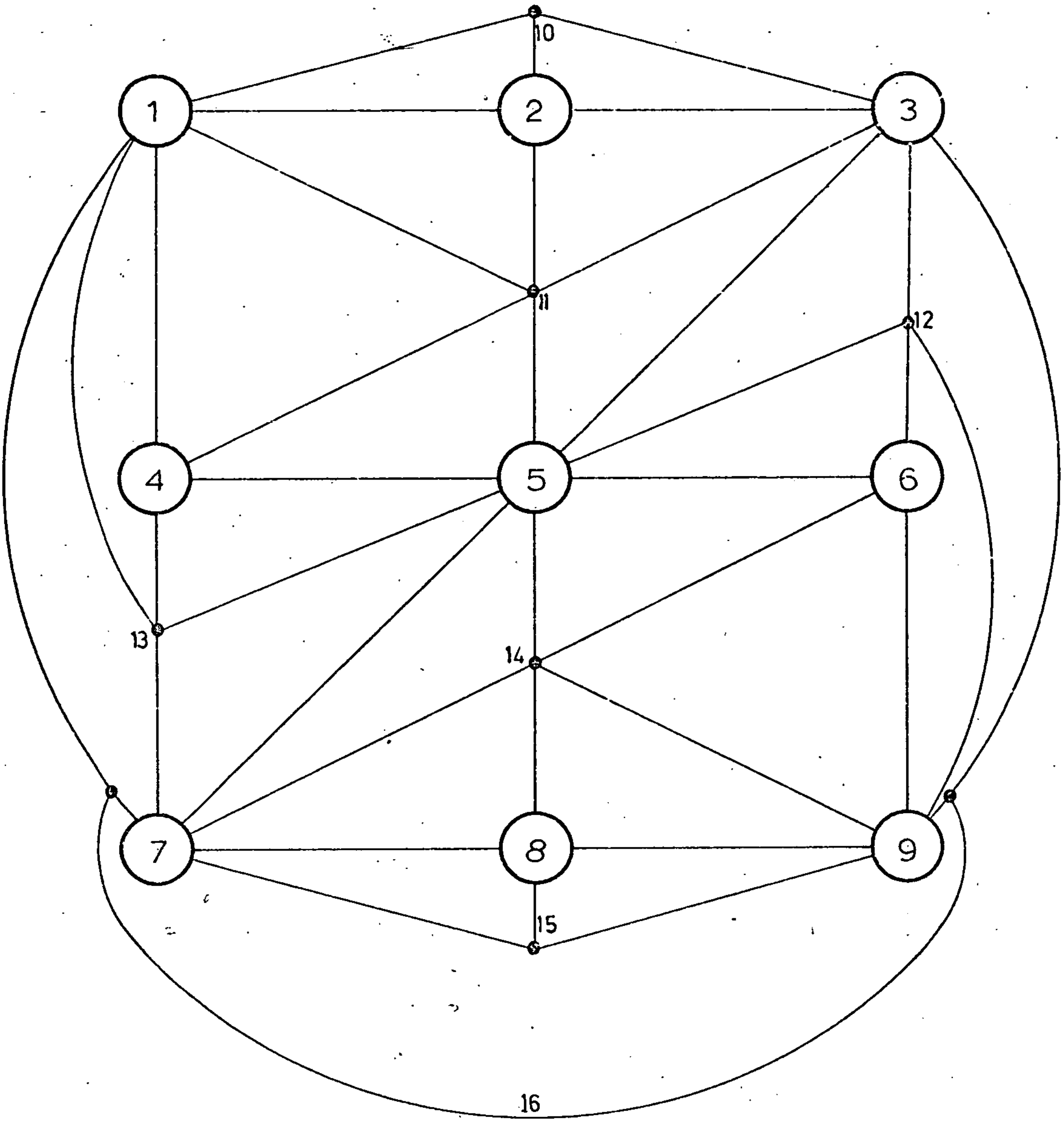


Figure 3.18 Comparison of Storage Requirements.



numbers 1 to 9 identify nodes
10 to 16 identify nets on 3 or more nodes

Figure 3.19 Network used for Comparative Net Representation Tests.

Table 3.4: Net Representation Results

	No.of Nodes	No.of Links	Crossovers		Time	Crossings in drawing	
			Init.	Final		actual	induced
Chain	9	31	3	1	14.94	3	7
Cycle	9	38	6	5	16.48	2	4
Complete Graph	9	56	24	16	38.90	1	0
Node	16	38	0	0	11.36	0	0
Tree i	9	31	0	0	7.04	0	0
ii	9	31	3	0	10.78	0	2*
iii	9	31	2	0	9.18	0	1
iv	9	31	0	0	7.34	0	0

*these can be removed via planar re-routing

The chain representation used the following nodal orders for the nets:

(1,2,3) (1,2,3,4,5) (3,6,5,9) (1,4,5,7) (5,6,7,8,9)
(7,8,9) (1,7,9,3)

and the following node sequences were used in representing the nets as cycles:

(1321) (135241) (39653) (15741) (579685) (7897) (19371)

When drawn, the chain and cycle representations each produced net crossings due to unilateral nesting of the nets. This was because the nets (1,2,3) and (1,2,3,4,5) both contained links (1,2) and (2,3) which were both routed in the same places. These drawings could be considered as induced crossings by under-node routing, however there were additional induced crossings in both cases. The main fault with this representation is that the algorithm cannot distinguish apart links between the same nodes belonging to different nets. Instead, such links will be routed identically, which is equivalent to inducing a crossing.

The complete graph representation similarly produced crossings, both real and induced, with the major difference that, when drawing the network, there is a choice of which links are to be used for the net as it is unnecessary to draw them all. This did not ease the problem any, rather it confused the procedure. The time requirement for this representation is suitably long and no advantage was gained.

The best results with no crossings were produced by introducing nodes to represent the nets and resulted in an accurate drawing of the network, displaying the same structure as that of figure 3.19. The time was faster than any of the above representations despite the presence of additional nodes, because the initial permutation was planar.

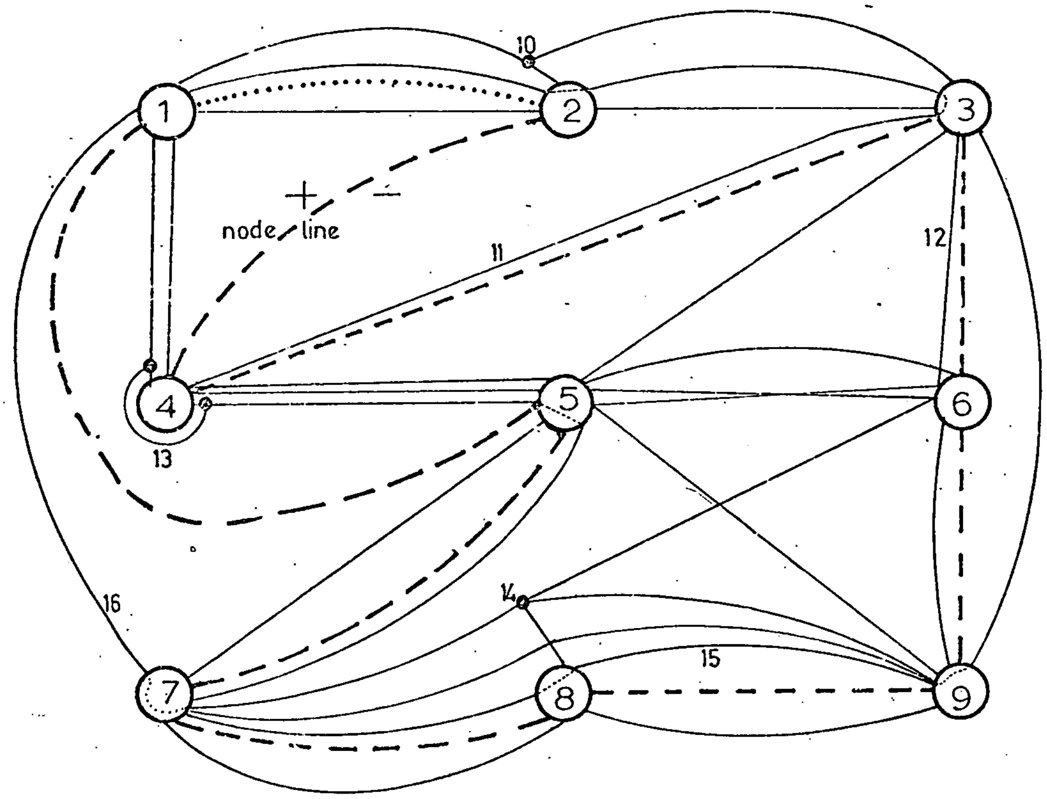
Four spanning trees representations were processed with the tree trunks being the nodes indicated in table 3.5. Only two cases produced induced crossings and in both cases simple re-routing removed them. They arose from the choice of routing above, or below, the node line with equal advantage and the machine chose the wrong side, although it could not be expected to differentiate as it is unable to recognise terminal pin ordering.

Table 3.5: Nodes used for spanning tree representation

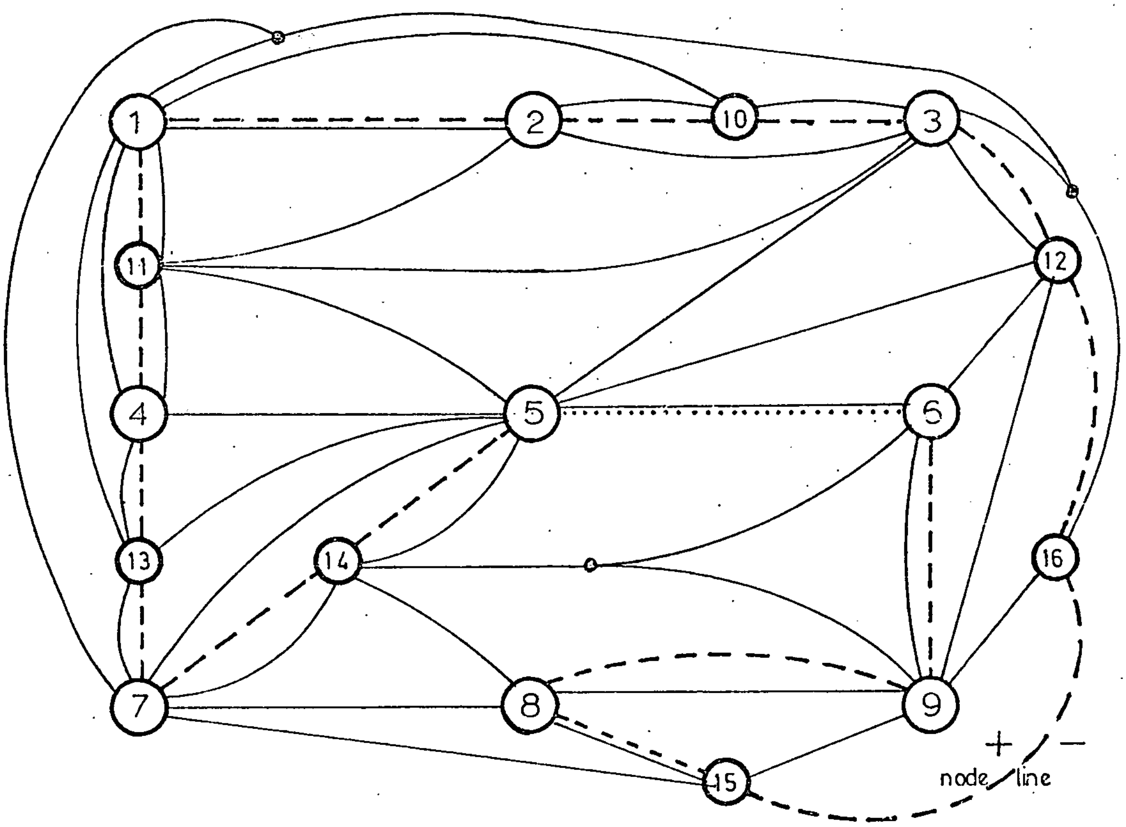
Case	(123)	(12345)	(3569)	(1457)	(56789)	(789)	(1279)
1	1	4	6	7	8	9	3
2	2	5	9	2	9	7	1
3	3	1	3	4	5	8	7
4	1	2	5	6	1	9	9

The results as drawn from chain, node and spanning tree i representations are shown in figures 3.20(a) (b) and (c) and indicate the correspondence between the output and the original. It should be noted that the node-line orientation could be drawn in either of two ways above,corresponding to the inside or outside of the curve. The orientation for the drawings was chosen to produce the simpler drawing in all cases. For example, the network drawn in figure 3.20(d) is the same as in figure 3.20 (b), but with the opposite orientation.

Thus the introduction of a node is, as expected, the most accurate representation producing the best results, especially when drawing a network. Spanning trees are almost as good, however they can produce induced crossings, which might not be as easily removed as in the above instances. Complete graphs, chain and cycle representations are of little use as they complicate the layout with real and induced crossings as well as a super-fluidity of links.

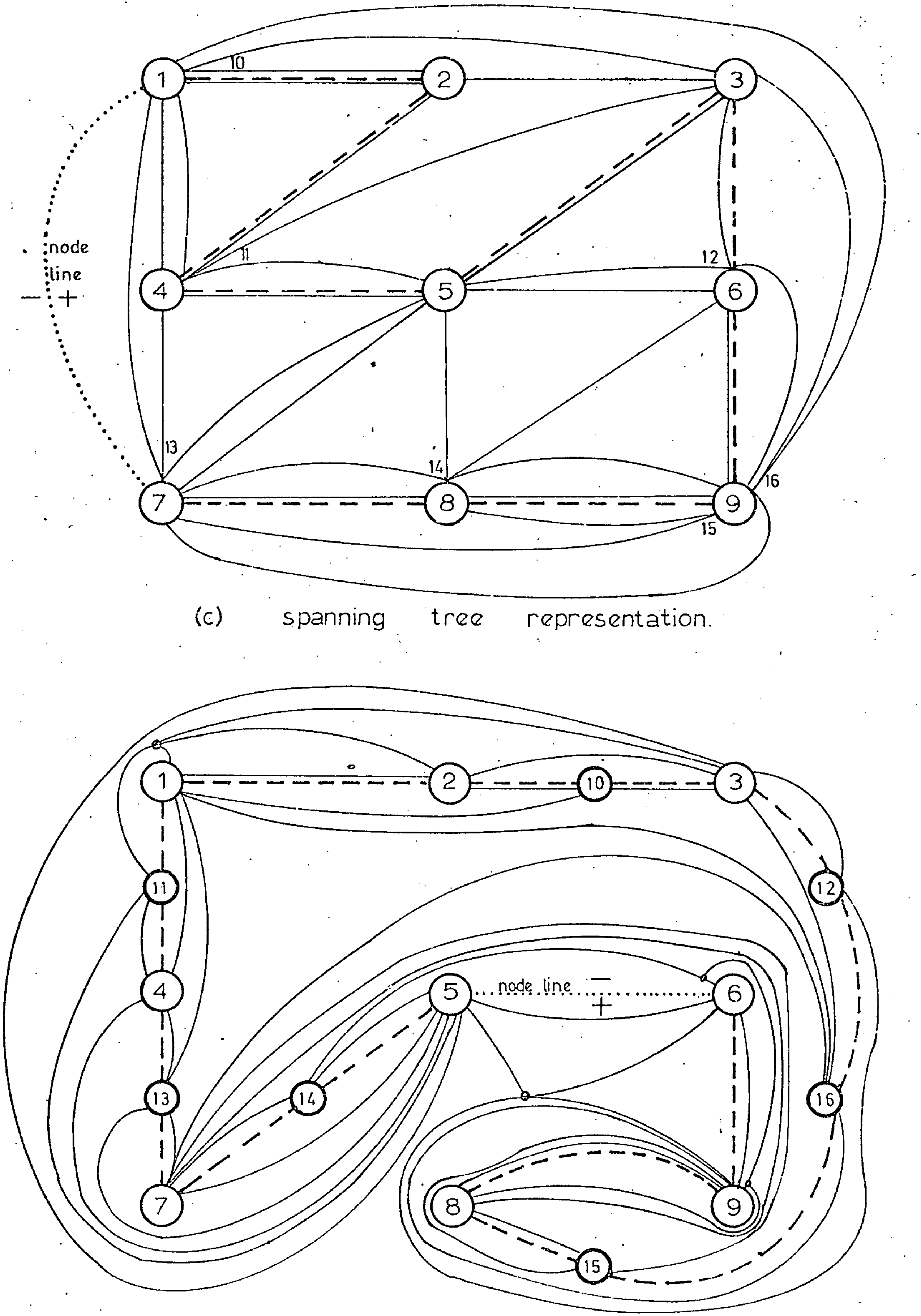


(a) chain representation.



(b) node (or spider) representation.

Figure 3.20 Net Representation Results.



(c) spanning tree representation.

(d) reversed node representation

Figure 3.20 Net Representation Results

3.11 CONCLUSIONS

The results produced by the modified algorithm indicate it is ideally suited for use in conjunction with a layout algorithm for several reasons. It is compact and fast, which is an essential feature for use on limited size time sharing systems. It provides detailed instructions for drawing the network so as to achieve the minimum number of crossovers and consequently, provides information relating component positions to track densities.

Throughout this chapter these features have been discussed and their advantages noted; the following chapters will discuss the realisation of a trial layout based on the topological model supplied by the procedure.

CHAPTER IV

PLACEMENT OF COMPONENTS

4.0 LAYOUT REALISATION

The layout phase may be partitioned into placement and routing, as discussed in chapter 2; although this division is convenient it creates problems as the two sections will have substantial interdependance. Thus there should be a means of communication between the two so that components are placed with some consideration of routing requirements. Communication between placement and routing is necessarily two-way if the system is to be an accurate model of the current manual layout method. Alternatively, layout may be executed with a single program which would automatically contain consideration of the inter-relationships but it would require considerably more machine space than either of the partitioned sub-problems. The use of a topographic analysis program provides a third alternative; positional data relating components and connections is supplied, in graph terms, which can be used as a substitute for the mutual considerations of the sub-problems by allowing each to reference the analysis data. This can be considered as replacing the direct two-way communication with two unidirectional channels to a common set of reference data. Each division can obtain information about the requirements of the other related to its own computations, thus allowing independant operation. The latter two alternatives shall be discussed in greater depth below with reference to component placement within the limitations of this work.

4.1.0 LAYOUT EXECUTION

The first method considered is the single program solution which implies that connections are routed concurrently with component placement; clearly, a route can only be determined when all the points that it connects have been placed on the substrate and so routing is necessarily the latter of the two processes in terms of single components; that is, a component must be positioned before it can be connected. Since we are discussing a non-partitioned solution it is inappropriate to consider a separation. It would be feasible to divide the circuit components into highly connected groups for individual laying out; this is a partitioning of layout elements rather than layout processes and within each group the same complications will arise. Furthermore such a course of action would require additional programming effort to supply the elemental partitioning. An alternative would be to modify the placement when routing difficulties arise, however this is unsatisfactory since each modification would probably involve a major reshuffling of components because moving one component would undoubtedly cause a chain reaction amongst the others. This implies that as each connection is routed a modified placement would have to be generated, thus requiring too much time for efficiency.

The main disadvantage of a single solution is the program size. Since timesharing bureaux supply limited segments of core, it is obviously better to break the solution down into sections, which can be executed consecutively rather than concurrently. Therefore a single program solution is not considered suitable for the scope of this work and will not be considered further.

4.1.1 PARTIONED PLACEMENT

The placement of components in this application is unconstrained, as seen in chapter 2. The existing techniques are all based on force placement. Connections are recognised solely as generations of attractive forces which have no spatial requirements; in addition, no allowance could be made for planarity considerations since any consideration of topographic relationships between components and connections is beyond the scope of these algorithms. To include consideration of a connection pattern, such as that defined by the permutation procedure, would remove much of the components' freedom of movement which is fundamental to force placement. The algorithm would therefore be severely handicapped and consequently inefficient. This disadvantage applies to any method which placed all components simultaneously since any predetermined relationships would be too severe a constraint upon movements. Therefore, placing components to take advantage of an optimum topographic configuration must be sequential, that is, each component must be placed individually, solely on the basis of the previously positioned components' locations and the results of the topographic analysis.

4.1.2 PLACEMENT BASED ON THE PERMUTATION DESCRIPTION

Sequential placement obviously requires that the components be ordered so as to maximise the placement algorithm's efficiency: the permutation description appears ideally suited to supplying a component sequence as it relies on the order of the nodes as they appear on the node-line. In addition, the permutation description supplies all the data on the connection pattern within the vicinity of each node,in terms of quantity and relative location of links. This information is suitable for calculating spatial requirements of connections in the neighbourhood of each component. These two features have resulted in the permutation description of an optimum network configuration being used as the basis for the layout procedure, specifically the component placement.

The corollary of the theorem in appendix III implies that if the components are placed so that a simple curve may be drawn to pass through them in the same order as the corresponding nodes appear on the node-line, then the connections may be routed with the same crossing structure (ignoring induced crossings). This is the basis of the placement procedure and provides crossover minimisation through the topographic analysis. Moreover, circuit area, as defined by the minimum boundary rectangle, must be minimised; this includes the space occupied by components and connections. Therefore, the components should be packed as tightly as possible while leaving areas for connections; to reserve too little would present major difficulties during routing, whereas too much would produce inefficient layouts. The placement aims may therefore be stated as follows:

i preserve node-line curve simplicity

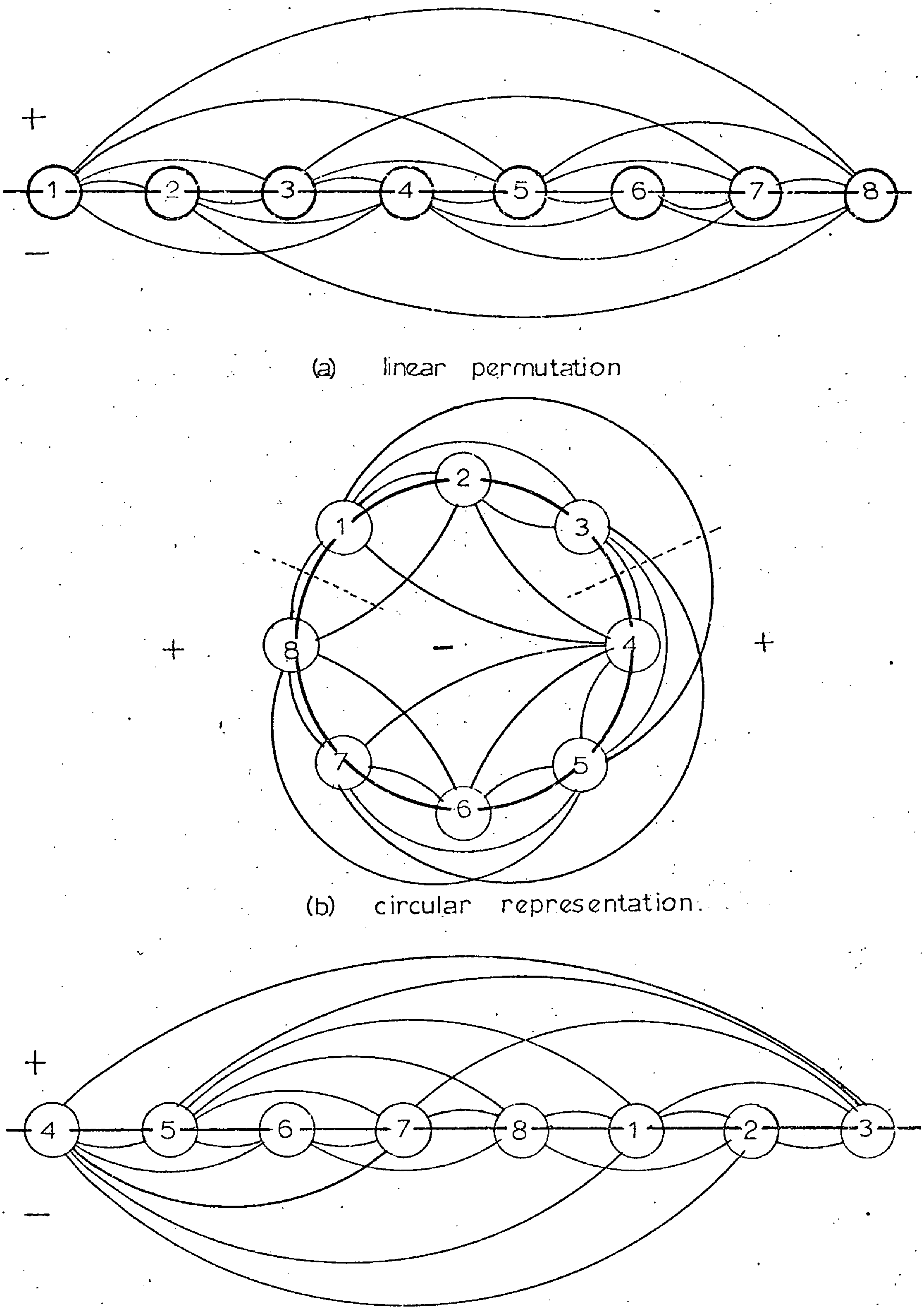
ii minimise circuit area

iii reserve space necessary and sufficient for connections

4.2.0 NODE LINE CONSIDERATIONS

A major feature of the node-line supplied by the permutation procedure is that it is not unique; it may be used to supply N-1 (N = the number of nodes in the network) different optimal permutations by shifting the end point of the node-line. This can be readily verified by considering figure 4.1. The linear permutation of 4.1(a) has been redrawn as a circle in figure 4.1(b), which has been used to produce the linear permutation shown in figure 4.1(c). It can be seen that the three drawings contain identical link structures, that is, the same nodes and links with the same crossings (1,4)x(2,8); (1,5)x(3,7); (3,7)x(5,8); (4,7)x(6,8). Therefore, since there are N positions for locating the break-point in the circular node line, there are N different permutations readily available from one permutation description.

If the placement is regarded as a positioning of the node-line, it may be thought of as a problem of continuously deforming a straight line to preserve a simple curve whilst complying with the spatial requirements. This may be crudely conceptualised as laying a string of rectangular beads flat, within a rectangular area, so that the string does not cross itself and the beads are adequately separated to allow interconnections as defined by the permutation links. This idea is used to generate different schemes for positioning the node line, these are discussed below and evaluated on the basis of their ability to embody the basic aims of the placement procedure. They are all based on a continuous deformation of the node line as a single entity.



(a) linear permutation

(b) circular representation.

(c) reformed linear permutation.

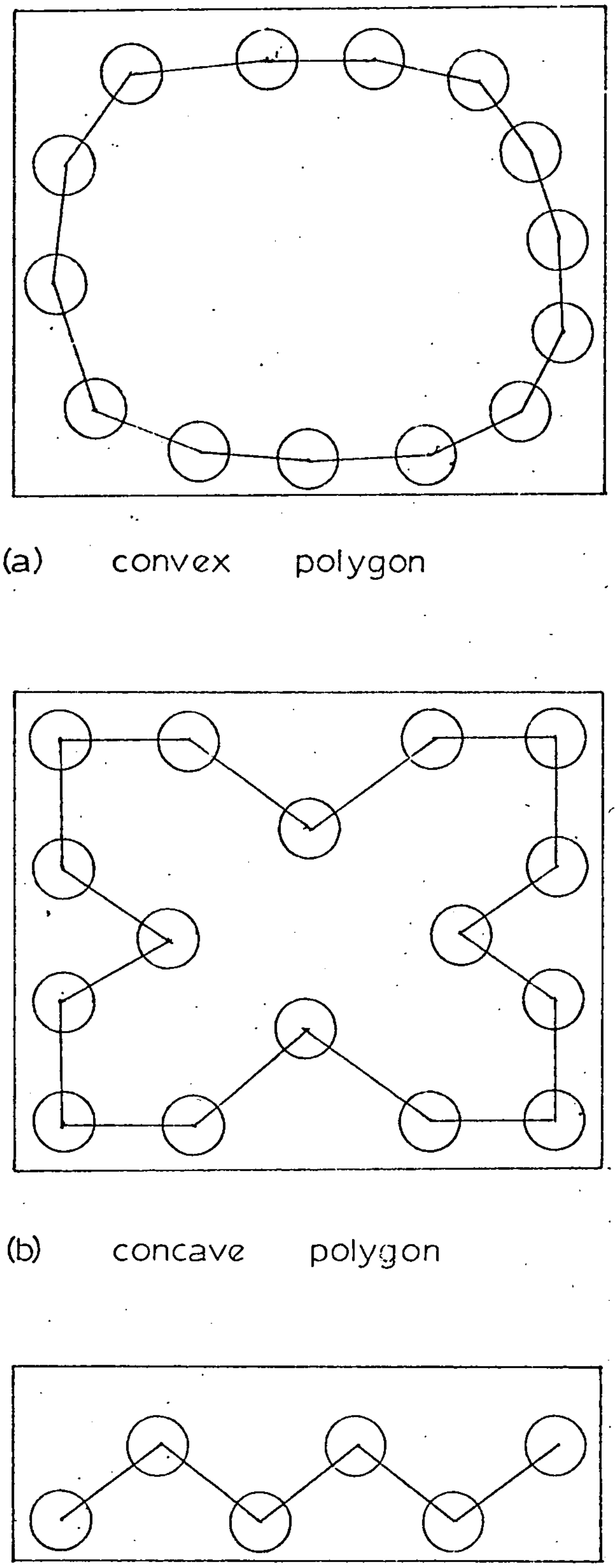
Figure 4.1 Permutation Cycling.

4.2.1 POLYGONAL DEFORMATION

The first and obvious method of arranging the node-line in compliance with the above is as a convex polygon, shown in figure 4.2(a). This is clearly of little use owing to the large amount of wasted internal space. To overcome this shortcoming, it is necessary to inject components into the central area with a concave polygonal deformation such as that shown in figure 4.2(b). Such a configuration is ideally suited as an abstract means of occupying space compactly, however it is not sufficiently versatile for applications involving non-standard shapes. Consider figure 4.2(c) where the substrate area is required to be a narrow rectangle, the first portion of the node-line is positioned as shown which leaves a narrow strip at the top for subsequent components, these will be jammed between previous placements and the substrate boundary. This produces a severe limitation on positions for larger components, resulting in a strong chance that some would be unplaced in addition to constricting connections in that area of the substrate.

4.2.2 SPIRAL DEFORMATION

The simple curve which can best be modified to fill a rectangular area with little waste area is a spiral. This is also quite versatile, as illustrated in figure 4.3(a), since it travels around the outside of the occupied area and can position components in any location on the boundary of previous placements. The major disadvantage with this scheme is the resulting routing configuration; the routes must not cross the node-line and hence long paths will have

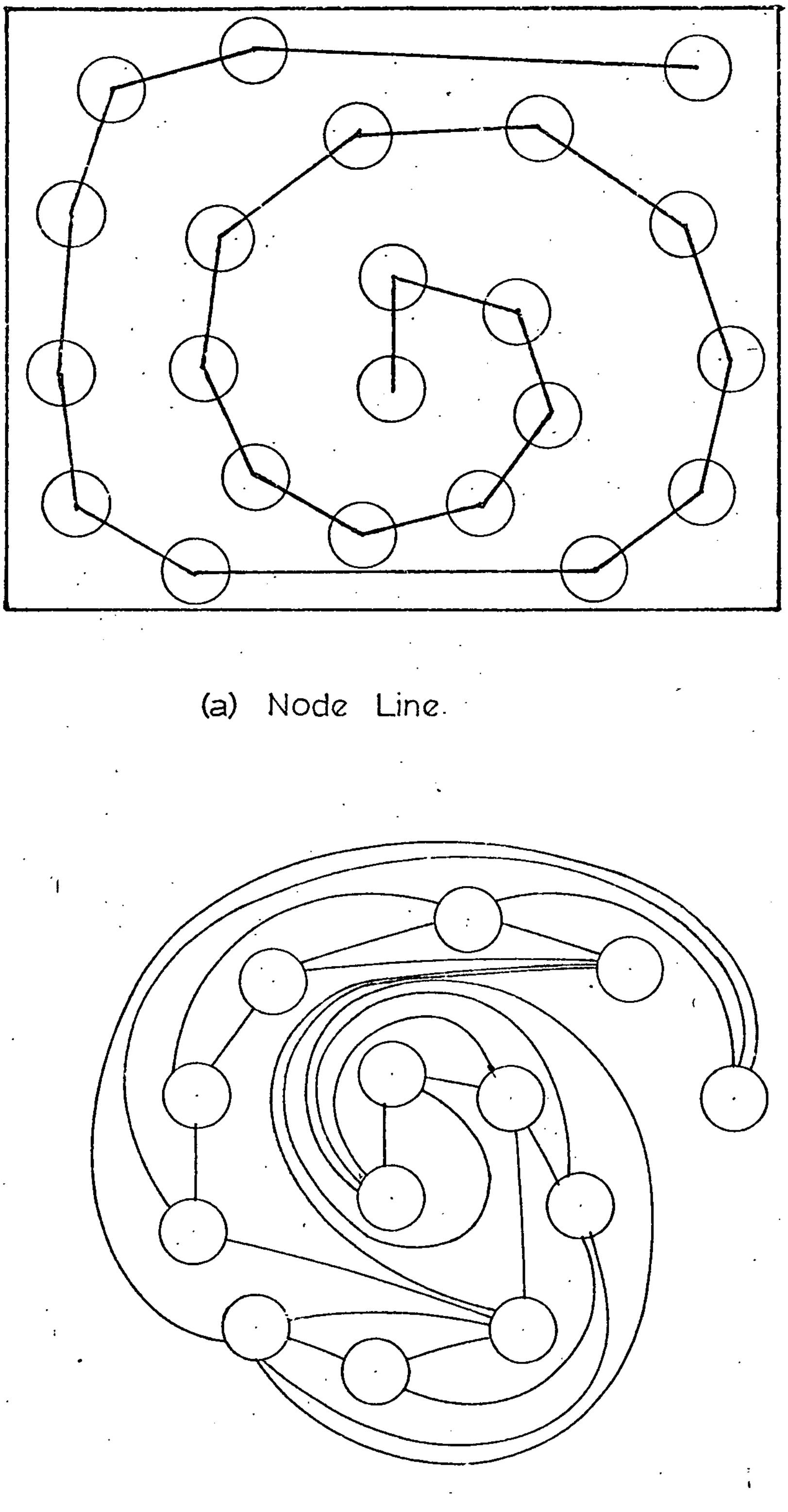


(a) convex polygon

(b) concave polygon

(c) limitations

Figure 4:2 Polygonal Node-line configurations.



(a) Node Line.

(b) Routing Pattern.

Figure 4.3 Spiral node line configuration

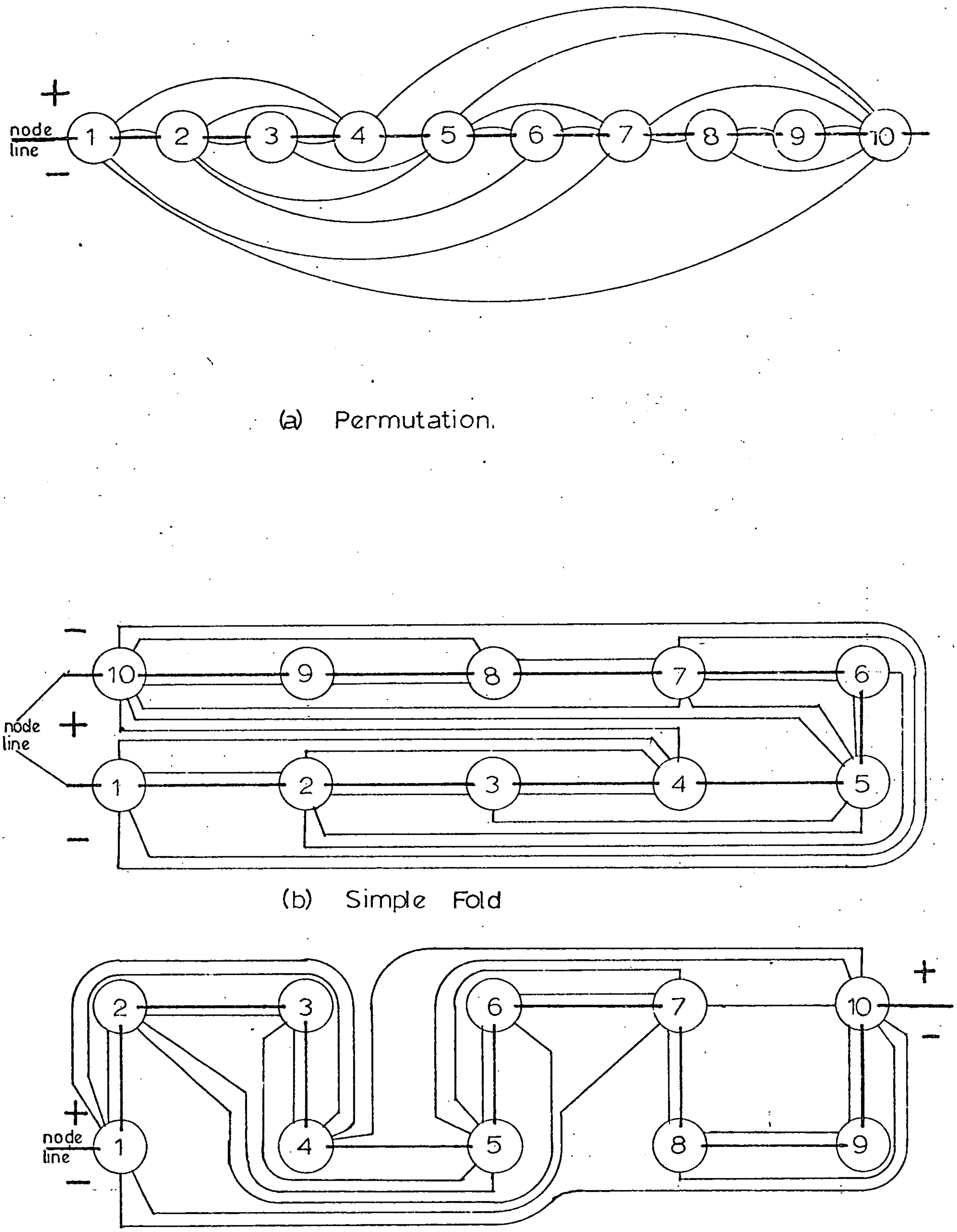
to travel parallel to the spiral. As can be seen in figure 4.3(b) this produces inefficient, long and convoluted routes and, therefore, the spiral configuration is deemed unsuitable.

The ideal configuration is one which allows each component to be placed on the perimeter of the area occupied by its predecessors, utilises area efficiently and avoids unnecessary limitations upon subsequent positional choices in addition to simplifying the routing configuration. Manual placements are generally compact and neat (visually pleasing), and this latter feature, although subjective, led to the next configuration, which is discussed below.

4.2.3. FOLDED CONFIGURATIONS

Constrained placements are based on a regular array of locations in rows and columns; this idea can be adapted, in this instance, to the unconstrained case by folding the node-line so that it travels back and forth along imaginary rows, assuming the shape of a folded firehose. This configuration produces layouts similar to figure 4.4(b), although the connections are simple, that is, composed mainly of rectilinear line segments, there are several links passing around the outside of the bends in the fold which implies that the connection lengths would not be minimal. To reduce these connection lengths, it is necessary to reduce the fold lengths (or alter the permutation, which is not considered here).

Corrugating the node line prior to folding, although producing many more folds, does halve the length of the firehose-type folds since the distance between the nodes at



(a) Permutation.

(b) Simple Fold

(c) Corrugated Fold

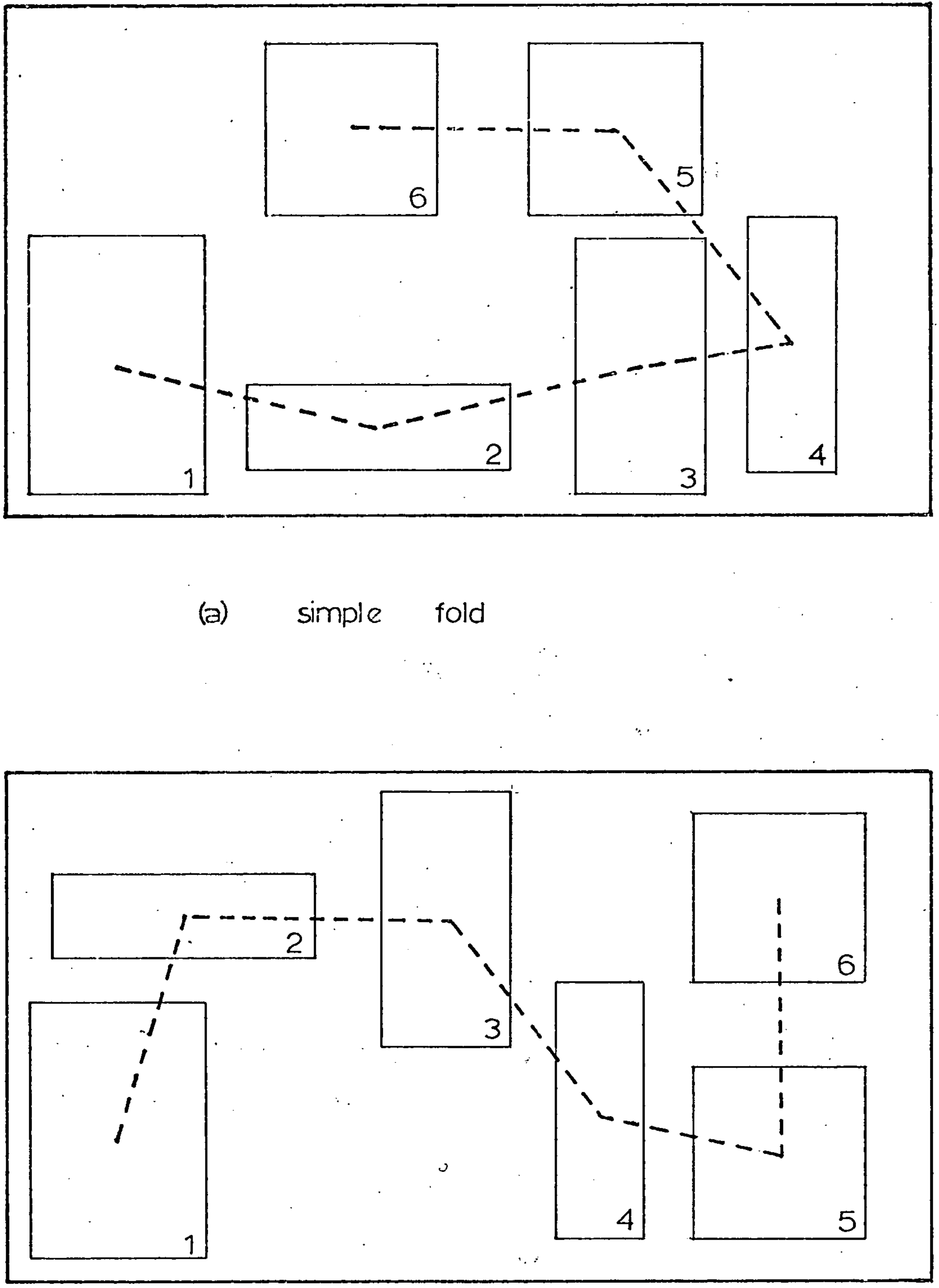
Figure 4.4 Folded Node Line Configurations.

each end of the original folds is halved. Connections are not required to follow the minor corrugations but only the major folds; this does reduce connection length as can be seen in figure 4.4(c). The permutation shown (figure 4.4(a)) has been folded (figure 4.4(b)) and corrugated (figure 4.4(c)) resulting in connection lengths of 54 units and 42 units respectively (a unit is equal to the corrugation depth), a decrease of over 20%.

Although it is possible to adapt a folding scheme to reduce connection length by additional corrugation, such schemes are not capable of dealing satisfactorily with components of differing shapes and sizes. This inadequacy is highlighted in figure 4.5 which shows the two above-mentioned folding techniques applied placing a sample set of components, without routing considerations. The two placements contain large gaps, which is clearly a violation of the first criterion of area minimisation. Node-line configurations based on rectilinear deformations are generally inadequate for dealing with a range of component dimensions.

4.2.4. CONCENTRIC POLYGONS

The final alternative considered is segmenting; this involves breaking the node-line up into a series of groups and placing each group as an entity, such as the concentric polygons shown in figure 4.6. For efficiency, the topographic analysis would have to supply the segmentation and include this when calculating the optimal permutation; it would probably be required to deal with several distinct, but related, node-lines. The segmentation would have to take into consideration number and size of components so that



(a) simple fold

(b) corrugated fold

Figure 4.5 Component Placement using Folding Strategies.

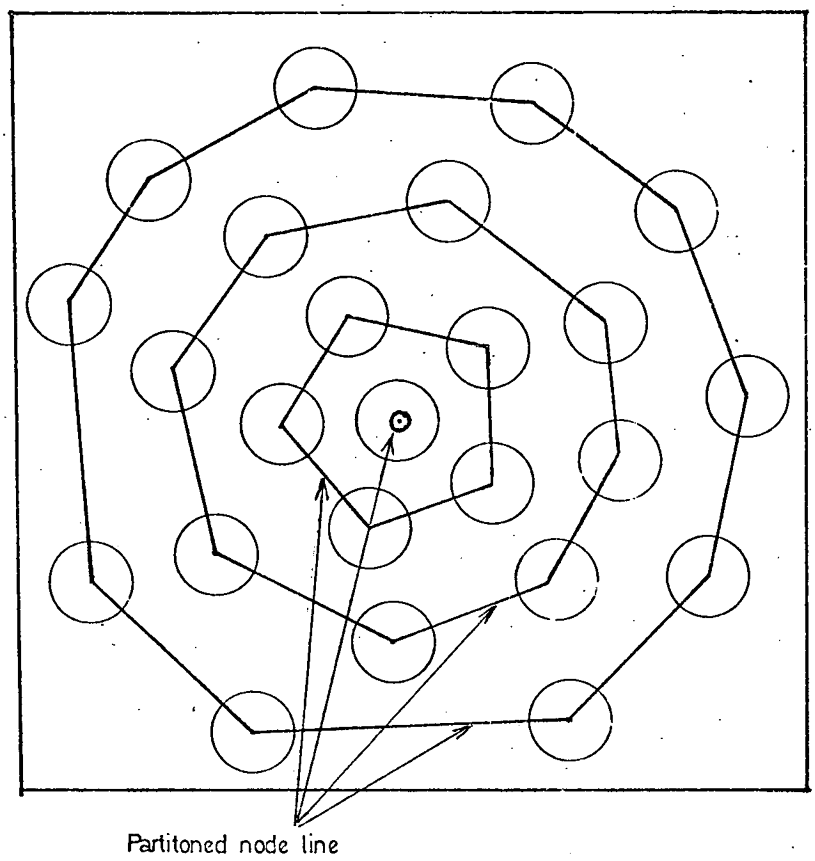


Figure 4.6 Concentric Polygon Node-Line Configuration.

internal segments would fit inside those around them. Therefore, segmenting the node-line would vastly reduce the effectiveness of the topographic analysis by introducing complications which would severely reduce that algorithm's efficiency.

4.2.5 THE INADEQUACIES OF PREDETERMINED CONFIGURATIONS

The main advantage of placing components so that the node-line assumes a predetermined shape is the ease with which the routing requirements are calculated by comparing the permutation and the placed node-line. In addition, the major placement decisions are taken when the configuration is decided as each component's location relative to the predecessor is implicitly specified and only fine adjustments for track areas and clearances are necessary. This suggests that such a scheme would be fast and could therefore process several node-lines in a short period and present the results for human selection of the optimum layout (the different node-lines could be obtained with no extra effort as described earlier in section 4.2.0). Despite these advantages, the schemes considered have all been rejected on one shortcoming or another, the major failure being the inability to adequately cope with components of different sizes. A predetermined node-line configuration is considered to be too restrictive and constraining to allow the degree of flexibility necessary to deal with a general circuit; an alternative method is presented below.

4.3 GENERAL PLACEMENT

Two major points have arisen from the above discussions of placement; first, the components must be positioned sequentially rather than simultaneously; secondly the placement algorithm must have a high degree of flexibility so that it can handle components with a wide range of dimensions. Therefore, each component should be placed purely on the basis of the positions assigned to its predecessors and the permutation description. Ideally, the placement algorithm should be capable of treating each component in a similar fashion so that the same coding is used for each placement, resulting in a maximally efficient algorithm and hence program. In addition, there must be as wide a range of choices at each stage to confer maximum flexibility. At the same time it is important to maintain the node-line simplicity to preserve the connection pattern and to avoid a convoluted shape which would produce a large wiring length. These considerations correspond to the order of priority presented in section 3.1.2.

First the area must be minimised; this is accomplished by packing the components as tightly as possible while allowing space for interconnection routing. Secondly, the crossovers must be minimised; this has already been done topographically and the placement is only required to preserve node-line simplicity to achieve a minimal crossing connection pattern. Finally, the total connection length must be minimised, which corresponds to minimising the major bends in the node-line so that, for instance, paths are not required to travel around three sides of a rectangle. Thus the criteria of optimality may be restated, for the placement procedure, as follows:

(i) pack components tightly

(ii) maintain node-line as a simple curve

(iii) minimise major node-line folds

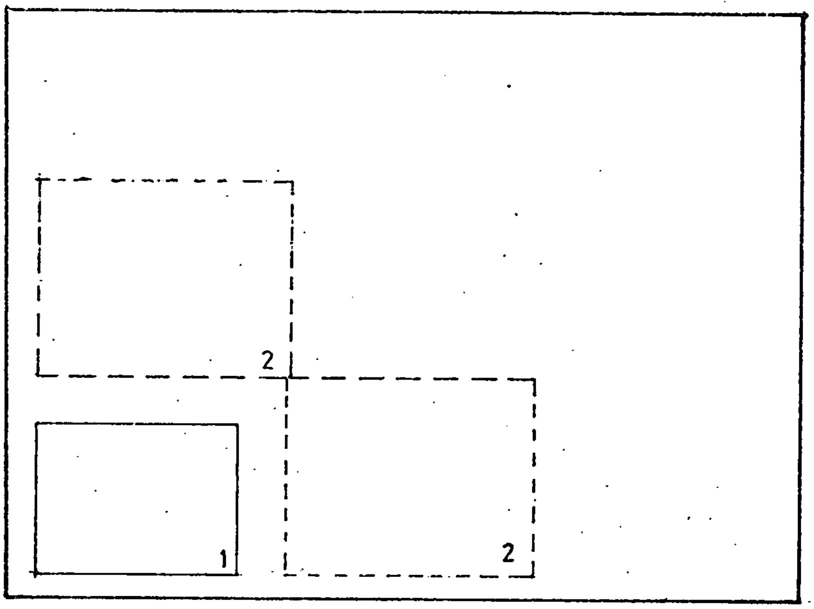
There are additional placement considerations which are concerned more with localised features; specifically space reserved for connections should be necessary and sufficient. Moreover the component placement should be in agreement with the design rules of the appropriate technology. The former point is a reiteration of the intention to dissociate the processes of placement and routing so as to produce a solution within a timesharing environment. The latter comment implies that all clearances and widths should not be less than the minimum dimensions specified; this can be a complex requirement, especially when dealing with integrated circuits described by the patterns of several masks, each of which has its own dimensional limits. A sample of MOS design rules is included as appendix II to illustrate this point. Only the most important rules which are generally applicable should be considered to avoid spurious precision as the program is only intended for trial layouts.

4.4 THE PLACEMENT ALGORITHM

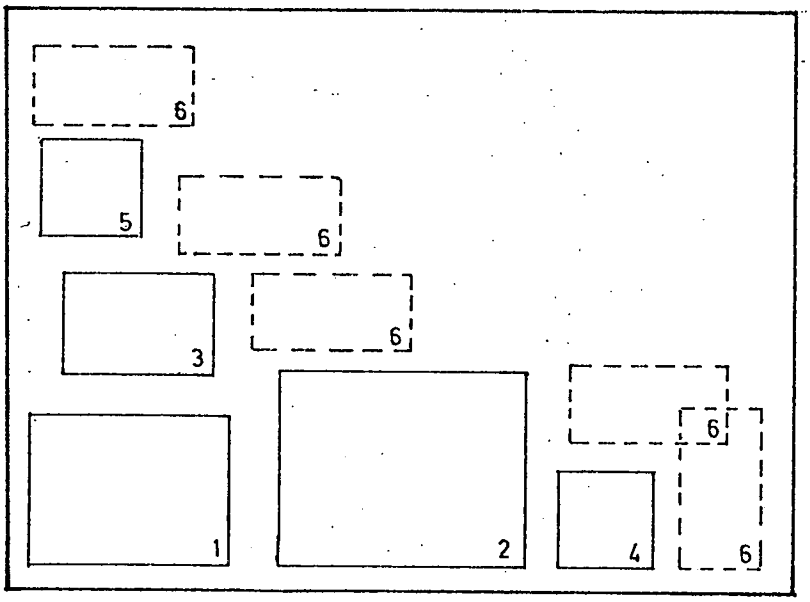
The remaining course of action in developing a suitable method of positioning components is to adapt an heuristic approach. The algorithm must be fairly straightforward to allow execution within the limitations of a timesharing system. A further discussion of the component placement aims and ideas is presented below, from which a suitable heuristic algorithm is developed.

Sequential placement implies that there is a first element to be placed which, at the moment of its positioning, may appear anywhere in the circuit area. Subsequent components may appear anywhere except in positions which include areas occupied by previously placed components. To allow maximum flexibility it is necessary to minimise the area occupied by elements and reserved spaces at all stages., thus providing a maximum choice for subsequent components. Elements should therefore be positioned as close as possible to their predecessors; in addition they should be positioned as close as possible to their immediate predecessor to avoid a complex node-line. This should also minimise the length of the transformed node-line and consequently the total connection length.

The starting point for the placement was conventionally chosen as the lower left hand corner of the substrate area, mainly to confine the number of directions in which the placement could expand and therefore increase the probability of producing a simple node-line curve. The component sequence was assumed to be that given by the permutation node order since it is that which defines the node-line. There are two available adjacent locations for the second component, above or to the right, as shown in figure 4.7(a) by the dashed outlines. This can be generalised for all subsequent placements as illustrated in figure 4.7(b). Therefore, the first feature of positioning any component, other than the first, is that it must appear above or alongside one of its predecessors; in this way the placement spreads from one corner as components are



(a) 2nd component.



(b) nth component (n= 6).

Figure 4.7 Compent Placement Positions.
(alternatives shown dashed)

added around the perimeter of the occupied area.

Another feature of this method is that, at every stage, the substrate area will be divided into two simple regions, occupied and vacant, which contain the unavailable and available locations respectively. Every component must be located within the vacant area, thus extending the occupied area by converting a section of the unoccupied region. It is therefore necessary to compute the available region at each placement, which infers the problem of internal layout representation; in other words, if a discretely shrinking area has to be identified after each contraction, how much storage space or time should be assigned for the computation? There is the usual trade-off between size and speed since it is generally true that the more storage reserved for containing data, the less time required for retrieval and processing.

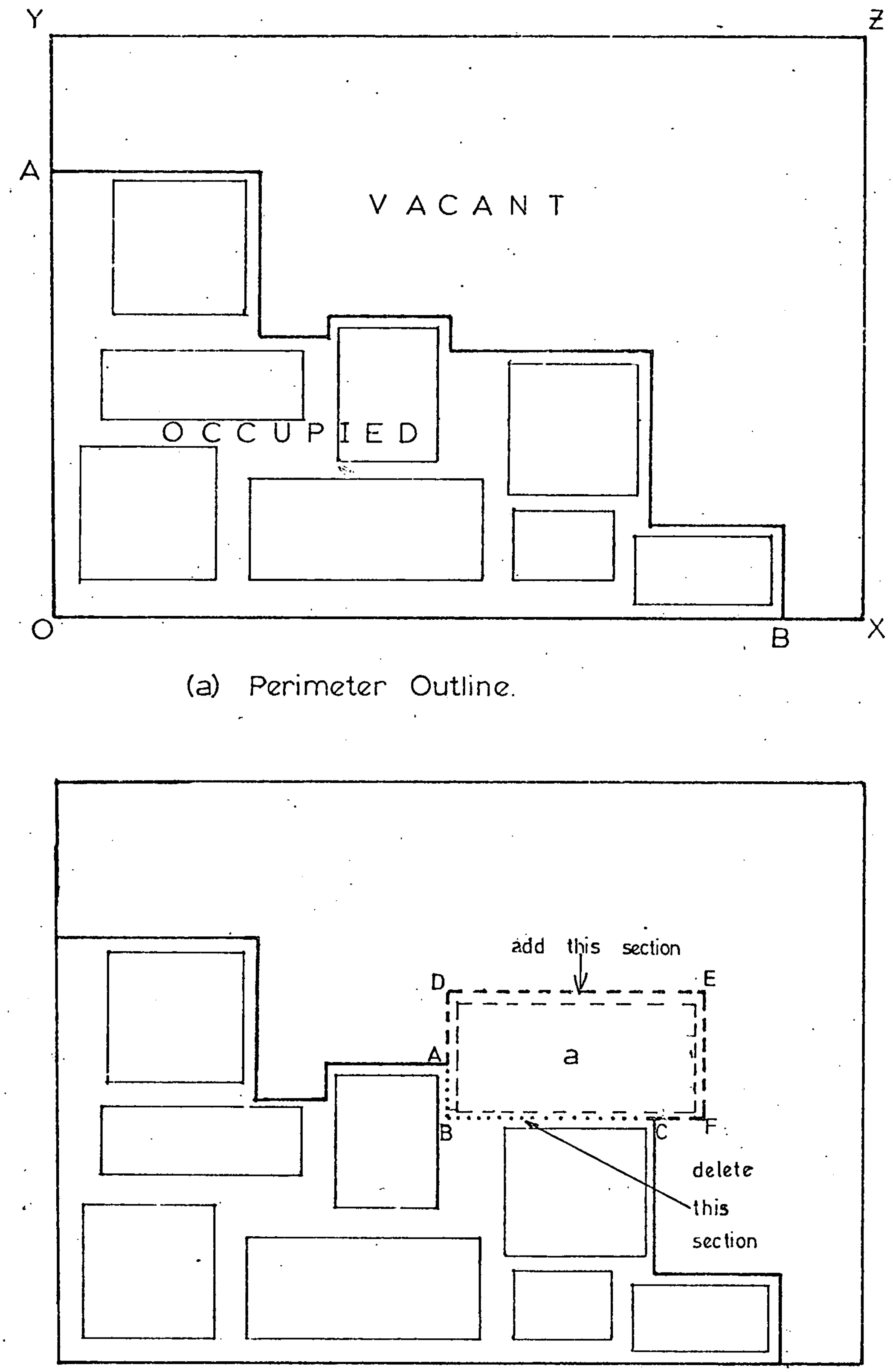
Basically, there are two techniques for storing area descriptions in a machine; these may be referred to as discrete and continuous description. Discrete description requires a grid to be superimposed on the area and each grid cell is described by one word of an array which is assigned flags to indicate the current stage of that cell on the substrate. A grid cell is either wholly occupied or wholly vacant; there is no means of half occupying a grid cell other than increasing the grid resolution, which increases storage requirements by the square of the factor of increase. The grid resolution therefore defines the maximum precision of the layout description. A minimum grid resolution for layout purposes would be 100 squares per side, requiring 10K words for the description, which would leave less than 4K for the program coding and additional data on a typical timesharing bureau. A discrete description is therefore out of the question. Continuous description consists of associating a set of parameters with each element which uniquely identify its position; the occupied area can then be calculated from the parameters assigned to placed components. This would involve a lengthy computation for each placement. It is better to reserve some storage for containing a brief description of the available area which can readily be modified as components are added.

As the placement proceeds there will always be a line separating the occupied and vacant area; this is illustrated in figure 4.8(a). The occupied area is bounded by the path OABO and the vacant by AYZXBA, where OXZY is the substrate area. This suggests that the best description would be in terms of the path from A to B; this is called the placement perimeter and consists of a series of rectilinear line segments. As each component is added, it must appear next to the placement perimeter which can then be easily modified to include the area occupied by that component. Modification is accomplished by altering the appropriate perimeter section as illustrated in figure 4.8(b); the addition of component a requires the section ABC to be replaced by ADEFC.

Component placement, viewed in terms of the placement perimeter, consists of positioning the component adjacent to the perimeter; from figures 4.7(b) and 4.8(a), it can be seen that this is in terms of vertical and horizontal perimeter segments. Inter-component spatial requirements may therefore be calculated as an X-displacement from vertical segments and a Y-displacement from horizontal segments. The placement decisions can be reduced to selecting the segments alongside which to position the component. This selection process is the appropriate place to include the criteria of optimality, that is, the segments should be selected so as to optimise the layout.

The layout compactness is a function of the components' coordinates (taken at the component's centre); smaller values imply a more compact layout since the placement starts at (O,O). A component placed at (x,y) will optimise the layout if (x+y) is the minimum available coordinate sum, or more generally circuit area will be minimised if components are positioned so as to minimise a function (ax+by) where a and b are constants. It would be more accurate to use Cartesian rather than Manhatten distances, that is, (ax^2+by^2), however the above expression is considered adequate for these purposes. It can be seen that lines of equal placement potential will belong to the family $y=k - (\frac{a}{b})x$, where k is a measure of potential, as shown in figure 4.9, and some control of the layout shape is available by manipulating the values of a and b since the placement perimeter should, ideally, approximate a line of equi-potential.

Control of the node-line is a more complicated matter since it is not a simple effect of coordinates, nevertheless it can be said thatthe closer two permutation-adjacent components appear, the less complex the resulting node-line path between them, and hence the greater the probability of



(a) Perimeter Outline.

(b) Component Addition.

Figure 4.8 Placement Perimeter :– Definition & Modification.

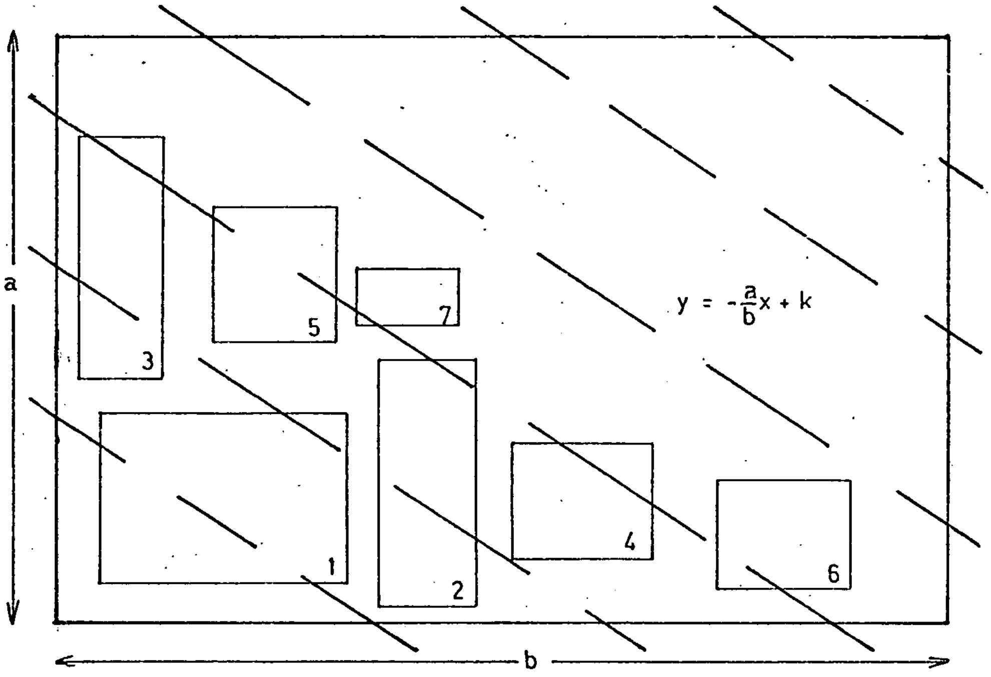


Figure 4.9 Equipotential Placement Lines.

Component ordering shows placement preferences.

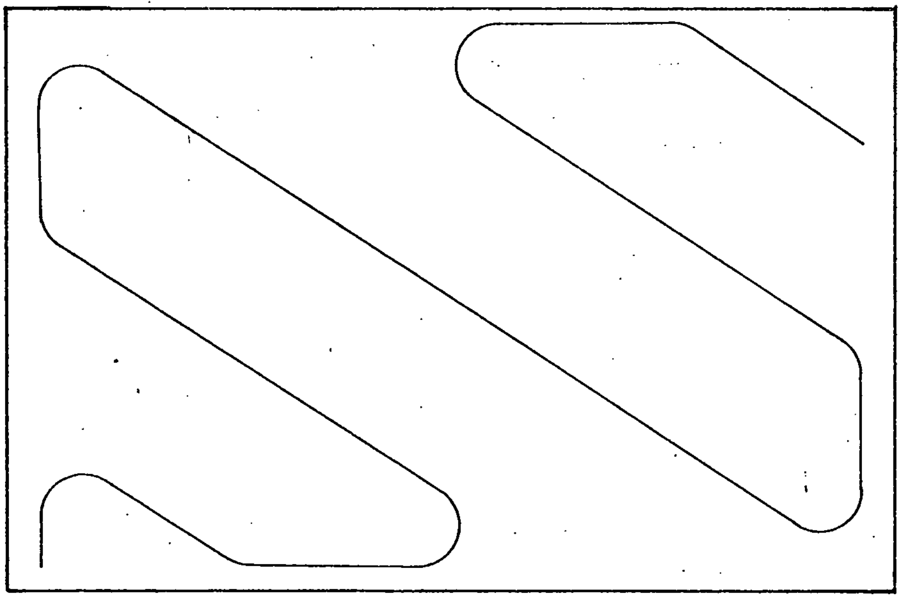


Figure 4.10 Node Line Configuration based on Equipotential Lines.

a compact node-line curve. A topologically simple curve is always possible since a simple path connecting any two consecutive nodes can be drawn along the placement perimeter, however to avoid a long and tortuous path the intervening distances should be as short as possible. In an ideal situation, it is expected that the node-line would assume a diagonally folded configuration following lines of equipotential such as that shown in figure 4.10.

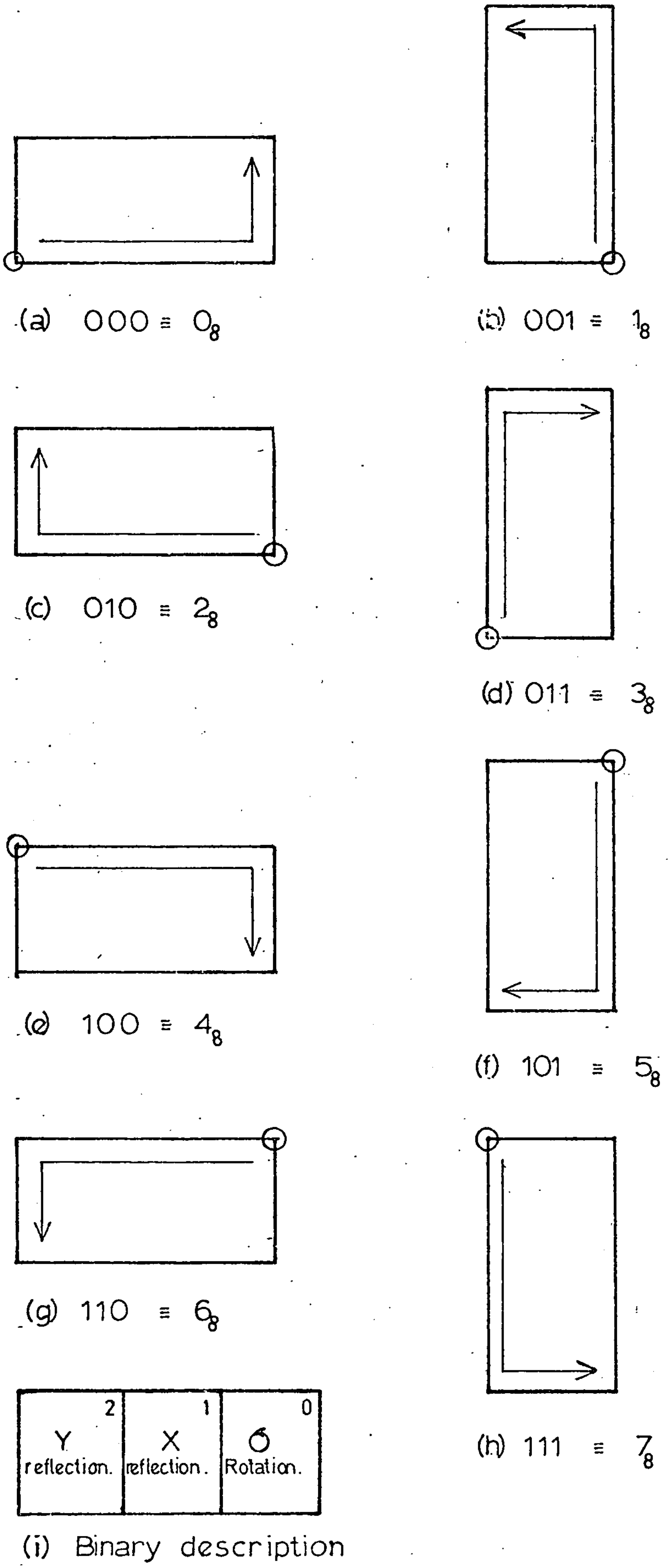
As these optimising controls can only be fully tested through use, it was decided to program an algorithm based on the above discussion and examine its performance through application to test circuits.

4.5.0 IMPLEMENTATION - DATA REQUIREMENTS

The initial concern with implementing any algorithm is data input and storage. The permutation is supplied, via a binary file, as the data structure built up by the program described in chapter 3. In addition, there will be a set of parameters, required by the placement algorithm, associated with each component which include dimensions and a classification; the classification is intended to enable the program to differentiate between types of components such as terminal pads, net nodes and transistors, should this prove advantageous. During placement, each component will be assigned a set of parameters which will identify its position and orientation; the position is given by the coordinates of the component's centre, the orientation by an integer between 0 and 7 inclusive. There are 8 possible orientations of a rectangle which may be uniquely obtained by the application of 3 binary operators, rotation clockwise through 90°, reflection in the x axis and reflection in the y axis. These are described by a 3 bit word segment as shown in figure 4.11(a) to (i). It should be noted that the reflection operations are only applicable individually to integrated circuits and not discrete components, since it corresponds to mounting components upside down or on the wrong side of the substrate. In addition, they do not affect the component's outline, only the positions of the terminal pins.

Link information will be required when a component is placed, to facilitate calculation of routing requirements; this information is calculated for all components prior to placement, as six quantities which are packed into two words, as shown in figure 4.12. It would also be advantageous to know where the connections are incident upon the component outline, however the exact locations are not necessary since the program deals only with rectangular representations, not precise descriptions, and is intended to provide trial layouts only. It was therefore decided that it would be sufficient to describe which faces the connections originate at, and their order on each face. To do this it is necessary to list the connections in anti-clockwise order as they appear in the 000 orientation and to add a 4 figure number describing the number of connections per face, taken in the same order. The complete component input description as implemented is shown in figure 4.13.

Additional data required by the program, common to all components, are the substrate dimensions (or upper limits thereon), minimum track width and minimum clearance for insulation requirements between electrically isolated



(a) 000 ≡ 0_8

(b) 001 ≡ 1_8

(c) 010 ≡ 2_8

(d) 011 ≡ 3_8

(e) 100 ≡ 4_8

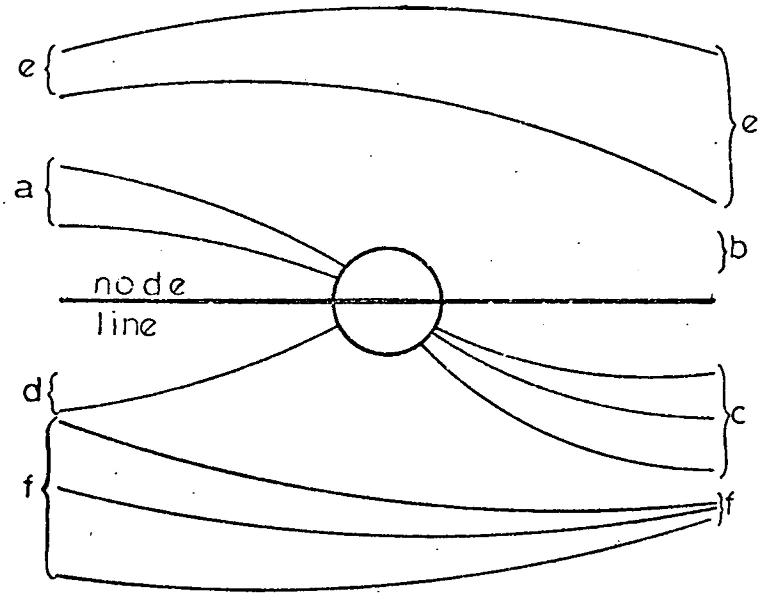
(f) 101 ≡ 5_8

(g) 110 ≡ 6_8

(h) 111 ≡ 7_8

(i) Binary description

Figure 4.11 Component Orientation.



(a) Linkage Pattern.

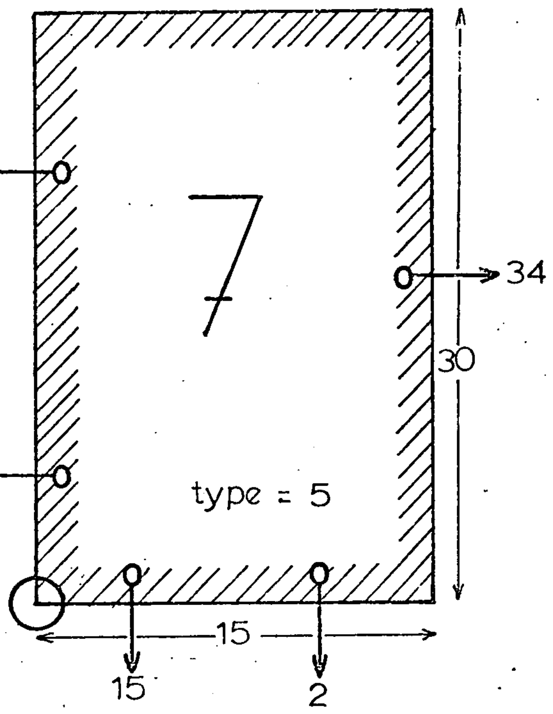
<table>
<tr><td>12
←
above
(a)</td><td>11 8
→
above
(b)</td><td>7 4
→
below
(c)</td><td>3 0
←
below
(d)</td></tr>
<tr><td colspan="2">Passing
above
(e)</td><td colspan="2">Passing
below
(f)</td></tr>
</table>

(b) Packing Format

<table>
<tr><td>2</td><td>0</td><td>3</td><td>1</td></tr>
<tr><td colspan="2">2</td><td colspan="2">3</td></tr>
</table>

(c) Description of above Pattern

Figure 4.12 Linkage Connection Pattern Description.



number dimensions type pads/face ordered connection list
7, 15,30 , 5,2102, 15,2 ,34 ,22, 4

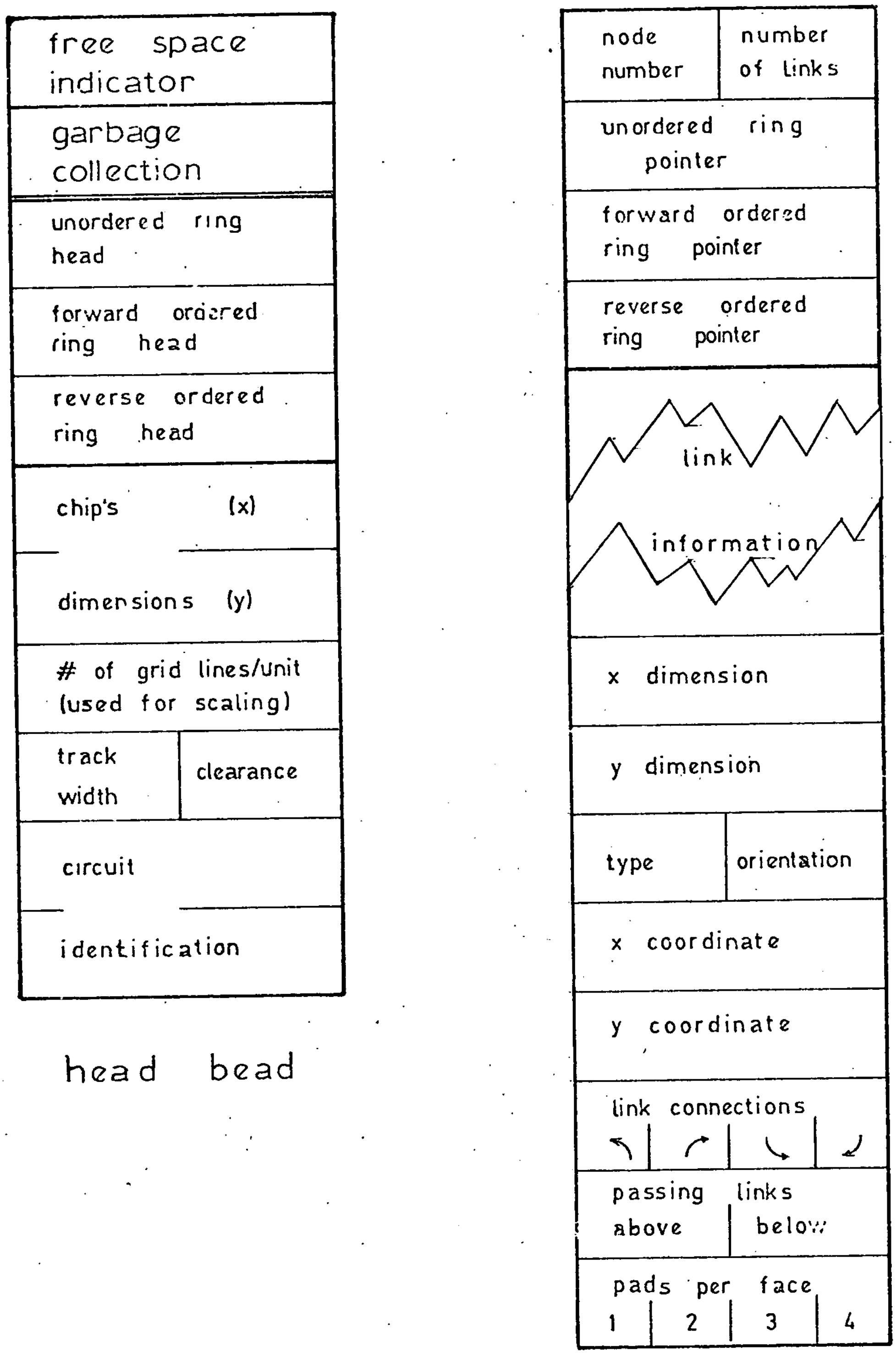
Figure 4.13 Input Component Description.

circuit elements. This information is requested, via teletype, after the circuit input description has been read and approved by the input stage of the permutation program. All component dimensions are in units of grid lines, substrate dimensions can be in any units and so the program will also request the number of grid lines per unit of the substrate dimension. These common data are appended to the head bead of the data structure and the individual parameters are included in the tail of the appropriate component beads. The resulting bead formats are shown in figure 4.14. The storage requirement for the circuit description is therefore (10+12N+2M) words which, as shown in figure 3.18, is still a considerable saving on the array-type storage used by Nicholson.

4.5.1 PERIMETER CALCULATIONS

The placement perimeter is stored as a list of the coordinates of the corner points commencing at the left hand end. The first component is placed in the bottom left hand corner, initialising the perimeter, subsequent placements are based primarily on the perimeter outline and the location of the component's predecessor.

As seen in figures 4.7 and 4.8, the positions considered best for component placement are in the vicinity of a concave corner of the perimeter (viewed from the occupied area's side), however there are complications which will arise, such as those indicated in figure 4.15. Any decisions based on the assumption that the perimeter outline is similar to the cross-section of a stair will be invalid for the concave corners a,b,c,d, and e, where the available



head bead

component bead

Figure 4.14 Finalized Data Structure :- Bead Formats.

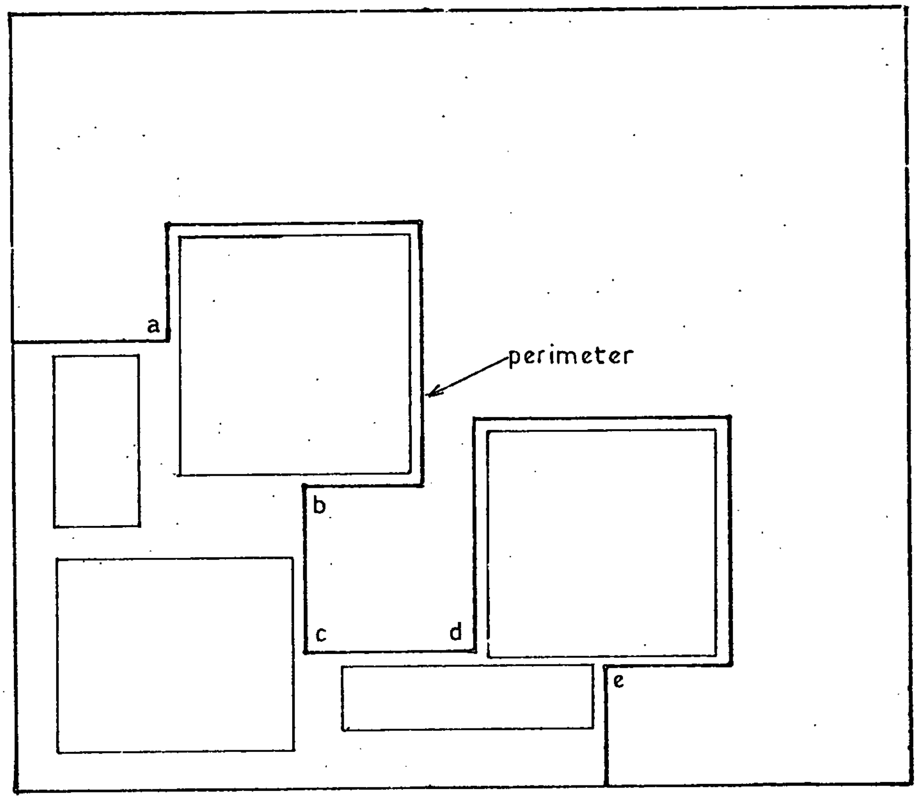


Figure 4.15 Constrictions which arise during Placement.

areas are closely bounded on more than two sides by placed components or the substrate perimeter. Therefore, there is more to placement than selecting the optimum segment, each selected segment must be checked for constrictions in all directions.

The placement decisions are based on selection of individual perimeter segments; since the vertical segments are between two horizontal segments it is only necessary to consider one class as this will automatically include consideration of consecutive members of the other type. The program will therefore confine segment selection to vertical perimeter line segments to maximise efficiency. Vertical segments are those which occur between consecutive corners with the same abscissa; they are initially classified as one of four types according to the adjacent horizontal segments' directions. The classifications are identified by the integers 1 to 4, obtained as follows :

1 both segments to left

2 first to left, second to right

3 first to right, second to left

4 both to right

In addition, the integers are given a sign depending on which direction the vertical section runs (positive if top to bottom); the resulting eight segment classifications are illustrated in figure 4.16.

Figure 4.17 shows a sample perimeter containing each segment type; it can easily be verified that the occupied area of the substrate is invariably to the right of the perimeter (facing in the direction of the perimeter description).

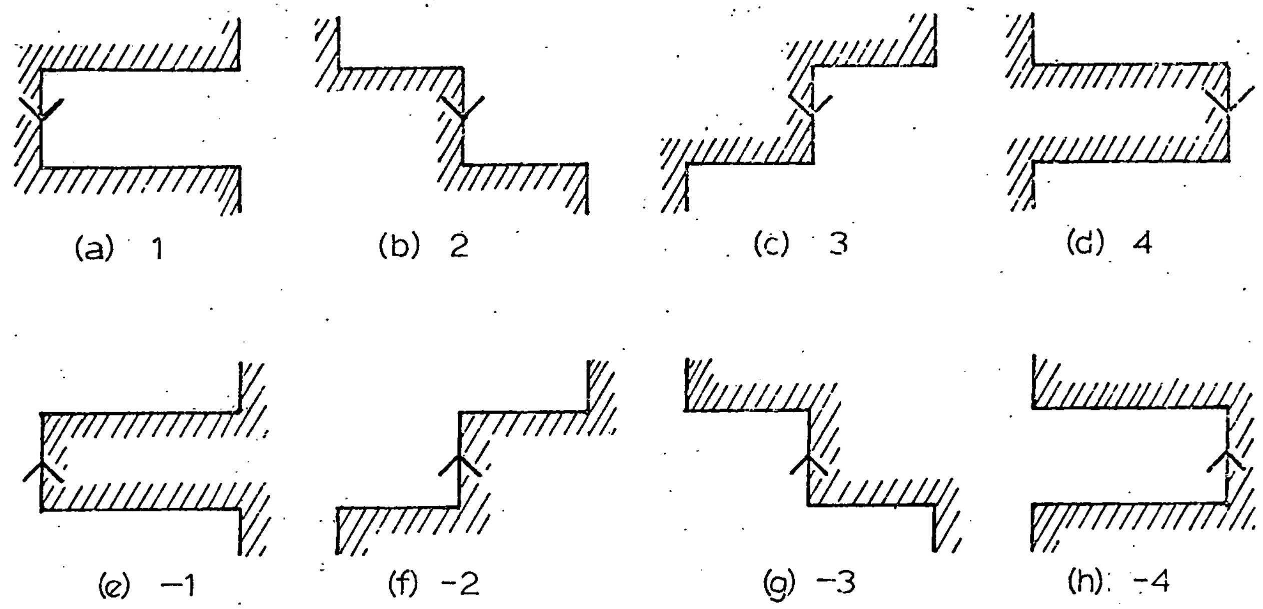


Figure 4.16 Placement Perimeter Segment Types

arrow indicates perimeter direction
shading indicates occupied side of perimeter.

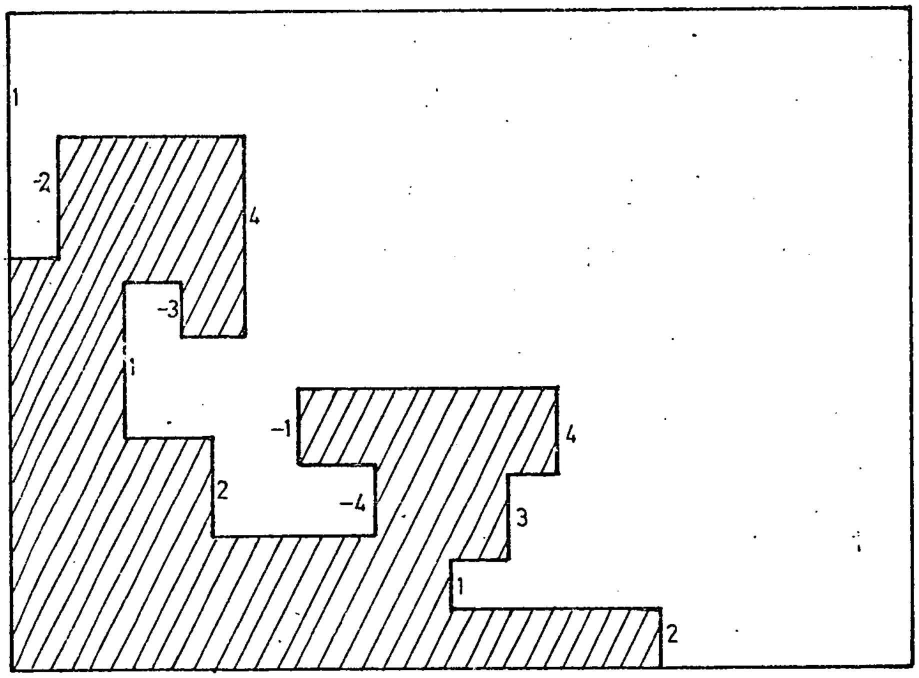


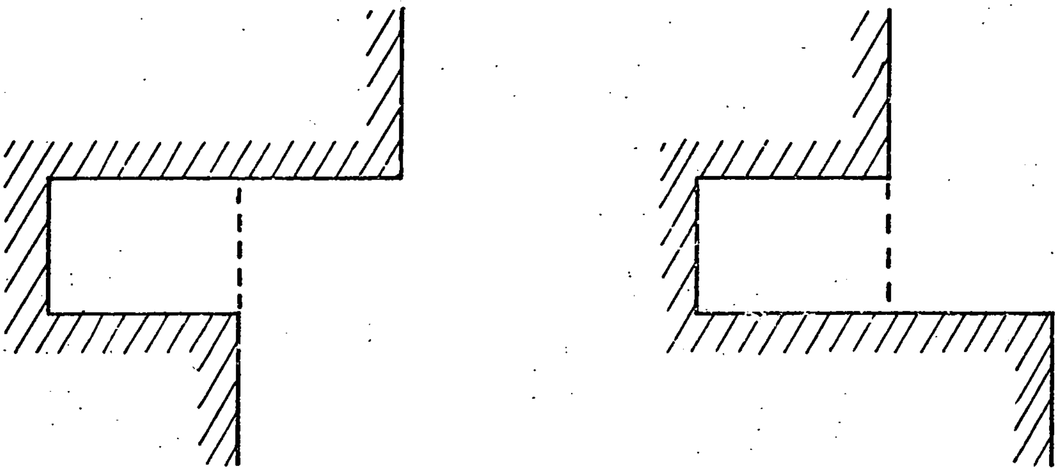
Figure 4.17 Sample Perimeter containing all Segment Types.

Another significant feature is that the negative segments always appear further from the origin (bottom left hand corner of the substrate area) than a positive segment (providing the upper section of the left hand substrate edge is taken as a type 1 perimeter segment). This means that for every negative segment there will always be a positive segment, which will offer a more compact placement since it has a smaller abscissa; therefore segment selection may be confined to the positive segments only.

4.5.2 SEGMENT SELECTION FOR PLACEMENT

The analysis of available locations for component placement is accomplished by compiling a list of the vertical segments obtained from scanning the perimeter description. The list contains data describing each segment's length and location; all negative segments are excluded from the list since they are considered redundant. This list is scanned for type 1 segments whose lengths are smaller than the current component's smaller dimension (the smaller side of the representative rectangle). It can be seen that, since a type 1 segment is bounded top and bottom by concave corners, it is the only type that will not accept a longer component. Any such segments found are amalgamated with a consecutive segment with a type change, as shown in figure 4.18 by the dashed line. This is done to avoid further consideration of unusable areas. At this stage the segment list contains all the feasible alternatives and the next step is to select the optimum location in terms of segments.

Circuit area can be minimised by minimising a linear function of the components' coordinates, however since only



(a) 1,2→1 or 1,4→3

(b) 4,1→2 or 3,1→1

Figure 4.18 Segment Amalgamation.

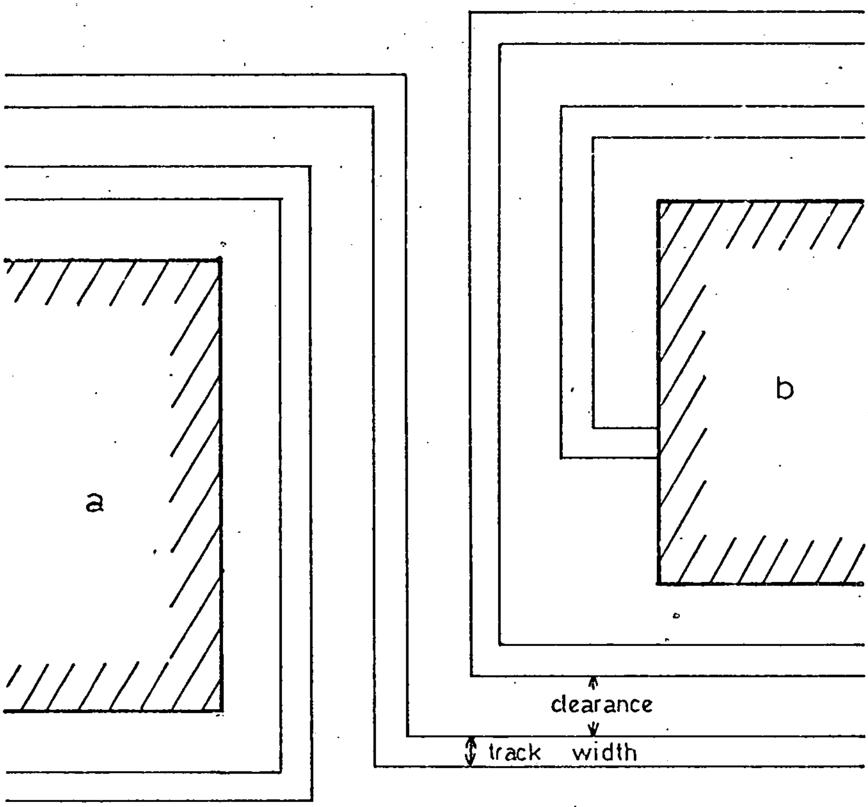


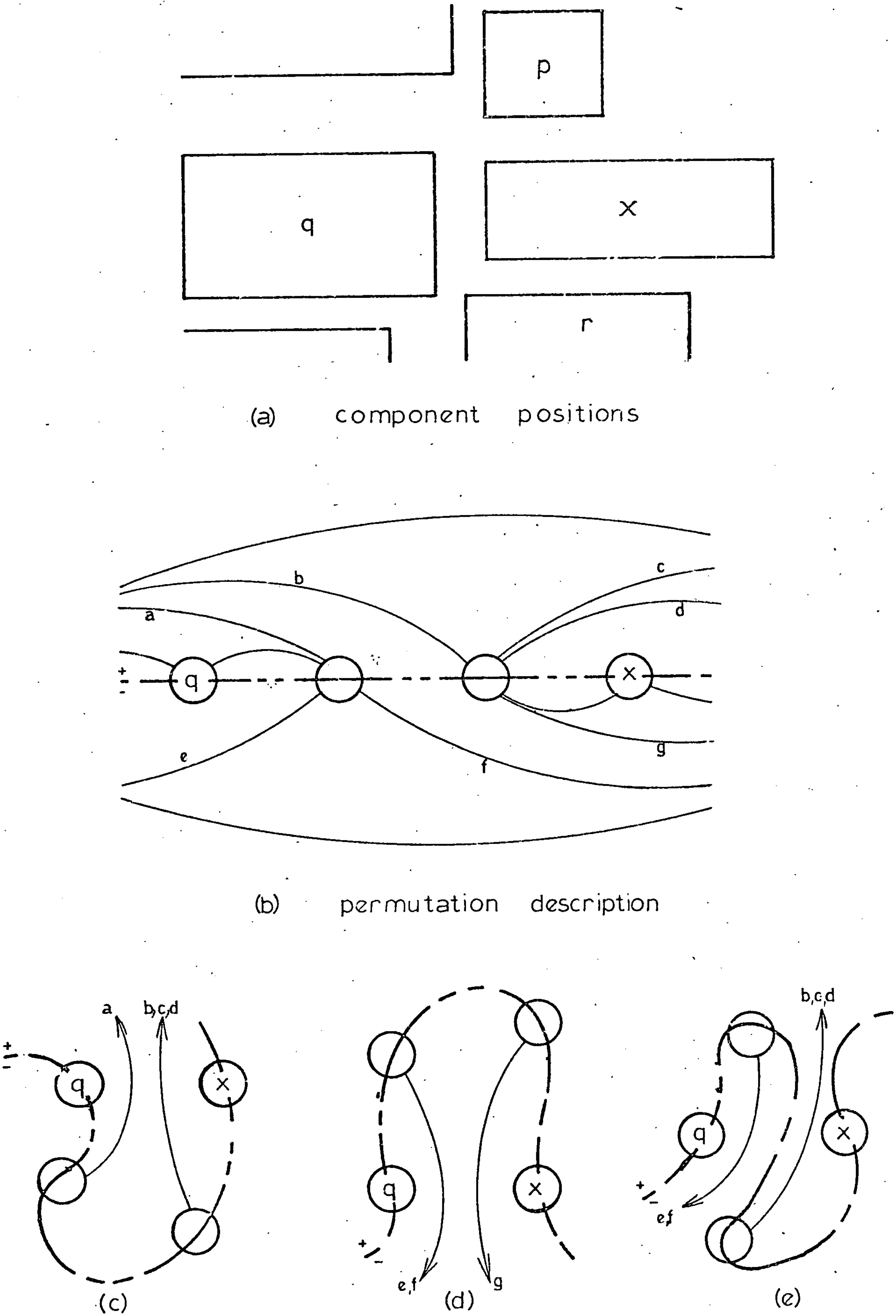
Figure 4.19 Component Separation.

segments are currently being considered, this may be accomplished by minimising the function using the centre point of the available segments, that is, selecting the segment whose central location has the lowest placement potential. Optimisation of the node-line is not so simple, however it can best be accomplished by minimising the distance between components whose nodes are adjacent on the permutation node-line. It would be too severe a constraint to insist that consecutive components were placed side by side since the resulting layouts would probably be linear and compactness would consequently suffer; the selection process must have balanced optimisation aims to avoid results which are good from one point of view but unacceptable from other viewpoints.

To facilitate segment selection weighted preference values are assigned on the basis of coordinate values and proximity to the predecessor (calculated on a straight line basis); the segment showing the highest overall preference is that selected for trial placement. The necessary clearances, for tracks and insulation, are calculated on the basis of the linkage connection pattern data compiled prior to placement (shown in figure 4.12). Each segment type will require an X-displacement for the separation between adjacent components on either side of that segment. In addition, there will be vertical displacements at concave corners; these occur at the top of types 1 and 3 and at the bottom of types 1 and 2. All other corners are convex and are therefore considered to be 'open' in that the component may extend beyond the segment. If this trial placement fails due to overlap with the substrate edge or another component or insufficient room for connections and clearances, that segment is flagged as unsuitable and the next best segment is selected for trial placement. When all segments in the list have been selected and none found suitable, the program produces a message that the current component has not been placed and advances to the next component.

4.5.3 CALCULATION OF SEPARATIONS

The separation between a component and its positional neighbours is a function of the number of intervening tracks, as illustrated in figure 4.19. During a trial placement positional neighbours are located by a search through the previously placed components to locate those forming the perimeter segments adjacent to the trial location. Figure 4.20(a) shows a typical trial situation; component X is being placed and the separations between component pairs (p,x), (q,x) and (r,x) are needed to determine the feasibility of the trial location and, if suitable, the exact location to be assigned to component x. Components p and r will only be present for certain segment types, specifically 1,3 and 1,2 respectively. They have been included here for the purposes of generality; q will always be present. If q and x are permutatively consecutive then there will be no unconnected intervening tracks and separation will be a function of the number of connections emerging from the opposing faces of the components. Since the component's orientation is not finalised as yet, those faces which oppose cannot be determined, and so these connections will be ignored in preference to calculating for the worst case situation.



(a) component positions

(b) permutation description

node line configurations

Figure 4.20 Calcualating the number of Intervening Links for Separation.

Figure 4.20(b) shows a permutation section which may be considered representative of the case where components q and x are not consecutive nodes in the permutation; the relevant nodes and links are shown. To calculate the number of intervening paths, it is necessary to establish the configuration of the transformed node-line between q and x on the trial location, this is done by scanning the positions of the intervening components as previously assigned. It must also be noted that the node-line orientation is significant; for this reason the positive side is always assumed to be above if the node-line is considered as extending horizontally to the left of the first component placed. Figures 4.20(c), (d) and (e) illustrate some possible configurations which place q and x horizontally adjacent; the appropriate links passing between the components are shown and identified on the permutation in figure 4.20(b). In all cases those tracks which pass from an interspersed node to a node beyond either q or x on the concave side of a fold pass between q and x. Separations arising from vertical displacements are obtained in a similar fashion.

Returning to figure 4.19, it can be seen that the required separation is equal to Nx(track width + clearance), where N is the number of intervening links, plus an additional clearance between components. Therefore, the separation displacements can be calculated and the component's location finalised if there is no spatial conflict. This intervening track calculation is somewhat over-simplified by omission of relevant details, such as the emerging tracks mentioned above. Another complication arises when considering nets; if two intervening links belong to the same net then they should obviously be merged unless there is a third track between them. This is also being overlooked as it is considered to be too fine a point for consideration in relation to the aims and specifications for the program. It should be noted that the two above-mentioned omissions will tend to counterbalance each other and, therefore, it is anticipated that they will not cause too much trouble.

4.5.4 FINAL POSITIONING

Once the segment has been selected and the separations calculated, the coordinates of the component's centre and the orientation must be assigned. The coordinates are calculated from the segment data, displacements and component dimensions,however it is first necessary to finalise the component rotation so as to identify the horizontal and vertical placed dimensions. In placing the component, the reflection operations are ignored since they do not affect the outline, only the rotation need be considered; the component is placed so that there is a minimum difference between the segment's length and the component's vertical dimension plus separation, to avoid wasting space.

Once the placement parameters have been assigned, the data structure is updated and the perimeter modified, these steps constitute the actual positioning of the component. The placement algorithm, as described above, was programmed with the results presented below.

4.6.0 INITIAL RESULTS

A typical example of the automatic placements produced is shown in figure 4.21. It can easily be seen that, although the node-line is as simple and compact as could be expected, the layout, even without connections, is unacceptable. The main defect is that the components, after an initial tight packing, straggle across the substrate in two lines with a large vacant space between them; this fault was common to all the initial trials. The first corrective action was to use a placement potential function with a steep negative gradient, equivalent to constraining the placement abscissa. This was found to be ineffective or, when it did produce effects, too uncontrollable for reliability and so the algorithm was re-examined for oversights or omissions. Eventually it was decided that the segment selection procedure was lacking in depth and so this was extended. The modifications are described below and the results achieved from their implementation are presented in section 4.7.

4.6.1 SEGMENT SELECTION MODIFICATIONS

It was clear that additional selection criteria were necessary to present consistent use of segments on the right hand extremities of the layout, since the original basis for selection were inadequate. The first addition was to bias selection in favour of type 1 segments and against type 4 segments to prevent rapid horizontal expansion; if the components are placed against concave corners then the development of large gaps will be prevented since they

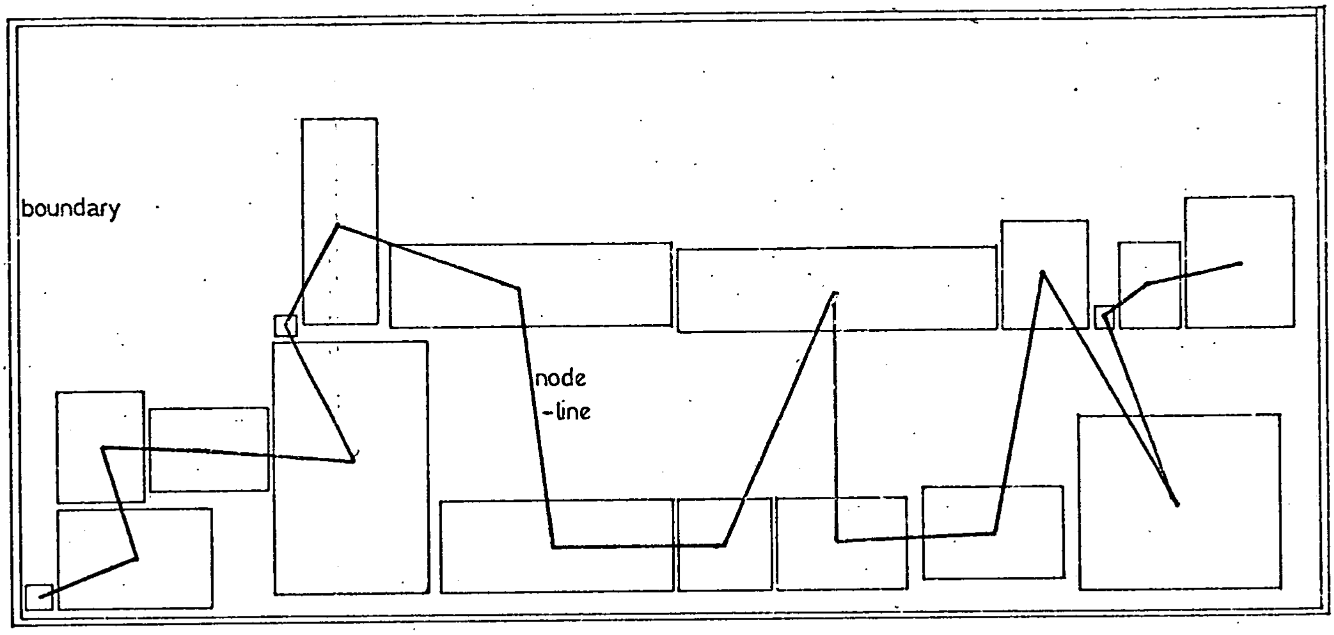


Figure 4.21 Typical result of Initial Placement Algorithm.

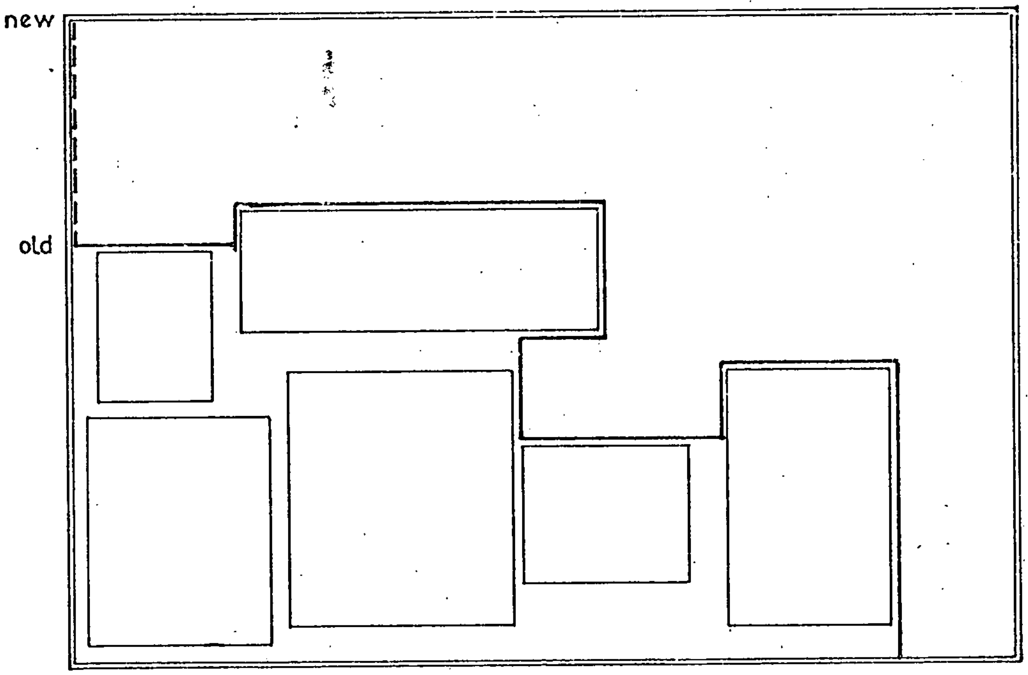


Figure 4.22 Perimeter Definition Modification.

will tend to be occupied as they develop. Initially, use of type 4 segments was prohibited, however this caused problems in locating large components which could not be placed in any of the gaps. The prohibition was consequently altered to an inhibition. Despite the improvements this caused, the layouts were still confined to the lower portion of the substrate. This deficiency was corrected by extending the perimeter definition to include the left hand edge of the substrate, as shown by the dotted line in figure 4.22, thus creating an additional type 1 segment with the minimum possible abscissa.

At this stage the layouts being produced were distributed adequately, but often with a lot of gaps caused by inefficient use of segments; components were being placed alongside segments with a large difference between the lengths, thus areas suitable for a large component were being occupied by a small component which then created an unusable gap. This deficiency was corrected by placing a preference on segments with lengths close to one or the other of the component's dimensions, which achieved an increase in packing density as desired. Since the algorithm was ineffective in choosing between a few segments it was found advantageous to place the first four components (rather than the first one only) alongside the ordinate axis to present the selection process with several segments at its first placement (the fifth component). These modifications were implemented and a typical result produced is shown in figure 4.23 which illustrates the vast improvement achieved in packing quality.

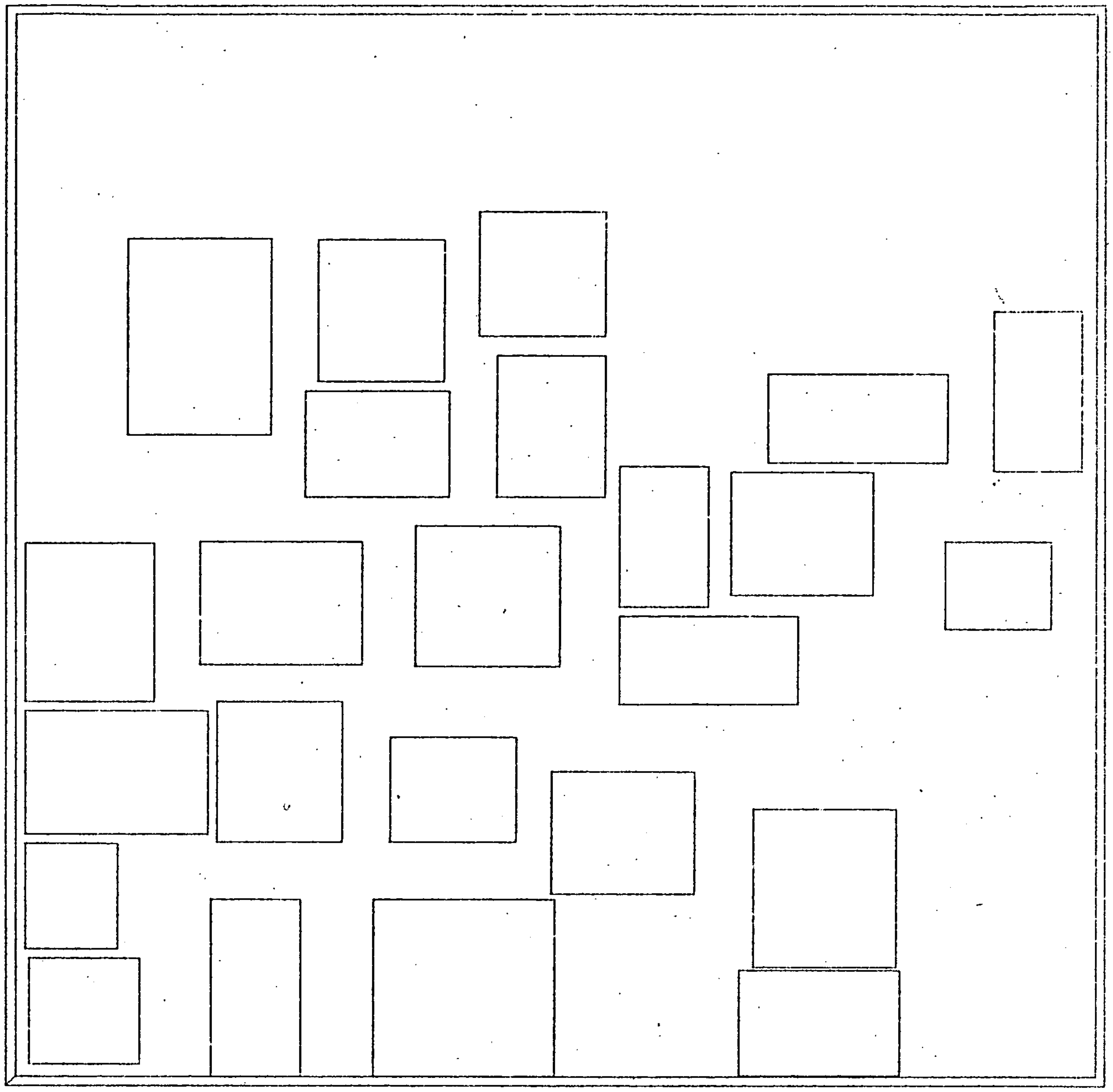


Figure 4.23 Typical Placement achieved with Modified Algorithm.

The segment selection process is now based on five parameters: abscissa, ordinate, segment type, proximity to predecessor and goodness of fit. With such a range it is necessary to provide some control over the selection process to allow the criteria variations in priority. Parameter weighting is the method used here and it is accomplished by supplying the program, conversationally, with a list of five integers which indicate the relative significance to be attached to each of the above parameters during selection. A boundary definition is also necessary to avoid unrealistic layout perimeters; this confines the available area during the latter stages of the layout but produces rectangular layouts as opposed to triangular ones. This is also of use when designing cells which must be of a definite shape but, within that shape area minimisation is not important, such as in integrated circuit cells.

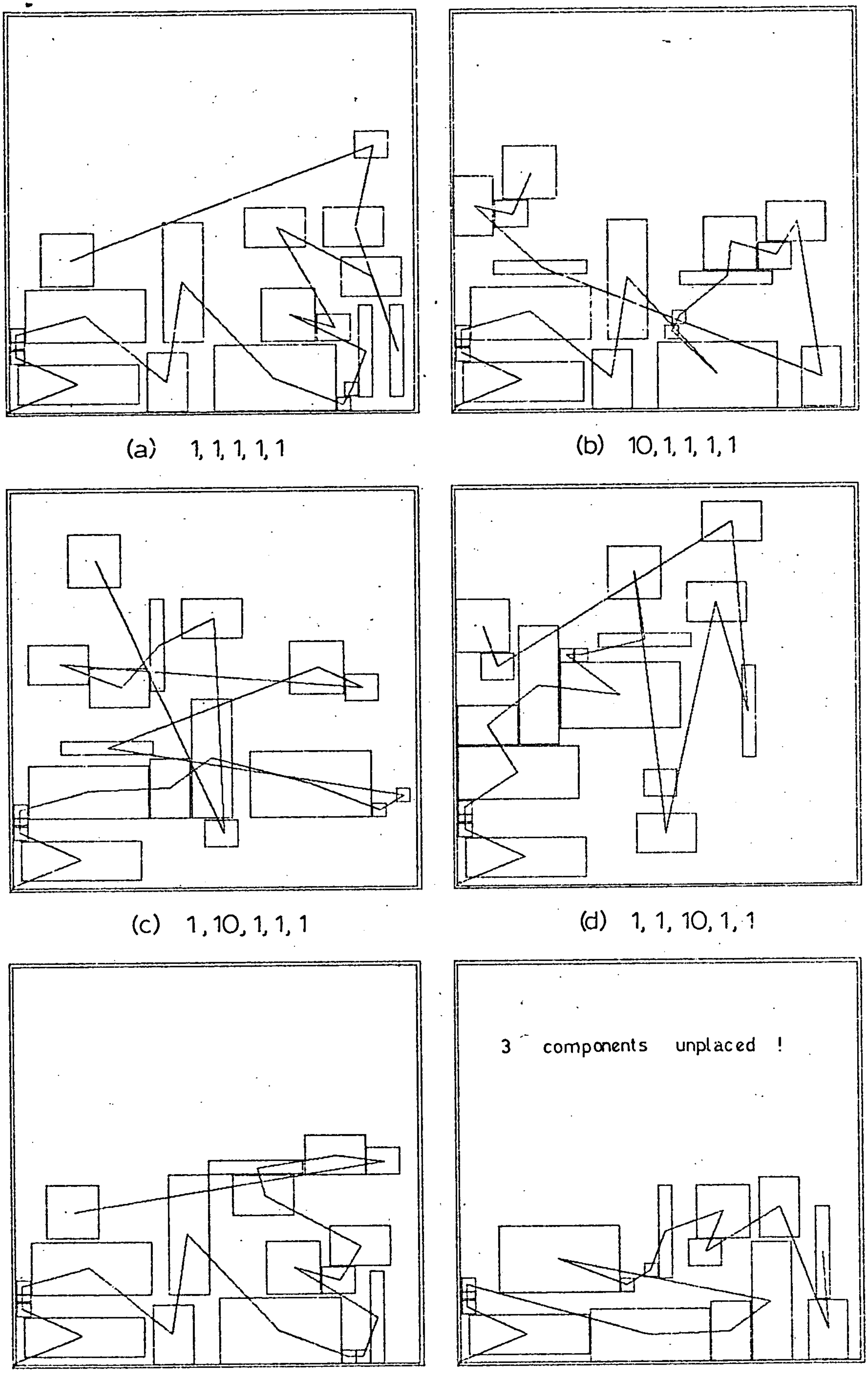
The final layout features considered were associated with terminal pads and net nodes. The topographic analysis cannot allow for peripheral placement requirements in that it cannot be constrained to prevent specific nodes from being surrounded by a closed circuit, therefore terminal nodes cannot always be placed on the circuit periphery. To prespecify component locations is also unsuitable in this instance as it would generally upset node-line considerations. Therefore the only action possible in placing components which are connected to the outside world,is to give preference to segments close to the substrate boundary. Net nodes, although they do not technically have a physical analogue, and therefore do not require consideration during placement, are treated as ordinary component representations. The main reason for this is so that a designer can readily locate nets in the layout by node rather than tracing connections. In addition, they are included to aid routing. The final section of this chapter consists of a discussion of the results produced by the modified algorithm.

4.7.0 RESULTS

The main interest in the results is associated with the effects of the selection parameters and the degree of control available through the parameter weights. It is not practical to include a comprehensive analysis here and so the results of each parameter applied to one circuit are shown. The circuit used is taken from a thin film, implantable avian temperature transmitter; the schematic and manual placements are shown in chapter 6. The results shown in figure 4.24 indicate the effects of weight values and, consequently, the control parameters. Figure 4.24(a) is the layout achieved with a uniform weighting while figures 4.24(b) to (f) are the results of assigning a relatively insignificant weight to four out of the five parameters. The weights used are shown under each placement.

4.7.1 EFFECTS OF CONTROL PARAMETERS

Specific effects are difficult to pinpoint in the placements produced, since the control parameters are not of primary significance. Component dimensions, connection pattern and perimeter shape have effects which cannot be estimated; for example a component might be ideally suited for a certain location, but cannot be placed there because there is insufficient space for routing requirements and is



(a) 1, 1, 1, 1, 1

(b) 10, 1, 1, 1, 1

(c) 1, 10, 1, 1, 1

(d) 1, 1, 10, 1, 1

(e) 1, 1, 1, 10, 1

(f) 1, 1, 1, 1, 10

Figure 4.24 Variations in Placement achieved with different Weights as shown.

therefore positioned elsewhere. Consequently the control parameters, and their weights, are ultimately of secondary significance in comparison to practicality; this is especially noticeable in placing the largest components of a circuit. Moreover, as the available area continually shrinks, the choice becomes more and more limited until the last components often have little or no choice.

Despite the above complications, some general tendencies have been observed, although they are not always present. Layout area is usually minimised by selecting type 1 segments in preference to others, rather than trying to minimise component coordinates. Node-line compactness also reduces layout size, probably because the spatial connection requirements are thus reduced. Coordinate minimisation is relatively ineffective in minimising layout area, although it does have a stabilising effect on the placement. Avoiding area wastage by using optimally sized segments is of little consequence since it is divorced from the criteria of optimality. Nevertheless, the parameters do have some effects and allow different layouts to be produced for a circuit, which is of great advantage as will be discussed later.

4.7.2 CONCLUSIONS

The placement algorithm cannot be fully assessed without connection routing since the layout quality is dependant on both placement and routing. However, the algorithm, as programmed, is very fast and takes an average of around .2 of a second to place each component; this implies that any qualitative deficiency could well be offset by the quantity of results available at a small cost.

CHAPTER V

ROUTING CONNECTIONS

5.0.0 INTRODUCTION

Routing the intercomponent connections constitutes the final step of the microcircuit layout process. The connections are (in this case) assumed to consist of metal tracks of uniform width running between the components and joining electrically common terminals on different components. The connection pattern is defined by the circuit input description (specifically the links thereof) and, consequently, the permutation produced by the topographic analysis which defines the minimal-crossing connection configuration. In addition, the component positions assigned by the placement algorithm define the origin and destination points of the tracks,and areas which they cannot enter (except at the start and finish of a track). Routing, in this system, may be viewed as a process of guiding links between the origins and destinations, avoiding areas occupied by other circuit elements, while conforming to the permutation link configuration to achieve a minimal number of crossovers.

5.0.1 OPTIMISATION

As has been discussed previously (sections 2.1, 3.2 and 4.4), each layout procedure requires criteria which enable the programs to optimise the appropriate features of the layout in keeping with the overall criteria of optimality. Routing can only optimise the layout by positioning tracks to maximum advantage. The specific areas where routing can directly optimise the layout are therefore crossover removal and minimising track lengths. In addition, the circuit area will be governed by peripheral tracks but this feature is considered to be far less significant than the above, because the relative increase in circuit area produced by a track around the perimeter would be negligible.

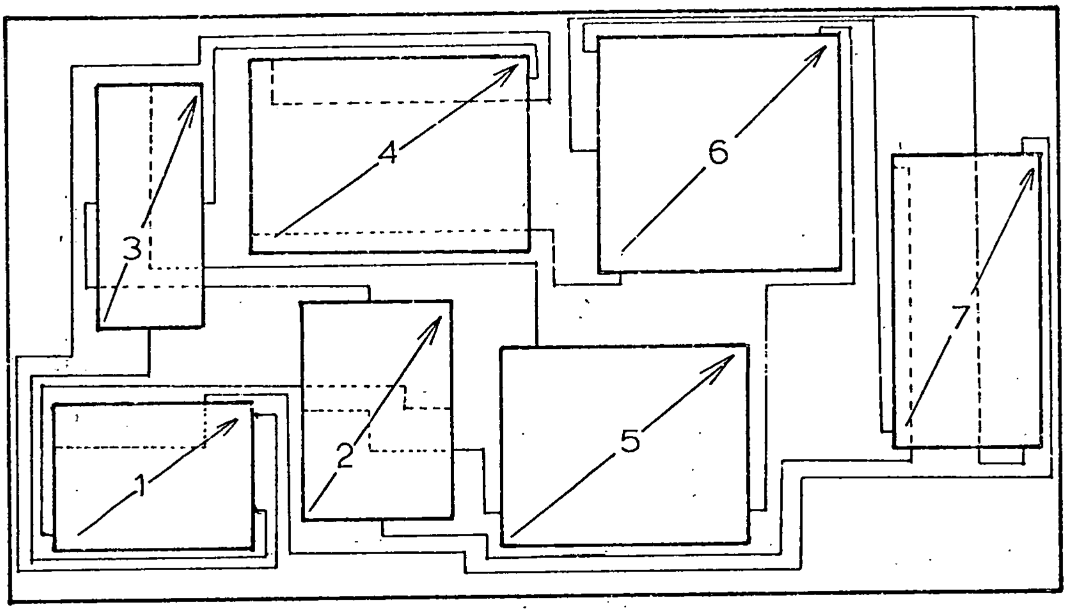
The system which has been proposed (chapter 3) and partially constructed (chapters 3 and 4) has provided a component placement with specific consideration of the routing requirements. This scheme, although designed to simplify the layout process, removes much freedom from the routing procedure. The placement reserves interstitial space for the connections as defined in the permutation description of the network. Therefore, if the reserved space is to be accurate, it is important that tracks appear in the particular locations which were reserved for them. Thus any of the routing algorithms discussed in chapter 2 would have to be constrained, to conform to the permutation configuration, if they were to be used here; such constraints would inevitably reduce any algorithm's efficiency to a point where it became largely ineffective. There is then little advantage to be obtained from applying a powerful routing algorithm in this case.

There are, however, alternative areas which afford scope for improving the connection pattern. Component orientation, apart from rotation, is not yet finalised and terminal pad positions can be changed by applying the reflection operations. Induced crossings can be removed through judicious orientation, and net lengths can also be minimised, as illustrated in figures 5.1 and 5.2 respectively. However, reflection is an operation with limited application since discrete components cannot be reflected and cannot therefore be fully exploited in a general program.

Other techniques for improving the wiring layout, such as reorientating the node-line to make best use of any folds (see figure 5.3), are inapplicable because they would upset the space reservations. Thus the only facility available for optimising during routing is to orient the components to greatest advantage by reflection alone.

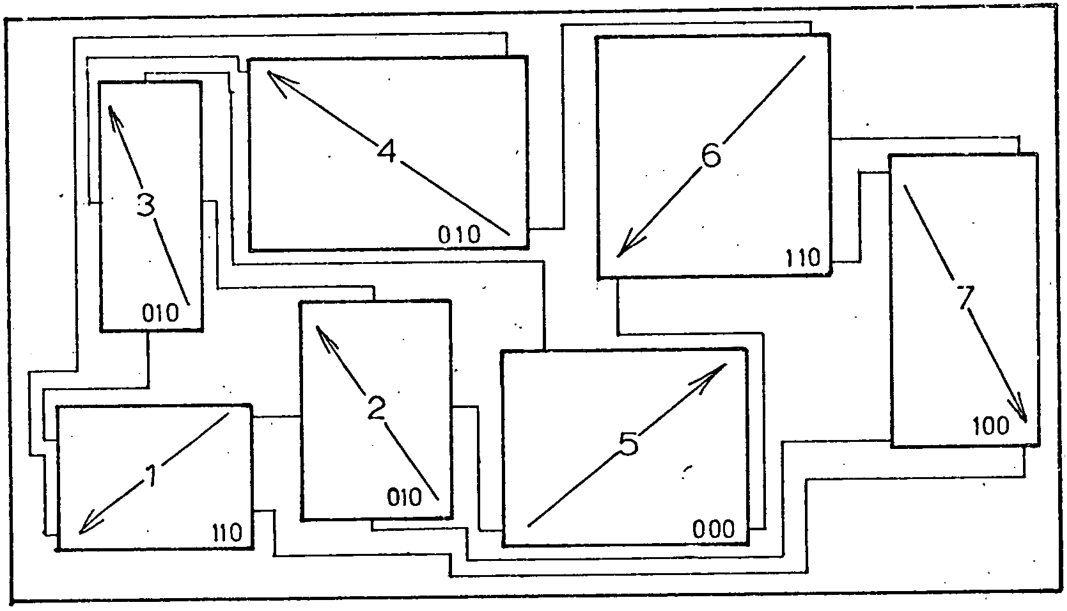
5.1.0 PROBLEMS OF ROUTING

The primary concern of routing is to eliminate crossovers and, in addition, to minimise track lengths. If the permutation is non-planar, then that configuration must be partially altered during routing. Each of the microcircuit technologies has at least one technique for incorporating crossings without producing a short circuit; these features are the only means whereby a routing algorithm can depart from the predefined connection routes in order to eliminate crossings. To include such considerations in an automatic algorithm would require either a separate program for each technology or a single program whose size exceeded the limitations of a timesharing bureau. Both of these alternatives were considered to be beyond the scope of this work and it was therefore decided that non-planar connections would be omitted and a list of the omissions provided with the layout so that these tracks could be inserted manually. This is not considered to be a serious omission in view of the fact that the aim of the system is to provide trial



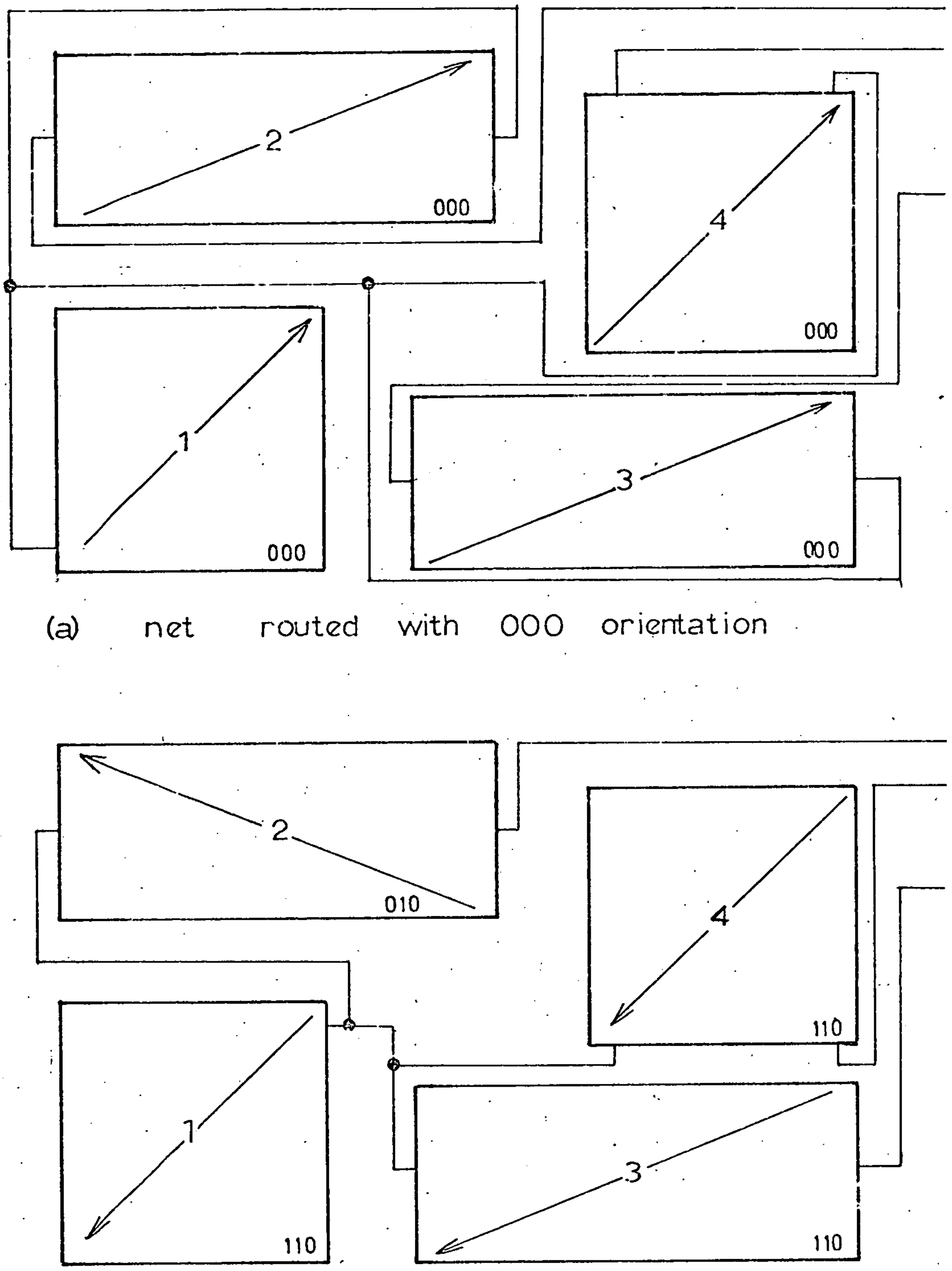
(a) tracks routed without reorientation.

(all component orientations = 000)



(b) components reflected for routing.

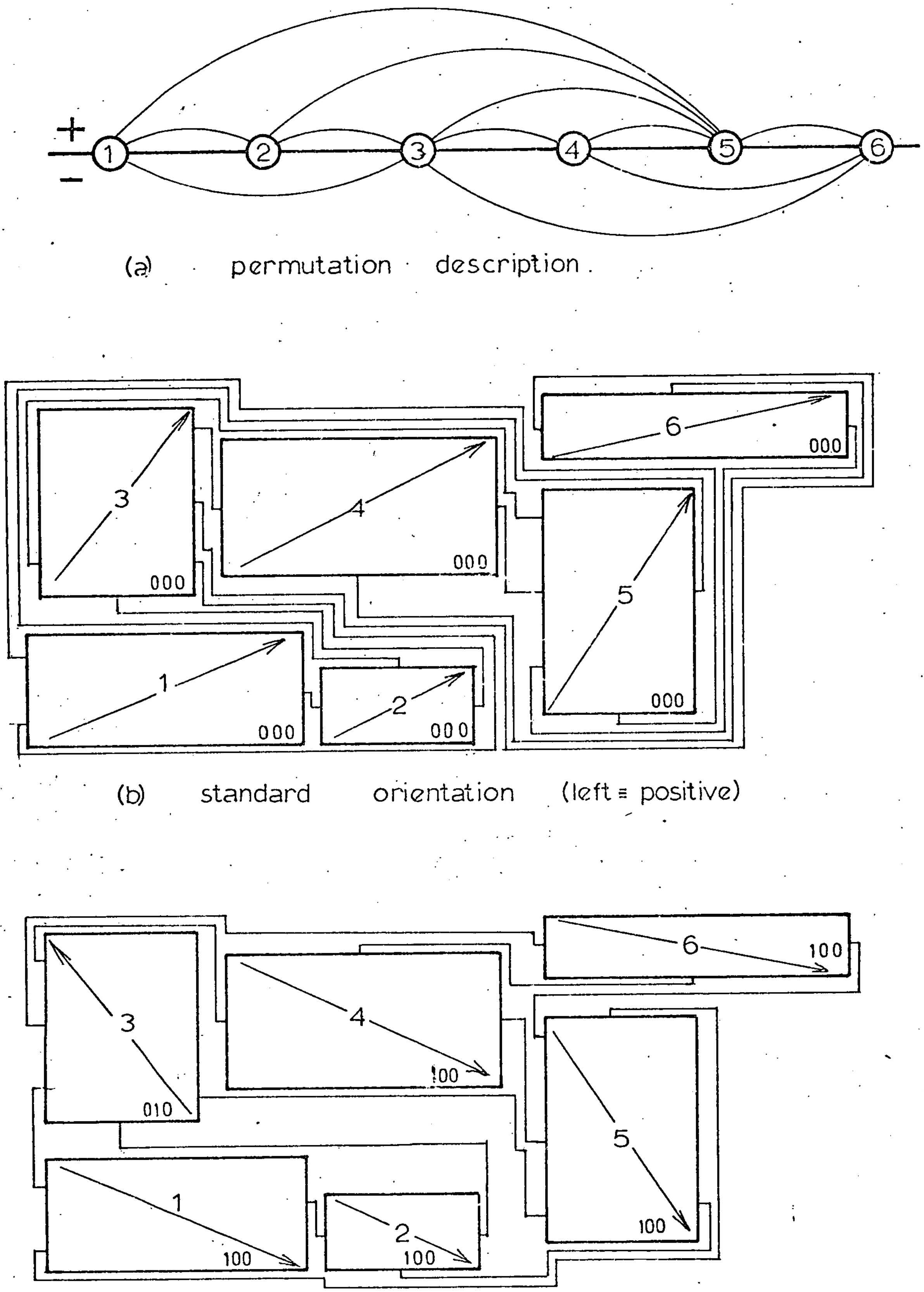
Figure 5.1 Reduction of Induced Crossings by Reflecting Components.



(a) net routed with 000 orientation

(b) 70% length reduction from reflection.

Figure 5.2 Net Length Reduction achieved by applying Reflection Operations.



(a) permutation description.

(b) standard orientation (left ≡ positive)

(c) reverse orientation

Figure 5.3 Wiring Length Reduction achieved by reversing Node Line Orientation.

layouts only.

5.1.1 CONTACT HOLES

Integrated circuits present a particular problem which deserves special mention: the need for contact holes between diffusions and metallisation tracks. In all integrated circuits, especially MOS ICs, there will be some nets which are incident on both diffused contacts and metallisation contacts. Such cases require a contact hole (such as is illustrated in figure 5.4) to provide electrical communication between the two layers. These holes occupy an area equivalent to a small device and do not need to appear adjacent to the contacts since MOS technology allows the use of diffused tracks for connections. Thus the decision of where to place a contact hole involves a subtle, intuitive juggling of many factors. It was therefore decided to omit contact holes from the automatic layout and to insert them manually.

5.1.2 MULTI-COMPONENT NETS

Another complication which arises in all technologies is concerned with routing nets which consist of more than one link. These are represented as spanning trees or spiders (section 3.6.3) and the routing process should obviously convert the constituent links into a minimum length spanning tree on the appropriate components. Net nodes (associated with spiders) should be ignored and the routing should connect net components to previously-mapped subnets if any are present. Such a scheme would require special treatment for links which belong to these nets; this would further complicate the routing algorithm. It

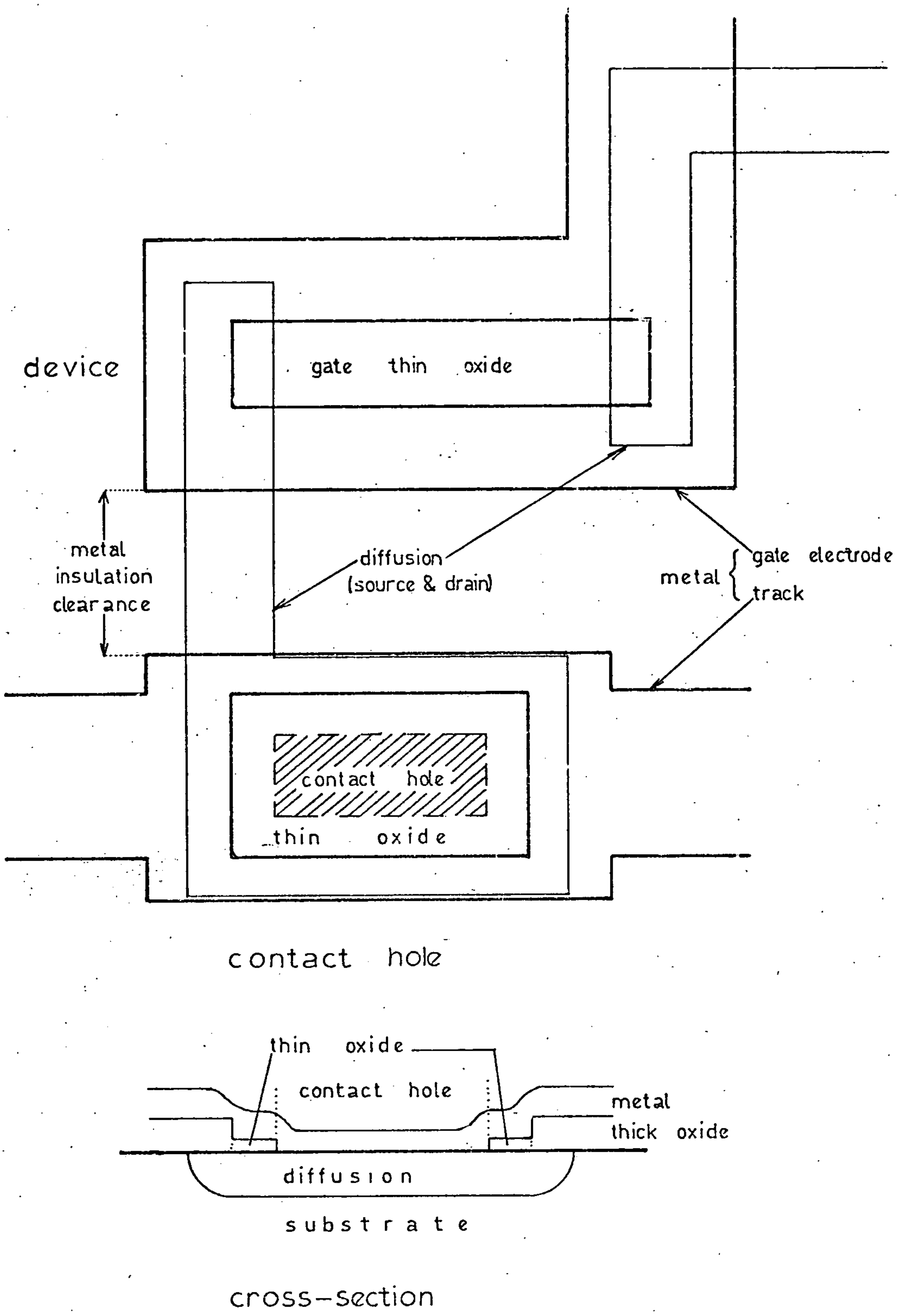


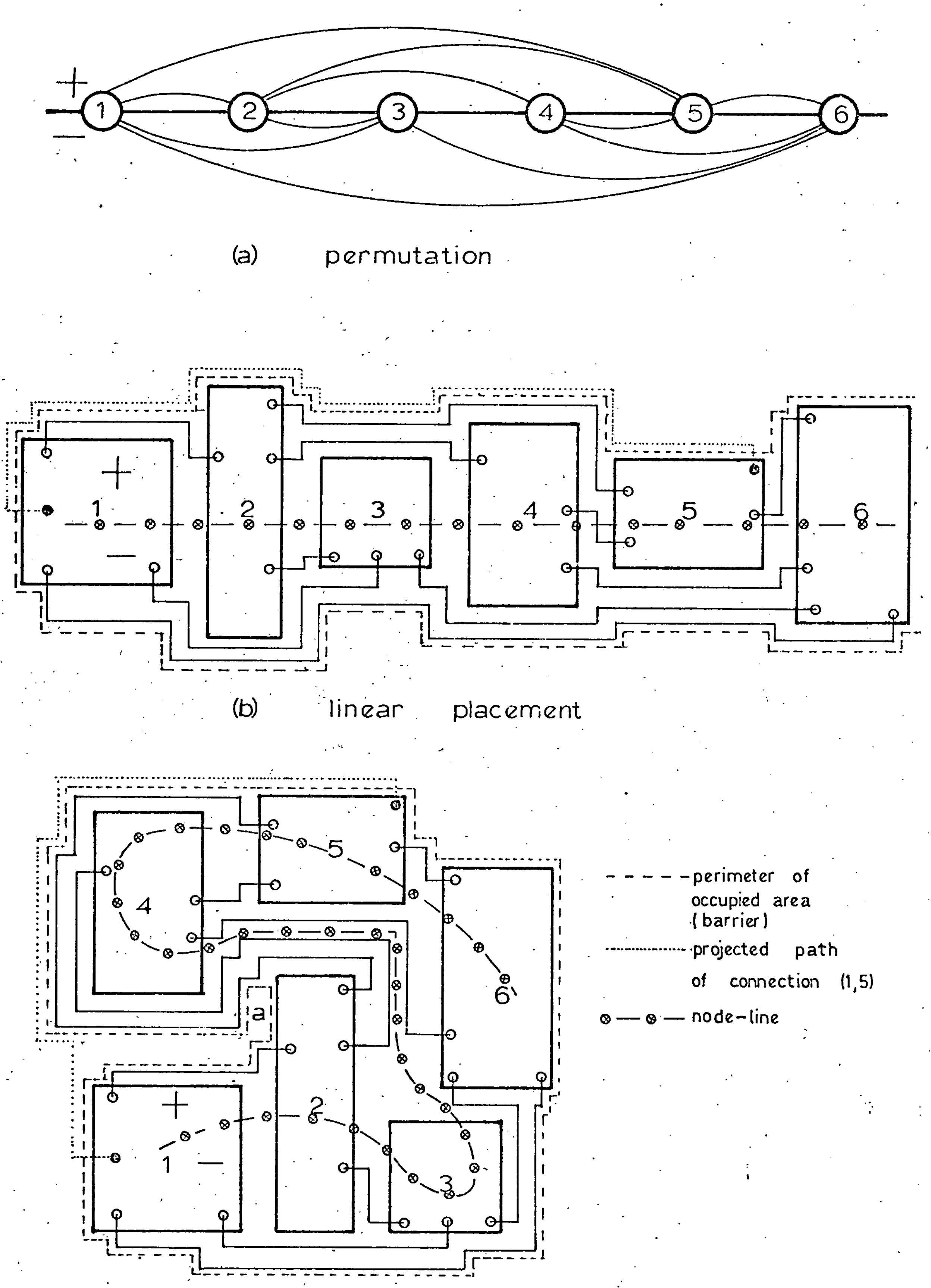
Figure 5.4 Contact Hole Construction.

was decided to initially ignore this problem in favour of manual net-merging working from a trial layout. It was hoped that a suitable scheme would arise from manual techniques, developed from working with trial layouts, which could be implemented in a subsequent version of the algorithm presented in the next section.

5.2.0 A ROUTING ALGORITHM

There are important features of this particular layout system which, in addition to normal requirements, outline an approach to routing connections. First, paths must not cross the node-line since the permutation representation of the links consists solely of single-semicircles. Secondly, routes must appear in the places that were reserved for them during placement. Figure 5.5 will be used to illustrate the implications of these features; figures 5.5(a),(b) and (c) respectively, show a permutation, a linear placement and a compact placement of the permutation.

The first constraint implies that the node-line may be effectively considered as a routing barrier. Since routes are confined to the areas between components and cannot cross other connections, the barrier may be extended to include all components and already routed connections. In figure 5.5(b), it can be seen that the perimeter of the extended barrier is represented by a dashed line which cannot be crossed except to connect a link with its incident components. The link from 1 to 5 is to be inserted, that is, the path must be selected. The permutation defines the side of the node-line on which it must appear. To minimise


+
−
1
2
3
4
5
6
- - - - -perimeter of occupied area (barrier)
·········projected path of connection (1,5)
⊗ — ⊗ — node-line
a

(a) permutation

(b) linear placement

(c) compact placement

Figure 5.5 Routing a Connection around Obstacles.

track length and the extension in the barrier's area once the path is routed, it is clear that the route should follow as close to the barrier perimeter as possible, while maintaining the minimum permissable clearance from all circuit elements within the barrier. The dotted path in figure 5.5(b) indicates the path selected on the above considerations.

This procedure can readily be applied to deformed node-line placements such as that shown in figure 5.5(c). The same reasoning applies except that it is now possible to take advantage of folds, and paths may 'spark' across narrow inlets in the barrier's perimeter such as the area labelled 'a'. It should be noted that the placement area calculations are based on such features which minimise track lengths (section 4.5.3). The selected path in figure 5.5(c) is also shown as a dotted line.

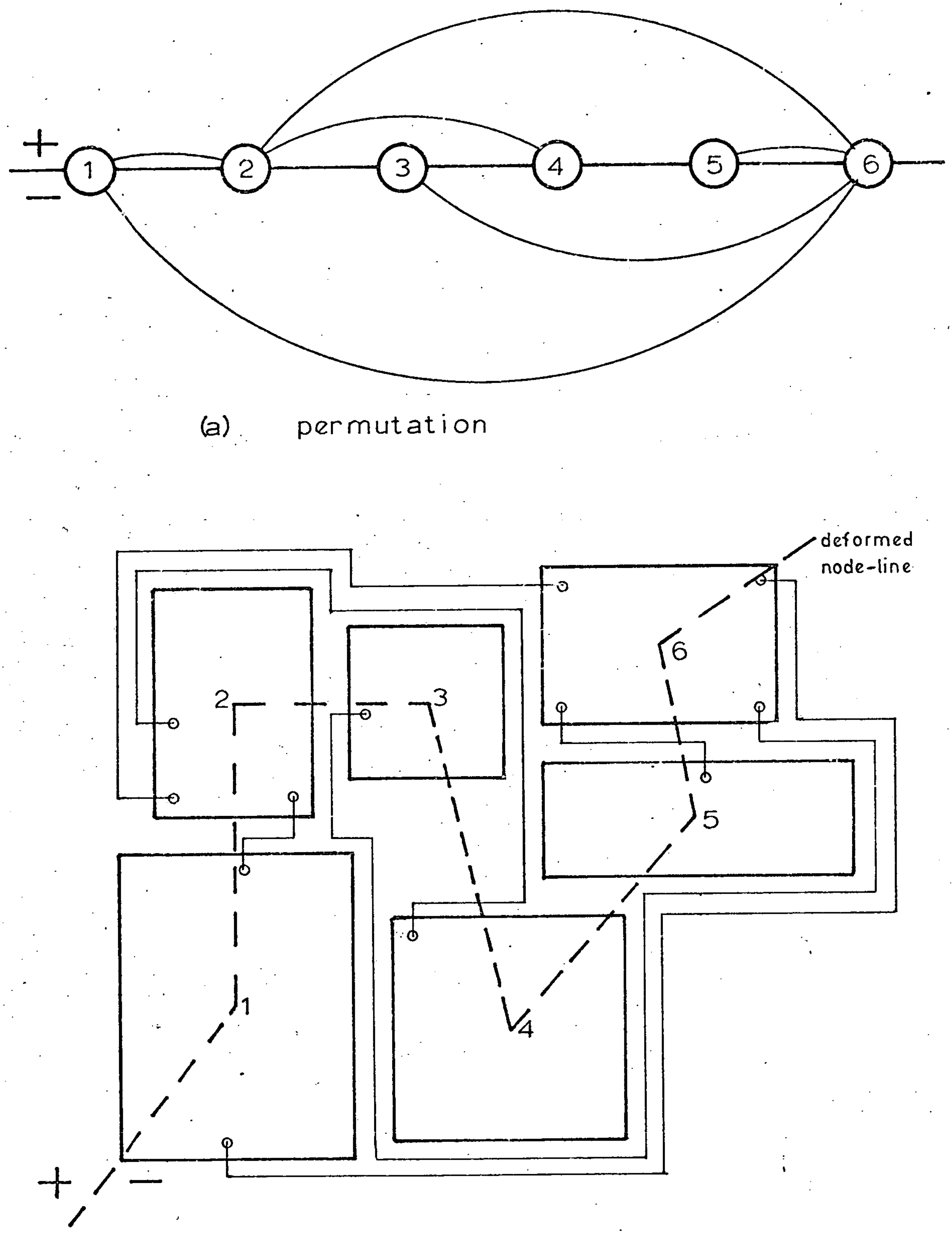
This scheme can only be used for sequential routing, that is, only one connection at a time can be inserted. Success is therefore dependant on the order in which the paths are inserted. For example, in figure 5.5(c) if link (1,6) was routed before either (1,3) or (3,6) then the latter would be wedged between the route (1,6) and component 3. This situation would require that the path of (1,6) was moved to accommodate the latter insertion. Such alterations should be avoided because they are inefficient and expensive; it is therefore necessary to provide a rigorous link ordering procedure which guarantees that each link is routed once and its path is permanent.

5.2.1 LINK SEQUENCING

Most sequential routing algorithms assign a higher priority to shorter connections since they have fewer alternatives and cause less interference to subsequent connections. This is not directly applicable here because the paths do not have freedom of choice between alternative routes. This can be seen in figure 5.6(b); although the connection (3,6) is longer than (1,6), it must be routed first. Similarly (2,4) must be routed before (2,6) although it is a longer connection.

On reflection it can be seen that links closest to the node-line should be routed first. This corresponds to a higher routing priority for shorter connections provided the length is measured in terms of a permutation which has the nodes appearing at regular intervals, such as in figure 5.6(a). There are other factors which affect the link sequencing because it is only important to provide relative priorities between links which compete for space. This means that since links which do not pass over a common section of the node-line do not affect each other, they do not need to be relatively sequenced on length alone. In addition, links which are on opposite sides of the node-line will not affect each other and so can be routed independantly. For example, in figure 5.6, link (2,6) can be routed before (1,2) and (3,6) although the latter pair are both shorter than (2,6).

It is impossible to sequence links within the data structure because there is not sufficient space available



(a) permutation

(b) layout

Figure 5.6 Example showing Link Length is not a measure of Priority.

for the necessary pointers, neither is it desirable to create an additional link list which would require extra space and require sequencing. It is possible to provide a suitable list sequence without generating lists or adding pointers by using the node sequence, which is defined by the ordered rings of the data structure, to define the link sequence. The proposed scheme, which is discussed in detail in the next section, may be stated as follows:

> Links are routed primarily in the order of their right hand nodes and secondarily in order of increasing length

This procedure provides the priorities shown in figure 5.7; it can easily be verified that the resulting sequence is as specified at the end of section 5.2.0.

5.2.2 SEQUENCING PROCEDURE

The procedure consists of two basic stepping processes using the ordered rings of the data structure. The first is a progression around the forward ordered ring; this gives the initial, partial sequence. The full sequence is generated by the second process which is applied whenever a new node is reached. It consists of following the reverse ordered ring from the primary node (that found by the first procedure). When a node which is connected to the primary is found, the connecting link is the next to be routed. Both procedures stop when the ring head is encountered; the second process stops when all links going to the left from the current primary have been found and the first procedure terminates when all the links in the network have been sequenced.

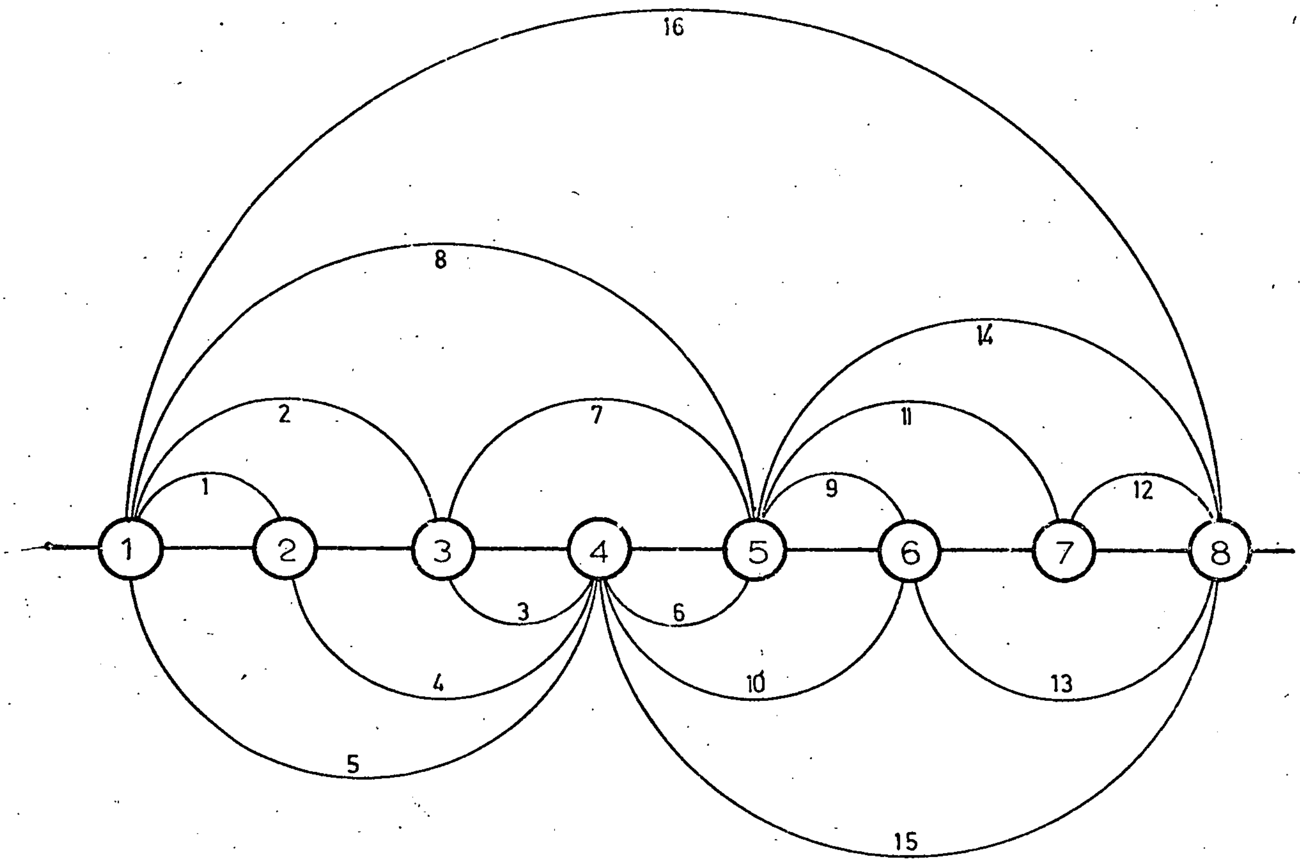


Figure 5.7 Permutation showing Link Sequence.

This technique has several advantages over other sequencing methods; first, the sequence can be generated dynamically and so avoids creating additional lists, secondly, each link is examined once only, which implies that the resulting program should be fast, and thirdly it provides a sequence which allows permanent routing of connections.

5.3 IMPLEMENTATIONS

There was not sufficient time available to implement the above described algorithm, consequently its effectiveness as a computer-based procedure cannot be properly demonstrated. However, it is felt that a measure of the system's overall effectiveness as an automatic layout aid can be obtained by manually routing connections according to the above algorithm. Results which were obtained in this fashion are presented in the next section.

5.4 RESULTS

The first layout is shown in figure 5.8; it represents a shift register stage (this is fully described in chapter 6 as circuit (i). The second example is a thin film multivibrator (described as circuit (iv) in chapter 6); figure 5.9 shows the straightforward routing (including a crossover) of the automatic placement. Figure 5.10 shows the same layout except that the nets have been merged to form minimum spanning trees and the crossing has been removed by routing the link (9,12) under component 4. The technique outlined in this chapter can therefore be seen to provide an effective means of routing connections.

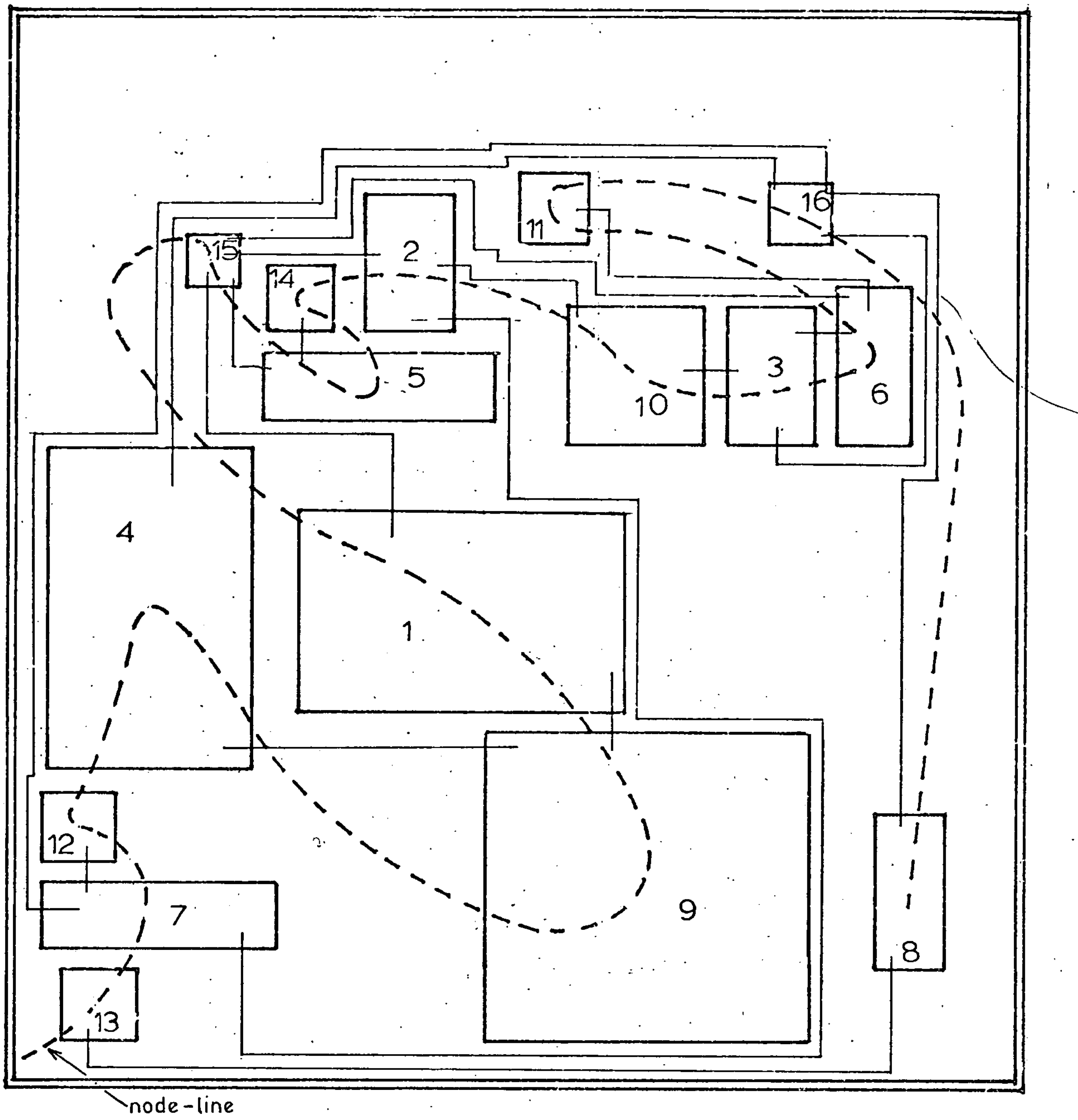


Figure 5.8 Shift Register Stage Trial Layout using automatic placement and manual routing as described in text.

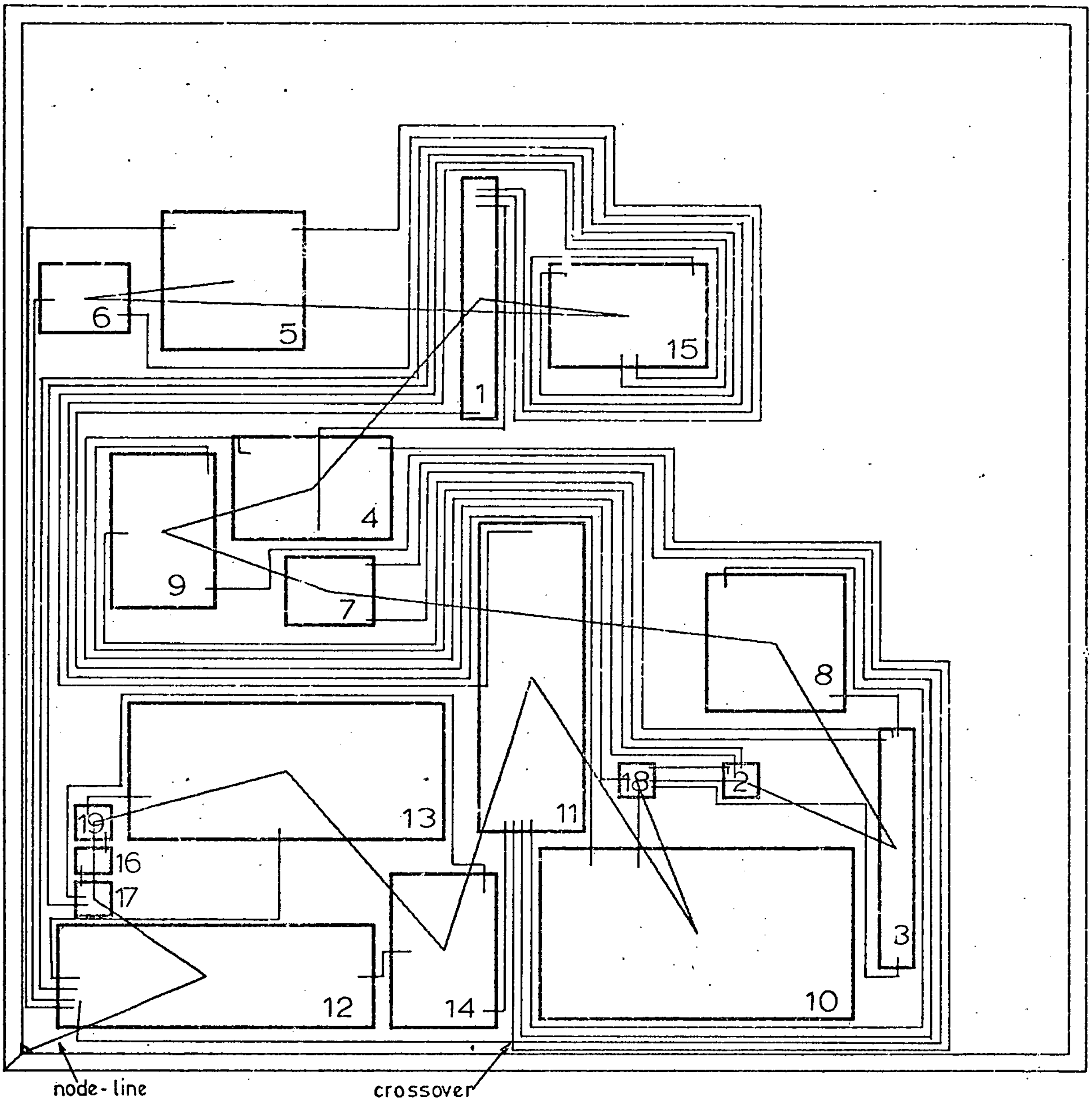


Figure 5.9 Thin Film Trial Layout using Straight-forward Routing.

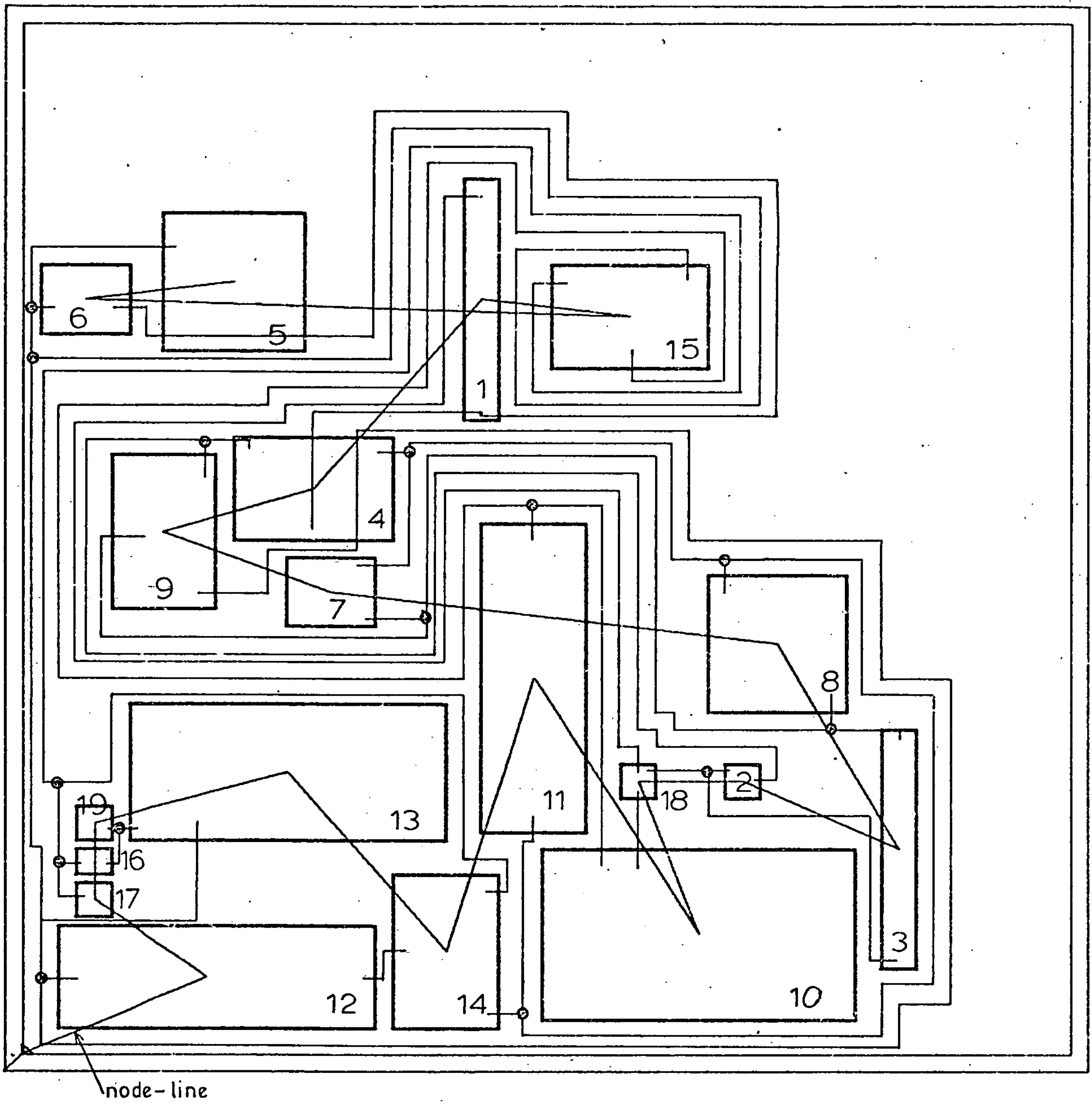


Figure 5.10 Thin Film Trial Layout using Minimal Trees for Nets.

Although the examples given all represent viable layouts, they are well below manual standards (see figures 6.4 and 6.16 for the corresponding manual layouts). Therefore there is additional, manual effort required to modify these layouts if they are to be effective. The next chapter presents an evaluation of the system compared with manual methods.

CHAPTER VI

RESULTS

6.0.0 EVALUATION REQUIREMENTS

To realistically appraise any assistance provided by the programs, it is necessary to provide a valid comparison between layouts based wholly on the results provided by the computer and manual layouts of the same circuits. The system provides trial layouts which require manual routing and it is not proper that these should be used for the basis of comparative evaluation of the system because several features have been overlooked during their construction. Moreover, the trial layouts require manual modification to convert the circuit element representations (of rectangles and lines) into accurate geometries. Therefore, it was decided that realistic layouts would be manually constructed using only the permutation and placement provided by the machine. Modifications would be allowed to take advantage of features such as net merging, under-component wiring, diffused connections and component orientations, which had not been fully incorporated in the system. This would provide machine-based layouts which were sufficiently realistic to permit worth-while comparison with equivalent, wholly manually produced layouts.

It was considered inadvisable to assign the modification process of the computer-aided layouts to a skilled designer, because his optimising skills and experience would significantly contribute to the quality of the results.

Therefore, in order to preserve as much of the computer's design effort as possible, it was decided that the modification would be carried out by a person with no layout experience* who would be required to comply with the appropriate design rules.

6.0.1 EVALUATION TECHNIQUE

Five circuits have been processed in the above manner, three MOS IC cells, a thin film multivibrator and a printed circuit board. All were laid out by minimally modifying the component placement and permutation connection configuration, except the last in which the computer's placement was not disturbed. The first four circuits have equivalent manual layouts as shown, which have been used for fabrication and may therefore be considered as optimal, from the manufacturer's point of view. The printed circuit has two comparative layouts included, one produced by an automatic PCB layout program (Rose, 1970) and the other by interactive modification of the automatic layout. These layouts are the basis of the comparison. The integrated circuits layouts have been constructed in compliance with the design rules listed in appendix II, using four masks which are respectively the diffusion, thin oxide, contact hole and metallisation geometries; they are all drawn in superimposition at a magnification of 500x.

Circuit input descriptions were compiled directly from the circuit schematics using component dimensions measured from the manual layouts. The permutation program occupied 7K and the placement program 11K of core on the University

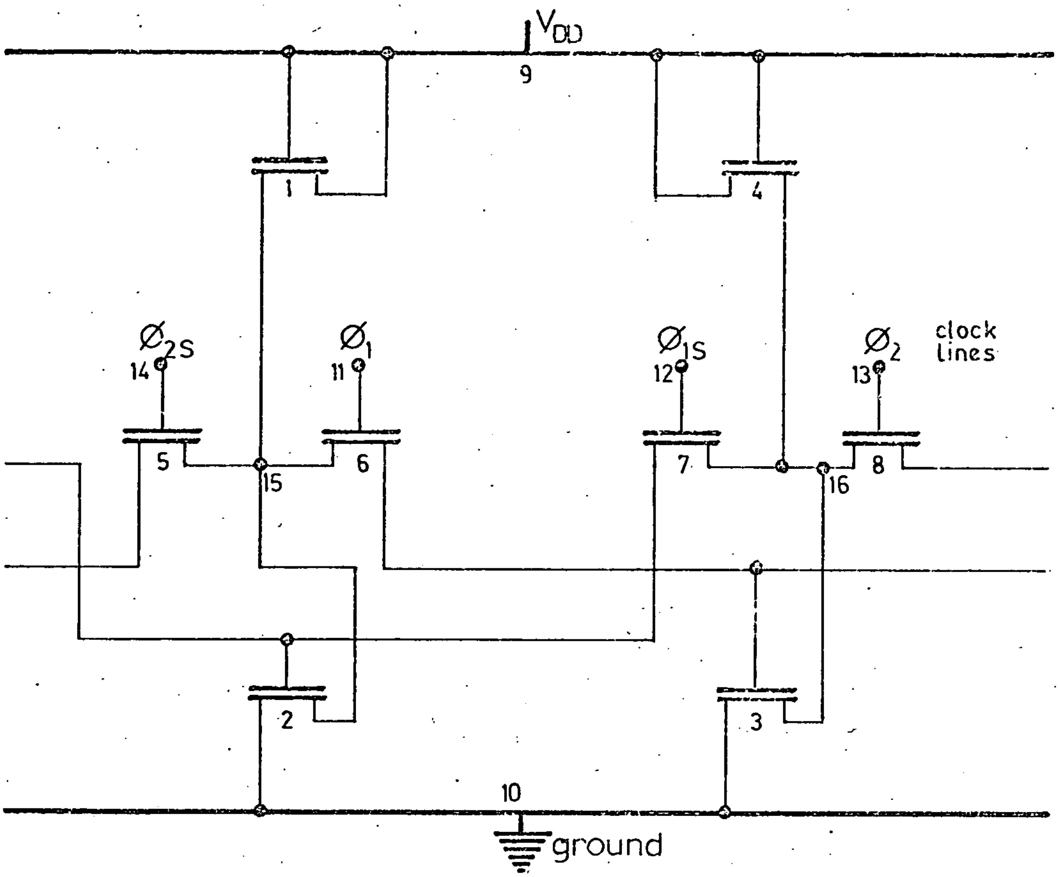
*the author

of Edinburgh CAD project's PDP 10. These programs are included in appendix IV in flowchart form. The comparisons are based on two criteria: time requirements (machine + man versus man alone) and overall layout area. These were chosen because they are suitably objective and are measures of design and production costs. The circuits will be presented individually with brief mentions of any particular features which affect the layout.

6.1.0 CIRCUIT (i) - MOS SHIFT REGISTER STAGE

The circuit schematic and input description are shown in figure 6.1, the computer output in figure 6.2. The resulting layout appears in figure 6.3 while figure 6.4 shows the equivalent manual layout. The main feature of a shift register is that the repetitive stages are usually placed side by side to conserve space, thus the power, ground and signal lines are all required to appear at the same elevation in both sides of the layout. This feature has been incorporated during the modification process and has resulted in a significant increase in circuit area as much real estate has been devoted to the traversing tracks.

It should be noted that the layout area could be substantially reduced by expanding the cell vertically and contracting it horizontally to put the devices in vertical alignment. This was felt to be too significant a modification as it would inevitably swamp the computer's design and therefore was avoided. The manual layout (figure 6.4) has taken advantage of this feature and is consequently much smaller.



(a) circuit schematic.

```
1,90,140,4,2001,9,9,15
2,60,40,5,2001,15,10,7
3,60,40,6,2001,6,16,10
4,90,140,4,2001,9,9,16
5,100,30,5,2001,15,14
6,70,30,6,2001,11,15,3
7,100,30,4,2001,12,2,16
8,70,30,5,2001,16,13
9,140,140,1,4000,1,1,4,4
10,60,60,1,2000,2,3
11,30,30,1,1000,6
12,30,30,1,1000,7
13,30,30,1,1000,8
14,30,30,1,1000,5
15,25,25,9,1111,2,6,1,5
16,25,25,9,1,7,3,8,4
-1,-1,-1,-1,-1,-1,-1
```

(b) circuit input description.

Figure 6.1 Shift Register Stage — INPUT

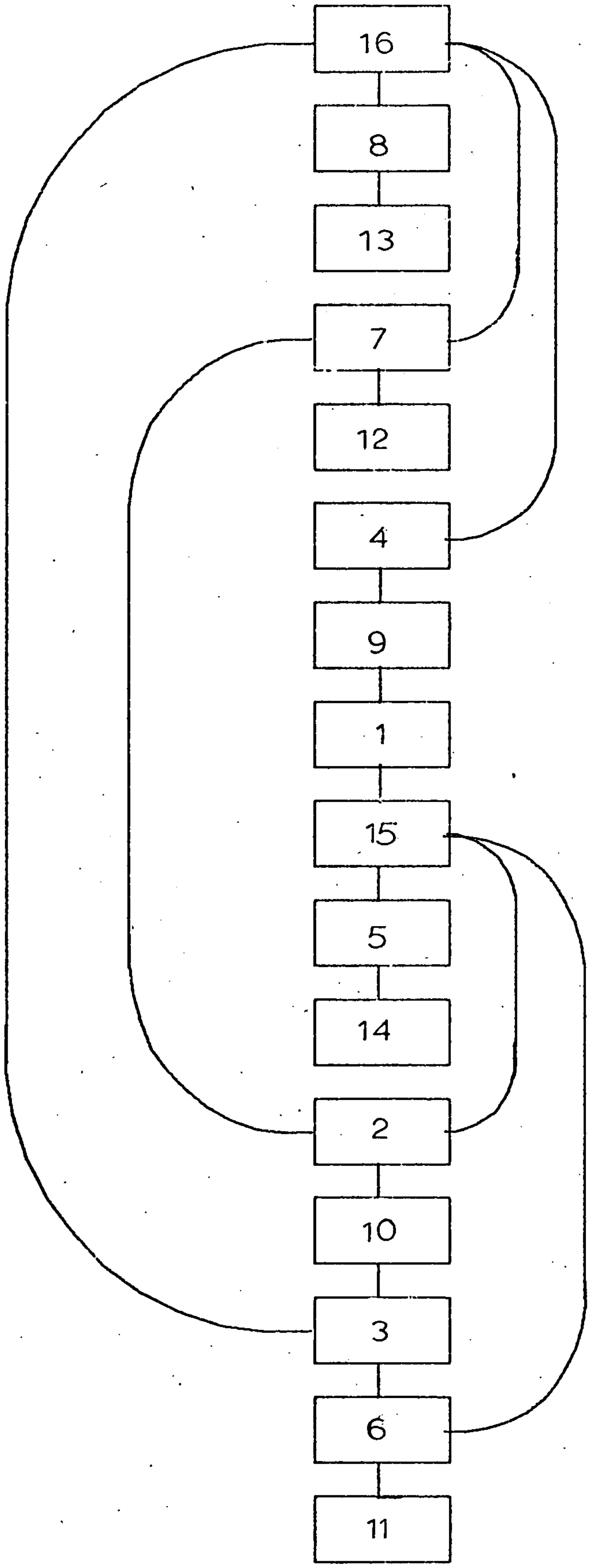


Figure 6.2 (a) Circuit (i) – permutation.

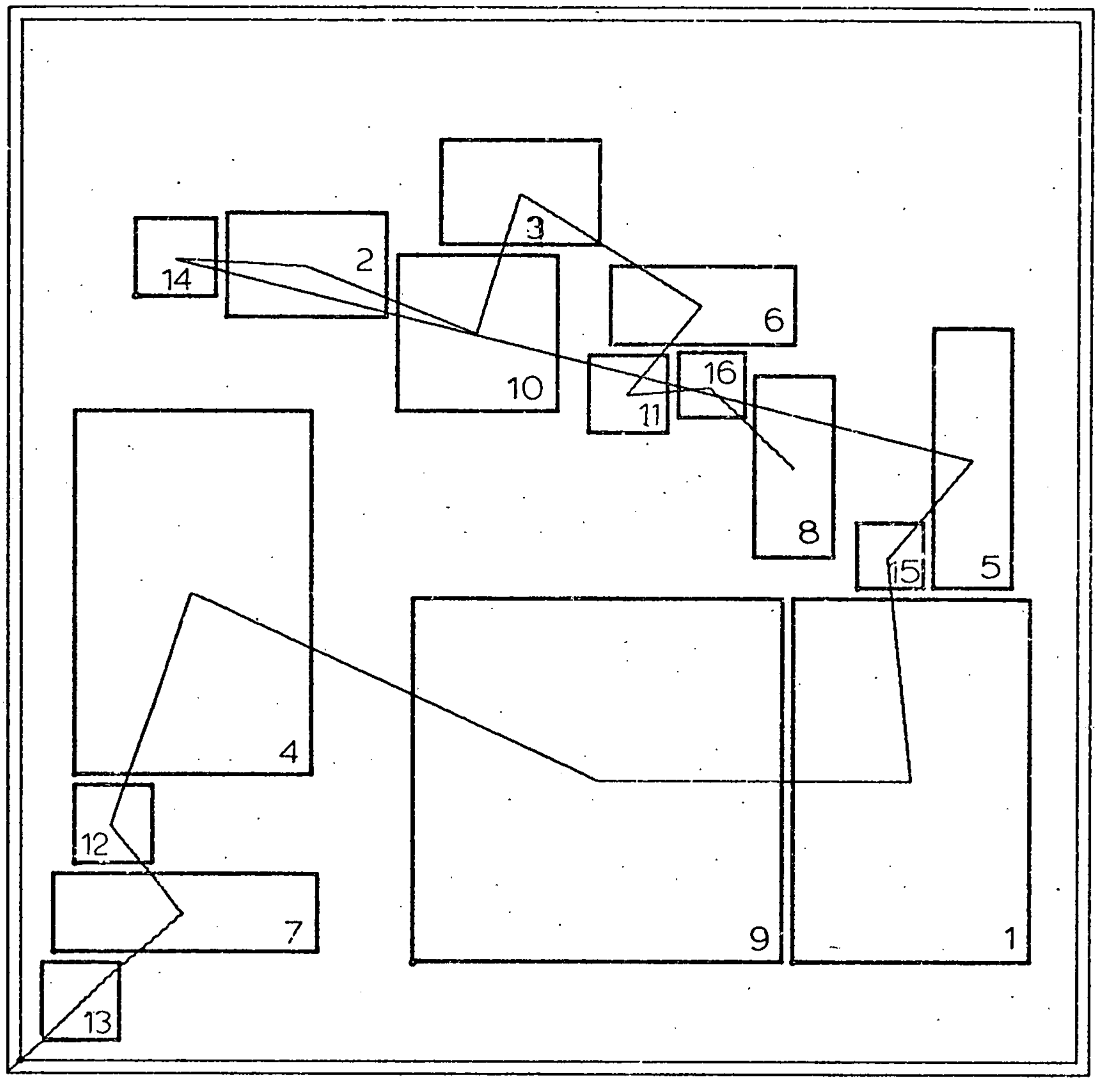


Figure 6.2 (b). Circuit (i) — automatic placement.

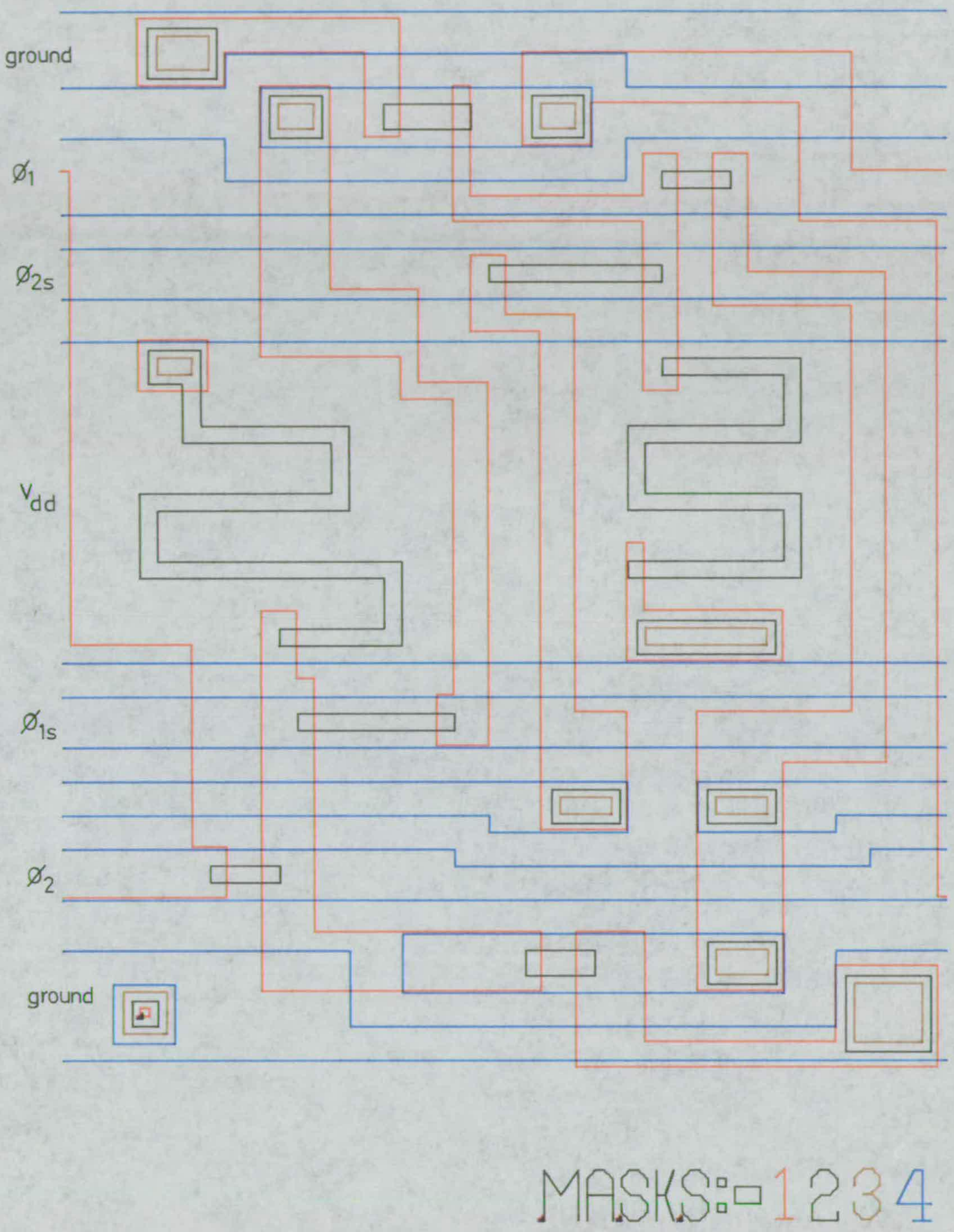


Figure 6.3 Circuit (i) - Computer Aided Layout

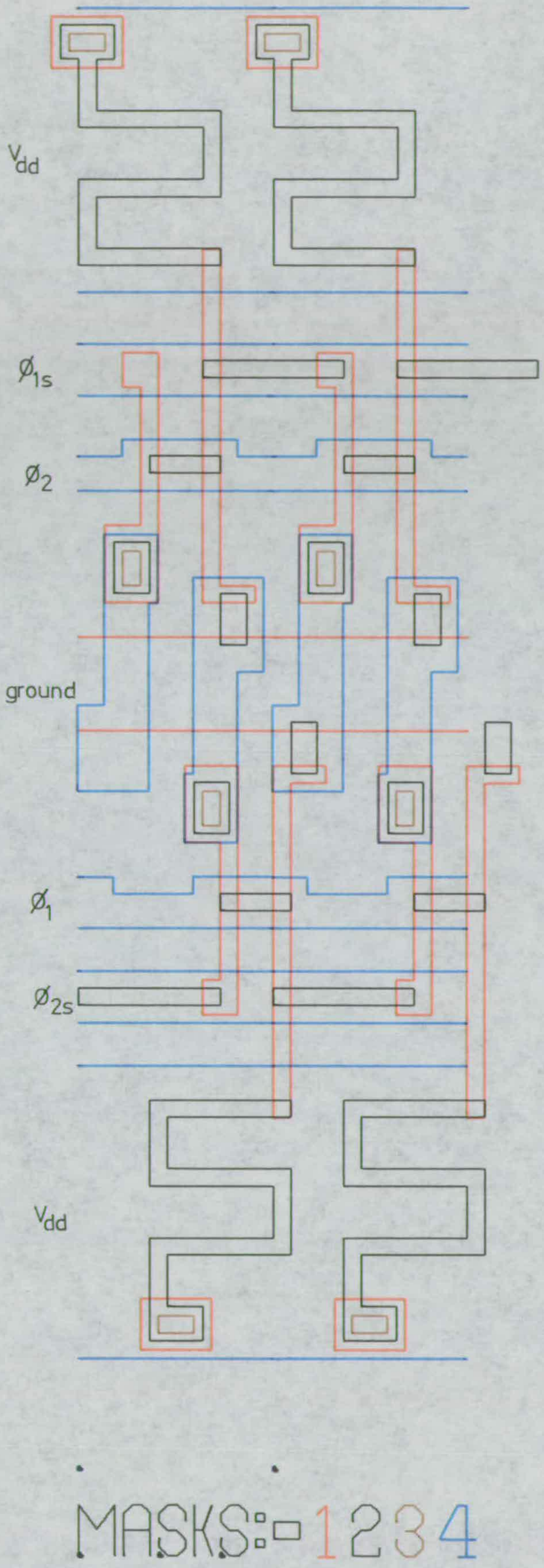


Figure 6.4   Circuit (i) - Manual Layout

6.1.1 COMPARISON

Table 6.1 contains the figures for the comparison, showing a 210% increase in area.

	Computer-aided layout	Manual layout
Time M/C (secs)	5	-
Man (hrs)	4	20*
Layout Dimensions (thou)	12.4 x 10.2	2.7 x 15.0
Area (sq thou)	126.5	40.5

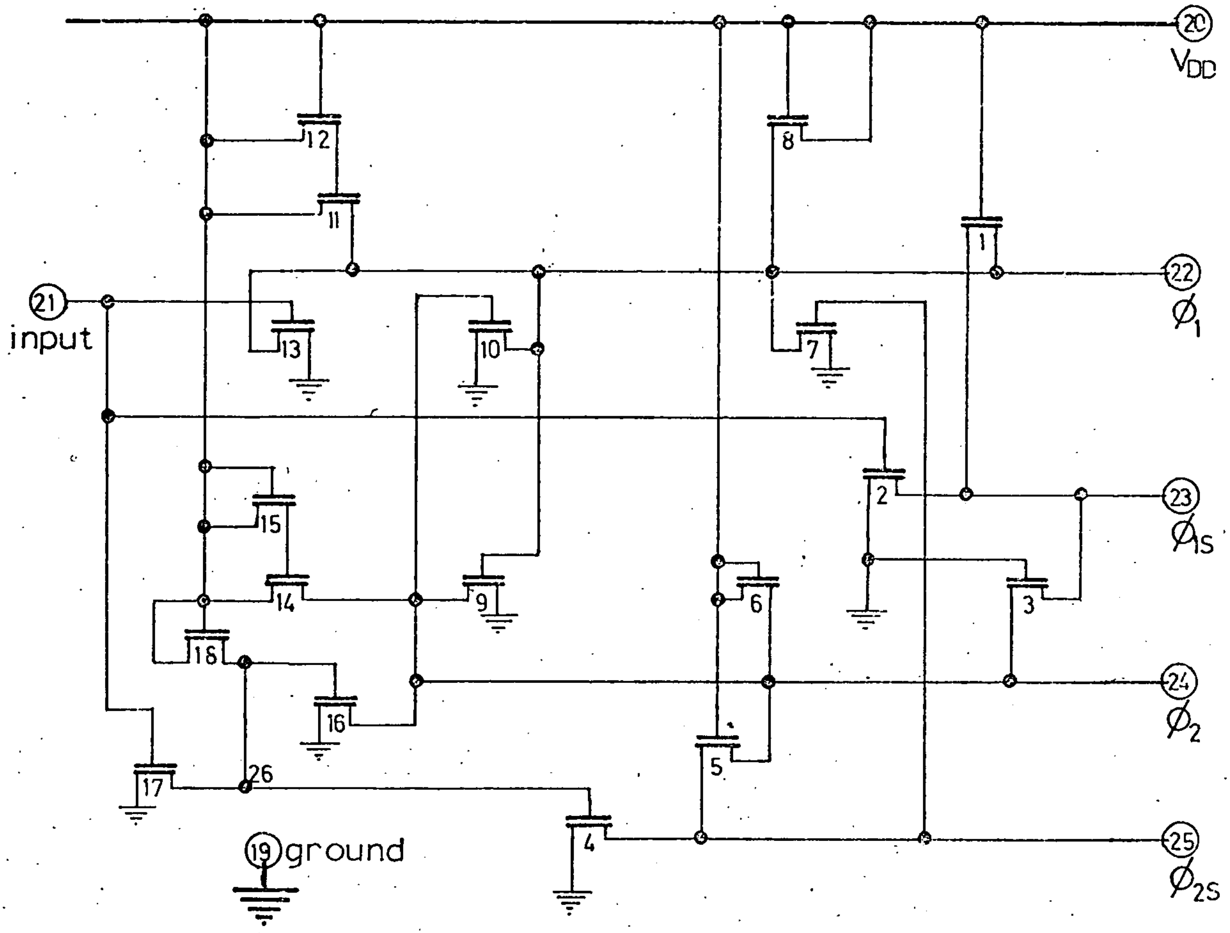
TABLE 6.1: Circuit (i) - comparative results

This increase is due more to the unique requirements of a shift register stage than any serious fault of the program. It indicates that the program cannot cope with special requirements, such as repetition, without substantial alterations or manual assistance.

6.2.0 CIRCUIT (ii) - MOS CLOCK DRIVER CELL

Figures 6.5 and 6.6 show the computer input and output respectively; figures 6.7 and 6.8 show the computer-aided and manual layouts respectively. The manual layout contains two capacitors and one device which were excluded from the schematic. One of these (a capacitor) has been manually added to the machine's layout and the others could be added with little effort and would produce an increase in the circuit area of less than 3%. During the modification process, several device outlines were altered, specifically those of components 1,5,8 and 15, to make better use of available spaces.

* All time figures are estimates derived from consultation with the circuit designers involved.



(a) circuit schematic

Figure 6.5 Clock Driver — INPUT

```
1,80,100,1,21,22,20,23
2,88,26,1,12,21,23,19
3,88,26,1,12,23,19,24
4,88,24,1,30,26,25,19
5,16,40,1,30,20,24,25
6,16,36,1,3,24,20
7,88,24,1,30,22,25,19
8,80,80,1,21,22,20
9,88,24,1,12,24,22,19
10,88,24,1,111,22,24,19
11,44,12,1,120,12,20,22
12,36,12,1,210,11,20
13,88,24,1,300,21,22,19
14,44,12,1,3,15,24,20
15,80,80,1,3000,14,20
16,88,24,1,1020,26,24,19
17,44,24,1,1011,21,26,19
18,40,40,1,2010,26,20
19,30,30,2,4211,2,3,4,7,9,10,13,16,17
20,30,30,2,3103,1,5,6,8,11,12,14,15,18
21,25,25,2,1011,2,17,13
22,25,25,3,3202,1,7,8,9,10,11,13
23,25,25,3,111,1,2,3
24,25,25,3,34,3,5,6,9,10,14,16
25,25,25,3,30,4,5,7
26,20,20,4,202,4,16,17,18
-1
```

Figure 6.5 (b) Circuit (ii) - Input Description.

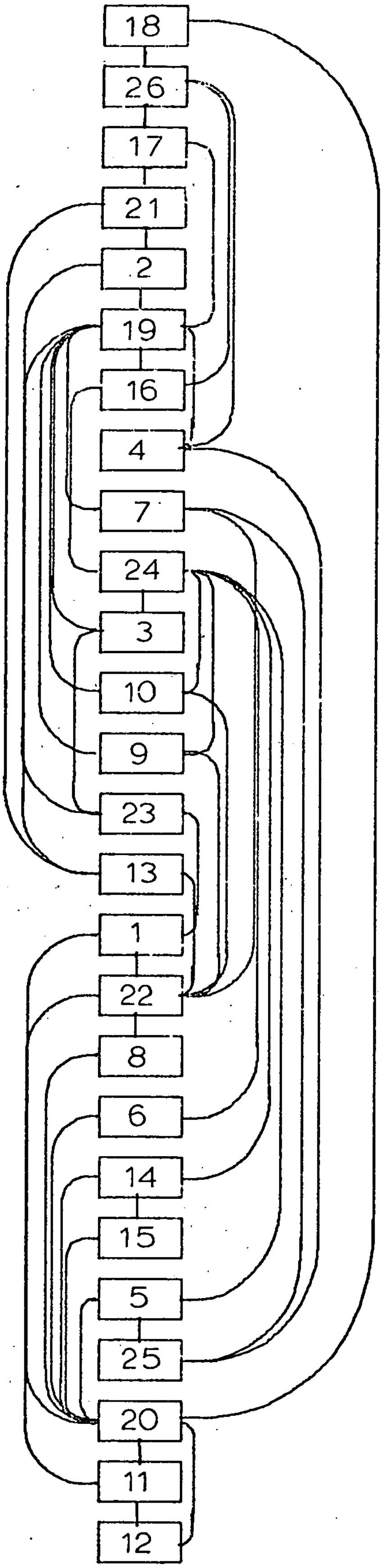


Figure 6.6 (a) Circuit (ii) — permutation.

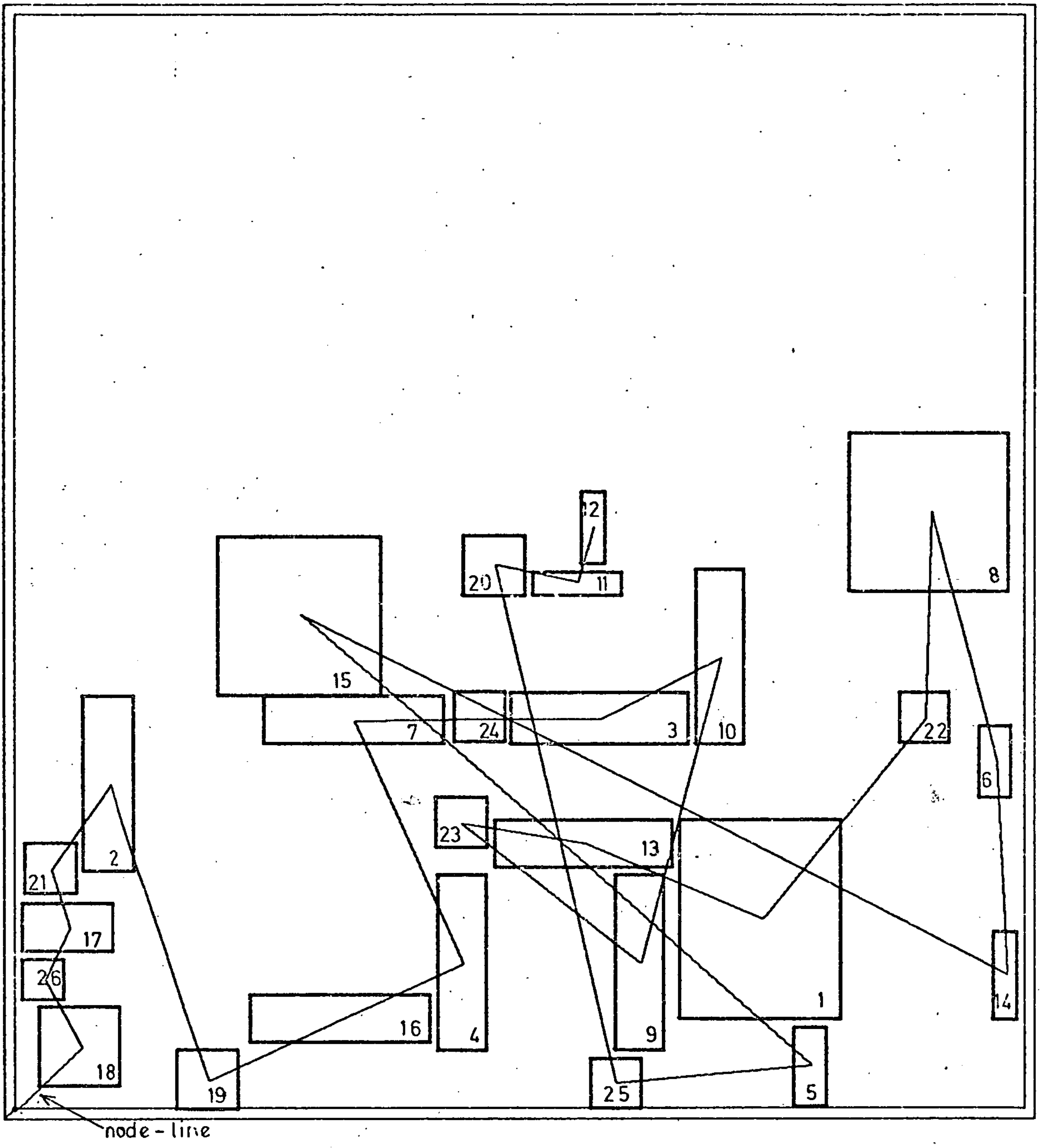


Figure 6.6 (b) Circuit (ii) — automatic placement.

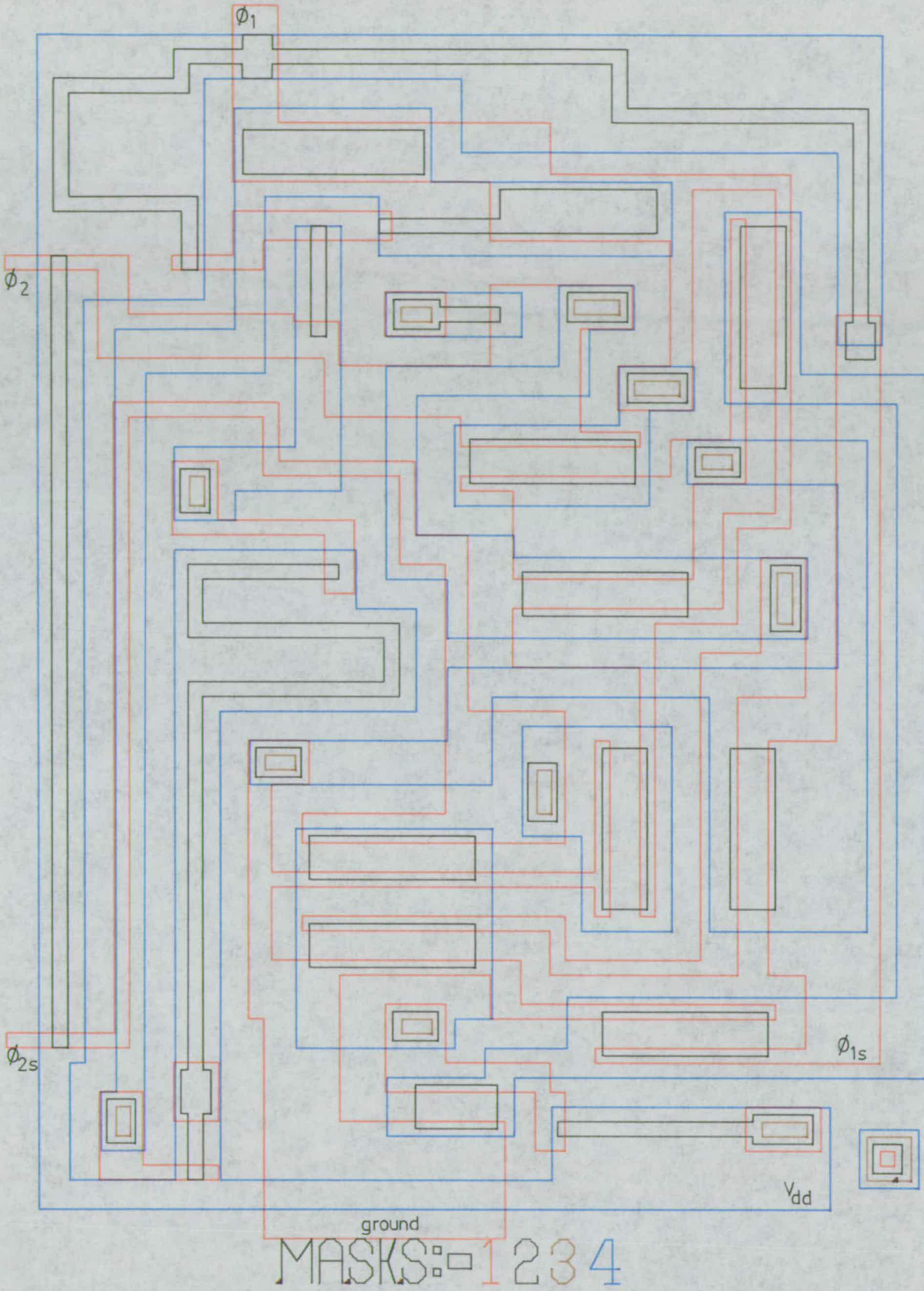


Figure 6.7 Circuit (ii) - Computer Aided Layout

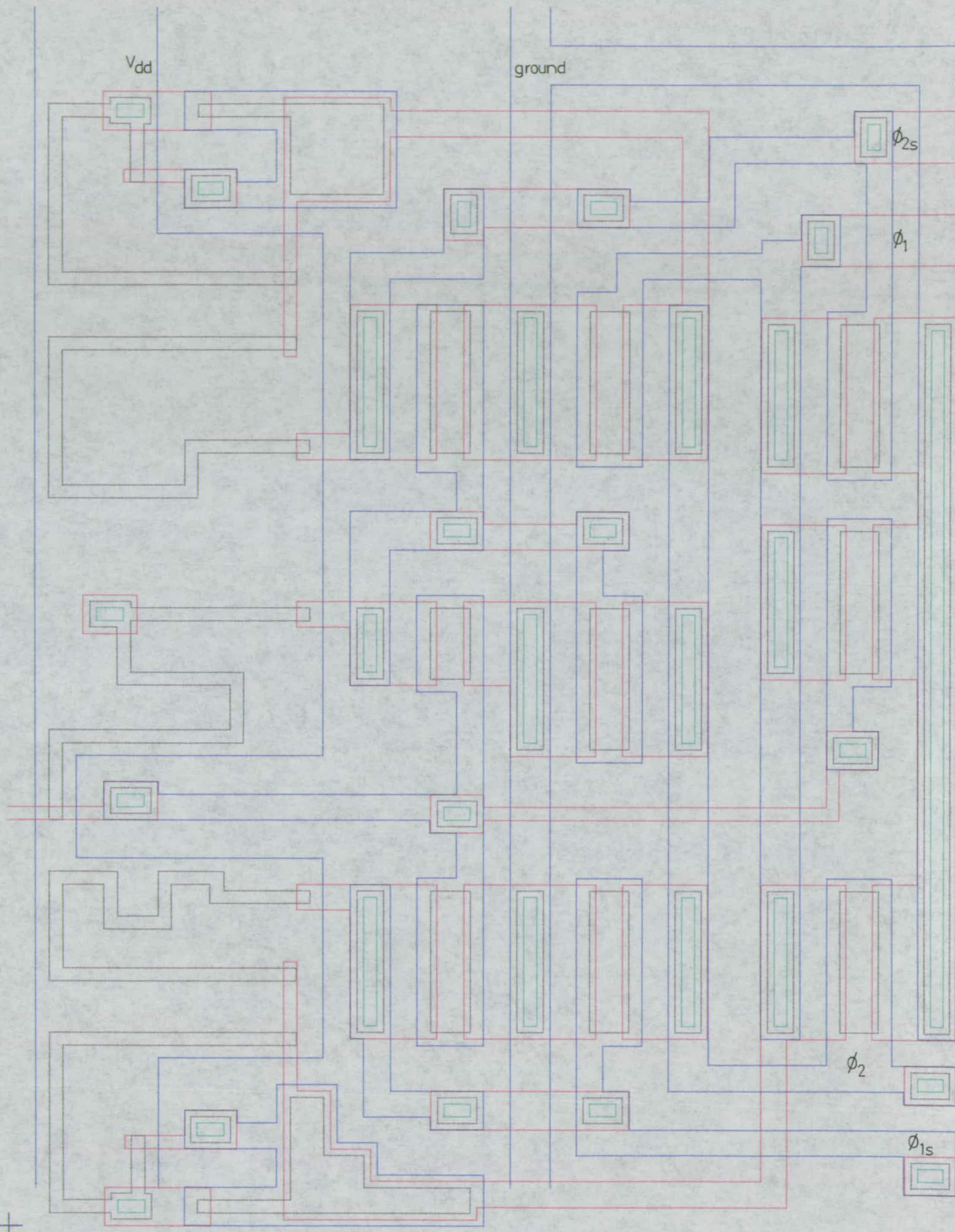


Figure 6.8   Circuit (ii) Manual Layout

6.2.1 COMPARISON

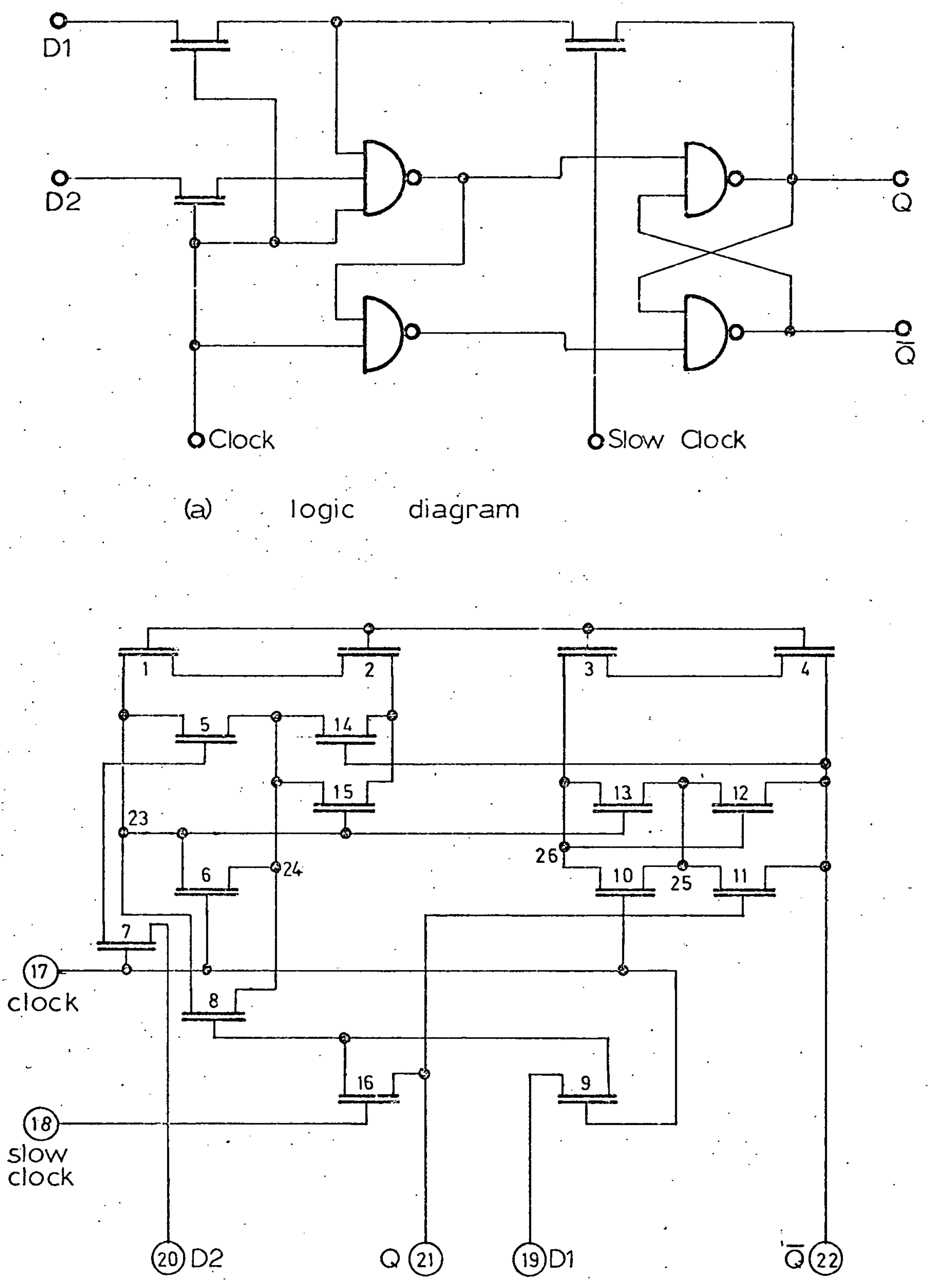
Table 6.2 contains the relevant layout figures which indicate a decrease in area of 25% in comparison with the manual layout. This shows that the programs are capable of supplying designs of a standard equal to manual methods, but with far less manual effort.

	Computer-aided layout	Manual layout
Time M/C (secs)	103	-
Man (days)	0.75	6
Layout Dimensions (thou)	11.8 x 16.0	13.9 x 18.3
Area (sq thou)	188.8	254.4

TABLE 6.2: Circuit (ii) - comparative results

6.3.0 CIRCUIT (iii) - MOS STATIC D-TYPE 2 INPUT FREE RUNNING BISTABLE

The logic diagram, schematic and input description are shown in figure 6.9, program outputs in figure 6.10. The computer-aided and manual layouts appear as figures 6.11 and 6.12 respectively. The manual design is used as a standard building block for logic chips manufactured by a large microcircuit company and so it has been subjected to much critical examination for possible improvements. The design may therefore be considered as the optimum layout. It can be seen that the manual layout is considerably neater, that is, systematic and visually pleasing. This is a feature which is intuitively incorporated by most designers since it enhances a layout, but which cannot be incorporated in any computer program because it defies precise mathematical definition. In this example the



(a) logic diagram

(b) circuit schematic

Figure 6.9 Static D-type Flip-Flop – INPUT

```
1, 200, 60, 5, 1020, 2, 2, 23
2, 200, 60, 5, 1020, 1, 3, 1, 14, 15
3, 200, 60, 5, 1020, 2, 4, 4, 26
4, 200, 60, 5, 1020, 3, 3, 22
5; 30, 20, 6, 1020, 23, 24, 7
6, 30, 20, 6, 1020, 23, 24, 17
7, 40, 10, 7, 1020, 5, 20, 17
8, 30, 30, 8, 1020, 23, 24, 16
9, 40, 10, 7, 1021, 9, 8, 19, 17
10, 30, 20, 6, 1020, 25; 26, 17
11, 30, 20, 6, 1020, 21, 22, 25
12, 30, 20, 6, 1020, 22, 25, 26
13, 30, 20, 6, 1020, 23, 25, 26
14, 30, 20, 6, 1020, 2, 24, 22
15, 30, 20, 6, 1020, 2, 23, 24
16, 40, 10, 7, 1020, 8, 21, 18
17, 30, 30, 1, 4, 7, 6, 10, 9
18, 30, 30, 1, 1, 16
19, 30, 30, 1, 1, 9
20, 30, 30, 1, 1, 7
21, 30, 30, 1, 2, 16, 11
22, 30, 30, 1, 4, 11, 12, 4, 14
23, 20, 20, 9, 6, 1, 5, 8, 6, 15, 13
24, 20, 20, 9, 6, 5, 6, 8, 14, 15
25, 10, 10, 9, 4, 10; 11, 12, 13
26, 10, 10, 9, 4, 3, 10, 12, 13
-1
```

Figure 6.9 (c) Circuit (iii) - Input Description.

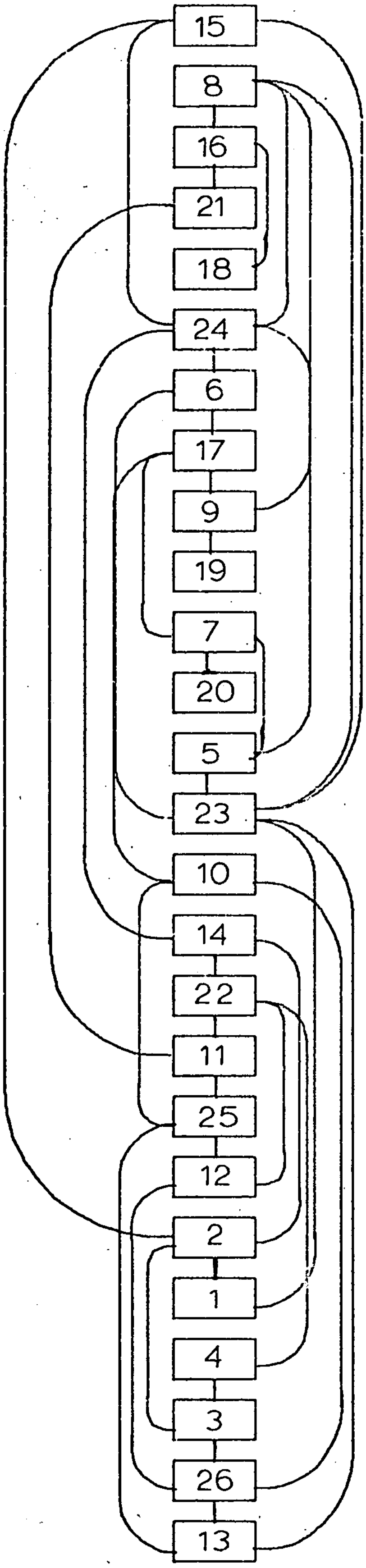


Figure 6.10 (a) Circuit (iii) – permutation

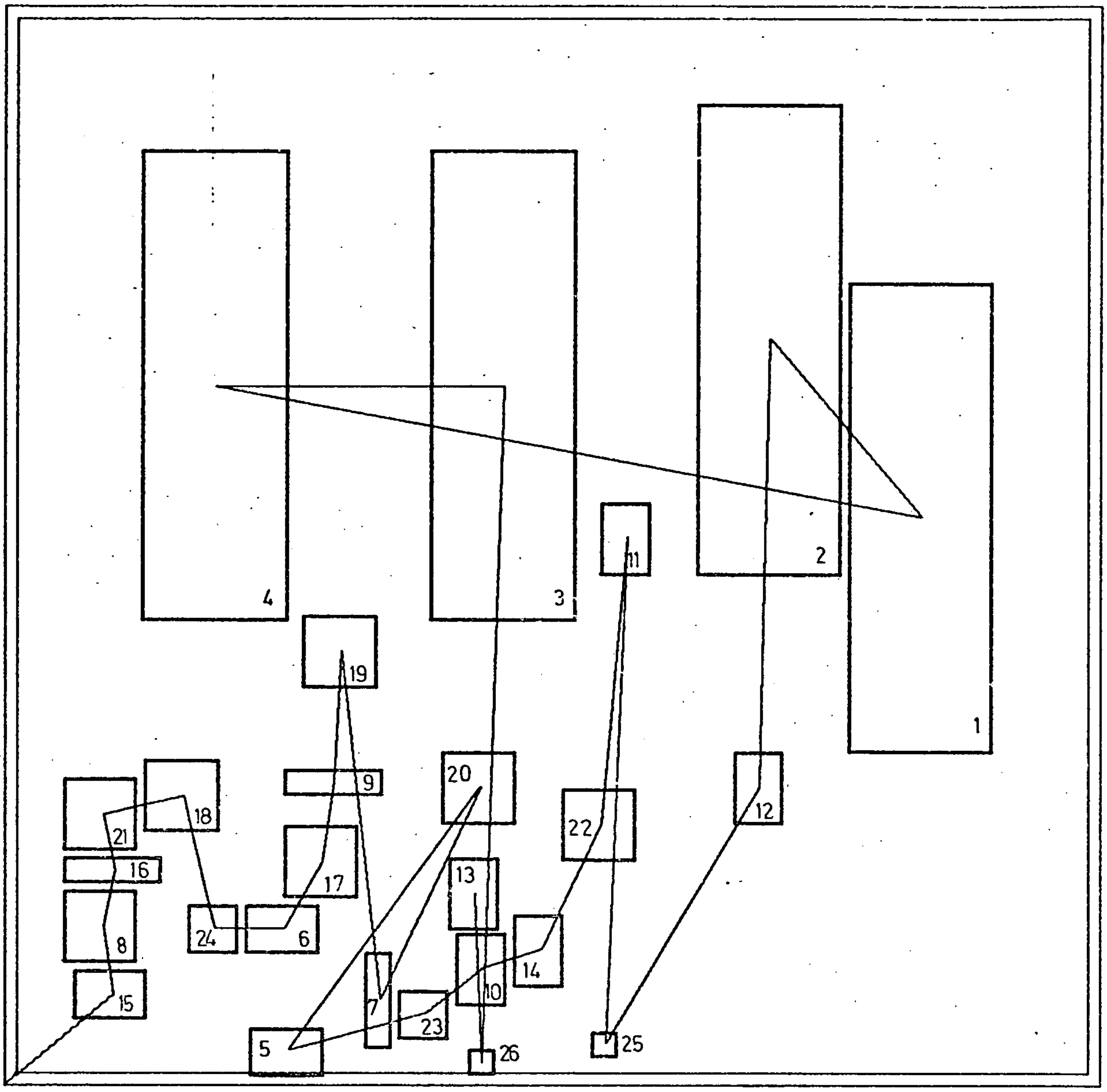


Figure 6.10 (b) Circuit (iii) – automatic placement.

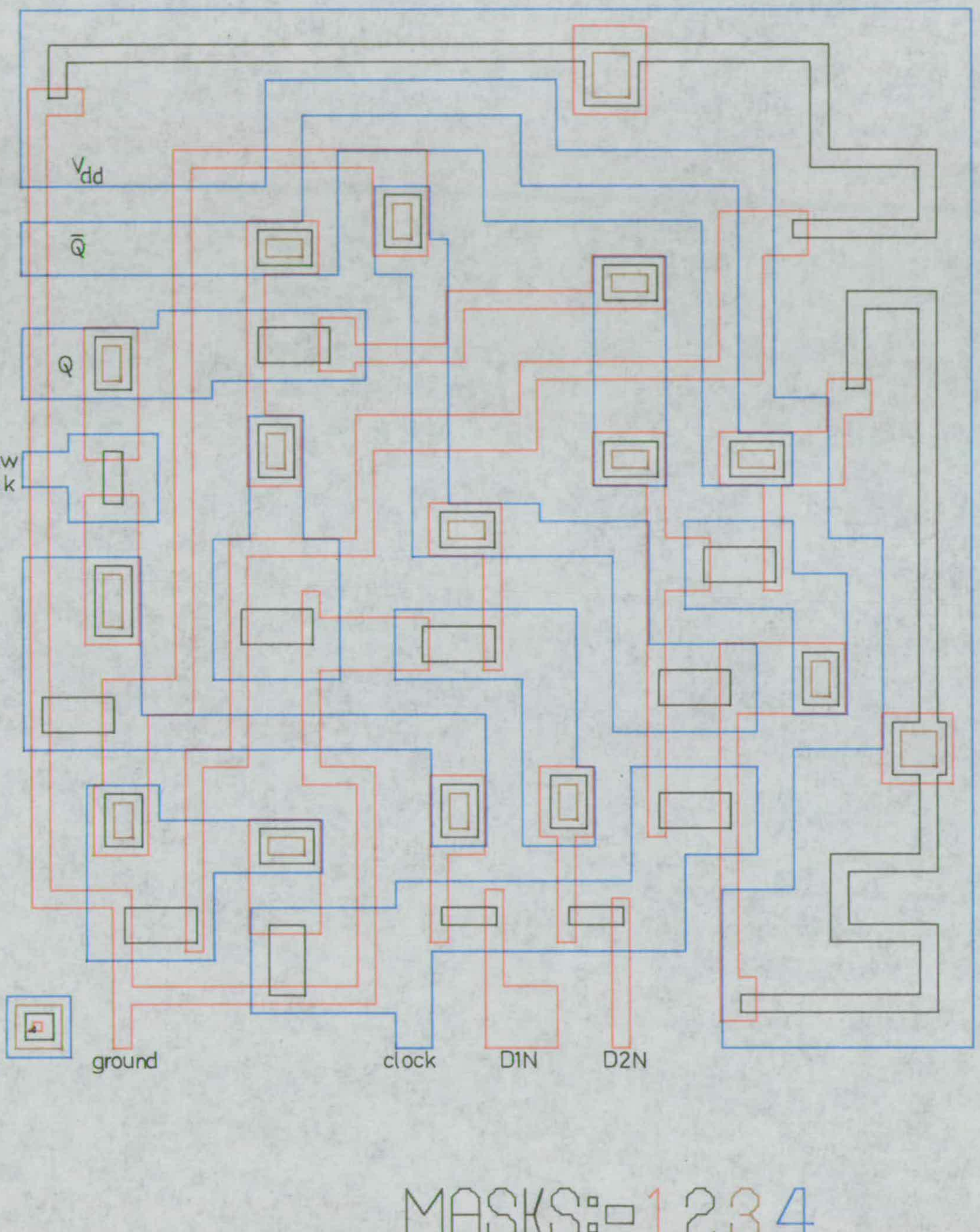


Figure 6.11 Circuit (iii) - Computer-Aided Layout

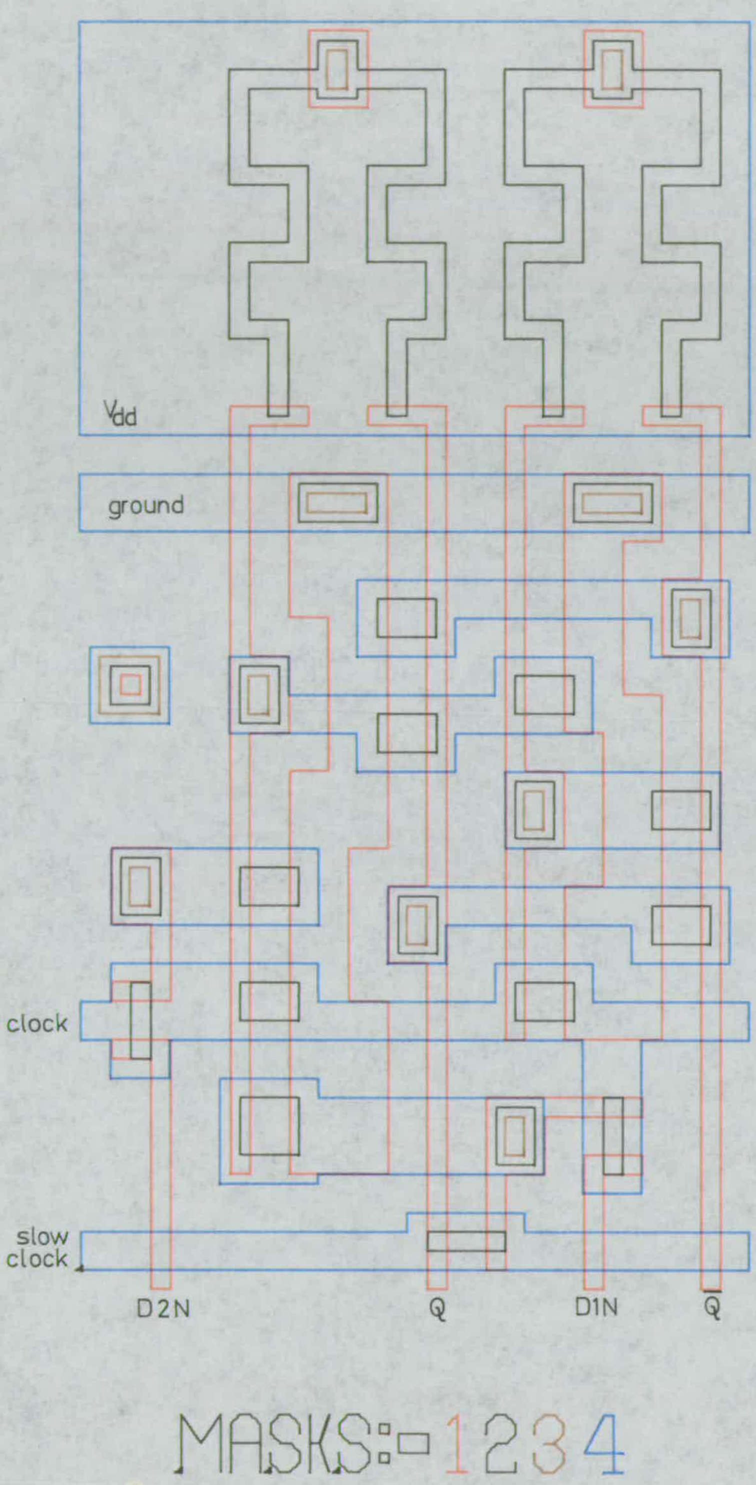


Figure 6.12 Circuit (iii) - Manual Layout

neatness has resulted in a minimum of real-estate being consumed by connections.

6.3.1 COMPARISON

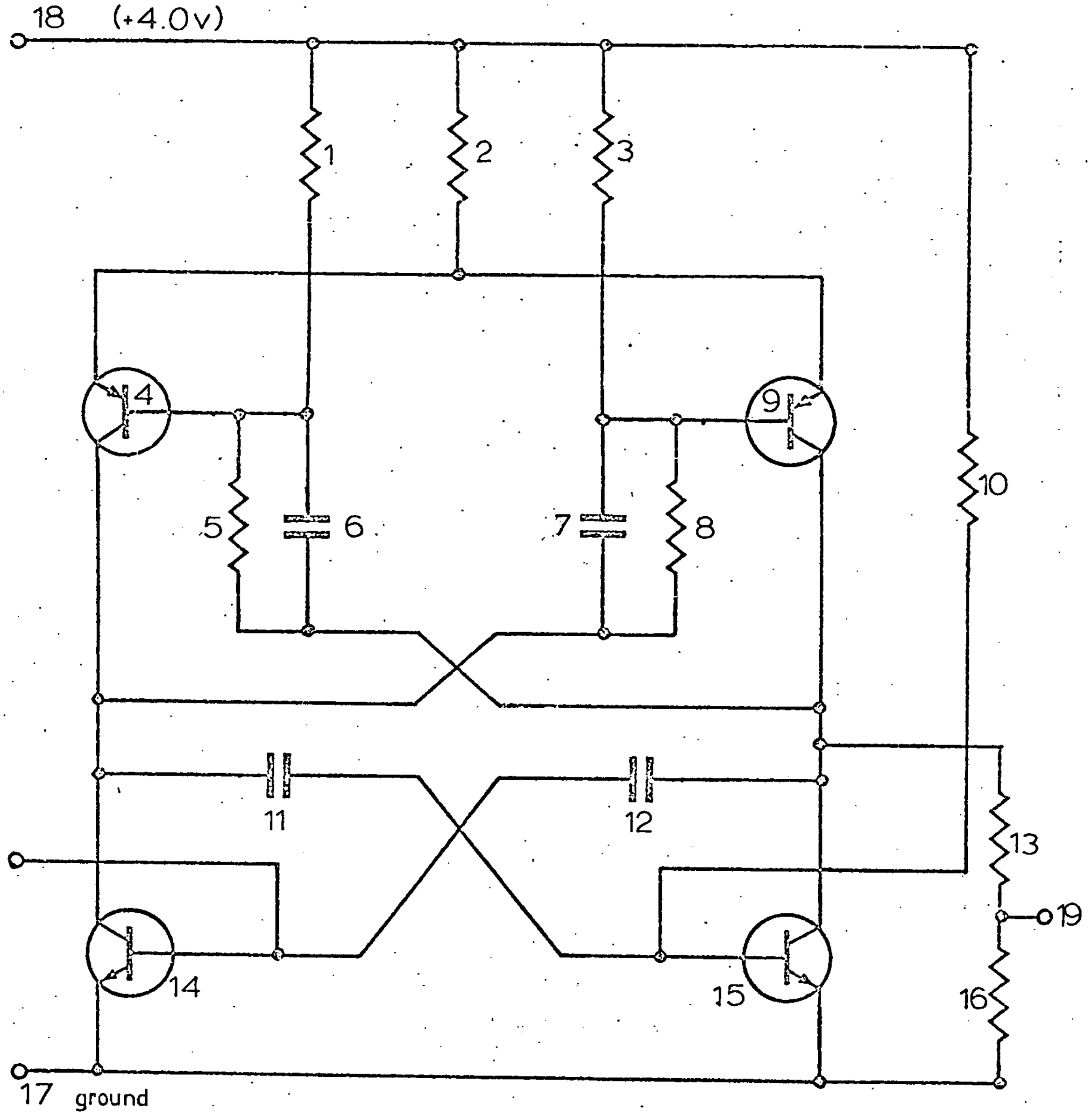
Table 6.3 contains the relevant figures indicating an increase of about 36%. This figure is quite acceptable in view of the quality of the manual design.

	Computer-aided layout	Manual layout
Time M/C (secs)	50	
Man (days)	.5	7(conservative estimate)
Layout Dimensions (thou)	10.4 x 11.6	6.8 x 13.0
Area (sq thou)	120.6	88.5

TABLE 6.3: Circuit (iii) Comparative Results

6.4.0 CIRCUIT (iv) - THIN FILM SYMMETRIC MULTIVIBRATOR

This circuit is taken from an avine-implantable temperature sensitive transmitter (Filshie and McGee, 1974). The schematic and input description are shown in figure 6.13 and the computer outputs in figure 6.14. The computer-aided layout is shown in figure 6.15 while the manual layout appears in figure 6.14. Originally, when the manual layout was first being constructed, the topological analysis, which was being tested, was applied to this circuit and, consequently, provided a basis for the manual design which was thought to be impossible to realise without a jumper wire. This application provided the first real indication that the programs were of use to microcircuit designers. Resistor outlines have been altered to occupy the available areas to maximum advantage in both layouts. The manual layout was



(a) circuit schematic

Figure 6.13 Thin Film Multivibrator - INPUT.

```
1, 140, 20, 1, 4, 18, 4, 5, 6
2, 20, 20, 5, 3, 18, 4, 9
3, 140, 20, 4, 4, 18, 7, 8, 9
4, 90, 60, 5, 3, 2, 1, 11
5, 80, 80, 4, 2, 1, 12
6, 50, 40, 6, 2, 1, 12
7, 50, 40, 4, 2, 3, 11
8, 80, 80, 6, 2, 3, 11
9, 90, 60, 6, 3, 2, 3, 12
10, 180, 100, 4, 2, 15, 18
11, 180, 60, 6, 5, 4, 7, 8, 15, 14
12, 180, 60, 7, 6, 14, 15, 9, 6, 5, 13
13, 180, 80, 6, 2, 19, 12
14, 90, 60, 4, 3, 11, 12, 17
15, 90, 60, 5, 4, 11, 10, 12, 17
16, 20, 10, 7, 2, 19, 17
17, 20, 20, 1, 3, 14, 15, 16
18, 20, 20, 1, 4, 1, 2, 3, 10
19, 20, 20, 1, 2, 13, 16
-1
```

Figure 6.13 (b) Circuit (iv) - Input Description.

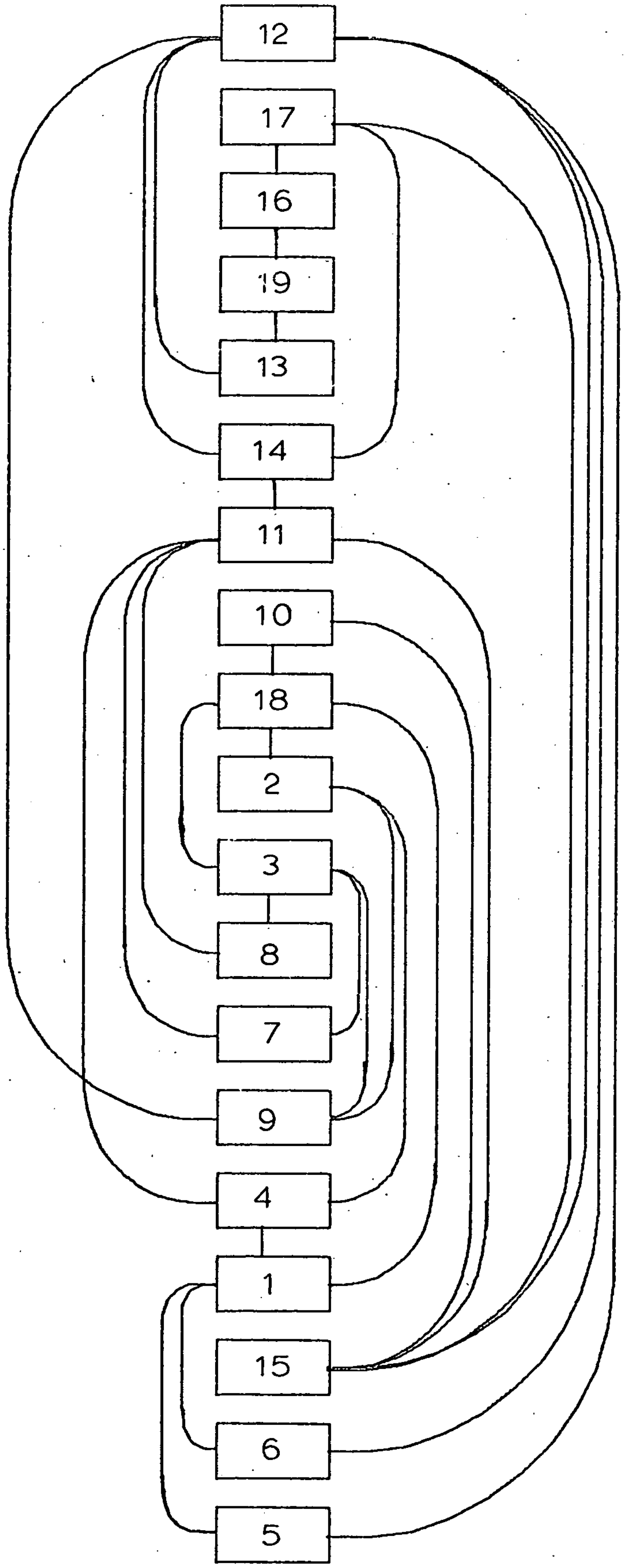


Figure 6.14 (a) Circuit (iv) - permutation.

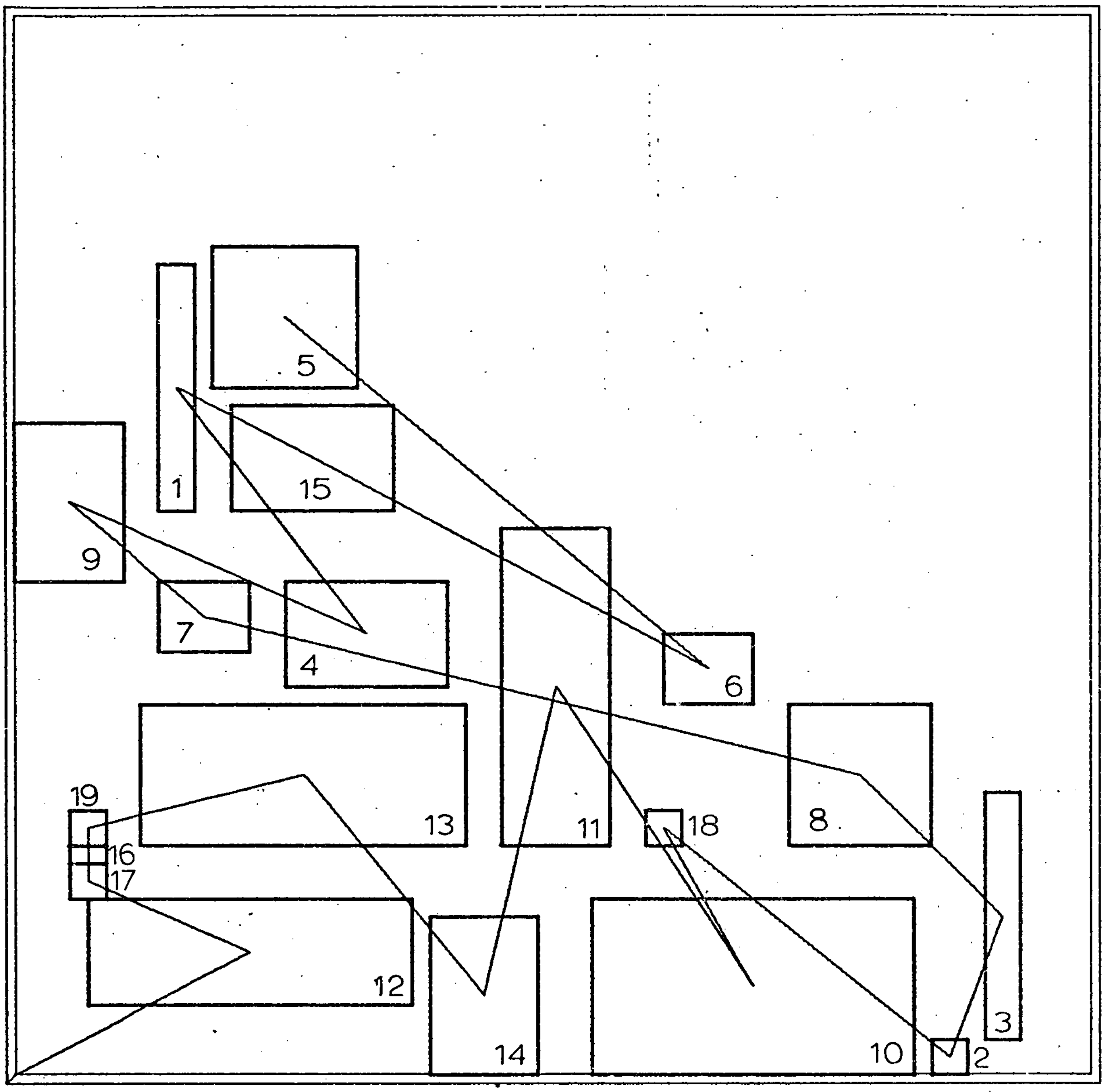


Figure 6.14 (b) Circuit (iv) - automatic placement

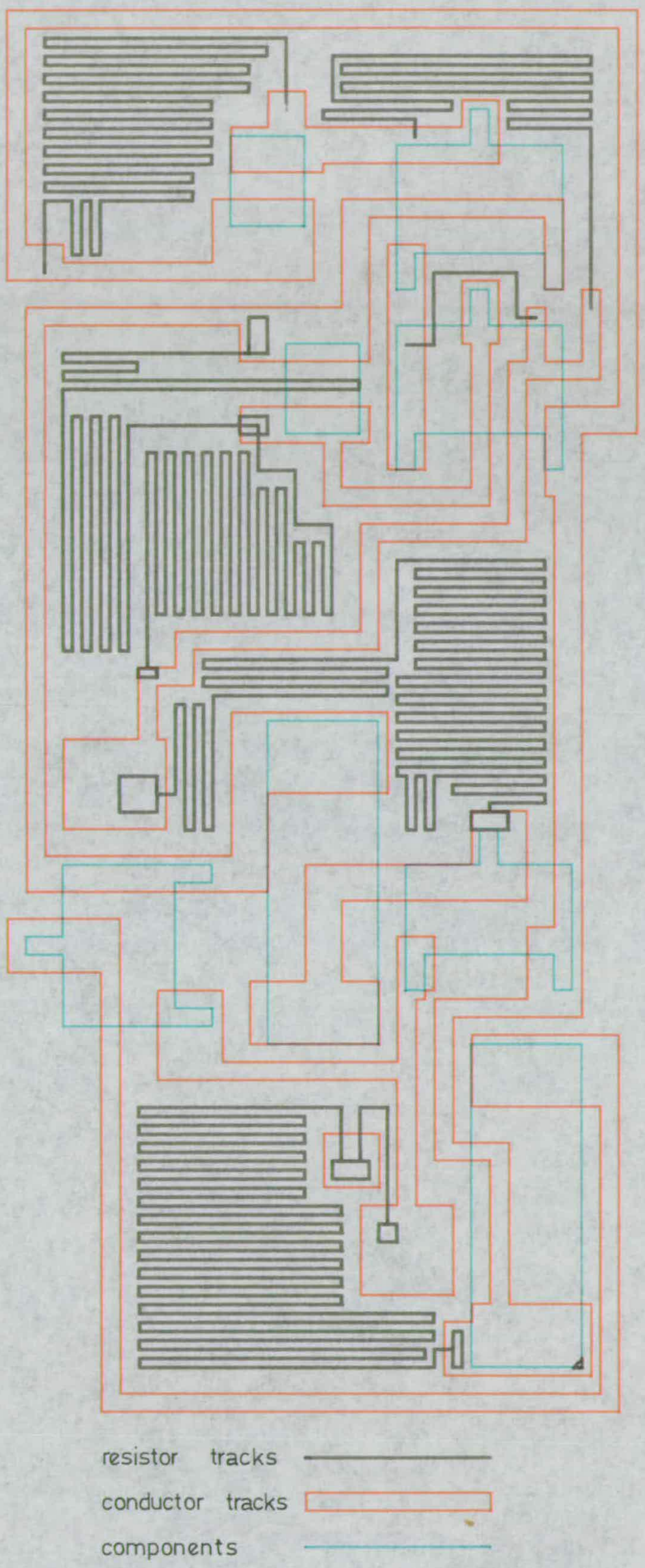
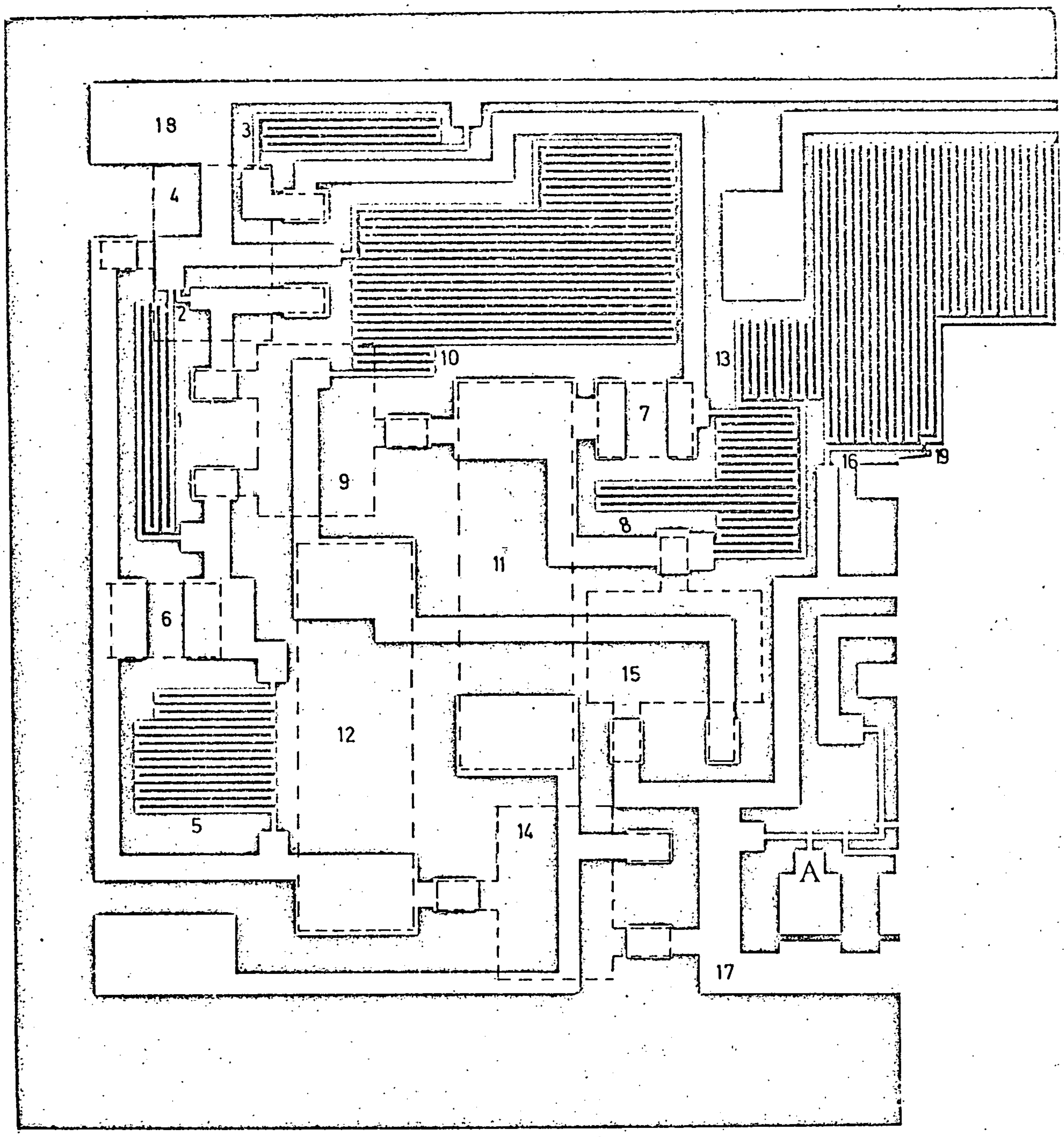


Figure 6.15 Circuit (iv) - Computer-Aided Layout



components are shown by dashed outlines.

Figure 6.16 Circuit (iv) – Manual Layout.

complicated by two factors, which were ignored in the modification process. First it had to appear on a standard 1 cm x 2 cm substrate and secondly it had to connect with standardised transmitter circuitry which occupied about half the substrate area.

6.4.1 COMPARISON

Table 6.4 contains the relevant figures which show an increase of 24.5%. This result indicates that the program provides similar results for different technologies, in keeping with the original specification of generality.

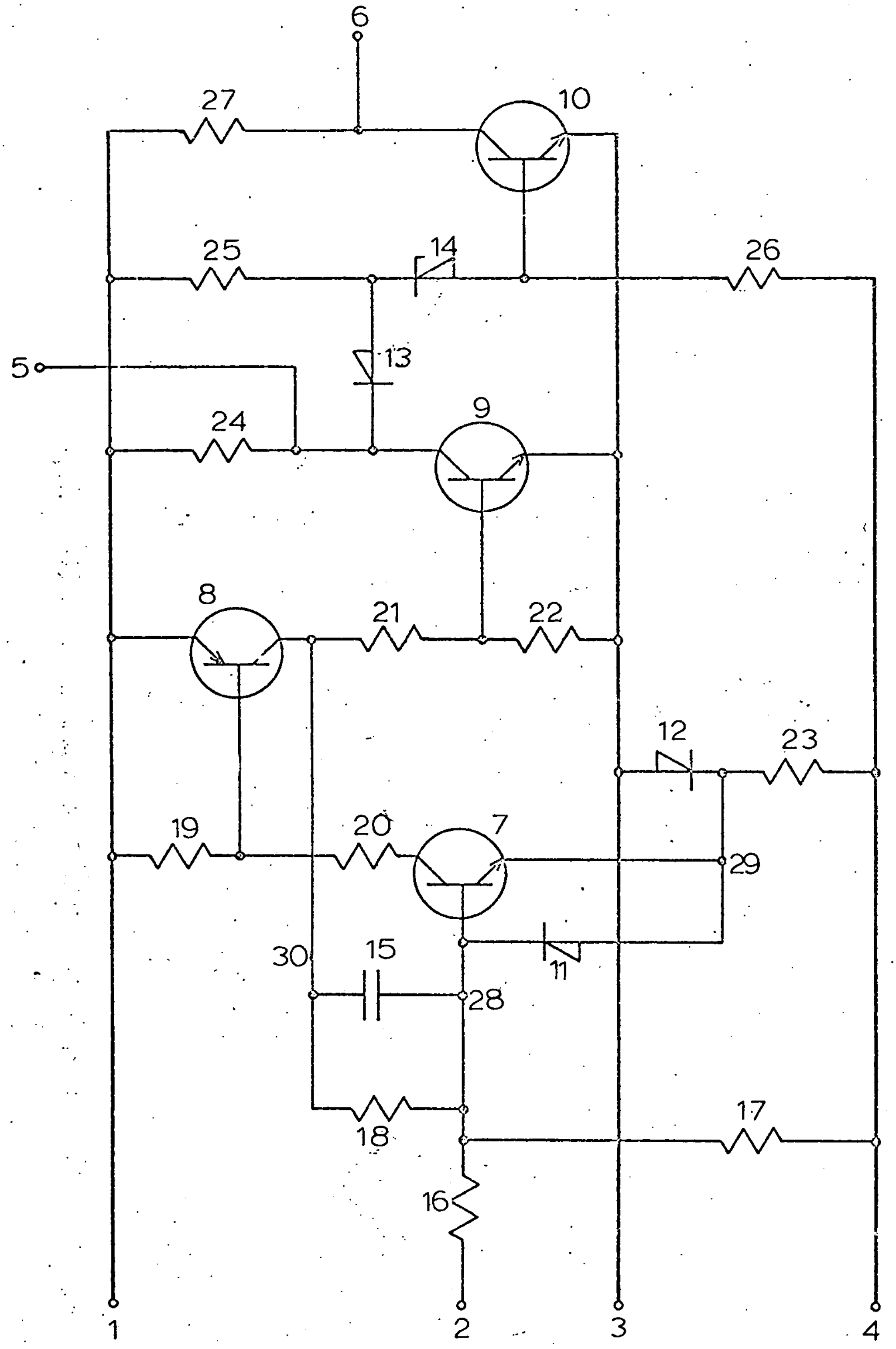
	Computer-aided layout	Manual layout
Time M/C (secs	24	
Man (hrs)	2	8*
Layout Dimensions (inches)	.35 x .8	.45 x .50
Area (sq inches)	.28	.225

* time excludes topographic layout (to remove crossovers)

TABLE 6.4: Circuit (iv) - comparative results

6.5.0 CIRCUIT (v) - PRINTED CIRCUIT BOARD

This circuit, although not strictly a microcircuit, was included in order to provide a comparison of this system with an automatic layout aid and interactive layout aid, as provided by Rose's program (Rose, 1970), which is discussed in chapter 2. The circuit schematic and input circuit description are shown in figure 6.17 and the computer results in figure 6.18. The three layouts shown in figures 6.19, 6.20 and 6.21 are respectively the manually-routed placement of figure 6.18(b), the fully automatic



(a) circuit schematic

Figure 6.17 Printed Circuit Board — INPUT

```
1,30,15,1,0,19,8,24,25,27
2,30,15,1,0,16
3,30,15,1,0,12,22,9,10
4,30,15,1,0,17,23,26
5,30,15,1,0,24,13,9
6,30,15,1,0,10,27
7,60,60,4,111,20,28,29
8,60,60,4,111,1,19,20,30
9,60,60,4,111,5,21,22,3
10,60,60,4,111,26,14,3,6
11,40,80,5,1010,28,29
12,40,80,5,1010,3,29
13,40,80,5,1010,5,25,14
14,40,80,5,1010,13,10
15,165,70,6,101,28,30
16,110,40,7,101,2,28
17,110,40,7,101,4,28
18,110,40,7,101,28,30
19,110,40,7,101,1,8
20,110,40,7,101,7,8
21,110,40,7,101,9,30
22,110,40,7,101,3,9
23,110,40,7,101,4,29
24,110,40,7,101,1,5
25,110,40,7,101,1,13
26,110,40,7,101,4,10
27,110,40,7,101,1,6
28,10,10,9,0,16,18,15,11,7,17
29,10,10,9,0,11,7,23,12
30,10,10,9,0,18,15,21,8
-1
```

Figure 6.17 (b) Circuit (v) - Input Description.

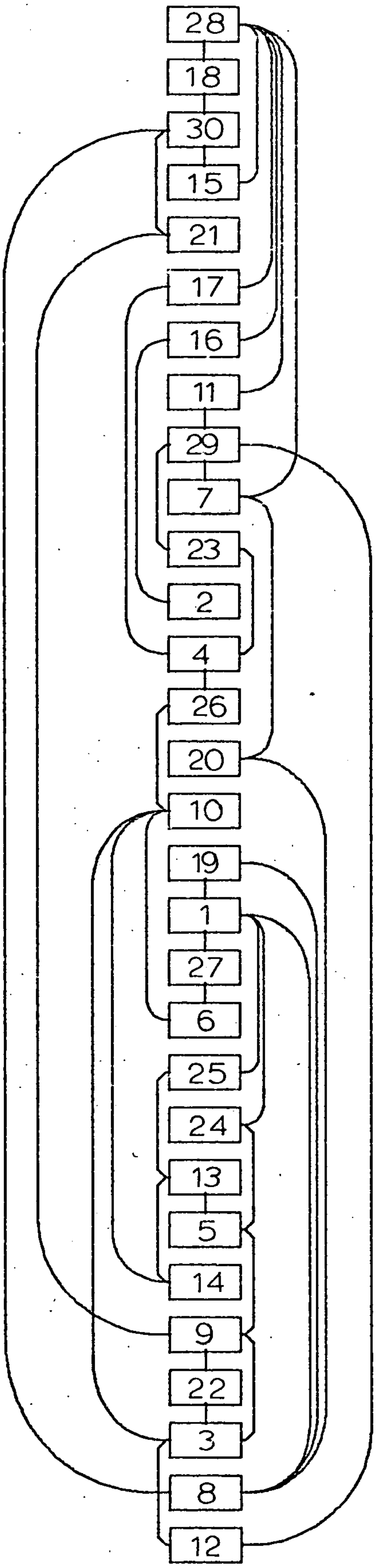


Figure 6.18 (a) Circuit (v) — permutation.

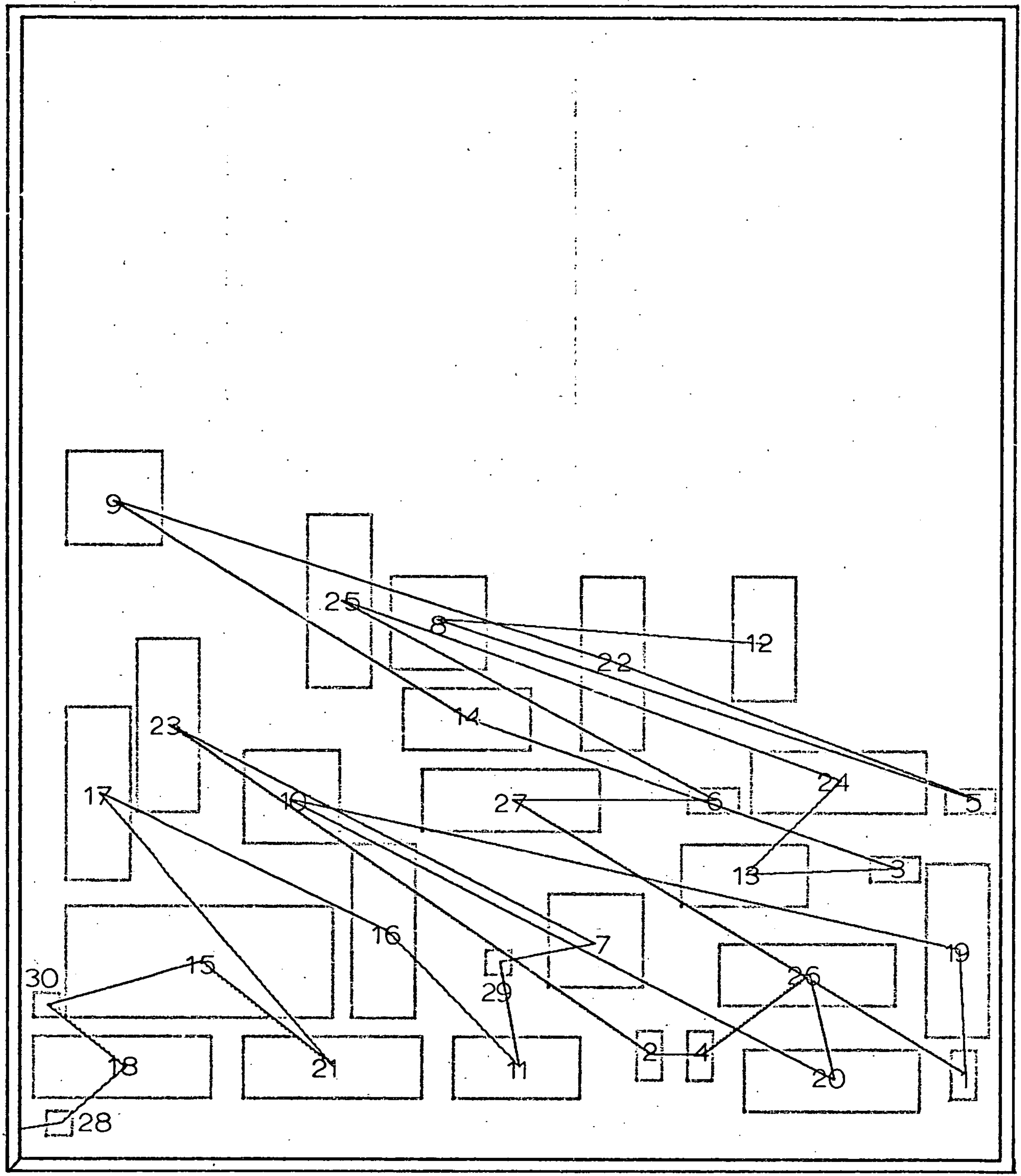


Figure 6.18 (b) Circuit (v) – automatic placement.

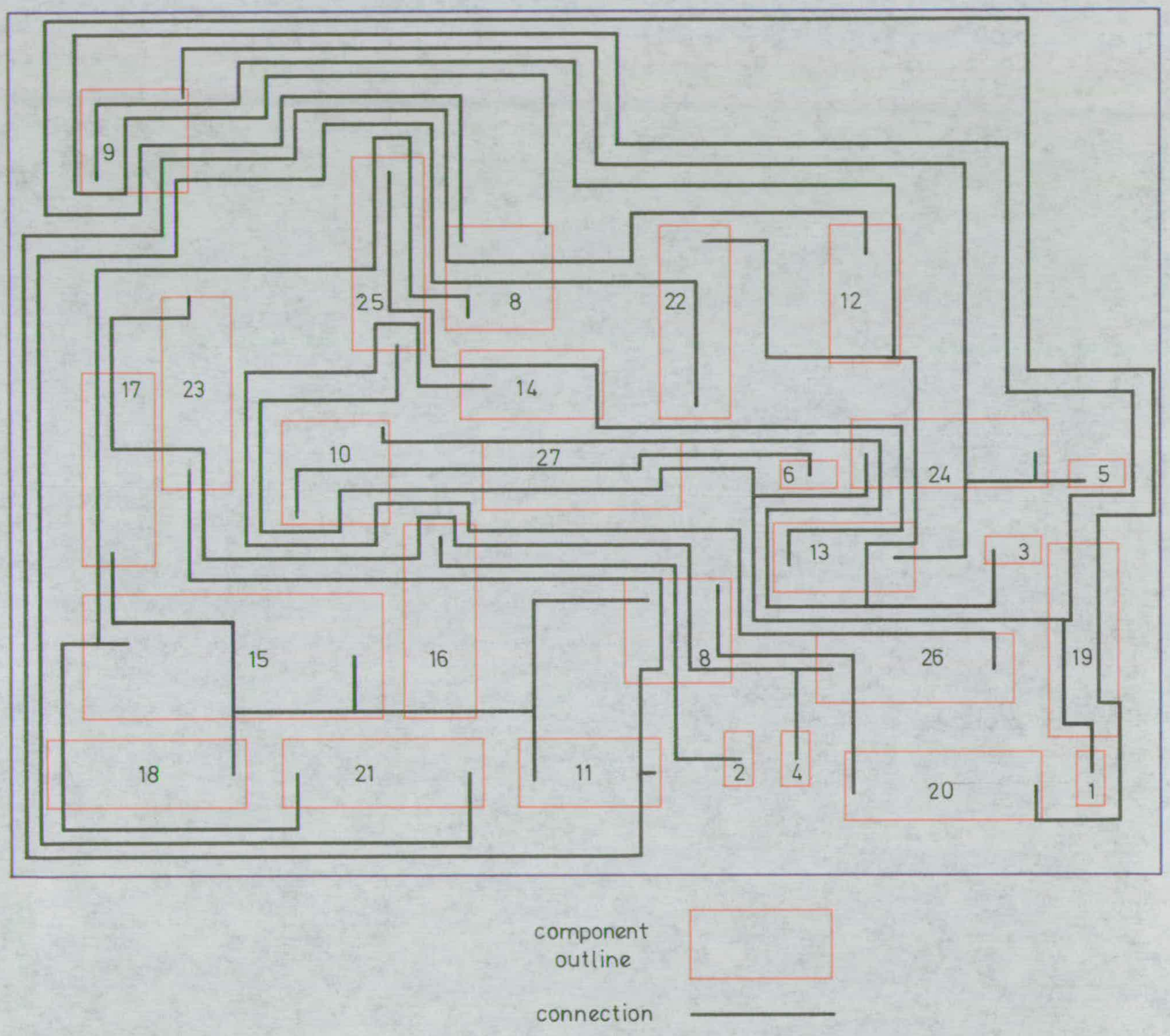


Figure 6.19 Circuit (v) - Computer-Aided Layout

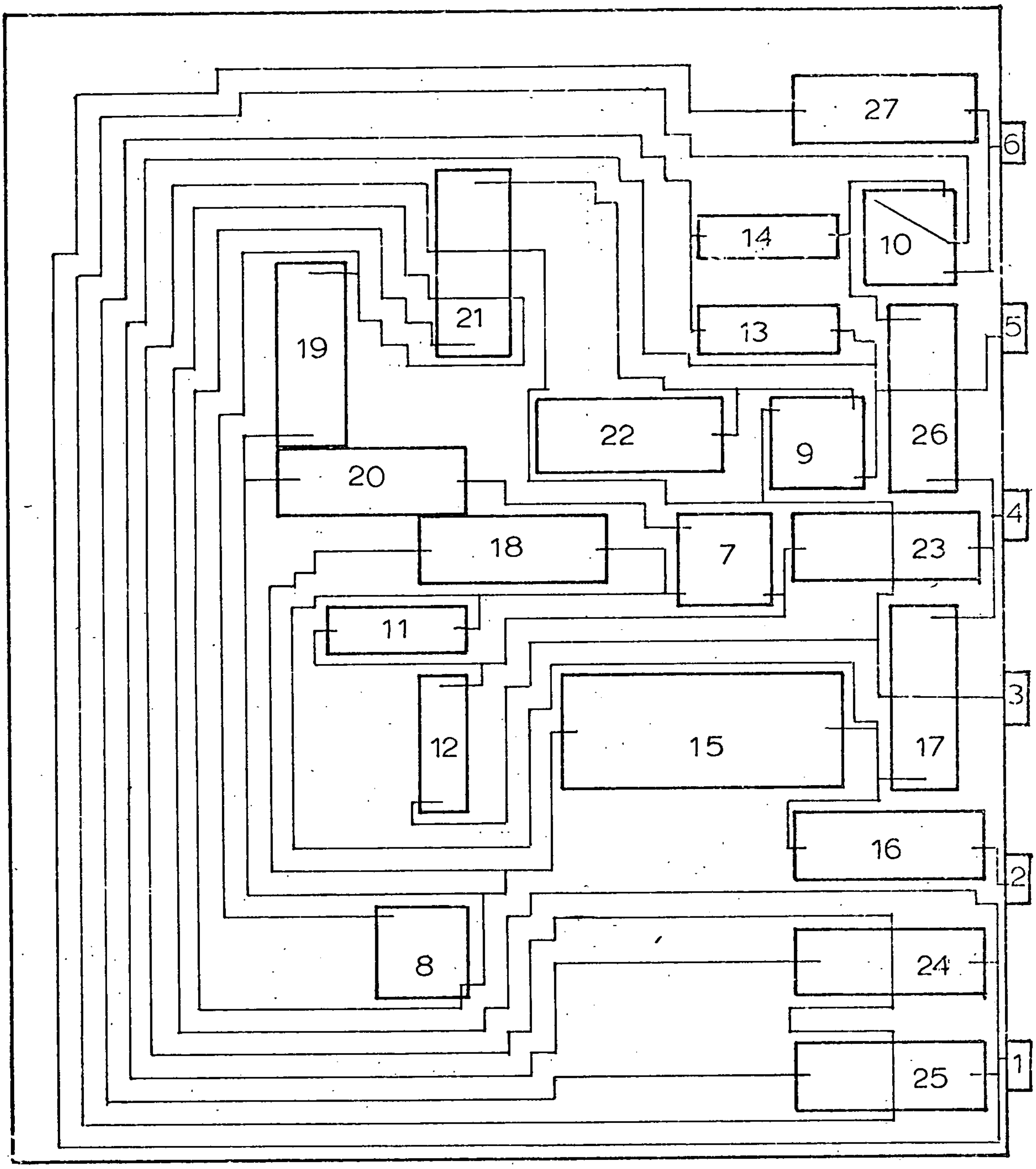


Figure 6.20 Circuit (V) – automatic layout (Rose's)

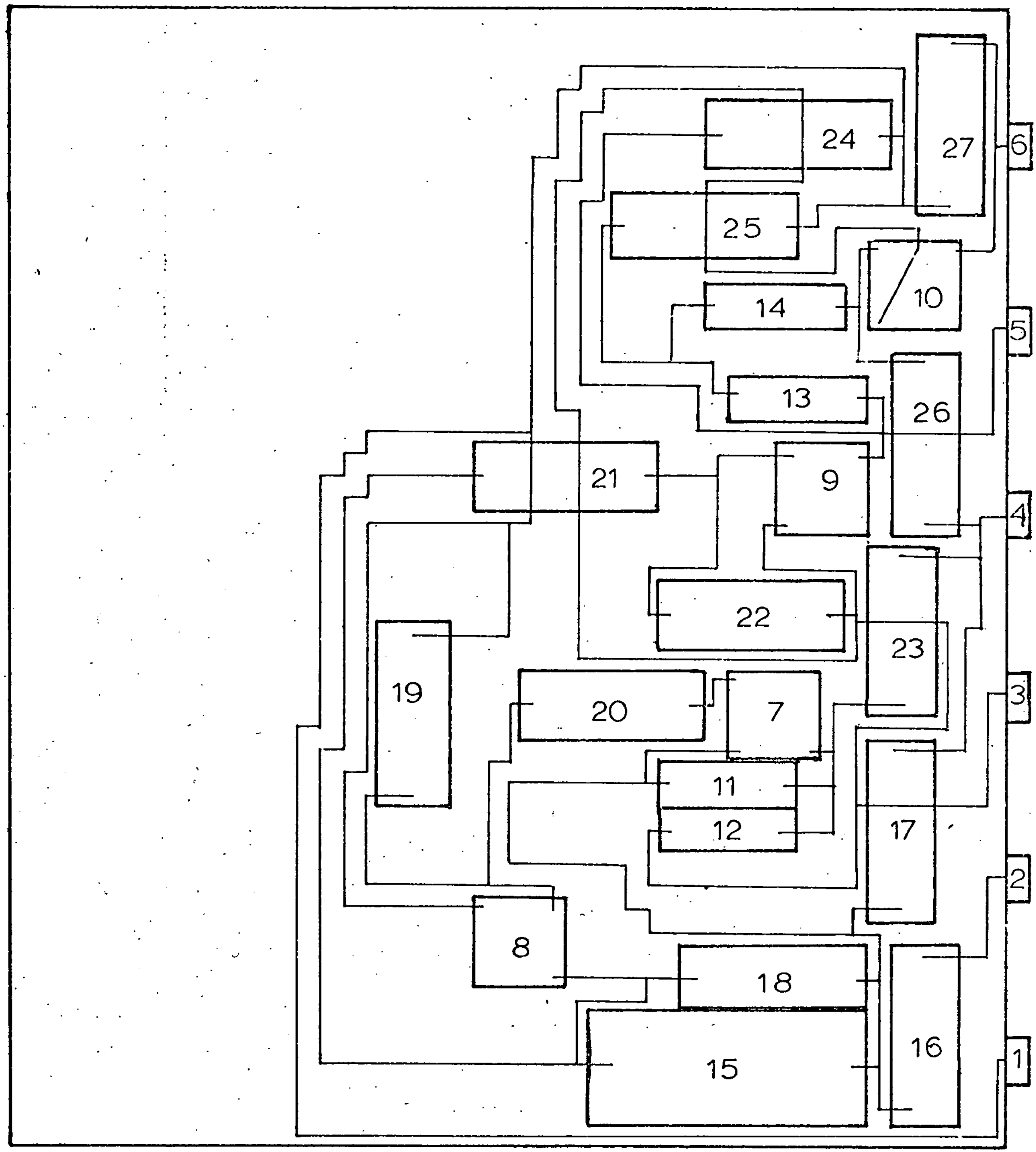


Figure 6.21 Circuit (v) — interactive layout.

layout and an interactively modified layout of figure 6.20 as produced with Rose's program. The computer-aided layout suffers from one defect only, the edge connector pads (components 1 to 6) do not all appear on the periphery of the layout area. However, it was decided not to modify this in order to provide an accurate comparison between the two programs.

6.5.1 COMPARISON

Table 6.5 contains the relevant figures which show a 21% decrease over the automatic layout and a 6% increase over the interactive layout. This indicates that the results produced by this system are comparable with those achieved with existing layout aids that have been designed for use with one specific technology.

	Computer-aided layout	Automatic layout	Interactive layout
Time M/C (secs)	48	13	
Man (hrs)	1.5	-	
Layout dimensions (inches)	6.2 x 4.7	6.5 x 5.5	6.8 x 4.2
area(sq inch)	29.2	35.75	27.6

TABLE 6.5: Circuit (v) - comparative results

6.6 SUMMARY OF RESULTS

The above results form the basis of this study of computer aids to layout. In all cases, except one, the computer-aided layouts were larger than their manually-constructed equivalents. However, in all cases, the time spent by an untrained person working from the computer outputs was about 20% of the time spent by a skilled designer. Therefore, the programs may be used to severely reduce layout turnaround time and, if the modification process is executed by an experienced designer, the quality of the layouts should not suffer too much. There will, of course, be a trade-off between the time spent on a layout and the quality achieved. However, it is not the purpose of this work to examine this complex interaction, that can only be done from within a manufacturing establishment and it is probable that different firms will have different opinions concerning the relative importances of quantity and quality. Nevertheless, these two factors are two of the most significant in costing microcircuit development and production, and the results show that a great decrease in the development time is obtainable through the use of this system without a great increase in circuit area.

The remaining chapters deal with improvements to the system which have been suggested through usage, and a discussion of the usefulness and future of automatic aids in microcircuit layout.

CHAPTER VII

SYSTEM ANALYSIS AND DISCUSSION OF IMPROVEMENTS

7.0 INTRODUCTION

This chapter comprises an analysis of the layout system which has been described in the four preceding chapters. The aim is to indicate weaknesses and omissions, in addition to drawing attention to areas which could be improved in both the algorithms and the overall method of constructing layouts. It would have been advantageous if the programs had been tested in a more exacting environment, such as within a microcircuit production company, however there was neither the time nor the opportunity for such a 'field-test'. Consequently, the assessment will be confined to features which have arisen during the course of development and testing. The discussion will be presented in several sections, dealing with the existing software, omissions (intentional and otherwise), the system structure and, finally, economic and user considerations.

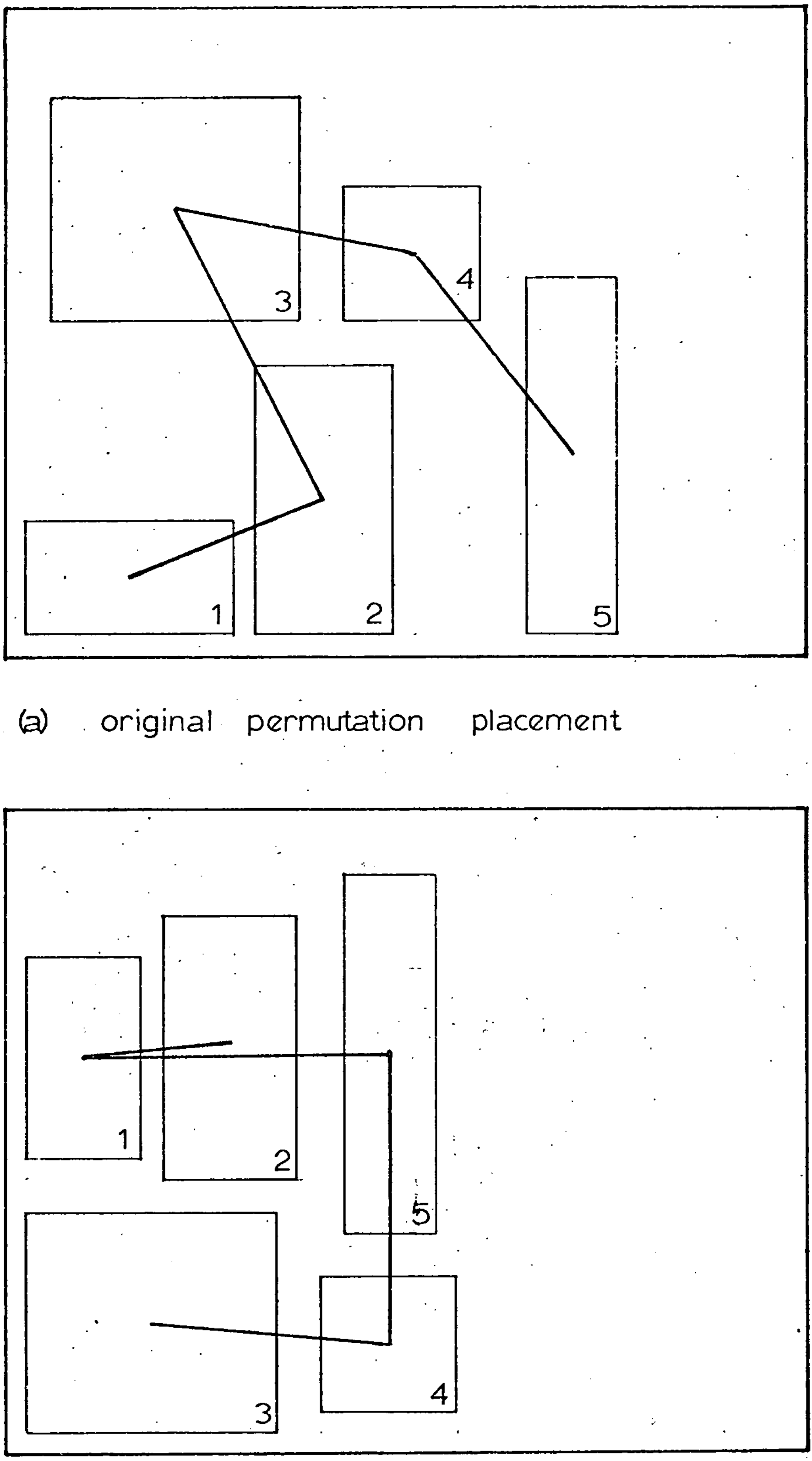
7.1.0 SOFTWARE LIMITATIONS

In analysing the software each algorithm will be treated separately. The aim is to indicate areas which could be improved, either by modification, addition or deletion of some subprocedure, and to indicate features which experience has shown to be of greater or lesser significance to the criteria of optimality (section 3.1) than was originally anticipated.

7.1.1 THE TOPOLOGICAL ANALYSIS

The topographic analysis deals with a graph representation of the circuit and so cannot directly affect layout area, nevertheless there is scope for improving the compactness of layouts. As has been demonstrated in section 4.2, altering the terminal nodes of the node-line without re-routing any of the links is a very simple way of producing a different, equally optimal permutation. Each different node-sequence will, in turn, produce a different placement and, as illustrated in figure 7.1, some will be more compact than others. The only certain method of deriving the best permutation from the viewpoint of compact placement is that of exhaustive trials, since it cannot be forecast, prior to placement, which node-sequence will produce the best placement. It is by no means certain that the potential improvement would warrant the increased cost of producing a placement for each different node-sequence.

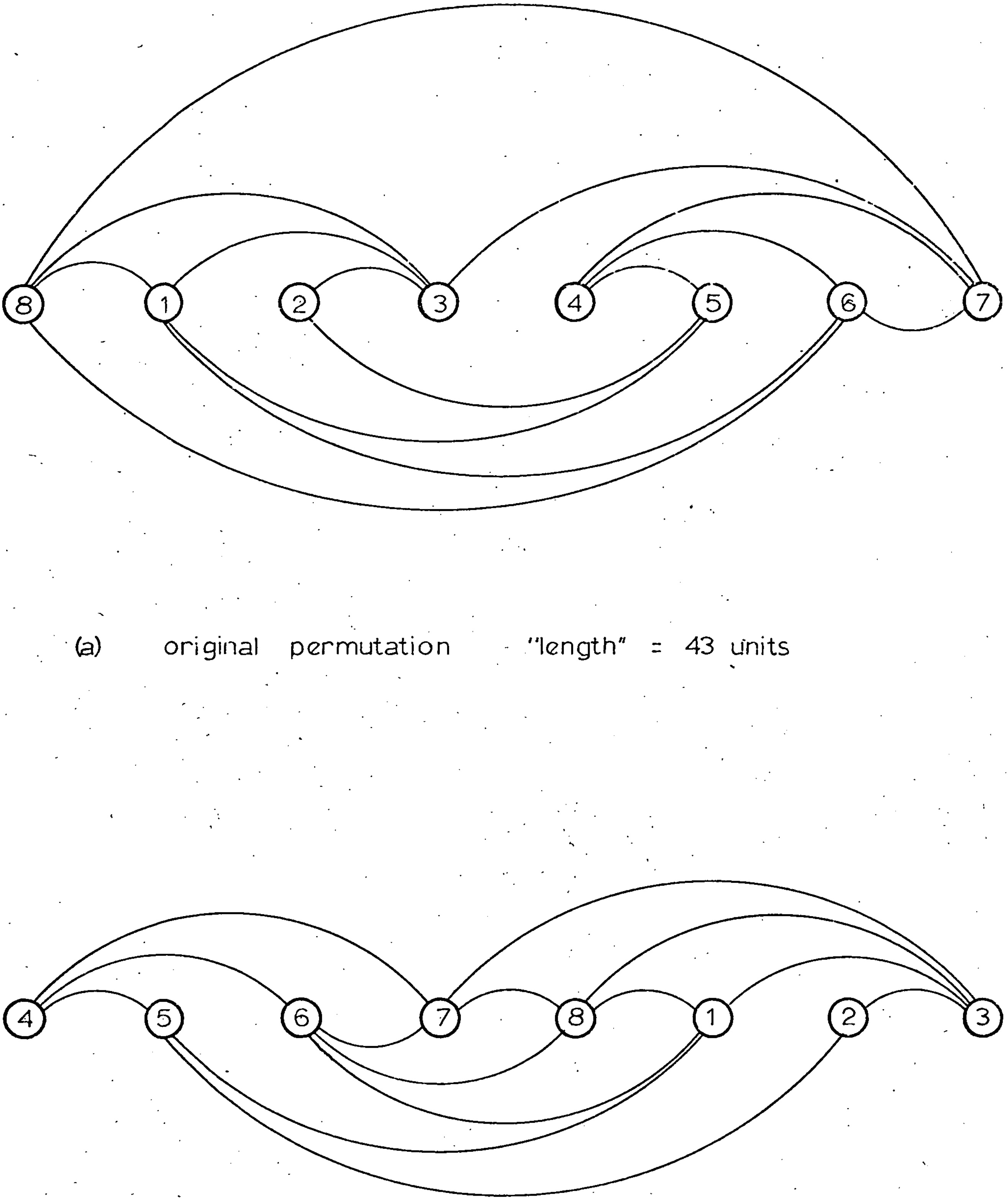
Crossover elimination has been fully discussed in chapter 3 and shall not be elaborated here as there is little or no scope for further improvements in that area. Routing length, however, can be improved with respect to the permutation in either of two ways. First, the terminal nodes of the node-line can be changed as shown in figure 7.2; secondly, individual nodes may be repositioned, as shown in figure 7.3. Both operations can reduce the topographic 'wiring length' (by 23% and 45% respectively in the above examples) and it is reasonable to expect that such reductions would be transmitted, at least partially, to the wiring length of the layout.



(a) original permutation placement

(b) placement with different starting point

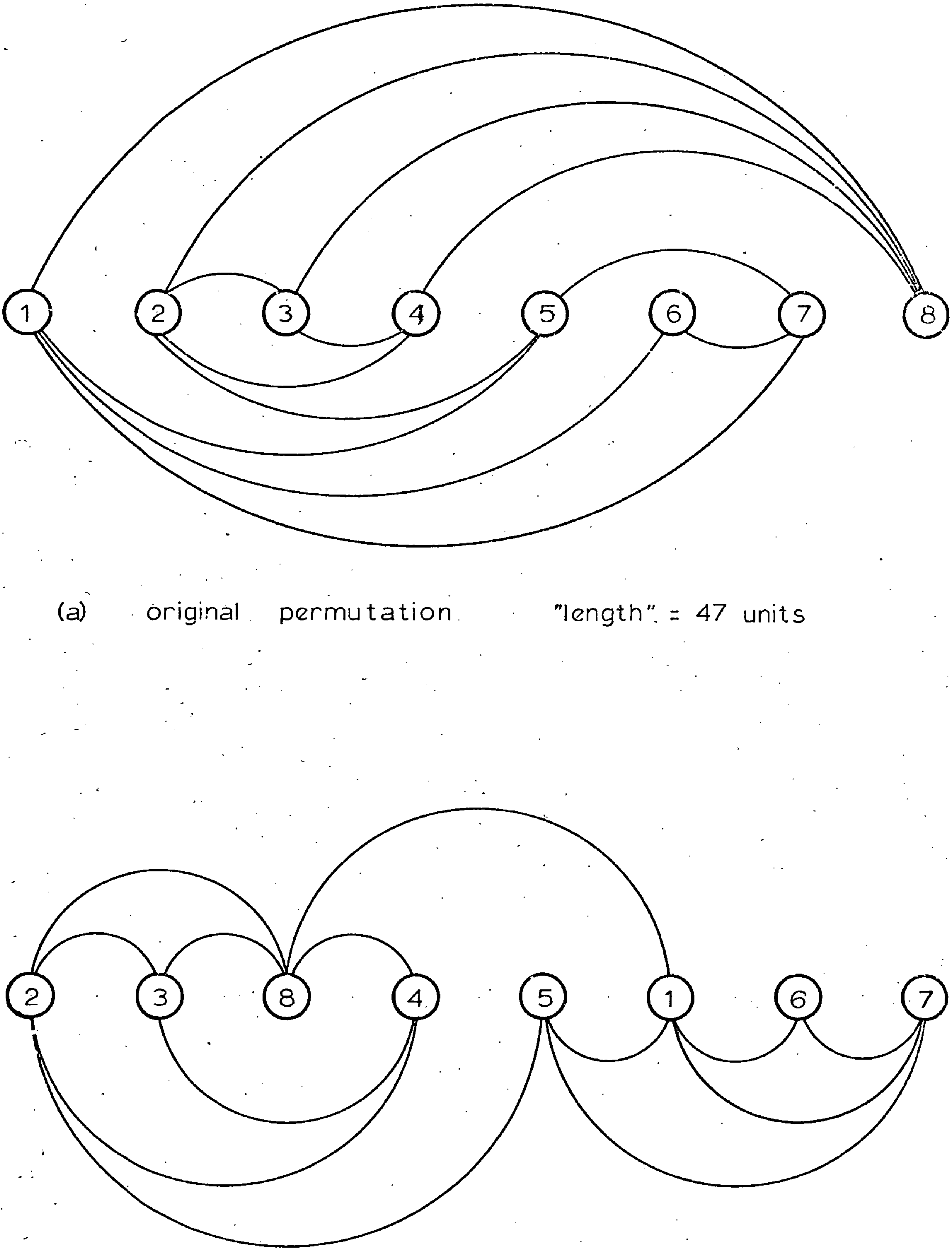
Figure 7.1 Increased Compactness as a result of changing Permutation Start-point.



(a) original permutation "length" = 43 units

(b) altered permutation "length" = 33 units

Figure 7.2 Reduced Topographic "Wiring Length" as a result of changing Permutation Start-point.



(a) original permutation "length" = 47 units

(b) altered permutation "length" = 25 units

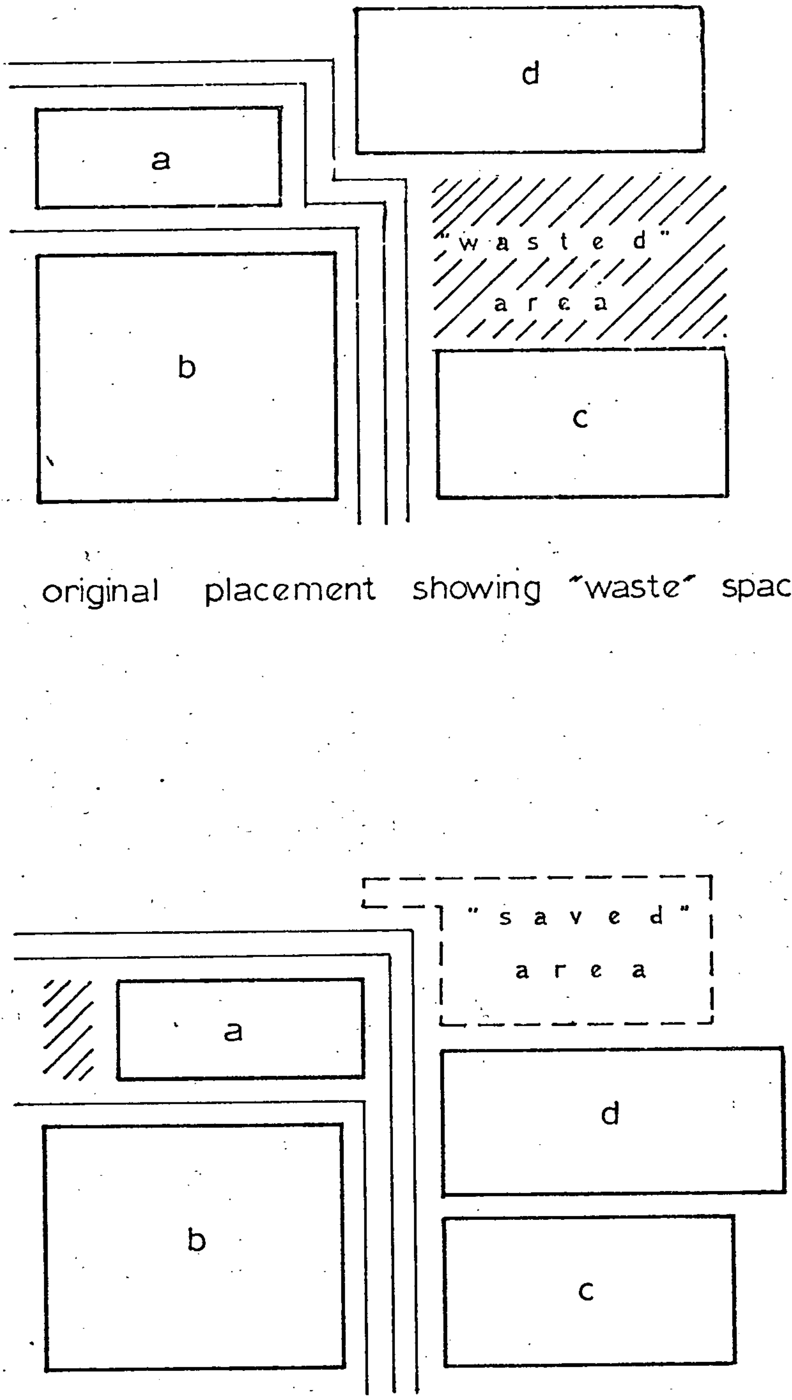
Figure 7.3 Reduced Topographic "Wiring Length" as a reult of changing the Node - order.

Thus, although the topographic analysis is concerned mainly with crossovers, it does afford scope for optimising layouts in other areas.

7.1.2 THE PLACEMENT ALGORITHM

The placement of components is the process which has the greatest affect upon the area of the layout. The total area is composed of the areas occupied by the components, their internal connections and external connections, such as bonding pads, in addition to unused areas required for electrical insulation of circuit elements and gaps formed by loose packing of components. In technologies which allow under-component wiring, there will be some overlap between these categories, however this does not directly effect this discussion. It is reasonable to expect that decreasing the area occupied by any one of the above categories would reduce the overall layout area (providing there was not a simultaneous increase produced elsewhere).

The dimensions of components and bonding pads are pre-specified and so cannot be altered; likewise insulation gaps cannot be reduced beyond the minimum defined by production tolerances. Some components, however, can have their shapes modified (section 3.1.1) in order to produce a better 'fit' with their neighbours. It is certain that this feature could be exploited to advantage during placement, as was found during the manual modification of the thin film trial layout (section 6.4). However, the best means of reducing overall area is to pack components efficiently, thereby minimising wasted gaps. For example,



(a) original placement showing "waste" space

(b) altered placement using "waste" space

Figure 7.4 More Efficient Space Utilization as a result of Manipulating unusable areas.

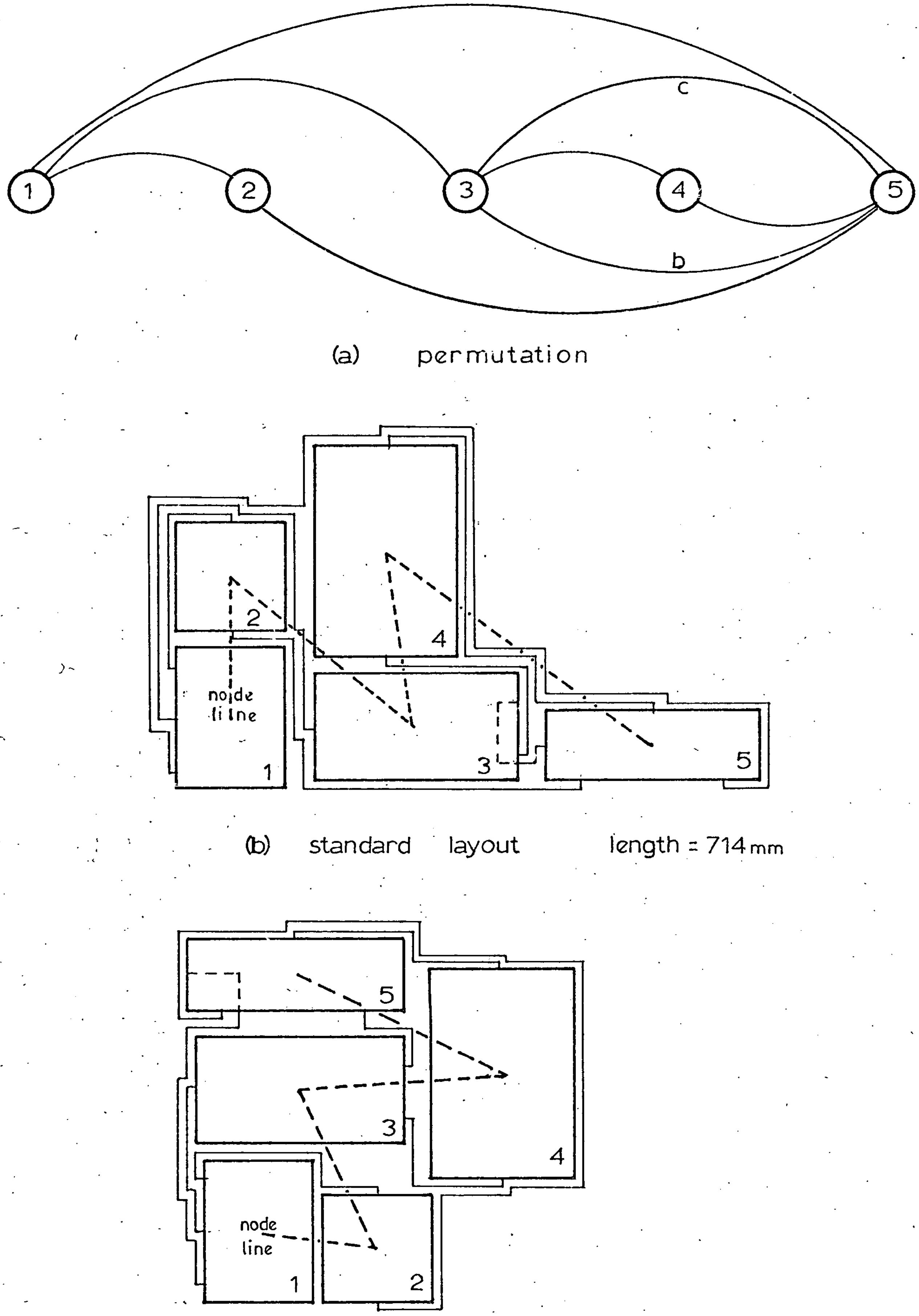
figure 7.4 shows a placement situation where component d is being placed adjacent to components a,b and c. Figure 7.4(a) shows a standard result, whereas in figure 7.4(b) component a has been moved (away from the origin) to align with the right-hand end of b, removing the track corners around component b. The two diagrams show how, by aligning two components to reduce track bends (equivalent to amalgamating perimeter segments as in section 4.5.2), subsequent placements may be made to produce a more efficient packing of components. This suggests that it would be advantageous to apply a 'shaking down' procedure, analogous to shaking a container to settle its contents, after placement so that components were simultaneously brought together while squeezing out the interstitial gaps not required for routing connections. Such a process might be better applied after routing, in order to avoid congesting tracks when components moved as a consequence of being shaken.

The routing and insulation areas are somewhat related in that every donductor requires insulation on both sides of its length (except, of course, where it connects with components). Therefore, since the connections are assumed to be of uniform width, track length is a contributing factor to the amount of real estate consumed by both the connections and the insulation. Thus the remaining avenues for reducing layout area during placement are concerned with minimising layout connection length through judicious positioning of components.

There are two main ways of reducing the length of a connection, either its end-points may be moved closer together, or it can be routed with a more direct path, if possible. In this sytem, however, connection paths are defined by the permutation and there is not generally the freedom available to redirect routes without seriously disturbing the entire routing configuration. Therefore, since the folds in the deformed node-line are the main factor which control both the distance between nodes in the layout and the convolutions of connections, the placement should produce a deformation which minimises distances and, more importantly, allows tracks to appear on the inside of folds rather than on the outside. Alternatively, the links could be reoriented (where possible) so that the interior of any folds was used to maximum advantage. This latter technique is, however, of limited application as has been pointed out in section 5.0.1.

The above techniques are illustrated in figure 7.5 which shows a permutation and two alternative placements. The first placement, figure 7.5(b), is standard and, as drawn, has a wiring length of 714 mm. Figure 7.5(c) is a placement which has been designed to produce optimal folding with respect to connection routing, in addition link (3,5) has been reoriented to avoid circumnavigating component 4. The resulting wiring length is 430 mms, which indicates a 37% reduction.

Thus it can be seen that the placement algorithm offers scope for optimising a layout by taking the length of connections into consideration, as well as their breadth.



(a) permutation

(b) standard layout length = 714 mm

(c) layout with optimized folding length = 430 mm

Figure 7.5 37% Wiring Reduction acheived by Optimized Folding.

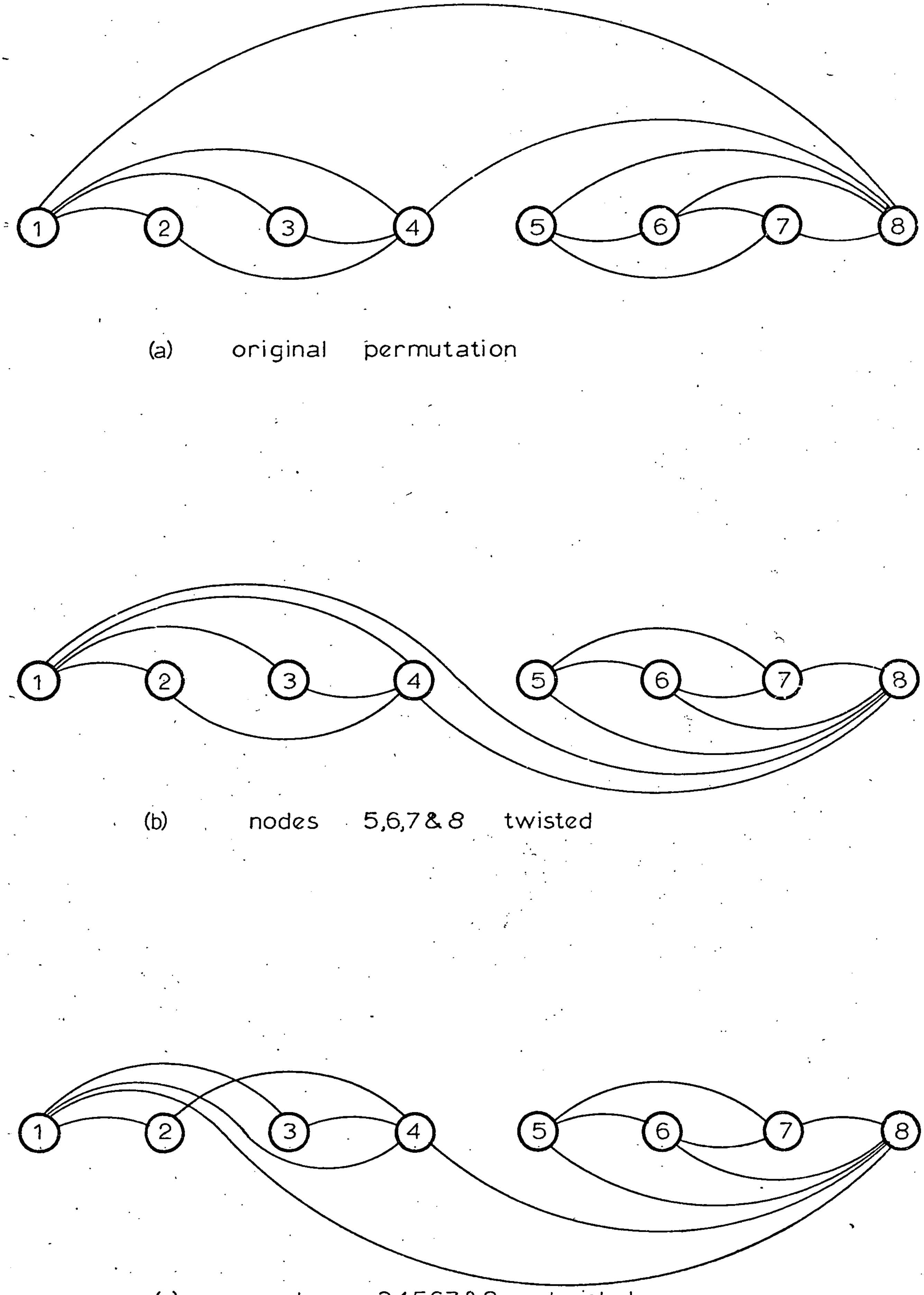
7.1.3 THE ROUTING ALGORITHM

Because the routing algorithm has not been implemented, its performance cannot be properly appraised and consequently deficiencies and improvements are difficult to locate. Nevertheless, there are some techniques which could improve the utilisation of folds in the deformed node-line. One example of such a technique is twisting the permutation so that one section assumes the opposite orientation. Figure 7.6 shows a permutation with two results of twisting; it can be seen that the technique can produce additional crossovers and links composed of double semi-circles, in addition to upsetting spatial reservations for connections. Thus this technique is also of limited application.

The basic structure of this layout system is such that there is little freedom of action for the routing algorithm, because connection paths are largely defined by the preceding algorithms. Thus improvements to the routing procedure are best included in the permutation and placement algorithms and have been discussed in the preceding sections.

7.1.4 THE PROGRAMS - A SUMMARY

There appear to be many ways in which the programs could be improved, however the above suggestions can only be verified (as being beneficial) by implementation and subsequent evaluation through comparative testing. Such action is beyond the scope of this work since the modifications are not fundamentally concerned with the basic study. Modifications to the overall systems' structure, which has been suggested through use of the programs, are



(a) original permutation

(b) nodes 5,6,7 & 8 twisted

(c) nodes 3,4,5,6,7 & 8 twisted

Figure 7.6 Node Line Twisting.

of greater interest. The next section of this chapter is devoted to a discussion of the possible areas for improvement of the underlying design of the layout aid, as an entity.

7.2.0 SYSTEM MODIFICATIONS

The individual programs, although by no means perfect, do perform their set functions (as specified in sections 3.1 and 4.1) satisfactorily. However, the specifications themselves are an important factor in determining effectiveness and evaluating performance in relation to providing realistic layouts. The most obvious weakness of the system is, probably, the features which have been overlooked. No automatic layout program can hope to take due account of every detail affecting layout, together with their prolific interrelationships. Nevertheless, this sytem excludes consideration of many significant features, the omissions having been justified on the grounds of limited program size and the trial nature of the resulting layouts. These will be examined in the light of results, as shown in chapter 6; they will be divided into two categories: circuit elements and interactions between circuit elements.

7.2.1 CIRCUIT ELEMENTS - OMISSIONS

The greatest omission in this class is concerned with communications between a circuit and the outside world, specifically lack of consideration of the requirements of bonding pads and edge-connectors. These elements must,for the purposes of realism, appear on or near the periphery of the layout. This requirement has been overlooked because it would probably prove to be too great a complication to allow the permutation procedure to remain effective; the neglect has been aided by the fact that the circuits have been considered in isolation, thus diminishing the significance of external factors. Some efforts were made to remedy this, chiefly consisting of introducing additional 'forces' which attract certain components to the boundary of the layout. Figure 6.18(b) shows the effect of this, that is, all the edge connector pads have been 'pulled' to the bottom right-hand corner of the layout. However, as can also be seen in figure 6.18(b), a subsidiary result is that the node-line loses much of its simplicity because adjacent components are pushed in different directions. Consequently, the treatment appeared inconclusive,that is, peripheral placement of bonding pads could be achieved only at the expense of some other feature (which had originally been assigned a higher priority). Nevertheless, as the results have demonstrated, little difficulty was encountered in routing external connections to the perimeter of the layout during the manual modification process.

Other circuit elements, which have been poorly incorporated, are contact holes and diffused tracks (ICs), under-component connections (film and PCBs) and elements whose length is fixed but whose shape is not (IC devices and film resistors). These elements have generally been overlooked because they only apply to a subset of the technologies under consideration and, as with the case of bonding pads, they can be dealt with much more effectively during the manual modification process. In addition, they are difficult to incorporate within the rigid structure of this system as defined by the specifications. For example, the best shape for a low aspect ratio device could not be specified when it was being positioned, because its neighbours would not all be present. To alter the shape once the device was surrounded would inevitably upset other placement calculations, which had been made in the interim.

These points suggest two major modifications to the system specifications. First, the placement algorithm should be more flexible in that assigned positions should not be made permanent until a component has become completely surrounded by neighbours. In other words, the algorithm should be allowed to alter previously computed locations of components when a subsequent component is being positioned, providing such alterations appear to be advantageous at the time. Moreover, to avoid such alterations producing congestion amongst the connections, both components and connections should be laid out together so that when a component is shifted, the affects on space allocations for connections can be calculated immediately. Although this course of action was previously considered to be disadvantageous (section 4.1), in retrospect it appears to offer a better solution in terms of producing realistic layouts of a standard comparable to that achieved using manual methods. This would require a single program solution for the layout execution, which would probably be too large for application on current commercial time-sharing bureaux. However this is a necessary consequence of producing realistic layouts, as opposed to trial layouts.

Secondly, it is inefficient in the long run to attempt to produce a generally applicable microcircuit layout aid, because important features which are not universally applicable are treated in insufficient depth, or not at all. It would be much better to construct layout aids with a strictly limited range of application if realistic layouts are to be produced. This is especially evident when one considers the crossover problem (section 5.1.0); if a program is to deal effectively with crossover elimination, as indeed it must for the purposes of realism, then it must be given every chance to exploit the peculiarities of the technology. Thus, relevant features must be treated fully and in depth; moreover programming space cannot be wasted on irrelevant features (which might be relevant to a different technology).

These points highlight the difference in constructing a realistic layout system compared with a trial one. Clearly, realism will require a greater programming effort, however the more factors taken into account by a trial layout program means the less effort required to convert the results into a satisfactory form.

7.2.2 ELEMENTAL INTERACTIONS - OMISSIONS

Omissions in this category are associated with visual and intuitive procedures of manual layout methods. They include parasitic effects, stray capacitance, repetitive requirements and neatness. There are no hard and fast rules for calculating the affects of interactions, only suggestions for keeping them to a minimum (see design rules, appendix II) and these are often in a form suited only for manual interpretation. Repetition, such as is required in shift registers (see section 6.1), is a feature which places severe constraints on the layout and therefore is difficult to incorporate in any automatic program without extensive modifications to the entire system. Finally, consideration of the neatness or regularity of a layout is impossible for a machine since the judgement is subjective and visual. Nevertheless, this feature is highly desirable for the purposes of checking and modifying layouts and is a factor which (as seen in chapter 6) frequently differentiates the manual layout from the computer-produced design.

These factors can only be effectively incorporated by introducing a human brain actively into the layout process. This would, by definition, destroy the automatic nature of any program, and would necessarily produce an interactive aid. Thus, in order to provide layouts of a standard equivalent to manual methods, the computer can only be used to assist designers rather than replace them.

7.3 SYSTEM ADAPTABILITY

An important feature of modern software is adaptability or transportability; since new hardware releases are made with increasing frequency, software must be capable of adaptation to the new equipment or else it faces the danger of becoming obsolete. Hardware trends are difficult to forecast with accuracy since they have a tendency to develop in discrete jumps. However, the major current trend, in layout equipment, is towards graphic display units. The storage tube is the most recent addition to computer graphics and has the advantage of being considerably cheaper to purchase than the continual-refresh type of display. To provide some experience of this system's adaptability as a measure of its extended usefulness to microcircuit layout, the programs were converted for use on a Tektronix 4010 storage tube attached to the PDP-10. The transfer proved relatively straightforward, since most of the basic drawing routines were available on the system (Sturgeon: User's Manual, 1972). This established that the programs were at least adaptable. The most noticeable affect was naturally the speed with which results could be displayed and this, in turn, gave rise to a secondary result which could prove to be of great significance.

Because the screen can present results in a fraction of the time required by a plotter, many trials can be run and superficially evaluated at the tube face in the time taken for one using a plotted display. This means that the control parameters (sections 4.6.7 and 4.7.1) can be exploited to considerable advantage. Specifically, it would be possible to generate and roughly evaluate up to 30 different trial layouts of one circuit per hour by manipulating the control parameters. Therefore it is feasible to establish the parameter settings which produce the best results for an individual circuit by exhaustive trials,and these particular values can be re-run to reproduce the better layouts on a hardcopy device (such as a plotter) for more extensive evaluation at the designer's leisure. This method would permit the plotter to draw only those layouts which had already been observed to be of above-average merit.

The above suggestion would make the programs a more powerful tool in the designer's hands since he gains a degree of quality control over the plotted results. The resulting programs would remain automatic in essence but the method of use would display similarity to an interactive system. There was, unfortunately, not sufficient time for a deeper investigation of the potential of this hybrid system, however a discussion of the advantages that such a method of approach would offer to microcircuit CAD is presented in chapter 8.

7.4 ECONOMIC CONSIDERATIONS

There are two major factors which determine the worth or economic value of a layout aid; the first is the cost of using the aid and the second is the value of the results obtained from the aid. This system has proved quite inexpensive to use. The examples included in chapter 6 would cost less than £8:00 each (excluding equipment hire) to produce at commercial rates,* with an additional requirement of less than 1 hour per circuit for data preparation and punching.

The results require manual translation and refining, however they can save a considerable amount of manual layout effort, up to 40 man hours in the case of the clock driver cell (section 6.2). If used by experienced designers, the system would constitute a valuable layout aid, although its precise value could only be estimated by thorough field testing.

*approximately 5p/sec CPU time and £5/hour terminal connect time

7.5 CONCLUSIONS

The system provides a rather limited, but nevertheless valuable, aid to the layout of microcircuits by producing a variety of component placements with a related wiring guide cheaply and quickly.

Because the permutation procedure execution time increases exponentially with the number of circuit elements, the system, as it stands, would be less efficient when applied to large circuits. However, this difficulty could be overcome by a cellular design approach, that is, by partitioning a large circuit into a group of subcircuits, each of which could be laid out individually.

The results indicate that automatic layout aids do not necessarily require large, dedicated computers; they can be effectively mounted on limited hardware and are therefore feasible for use by firms with small CAD budgets. The next chapter discusses the conclusions reached and their implications on the future of automatic layout.

CHAPTER VIII

CONCLUSIONS

8.0.0 LAYOUT AIDS - SUMMARY

The programs which have been developed during the course of this project have established that effective automatic layout aids can be incorporated in a design process despite a limited CAD budget. However, it is also important to delineate the characteristics of such low-cost aids. Up until now the classification of aids has consisted of two groupings, automatic and interactive; in the light of experience gained in using a storage tube to display results (see section 7.3) this classification has proved inadequate. Consequently, the following sections of this chapter comprise discussions of layout aids, which will outline the advantages of combining certain features of automation and interaction to form a third, hybrid class of layout aids. The last sections include a brief discussion of a comprehensive CAD process for microcircuit layout and a summary of the project.

8.0.1 THE AUTOMATIC AID

Automatic aids do not require human supervision during layout construction and are consequently fast because they can work at their own speed. Moreover, they are convenient to use in either batch mode or conversationally via a remote teletype. On the other hand, they cannot embody the experience, intuition, perception and imagination which are the fundamental tools of a human designer and so automatic programs cannot produce layouts of the same standard without manual assistance. Therefore there exists a trade-off between the quality available through manual methods and the quantity available with automatic programs when applying the two techniques to microcircuit design.

8.0.2 THE INTERACTIVE AID

With the advent of the graphics screen, visual interaction between man and machine became possible on a scale which has made circuit layout possible, using a graphic display instead of a drawing board. Thus interactive layout aids came about and today almost all commercial layout software is based on the interactive graphics screen. The advantages are obvious, human designers can produce manual-quality layouts much faster and more accurately; moreover artwork can be prepared automatically from the computer's layout description. The disadvantages are the high capital cost of the equipment and the basic inconvenience of adapting to a new technique; a fuller discussion of the problems associated with interactive layout is included in appendix I. Nevertheless, many firms believe that CAD offers the only means of remaining competitive in the field of microcircuit production and interaction has proved more practical than automation.

8.0.3 THE THIRD ALTERNATIVE

A significant feature of the layout programs developed for this study is the ability to manually set parameters which control the component placement decisions. Although the affects of manipulating the parameter values are not consistent (see section 4.7), they can be used to provide a wide range of layouts with little effort. There is considerable advantage in such a scheme. In order to illustrate its potential, a system is proposed below which forms the basis of a discussion of the advantages of remote-control of automatic layout aids.

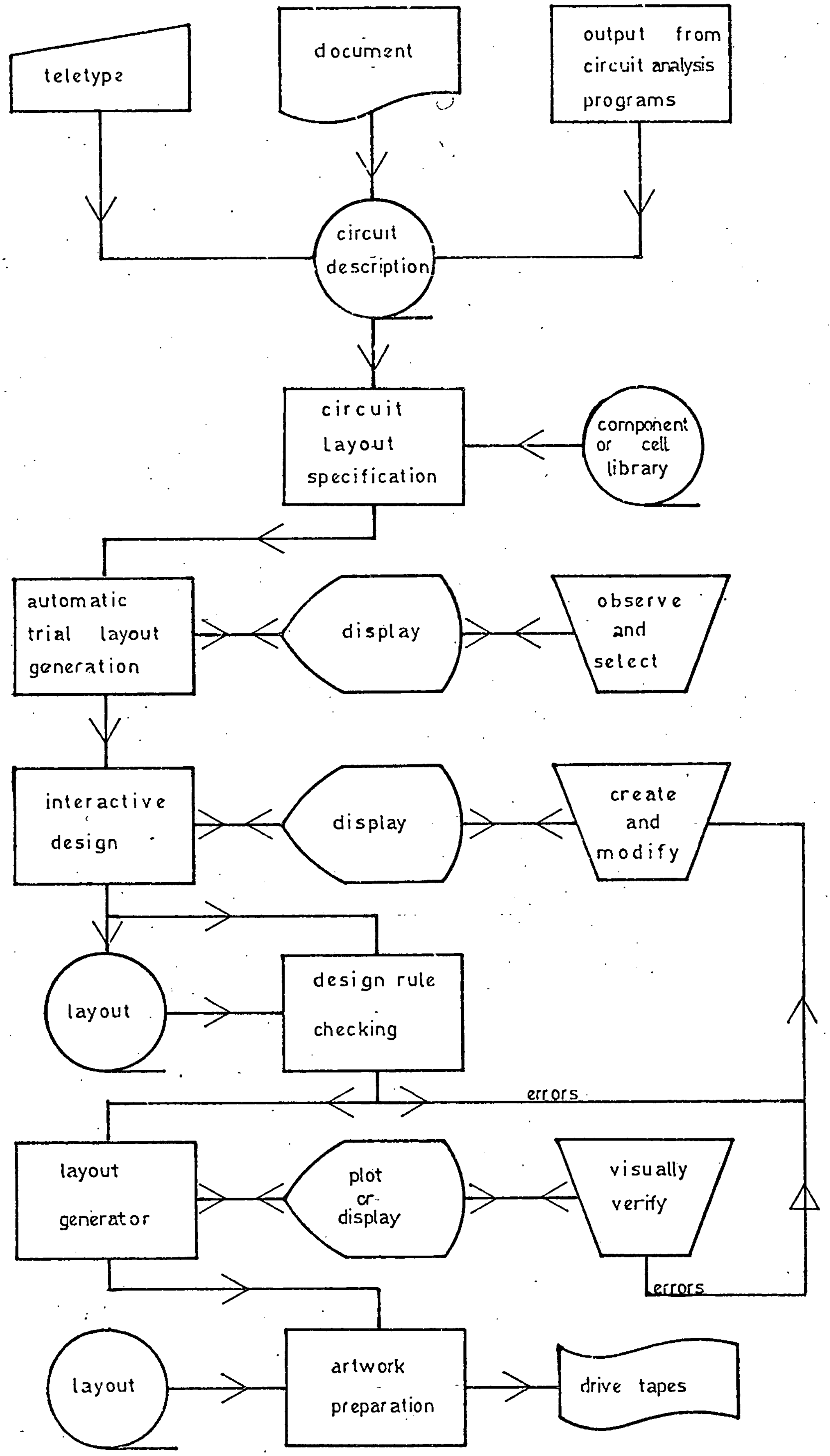
A suitable system would consist of automatic layout programs, which have been modified so that significant decisions are controlled by a set of parameters, which dictate the relative preferences between alternative courses of action. The parameter could be preset via additional input for batched operation with hardcopy output, or set conversationally for operation over a remote terminal with a non-interactive display facility. The programs need only recognise the significant factors affecting layout decisions; the relative importances and inter-relationships would be supplied by a designer via parameter settings. Thus the system would allow indirect manual control so that a designer could supervise the layout process without having to spend time at the screen designing the layout himself.

The advantages of the above system are twofold. First, because the programs operate automatically, all the desired features of automation, that is, speed, convenience and economy, are incorporated. Secondly, there is a degree of quality control, similar to that of interaction, without any of the disadvantages associated with interactive layout programs. Because such a scheme could provide many different layouts,it would be desirable to provide a means of retaining portions of one layout to combine with complimentary portions of other layouts. Thus it would be possible to form a high-quality layout by combining the best features of several average-quality layouts. The combination process could be executed manually or interactively by a draughtsman,or even by an additional automatic program.

The above proposed scheme provides a third approach to constructing a layout aid. Because it is based on automatic programs, it also suggests a means of conveniently and inexpensively increasing the power and flexibility of existing automatic layout aids. However, before such a scheme could be fully implemented, it would be advisable to gain a deeper understanding of layout in order to use the parameters to full advantage. Ideally, such a system should be built by a collaboration between microcircuit designers and programmers in order to combine layout expertise with programming skill.

8.1 A COMPREHENSIVE LAYOUT SYSTEM

It is appropriate here to present some comments on what is felt to be an ideal layout system based on CAD. Figure 8.1 shows such a system in flowchart form. The aim has been to apportion tasks between man (trapezia) and machine (rectangles) so that each performs those tasks at which he or it is superior. Manual handling of data is minimised by a system of files which also allow each



Figur 8.1 The Ideal Microcircuit CAD System.

sub-process independant operation. Although the main design steps are included interactively, it is envisaged that this would consist of combining features of various layouts, supplied by a trial layout generation program as proposed in the preceding section. Although this system would require a dedicated machine with interactive hardware, it would be feasible to use a mini-computer with disc store which, if equipment prices continue to fall, could well prove to be an economic proposition for small microcircuit firms.

8.2 SUMMARY

This study has been based on the construction of a practical, non-interactive layout aid designed for a minimal expense. The immediate results have shown that effective layout aids do not, as was previously thought, require a large and powerful computer, nor do they require expensive display hardware. This result will be of interest to many of the smaller microcircuit firms who feel that CAD capability is necessary to compete in today's custom-design environment, but cannot afford to acquire expensive equipment. In particular, the indication is that much effective assistance can be provided from time-sharing bureaux or a mini-computer. Of secondary importance to the same people will be the conclusions reached in chapters 7 and 8, which will be of assistance in evaluating the requirements of a CAD system within an already-established manual design process.

Finally, the proposed method of approach developed in section 7.3 and described in section 8.0.3 will be of interest to all concerned with constructing layout aids as a means of combining the power of interaction with the low cost of automation. Moreover, the concluding chapters have indicated some of the major pitfalls in constructing a practical layout aid and this will be of use to people facing a similar task.

APPENDIX I

This appendix consists of a brief description of a commercial, interactive PCB layout system which includes some automatic features, followed by some users' comments on the system and its usage. The layout system comprises Redac's Redal-3 program mounted on a 32K PDP-15 with a VT 15 graphic display. The source of the users' comments is the British Aircraft Corporation's (BAC) report R84-C1. The purpose of this discussion is to highlight some of the problems associated with converting a manual design process to one incorporating interactive CAD.

THE SYSTEM

Redal-3 is a suite of programs designed for laying out PCBs with a mixture of discrete components and packaged ICs using single or double-sided wiring. There are eleven main routines which the operator may select from the light button menu, four are associated with data input, output and modification and presenting the layout on the graphics screen. Two additional routines are used for manual layout, one for modifying component positions, the other for modifying connection routes. The remaining five routines all contain some automatic features, as follows:

PLACE: an automatic centre-of-gravity placement routine which gives a guide to the best possible placement

ROUTE: an automatic routine for Manhattan routing

AUTORTUTE: a series of automatic, semi-automatic and manual routines for minimising crossovers

MANHAT: an automatic routine for checking the spacing of routes

RECONN: a routine which sorts nets for minimum connection trees

During operation the screen is divided into three areas: the working area, comprising the left hand 3/4 of the total screen area, used for layout display and construction; the message area in which the computer displays messages to the operator,and the light button menu which is displayed below the message area and which is the means whereby the operator controls the selection of routines. When a routine is selected from the menu, it causes a submenu for that particular routine to replace the main routine menu. The submenu is used for directing the particular functions of the selected routine.

The layout is constructed using the above routines and, when the designer-operator is satisfied with the layout, the system transfers the data to a Master Plotter which produces the artwork.

THE USAGE

BAC initially used Redac's bureau design service unsuccessfully, owing to a difference in standards between Redal-3 and the BAC drawing offices. However, it was thought that the service rather than the program was at fault and so a team of four BAC draughtsmen were sent for training on the use of the program, using Redac's installation. After a three week training course, the men began to design PCBs on the system at a rental charge of £31 per hour. The team was unfortunately required to handle contract work although their expertise was, as yet, slowly developing. This led to a far higher design cost than had been anticipated. The work was consequently stopped and an investigation carried out to establish why the designs produced were not produced quickly and cheaply. The following comments, based on the report of the investigation, serve to pinpoint some of the major pitfalls in converting manual design to interactive CAD.

SPECIFIC COMMENTS

Primarily it was found that the system was not being used to maximum benefit because of lack of expertise. It is essential that the operators be given considerable training, free from the worry of the cost of using the system; to skimp training unavoidably leads to inefficient use of the software and hardware. It is also inefficient to use another firm's facilities if you do not also adopt their standards, because the operator is forced to expend effort in converting between the two different sets of standards. Once a design is started the cost will soar if design modifications are introduced; this is not as noticeable with manual design but, because CAD is much faster, modifications frequently require re-design. In addition, BAC found that a good deal of their trouble was caused by starting the conversion process without a program of suitable work. This caused the team members long delays between being given work, which often turned out to be of too high a standard for their developing expertise.

The report concludes by stating that the drawing office of the future will regard the following equipment as essential:

dedicated computer, digitiser, interactive display unit, master plotter, microplotter, on-line printer and appropriate software

However, the cost (quoted at around £110K) is considerable and before such equipment could realise its potential benefits, a reorganisation of the design process is required so that CAD becomes the rule rather than the exception.

GENERAL COMMENTS

Additional comments are included, again based on the report, which are concerned mainly with using any interactive system. They are presented here as a guide to operating pitfalls which detract from system performance and arise through inefficient interfacing of man and machine.

One major fault was found to be the method of using the system which was booked for five-hour blocks in the evenings. This encourages the operator to remain at the display irrespective of whether he is doing useful work or not, because the cost is the same whether he remains or goes. This can quickly lead to a semi-soporific state where the operator makes changes purely to avoid 'wasting' money by being idle. It would be much better if the operator was to stay until he became tired or reached an impasse, whereupon he could stop and examine his current results at leisure, returning to the display only when he had a clear idea of what he was going to do next. This was aggravated with the Redac installation by the lack of a hardcopy facility,other than the master plotter,which made it difficult to obtain copies of the work which could be studied at leisure.

Another difficulty was caused by the lack of any computer knowledge in the members of the design team. This meant that the programs were complete 'black-boxes' to them, and they could not anticipate the likely consequences of any but the most straightforward operations and so, in the initial stages, some system failures were produced because the operators did not clearly understand how the machine was functioning. In addition, the Redal-3 system itself was considered to be at fault by allowing the operator to inadvertantly generate invisible tracks which only the computer could recognise, and by sensitising a menu before it was displayed.

A system failure during design, either hardware or software, results in a loss of all steps since the last 'dump' (when layout data is copied from core to backing store). The operators frequently found difficulty in regenerating the layout just prior to a failure, probably because the failure occurred 10 or 15 minutes after a dump and a considerable amount of work can be performed in such an interval. Such failures are inevitably time-consuming and extremely irritating to the operator.

Despite the above difficulties, the investigation considered that, if used efficiently, the Redal-3/PDP-15 combination should provide a cost effective tool for PCB layout. However, the major point raised is that any system is only as good as the people using it, and can therefore only produce maximum benefit if used as and how it was designed to be used and with maximum efficiency.

APPENDIX II

This appendix consists of a brief list of design-rules for MOS devices processed by a standardised technique.* The list includes only those rules which were applied in preparing the MOS layouts shown in chapter 6 of this thesis. Nevertheless, the rules indicate the complexity of interrelationships which affect layout, and why they are best applied by a human rather than electronic brain.

1. INTRODUCTION

- The rules set forth are minimum dimensions. When electrical design permits the use of larger dimensions and spacing is desirable for yield improvement.
- Dimension in mils except when indicated

2. P-REGION (FIRST MASK)

- Absolute minimum width	0.2
- Minimum width when size and capacitance are of secondary importance	0.4
- Minimum non-related P-to-P spacing	0.6
- Desirable P-to-P spacing when both P-regions can go negative and metal crosses between them	0.8
- Minimum extension of P-region beyond oxide on channel ends (W direction) if width is less than 0.8 mils	0.1
- Minimum spacing between source and drain P-regions of the same device (related to P-regions)	0.4

* Process 7600 used by General Instruments Limited

3. GATE OXIDE (SECOND MASK)

- Transistors

  Minimum width 0.2

  Minimum length not including overlap 0.4

- Resistors

  Minimum width 0.2

- Minimum overlap beyond source and drain in L direction

  For devices of W = 0.2 0.2

  For devices of W 0.2 0.1

- Minimum spacing between gate oxide areas of different devices or between adjacent areas of a bent device 0.6

- Minimum spacing between gate oxide and non-related P-region 0.7

- Second mask opening appears at contact holes

4. CONTACT HOLES (THIRD MASK)

- Minimum contact hole (third mask opening) 0.2 x 0.4

- Minimum contact hole (second mask opening) 0.4 x 0.6

- Minimum overlap of second mask beyond third mask - all sides of hole 0.1

- Minimum extension of P-region beyond third mask opening of contact holes 0.2

5. METAL (FOURTH MASK)

- Minimum overlap beyond gate oxide in L and W direction 0.2

- Minimum overlap beyond third mask opening of contact holes 0.2

- Metal runs

  Minimum width 0.4

  Minimum metal to metal spacing 0.4

6. PADS

- Minimum size	4 x 4
- Desirable size when increased area not critical	5 x 5
- Minimum distance from inside edge to active circuitry	1.0
- Minimum separation between pads	3.0
- Preferable centre to centre spacing of corner pads	10.0

7. STRAY CAPACITANCE

- P-regions @ OV Bias	0.1 pf/mil^2
- P-regions @ 10V Bias	0.07 pf/mil^2
- Metal over field oxide (thick oxide)	0.02 pf/mil^2
- Metal over gate oxide (thin oxide)	0.2 pf/mil^2

APPENDIX III

This appendix comprises the proof which allows a general network to be redrawn as a permutation without affecting the link crossing structure.

Theorem: Any general network drawn in a plane with a minimum crossing configuration can be redrawn as a second network with an equivalent number of crossings,such that the second network possesses the following properties:-

(i) all nodes lie on a straight line called the node-line

(ii) connections consist of a series of semicircles in which the successive semicircles lie on alternate sides of the node-line

Proof: Given the following:-

G - a general network with N nodes, drawn in the plane μ

G^1 - network to be drawn in plane μ^1 with the required features (i and ii above)

First establish the node positions in μ^1 by drawing a simple polygon C in μ passing through each node of G once, fix a point p on the contour C, which allows measurement of length along contour, in given direction, such that each node i lies at a unique distance l(i) from p. Let L^1 be any straight line in μ^1 passing through a point p^1, the point p^1 is used as the corresponding reference point for L^1.

that the point p is for C, therefore node i can be represented on L^1 at a distance l(i) from p^1 measured in the same direction. Thus all nodes can be uniquely located in μ^1 such that the node sequence along L^1 corresponds to the sequence of nodes belonging to the network G along C.

The corresponding link paths in μ^1 are uniquely defined as follows: each link in G must terminate on C and cross C in a finite number of points $p_1 \ldots p_n$ say; these points can be measured as the nodes above to give unique distances $l(p_1) \ldots l(p_n)$ from p, equivalent points $p_1{}^1 \ldots p_n{}^1$ can then be marked on L^1 a distances $l(p_1) \ldots l(p_n)$. The inside and outside regions of μ defined by C can be conventionally related to the two halves of μ^1 defined by L^1; semicircles are drawn in μ along the node line joining consecutive points to represent to the arcs in G.

If arc joining $(p_i \ldots p_j)$ crosses arc (g_k, g_l) then they must only cross once since G contains a minimum number of crossings, if they cross once then, if $l(p_i) < l(p_j)$ and $l(g_i) < l(g_j)$, either $l(p_i) < l(g_i) < l(p_j) < l(g_j)$

or $l(g_i) < l(p_i) < l(g_j) < l(p_j)$

and both arcs are in the same region of μ. Relating this to the equivalent network components in μ^1, it can be seen that these conditions are exactly those for the intersection of corresponding semicircles. Therefore the number of crossings in G and G^1 will be the same QED.

There is a corollary to this theorem which is of greater significance to the algorithm described in chapter 3.

COROLLARY

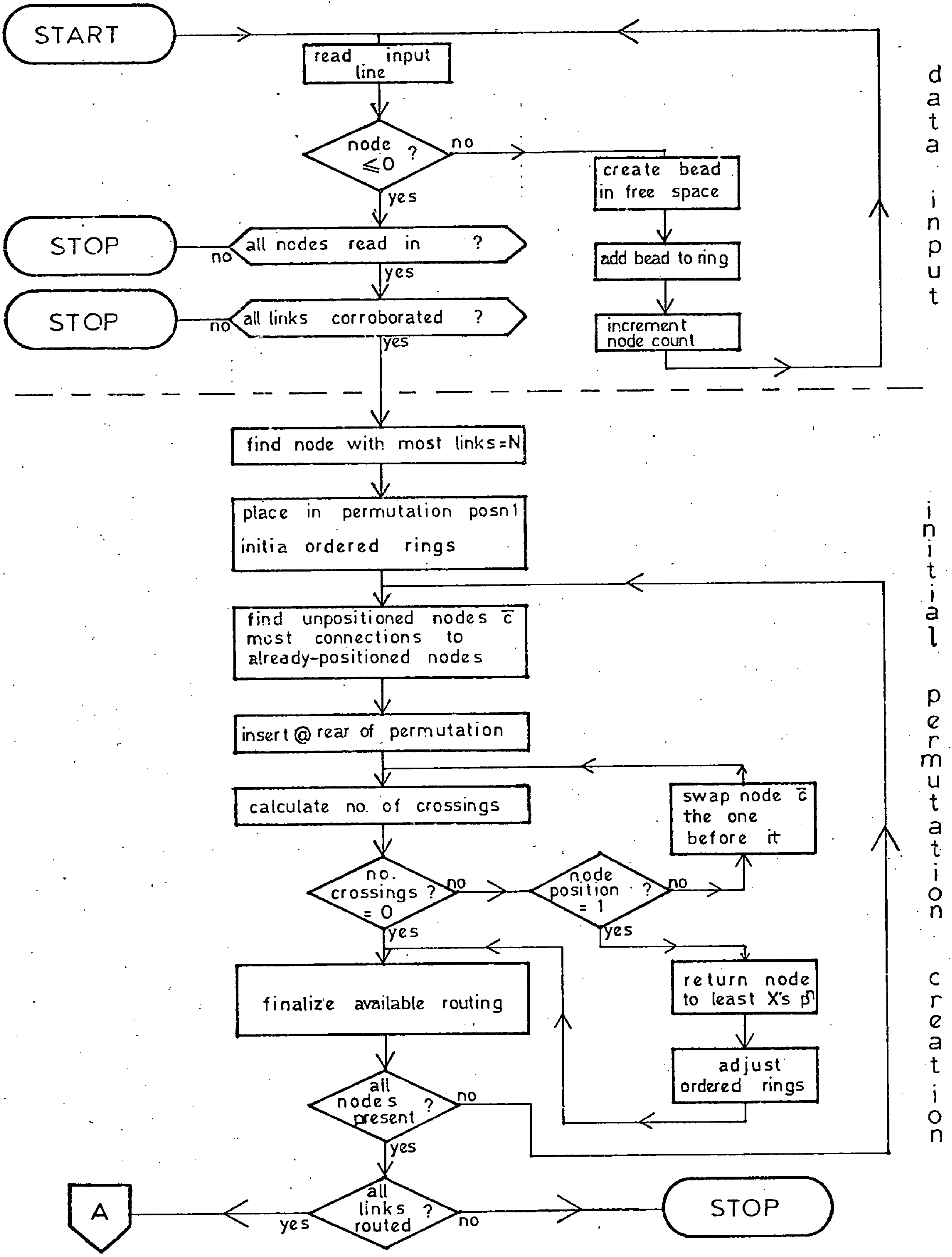
If there exists a minimal crossing representation of a network such that there exists a Hamilton chain on which no crossings occur, the corresponding links (in μ') may all be drawn as single semicircles.

PROOF

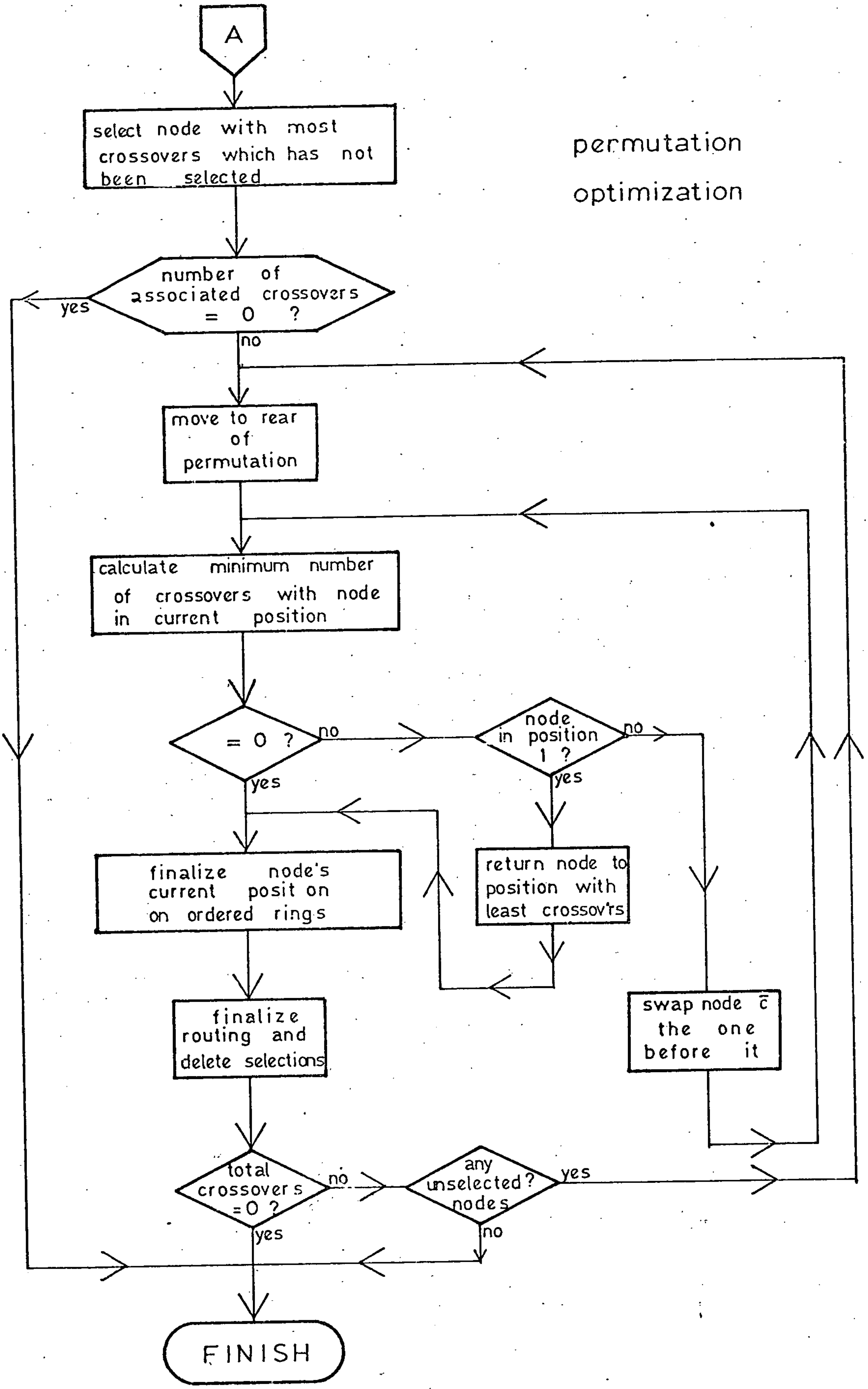
This follows from selecting C as the Hamilton chain; this implies there are no crossings on L', hence all connecting links must be drawn as single semicircles.

APPENDIX IV

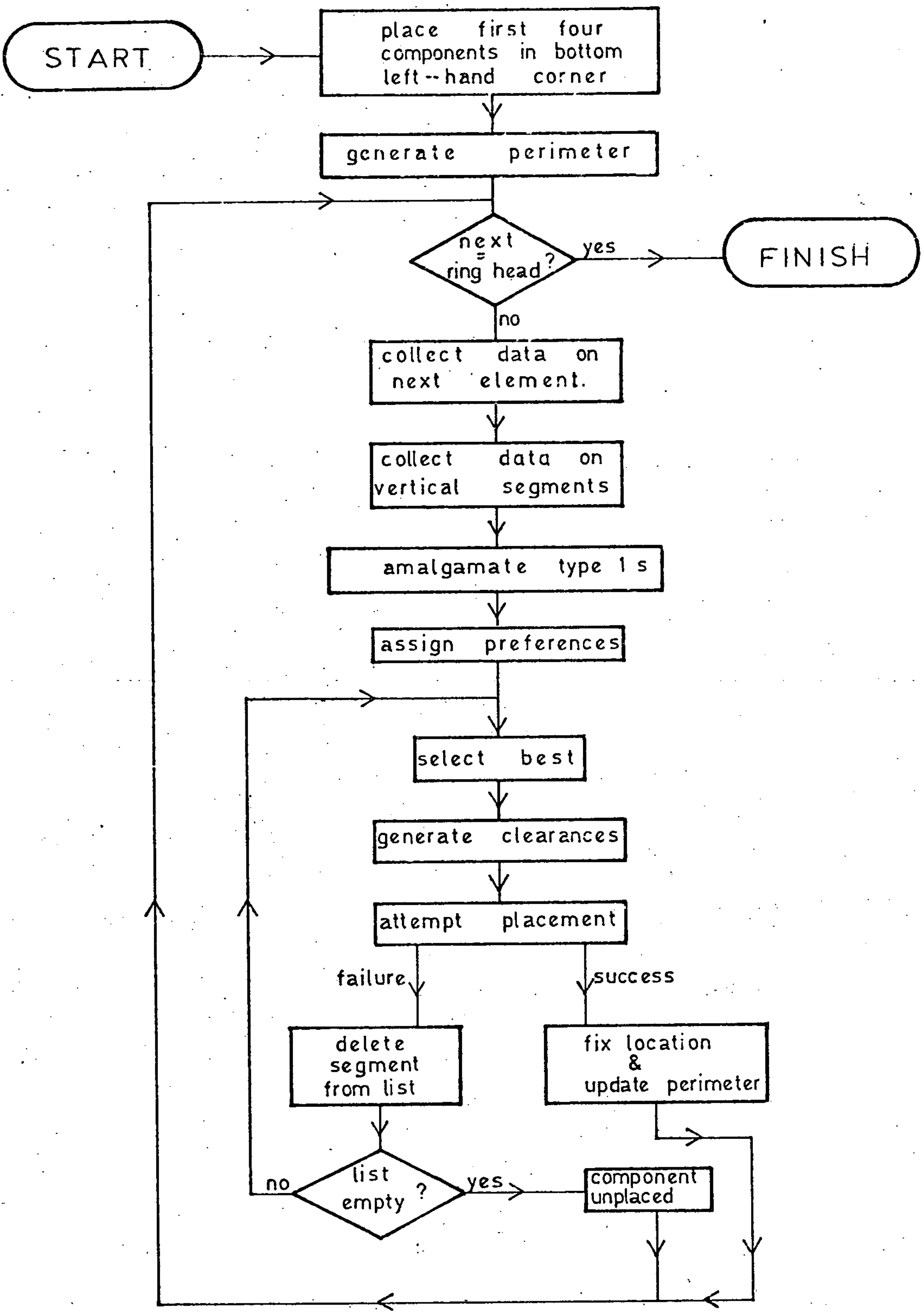
This appendix comprises a description of the programs developed during the course of this project. Owing to pressure of space, the descriptions are confined to flowcharts which depict only the major program steps.


START
read input line
node ≤0 ?
no
yes
create bead in free space
add bead to ring
increment node count
STOP
no
all nodes read in ?
yes
STOP
no
all links corroborated ?
yes
data input
find node with most links=N
place in permutation posn1
initia ordered rings
find unpositioned nodes c̄ most connections to already-positioned nodes
insert @ rear of permutation
calculate no. of crossings
swap node c̄ the one before it
no. crossings ? = 0
no
node position ? = 1
no
yes
yes
return node to least X's pn
adjust ordered rings
finalize available routing
all nodes present ?
no
yes
A
yes
all links routed ?
no
STOP
initial permutation creation

Flowchart (i) :- permutation procedure, part (a)


A
select node with most crossovers which has not been selected
permutation optimization
number of associated crossovers = 0 ?
yes
no
move to rear of permutation
calculate minimum number of crossovers with node in current position
= 0 ?
no
yes
node in position 1 ?
no
yes
return node to position with least crossov'rs
finalize node's current posit on on ordered rings
swap node c̄ the one before it
finalize routing and delete selections
total crossover s =0 ?
no
yes
any unselected? nodes
yes
no
FINISH

Flowchart (ii) :- permutation procedure , part (b)



Flowchart (iii) :- placement algorithm

GLOSSARY OF TERMS

Adjacent (nodes)	graph nodes connected by a link
Array	a series of items arranged in a meaningful pattern, usually a block of consecutive words in memory
Auxiliary (graph)	graph obtained by representing links as nodes and connecting nodes which represent links which cross
Branch	see Link
Capacity Set	parameter associated with graph component (usually link) defining how many links may cross the component
Cell	section of a microcircuit which performs a specific task and may be considered as a unit
Chain	series of alternating nodes and links of a graph such that each consecutive node-link and link-node pair are incident
Circuit (graph)	a chain whose terminal elements are the same
Coding	1. the ordered list in computer code, or pseudocode, of the successive computer operations for solving a specific problem 2. writing instructions for a computer either in machine or non-machine language

Coloured Graph

graph which has a colour associated with each node such that no pair of adjacent nodes are of the same colour

Complete Graph

graph with every pair of nodes adjacent

Connection Matrix

matrix $C = C_{ij}$ $i = 1,N$, $j = 1,N$ describing an N node graph where C_{ij} = the number of links between nodes i and j

Connectivity Index

ratio of links to nodes in a network

Contracted Graph

graph which is obtained from another by replacing at least one subgraph with a node

Crossing, also crossover (graph)

a point shared by two links, which is not a node

(circuit)

place where a connection passes above another circuit component

Crossover

see Crossing

Crossover Site

place where a crossing may be made legitimately in a circuit (such as a bipolar IC resistor)

Daughter Board

small printed circuit board containing 6-24 ICs mounted in arrays on mother board (usually by plug-in type connections)

Display File

data file created by program containing information to enable display of desired picture on graphics screen

Dual (of a graph)

graph obtained from another by representing links as nodes and nodes as circuits containing the nodes representing the links incident on them

Edge

see Link

Edge Capacity

see Capacity Set

Garbage Collection

the process of returning storage space occupied by unwanted data to the list of unoccupied space in a dynamic data structure

Graph

a set of nodes and links with a relationship of incidence which associates two nodes with each link; these nodes are the ends of the link and are said to be adjacent

Hamilton (chain or circuit)

adjective describing a set of graph elements which contains every node of the graph

Hard Copy

computer output in a permanent form which is visually readable

Hardware

the electrical, electronic,magnetic and mechanical devices or components of a computer

Incidence Matrix

a matrix $I = \left[i_{kl} \right]$ $k = 1,N, l = 1,M$ describing the incidence relationship of a graph by $i_{kl} = 1$ if node k is an end of link l $= 0$ otherwise

Link

see Graph

Manhattan
geometry in which all lines are either horizontal or vertical

Mother Board
large printed circuit board on which are mounted an array of small boards called daughter boards

Net
group of electrically common points in a circuit

Network
1. a graph
2. term used for connections of a circuit

Node (or vertex)
see Graph

Permutation
one possible order of arrangement of a given number of nodes

Permute
process of altering the order of arrangement of a permutation

Planar (graph)
graph without any crossings

Pseudo-Branches
links included in a graphical circuit representation to represent physical relationships between nodes other than circuit connections

Region (of a graph)
area bounded by a circuit which has no graph elements inside it

Software
instruction sets necessary for a computer to perform its specific tasks

Spanning (tree)
a tree which contains all the nodes of a graph

Spider
a subgraph in which all links are incident on one node

Subgraph
a subgraph of a graph is obtained by removing a set of graph elements

Tree — a connected subgraph (i.e. one in which there is a chain between every node pair) which contains no circuits

Vertex — see Node

REFERENCES

Abruca, A
Algorithm 1 Computer Bulletin,
Sept. 1964

Altman, De Campo and Warburton
'Automation of Computer Panel Wiring'
AIEE Transactions, 79 (May 1960)
118-125

Atiyah, J and Wall, P.K.
'Practical layout of PC boards using interactive graphics'. Int. Symp.
on Computer Graphics, Brunel University
1970 (Abst 13530, Computer Control
Abstracts, 5 1970) no. 51

British Aircraft Corporation
Report on the activities of BAC-Redac Software team during 1971.
BAC report R84-C1.

Bader, W
'Das topologische Problem der gedruckien
Schaltung und Seine Losung'. Archiv
fur Electrotech 49 (April 1964) 2-12

Basden, A and Nichols, K.G.
'New topological method for laying
out printed circuits' PROC IEEE 120
no. 3 (March 1973) 325-328

Beardsley
'Computer aids for IC design, artwork
and mask generation' IEE Spectrum,
Sept 1971, 64-79

Breuer, M.A.
'General survey of design automation
of digital computers' PROC IEEE 54
no. 12 (Dec 1966) 1708-1721

Brown, D.A.H. and James, A.H.
'CAD of PC boards' RRE Memorandum
2764, July 1972

CAD Project, Edinburgh University
'Interactive Graphics applied to
electronic design' Second Report to
the S.R.C., February 1973

Capocaccia, F and Frisiani, A.L.
'An algorithm for the placement of large-scale integrated circuit elements' ALTA FREQUENZA 39 no. 2 (Feb, 1970) 109-113

Case, P.W. et alia
'Solid logic design automation' IBM Journal, (April 1964) 127-140

Cooper, L
'Heuristic methods for location allocation problems' SIAM review 6 no. 1 (Jan. 1964) 37-53

Crockford, L.E., Maller, V.A.J. and Carnell, P.E.
'Procedures for the placement and interconnection of integrated circuits in digital systems' IEE conference on Numerical Control, 1967

Cullyer, W.J., Stubb, S. and Stockton, A.P.
'Computer aided layout of micro-circuits' REE 35 no. 5 (May 1968.)

Cullyer, W.J., Jones, S., Stockton, A.P. and Guscott, A.R.
'Automatic formation of trial layouts of thin film microcircuits' PROC. Int. Conf. on CAD, Southampton, April 1969, p.97

Dantzig, J.B.
'Application of the simplex method to a transportation problem' Activity Analysis of Production and Allocation, Chapter 23, Cowles Commission Monograph no. 13, New York, 1951

Dertouzos, M.L.
'An introduction to on-line circuit design' Proc. IEEE 55 no. 11 (Nov. 1967) 1961-1970

Digital Equipment Corporation
Time Sharing Handbook 1970

ibid.
Reference Handbook 1970

Ebertin
'Design of MOS/LSI with CAD techniques' Int. Conf. on Advanced Microelectronics, Paris,. 1970

Eades J.D.
'An Integrated Circuit Layout Program' BCS Conf. on Data Structures, Cambridge, 4-5 Sept 1973.

Engl, W.L. and Mlynski, D.A. 'Topological theory of ICs' NATO Conf. on Cct Thry, Bournemouth, Oct. 1972

ibid. 'Synthesis Procedure for circuit integration' Proc. IEEE Int. Conf. on Sold-State Ccts., 1969

Evans, G.C. and Gribble, M.W. 'Automatic interconnection system for electronic components' Proc. IEEE 116 no. 12 (Dec. 1969) 1992-2000

Filshie, J and McGee, I 'Implantable radio-telemetry transmitter for measuring avian physiological parameters' due for publication in Electronic Equipment News, March 1974

Fisher, G.J. and Wing, O 'Computer recognition and extraction of planar graphs from the incidence matrix' IEEE Trans. Cct Thry CT13 no. 2 (June 1966) 154-163

Fisk, C.J., Caskey, D.L. and West, L.E. 'Taking the puzzle out of PC design' Electronics, 4th Sept, 1967

ibid. 'ACCEL: Automated circuit card etching layout' Proc. IEEE 55 no.11 (Nov 1967) 1971-1982

Fletcher, A.J. 'Computer aided design of ICs' Proc. Int. Conf on CAD, Southampton 1969, p571

ibid. 'Computer programs for the layout of integrated circuits' PhD thesis, Univ of Manchester, May 1970

Fogiel, M 'Modern Microelectronics' Research of Education Association, 1972

Ford, L.R. and Fulkerson, D.R. 'Solving the transportation problems' Man. Sc. 3 (Oct 1956) 24-32

Garside, R.G. and Nicholson, T.A.J. 'Permutation procedure for the backboard wiring problem' Proc. IEEE 115 no. 1 (Jan 1968) 27-30

General Instruments Limited Design rules for the 7600 process

Ginsberg, G.L. Maurer, C.R. and Whitley, E.H. 'An updated multilayer printed wiring CAD capability' Proc 6th Share-ACM-IEEE Design Automation Workshop, June 3-12, 1969, 145-154

Goldstein, A.J. and Schweikert, D.G. 'A proper model for testing the planarity of electrical systems' Bell.Sys.Tech.Jour.52 no. 1 (Jan 1973) 135-142

Hazlett, L.H. 'I am an integrated circuit design engineer' Electronic Design News 14 (June 1969) 39-51

Hightower, D.W. 'A solution to line-routing problems on the continuous plane' Proc. 6th Share-ACM-IEEE Design Automation Workshop, June 3-12, 1969, 1-24

Hitchcock, R.B. 'Cellular wiring and the cellular modelling technique' Proc. 6th Share-ACM-IEEE Design Automation Workshop, June 3-12, 1969, 25-41

Houghton, J 'A system for the placement of circuit modules' Proc. Int Conf on CAD, Southampton, 1969, p82

Huttley 'Automated production of wiring schedules and printed circuit layouts' Marconi review, 1st quart. 1968, 20-31

James, A.H. 'Computer aided layout of PC boards' RRE Research Review 7, (1968) James, G and James, R.C. Mathematics Dictionary, D Van Nostrand Co.,1959

Kodres, U.R. 'Formulation and solution of circuit card problems through the use of graph methods' Electronic Cct Packaging 2 (1962) 121-142

Kruskal, J.B. 'On the shortest spanning subtree of a graph and the travelling salesman problem' Proc. Am Math Soc. 7 (1956) 48-50

Kuhn, H.W. 'The Hungarian method for the assignment problem' Naval Res. Log. Quart. 2 (1955) 83-97

Kuratowski, C 'Sur le probleme des courves gauche en topologie' Fund. Math. 15 (1930) 271-283

Lee, C.Y. 'An algorithm for path connections and its application' IRE trans. Elec Comps. 10 no. 3 (Sept 1961) 346-365

Lerda, F and Marjorani, E 'An algorithm for connecting N points with a minimum number of crossings' Calculo 1 (1964) 257-265

Levers, D.F.A. 'The use of a graphical display in the automatic design of PC boards IEE Conf on CAD, April 1969, Conf Publication no. 51

Loberman, H and Weinberger, A 'Formal procedures for connecting terminals with a minimum total wire length' JACM 4 no. 4 (Oct. 1957) 428-437

Lynn, D.K. 'Computer aided layout system for integrated circuits' IEE Trans Cct Thry CT-18 no. 1 (Jan 1971) 128-139

Majorani, E — 'Simplification of Lee's algorithm for special problems' Calculo 1 (1964) 247-256

Mamelak, J.S. — 'The placement of computer logic modules' JACM 13 no. 4 (Oct 1966) 615-629

Markstein, H.W. — 'Are you ready for automated artwork?' Electronic Packing & Production 10 (Dec 1970) 12-22

Mays, C.H. — 'A brief survey of computer-aided integrated circuit layout' IEE Trans Cct Thry CT-18 no. 1 (Jan 1971) 10-13

Mikami, K and Tabuchi, K — 'A compute program for optimal routing of PC conductors' 4th Congress IFIPA, Aug 1968

Munkres, J — 'Algorithms for the assignment and transportation problems' J. SIAM 5 (1957) 32-38

Narraway, J.J. (a) — 'Untangling a graph with CAD applications' BAC int tech note ST5970, May 1971

ibid. (b) — 'Component orientation in PCB layouts' BAC int tech note ST6027, July 1971

ibid. (c) — 'Component distribution in PCB layouts' BAC int tech note ST6147 August 1971

Nicholson, T.A.J. — 'Permutation procedure for minimising the number of crossings in a network' Proc. IEE 115 no. 1 (jan 1968) 21-25

Pearce, N — 'Printed circuit board layout using an interactive graphics display' Computer-Aided Design, (Winter 1970) 9-18

Peze, F.A. 'A program for semi-automatic tracing of printed circuit connections' Proc. Int Conf on CAD, Southampton, 1969, p89

Pomentale, T 'An algorithm for minimising backboard wiring functions' CACM 8 no. 11 (nov 1965) 699-703

Prince, M.D. 'Interactive graphics for computer-aided design' Addison Wesley, 1971

Radley, P.E. 'The automatic design of interconnection patterns for LSI' Proc Int Conf on CAD, Southampton, 1969, p114

REDAC Software Ltd REDAL-3 User's Manual

Rose, N 'Computer-aided design of printed wiring boards' PhD thesis, University of Edinburgh, June 1970

Rutman, R.A. 'An algorithm for placement of interconnection elements based on minimum wire length' Proc SJCC, AFIPS, 25 (1964) 477-491

Saaty, T.L. 'A model for the control of arms' O.R. Quart 12 (1964) p586

Silver, R 'An algorithm for the assignment problem' CACM 3 (Nov. 1965) 605-606

Sinden, F.W. 'Topology of thin-film RC circuits' Bell Sys Tech Journal 45 no. 10 (nov. 1966) 1639-1661

Sippl, C.J. 'Computer Dictionary and Handbook' H.W. Sams & Co., 1966

Spitlany, A and Goldberg, M.J. 'On-line graphics applied to layout design of integrated circuits' Proc. IEEE 55 no. 11 (Nov. 1967) 1982-1988

Spring, M.S. Sturgeon - User's Manual, ARU, University of Edinburgh, 3rd Ed. May 1972

Steinberg, L 'The backboard wiring problem: a placement algorithm' SIAM Review 3 no. 1 (Jan 1961) 37-50

Stone, H and Dietz, P 'Computer-aided systems-design service' Electronic Products, (April 1968) 78-83

Warner, R.M. and Fordemwalt, J.M. (Editors) 'Integrated circuits - design principles and fabrication' McGaw-Hill, 1965

Warshawsky 'Optimisation problems' Proc. Share Design Automation Workshop, 1964, 69-105

Weindling 'A method for best placement of units on a plane' Proc. Share Design Automation Workshop, 1964, and Douglas Aircraft Paper 3108

Wise, D.J.K. 'LIDO- an integrated system for computer layout and documentation of digital electronics' Proc. Int. Conf on CAD, Southampton, 1969, p72

Zarankiewicz, K 'On a problem of P. Turan concerning graphs' Fund Maths 41 (1954) p137

Zucker, J.S. 'Graphical layout system for IC mask design' IEE Trans Cct Thry CT-18 no. 1 (Jan 1971) 163-173