

# Computer aids to the design of thin film circuits.

DOIG, R.C.

1980

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

Computer Aids To The Design Of  
Thin Film Circuits

Robert C. Doig

This thesis is submitted to the CNAAB in partial fulfillment of the degree of PhD.

Robert Gordon's Institute Of Technology

December 1980

	MAN		CEN
	ME		ARCH
	ED		ART
	KEP		

## CONTENTS

PREFACE

SUMMARY

CHAPTER 1	AN INTRODUCTION TO THIN FILM CIRCUITS	1
1.1	Introduction	1
1.2	Composition And Fabrication	2
CHAPTER 2	METHODS OF COMPUTER AIDED DESIGN	10
2.1	Introduction	10
2.2	Equipment	10
2.2.1	Mainframe Computer	10
2.2.2	Peripherals	10
2.3	Resistor Design	12
2.4	Planarity	13
2.5	Automatic Layout	14
2.5.1	Force Placement	15
2.5.2	Sequential Placement	17
2.5.3	Choice Of Placement Algorithms	18
2.6	Automatic Routing Of Interconnections	18
2.6.1	Wavefront Routing Algorithms	18
2.6.2	Ray-optic Routing Algorithms	19
2.6.3	Choice Of Routing Algorithm	20
2.7	Computer Aided Photolithography	20
CHAPTER 3	AUTOMATIC DESIGN OF MEANDERING THIN FILM RESISTORS	22
3.1	Introduction	22
3.2	Minimum Area Resistors	22
3.2.1	Geometric Considerations	22
3.2.2	Simple Resistor Geometry	29
3.2.3	Decreasing Resistance By Meander Shrinkage	33
3.2.4	Increasing Resistance By Adding Meanders	37
3.2.4.1	Case (a) - Extra Whole Meander	37
3.2.4.2	Case (b) - Extra Partial Meander	40
3.2.4.3	Case (c) - Straight Extension	41

3.2.5	Simple Design Of Minimum Area Resistors	42
3.2.6	Laser Trimming	42
3.2.6.1	Resistance Of Untrimmed Block	45
3.2.6.2	Resistance Of Trimmed Block	45
3.2.6.2.1	Resistance RA	45
3.2.6.2.2	Resistance RB	48
3.2.6.2.3	Resistance K	48
3.2.7	Choosing The Number Of Trim Blocks	48
3.2.8	Effective Area Of The Resistor	49
3.2.9	Minimum Area Resistor Design Program - DESRES	51
3.3	Defined Area Meandering Resistors	57
3.3.1	Introduction	57
3.3.2	Simple Design Of Defined Area Resistors	61
3.3.3	Laser Trimming	62
3.3.4	Bounded Area Resistor Design Program - DEBOR	62
CHAPTER 4	DESCRIBING AND REPRESENTING A CIRCUIT	69
4.1	Introduction	69
4.2	Master Component Libraries	69
4.3	Preparation Of The Circuit Topology	73
4.4	Representing A Two-Pad Component In The Data Structure	75
4.5	Multi-Pad Components In The Data Structure	78
4.6	Conductor Paths	80
4.7	Representing Planarity In The Data Structure	80
CHAPTER 5	THE PLANARITY ALGORITHM	85
5.1	Introduction	85
5.2	Tree Search Procedure	87
5.3	Formation Of Regions	87
5.4	Non-Planar Graphs	89
5.5	Conversion Of Non-Planar To Planar Graphs	91
5.6	Choice Of Planar Graph	95
5.7	Planarity Program - PLANAR	98
CHAPTER 6	THE START OF A LAYOUT - PROGRAM "PLACE"	101
6.1	Geometry Of Crossover Sites	101
6.2	Board Dimensions	104
6.3	Edge Pad Positions	104
6.4	Initial Placement	104

6.4.1	Component Orientation	104
6.4.2	Component Rotation	109
6.4.3	Removing Components To Upper Levels	115
6.5	Program "PLACE"	117
CHAPTER 7	THE PLACEMENT ALGORITHM	120
7.1	Introduction	120
7.2	Choosing The Next Region	120
7.3	Forming The "Slot Boundary"	122
7.4	Placement Techniques	125
7.4.1	Filling The Slots	127
7.4.2	Filling The Slots/Mounds	127
7.4.3	Component Rotation	131
7.4.4	X Boundary Expansion	131
7.4.5	Component Removal	133
7.4.6	Rough Placement	133
7.5	Using The Placement Algorithm	133
CHAPTER 8	THE ROUTING ALGORITHM	137
8.1	Definitions	137
8.2	Basic Algorithm	137
8.3	Approach Distances	137
8.4	Initial Shunting	142
8.4.1	Introduction	142
8.4.2	Single Shunt	142
8.4.3	Double Shunts	142
8.4.4	Shunt Selection	144
8.4.5	Shunt Distances	146
8.5	Final Shunting	146
8.6	Under-Run	149
8.7	Over-Run	149
8.8	Extrapolation	149
8.9	Intermediate Shunting	151
8.10	Following The Slot Boundary	154
8.11	Using The Algorithm	156

CHAPTER 9	INTERACTIVE DESIGN PROGRAM "LAYOUT"	159
9.1	Introduction	159
9.2	Command Modes	159
9.2.1	Menu Mode	159
9.2.2	Cursor Mode	161
9.3	Facilities	164
9.3.1	"AUTOMATIC" Connection	164
9.3.2	"BOARD" - Increase Board Area	167
9.3.3	"CONNECT" Up Components	167
9.3.4	"END" Program	173
9.3.5	"EXPAND" Layout	173
9.3.6	"HARD COPY" Generation	175
9.3.7	"HELP" List	179
9.3.8	"IDENTIFY" Component	179
9.3.9	"MANUAL" Connection	180
9.3.10	"MASK" Selection	183
9.3.11	"MOVE" Component	186
9.3.12	"NEXT" Components	189
9.3.13	"NO CONNECTIONS"	189
9.3.14	"NO NAMES"	189
9.3.15	"NO NODES"	189
9.3.16	"PLACE" Component	190
9.3.17	"QUERY" A Component's Connections	190
9.3.18	"REDRAW" Layout	190
9.3.19	"REMOVE" Component/Connection	191
9.3.20	"RESTART" Program	191
9.3.21	"ROTATE" Component	191
9.3.22	"SHORTEN" A Connection	194
9.3.23	"SHRINK" Board Area	196
9.3.24	"SLOTS" - Display Slot Boundary	196
9.3.25	"SPRINGS" On Components	196
9.3.26	"WINDOW" Layout	197
9.3.27	"ZOOM" In Or Out	197
CHAPTER 10	PRODUCING PHOTOLITHOGRAPHIC MASKS USING A TAPE-CONTROLLED PHOTO PLOTTER	201
10.1	Introduction	201
10.2	Co-ordinate Representation	203
10.3	Plotter Driving Codes	203

10.4	Plotter Code Producing Program - "PHOTO"	205
10.5	Straight Line Polygons On The Photoplotter	207
10.6	Rectangle Forming Algorithm	207
10.7	Straight-Line Polygon Splitter - "CUTTER"	211
10.8	The Use Of Different Aperture Shapes	215
10.9	Filling In Rectangles	223
10.9.1	Method (a) - Using Circular Track Only	223
10.9.2	Method (b) - Using Track And Square Apertures	230
10.9.3	Method (c) - Using Slit Shaped Apertures	230
10.10	Filling In Triangles	233
10.11	Filling In Trapeziums	239
10.12	Conductor Track Filling	239
10.13	Mask Production Programs	243
10.14	Evaluation Of Various Slit Ranges And Filling Techniques	246
CHAPTER 11	CONCLUSIONS AND POSSIBLE IMPROVEMENTS	252
11.1	Component Design	252
11.1.1	Conclusions	252
11.1.2	Possible Improvements	252
11.2	Describing And Storing The Circuit	253
11.2.1	Conclusions	253
11.2.2	Possible Improvements	253
11.3	Planarity	254
11.3.1	Conclusions	254
11.3.2	Possible Improvements	255
11.3.2.1	Node Splitting	255
11.3.2.2	Re-Sequencing The Edge Pad List	255
11.3.2.3	Component Pad Swapping	257
11.3.2.4	Routing Outside The Edge Pads	257
11.3.2.5	Multiple Crossover Jumps	257
11.3.2.6	Routing Underneath Attached Components	261
11.4	The Placement Algorithms	261
11.4.1	Conclusions	261
11.4.2	Possible Improvements	264
11.4.2.1	X Boundary Expansion Limits	264
11.4.2.2	Edge Pad Placement	264
11.4.2.3	Choice Of Placement Technique	269
11.5	The Routing Algorithm	269
11.5.1	Conclusions	269

11.5.2	Possible Improvement	270
11.6	Photoplotting	273
11.7	Final Conclusions	273
APPENDIX 1	EXAMPLE OF LAYOUT DESIGN USING A SIMPLE TEST CIRCUIT	274
APPENDIX 2	RESULTS OBTAINED FROM PRACTICAL CIRCUITS	299
REFERENCES		307
BIBLIOGRAPHY		310



## PREFACE

This dissertation presents work carried out by the author in the department of Electrical and Electronic Engineering at Robert Gordon's Institute of Technology from September 1976 to December 1979.

I would like to thank my supervisor, Dr. J.D. Eades, for his valuable help and encouragement throughout. I would also like to express my appreciation to Mr. B. Davidson for his assistance in the preparation of all the illustrations, my fellow post-graduate students for their constructive criticism, Margaret Reynolds for typing the text, and the department staff in general.

No part of the work described in this dissertation has been submitted at any other university, and, except where otherwise stated, the work is original.

## SUMMARY

The work described in this thesis is concerned with the ways in which a digital computer can assist in the design of thin film circuits.

One of the main advantages of thin film technology is that very accurate resistors can be fabricated on the slice. A set of programs have been developed such that the resistor geometries can be synthesised entirely automatically. The results are difficult to match by a conventional manual approach because of the iterative nature of the problem - manual techniques tend to make over-simplifications in the design. This increases the amount of adjustment required to each resistor after manufacture.

The user can merge resistor geometric information with additional topological circuit data to drive the main interactive graphics programs. These combine automatic placement and routing algorithms with general purpose manipulative facilities, to provide a substitute for the traditional pencil and paper.

The automatic placement facility incorporates the concept of "planarity", which attempts to minimise the number of conductor crossovers required by the automatic routing algorithm. This is important since each crossover has to be added by hand after fabrication. Initially developed for the design of single-sided printed circuit boards, the planarity concept is equally applicable to thin film circuits, which are made with a single layer of interconnections.

The user has complete control over the design process at each step - the programs allow him to adjust the layout interactively after each placement or routing phase. This ensures that any inadequacy in the automatic routines is corrected before it can be compounded in further processing.

When the layout has been finalised, it is possible to fabricate a set of photolithographic masters directly from the data structure. The programs which have been written for this purpose, produce paper tapes to drive a flat bed plotter fitted with a light attachment. The mask-shapes are

systematically exposed to light on photographic film using a variety of different aperture shapes.

The design suite is completely modular in structure and may easily be extended. Future improvements should serve to reduce the amount of manual interaction required at each stage.

## Chapter 1

### An Introduction To Thin Film Circuits

#### 1.1 Introduction

More and more electronic circuits are now available in miniature form. While printed circuit boards (P.C.B.'s) are preferred when limited numbers are required, it is economically better to use integrated circuits (I.C.'s) for mass production. The cost of producing photolithographic masks to fabricate an I.C. is very high both in man-hours and capital, but the cost of making the circuits thereafter is very low. If very large quantities are required, then, the cost per circuit can be much less than in the printed circuit board case. There is also the advantage of compactness and reliability associated with the I.C. and it is clear that much more complex circuits can be achieved.

The time taken to make each I.C. is relatively small compared to the P.C.B. equivalent. This is due to the fact that there is little manual interference other than in connecting the circuit to its package terminals. In the P.C.B. case, the components must be individually attached to the board and soldered into the circuit. While P.C.B.'s are generally fabricated individually, I.C.'s are made in batches of several hundred or more at once.

Thin film circuits may be regarded as a half-way house between these two techniques. Like the P.C.B., the initial cost of producing the circuit is much lower than the I.C. but the cost per unit is still reasonably low given sufficient quantities. They can be described as true micro-circuits since their dimensions are measured in fractions of an inch. I.C.'s, of course, are generally more compact and complex.

Like the P.C.B., one can attach discrete components to the substrate and connect them into the circuit with metal bonding wires. It is also possible, however, to fabricate devices in the film itself. By far the most common of these is the resistor and extremely high tolerances can be achieved. Most applications are in those areas which require very accurate resistors. An example of this is in the fabrication of communications amplifier circuits, where the gain is defined by resistor values and an

accurate gain is essential.

## 1.2 Composition And Fabrication

Figure 1.1 illustrates a simple thin film circuit of the type which will be considered. The circuit starts out as a small ceramic substrate an inch or so across. The material chosen for this base layer must be able to withstand high temperatures without cracking, and should take no part in chemical reactions with other substances placed in contact with it. Its co-efficient of expansion should also closely match these substances to prevent fractures.

Several other materials are often used as an alternative to ceramic. Among the most common of these are glass, oxidised silicon and sapphire.

The substrate is usually glazed or polished to give a smooth surface before deposition begins.

A resistive film is grown onto the surface of the substrate. The actual composition of this film varies from manufacturer to manufacturer, but Nichrome metal and Tantalum Oxide are commonly used.

A close check is kept upon the film resistivity throughout the deposition to achieve a very accurate value. This is important since the resistors mentioned in the previous section are made from this layer.

Gold, or a similar low resistance metal is then spread over the resistive coating. This is used to create a set of conductor tracks linking the components to form the desired circuit.

The patterns used for a thin film circuit determine both the locations and the actual values of the in-slice components. They also define the necessary interconnection tracks. There are two common methods of creating these patterns on the substrate - the first is "Rejection Masking" and the second is "Selective Removal". In the former case, a positive of the desired shapes is cut into an inert substance (usually metal, glass or graphite) to produce a "Mechanical Mask". Both the resistive and metallic layers are then deposited on the substrate through the mask shapes. This method is commonly used for "thick film" circuits but seldom for thin film or integrated circuits.

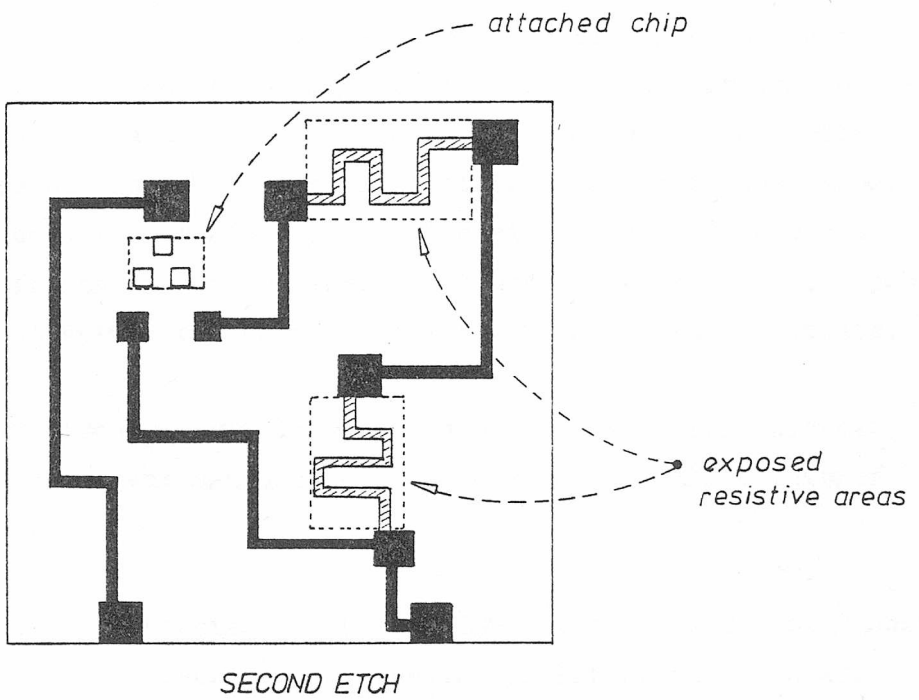
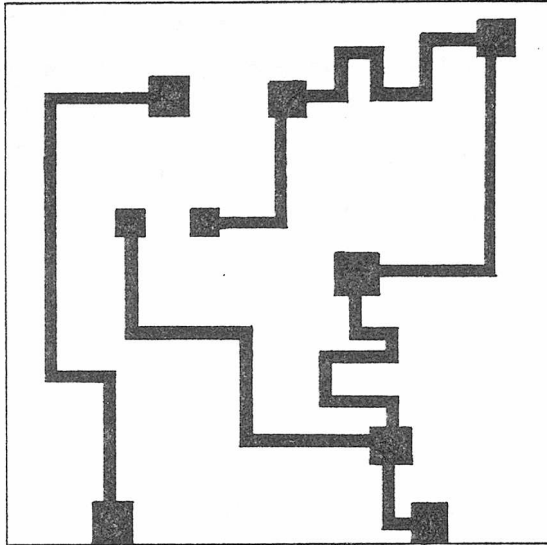
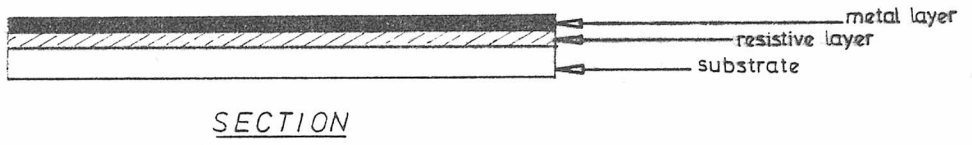


Figure 1-1 Thin film composition and fabrication

"Selective Removal", or "Photolithography" as it is often called, achieves the highest pattern complexity with the finest lines and the highest accuracy. The technique is widely used in the electronic industry for both solid state and thin film circuits. The surface to be chemically treated is covered with a polymer etching solution called "Photo-Resist". These solutions can be obtained to form either positive or negative patterns when exposed to light through a photographic mask and heat treated. Excess Resist is washed away using a solvent to leave a hard protective coating in the shape of the mask elements. The use of a suitable acid or reagent removes the untreated areas of the circuit.

In the definition given in Figure 1.1, two "etches" are needed to fabricate the circuit. The first removes everything but the circuit patterns - right down to the substrate - and the second exposes resistive patches to form the necessary in-slice resistors.

It is also possible to fabricate capacitors and inductors on the substrate but only small values can be obtained and extra deposition stages may be required in fabrication. Small solid state devices can be attached to the circuit, however, using an epoxy resin or similar compound and the devices connected in using a wire bonding technique. In addition to capacitors, transistors, diodes and inductors, very high or very low valued solid state resistors may be used to save board size (or to prevent over-heating effects). Since the design of in-slice non-linear components is rather inaccurate and varies from manufacturer to manufacturer, only resistors will be considered in the suite of programs to be described. The user will, however, have the option of designing his own devices manually then defining the shapes inside a bounding rectangle. The programs will treat them as normal components until the mask generation stages.

Figure 1.2 gives an example of an industrial thin film circuit. This particular layout was manually designed using the GAELIC suite of programs (Ref. 6).

The circuit is connected to the outside world via a set of connector pins protruding from the package - not unlike an Integrated Circuit. Wires are bonded between these pins and a set of conductor "Edge Pads" on the board itself. The wires are attached at the same time as the links between components and tracks.

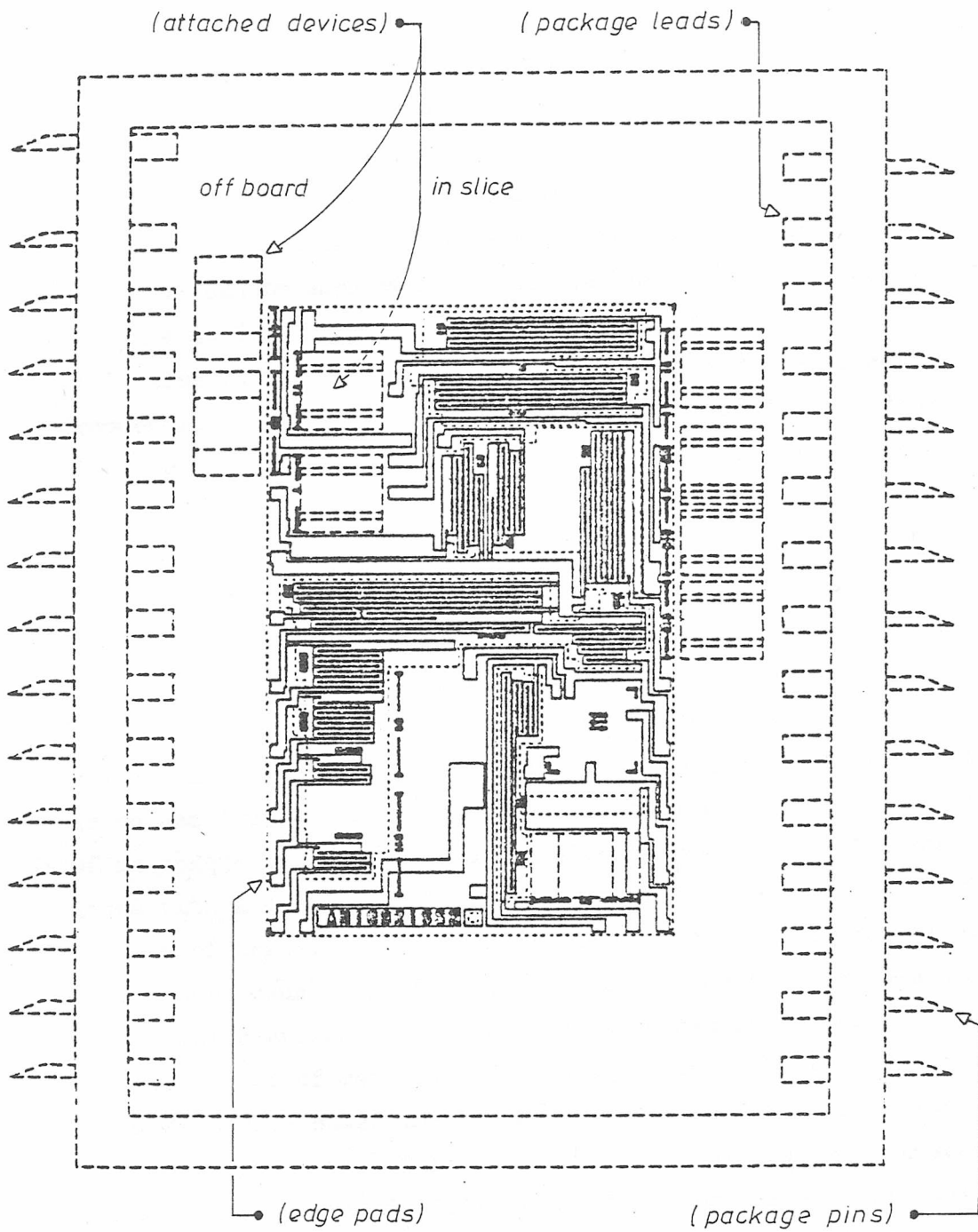


Figure 1-2 Industrial example of a thin film circuit  
(manual layout)



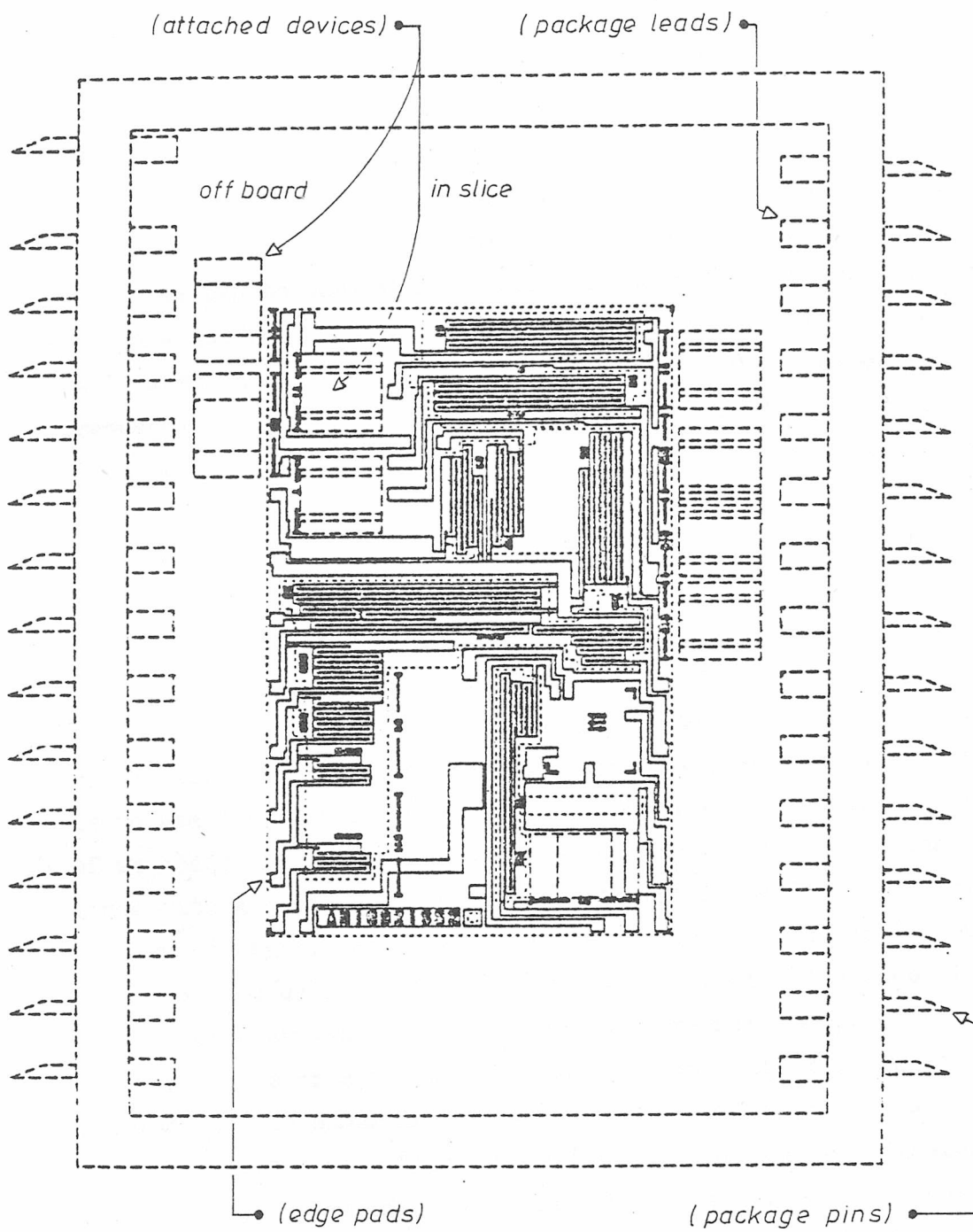


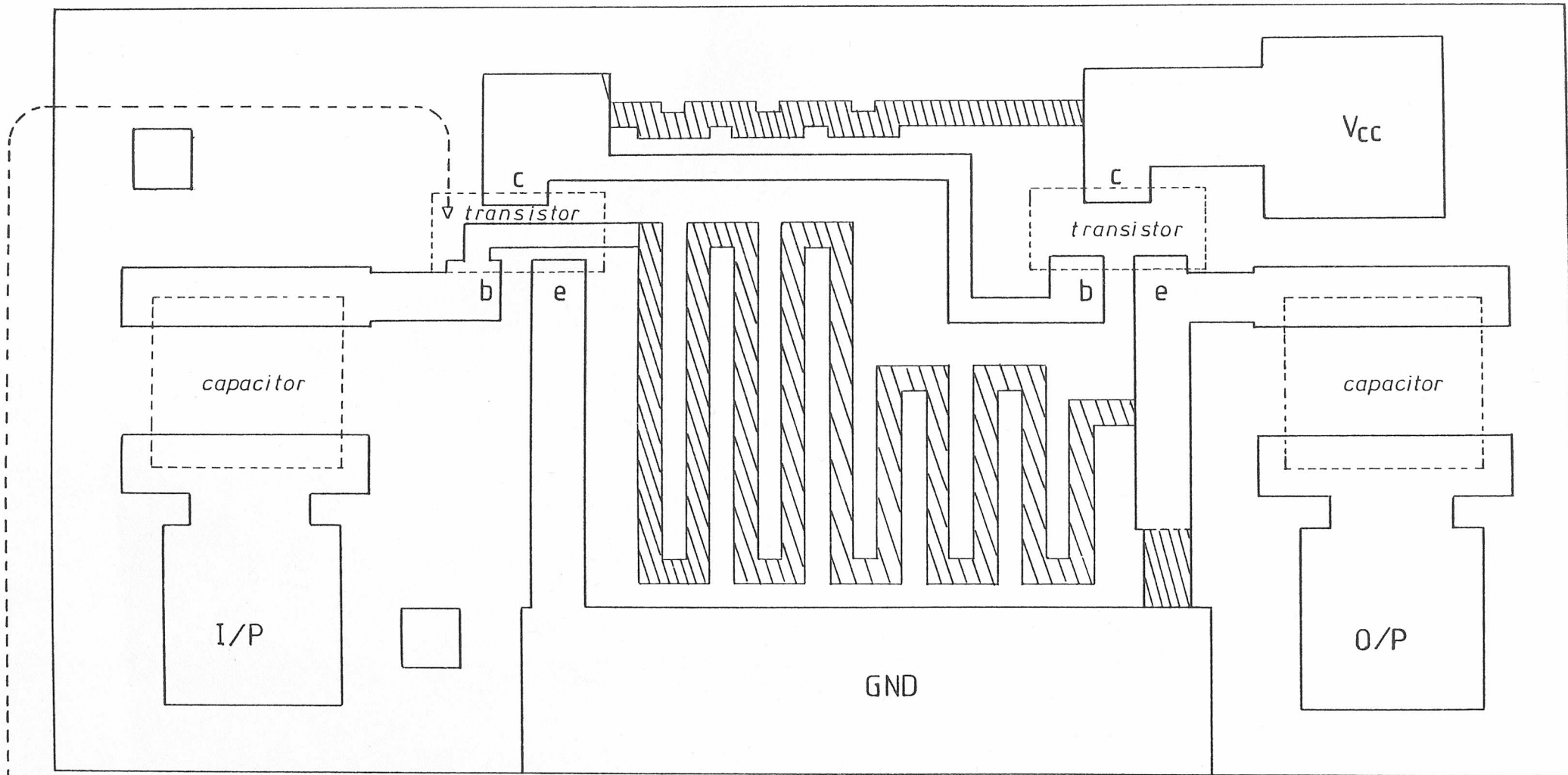
Figure 1-2 Industrial example of a thin film circuit  
(manual layout)

The layout task is greatly eased if some of the components are left outside the board perimeter as shown in the diagram. This does not cause any extra problems since they have to be wired up in any case.

Figure 1.3 is a further example - in this case a wideband amplifier (Ref. 23). This is a much simpler circuit than before but it demonstrates that external components can actually be positioned on top of conductor tracks. Another way of describing this is that the connections can be routed underneath the components in the same manner as P.C. boards. The technique can be employed to minimise the number of crossover jump wires that are needed. A serious problem with this is that some components will cause an unacceptable amount of interference - especially at high frequencies. This facility will be neglected for the time being but is discussed in Chapter 11 as a possible future addition to the program suite.

It is the design of the photolithographic masks that is the important step from the computer-aided design point of view, since the geometry of the system completely determines the nature of the electronic circuit. The most common method of mask production is to make a large scale drawing - usually about twenty times final size - and then to reduce it photographically. There are several ways of doing this but the most common is to use "Cut And Peel" material as shown in Figure 1.4. This consists of an opaque material with a translucent backing. The shapes are outlined with a knife and the opaque material is peeled off leaving the desired pattern of translucent areas. The cutting can be carried out by hand but is more usually performed using a tape-controlled co-ordinatograph with a knife attachment. It is time-consuming to peel off all the undesired strips of material by hand and very often the smaller strips cause problems due to human in-accuracy. A more direct approach is to use a "Photoplotter". This method uses a modified co-ordinatograph to expose areas on a photographic film with a light beam and is discussed in detail in Chapter 10. The resulting transparency need only be photographically reduced to obtain the mask.

The most common component to be fabricated in the thin film is the resistor. Unlike the Integrated Circuit resistor, it can be produced to a fine tolerance. Whereas I.C. resistors can be fabricated to  $\pm 10\%$ , figures of  $\pm 0.1\%$  are not uncommon in the thin film field. The reason for this difference in accuracy is the fact that thin film resistors can be



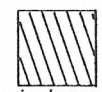
- 7 -

○ connection routed underneath attached device

materials key



gold



nichrome



solid state chip

Figure 1-3 A thin film circuit lay-out with solid-state devices attached

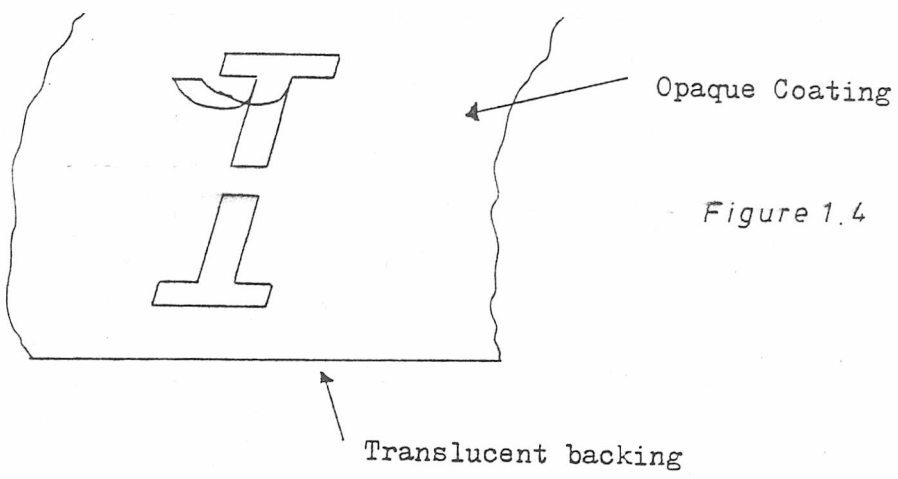
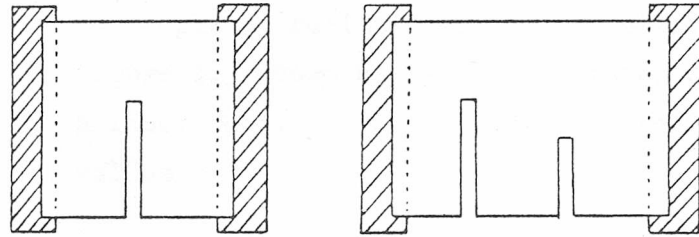
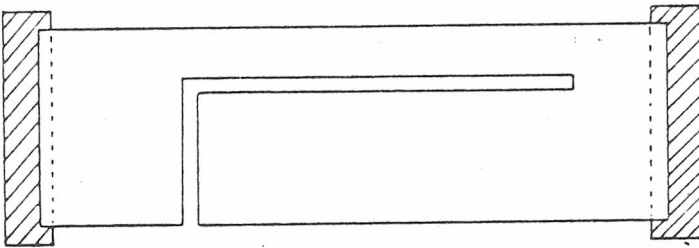


Figure 1.4 Cut and peel material

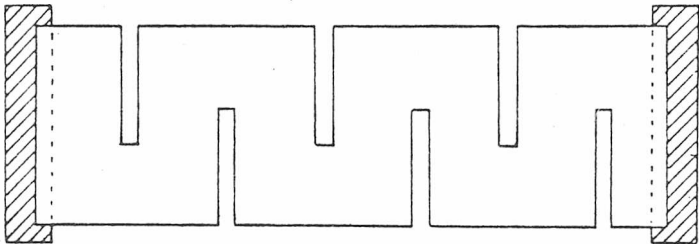


Straight-in cut

Double cut



L-shaped cut

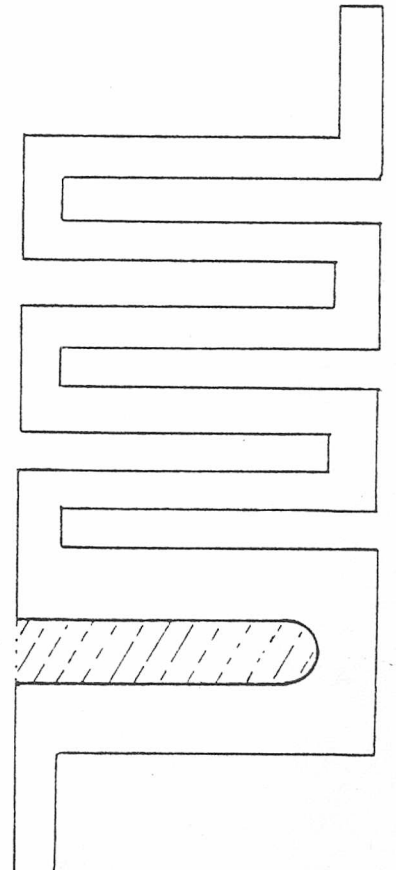


Serpentine cut

Figure 1.5 "Straight-line" resistors with laser trimming methods

Figure 1.6 Meandering resistor with laser trimming

section removed by laser beam



trimmed after they have been fabricated. There are several possible trimming methods but the system adopted in this case is to use a laser beam to burn away resistive material until the desired value is reached.

The uses of thin film resistors are widespread, but one particular example is in the construction of resistor ladder networks for Analogue to Digital Convertors where sets of resistors are required to be in exact proportion to each other. There are two main types of resistor, the "Straight Line" and "Meandering" cases. The first is simply an isolated rectangle of resistive film with metal connecting pads at each end. Figure 1.5 shows how such a resistor can be trimmed to an exact value using a laser beam. The "Meandering" resistor configuration is used for higher values of resistance to fit a long section of track onto the substrate - Figure 1.6 shows a typical pattern. The geometric design of such a resistor is quite complex and is explained in Chapter 3.

## Chapter 2

### Methods Of Computer Aided Design

#### 2.1 Introduction

This chapter describes ways in which the numerical processing ability of a large mainframe computer can be used to assist in the design of thin film circuits.

Very little material has been published specifically on this subject, but there has been a lot of interest in computer aided production of Integrated and Printed Circuits. A large proportion of this work is directly relevant to the thin film problem and is discussed where appropriate in the following sections.

#### 2.2 Equipment

##### 2.2.1 Mainframe Computer

The research was initially carried out using the DEC 10 computer at the computing centre in Edinburgh University. A G.P.O. telephone modem was employed to link the computer with a graphics terminal in Aberdeen.

The situation changed in 1977 when a DEC 20 computer became available in Aberdeen itself. The machine, owned by Robert Gordon's Institute of Technology, was used for the remaining period of research.

##### 2.2.2 Peripherals

Figure 2.1 shows the Tektronix 4014 storage visual display terminal used throughout. Once a shape has been drawn on the screen, it remains visible until the whole drawing area is cleared. This makes it impossible to selectively erase a portion of the screen, but a high resolution can be obtained and the drawings need not be stored in computer memory.

The terminal is simply used as a graphical display device in the case of the meandering resistor design programs described in Chapter 3. It can also be employed as a substitute for a penplotter/photoplotter to

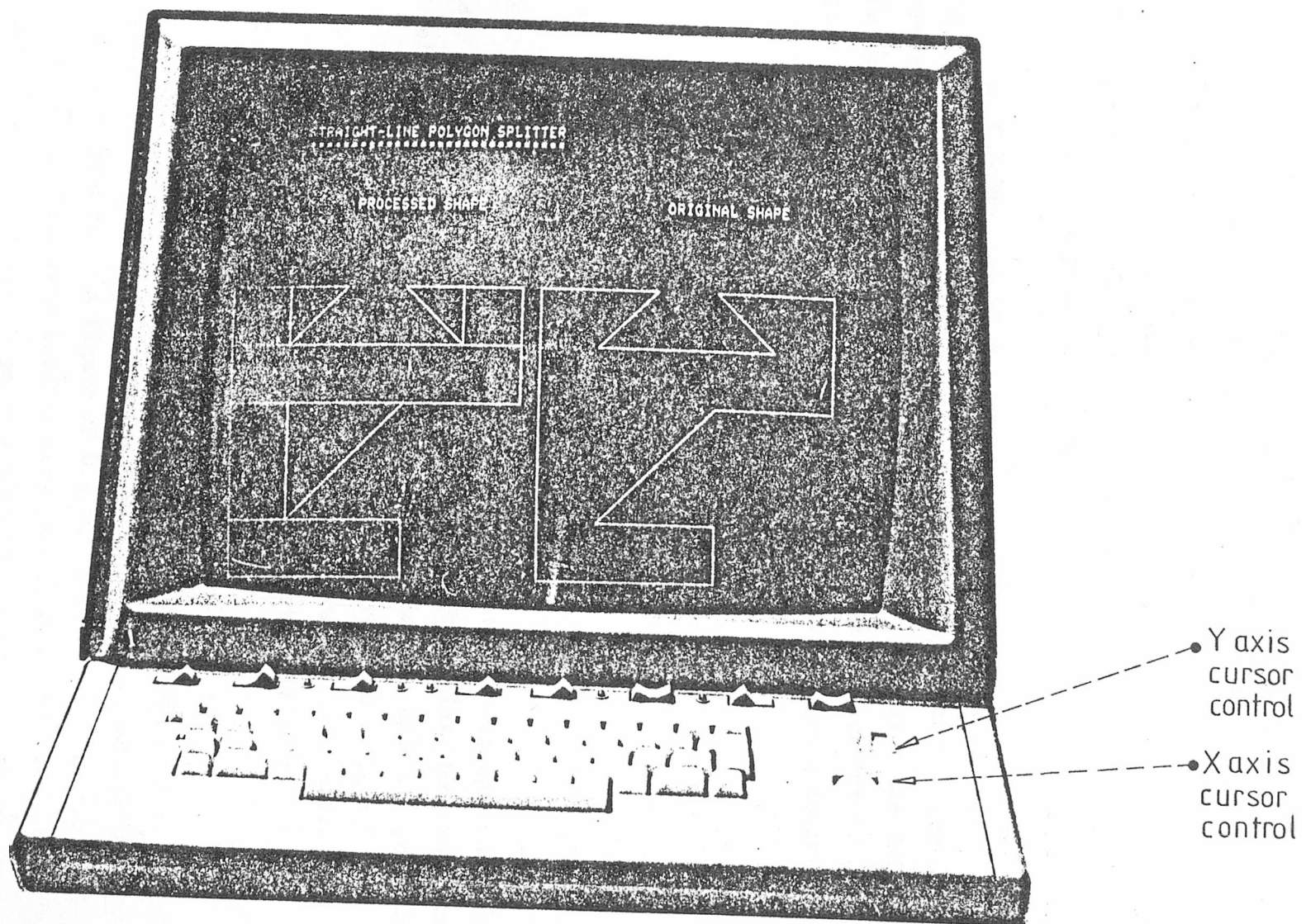


Figure 2-1 The Tektronix 4014 Visual Display Terminal

show the output from the mask production programs developed in Chapter 10.

The user can indicate a point on the screen using a set of two thumbwheel controls which move horizontal and vertical cursor lines. This facility enables him to define and move shapes interactively. "LAYOUT", the combined placement and interconnection program, uses this technique such that the user can apply his own skill and spatial awareness throughout the design (refer to Chapter 9).

A Calcomp pen plotter and a Ferranti flat bed photoplotter were available to generate experimental photolithographic mask patterns. The former was located in Aberdeen and operated directly from the DEC 20 machine. The latter belonged to Ferranti Ltd at Crewe Toll in Edinburgh. It was driven using paper tapes generated by the artwork programs described in Chapter 10.

An alternative type of permanent graphical record could be obtained using a Tektronix hard copy unit situated near the visual display unit.

### 2.3 Resistor Design

The design of straight-line thin film resistors is quite a simple task. Meandering resistors are very much more complex, however, since the effect of right angled bends in the track depends upon the geometry of the resistor near the corner. This was approximated by Radley (Ref 18) in his program to design resistors within a straight sided polygon. The user supplies "correction factors" in an attempt to allow for the inaccuracies involved.

A fixed value for each corner was assumed by CRUM (Ref 4). Her program, called "GADOR", designs resistors to fit a rectangular area defined by the user. A figure of 0.5 was assumed for each track bend. Research by Bell Telephone Laboratories (Ref 2) has shown that this value can vary from 0.469 to 0.559 depending on the length of track at either side of the corner. This can introduce inaccuracies of up to 6% and 3% on the higher and lower sides of the design value. In practice the overall effect of the corners is much less than these maximum figures would imply, but there is clearly room for improvement.



A program was written (Chapter 3.3) using the GADOR resistor definition but with formulae to calculate the individual corner allowances. The program, called "DEBOR", also incorporates special laser trimming blocks into the resistor geometry. These are used to finely adjust the resistor value after manufacture and there is no equivalent facility in GADOR. Since the major advantage of the thin film circuit is that highly accurate resistance values can be obtained, GADOR resistors are usually graphically altered to include trim blocks. This is a manual task carried out using the GAELIC suite (Ref 6) entirely at the estimation of the engineer. DEBOR makes such an arbitrary decision unnecessary as it automatically calculates the number required and edits the resistor before it is displayed for verification.

A further refinement was to include the facility of auto-expansion in one or both bounding rectangle dimensions. The user can fit resistors into narrow horizontal or vertical channels by allowing one dimension to be increased until a solution is achieved.

DEBOR can also be used in a completely automatic design suite by setting a standard rectangle size and forcing a two-dimensional expansion. This proved to be a relatively slow process, so a program was written to design resistors with automatic placement and interconnection in mind. The program is named "DESRES" (Chapter 3.2) and assumes a standard resistor shape such that physical size becomes unimportant provided that its geometry is scaled in strict proportion. Trim blocks are automatically included as before and the program chooses a resistor size such that overheating effects are unlikely to affect its performance.

Both DESRES and DEBOR include a number of iterative loops to achieve a solution. These would be very tedious to reproduce by hand using the same formulae, so a manual approach would involve a number of approximations. This would lead to a less accurate resistor and an increase in the trimming required. From the manufacturing point of view alone, then, an automatic solution is both faster and more economical.

## 2.4 Planarity

Thin film circuits are constructed with a single layer of connections. This makes it impossible to cross two conductor tracks without the use of

a jumping wire. Since this is added by hand, it is important to restrict the number of crossovers to a minimum.

A similar problem was encountered by Rose (Ref 19) in his programs to design single-sided printed circuit boards. He developed a "Planarity" algorithm to find a solution with the minimum of crossover sites. This was also used by "Fletcher" (Ref 10) to design Integrated Circuits.

A version of Rose's algorithm has been implemented and is described in Chapter 5. The algorithm was extended such that crossovers were automatically included where necessary.

It is very difficult to guess the number of crossovers required simply by looking at a circuit diagram, since there are any number of ways in which it can be drawn. Indeed, this fact was demonstrated quite well while testing the automatic crossover insertion routines. A simple circuit with one crossover was formulated and tested. Results showed that the circuit was completely planar - the circuit had simply been drawn such that a crossover seemed necessary. The automatic planarity checking, then, is an important part of the design process.

## 2.5 Automatic Layout

Many algorithms have been developed in the last two decades concerned with component placement. The earliest work in this area was devoted to improving existing layouts rather than generating layouts from scratch. The most successful techniques worked on the principle of "Pair Swopping". Random pairs of components are interchanged to investigate the effect on the layout. If an improvement is detected with regard to connectivity or total conductor length, the transposition is made permanent and the process continues with another pair.

An early algorithm of this type was written by Steinberg (Ref 22). This was subsequently improved by Rutman (Ref 21) by increasing the sensitivity of the success achieved in each interchange - with regard to conductor length in particular. Several other variations were written using the basic Pair-Swopping algorithm, notably by Nugent (Ref 16) and Mamelak (Ref 15).

More recent layout suites generate a rough placement automatically and then apply the improvement techniques. The success of the post-processing, however, is strongly dependent upon the quality of the original placement.

Most automatic layout methods treat the positioning of the circuit elements as a separate task from that of routing the interconnections although the minimisation of conductor length is an important design rule. At the placement stage the connections are generally regarded as straight lines between components which can intersect freely.

Two main techniques have been developed to generate a provisional layout - "Force Placement" and "Sequential Placement".

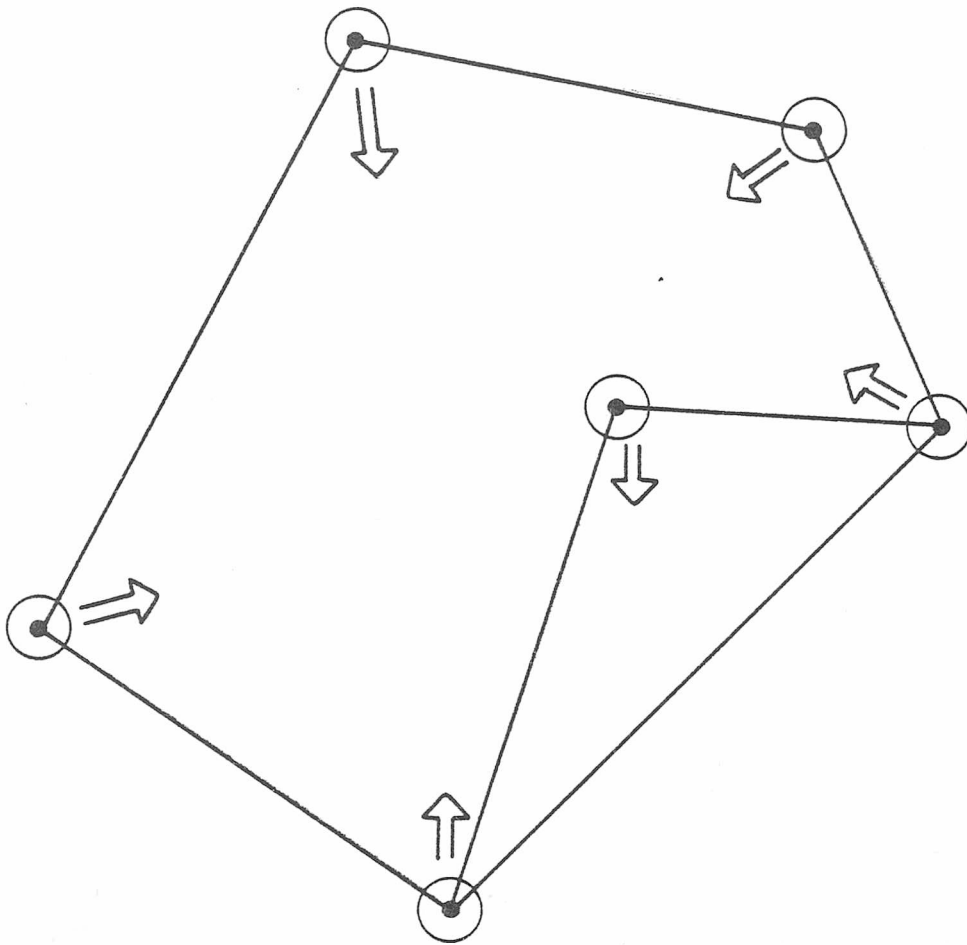
#### 2.5.1 Force Placement

This approach is also referred to as the "Rubber Band" method since the circuit connections are assumed to be springs or rubber bands. Each component is examined in turn, the forces due to its connections calculated, and moved a short distance in the resultant direction. The circuit elements are generally considered to be points where the connections meet and the length of each connection is directly related to the force it exerts.

The components are placed at random in the initial stage and allowed to move to new positions through repeated use of the force algorithm (see Figure 2.2). An early suite of programs were written by Rowles and Toome (Ref 20) using this approach, and a variation was implemented more recently by Leever (Ref 14).

The idea of the algorithm is that components which are closely connected tend to cluster together which should theoretically ease the routing task and give a solution with minimum conductor length. The process must periodically halt to expand the circuit since the components are made to crush together due to the attractive forces involved.

Improved versions of this technique have been developed by Capocaccia and Frisiani (Ref 3) and by Fisk, Caskey and West (Ref 9) in which repulsive forces have been added to prevent the components from crushing, and make expansions unnecessary. The forces are calculated with regard to the shape and size of each component.



component



connection attractive forces



direction of motion

Figure 2-2 Rubber band placement technique

The major disadvantage with the Force Placement method is that many iterations may be required to reach a satisfactory solution. This makes for a relatively slow process which may be unsuited to on-line interactive programs.

### 2.5.2 Sequential Placement

In this method the components are placed individually starting with a board containing only terminal pads. The components are positioned individually with regard to those already placed. A common technique is to place the component at the "Centre of Gravity" of the components on the board to which it is attached. Radley (Ref 18) uses this method in his suite to design single layer electronic circuits. He tries the component in every possible orientation and then selects the solution which gives minimum conductor length.

Rose (Ref 19) uses a sequential placement technique with a "Slot Boundary" dividing the board into used and free areas. This approach is very useful since the components may be positioned into the "SLOTS" to use the board area efficiently.

The quality of layout achieved using a Sequential Placement algorithm is very dependent upon the order in which the components are selected for placement. Many heuristic rules have been implemented governing the order of selection, but the usual technique is to choose the component which is most closely connected to the existing layout at each step. This can introduce an arbitrary decision into the design if several components satisfy the criteria simultaneously. More sophisticated programs exploit this choice by selecting the best layout from a number of different solutions.

Given that each component is placed in an optimal manner, experience has shown that this does not necessarily produce an optimal overall layout. The execution time of this approach, however, is sufficiently lower than the Force Placement method to make this technique particularly attractive for on-line design.

A characteristic of the Sequential Placement algorithm is that the earlier components tend to be well placed while the later ones are rather badly placed.

This is in contrast to the results obtained from the Force Placement algorithm which generally produces layouts in which all the components are equally well (or badly) placed.

### 2.5.3 Choice Of Placement Algorithm

The aim was to produce a suite of programs as a design tool for the engineer rather than a completely automatic system. It was felt that the problems involved in single layer topology require the skill and intuition of an experienced designer to produce a satisfactory layout. Using a Force Placement type algorithm would undoubtedly involve a considerable amount of processing time which is rather unsuited to on-line interaction. It is, of course, possible to carry out the lengthy placement un-supervised, but this would result in a completed layout which may be extremely difficult to edit.

A Sequential Placement algorithm was adopted (Chapters 6 and 7) because of its speed and the fact that the operator can interact throughout the design as opposed to editing a completed layout. In this way he can counteract the inherent fall in placement standard inherent in a sequential approach. This means that the final layout should be very much better than the equivalent un-edited Force Placement solution.

Since an interactive placement technique has been chosen, subsequent Pair Swopping is not required. It would be rather difficult to implement such a system in any case because there is little uniformity in component size.

## 2.6 Automatic Routing Of Interconnections

This facility has been widely developed in other fields - particularly in the case of single sided printed circuit boards. There are basically two methods of doing this for circuits with a single layer of interconnections - "Wavefront" and "Ray-Optic" routing.

### 2.6.1 Wavefront Routing Algorithms

The basic principle of this technique is to start at one component then to search the immediate vicinity in ever widening circles until the target component is found.

This method was first implemented by Lee (Ref 13) in which the board is divided into a grid of squares. Each square round the start component is marked to indicate that it has been examined - any obstacles are represented by special marked squares. A wave of marked squares thus spreads out from the start position until the target is reached. It is then a simple matter to retrace the successful path.

Although this method guarantees to find a path if one exists, it is a relatively slow process and requires a tremendous amount of storage to represent the individual squares. Many versions of the Lee program have been implemented with a view to reducing the execution time and storage required.

One variation proposed by Fisk, Caskey and West (Ref 9) assigns an "altitude" to each square. Obstacles and board edges are represented by "ridges" and "peaks" and the target as a depression. The algorithm finds the most "downhill" solution to the target component.

The author has recently implemented a version of the Lee algorithm to route multi-layer printed circuit boards (Ref 5). Plated through holes are marked with a special code such that the wavefront can change layer at these positions. In addition, each layer can be assigned a bias orientation. In effect, the connections will prefer to run several units along the bias direction than a single unit against it. The result is a set of horizontal and vertical tracking planes. This makes it very easy for the engineer to insert any failed connections interactively.

#### 2.6.2 Ray - Optic Routing Algorithms

An alternative routing method is to project lines along the horizontal and vertical planes in the manner of "Rays" of light. When a Ray encounters an obstacle of some kind it splits into two along the opposite plane. The target is eventually reached by one or more Rays.

A typical example of this technique is the algorithm devised by Hightower (Ref 11). The advantage of this method is that the board need not be divided into a large number of squares, as in the Lee algorithm, so there is a significant saving in core storage. It is also much faster than the wavefront algorithm.

### 2.6.3 Choice Of Routing Algorithm

It is important to decide the criteria which should be used in the routing algorithm to be adopted. The programs written by Agrawal and Breuer (Ref 1), for example, place maximum emphasis on 100% routing. Connection length and processing time are regarded to be of secondary importance. This is reflected in the use of "back tracking" by Kinniment and Weston (Ref 12). All connections are initially routed to find which will fail due to congestion. The connections are then deleted and the "stubborn" ones routed first before the algorithm is re-applied. This increases the processing time significantly, but does ensure that a high percentage of the connections are routed.

Having opted for a quick placement algorithm in section 2.5.3, it is logical to pick a speedy routing algorithm for the same reasons. The philosophy was to have a convenient tool for routing the straight-forward connections which is quick to fail on the more complex ones. A Ray - Optic type algorithm was developed (Chapter 8) which does not split into two when an obstacle is found. Instead, it picks the direction which is most likely to produce the shorter connection. Although the algorithm will not always find a solution, it uses minimal memory storage and is fast enough to be used in an interactive environment.

### 2.7 Computer Aided Photolithography

When the layout has been decided, the computer can be used to produce actual fabrication masks. The advantages of this over a manual technique have been discussed in Chapter 1.

The circuit is usually input to the machine as a list of co-ordinate data. If the design has been performed automatically then there is the additional advantage that the circuit can be accessed directly.

Automatic mask generation using cut and peel material (refer to Chapter 1.2) presents no particular problems to the programmer. An incremental plotter with a knife attachment simply cuts round each circuit element. The only restriction is that the knife must be orientated correctly at each cutting stroke.

An alternative is to use a photoplotter to expose the desired areas



on a sheet of light sensitive material. The programs required to do this are very dependent upon the light source geometries which are available. It is therefore generally difficult to write a program which will control a large range of plotters. A program was, in fact, developed to drive a Ferranti flat bed photoplotter (Chapter 10).

In order to expose the circuit elements in an efficient manner, it is generally accepted that a decomposition into simple geometries is necessary. The problem has also been encountered in the case of electron beam exposure onto integrated circuit photoresist. Direct irradiation is substituted for the photographic masters in this technique. Patel (Ref 17), for example, splits complex polygons into basic three and four sided shapes which are then processed individually. This basic idea has been used in the photoplotter programs described in Chapter 10.

## Chapter 3

### Automatic Design Of Meandering Thin Film Resistors

#### 3.1 Introduction

This chapter describes two programs which can be used to calculate the geometrical shape and size of thin film meandering resistors. The first is very suitable for use with an automated layout and interconnection suite, since it produces resistors with minimum area as the main design criterium. Resistors of this type are constructed such that the area of film affected by heat dissipation is large enough to prevent damage to the circuit caused by overheating.

The second program is particularly useful for manual layouts. It allows the user to define a rectangular boundary then proceeds to fit a resistor within it. This is not always possible so there is the facility for auto-expansion in one or both rectangle dimensions until a solution is achieved. In this program, the onus is on the designer to ensure that the resistor occupies a large enough area to prevent overheating.

Both programs include a number of special "trim" blocks in the resistor geometries. These are used to finely adjust the resistance values after manufacture. Complex formulae and numerous iterations are needed to calculate the number required in each case - a similar manual approach is therefore out of the question.

#### 3.2 Minimum Area Resistors

##### 3.2.1 Geometric Considerations

A thin film resistor relies upon geometric shape rather than size for its resistance. To demonstrate this, consider the square of side 1 unit shown in Figure 3.1. Assume that the resistance from one face to the other is  $R_0$  Ohms. Now consider the larger square measuring 2 x 2 units shown in Figure 3.2. One can see that it can be split into two parallel resistances each of  $2xR_0$  Ohms. Thus the overall resistance measured between opposite faces is still  $R_0$  Ohms, and it follows that a square of ANY size will have the same resistance. The resistance of the thin film layer is quoted in "Ohms per Square" for this reason, which makes the geometric design of a resistor a matter of counting squares.

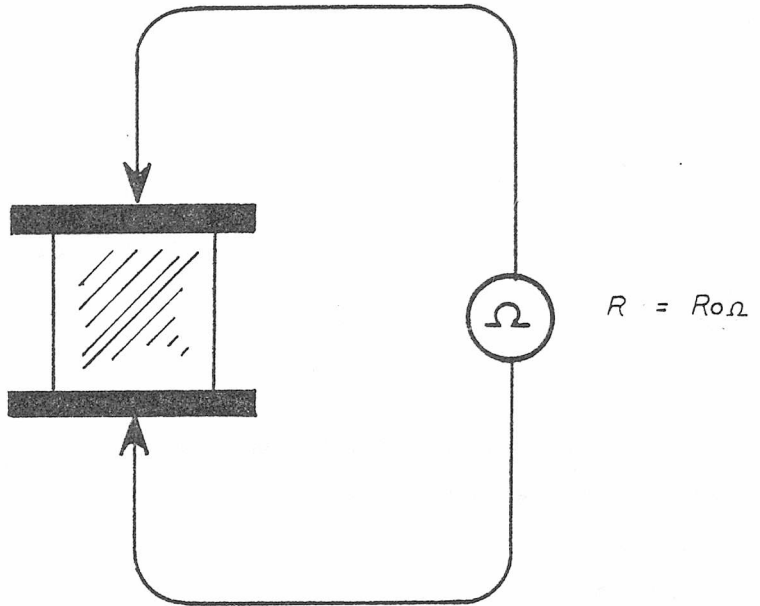


Figure 3.1 RESISTANCE OF A SQUARE OF SIDE 1 UNIT

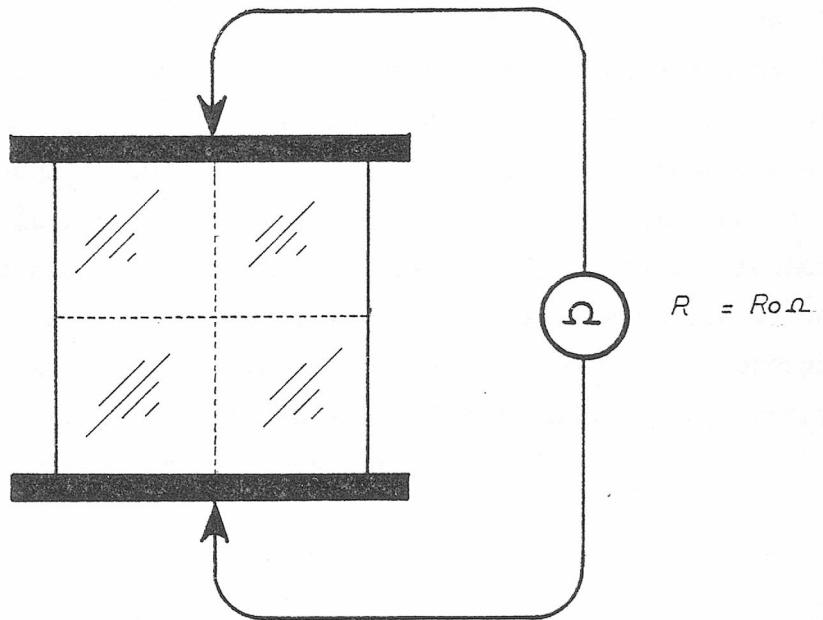


Figure 3.2 RESISTANCE OF A SQUARE OF SIDE 2 UNITS

An example of a "Straight-Line" resistor is shown in Figure 3.3. This is the easiest form of resistor to design, since the number of squares inherent in its shape is simply its length divided by width. We therefore arrive at the equation:

$$R = R_o \times L/W$$

where R = overall resistance

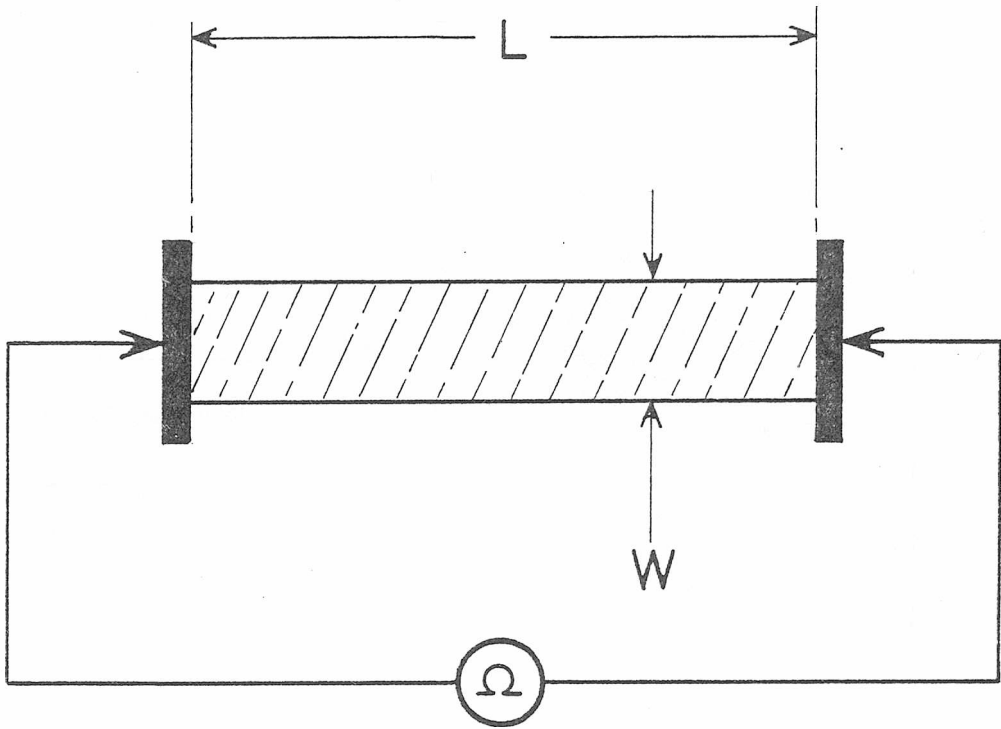
$R_o$  = Resistivity in ohms/sq

L = Length

W = Width

For higher values of resistance, meandering shapes like the resistor shown in Figure 3.4 are preferred in order to use board area more efficiently. Very high resistances need large areas, and it is sometimes better to attach solid state resistor chips to the circuit rather than fabricating them in the film. At the other extreme, very low resistances may be achieved using this technique.

In order to design meandering resistors, some basic knowledge of the mechanics of track corners is needed. Consider Figure 3.5 which shows the current stream lines round a right-angled bend (Ref 2). It is apparent that one cannot assign a whole square of resistance to the corner since a significant area is not used by the flow lines. Indeed, the fraction to be considered depends upon the length of straight track on either side of the corner - the flow lines may not have returned to their normal positions due to a previous bend in the track. Figure 3.6 shows three cases of a right angle bend where the length at one side of the bend is varied (Ref 2). It is assumed that the other side of the bend is not short enough to influence the current flow. From these experimental results it appears that, for zero length, an allowance of 0.469 squares is adequate. This value rises to 0.559 when the length is larger than the track width. To obtain a reasonably accurate value for any length of track, a rising exponential was drawn to pass through these three known points. This is shown in Figure 3.7. The general equation of such a function can be given as:



$$R = L/W \text{ squares} = L \cdot R_0 / W \Omega$$

Figure 3.3 A STRAIGHT LINE RESISTOR

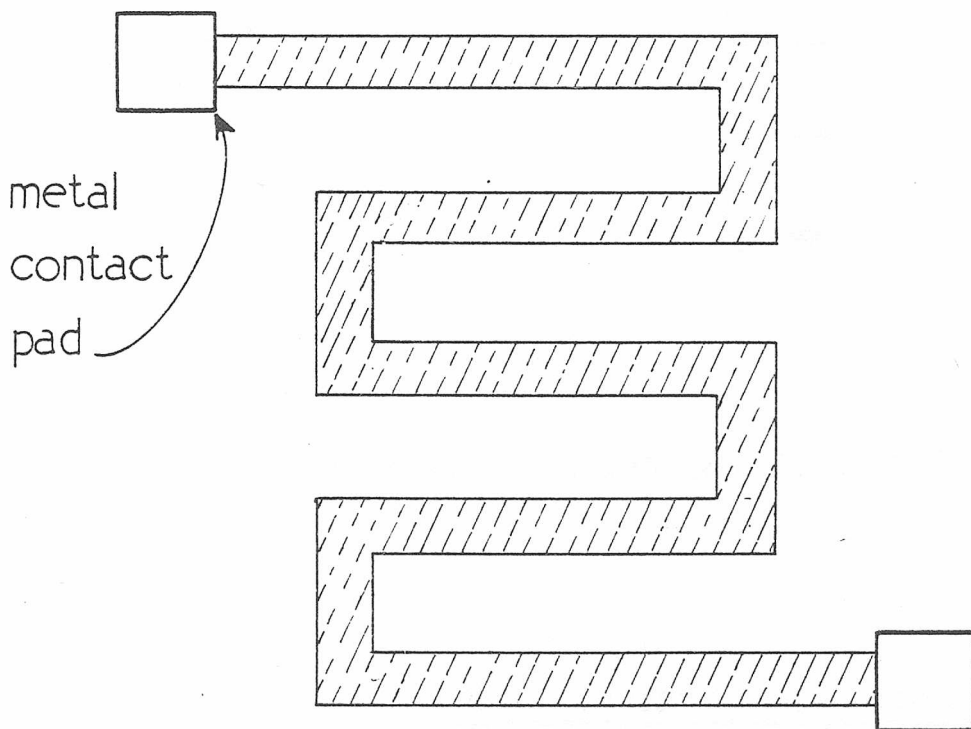
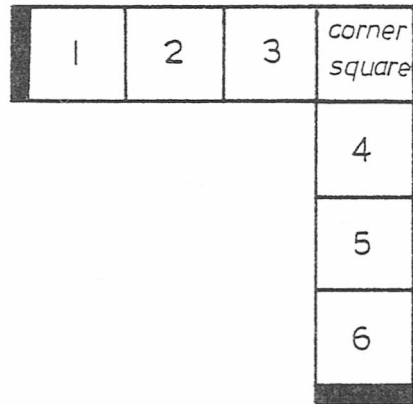
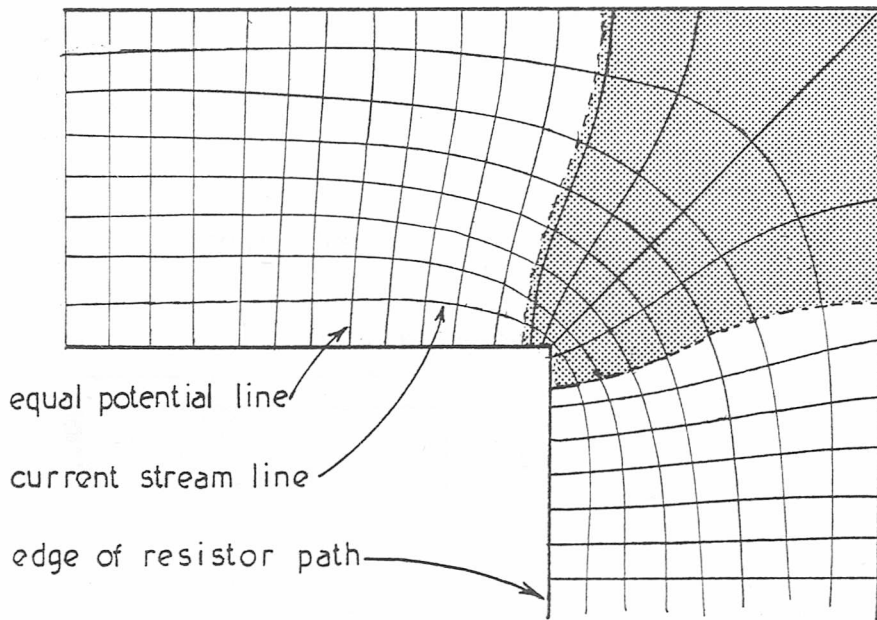
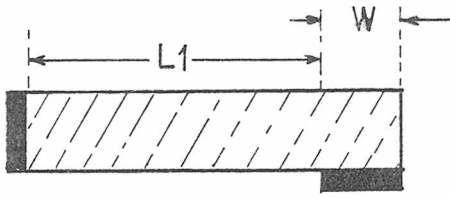


Figure 3.4 A MEANDERING RESISTOR

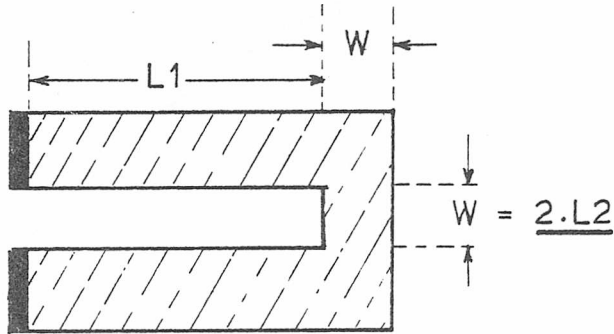


overall resistance = 6 + CORNER ALLOWANCE  $\sim$  6.5 squares

Figure 3.5 RIGHT-ANGLED TRACK CORNERS

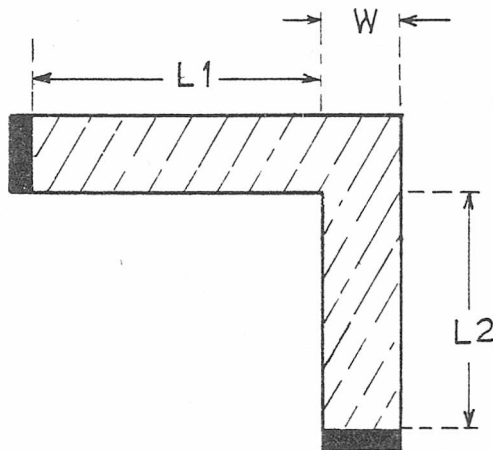


(a) Allowance =  $L1 / W + 0.469_{sq.}$      $L2 = 0$   
 $\Rightarrow$  corner allowance =  $0.469_{sq.}$



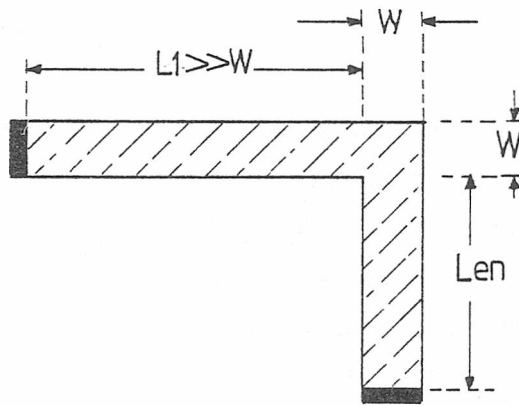
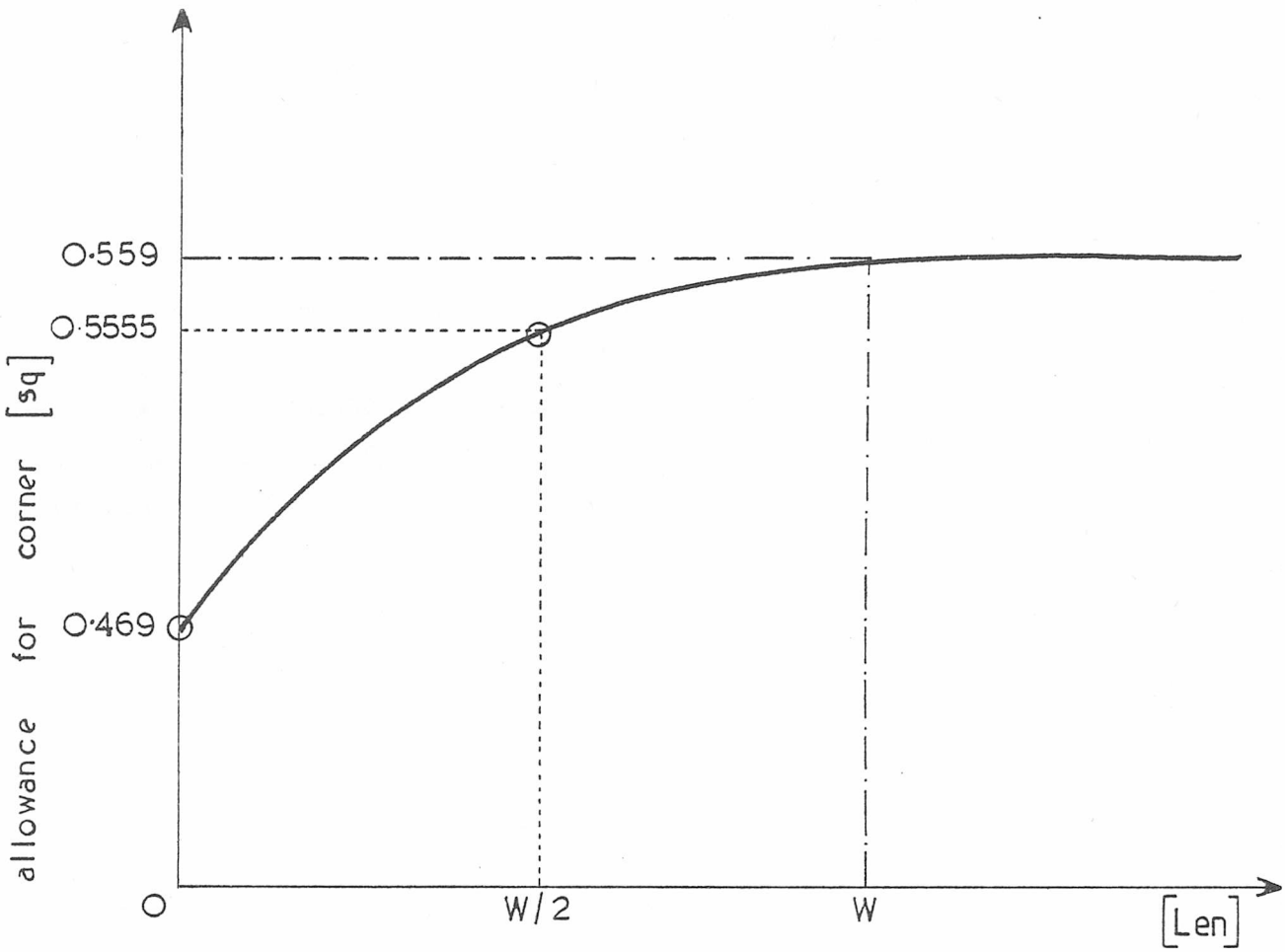
(b) Allowance =  $2.L1 / W + 2.111$      $L2 = W/2$   
 $\Rightarrow$  corner allowance =  $0.5555_{sq.}$

note     metal terminations



(c) Allowance =  $(L1 + L2)W + 0.559$ ,     $L2 > W$   
 $\Rightarrow$  corner allowance =  $0.559_{sq.}$

Figure 3-6 CORNER ALLOWANCES FROM FORMULAE



$$\text{corner allowance} = 0.559 - 0.09 \exp. \left[ -6.4941 \frac{Len}{W} \right]_{sq}$$

Figure 3.7 EQUATION FOR CORNER ALLOWANCE



$$\text{All} = A - B \times \text{Exp} (-C \times \text{Len}/W)$$

where All = Allowance in squares  
 Len = Length of track after bend  
 W = Width of track  
 A, B and C are constants

When Len is very much greater than W we have that:

$$\text{All} = 0.559 \approx A - 0$$

$$\Rightarrow A = 0.559$$

When Len = 0, All = 0.469

$$\Rightarrow 0.469 = 0.559 - B$$

$$\Rightarrow B = 0.09$$

When Len = W/2, All = 0.5555

$$\Rightarrow 0.5555 = 0.559 - 0.09 \times \text{EXP} (-C/2)$$

$$\Rightarrow C = 6.4941$$

We now have the full equation as:

$$\underline{\text{All} = 0.559 - 0.09 \times \text{EXP} (-6.4941 \times \text{Len}/W)}$$

This equation is needed for all right angle Bends.

### 3.2.2 Simple Resistor Geometry

Figure 3.8 shows the basic geometry which has been adopted. The resistor is designed to fit a square bounding rectangle for optimal packing and is defined by the variables N and T. N is the number of half meanders (two in this example), and T is the track width. The terminations are set to have a length of three times the track width and are located at opposite ends of the resistor to counteract any misalignment of the photolithographic mask used in fabrication. In the geometry definition, N is constricted to be an even whole number.

To form an equation for the total resistance it is necessary to break the resistor into simpler geometric shapes. Two distinct shapes are evident here, and are shown in Figure 3.9. Shape A occurs at the beginning and end of the resistor, while shape B occurs "N" times.

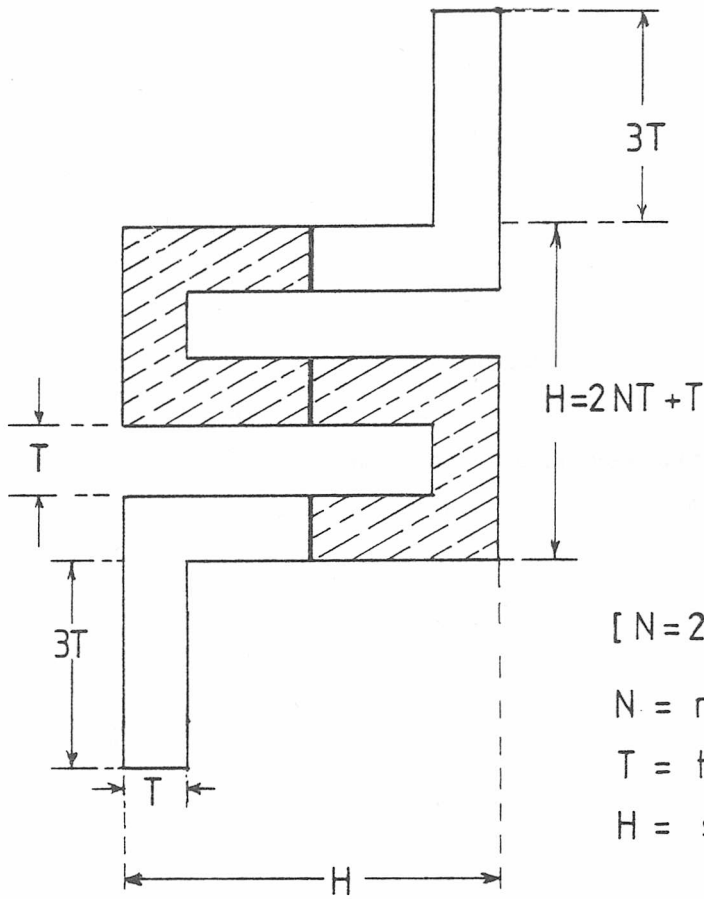
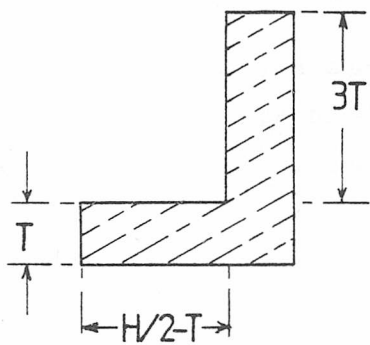


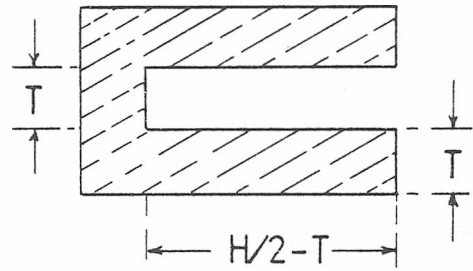
Figure 3.8 Simplest meander resistor



shape [A]

$$R_A = 0.559 + \frac{H/2 - T + 3T}{T}$$

$$= H/2T + 2.559$$



shape [B]

$$R_B = \frac{2(H/2 - T)}{T} + 2.111$$

Figure 3.9 Basic meander shapes and their formulae

Using the corner square formula developed in the last section, we can derive the individual equations for the resistances of these shapes (Ra and Rb). The overall resistance can then be calculated from:

$$R_t = 2 \times R_a + N \times R_b$$

i.e.  $R_t = 2 \times \left( \frac{H}{2 \times T} + 2.559 \right) + N \times \left( 2 \times \left( \frac{H/2-T}{T} \right) + 2.111 \right)$  squares

where

- H = height of resistor (neglecting terminations)
- R<sub>t</sub> = overall resistance
- N = Number of half meanders
- T = Track width

This can be simplified to:

$$R_t = \left( \frac{H-T}{2 \times T} \right) \left( \frac{H}{T} + 0.111 \right) + \frac{H}{T} + 5.118 \text{ squares}$$

From the geometry of the square boundary we have that:

$$H = 2 \times N \times T + T$$

Thus, by substitution:

$$R_t = 2 \times N^2 + 3.111 \times N + 6.118 \text{ squares}$$

i.e.  $R_t = R_o \times (2 \times N^2 + 3.111 \times N + 6.118)$  Ohms

Where R<sub>o</sub> = resistivity in ohms/square.

This equation shows that the track thickness (T) does not affect the overall resistance but merely the physical size of the component and the area it takes up on the thin film slice. It is also apparent that only certain discrete values of resistance can be achieved. These correspond to N = 2, 4, 6, 8, 10 etc. (N must be even to give a solution with terminations at opposite sides of the resistor). A table of resistances for several values of N is given in Figure 3.10.

For any given resistance (R<sub>g</sub>), the required value of N can be found from the root of the above quadratic equation.

N	R (Squares)	R(Ohms) (Res = 400 Ohms/Sq )
2	20.34	8316
4	50.562	20224.8
6	96.784	38713.6
8	159.006	63602.4
10	237.228	94891.2
12	331.45	132580
14	441.672	176668.8
16	567.894	227157.6
18	710.116	284046.4
20	868.338	347335.2
22	1042.56	417024
24	1232.782	493112.8
26	1439.004	575601.6
28	1661.226	664490.4
30	1899.448	759779.2

Figure 3.10

The discrete resistances obtainable

$$\text{i.e. } N = \frac{-3.111 + \sqrt{(3.111)^2 - 8 \times (6.118 - R_g/R_o)}}{4}$$

Naturally, the value of N must be rounded to an even integer. This means that the lower and higher discrete values nearest Rg must be examined to see which is the most suitable to use.

### 3.2.3 Decreasing Resistance By Meander Shrinkage

To reach any arbitrary resistance value, there are two techniques which can be applied. The first is to extend a lower discrete value by the addition of several extra meanders, and the second is to reduce a higher valued resistance by shrinking meanders. The latter will be explained first.

Figure 3.11 illustrates a resistor in which one meander has been "fully shrunk" and one meander has been "partially shrunk". By reducing overall meander length in this fashion, lower resistances may be reached. The drop in resistance due to a fully shrunk meander is a function of N - Figure 3.12 shows the geometry of both a fully shrunk and normal meander. A minimum length of 3 x T has been chosen to ensure a length of at least "T" before and after each right angled bend in the track. This means that the formula used to calculate the corner allowances will still be valid (Refer to 3.2.1.).

It can be seen that the loss in meander length is  $(H - 2xT) - (2 \times T) = (H - 4xT)$ . Thus the TOTAL loss in resistance (allowing for both sides) is  $2 \times (H - 4T)/T$  squares. Using the fact that  $H = 2 \times N \times T + T$  we find that:

$$\begin{aligned} \text{Loss} &= 4 \times N - 6 \text{ squares} \\ &= (4 \times N - 6) \times R_o \text{ ohms} \end{aligned}$$

It will often be necessary to only partially shrink a meander, so we also require a formula to calculate the final length of a partially shrunk meander, given a known resistance loss. Refer to Figure 3.13 which shows the geometry involved.

Assume that the change in resistance must be dR ohms. From the diagram:-

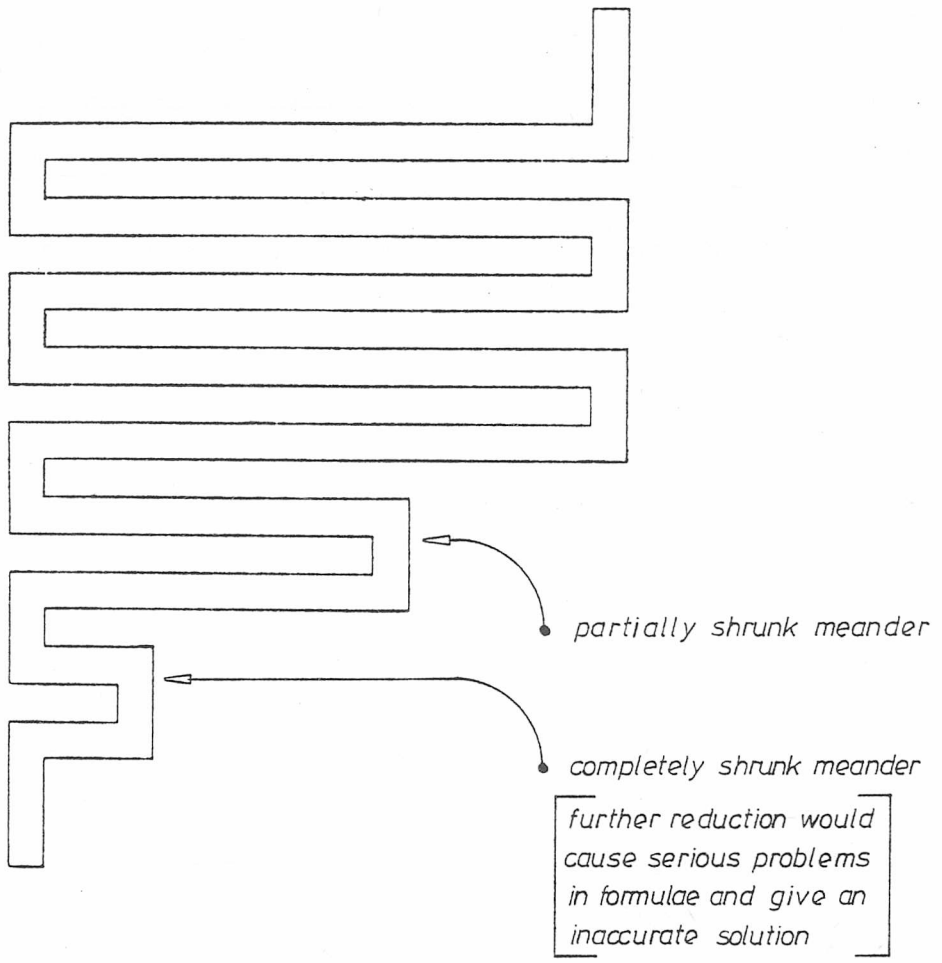
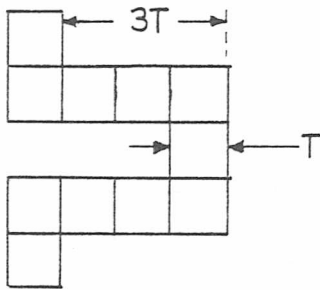
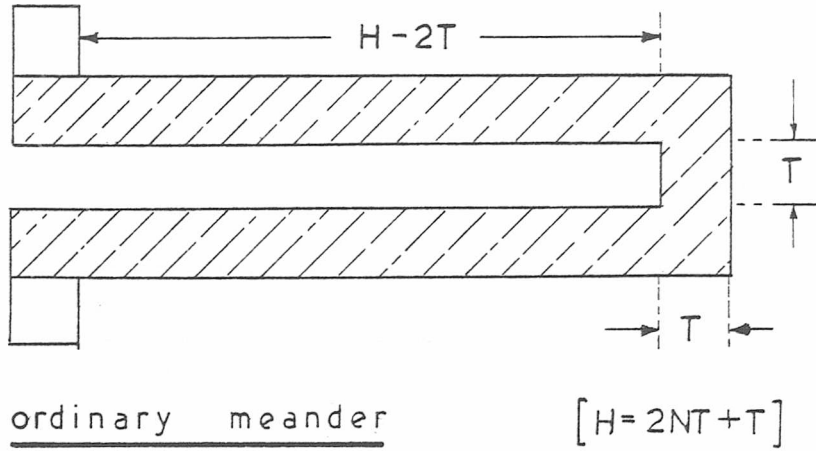


Figure 3.11 Minimum area resistor with  
1 completely shrunk meander and  
1 partially shrunk meander



fully shrunk meander

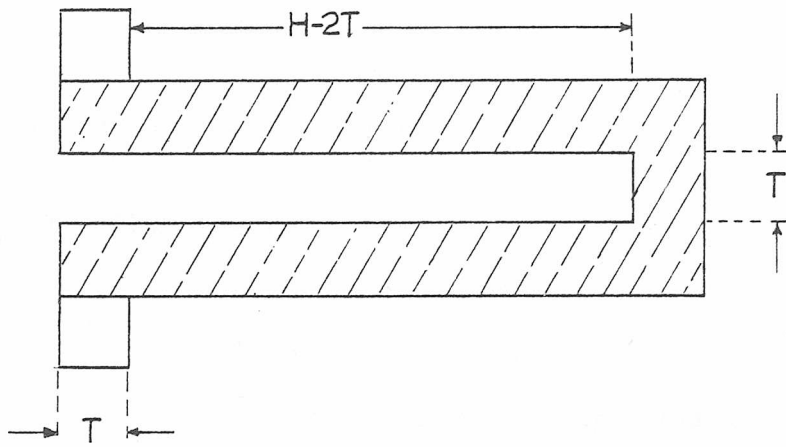
Loss in resistance due to shrinkage =  $R[4N-6]\Omega$

$R$  = resistivity in ohms/square

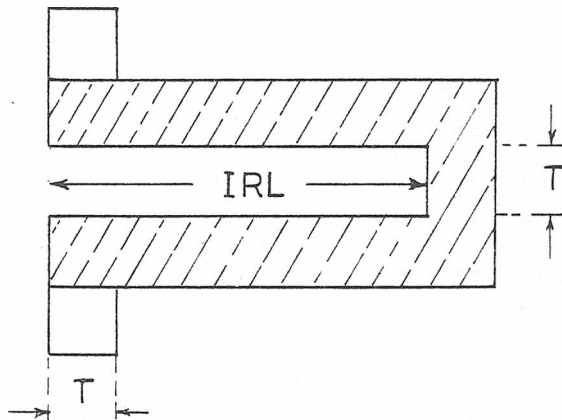
$N$  = number of half meanders in design

$T$  = track width

Figure 3.12 GEOMETRY OF SHRINKAGE



normal meander



partially shrunk meander

$$IRL = 2NT - \frac{dR \cdot T}{2R}$$

where

$R$  = resistivity  $\Omega/\square$

$T$  = track width

$dR$  = loss in resistance ( $\Omega$ )

$IRL$  = length of partial meander

Figure 3.13 GEOMETRY OF PARTIAL SHRINKAGE



$$\begin{aligned} \text{Length of meander} &= H - T = (2 \times N \times T + T) - T \\ \text{before shrinkage} &= 2 \times N \times T \end{aligned}$$

If we create a variable "IRL" to represent the required meander length then the change in length can be expressed as  $2 \times N \times T - \text{IRL}$ . The total change in resistance must then be:

$$dR = 2 \times (2 \times N \times T - \text{IRL}) \times R_0/T$$

Thus:

$\text{IRL} = 2 \times N \times T - dR \times T/(2 \times R_0)$  which will always give a solution for IRL provided that  $dR$  is less than  $(4 \times N - 6) \times R_0$  - i.e. less than the resistance change available from a fully shrunk meander.

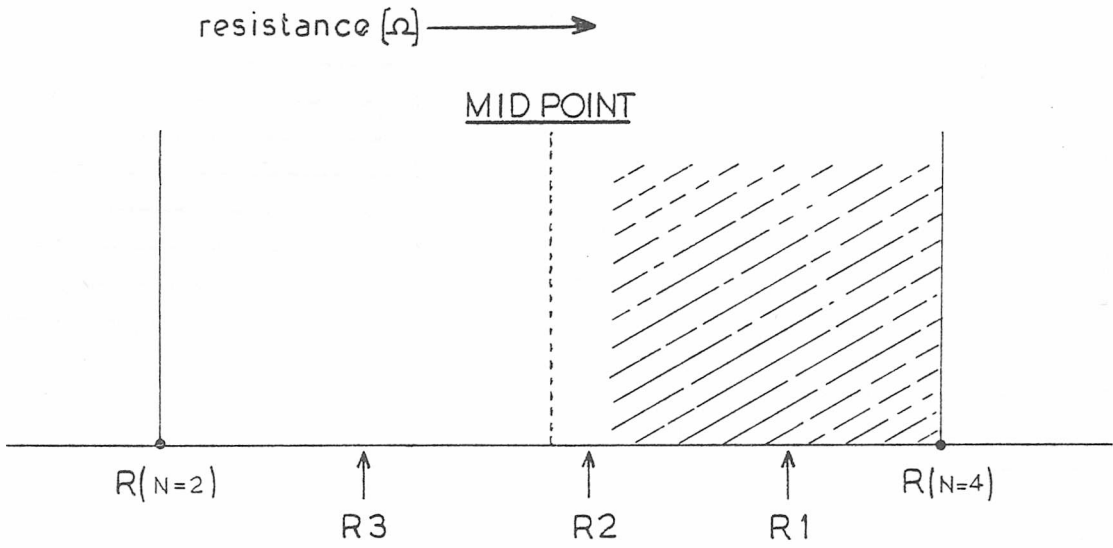
### 3.2.4 Increasing Resistance By Adding Meanders

Consider the resistance values indicated in Figure 3.14.  $R(N=2)$  and  $R(N=4)$  are two discrete resistance values, and the shading indicates those resistance values which can be reached by shrinking the latter. This results from the fact that only  $N/2$  meanders are available for shrinkage in any particular resistor. Values  $R_3$  and  $R_2$  can only be achieved by extending the lower resistor, whereas value  $R_1$  can be reached from either the higher or lower discrete values. The mid-point line usually determines whether shrinking or extension is to be adopted, but only when the shaded region reaches this boundary. Resistances to the left of the line are extended from the lower value and those to the right are reached by shrinking the higher.

The mechanism of extension is illustrated in Figure 3.15. Case (a) shows an extra whole meander added to the resistor, case (b) shows the addition of an extra partial meander and case (c) shows the extension of one termination.

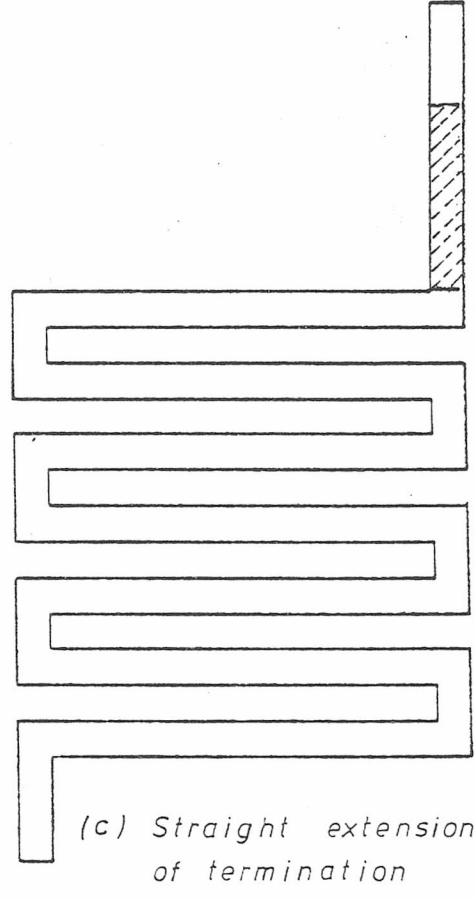
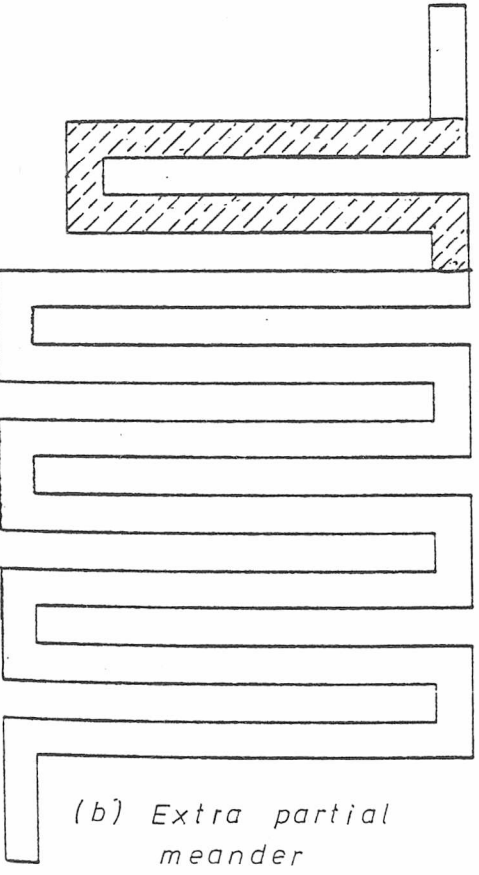
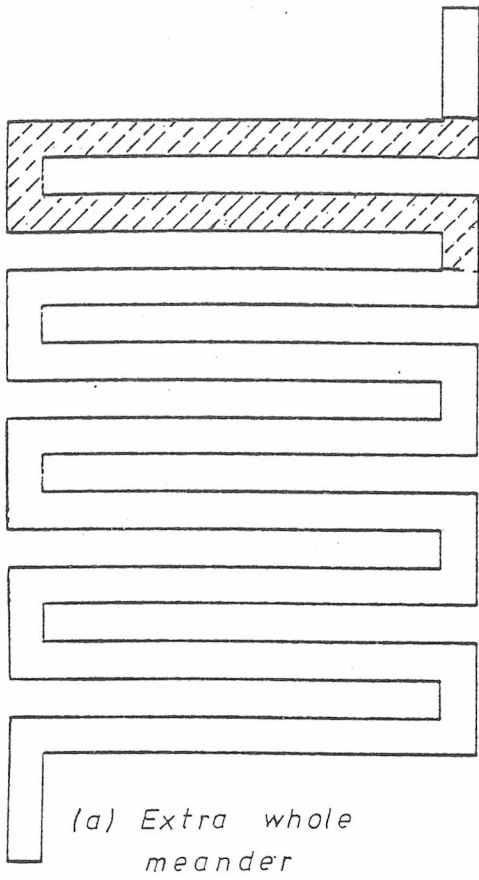
#### 3.2.4.1 Case (a) - Extra Whole Meander

Consider the diagram shown overleaf:



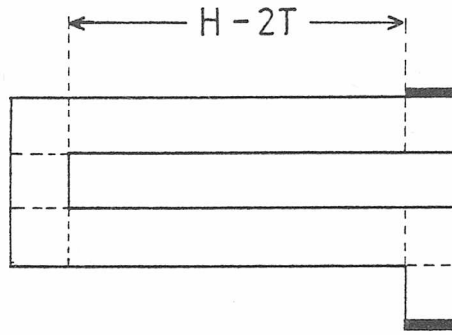
shading indicates resistances  
 which can be reached by  
 "shrinking".  $R(N=4)$

Figure 3.14 Graph of resistance values



[reduction of up to  $2T$  is also possible]

Figure 3.15 Extending minimum area resistors



The shape is made up of four corner squares, two single squares and two lengths of straight track (each  $H - 2 \times T$  long). Using the equation given in Figure 3.7 to calculate the corner allowances, we can formulate an equation for the overall meander resistance to be:

$$\begin{aligned}
 R_m &= 0.559 + 0.5555 + 2 \times (0.5555) + 2 \times (1) \\
 &\quad + 2 \times (H - 2 \times T)/T \\
 &= 2 \times H/T + 0.2255
 \end{aligned}$$

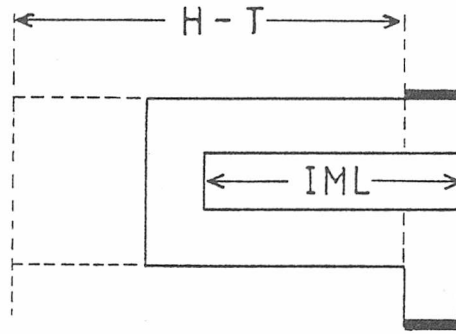
Again, using the fact that  $H = 2 \times N \times T + T$  we have that:

$$\begin{aligned}
 \underline{R_m} &= \underline{4 \times N + 2.2255} \text{ squares} \\
 &= \underline{(4 \times N + 2.2255)} \times R_o \text{ ohms}
 \end{aligned}$$

It is usually only necessary to add one meander, and for most values of error resistance a partial meander is all that is required. This was verified using a short program to iterate through a large range of resistance values and to print out the type of alteration required in each case.

#### 3.2.4.2 Case (b) - Extra Partial Meander

The diagram shown below is similar to that in the previous section except that the meander length is now variable (IML) and is less than  $(H - T)$  which corresponds to a complete meander.



Let the resistance of the meander be  $dR$ . Using the same technique as before we have that:

$$\begin{aligned}
 dR &= 0.559 + 0.5555 + 2 \times (0.5555) + 2 \times (1) \\
 &\quad + 2 \times (IML - T)/T \\
 &= 2 \times IML/T + 2.2255 \text{ squares} \\
 &= (2 \times IML/T + 2.2255) \times R_o \text{ ohms}
 \end{aligned}$$

Thus:

$$\underline{IML = (dR / R_o - 2.2255) \times T/2}$$

and so  $IML$  can be calculated to give the desired resistance increase.

### 3.2.4.3 Case (c) - Straight Extension

The minimum length of  $IML$  in the above equation is restricted to  $3 \times T$  (which corresponds to the same geometry as a fully shrunk meander). This gives a minimum resistance of 8.22 squares. If  $dR$  is less than  $8.22 \times R_o$  ohms then a simple extension is applied to one of the resistor's terminations - as demonstrated in Figure 3.15 (c). The increase in length is calculated from the simple formula:

$$dR = \text{Increase}/T$$

i.e. Increase = dR x T

It is also possible to reduce the termination length provided that the final length is greater than the track width. This can provide a useful drop of up to 2 squares.

### 3.2.5 Simple Design Of Minimum Area Resistors

The flowchart shown in Figure 3.16 shows the "simple resistor" design process. The lowest resistance which can be achieved is when  $N = 2$  which gives a figure of 20.34 squares. Any resistance smaller than this value cannot be designed and the first step in the flowchart is to check for this and, if necessary, initiate the design of a simple "straight line" resistor (refer to section 3.2.1). The next step is to calculate the nearest fixed "N" value resistances on either side of the target value. Several factors decide whether shrinkage or extension is to be applied - if the resistance is nearest the lower value then extension will take place. Conversely, if it is nearer the higher value, shrinkage will occur provided that there is enough "shrinkage" available. The exception to this is when the terminations are lengthened in preference to either of these two methods. Due to the geometry of the resistor it is preferable to shrink the higher valued resistor rather than incur an unusually long final termination.

A composite resistor showing all possible design parameters is shown in Figure 3.17 for reference purposes. This resistor could never occur since the IM, IML and the IR, IRL parameters are mutually exclusive.

### 3.2.6 Laser Trimming

To allow for inaccuracies in fabrication and design, it is necessary to be able to finely adjust completed resistors to their desired values. There are various ways of doing this, but in each case some resistive material is removed causing an increase in resistance. Resistors, therefore, must be designed on paper to be smaller than needed and then "trimmed" after fabrication. The method adopted here is to incorporate one or more "laser trim blocks" into the simple resistor geometries described in the previous section. Figure 3.18 shows a

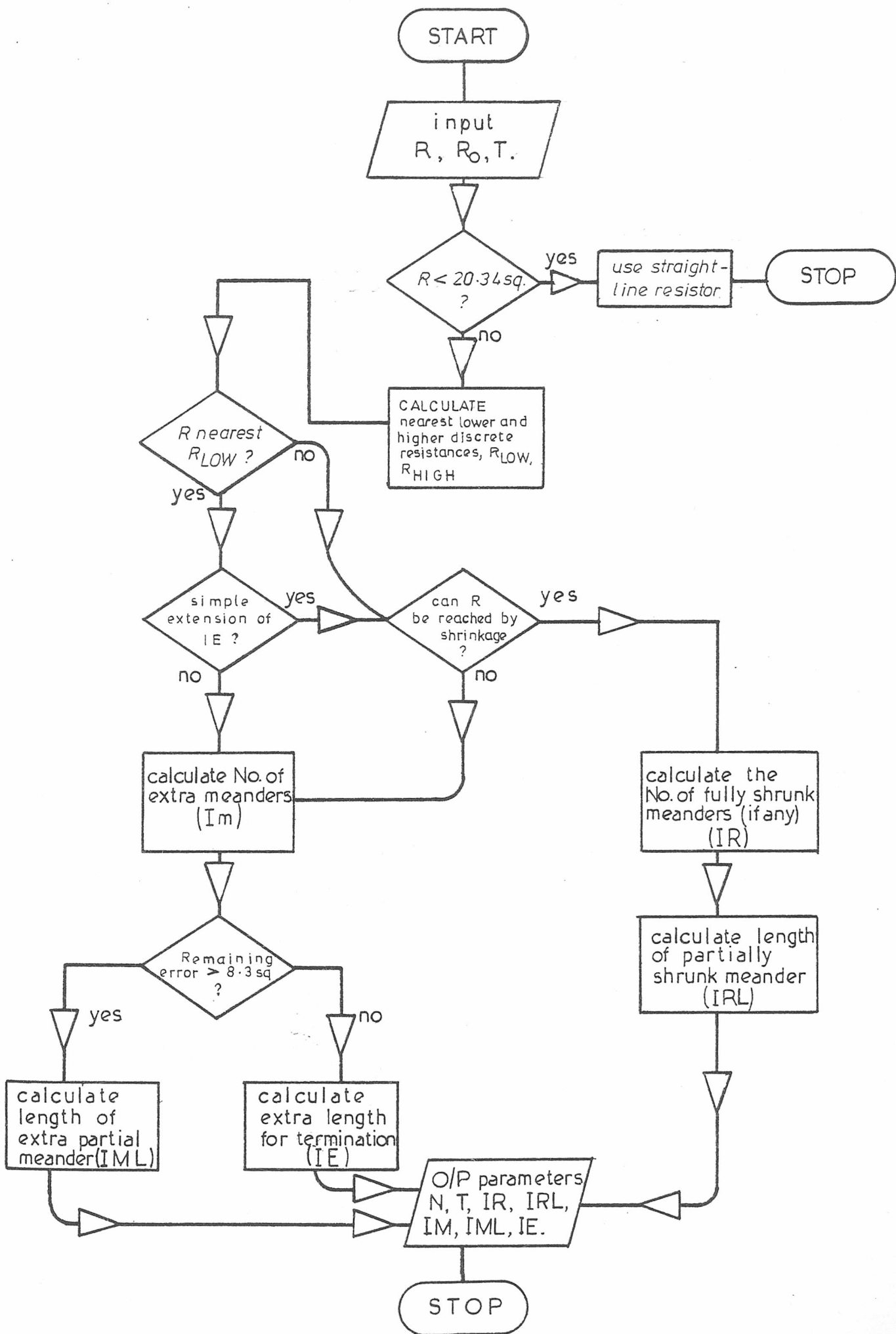
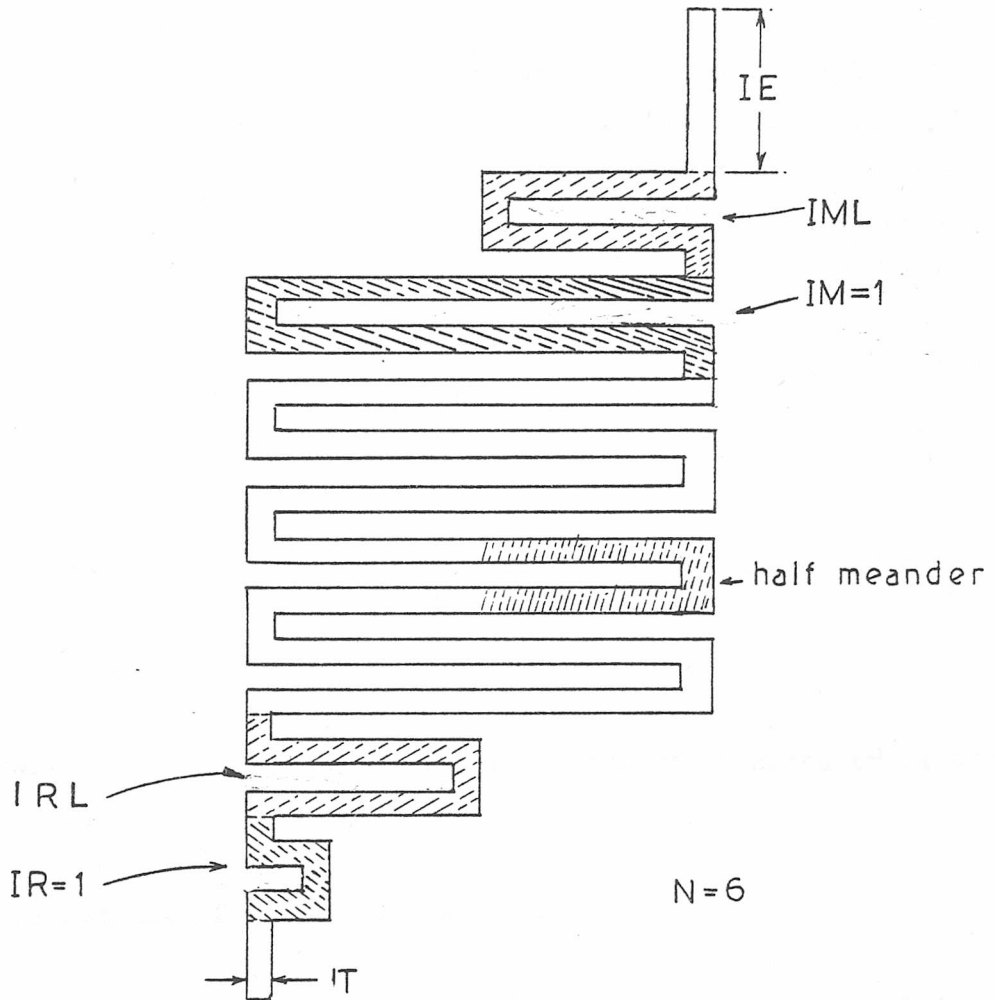


Figure 3.16 Simple Design of Minimum Area Resistors



parameters

- IE - final length of track
- IM - No. of extra meanders
- IML - length of extra partial meander
- IR - No. of completely shrunk meanders
- IRL - length of partially shrunk meander
- N - number of half meanders
- IT - track width

Figure 3.17 MINIMUM AREA RESISTOR PARAMETERS



resistor with a single trim block and also indicates how the block is trimmed. As the laser burns away the resistive material in the block, the overall resistance is continually monitored until the desired value is reached. The number of blocks needed for any particular resistor depends upon the accuracy required and the increase in resistance which can be gained from each block.

### 3.2.6.1 Resistance Of Untrimmed Block

Consider Figure 3.19 which shows the geometry of an untrimmed block. If we bisect the shape we can obtain two new shapes each with resistance R1 (diagram (b)). The value of R1 can be calculated from a formula developed by Bell Telephone Laboratories (Ref 2) to represent the effects of a narrow track merging with a wide track. This gives:

$$R1 = \frac{B/2}{2 \times N \times T + T} + \frac{1}{\pi} \times \left[ \left( \frac{S^2 + 1}{S} \right) \ln \left( \frac{S + 1}{S - 1} \right) - 2 \times \ln \left( \frac{4 \times S}{S^2 - 1} \right) \right]$$

$$\text{where } S = (2 \times N \times T + T)/T = \underline{2 \times N + 1}$$

The resistance of the untrimmed block can now be calculated from:

$$R \text{ untrimmed} = 2 \times R1 - 1 \text{ squares.}$$

The constant of 1 square represents the portion of unwanted track shown in Figure 3.19 (a).

### 3.2.6.2 Resistance Of Trimmed Block

When a block has been completely trimmed out, the resulting shape approximates that shown in Figure 3.20. To calculate the total resistance of this complex shape it is necessary to break it up into several simple shapes with well defined formulae. Three distinct shapes arise - a straight section, two corner squares and two right angled track bends. The total resistance can be found by summing the individual resistances which have been labelled RA, RB and K in the diagram.

#### 3.2.6.2.1 Resistance RA

The right angled bend shown in the diagram has the most complex formula of the three. It is a direct implementation of another formula

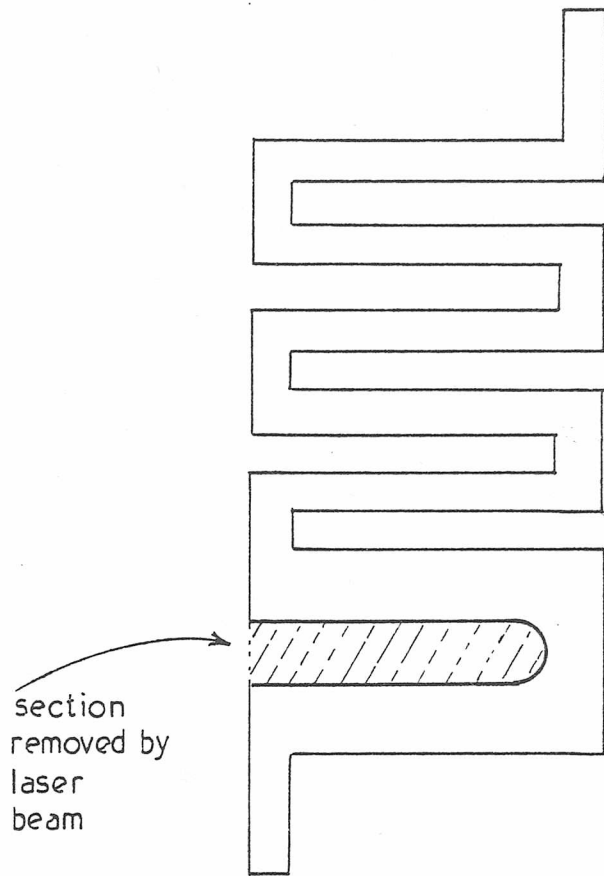
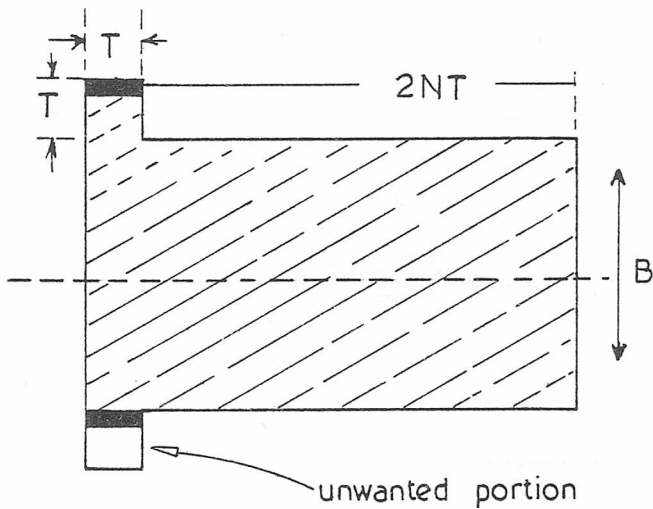
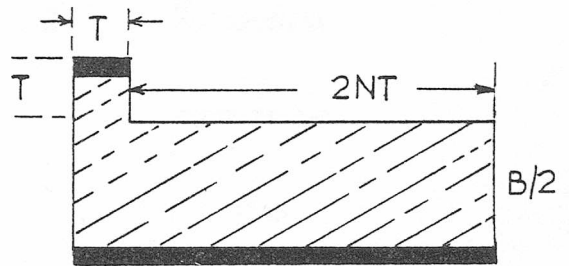


Figure 3-18 LASER TRIMMING



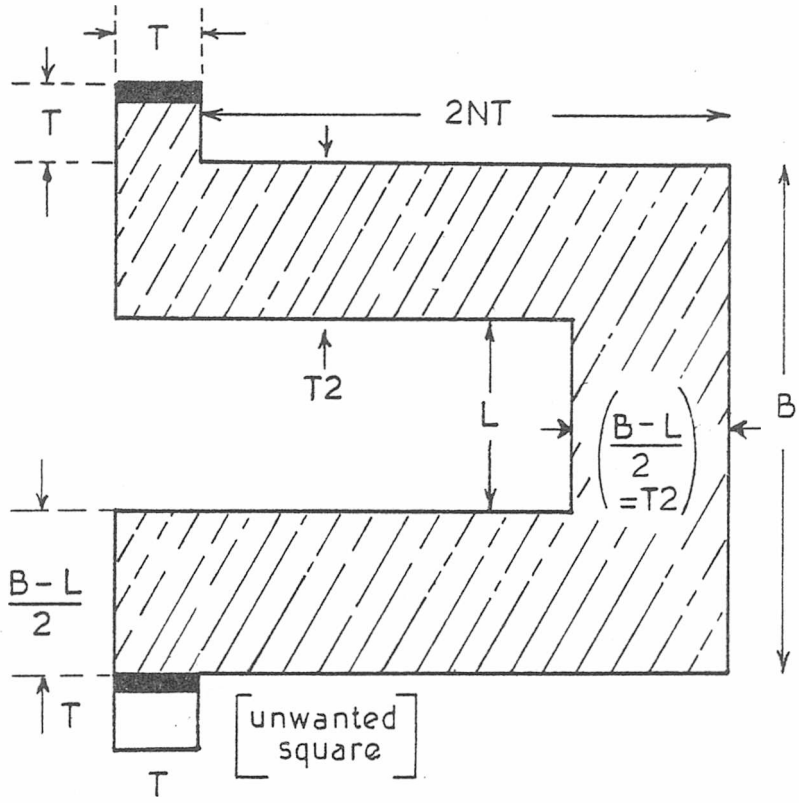
(a) dimensions of untrimmed block



resistance "R1" between indicated faces

(b) basic shape

Figure 3-19 GEOMETRY OF UNTRIMMED BLOCK



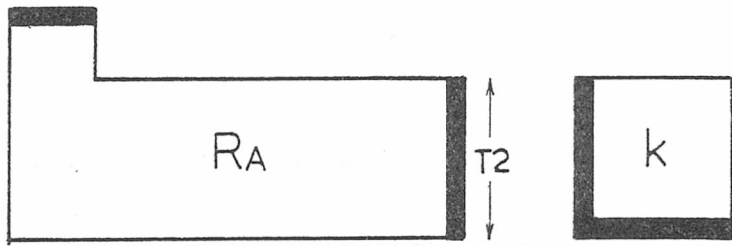
N.B.

L = Laser beam diameter

B = Block thickness

T = Track width

N = No half meanders



TOTAL RESISTANCE =  $(2R_A - 1) + 2k + R_B$

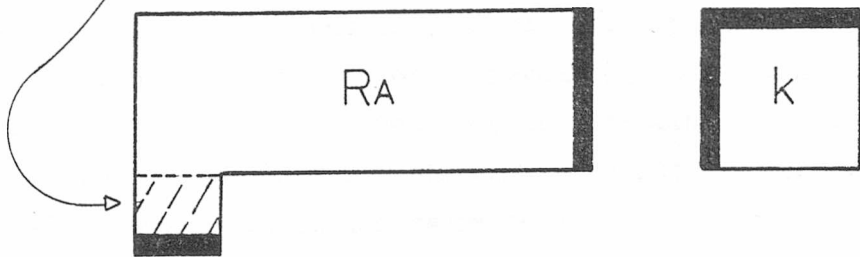
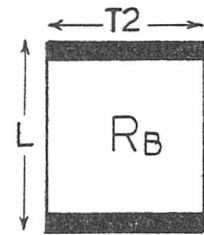


Figure 3.20 GEOMETRY OF A FULLY TRIMMED BLOCK

devised by Bell Telephone Laboratories (Ref 2) and is given by:

$$RA = 2 \times N \times T/T2 + 1/S - \frac{2}{\pi} \ln \left( \frac{4 \times S}{S^2 + 1} \right) + \left( \frac{S^2 + 1}{\pi \times S} \right) \cos^{-1} \left( \frac{S^2 - 1}{S^2 + 1} \right)$$

where  $S = T2/T$  if  $T2 \geq T$

and  $S = T/T2$  if  $T > T2$

$$T2 = (B - L)/2$$

B = trim block thickness

L = laser beam diameter

N and T as before.

### 3.2.6.2.2 Resistance RB

The resistance RB is simply that of a straight length of track thus  $RB = L/T2$  (its length divided by its width).

### 3.2.6.2.3 Resistance K

The resistance to be assigned to the corners depends on the shortest straight length of track on either side. Since both corners share a length of "L", the effective length after each corner is L/2. Using the formula developed in section 3.2.1 we can now find the resistance from:

$$K = 0.559 - 0.09 \times \text{EXP} (-6.4941 \times (L/2)/T2)$$

Having found the resistances of the component shapes it now remains to sum them to find the total trimmed resistance:

$$R_{\text{trimmed}} = 2 \times RA + 2 \times K + 2 \times RB - 1$$

### 3.2.7 Choosing The Number Of Trim Blocks

To allow for variations in film resistivity, mask alignment errors, and other manufacturing inaccuracies, a percentage tolerance is quoted by the designer. This figure is then used to determine the number of trim blocks which should be included. Since the resistance can only be increased with trimming, it is necessary to design the resistor to be smaller than the desired value. For example, if the user specifies

an accuracy of  $\pm X\%$ , the resistor would be designed to be exactly  $X\%$  less than required. In this way, if it turns out to be  $X\%$  higher than it should be, no trimming is needed. Alternatively, should it be  $X\%$  too small, a trim of  $2X\%$  is required to reach the target resistance.

The minimum resistance change required for resistance "R" with percentage accuracy  $\pm X\%$  is given by:

$$dR = 2 \times X \times R/100 = \underline{X \times R/50}$$

Assuming that "If" blocks are needed, we require that:

$$(R \text{ trimmed} - R \text{ untrimmed}) \times \text{If} \geq dR$$

i.e.

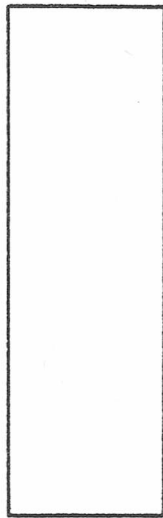
$$\underline{\text{If} \geq dR/(R \text{ trimmed} - R \text{ untrimmed})}$$

where the two resistances refer to the trim block in its trimmed and untrimmed states. It is important to note that the latter values cannot be calculated unless the number of half meanders (N) and the track width (T) are known. (Refer to the formulae derived in sections 3.2.6.1 and 3.2.6.2).

### 3.2.8 Effective Area Of The Resistor

Power rating is an important factor in the design of a resistor. Overheating can result from a badly designed resistor, while it is in-efficient to overestimate the safety margin required. From recent research (Ref 7) it appears that the maximum allowable power dissipation in a thin film resistor is governed by the highest temperature which the film can withstand - itself governed by stability considerations.

Heat flow from the resistor is in two directions - into the substrate and along the surface of the film. Consider the simple resistor shown in Figure 3.21. The effective area is its own area together with the area of film affected by the heat flow perpendicular to the resistor boundaries. It can be seen that the flow in this direction only affects a well defined region of distance "K" away from the resistor. *It has been* shown that the value of K is constant for a

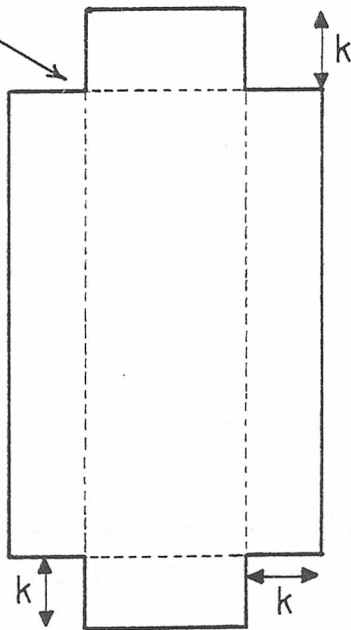


simple

"straight-line"

resistor

NO HEAT FLOW ASSUMED AT SHARP CORNERS



effective area

taken up

on substrate

Figure 3.21 EFFECTIVE RESISTOR AREA DUE TO TRANSVERSE HEAT FLOW

particular microcircuit package (Ref 7 Private Communication).

Since it is only important to find the MINIMUM effective area, we can assume that there is negligible heat flow from corner points. This is quite acceptable since very little heat flow has been observed from sharp points.

Figures 3.22, 3.23 and 3.24 show some of the situations which can arise when meandering resistors are considered. Again, it is assumed that no heat flow arises from corner points and that overlapping areas need only be counted once in the total area calculation. An algorithm has been written to calculate the overall affected film area given any set of resistor parameters, and was tested on all possible configurations. The bulk of the programming is concerned with the detection and processing of overlapping areas.

A resistor which is physically too small can cause serious overheating effects in the resistive layer and an overlarge resistor wastes valuable board area. In the design, an estimation is made of the maximum current or voltage which could arise - this has an associated maximum power dissipation. Every resistive film has an inherent power dissipation figure measured in watts per square metre, so it is a simple task to calculate the minimum permissible area for any particular resistor.

The flowchart given in Figure 3.25 shows the iteration necessary to increase the resistor area from its lowest (using the smallest track width which can be manufactured), to an acceptable value.

The track width is increased by an exponentially decreasing amount as the target value is approached. This minimises computation time.

### 3.2.9 Minimum Area Resistor Design Program - DESRES

Figure 3.26 shows the complete design process needed to synthesise minimum area meandering resistors. This has been implemented in a program called DESRES.

The necessary input data is as follows:-

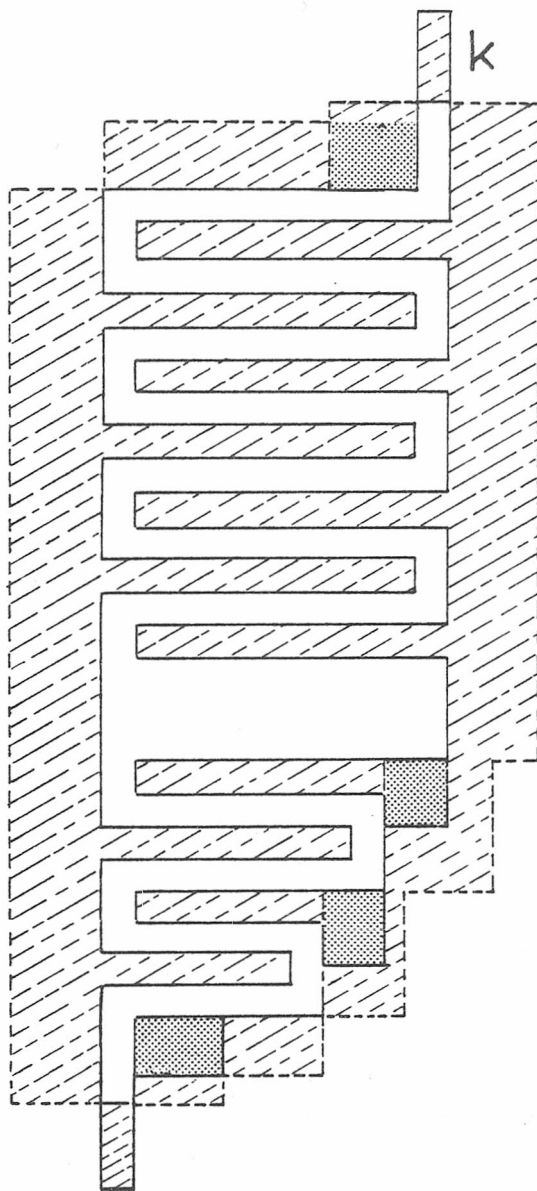


Figure 3-22 Meandering resistor area



*effective area*



*areas to be counted only once*

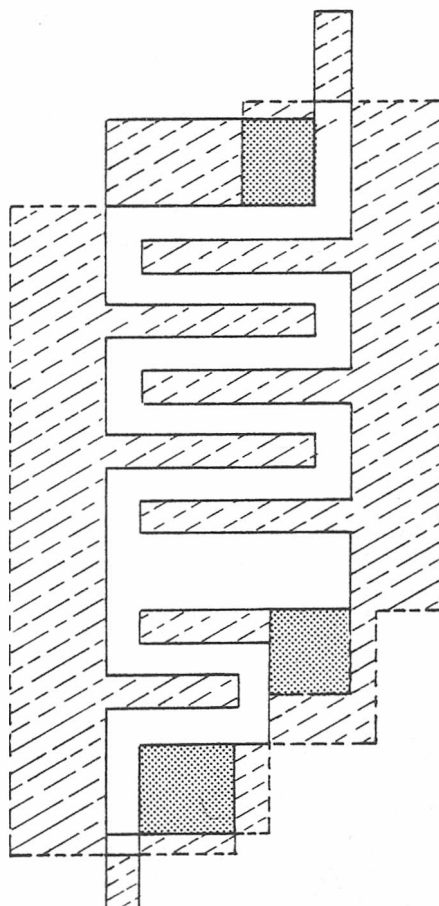
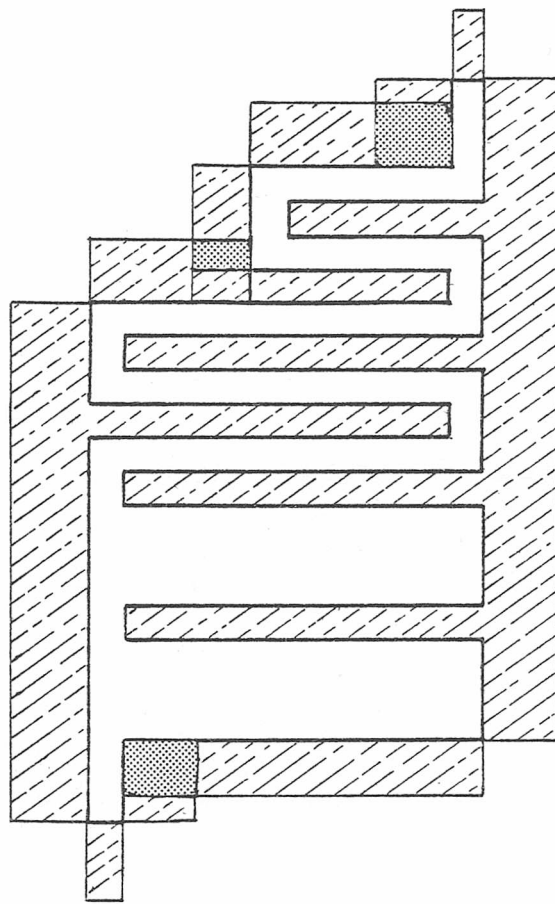


Figure 3-23 Meandering resistor area





*effective area*



*areas to be counted only once*

Figure 3.24 MEANDERING RESISTOR AREA

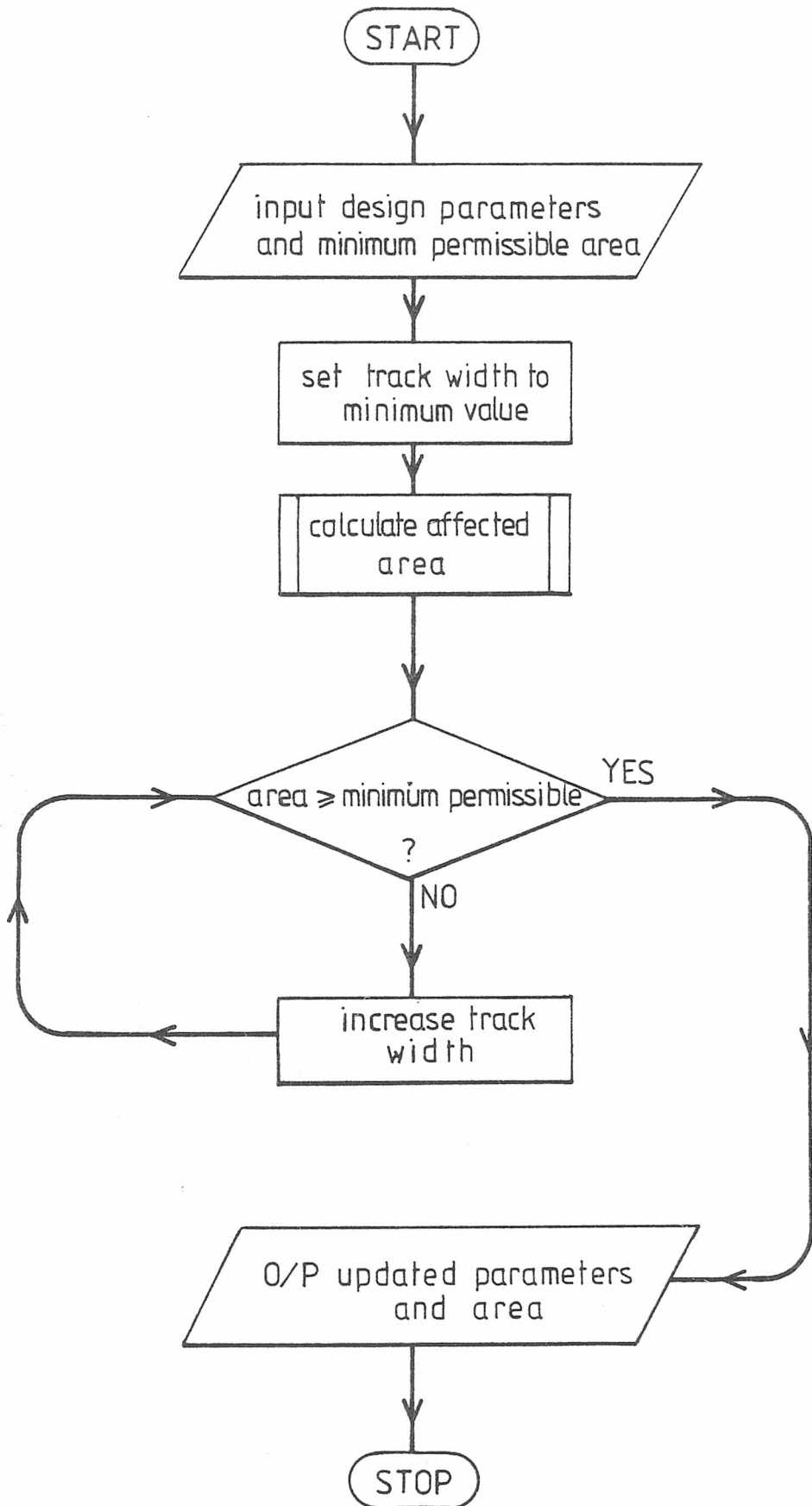


Figure 3-25 AREA ITERATION

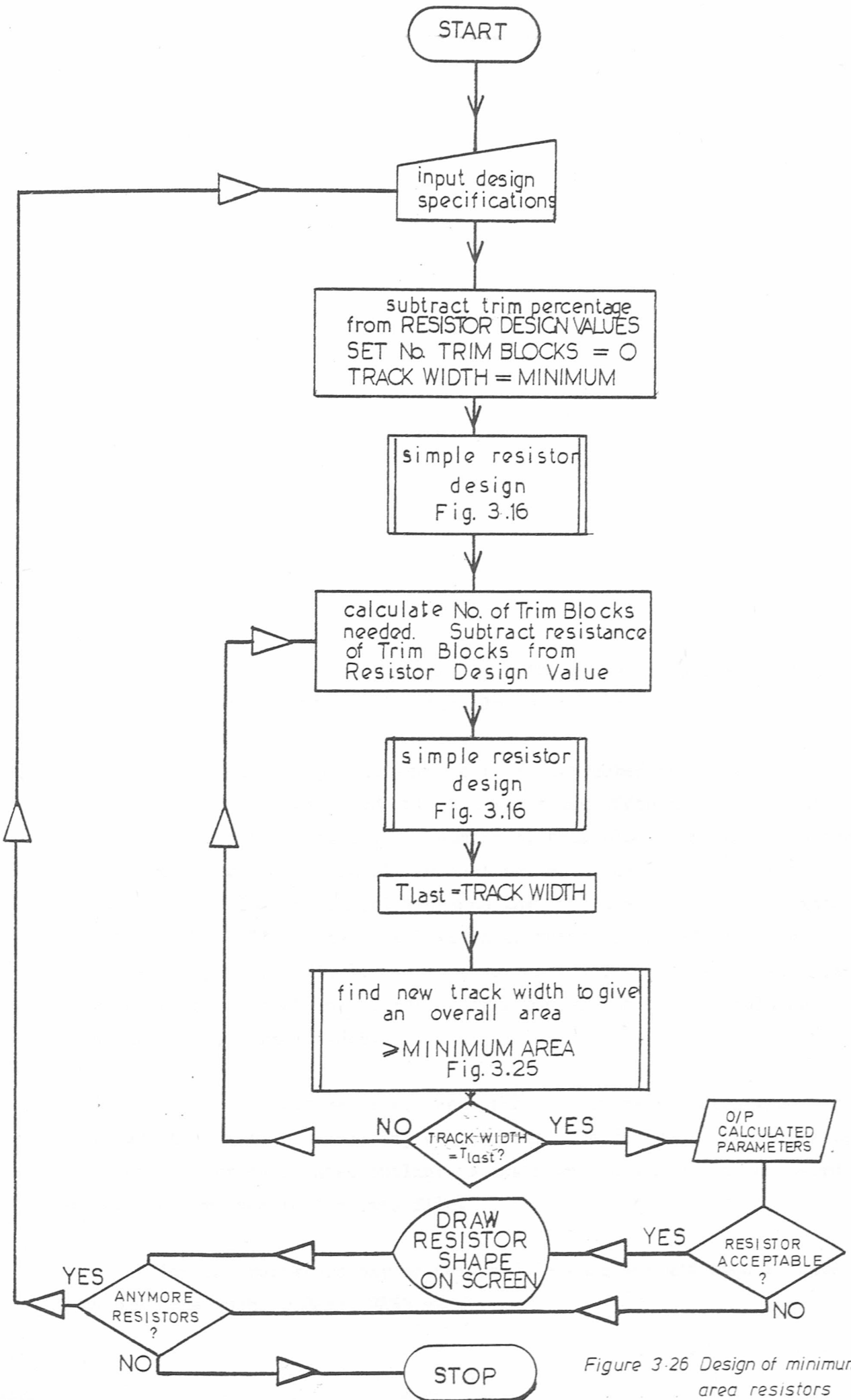


Figure 3-26 Design of minimum area resistors

- (a) Output data file name. This file will ultimately contain the resistor geometries in co-ordinate form. It is used by later interactive placement and routing programs.
- (b) The sheet resistivity in ohms/square
- (c) The laser beam diameter
- (d) Trim block thickness.
- (e) Affected area factor (K) as described in section 3.2.8.
- (f) Minimum permissible track width.
- (g) Resistor value to be achieved.
- (h) Minimum allowable area.
- (i) Percentage tolerance required.
- (j) Resistor's origin coordinates. These are not needed if the resistor is to be automatically positioned on the substrate.

When the trim percentage has been subtracted from the target value, a resistor is designed with minimum track width and no trim blocks. From the parameters produced, the number of trim blocks can be estimated and the resistor is immediately re-designed with the blocks present. The track width is then calculated such that the minimum area restriction is satisfied. If the new track width is the same as before, then the design is complete. If, however, it has changed, an iterative loop is entered to ensure that the number of trim blocks has been calculated using the latest track width.

When a compromise is reached, the parameters are displayed and the user has the option of accepting or rejecting the design. If the resistor is accepted, its calculated outline is drawn on the screen and a co-ordinate description written to the data file.

Several resistors may be designed in this way and each is given a unique name such as RES1, RES2 etc.

Figures 3.27 and 3.28 illustrate the dialogue between user and machine in the design of a typical resistor. This version produces GAELIC (Ref 6) manual input language in the output file - as shown in Figure 3.27(b). There is also a version of this program which produces a file which is compatible with the layout and interconnection programs to be discussed in chapters 4 through 9. In this way the program can be used as an aid to manual layout as well as being an integral part of an overall design suite.

### 3.3 Defined Area Meandering Resistors

#### 3.3.1 Introduction

A second resistor design method has been developed to fit a resistor into a defined area on the substrate. This facility is useful when developing a manual layout since the designer can select his own bounding rectangle, then allow the computer to do the work of calculating a feasible geometry.

A solution is not always possible, of course, since it may be physically impossible to fit the resistor into the given area. The user must then re-specify the rectangle or, more commonly, allow the computer to incrementally expand the rectangle in one or both dimensions until a solution becomes possible. This auto-expansion feature is very useful when one dimension is strictly limited because of other components on the board.

This design method could also be used in the automatic layout suite by deliberately underestimating the required bounding rectangle, thereby forcing an expansion to result. Overheating effects would not be taken into account, however, and the program would invariably take much longer to achieve a solution than the "DESRES" algorithm described in the previous sections.

Figure 3.29 shows the standard geometry assumed in this design algorithm. This is identical to that used in a resistor design program called "GADOR" (Ref 4), but a rather different technique is used to carry out the design. This is due to the fact that GADOR does not automatically include Trim Blocks.

DESRES.FOR

A program to design Thin Film Resistors to occupy an area just larger than that specified

This version is compatible with GAELIC

Enter name for proposed DATA FILE  
TEMP

Input LASER WIDTH and trim BLOCK SIZE  
10,30

Input calculated AREA FACTOR for this circuit  
5

Input SHEET RESISTIVITY value in ohms/sq  
400

Input required RESISTOR ORIGIN coordinates  
354,876

Input RESISTANCE required  
4E4

Input MINIMUM AREA and MINIMUM TRACK width  
0 10

PERCENTAGE +/- trim required ?  
5

N=	6	IT=	10	IE=	30	NTRIM=	1
IR=	0	IRL=	81	IM=	0	IML=	0

Do you accept this resistor solution ?  
YES

(a) Program Dialogue

```
"POLYGON"(1) 354,876:0,30,0,10,81,10,-81,20,0,40,0,10,120,10,  
-120,20,0,10,120,10,-120,20,0,10,120,30,10,-40,-120,-10,120,  
-30,-120,-10,120,-30,-120,-10,120,-30,-120,-10,81,-30,-81,  
-10,0,-20,-10,0;  
"FINISH";
```

(b) "GAELIC" data produced to describe Resistor

Figure 3.27 Operation of the program "DESRES"

LASER WIDTH = 10 BLOCK SIZE = 30  
RESISTANCE = 38000. RESISTIVITY = 0 WITH 5 +/-  
AREA = 24665 TRACK = 10  
NO. TRIM BLOCKS = 1 IE = 30 N = 6  
IR = 0 IRL = 81 IM = 0 IML = 0

Resistor height is 11.5000 thous

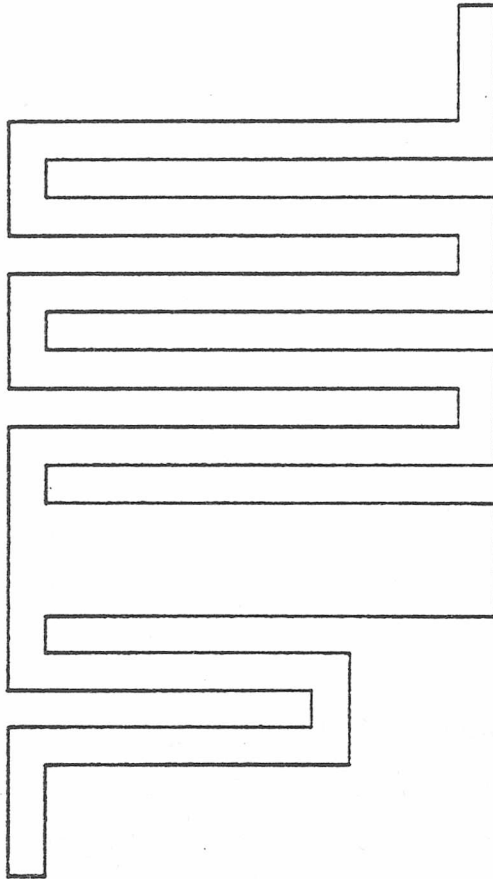
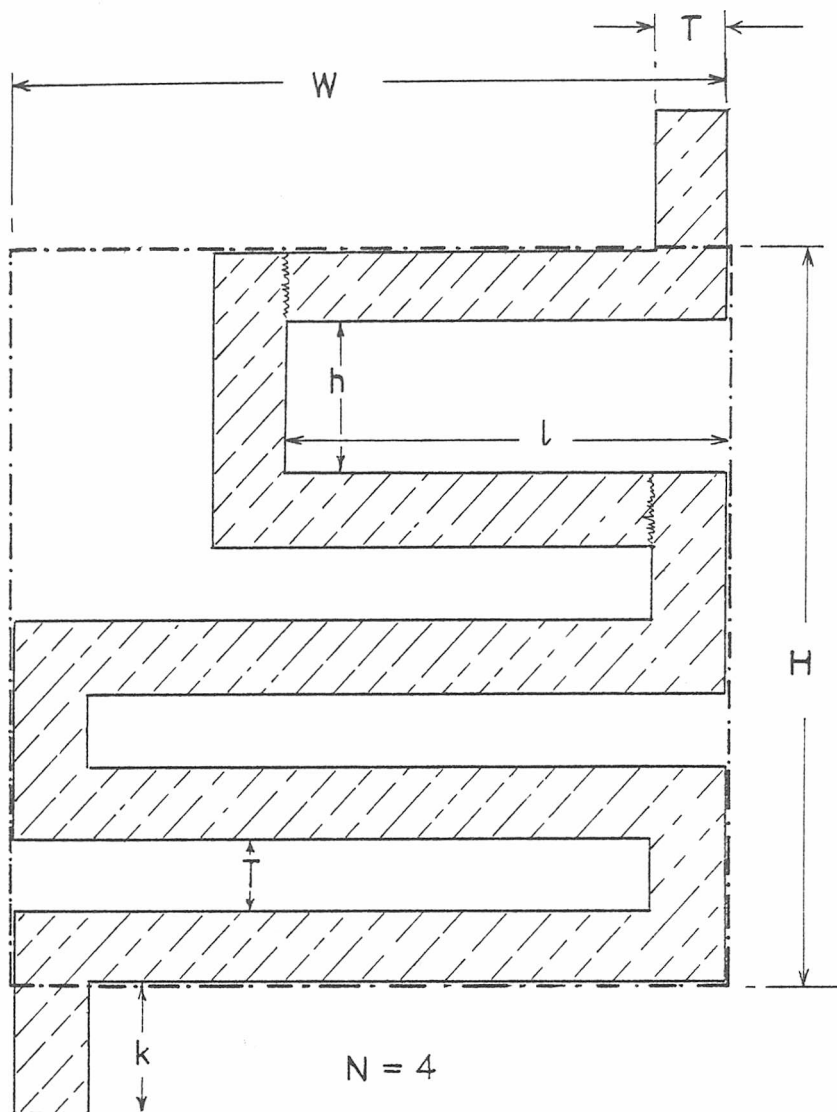


Figure 3-28 Graphical output from "DESRES"



- T = width of resistor track
- h = height of final meander
- l = length of final meander
- k = length of terminations
- W = defined width of bounding rectangle
- H = defined height of bounding rectangle
- N = number of half meanders
- $R_0$  = resistivity in  $\Omega/\text{square}$
- R = overall resistance in  $\Omega$

$$R = R_0 [ (N-1) W + 2l + 2k + h - 2T ] / T$$

Figure 3.29 Geometry of a defined area resistor



### 3.3.2 Simple Design Of Defined Area Resistors

The resistor is defined by the number of half meanders (N), its track width (T), the dimensions of the rectangle (H and W), and the length of its terminations outside the rectangle (K). It also has two areas of freedom, namely its final meander height (h) and length (l). The formula for overall resistance (as developed in GADOR) can be defined by:

$$R = R_o \times ( (N-1) \times W + 2 \times l + 2 \times K + h - 2 \times T ) / T \quad - (a)$$

where:

R = overall resistance

R<sub>o</sub> = sheet resistivity in ohms/square

The bounding rectangle height (H), is related to the resistor parameters by the formula:

$$H = 2 \times N \times T + h \quad - (b)$$

If we initially assume that the resistor is a perfect fit - i.e.  $h = T$ , then we can re-arrange this formula to calculate N:

$$N = (H/T - 1)/2$$

N must be an integer - if it is odd, then the terminations will be at the same side of the bounding rectangle. An even-numbered N will result in terminations at opposite sides. It is up to the user to decide which option is to be preferred. (The resistor in Figure 3.29 has terminations at opposite sides of the bounding rectangle. An example with terminations at the same side is given in Figure 3.31).

Having determined an integer value for N using the minimum track width, we can now calculate the final meander height by re-arranging equation (b) to give:

$$h = H - 2 \times N \times T$$

If this value of h is less than zero, it means that the design is impossible and the user must re-define the bounding rectangle. This can either be achieved manually or using the auto-expansion facility.

The process is completed by calculating the length of the final meander (l). By re-arranging formula (a), we have that:

$$l = (R \times T / R_o - (N - 1) \times W - 2 \times K - h + 2 \times T) / 2$$

If the value for  $l$  turns out to be negative, it implies that the track width ( $T$ ) is too small (i.e. the resistance is too high). When this situation arises, the track width is incremented and the design process repeated.

To obtain a meaningful solution we require that " $l$ " be greater than  $2 \times T$ , and less or equal to  $(W - T)$ . The track width is initially set to the minimum permissible value and gradually increased until the final meander length is greater than the minimum figure. If " $l$ " is then found to be greater than  $(W - T)$ , it is clipped to this value thereby causing an unavoidable inaccuracy in the resistor design. This % error is given as output data from the program.

Figure 3.30 illustrates the flowchart describing this technique in full.

### 3.3.3 Laser Trimming

Trim blocks similar to those described in section 3.2.6 are now assigned. The formulae for the block in its trimmed and untrimmed states still apply. The only difference is that the horizontal width of the trim block is now fixed at " $W$ ", the width of the bounding rectangle. In the original formulae it was a function of track width and the number of half meanders.

Figure 3.31 shows how the blocks may be added to the resistor patterns. The required number of blocks is calculated and the defined rectangle height " $H$ " reduced accordingly. A resistor is then designed to fit the remaining space in the rectangle. The resistance of the untrimmed blocks must be taken into account, of course, and this further complicates the algorithm.

### 3.3.4 Bounded Area Resistor Design Program - DEBOR

Figure 3.32 shows the flowchart of a program called "DEBOR" which has been written to design Bounded Area resistors using the techniques explained in the previous sections. The input data required is very similar to that of "DESRES", except that the user must specify a set of rectangle dimensions. A typical dialogue is given in Figure 3.33(a).

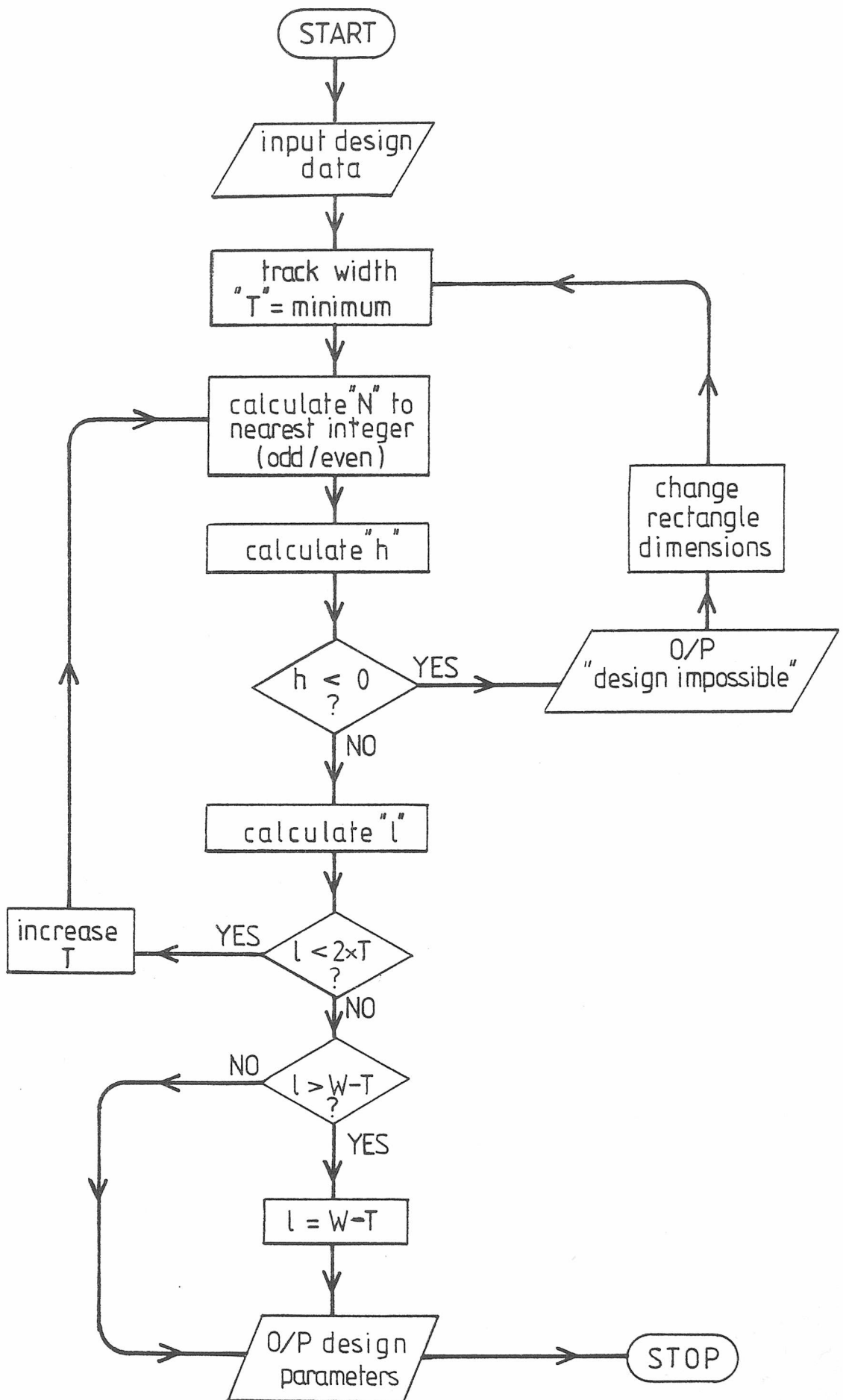
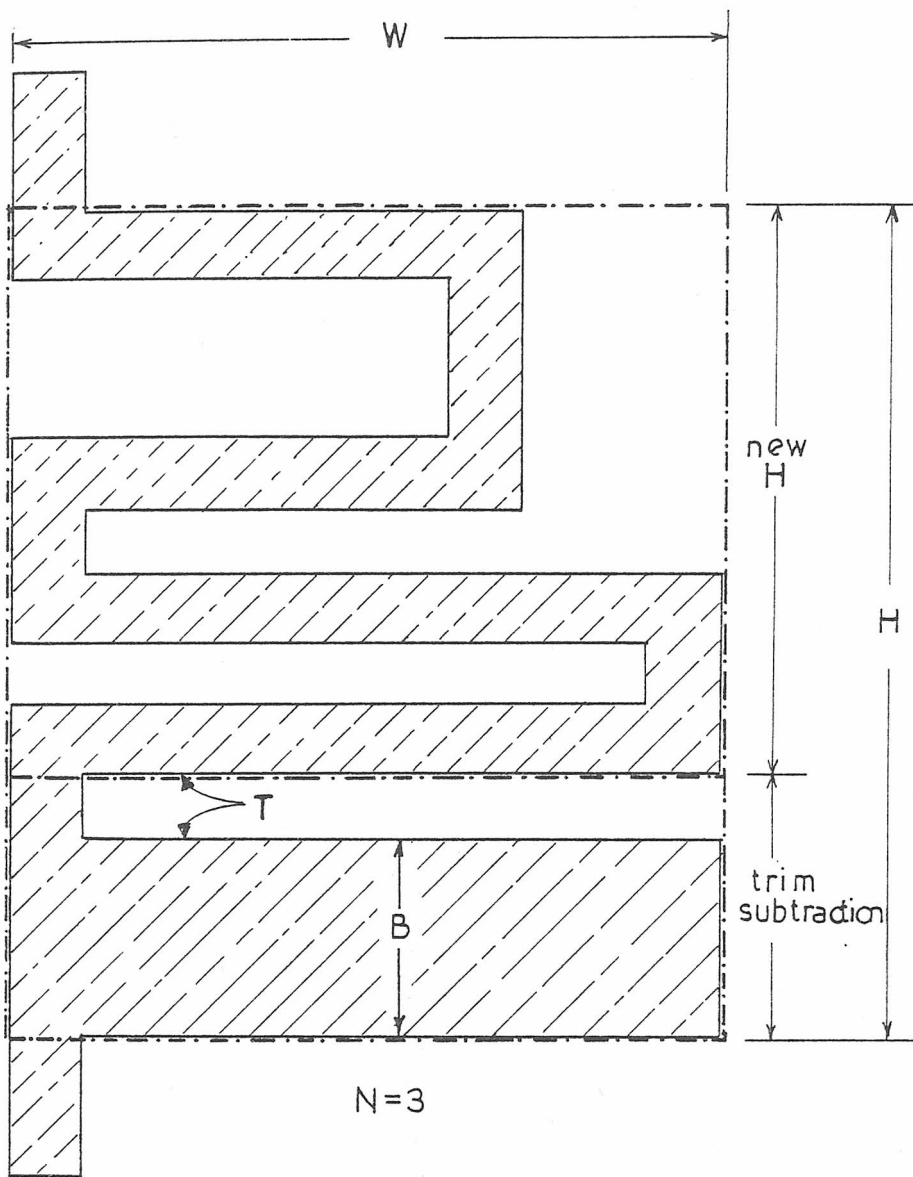


Figure 3.30 Simple "defined resistor" design



Number of trim blocks =  $N_B (=1)$

Reduction in defined height =  $N_B [B+T]$

$B$  = trim block thickness  
 $T$  = track width

Figure 3.31 Defined area resistor with trimming block

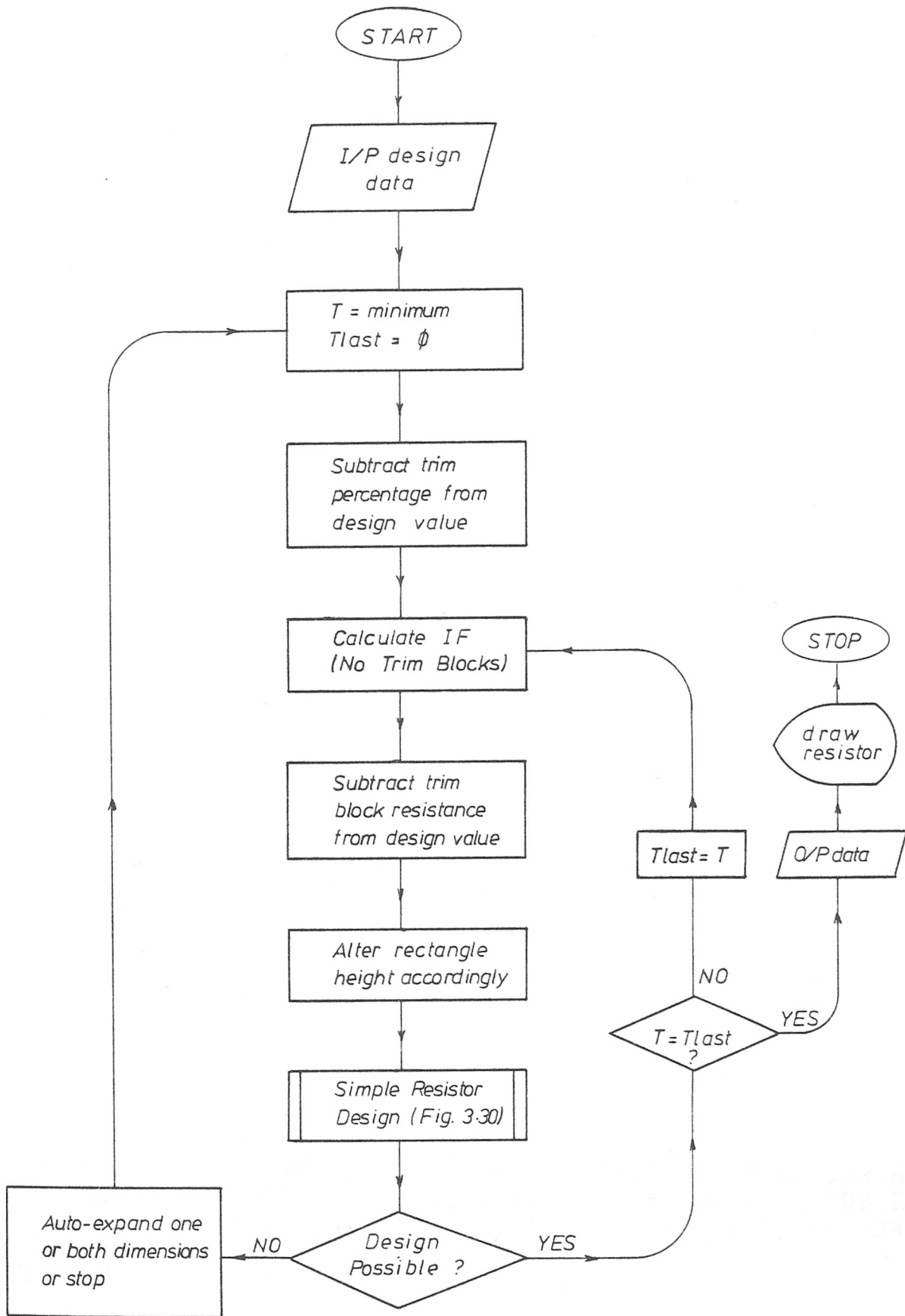


Figure 3-32 The full design process (program DEBOR)

DEBOR.FOR

A program to design meandering Thin Film Resistors to fit a bounding rectangle

Enter name of DATA FILE to be produced  
TEMP

Enter LASER DIAMETER and TRIM BLOCK size  
10,30

Enter sheet RESISTIVITY in ohms per square  
400

INPUT RESISTOR ORIGIN COORDINATES  
867,489

Enter WIDTH and HEIGHT of bounding rectangle  
400,460

Enter RESISTANCE value required  
8E4

Enter trim PERCENTAGE +/- required  
5

How long do you want the TERMINATIONS to be ?  
30

Do you want the terminations to be on the same side of the bounding rectangle ?  
NO

(a) Computer dialogue

```
"POLYGON"(1) 867,489:0,30,0,50,0,20,380,20,-380,60,380,20,-380
60,380,20,-380,60,380,20,-258,130,258,30,20,-50,-258,-90,258,
-60,-380,-20,380,-60,-380,-20,380,-60,-380,-20,380,-60,-380,
-20,380,-30,-380,-30,-20,0;
"POLYGON"(2) 867,489:400,520,-400,-520;
"FINISH";
```

(b) "GAELIC" data file produced

Figure 3.33 Operation of program "DEBOR"

There are three main stages in the program:-

- (a) Calculation of the number of Trim Blocks.  
This uses the current value of track width and the bounding rectangle width.
- (b) Calculation of available rectangle height. This requires the number of Trim Blocks and the track width.
- (c) The simple resistor design which uses the available rectangle height, rectangle width and the number of trim blocks to produce new values for the track width and the number of half meanders.

Since each stage depends upon the results of at least one other, the program must cycle through this sequence of calculations until the design parameters reach steady values. This is shown as a simple loop in the flowchart given in Figure 3.32.

If the program fails due to lack of space, there is the option of auto-expansion. This involves a gradual increase in one or both rectangle dimensions until a solution is obtained. The increment size is initially set high and rapidly reduced to give a result accurate to 1 screen unit (0.00005 inches, if photoplotting carried out at a scale of 20:1).

The calculated parameters and error percentage are displayed, and the user has the choice of accepting or refusing the resistor. If accepted, it is drawn on the screen and a set of co-ordinate data written to a GAELIC language file (an example of which is shown in Figure 3.33(b)).

Figure 3.34 shows the type of graphical output that is obtained.

DATA GIVEN BEFORE DESIGN :

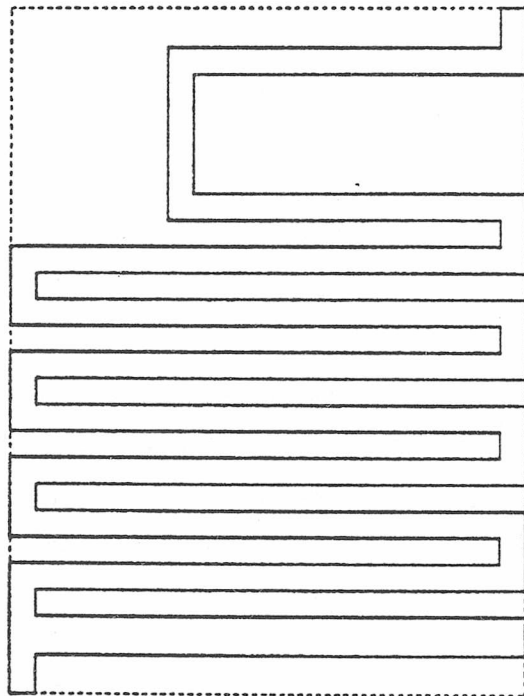
DESIGN RES = 80000.00     $\times$  +/- = 5.000000  
LASER BEAM SIZE = 10 TRIM BLOCK SIZE = 30 TERMINATIONS = 30

CALCULATED PARAMETERS:

N = 8 IF = 1 LAST MEANDER LENGTH = 278  
LAST MEANDER HEIGHT = 90 TRACK WIDTH = 20  
FINAL HEIGHT = 460 FINAL WIDTH = 400  
 $\%$  OUT ON DESIGN = 0.2398546E-01

DO YOU ACCEPT RESISTOR ? YES

Height of resistor is 26.0000 thous



—— mask 1

----- mask 2

Figure 3-34 Graphical output from "DEBOR"



## Chapter 4

### Describing and Representing A Circuit

#### 4.1 Introduction

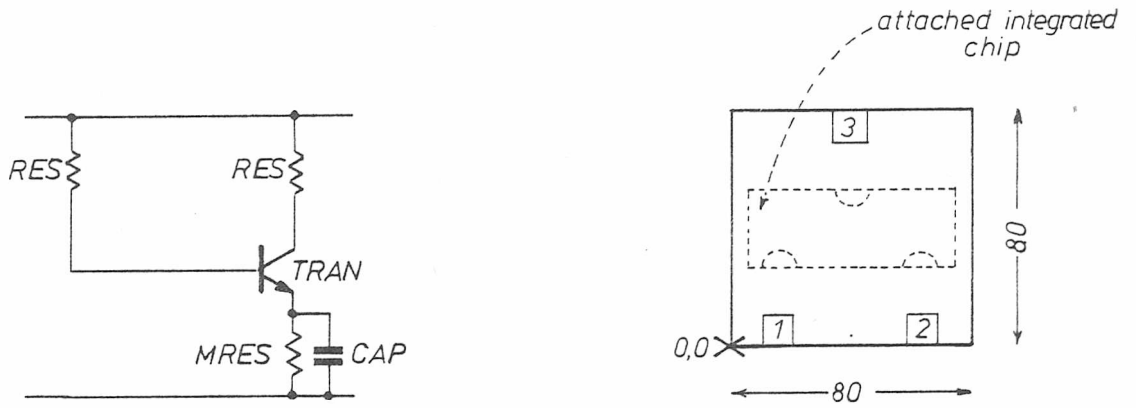
The resistor design programs, described in Chapter 3, generate data files which can be manipulated interactively using the GAELIC graphics suite (Ref 6). The same data can alternatively be merged with geometric and topological information supplied by the designer and used to create a numerical model of the circuit. Having constructed such a model, a series of algorithms can then be applied with a view to generating a layout automatically. This chapter describes the formal input language which has been adopted as a means of entering the necessary circuit information. The structure of the model is explained in detail.

The data formats described in the following sections are based upon the work carried out by Rose (Ref 19) in the design of single-sided printed circuit boards.

Having designed the necessary meandering resistors, the first step in producing a layout is to prepare the circuit data in a suitable format for input to the computer. This consists of two distinct stages and is best illustrated with reference to the simple circuit described in Figures 4.1 and 4.4.

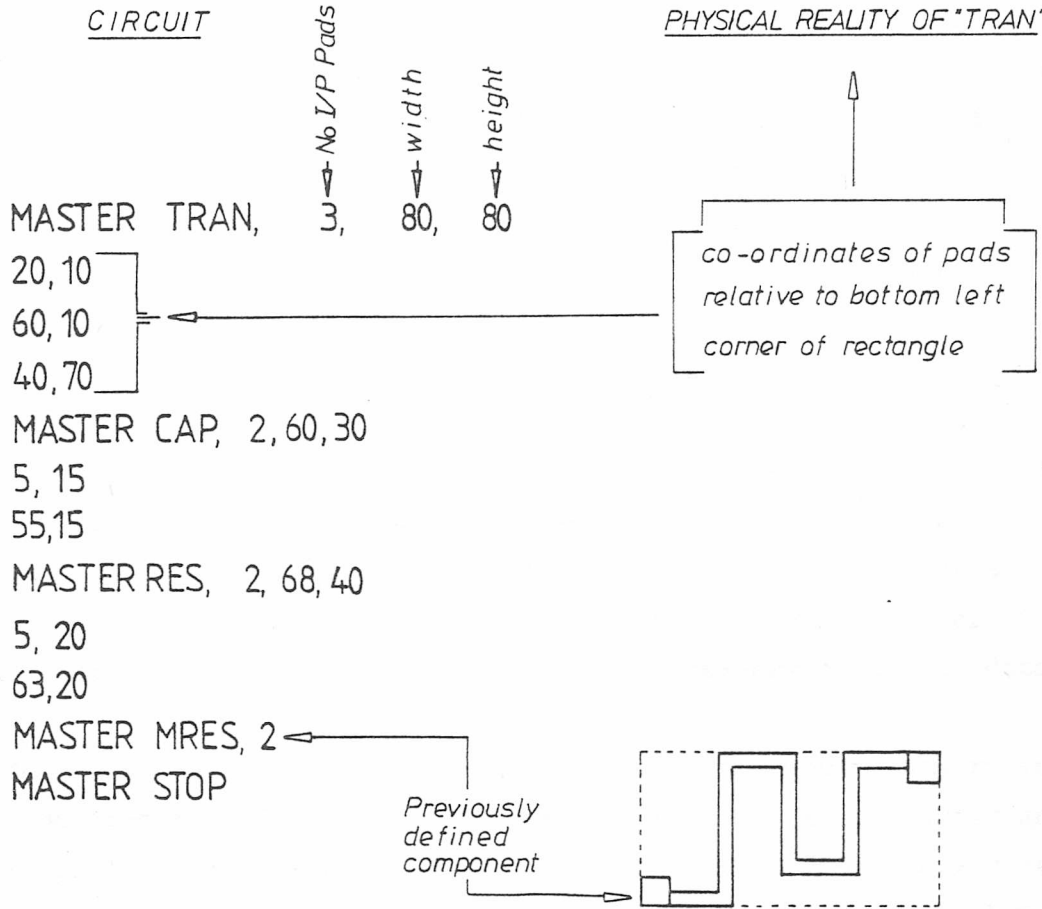
#### 4.2 Master Component Libraries

The initial stage is a set of data describing the geometric shape and dimensions of the circuit elements. Since several components may be physically identical, these definitions are referred to as "MASTER" shapes and can be applied to more than one device. The descriptions are stored in "MASTER COMPONENT" blocks (groups of data) which are linked together to form a list, and each is given a unique name. For example, the geometries representing  $\frac{1}{4}$  watt and  $\frac{1}{2}$  watt resistors could be called RES1 and RES2 respectively.



CIRCUIT

PHYSICAL REALITY OF "TRAN"



PHYSICAL REALITY OF "MRES"

Figure 4.1 Master Component Descriptions

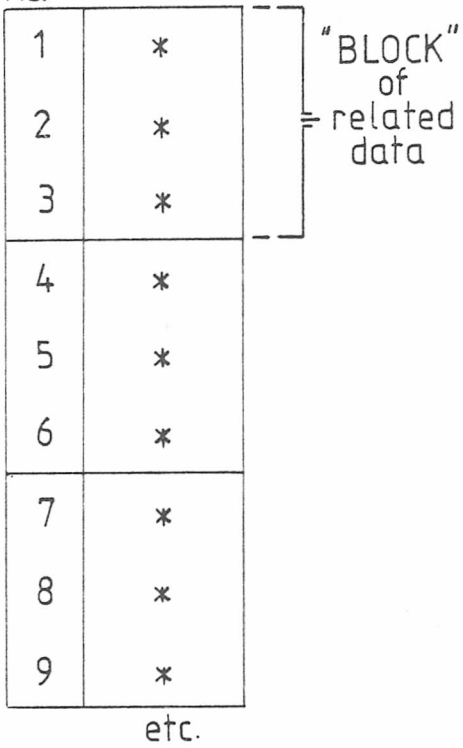
The geometry of a component is defined by a bounding rectangle which allows space for the component and several connection pads. If the rectangle represents an attached device then the pads will be used to connect the chip into the circuit using a wire-bonding technique. In the case of an in-slice component such as a meandering resistor they simply act as metallised terminations.

Each Master Block holds geometric information concerning the component it represents. The bounding rectangle dimensions are stored, together with the number of input pads and their individual co-ordinates - the origin of which is taken as the bottom left hand corner of the rectangle. It is important to retain the pad positions in a consistent cyclic direction and a convention has been adopted to use a counter-clockwise list.

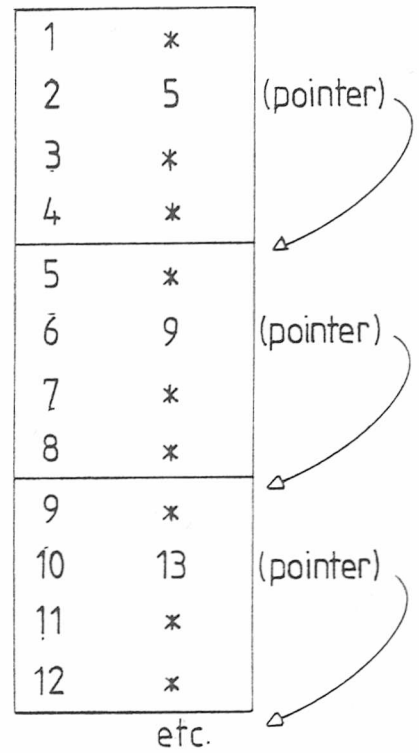
Component Master "TRAN" in Figure 4.1, for example, has been defined as a square of side 80 units with three input pads. This particular description assumes that the component is a solid state chip which will eventually be attached to the substrate. Thus the bounding rectangle with its associated pads is all that need be specified. In contrast to this, component "MRES" represents a previously defined meandering resistor. The computer assumes this when no rectangle height and width are given. It then refers to the data files produced by the resistor design programs to obtain the necessary geometric data.

The component descriptions must be stored in computer memory in an efficient manner. A convenient method is to use a long array and simply to list the descriptions one after the other as shown in Figure 4.2(a). This is very economical with regard to storage space, but, if we assume an irregular data block size, then each and every location must be examined in further processing. A better system is to include an extra item of data in each block of related elements to hold the "address" of the next block in sequence (refer to Figure 4.2(b)). Such an addition is called a "Data Pointer" and can be used to jump past irrelevant data with a significant saving in processing time. This can be represented schematically by omitting the location numbers and drawing each isolated group of data elements as a separate block. This is demonstrated in Figure 4.2(c). It is, of course, perfectly possible to include more than one pointer in each block, and a number of interlinking "Lists" and "Rings" can be produced to speed up access time.

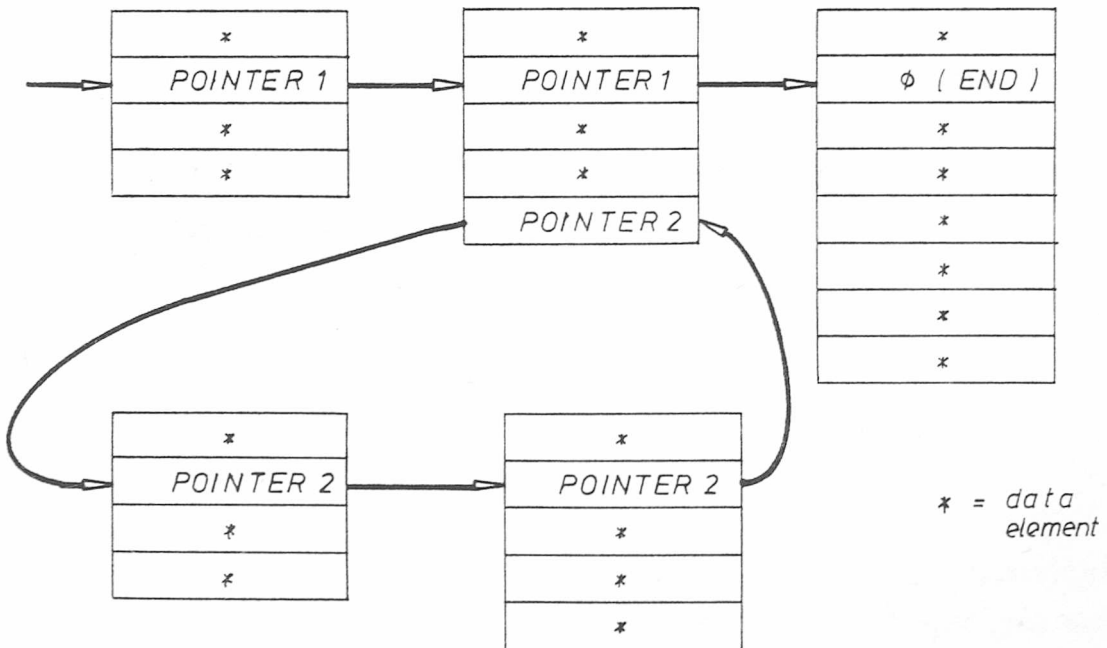
element  
No.



(a) Simple array



(b) List data structure



(c) Schematic diagram

A "List" is defined as a string of blocks which terminates in a zero or negative data pointer and is demonstrated as "POINTER1" in the diagram. The alternative is a data "Ring" where the pointers are arranged to form a never-ending loop. This is shown as the pointer group "POINTER2" in the diagram.

Figure 4.3 illustrates the data format employed to describe a Master Component geometry. Each "Master Block" has exactly  $6 + n$  entries where "n" is the number of terminal pads incorporated within the bounding rectangle.

The block stores:

- (a) The description name
- (b) A pointer to the next Master Block in sequence. This forms a simple data list as demonstrated in Figure 4.2(c).
- (c) The number of terminal pads associated with the component geometry.
- (d) The Width and Height of the bounding rectangle.
- (e) A set of pad co-ordinates relative to the bottom left hand corner of the bounding rectangle (Both the X and Y values are coded into the same data word using the formula  $10000X + Y$ . This technique reduces the size of the data files considerably).
- (f) A pointer to one of a separate group of data blocks containing information regarding the shape of in-slice components. This pointer is zero-valued in the case of attached devices.

The example given in Figure 4.3 refers to the component Master "TRAN" described in Figure 4.1.

#### 4.3 Preparation Of The Circuit Topology

Having established a list of available component geometries, the circuit topology must now be specified. Each electrical node is first labelled with a unique positive integer, referred to as a "Node Number". Each is associated with a separate "Node Block" in the data structure. The connections to each component are defined by listing

Location			
47	TRAN	Pointer from last master block	
48	56	Pointer to next master block	
49	3	Number of terminal pads	
50	80	Width of bounding rectangle	
51	80	Height of bounding rectangle	
52	200010	Co-ord. of pad 1 ( 20,10 )	
53	600010	Co-ord. of pad 2 ( 60,10 )	
54	400070	Co-ord. of pad 3 ( 40,70 )	
55	0	Pointer to shape description if applicable	

56	RES	Next master block
----	-----	-------------------




Figure 4.3 MASTER BLOCK

the Node Numbers to which it is attached. Again, these must be listed in an anti-clockwise direction round the device. An additional constraint is that two-pin components with a marked pin of polarity such as diodes or electrolytic capacitors are listed with the marked pin first.

To code the data from a circuit diagram, then, each component is described by a unique name and a list of "Node Numbers". Components of the same type are listed consecutively in a group and each group is preceded by the name of its Master Component description.

Two additional dummy Master names are used - "STOP" and "EDGE". The former simply terminates the circuit data and the latter signifies the start of a list of circuit nodes, which represent the inputs and outputs to the circuit. These nodes, (specified once more in a counter-clockwise direction) are the only nodes to eventually have physical reality as "Edge Pads".

Figure 4.4 demonstrates the topological data needed to describe the simple circuit shown. In this case, five Node Numbers are needed - four of which will be Edge Pads in the completed layout.

#### 4.4 Representing A Two-Pad Component In The Data Structure

Two-pad and multi-pad components are represented in different ways in the data structure. The former case is the simplest and can be explained with reference to Figure 4.5.

Three new types of data block are needed to describe the electrical circuit. The first is called a "Node Block" and represents a circuit node. Edge Pads, which are fundamentally Edge Nodes, need special Node Blocks to store X and Y co-ordinates. The other nodes, however, have no physical significance on the board.

"Branch Blocks" are then created as topographical representations of circuit components. Each Branch Block points to two Node Blocks representing the Node Numbers assigned to the component's input terminals.

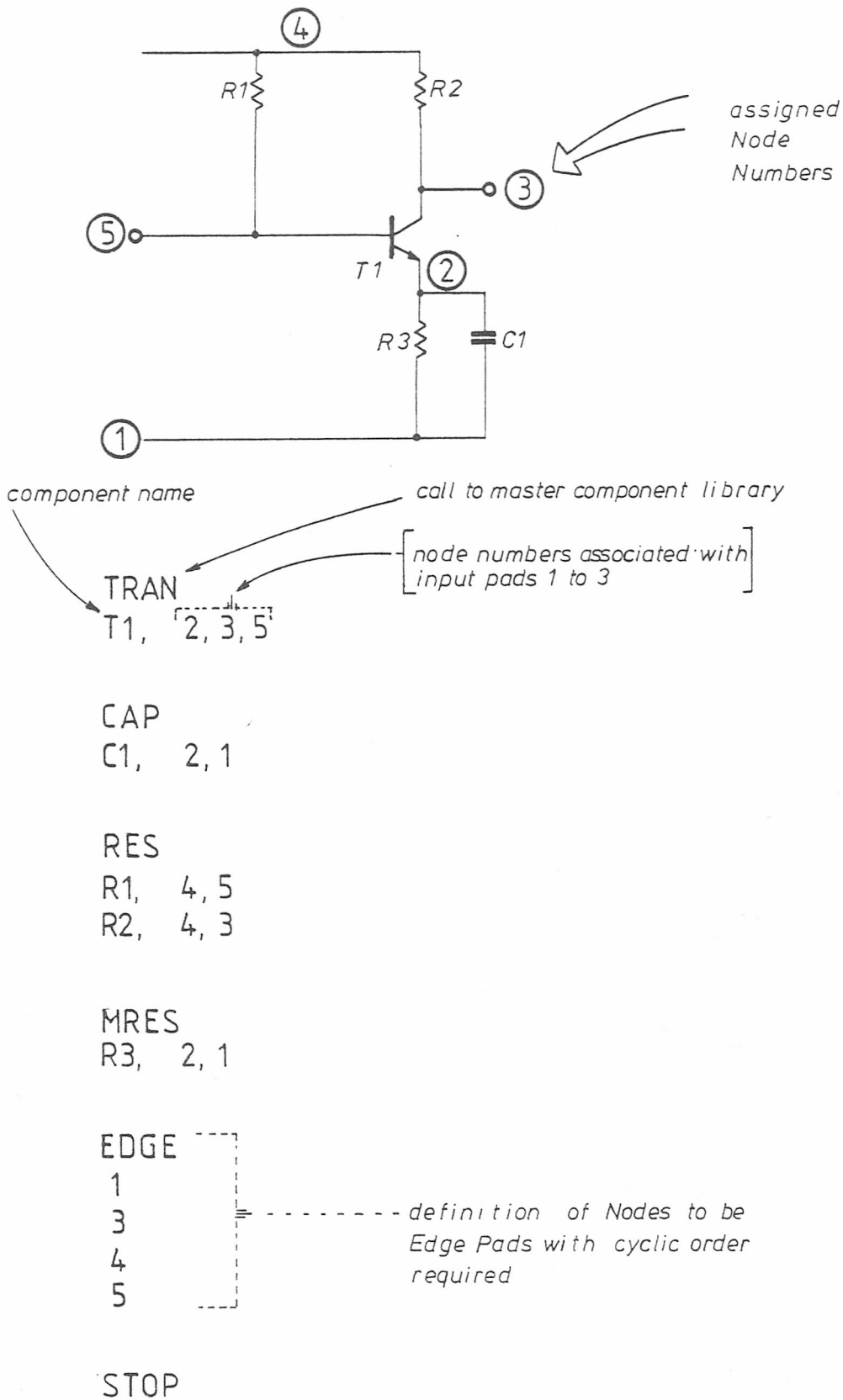


Figure 4-4 Circuit Description



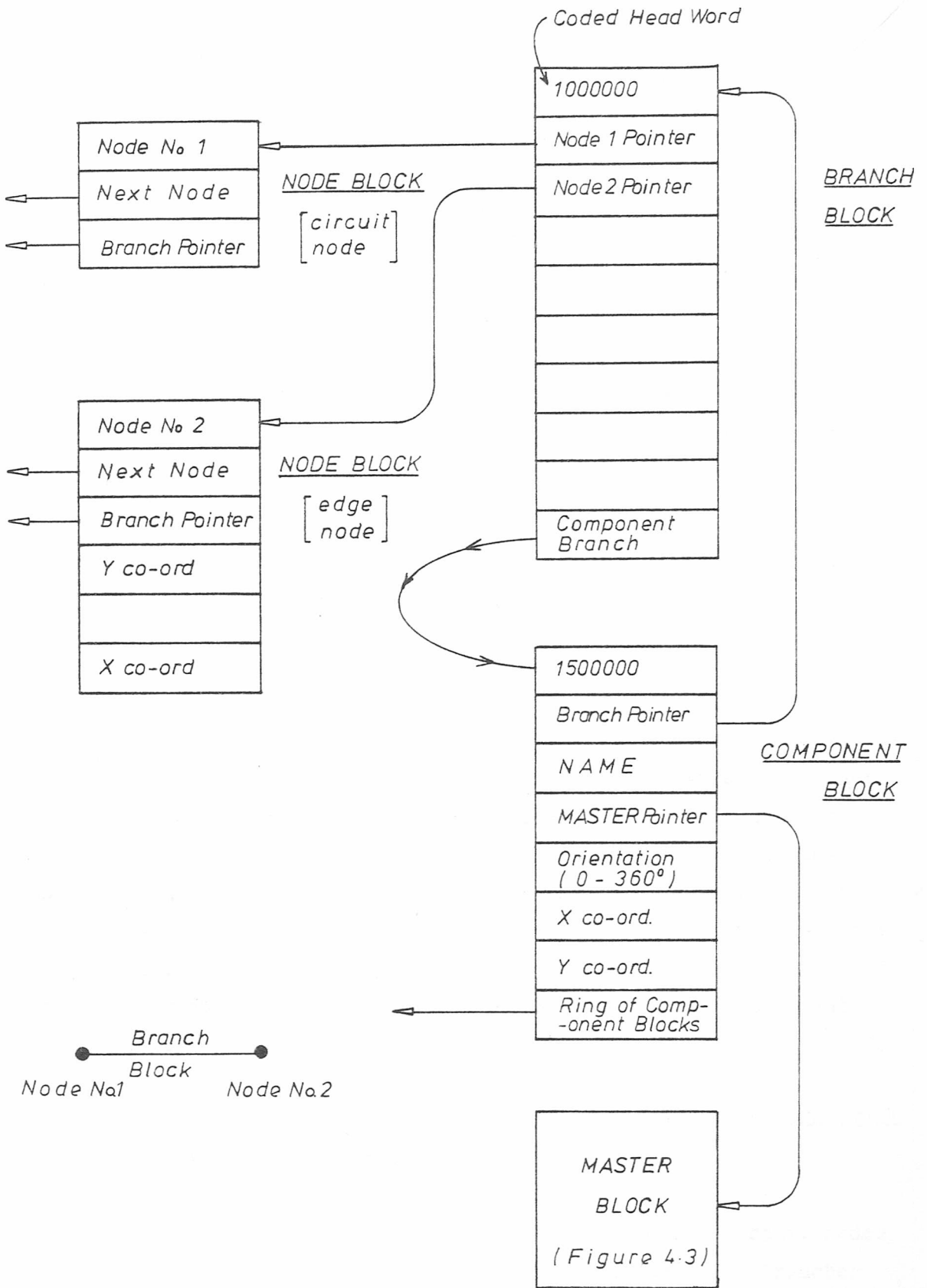


Figure 4.5 Data Storage of a Two Pad Component

The next stage in the design process is the use of a "Planarity Algorithm". This involves creating a set of "Regions" which could be placed side by side on the substrate. A Region is formed by linking Component Branch Blocks together in a ring. Since the Regions are adjacent to each other, every Branch will appear in two Regions.

The third block type is the "Component Block", which has storage space for co-ordinate and rotational information. This block is labelled with the component's name, and is linked to all the other Component Blocks by a series of pointers. The Block will store the component's X and Y position on the board, together with the angle through which its Master description has been rotated. A direct pointer to the appropriate Master Block allows the layout to be drawn out simply by accessing each Component Block in turn.

Figure 4.5 illustrates how the three types of blocks are linked, using data pointers.

#### 4.5 Multi-Pad Components In The Data Structure

The Branches described in the last section represent individual two-pad components. When dealing with multi-pad devices, a slightly more sophisticated technique must be applied. Figure 4.6 illustrates how a three-terminal device can be described using several blocks. Each terminal of the component is first assigned a Node Block, but since these nodes are not true circuit nodes, they are called "Pseudo-Nodes" and are marked with a special code (PN1, PN2 and PN3 in the example). Corresponding "Pseudo-Branch" blocks are used to form a little sub-circuit representing the component (PB1, PB2 and PB3)

Only one Component Block is needed, of course, and is identical to that of a two-pad component.

A set of "Link Branches" are created to link the circuit nodes to the Pseudo-Nodes. These are very similar to ordinary Branches except that they are marked with a coded word, as in the case of the pseudo-blocks mentioned above. These are labelled L1, L2 and L3 in the diagram.

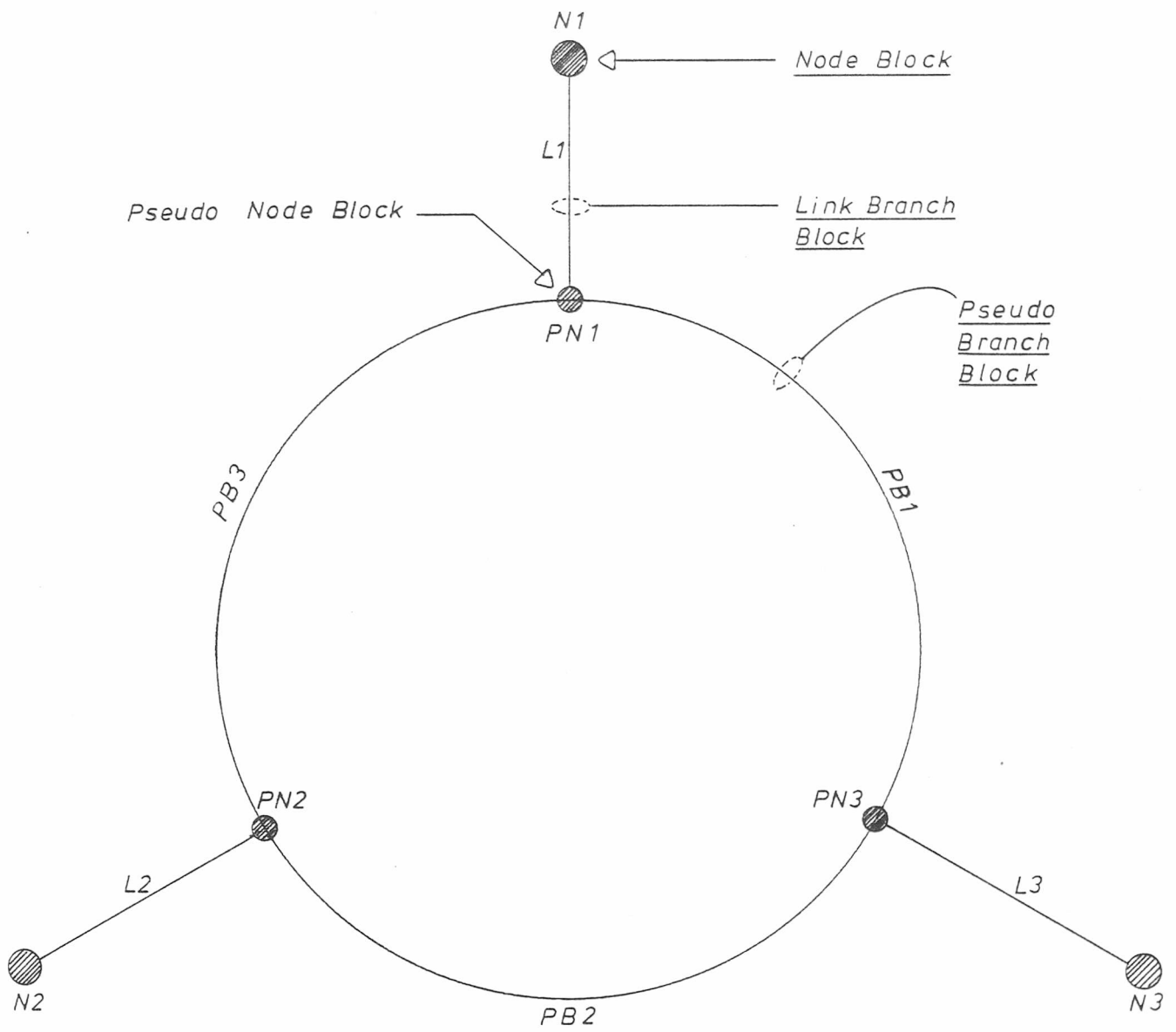


Figure 4.6 NODE BLOCKS AND BRANCH BLOCKS  
Associated with a multi-pad component

It can be seen that while a single Branch Block describes a two-pad component, we require  $3N$  blocks of various types to represent an  $N$ -pad component.

#### 4.6 Conductor Paths

When the components have been placed on the substrate, inter-connection paths are created to form the desired circuit. The former process merely involves entering  $X$  and  $Y$  co-ordinates into existing Component Blocks, but a new set of blocks are needed to store the conductor patterns generated in the latter.

Figure 4.7 shows how a connection is represented using a "Connection Block". It is assumed that the width of all the conductor paths is constant, so it is only necessary to store corner and termination co-ordinates. For processing purposes it is convenient to have the start and end points together at the start of the block. This avoids an un-necessary search through all the corner points in later processing. Since each connection represents a circuit node and several may represent the same node, it is also necessary to store this value in the Connection Block. This enables the Routing Algorithm to cross or intersect connections of identical circuit potentials.

Another feature of this block type, is that it can be shortened, lengthened or re-used very easily using a system of pointers. When reading the block, a negative number denotes a jump to another data location, while a positive signifies a legitimate piece of information. This arrangement means that any connection can be altered as the need arises, by jumping past irrelevant co-ordinates or adding small blocks of additional data. Should the whole connection become deleted then the data space is re-used when the next connection is formed. The system effectively reduces the size of the data structure needed for any particular circuit by re-utilising redundant memory locations.

#### 4.7 Representing Planarity In The Data Structure

The Planarity Algorithm described in Chapter 5 forms a number of circuit loops or "Regions". Figure 4.8 demonstrates a simple circuit with its associated Regions.

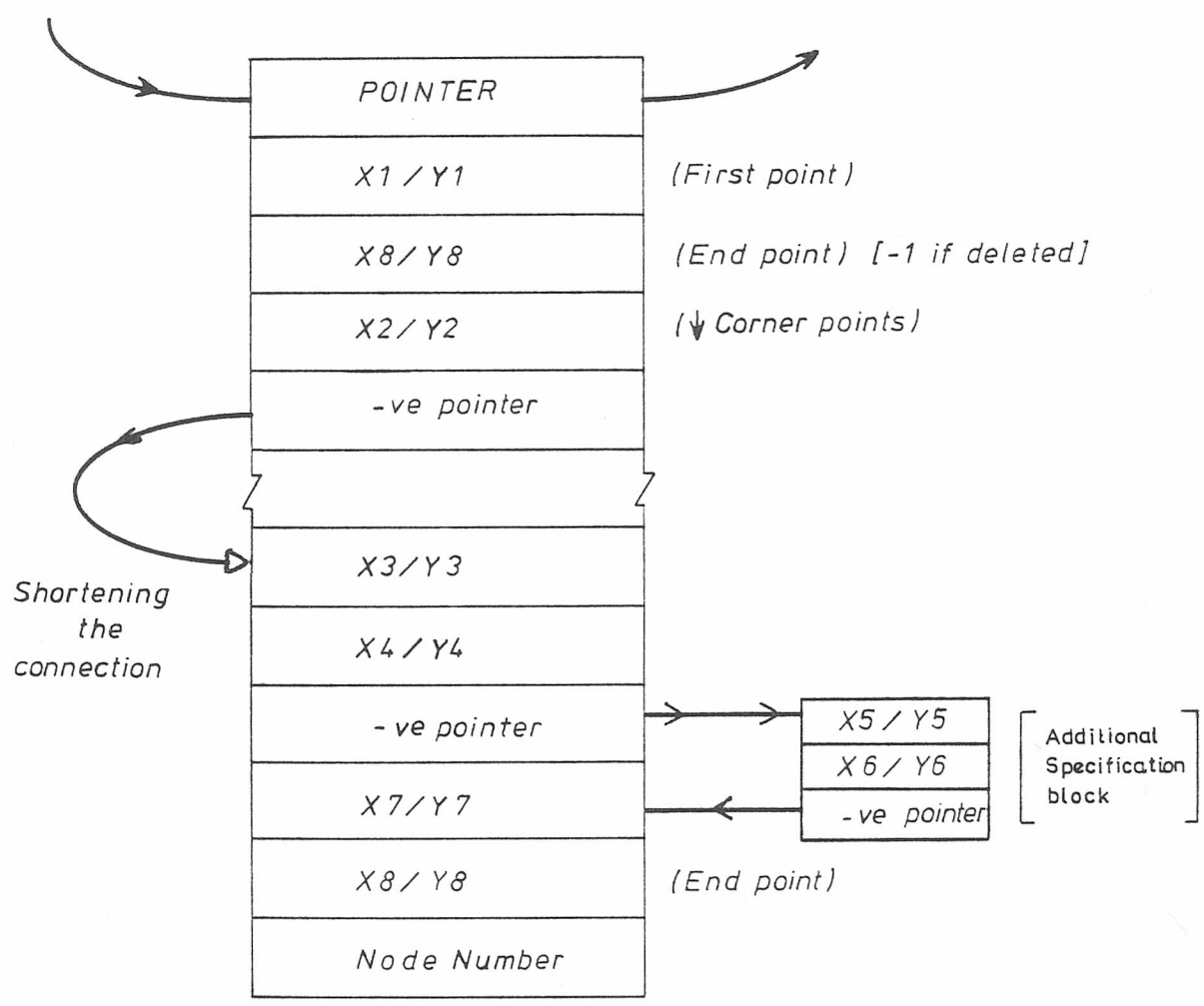
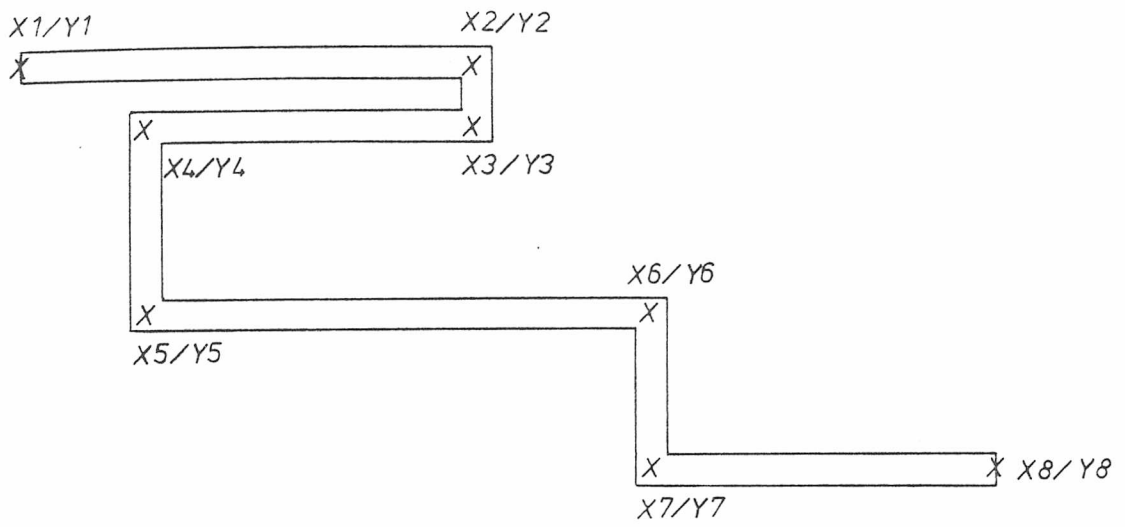


Figure 4.7 THE CONNECTION BLOCK

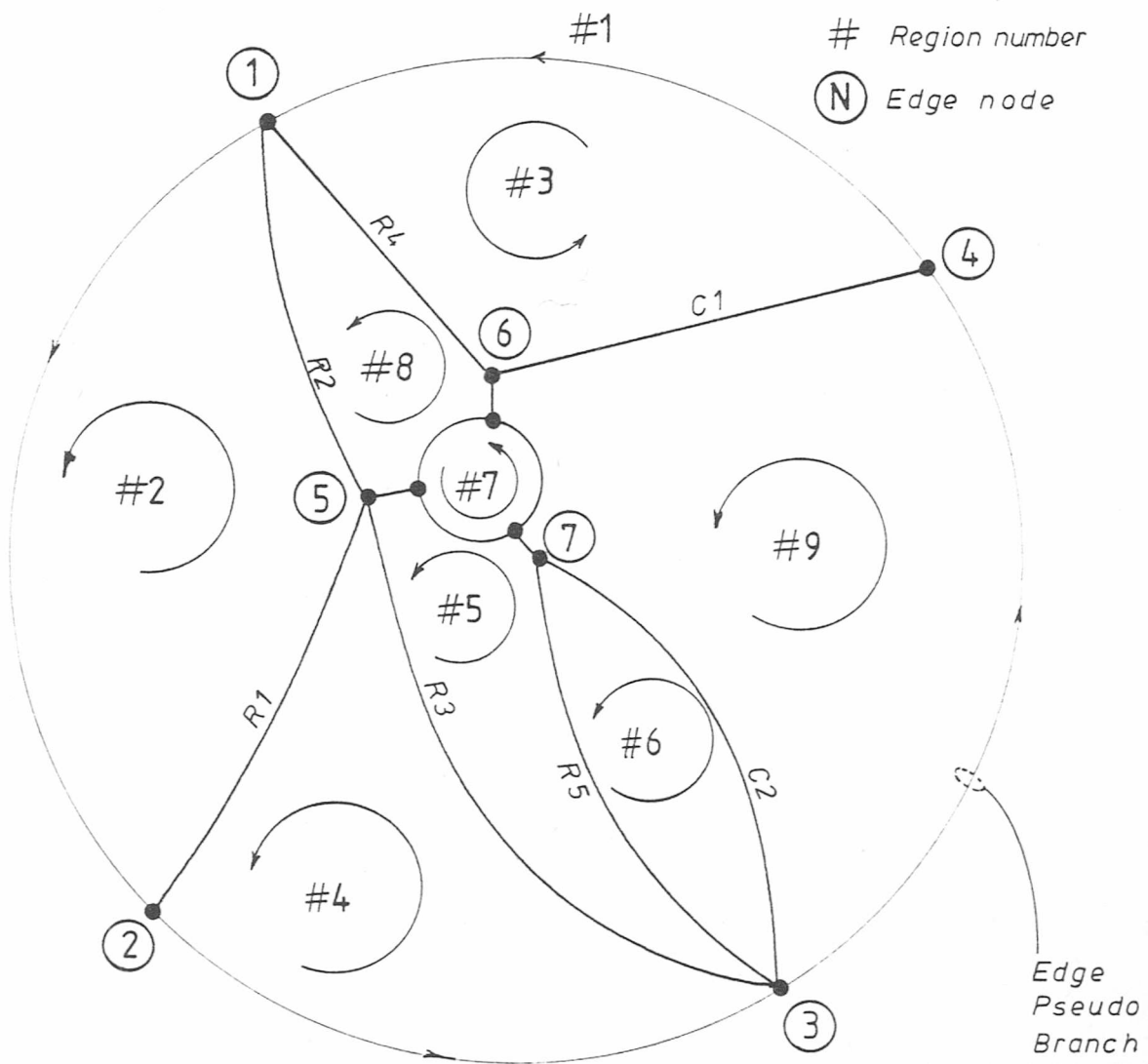
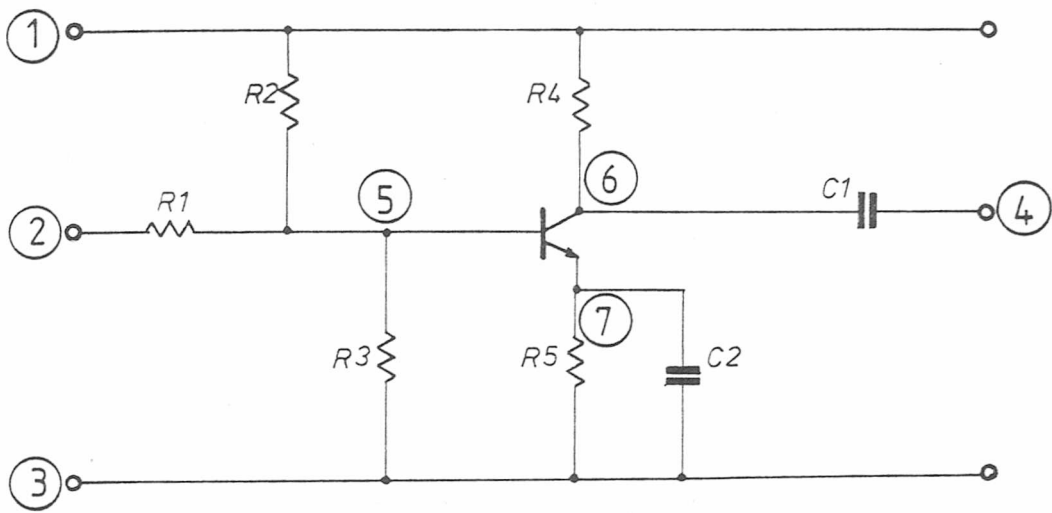


Figure 4.8 Planar Representation of Circuit

Each Region is simply a series of Branch and Pseudo-Branch Blocks linked via pointers to form a ring. The first Region links a set of "Edge Pseudo-Branches" together, to simulate the available board area. This Region makes it possible to locate the Edge Pads quickly when redrawing the layout.

It can be seen that every Branch Block appears in exactly two Regions, so each is assigned a pair of "Tie Blocks". The Tie Blocks are linked together and to the Branch Block in question. Figure 4.9 shows how a Region is formed by linking a series of "Tie Blocks" to a special "Region" block. The latter is labelled with a unique number and added to a data ring. This allows each Region to be examined sequentially in subsequent processing.

In Figure 4.9 (which represents Region 2 in Figure 4.8), components R1 and R2 have been linked with an Edge Pseudo-Branch. The Region Block is both the start and end point of the loop.

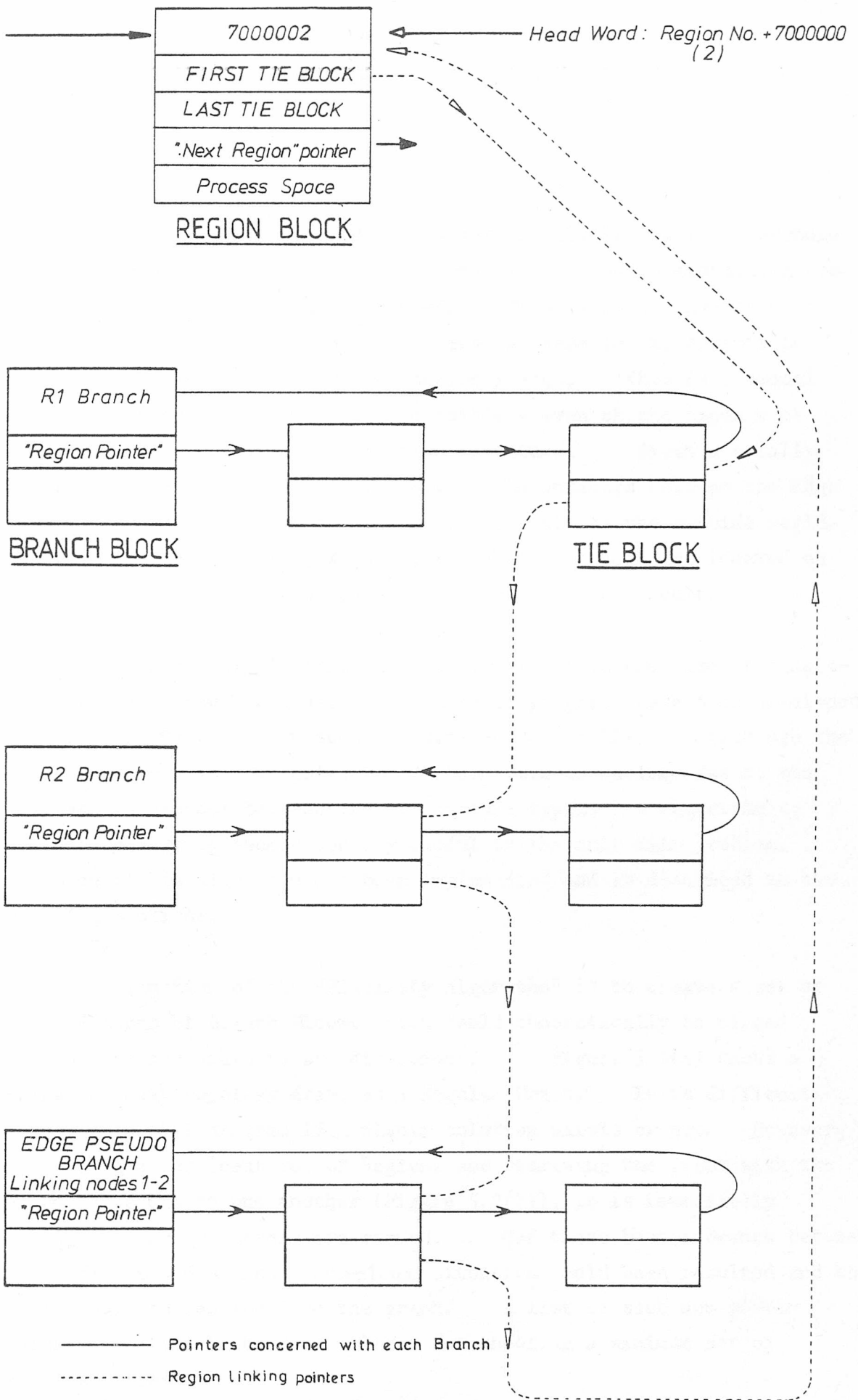


Figure 4.9 Typical REGION Loop



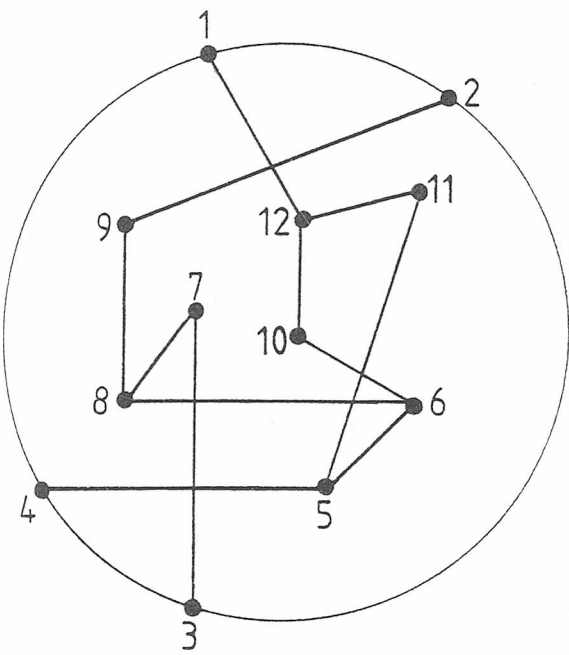
Chapter 5  
The Planarity Algorithm

5.1 Introduction

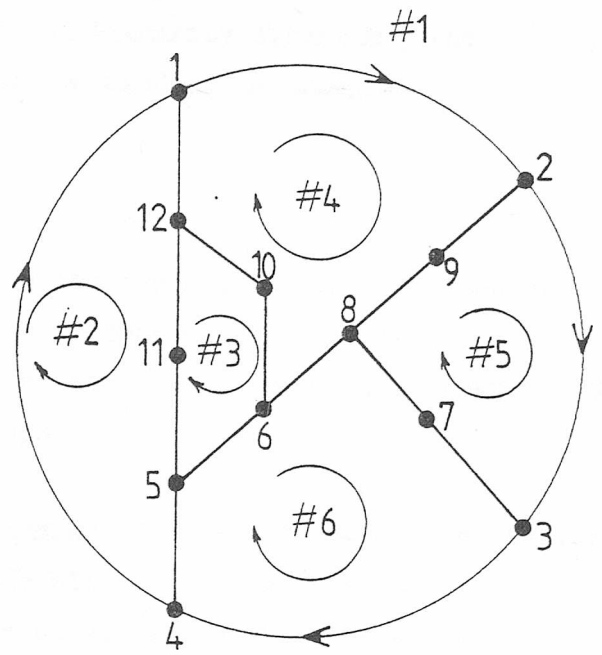
The main objective in designing a thin film layout is to arrange the components and their interconnections with a view to minimising the number of conductor crossings required. Crossovers are achieved in practice by bonding metal wires on to special pads in the circuit in order to jump over one or more connection tracks. This is a manual process and hence to be avoided if possible - even at the expense of a longer, more complex interconnection structure. Given a totally planar representation, wires still have to be attached between the Edge Pads and package terminals to connect the circuit to the outside world. This is a much easier task, however, since all the pads are located on the board perimeter - not hidden in the midst of the circuit.

The concept of planarity is also important in the case of single-sided printed circuit boards. A suite of programs have been developed by Rose (Ref 19) to design such circuits automatically. Although the bulk of his work is not applicable to thin film technology due to the inherent differences between the two circuit types, his algorithm to determine planarity seemed equally useful to the thin film problem. A version of his algorithm has been implemented and is described in the following sections.

The function of the "Planarity Algorithm" is to create a set of "Region" loops of Branch Blocks which could theoretically be placed side by side and drawn in two dimensions. Figure 5.1(a) shows a typical circuit topology drawn as a Regular Graph. It is difficult to know from this diagram if a planar solution exists or not. However, by choosing a pertinent set of Regions and redrawing the graph with the Regions adjacent to one another (Figure 5.2(b)), it is immediately obvious that no crossovers are needed. Had there been a branch between, say, nodes 10 and 3 then a non-planar situation would have resulted and the branch would be removed from the graph. A list of such non-planar branches can then be used as a basis for choosing a minimum set of crossover sites.



(a) Topological Representation



(b) Identical representation after Regions have been chosen

Regions defined as :-

- #1) 4 → 1 → 2 → 3 → 4
- #2) 4 → 1 → 12 → 11 → 5 → 4
- #3) 5 → 11 → 12 → 10 → 6 → 5
- #4) 6 → 10 → 12 → 1 → 2 → 9 → 8 → 6
- #5) 8 → 9 → 2 → 3 → 7 → 8
- #6) 4 → 5 → 6 → 8 → 7 → 3 → 4

Figure 5.1 The use of the Planarity Algorithm

To explain the mechanics of the Planarity Algorithm, the formation of graph (b) from (a) will be used as an example.

## 5.2 Tree Search Procedure

An important sub-algorithm in the process is the "Tree Search", which is used to find the shortest path between two circuit nodes. Refer to Figure 5.2 which illustrates the search for a path between nodes "A" and "L" in a simple Regular Graph.

The search is split into a number of "Levels", each representing the traversal of a Branch in the circuit. It is obvious that the fewer levels needed to complete the path, the shorter the path. "A", in this case, is referred to as the "Start" node and "L" as the "Target" node. There is only one possible route from the Start node and this leads to node "C", Level 2. Node "C", however, is connected to two additional nodes namely "B" and "D". Since it is not obvious which path will ultimately lead to the Target, both routes must be investigated simultaneously.

By Level 4, three possible destinations could have been reached, and this increases to five in Level 5. Nodes "F", "G" and "A" can be disregarded, however, as they have already been encountered in the search. The Target is finally reached in Level 6, giving the shortest path to be A - C - B - G - H - L.

This system will invariably find the shortest possible route between any two nodes.

## 5.3 Formation of Regions

The first step in constructing a Planar Graph is to define the allowable board perimeter, so an initial Region is constructed linking all the Edge Pseudo-Branched. A convention has been adopted to specify Regions in a clockwise direction, so the first Region in the example of Figure 5.1 would be defined as 4 - 1 - 2 - 3 - 4. This is also known as the "Free Region", since it indicates the board space available for placement. Consecutive nodes on the perimeter of the Free Region are selected at random as "Start" and "Target" nodes in a Tree Search. In later stages the criterion for selection is that the node must have at

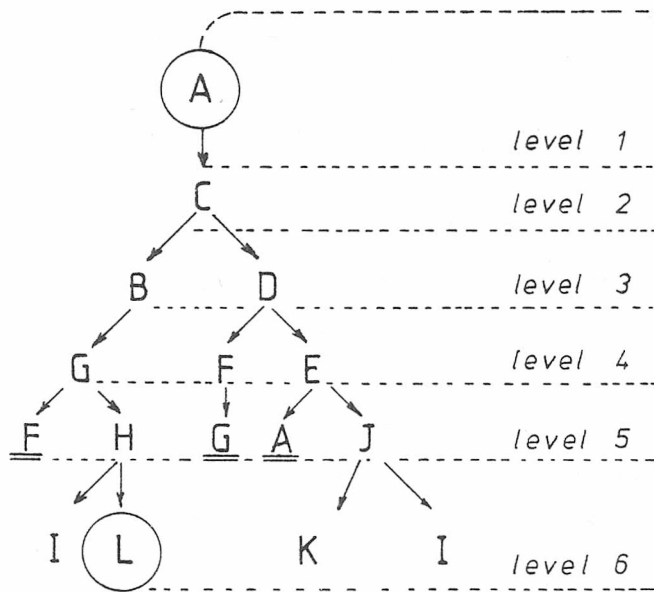
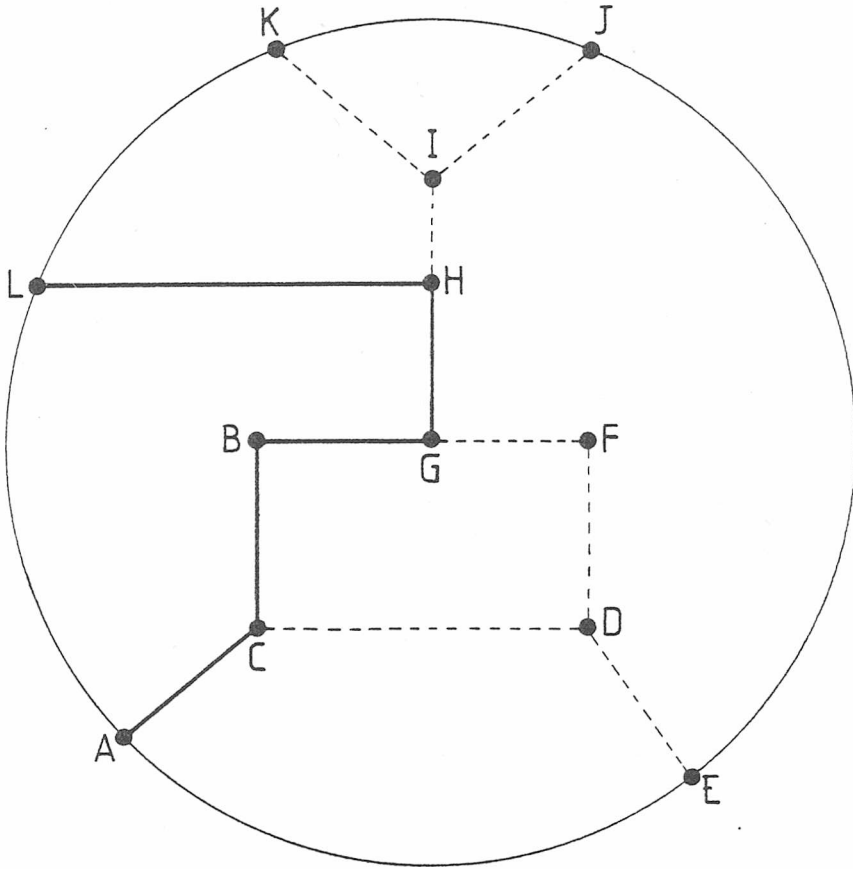


Figure 5.2 Tree search from nodes A to L

least one unused Branch connected to it (otherwise no possible Region could start from it). The corresponding Target node is chosen as the next in a clockwise (or, when this fails, anti-clockwise) direction round the Free Region with a similar "Free Branch".

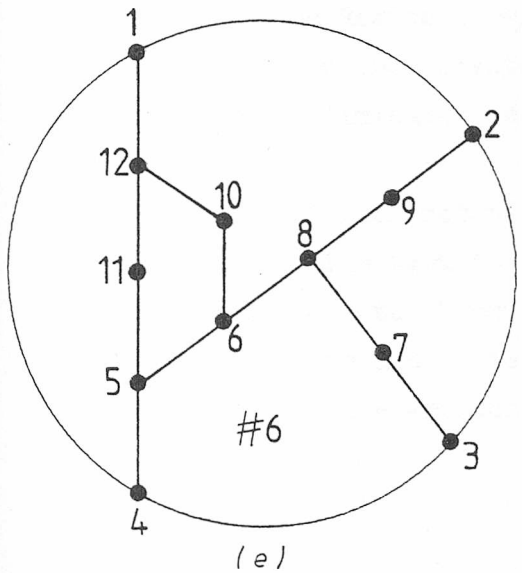
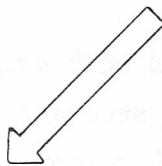
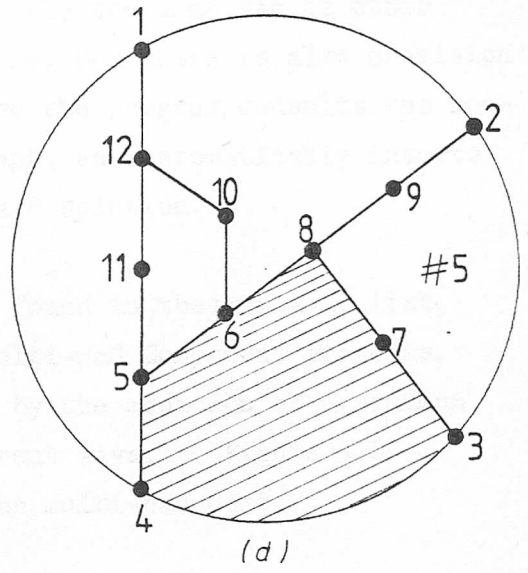
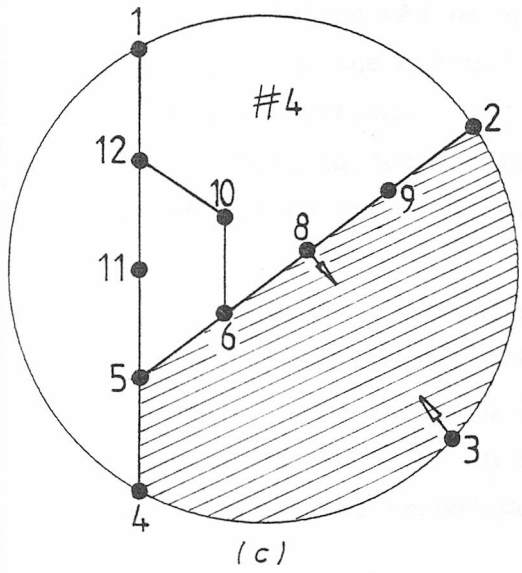
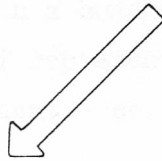
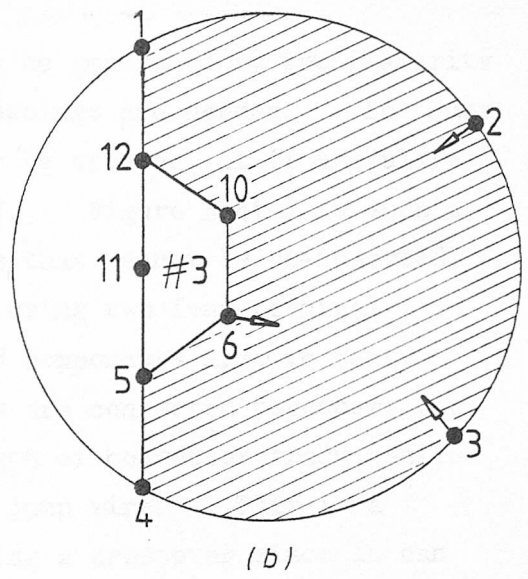
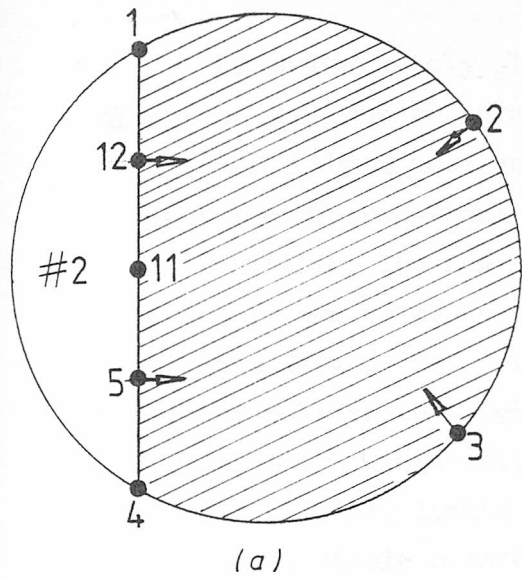
Figure 5.3 illustrates the process using the example given in Figure 5.1. Nodes 4 and 1 have been selected as Start and Target in (a), and the shortest path found to be 4 - 5 - 11 - 12 - 1. It is now possible to form a second Region (#2) defined as 4 - 1 - 12 - 11 - 5 - 4 and the Free Region is re-defined to be 4 - 5 - 11 - 12 - 1 - 2 - 3 - 4 (shown as a shaded area in diagram (a)). The Region which has just been formed contains no nodes with Free Branches other than those it has in common with the Free Region itself, so there is no doubt that it is a planar subset of the total graph.

Since node 4 has no remaining Free Branches, the Start node is next chosen as node 5 which is the next "Free Node" in a clockwise direction round the Free Region. Node 12 is selected as the Target for similar reasons and the process continues.

Eventually there are no Free Branches left in the Free Region, as in (d), and the Free Region itself becomes the final Region.

#### 5.4 Non-Planar Graphs

A completely planar set of Regions have been developed in this example from the graph given in Figure 5.1(a), but this is not always the case. Should two branches be found to conflict with one another (i.e. prevent the formation of a planar Region), then one is immediately removed from the graph and stored in a data list for future reference. At the end of the algorithm this set may contain several branches which could be re-inserted without loss of planarity. (A branch may have been added to the list unnecessarily if all the branches it conflicts with have themselves been removed). This is not a serious problem since a further algorithm is used to replace legitimate branches immediately after the Planar Graph has been formed.



"Free Region"

#

Newly formed Region



Free Branch

Figure 5-3 The Planarity Algorithm

## 5.5 Conversion Of Non-Planar to Planar Graphs

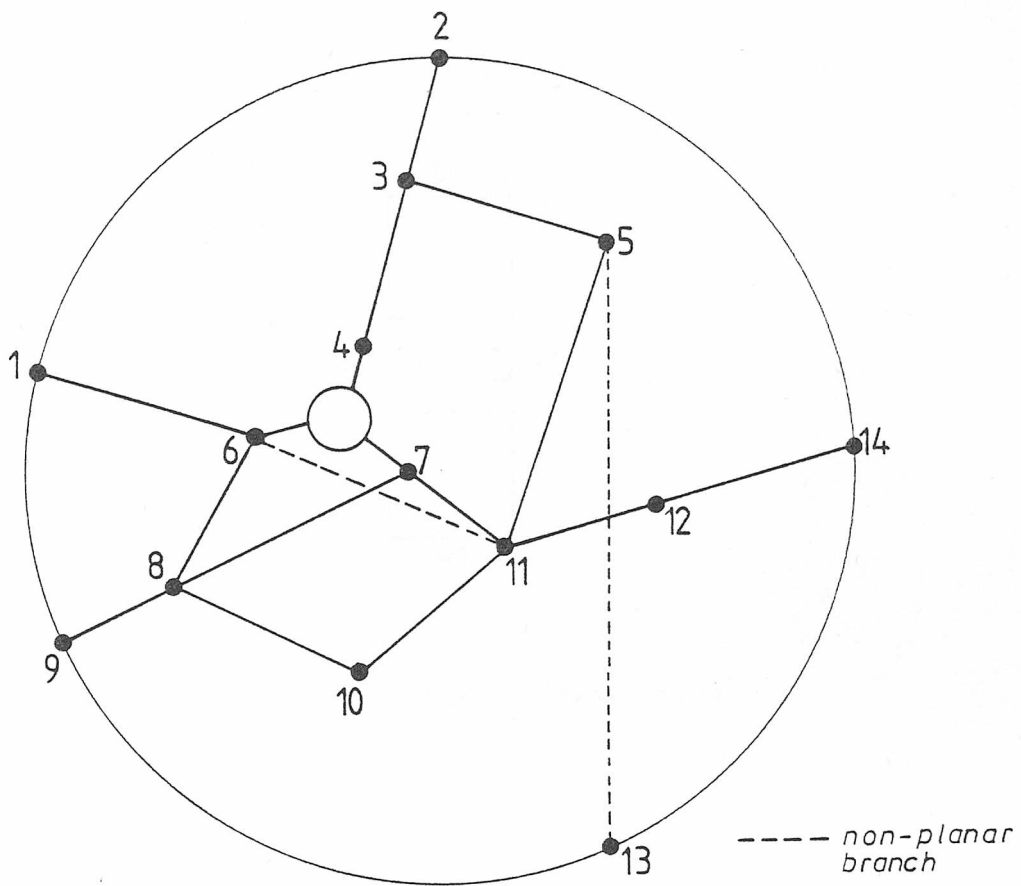
If no completely planar graph can be generated by the Planarity Algorithm then one or more conductor crossings are needed. In order to determine which node connections must be crossed, it is useful to consult the "Pseudo-Planar Graph" itself. Figure 5.4(a) is such a graph exhibiting two non-planar branches that cannot be re-inserted. This can be converted to a Planar Graph using two four-terminal components C1 and C2. These multi-pad components are, in fact, crossover devices in which opposite pads are connected together; one set of pads are linked with a short length of conductor track, while the other pair are linked with a bonded jump wire. This is a conveniently simple method of representing a crossover since it can be treated as a normal component - see Figure 5.5.

Crossings can be specified in exactly the same way as other components in the circuit description file, but there is also provision for auto-insertion. In the latter case the program consults the non-planar list, in conjunction with the Graph, and automatically inserts crossovers to create a completely "Planar" solution.

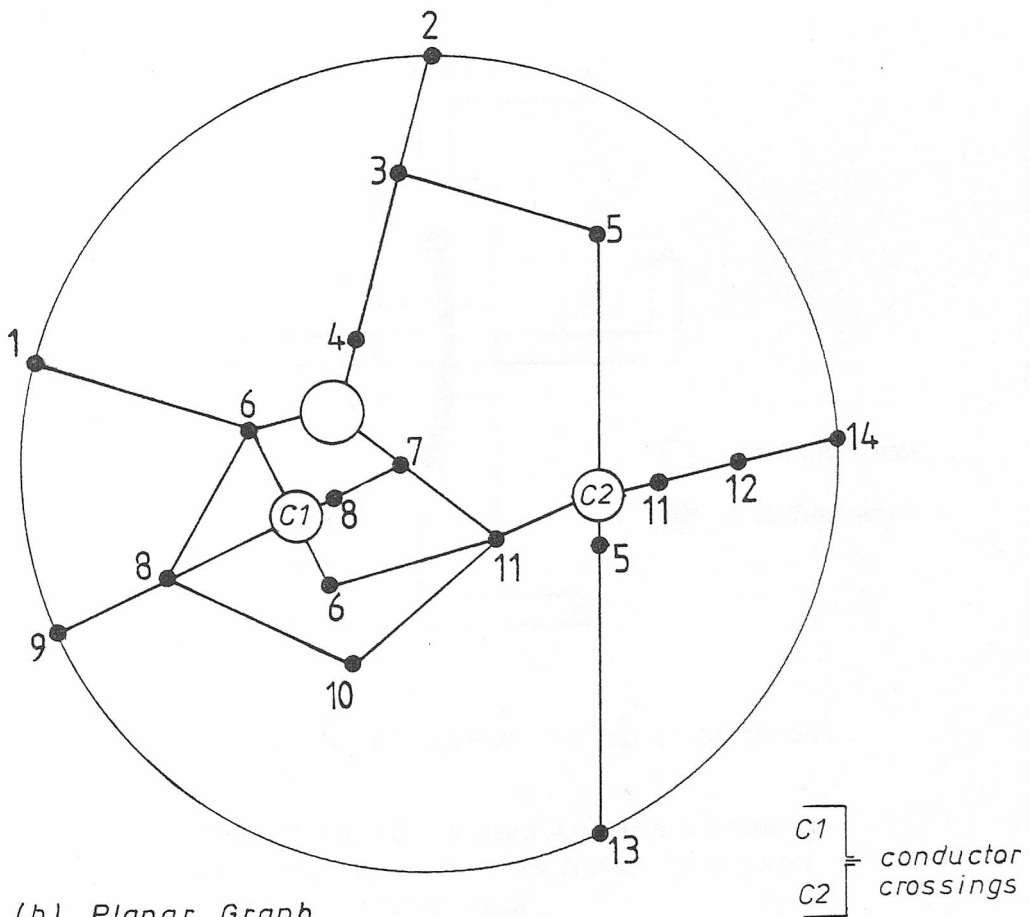
Only two types of element can be found in the non-planar list, namely two-pad Component Branches and multi-pad Component Branches. Each type is re-inserted into the Graph by the addition of a minimum set of crossovers but in slightly different ways. Figure 5.6 demonstrates the technique applied to the multi-pad case.

A Tree Search is used to find a path to the nearest Region containing a circuit node of the same number. In this example the path would be Region 5 => Region 3 => Region 2. A series of crossovers are then created and incorporated into the Graph which effectively eliminates the non-planarity caused by the branch.

A different method is employed to deal with non planar two-pad branches. The task is to find a pair of Regions containing the two different nodes to which the component is attached. There may be many such pairs and it is necessary to examine all of them to determine the shortest inter-regional path.



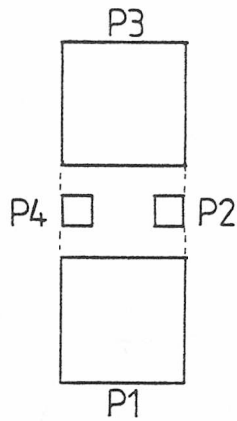
(a) Pseudo-Planar Graph



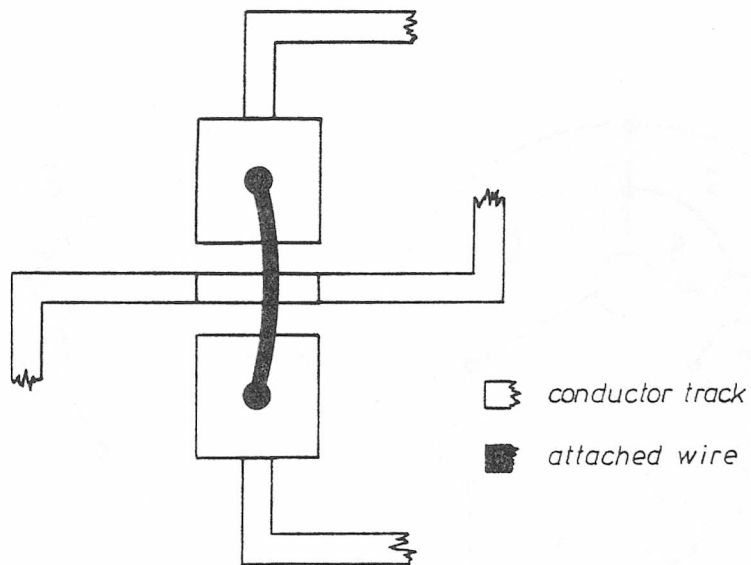
(b) Planar Graph

Figure 5-4 Pseudo-Planar Graph to Planar Graph





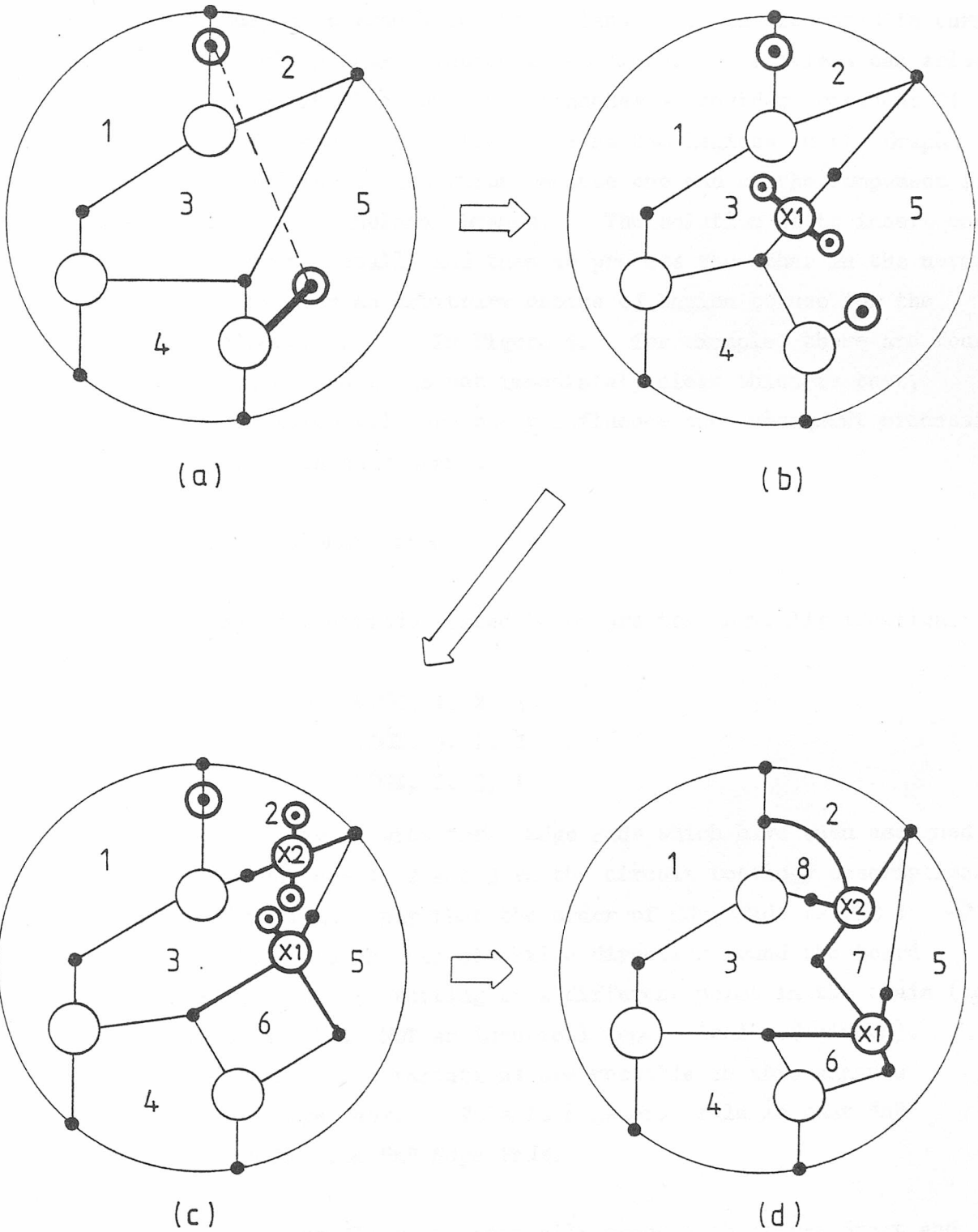
(a) representation in layout suite



(b) how it will appear in a fabricated circuit

*Pads P1 and P3 are used to attach a crossing wire. Pads P2 and P4 are linked by a short length of conductor track*

Figure 5.5 Crossover "component"



⊙ points representing the same node number

Figure 5.6 Auto-insertion of crossovers to deal with unplanar multi-pad components

The component is incorporated into the Graph immediately after the first crossover has been specified, and the process then continues as an unplanar multi-pad branch. This is demonstrated in Figure 5.7.

Each Component Branch in the unplanar list is processed in turn, until a completely planar solution is obtained. Problems can arise, however, with "Strings" of unplanar branches - consider component C1 in Figure 5.8 for example. There are no two Regions in the Graph which can be used for re-insertion because one end of the component is connected to another unplanar branch. The solution is to insert one of the string provisionally and then to process the other in the normal way. This involves an arbitrary choice of Region to use for the provisional placement. In Figure 5.8, for example, there are four possible solutions and it is not immediately clear which is best, although the decision will obviously influence the subsequent processing of the remaining unplanar branches.

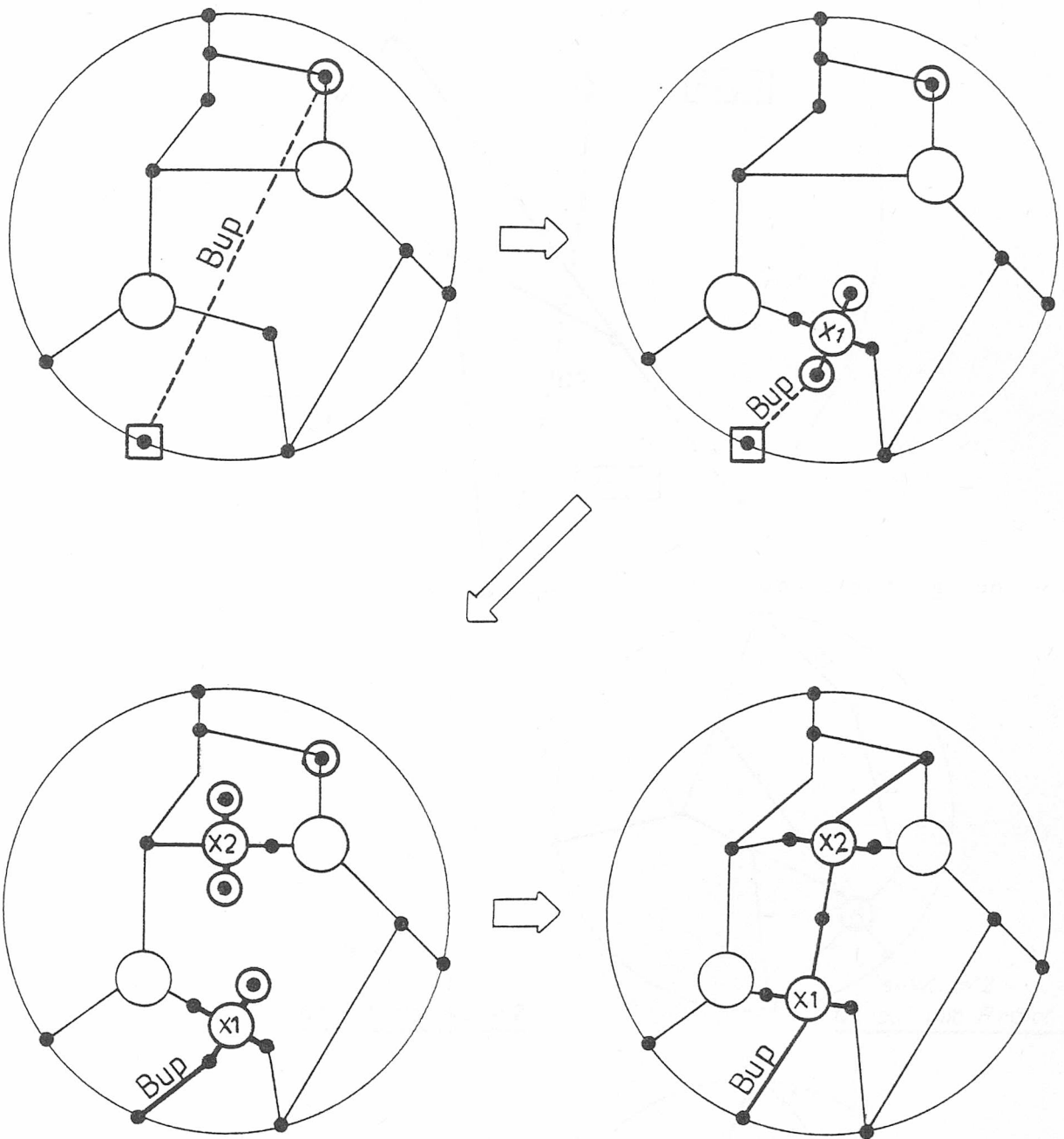
#### 5.6 Choice Of Planar Graph

The edge definitions listed below are topologically identical:

```
EDGE, 1, 2, 3
EDGE, 3, 1, 2
EDGE, 2, 3, 1
```

Each refers to a layout with three Edge Pads which have been assigned arbitrary node numbers 1, 2 and 3 in the circuit topology description. All three definitions imply that the order of Edge Pads is PAD 1 => PAD 2 => PAD 3 in an anti-clockwise direction round the board perimeter - each merely starting at a different point in the chain (note that "EDGE, 1, 3, 2" is NOT an identical topological definition). Exactly three equivalent variations are possible in this example involving three Edge Pads. This is a general rule in that "n" definitions result from "n" Edge Pads.

The Planarity Algorithm initially chooses the first Start and Target nodes arbitrarily as the first two Edge Nodes in the description file. This means that each definition, although topologically identical, may generate a slightly different Planar Graph. Assuming that the circuit is inherently planar (no crossovers are needed), then the only advantage in examining all possible graphs would be to obtain a solution



Unplanar 2pin component Bup nodes  $\square$  to  $\circ$

Figure 5.7 Auto-insertion of crossovers to deal with unplanar two-pad component

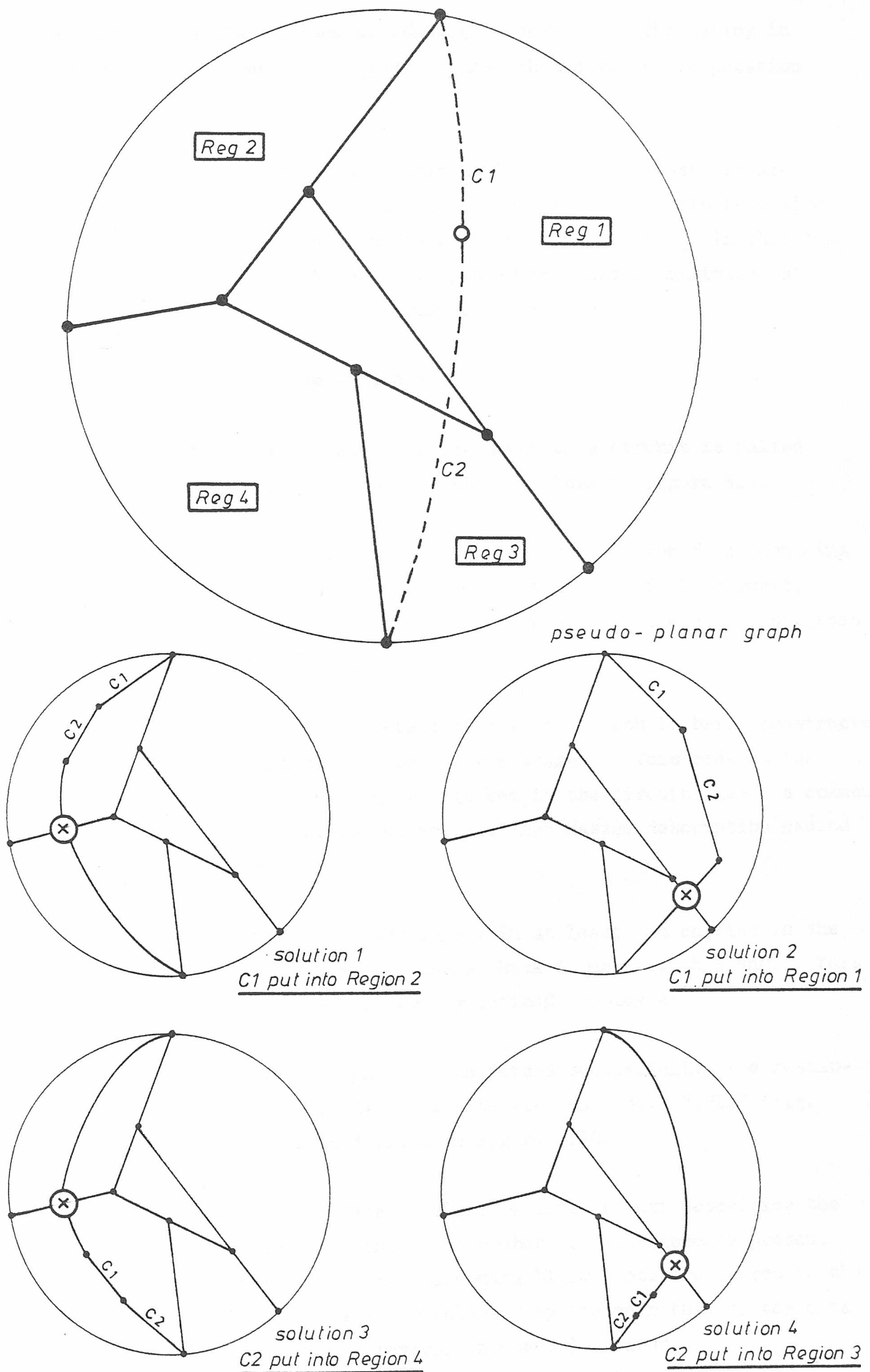


Figure 5.8 Unplanar STRING of two-pad components

which takes up the minimum of computer storage. The saving in memory, however, must be weighed against the increased computation involved.

If the circuit is inherently unplanar (some crossovers are needed), then the emphasis must be switched from data structure size to the number of crossings generated in each solution. In this case it is usually well worth the extra processing time to minimise the time taken to fabricate each circuit.

## 5.7 Planarity Program - PLANAR

The program to create a Planar Graph of a circuit is called PLANAR. An example of its operation is shown in Figure 5.9.

Two data files are required by the program - the first contains the Master Component descriptions and the topology of the circuit, the second holds resistor descriptions generated by the design programs described in Chapter 3.

If an abnormality is detected while the Graph is being constructed, the program halts and outputs an error message. This reduces the chance of a typing error being overlooked in the circuit file - a common error to be found is the use of an undefined Master description caused by a spelling error.

Node Numbers must always appear in at least two entries in the circuit topology to make sense, and a check is made to this end. This traps both typing errors and wrongly defined topologies.

Assuming that a Graph has been formed successfully, the resulting data structure is written to a file with extension ".PLA" (e.g. TEST.PLA) in the format defined in Figure 5.10.

The first item of data is simply a line of text describing the circuit and is followed by the total number of data elements present in the file. Pointers to the main data block lists are stored in the first six positions. This is followed by the main bulk of the data structure and may extend to several thousand locations.

RUN PLANAR

PLANAR.FOR

This program constructs a planar representation of a circuit and stores it in a data structure.

Enter name of file containing circuit AMPUN

LOW VOLTAGE AUDIO POWER AMPLIFIER

Will I proceed to enter crossovers if necessary? YES

INSERTING CROSSOVERS - DON'T GO AWAY!

CROSSING NUMBER 1 INSERTED!

CROSSING NUMBER 2 INSERTED!

CROSSING NUMBER 3 INSERTED!

CROSSING NUMBER 4 INSERTED!

CROSSING NUMBER 5 INSERTED!

5 CROSSOVERS INSERTED

Do you require a PLANAR GRAPH description file? YES

The following components need a description :

RES 1

RES 2

RES 3

RES 4

Enter name of component description file AMP

Enter Data Output File Name OUTPT

END OF EXECUTION

start  
of  
file

TITLE OF CIRCUIT
MAXD=(number of elements in data)
DATA(1)= pointer to first master description
DATA(2)= pointer to first region block
DATA(3)= pointer to first node block
DATA(4)= POINTER TO FIRST UNPLANAR BRANCH
DATA(5)= pointer to first component block
DATA(6)= pointer to first connection block
DATA(7)
-
-
-
-
-
-
DATA(MAXD)

data structure  
elements

Figure 5.10 Format of PLANAR data structure files



## Chapter 6

### The Start Of A Layout - Program "PLACE"

An intermediate program is used to read in pertinent physical data and to position the first components on to the board as a basis for future interaction. Figure 6.1 shows a typical dialogue produced when running the program, which is called "PLACE".

A PLANAR type data structure is needed as input, and a file with the same name, but extension ".LAY" will be produced. The file TEST.PLA, for example, will cause an output file named TEST.LAY. This can then be used by the layout and interconnection program, described in Chapter 9.

#### 6.1 Geometry Of Crossover Sites

Having input values for conductor width and spacing allowance, it is possible to determine the geometry of crossover "components". Unless the user is convinced that the circuit is completely planar, he will have included a Master Crossover description in the circuit data file. This takes the form of:

MASTER CROSS, pad

where "pad" is the size of the incorporated bonding pads. Figure 6.2 illustrates a crossover site graphically.

Pads 1 and 3 will be connected with a short jump-wire and their size is defined by the Master Crossover description given above. The other two pads (4 and 2) will be replaced with a short length of track. It is therefore sensible to make them the same size as the conductor width itself.

Normal spacings are assumed, so the bounding rectangle dimensions and pad coordinates can be calculated directly. These are entered into the crossover Master Component Block (see Chapter 4.2). The crossover now appears to be a normal four-terminal component and is treated as such in future processing.

PLACE.FOR

This program reads in the physical parameters required for a thin film layout and places the first components onto the substrate

Enter name of planar graph data structure file YUK

LOW VOLTAGE AUDIO POWER AMPLIFIER

Enter CONDUCTOR WIDTH and SPACING allowance 30, 30

Enter EDGE PAD size 100

Considering the total component area,  
I suggest a board area not smaller than :

5257 by 5257

Input BOARD X and Y dimensions 6000, 6000

Edge pad positions

NODE NO 4 T/B? T

NODE NO 5 T/B? T

NODE NO 12 T/B? T

NODE NO 1 T/B? B

NODE NO 9 T/B? B

NODE NO 8 T/B? B

NODE NO 2 T/B? B

CHIP XS ORIENTATED 0

COMPONENT R4 ORIENTATED 180

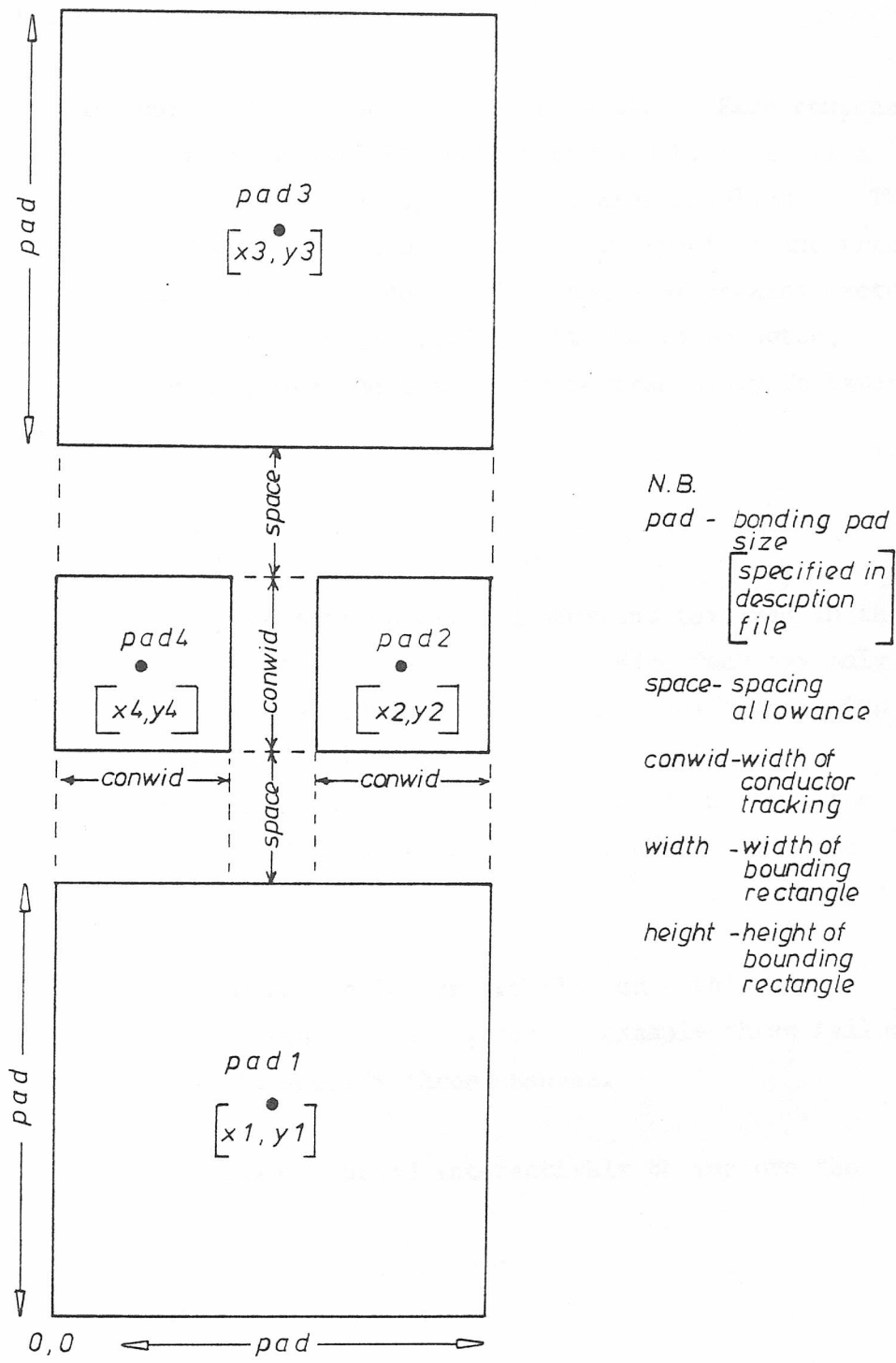
CHIP X3 ORIENTATED 270

COMPONENT R1 ORIENTATED 90

Enter Data Output File Name - OUTFIL

END OF EXECUTION .

Figure 6.1 Dialogue from PLACE



- $[x_1, y_1] \rightarrow [pad/2, pad/2]$
- $[x_2, y_2] \rightarrow [pad - conwid/2, pad + space + conwid/2]$
- $[x_3, y_3] \rightarrow [pad/2, 3 \times pad/2 + 2 \times space + conwid]$
- $[x_4, y_4] \rightarrow [conwid/2, pad + space + conwid/2]$
- width*  $\rightarrow [pad]$
- height*  $\rightarrow [2 \times pad + 2 \times space + conwid]$

Figure 6.2 Physical characteristics of the Crossover Master description

## 6.2 Board Dimensions

The user must next select a suitable board size. Each component in the data structure now has a definite height and width, so it is a simple matter for the computer to add up the total area involved. This area is multiplied by a "packing" factor to allow for spacings and tracks, then used as a guide to board selection. The choice of packing factor is explained more fully in section 11.4.2.2. It should be noted, however, that the user can change the board size interactively in later stages of the design.

## 6.3 Edge Pad Positions

Having set all the necessary physical dimensions involved in the layout, the Edge Pads are placed on the board. Edge Pads may only exist on the top or bottom edges and are automatically positioned with regular spacing. The computer requests the code T (Top) or B (Bottom) for each Edge Pad in an anti-clockwise direction round the board perimeter. Figure 6.3 gives three examples of the dialogue involved.

Only two changes from Top to Bottom are allowed - this corresponds to the left and right board edges. Example three failed because the user attempted to specify three changes.

The Edge Pads can later be moved interactively to improve the layout.

## 6.4 Initial Placement

The final function of program PLACE is to position the first components on to the board, to provide a starting point for the interactive layout stage. All the Regions containing bottom Edge Pads are examined in turn, starting with the pads nearest the right-hand edge and working to the left. The data pointers from the Region loops are compiled in an array to produce a list of components which will be placed along the bottom edge from right to left.

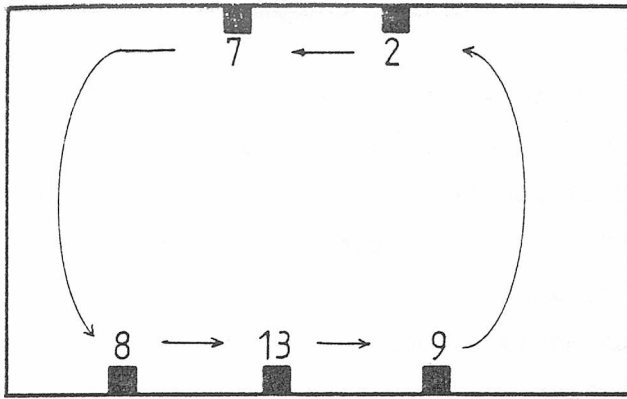
### 6.4.1 Component Orientation

Each component has so far been defined as a bounding rectangle

placement

COMPUTER DIALOGUE

✓



Example 1

NODE No. 7 T/B ? T

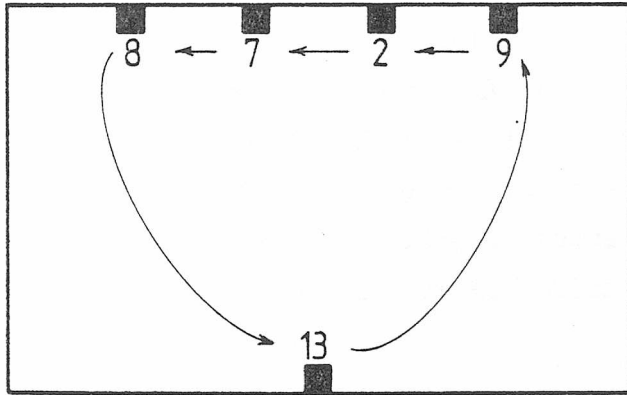
NODE No. 8 T/B ? B

NODE No. 13 T/B ? B

NODE No. 9 T/B ? B

NODE No. 2 T/B ? T

✓



Example 2

NODE No. 7 T/B ? T

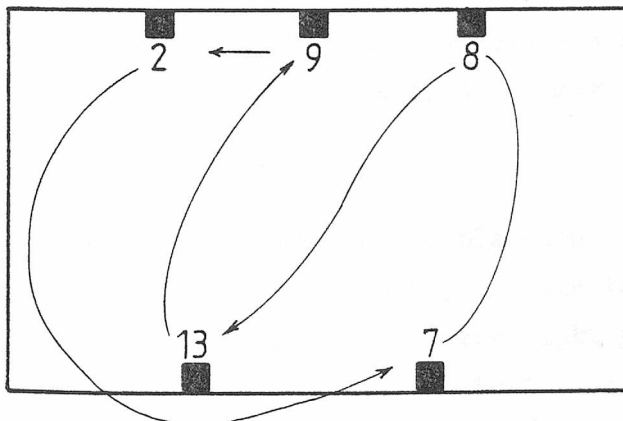
NODE No. 8 T/B ? T

NODE No. 13 T/B ? B

NODE No. 9 T/B ? T

NODE No. 2 T/B ? T

✗



Example 3

NODE No. 7 T/B ? B

NODE No. 8 T/B ? T

NODE No. 13 T/B ? B

NODE No. 9 T/B ? T

you cannot define them  
like this-- the cyclic  
order is all wrong !

Figure 6.3 Edge - pads, cyclic definition

encompassing a number of terminal pads. The first task is to decide which angle of rotation is required for each member of the list.

There are exactly eight different pad positions within the bounding rectangle - these are shown in Figure 6.4. If the coordinates of a pad are equi-distant from an X or Y boundary line, the pad is deemed a "Corner Pad". The four Corner Pads have been labelled TL (Top Left), BL (Bottom Left), TR (Top Right) and BR (Bottom Right). A conductor track can start at a Corner pad in one of two ways; parallel to the X axis or parallel to the Y axis.

The other four positions, namely L (Left), R (Right), T (Top) and B (Bottom), force the track to start in one direction only. It is good sense, then, to position pads at the vertices if possible as this may save connection length.

The component is connected to the Region by two of its pads. One pad, the "Incoming" pad, will be connected to another component or Edge Pad to the RIGHT of it, while the "Outgoing" pad will be connected to the LEFT. The ease of routing the connections is obviously very dependant upon the positioning of these two pads within the bounding rectangle. Consider Figure 6.5 which shows a two-pad component with its associated connections.

In the original definition, the Incoming pad is positioned on the Left and the Outgoing pad on the Right. The component can be orientated in one of four different positions, since the bounding rectangle edges are restricted to lie on either the X or Y axis. The allowable angles are 0, + 90, - 90 and  $\pm$  180 degrees - where a positive angle refers to a clockwise rotation. Figure 6.5(b) shows the best solution for this example - a rotation of  $\pm$  180 degrees. This is only an initial orientation and may subsequently be changed to improve the layout.

Each component in the placement list must be assigned an orientation angle in this way. The Incoming and Outgoing pads are first classified into one of the eight position codes, then a look-up table is used to read off the best orientation. The table can be seen in Figure 6.6.

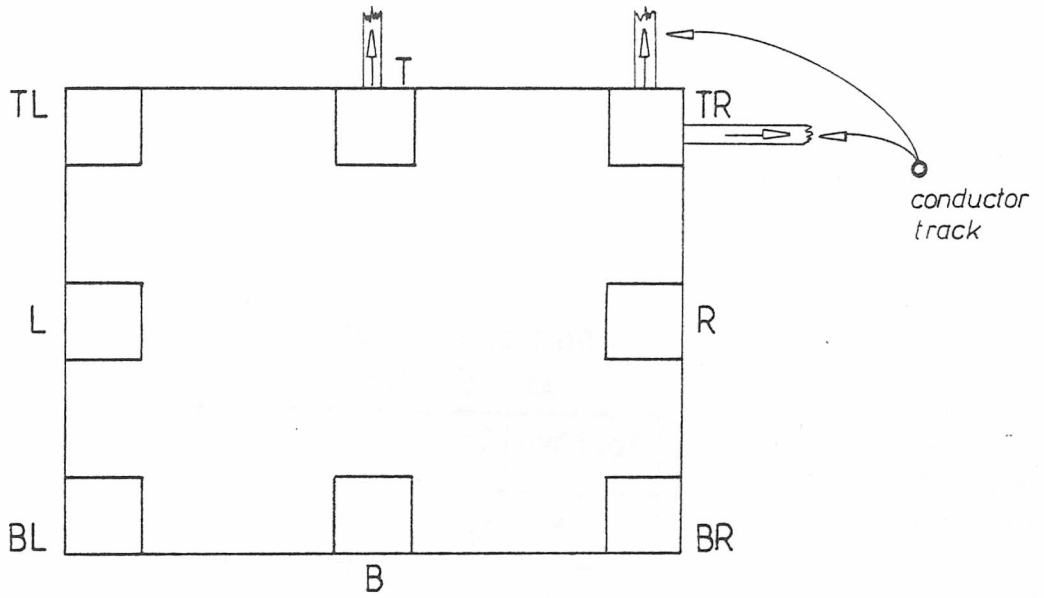
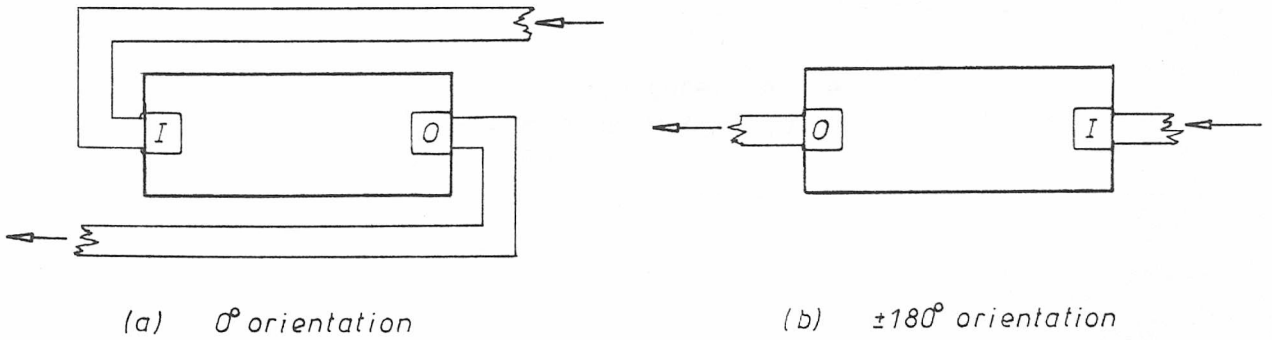


Figure 6.4 Possible pad locations inside a component boundary



$I$  = Incoming Pad (from Right)  
 $O$  = Outgoing Pad (to Left)

Figure 6.5 Component orientation

		outgoing pad position							
		TL	T	TR	R	BR	B	BL	L
ingoing pad position	TL		$\pm 180^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$+90^\circ$	$+90^\circ$	$+90^\circ$
	T	$0^\circ$	$\pm 180/0$	$\pm 180^\circ$	$+90^\circ$	$+90^\circ$	$+90^\circ$	$+90^\circ$	$0^\circ$
	TR	$0^\circ$	$0^\circ$		$+90^\circ$	$+90^\circ$	$+90^\circ$	$0^\circ$	$0^\circ$
	R	$0^\circ$	$0^\circ$	$-90^\circ$		$+90^\circ$	$0^\circ$	$0^\circ$	$0^\circ$
	BR	$0^\circ$	$-90^\circ$	$-90^\circ$	$-90^\circ$		$0^\circ$	$0^\circ$	$0^\circ$
	B	$-90^\circ$	$-90^\circ$	$-90^\circ$	$-90^\circ$	$\pm 180^\circ$		$180/0$	$0^\circ$
	BL	$-90^\circ$	$-90^\circ$	$-90^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$-90^\circ$		$-90^\circ$
	L	$\pm 180^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$\pm 180^\circ$	$+90^\circ$	

Where two angles are given, the figure on the left applies if the Ingoing Pad is to the left or lower than the Outgoing Pad

Figure 6.6 Component Orientation



Special cases arise when the position codes of the pads are the same - this corresponds to the main diagonal in the table. It is impossible to have two identical Corner Pad codings since this would imply that the pads overlap one another. The remaining elements along the diagonal need a further test before an orientation can be chosen. Consider Figure 6.7 which illustrates the situation when both pads are on the right edge of the bounding rectangle.

The Incoming pad in (a) is lower than the Outgoing pad. This requires an anti-clockwise rotation of 90 degrees (-90). If the pad positions had been reversed (shown in diagram (b)), a clockwise rotation is needed (+90). Each legitimate position on the table diagonal, then, has two arguments instead of just one.

When every component has been assigned an initial orientation, the program proceeds to the actual placement algorithm shown in Figure 6.8. A total is first made of all component widths in the X direction. Suitable spacing is added between components to allow the connections to be routed and the total compared with the board width. If the board is wide enough the components are placed along the lower board edge and their co-ordinates entered into the relevant component data blocks.

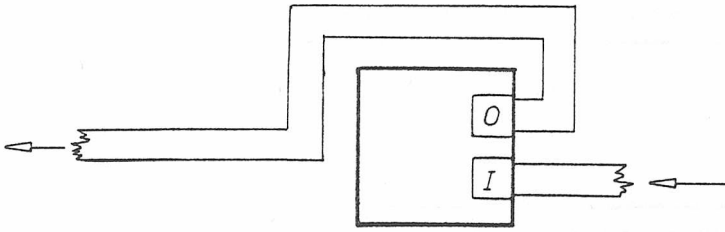
Figure 6.9(a) shows such a case where the component string has fitted neatly into the available space. The components are packed from the left with minimum spacing and the user will have the option of reducing the board width to save wasted space. It is also possible to automatically increase the inter-component spacings in order to take up the whole board width.

#### 6.4.2 Component Rotation

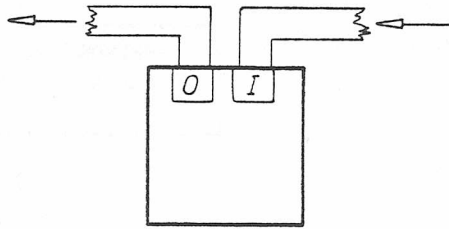
If the string of components do not fit the board, one or more of them are rotated to save space. Each component's horizontal width is determined from its physical dimension and necessary connection spacings, then compared with its corresponding vertical height. (Refer to Chapter 8 for details regarding connection spacing allowances). The possible saving in width is simply the difference between these two figures.

*I = Incoming Pad*

*O = Outgoing Pad*

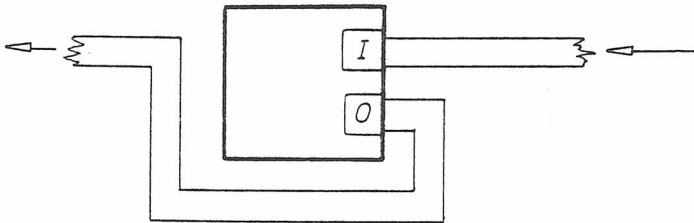


*0° Orientation*

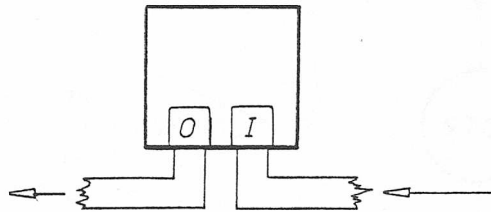


*-90° Orientation*

*(a) Incoming Pad lower than Outgoing Pad*



*0° Orientation*



*+90 Orientation*

*(b) Incoming Pad higher than Outgoing Pad*

*Figure 6.7 Orientation decisions for pad positions "R"/"R"*

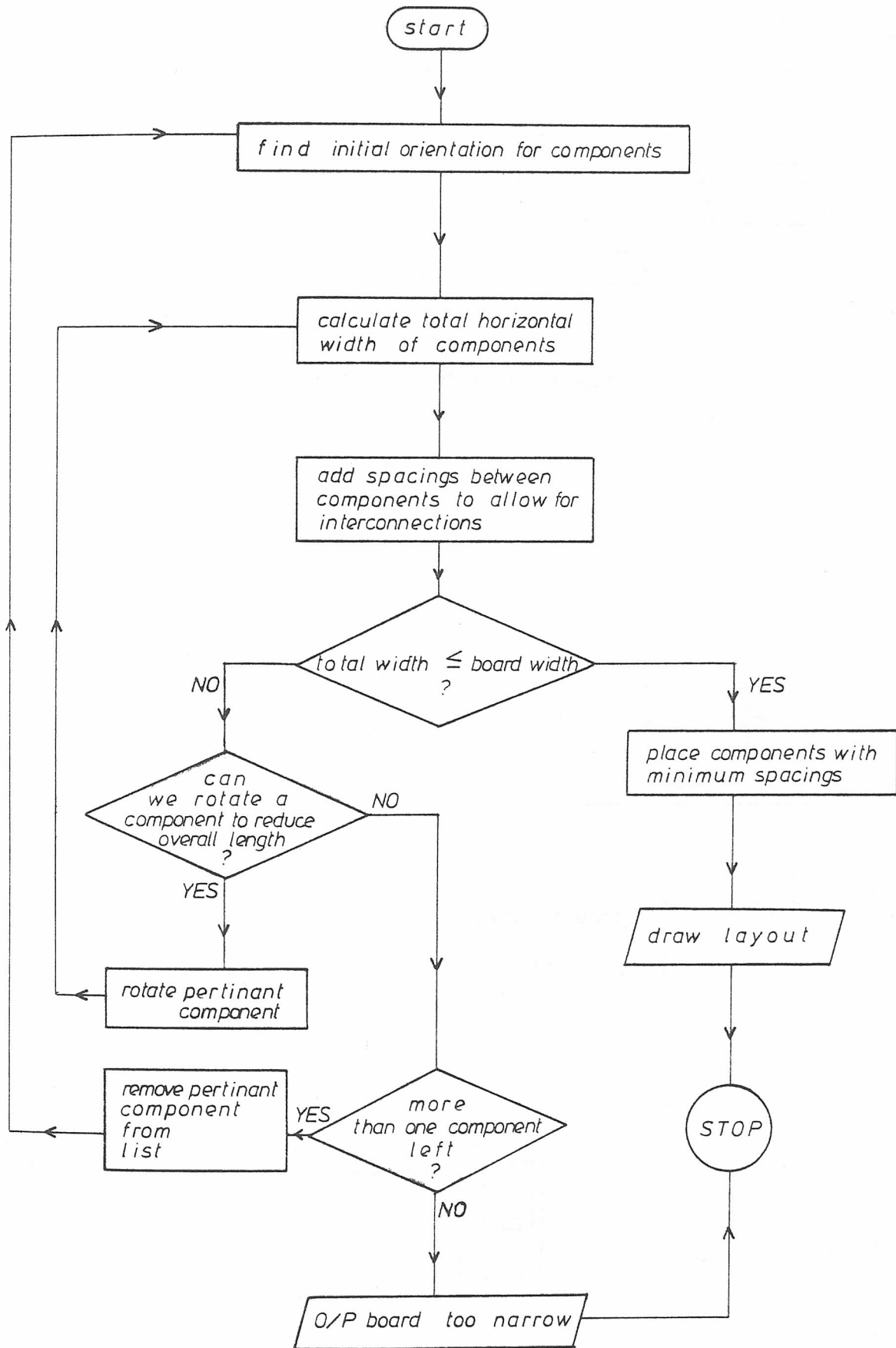
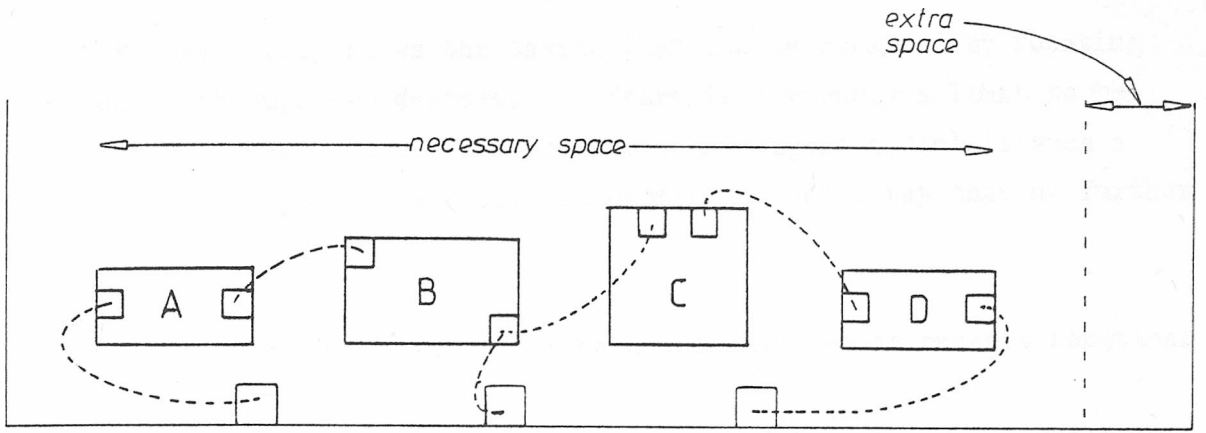
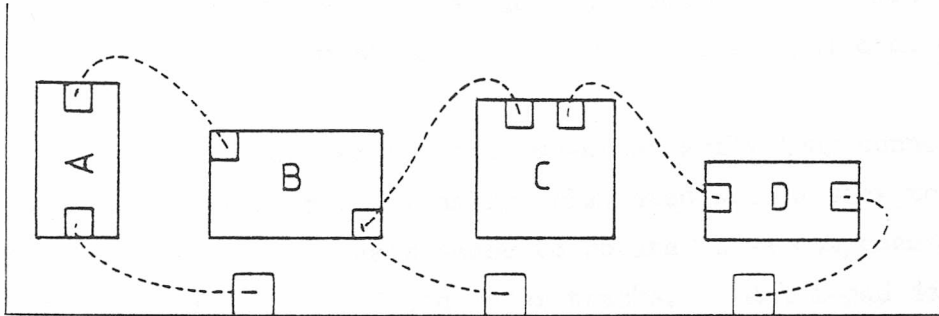


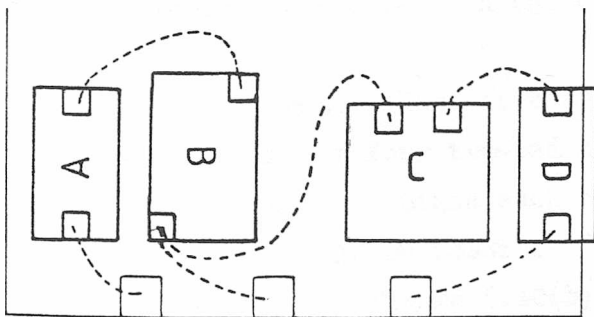
Figure 6.8 Initial placement algorithm



Example (a) board width adequate



Example (b) one component rotated



Example (c) all components rotated but board too small

Figure 6.9 Initial placement technique

Figure 6.9(b) shows the saving that can be obtained by rotating component A through  $-90$  degrees. There is obviously a limit to the saving which can be gained from rotation, and Figure 6.9(c) is such a case. The components are all orientated in such a way that no further saving can be obtained.

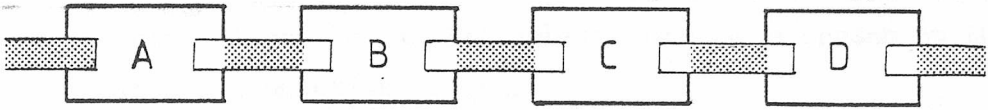
Three basic heuristic rules have been adopted as regards rotation:

- (a) Two-pad components should be rotated in preference to multi-pad components.
- (b) Components which are connected to bottom Edge Pads should be rotated such that the relevant connection pad is nearest the bottom of the board.
- (c) Any two adjacent components which are to be rotated must be rotated in opposite cyclic directions.

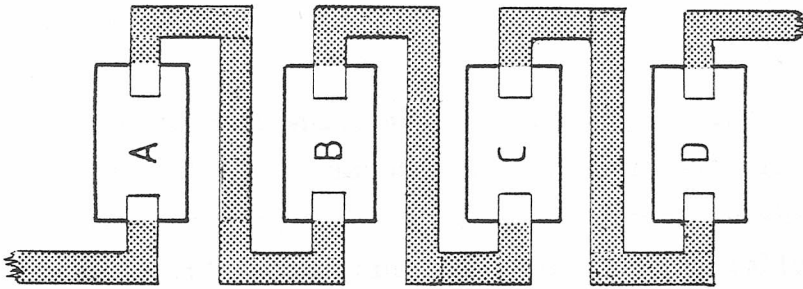
Rule (a) has been adopted to avoid un-necessarily long connections. The components have been provisionally orientated with a view to easing the routing process, so it makes sense to rotate those components which will affect the least number of conductor tracks. Multi-pad devices should only be rotated as a last resort.

The second rule applies to components which are directly connected to an Edge Pad. Usually it is not clear whether a clockwise or anti-clockwise rotation will cause the least increase in connection length, but in this case the component is rotated such that the appropriate pad is nearest the bottom edge. This minimises the conductor length to the Edge Pad, although the effect on the other connection is not clear.


Rule (c) is best illustrated by example. Consider Figure 6.10(a) which shows a string of four two-pad components A, B, C and D in their original orientations. Since each component is larger in the X direction than in the Y, it should be possible to reduce the total width of the string by rotation. Figure 6.10(b) results from rotating each component through  $90$  degrees in an anti-clockwise direction. The saving in width is less than one might have expected because extra space must be allocated between the components for interconnections. A much better solution is to rotate adjacent component pairs in opposite directions.

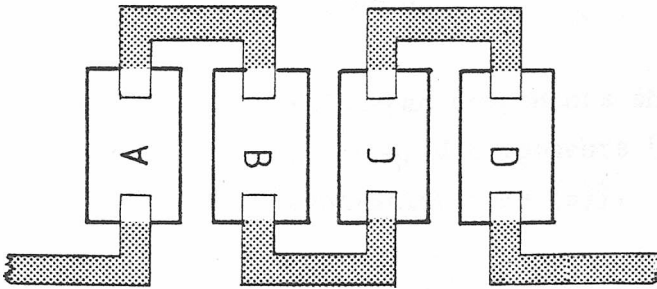


(a) Original placement



(b) All components rotated  $-90^\circ$

 conductor track



(c) Components rotated alternatively  $-90^\circ$  and  $+90^\circ$

Figure 6.10 Rotation Rule (c)

This eliminates the need for extra spacing and produces a compact result. The total conductor length has still increased significantly, but this is only to be expected since the initial orientations were chosen on the principle of minimum connection length.

#### 6.4.3 Removing Components To Upper Levels

When the rotation method fails to reduce the component string width sufficiently to fit the board, a technique known as "Component Removal" is employed. A component is selected and removed from the placement list. The orientation angles for the other components are then reset to their original values and the placement algorithm repeated. Several components may have to be removed in this fashion until a solution can be found.

The components which are removed play no further part in the initial placement algorithm and will be dealt with in the actual Placement Algorithm mentioned in Chapter 7. The order in which the components are removed is quite important, however, and the following rules have been formulated as a guide to this decision:

- (a) The components in the list are numbered sequentially from the left. Only even-numbered components can then be removed.

When all of these have been removed (excluding end components), the remaining components are re-numbered.

- (b) Multi-pad components should be removed in preference to two-pad components (providing that this does not contradict rule (a)).

The first rule describes a strict removal algorithm designed to minimise the connection length between layers, or "stacks", of components with the consequent saving in board area that this implies. The algorithm is explained with reference to Figure 6.11.

Seven components have been placed in their original orientations in diagram (a). Smaller board widths would cause rotations to occur until a critical point is reached where Component Removal is required.

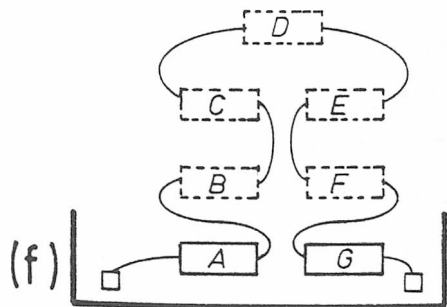
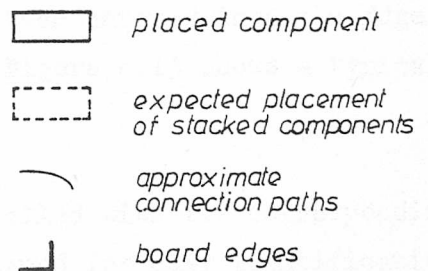
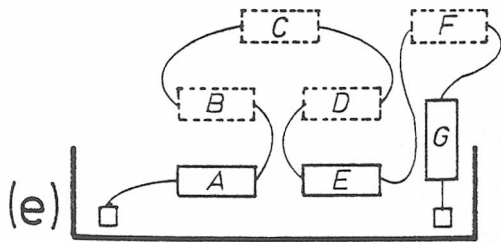
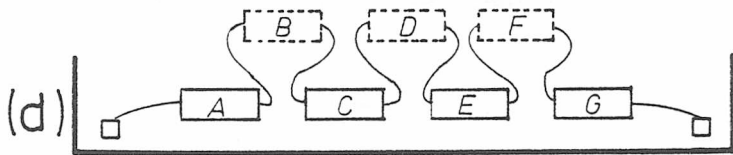
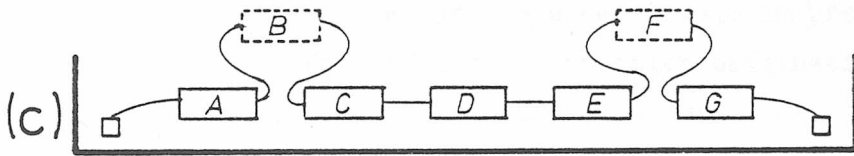
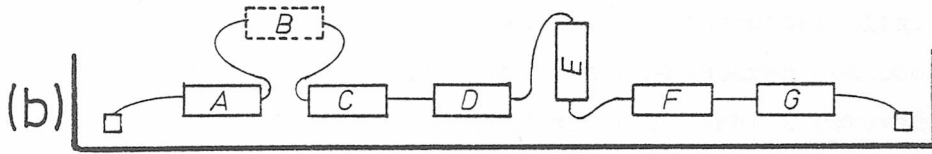
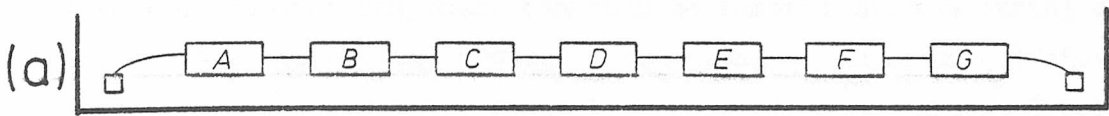


Figure 6.11 Component Stacking



Any even-numbered component can then be removed and the string checked to see if rotation will produce a solution. Diagram (b) shows the situation where component B has been removed and component E rotated.

All even-numbered components have been removed in diagram (d), so it is necessary to re-number the string. Component C then becomes number 2 in the string and is therefore eligible for removal. There being no other legitimate even-numbered component in the string, re-numbering is again applied and component E removed (diagram (f)).

A similar removal technique is applied in later processing, and this produces a multi-tier solution. Rule (b) ensures that multi-pad components are removed to upper levels in preference to two-pad components. This avoids a connection originating at the bottom of the board and having to travel through several layers of components.

The algorithm tends to produce a neatly stacked layout in which components are connected to immediate neighbours with a minimal amount of spacing and conductor length.

## 6.5 Program "PLACE"

PLACE produces a data structure file to be used by LAYOUT, the program for interactive layout and interconnection. The output format is similar to a PLANAR data structure except that it contains additional physical dimensions (see Figure 6.12).

It is now possible to obtain a graphical representation of the layout as it stands, since co-ordinates have been entered into the Edge Node blocks and certain Component Blocks. Figure 6.13 shows a typical graphical output from PLACE.

Each Edge Pad and component pad is labelled with its corresponding circuit Node Number, and the components are named for easy identification. These drawings provide the operator with a means of visually checking the effects of different Edge Pad positions before a final choice is made. Chapter 11, section 4, discusses this process.

start  
of  
file

TITLE OF CIRCUIT
board width
board height
edge pad size
conductor track width
spacing allowance
MAXD = number of elements in data
DATA(1) = pointer to first Master Description
DATA(2) = pointer to first Region Block
DATA(3) = pointer to first Node Block
DATA(4) = pointer to first Unplanar Branch
DATA(5) = pointer to first Component Block
DATA(6) = pointer to first Connection Block
DATA(7)
...
DATA(MAXD)

data structure elements

Figure 6.12 Format of PLACE/LAYOUT data structure files

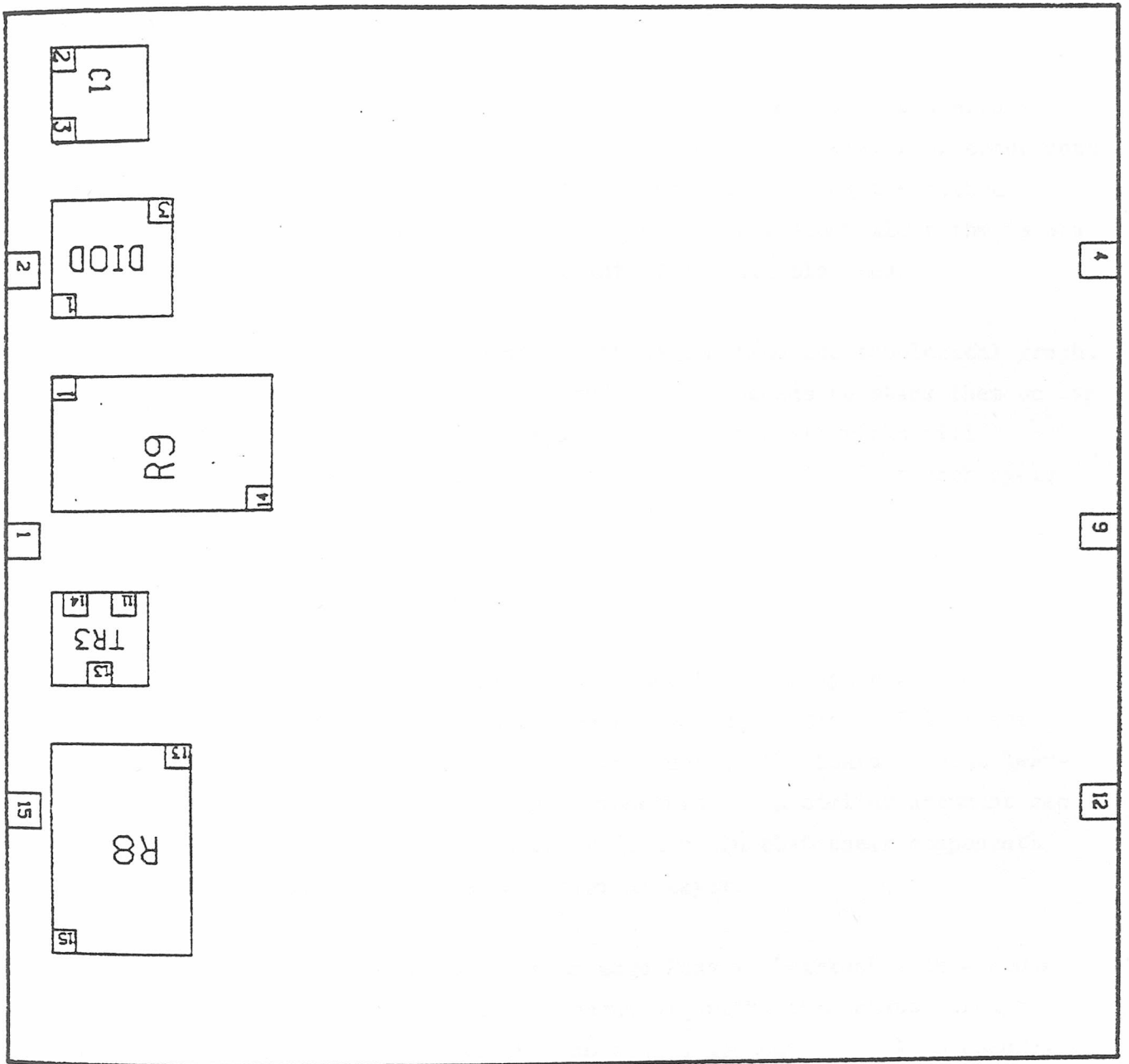


Figure 6.13 Graphical output from PLACE

The Placement Algorithm

7.1 Introduction

The layout at this stage consists of a rectangular board with a number of Edge Pads along its top and lower edges. A layer of components have already been placed at the bottom of the board using the methods described in the last chapter. A sequential placement algorithm is now used to fit the remaining components into the available area.

The Algorithm chooses a pertinent Region from the topological graph, forms a list of components to be placed, then proceeds to stack them on top of the existing components. Repeated use of the Algorithm will generate a complete layout and it is convenient to halt after each cycle for human interaction.

7.2 Choosing The Next Region

The Initial Placement Algorithm described in Chapter 6 used components from the Regions containing bottom Edge Pads. This meant that they could be placed along the lower edge of the board without leaving space underneath them for other components. A similar argument can now be applied to immediately adjacent Regions, in that their components can be stacked directly above the original layer.

Every Region containing bottom Edge Pads is "Marked" with a coded word in its data block. The Placement Algorithm then hunts through the Marked Regions looking for any unplaced components. This ensures that any components which have been removed from the Edge String during the initial placement are dealt with first.

Figure 7.1 shows the Region selection process in the form of a flowchart. A Region is chosen according to the following rules:

- (a) The Region must be Marked
- (b) It should contain the fewest unplaced components of all the Marked Regions.

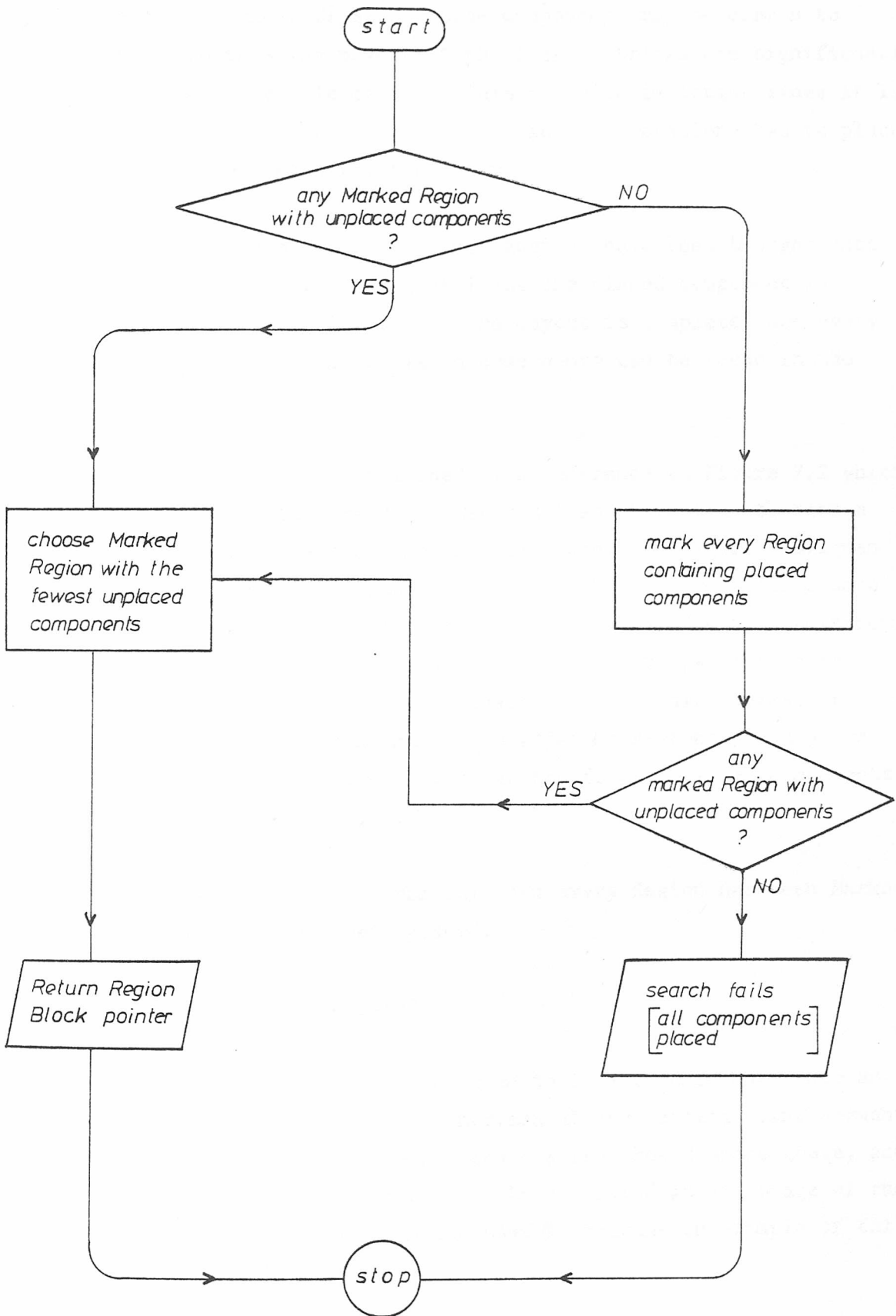


Figure 7.1 Flowchart of Region selection process

The second rule simplifies the placement task by processing smaller component strings first. Some components may be common to several strings so this may mean that the longer strings are significantly reduced before they are selected. This is quite important since it is much more difficult to place a long string in one iteration than to place several short strings one after the other.

When all the components in Marked Regions have been brought onto the board, every Region containing at least one placed component is Marked and the process continues. The layout is complete when every Region has been Marked and no unplaced components can be found in the search.

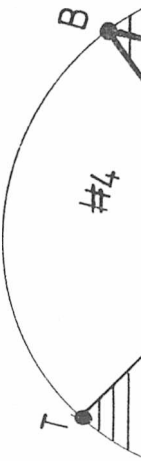
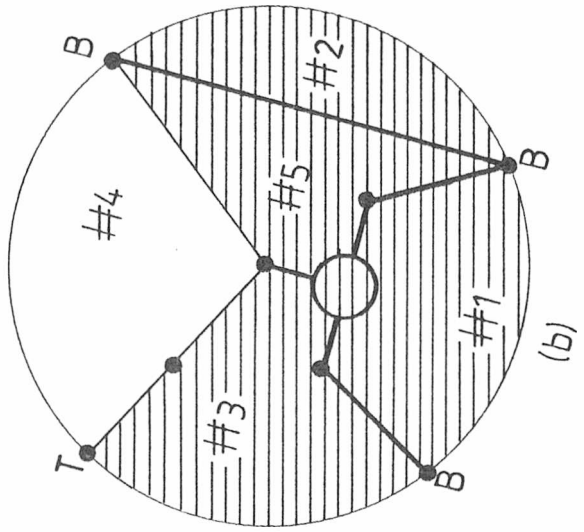
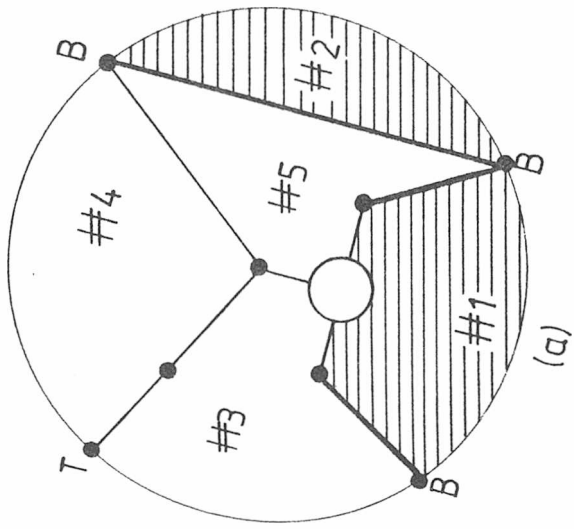
The technique is best explained with reference to Figure 7.2 which shows a simple topological graph. Regions 1 and 2 contain the bottom Edge Pads and have been used at the initial placement stage. Diagram (a) indicates that three 2-pad components have already been placed onto the board and that a multi-pad device has been removed due to a restrictive board width. Both Regions are Marked and the multi-pad device is selected as the first component to be dealt with. This exhausts the Marked Regions of unplaced components, so adjacent Regions 3 and 5 are added to the list. Since the latter has the fewer unplaced components it is chosen first (diagram (c)).

The process halts at Diagram (d) when every Region has been Marked and the components have all been placed.

### 7.3 Forming The "Slot Boundary"

The Placement Algorithm is designed to insert components into an irregular lower boundary composed of horizontal and vertical line segments. The "Slots" are the dividing line between the free board space above, and the occupied area below. The Slots can be displayed at any stage of the design using the "SLOT" command in the LAYOUT program- an example of this is given in Figure 7.3.

A simple one dimensional array is used to store the co-ordinates of each vertice in the Slot Boundary - which is defined from the left side of the Board. The "Slot Array", as it is called, is originally set up to be



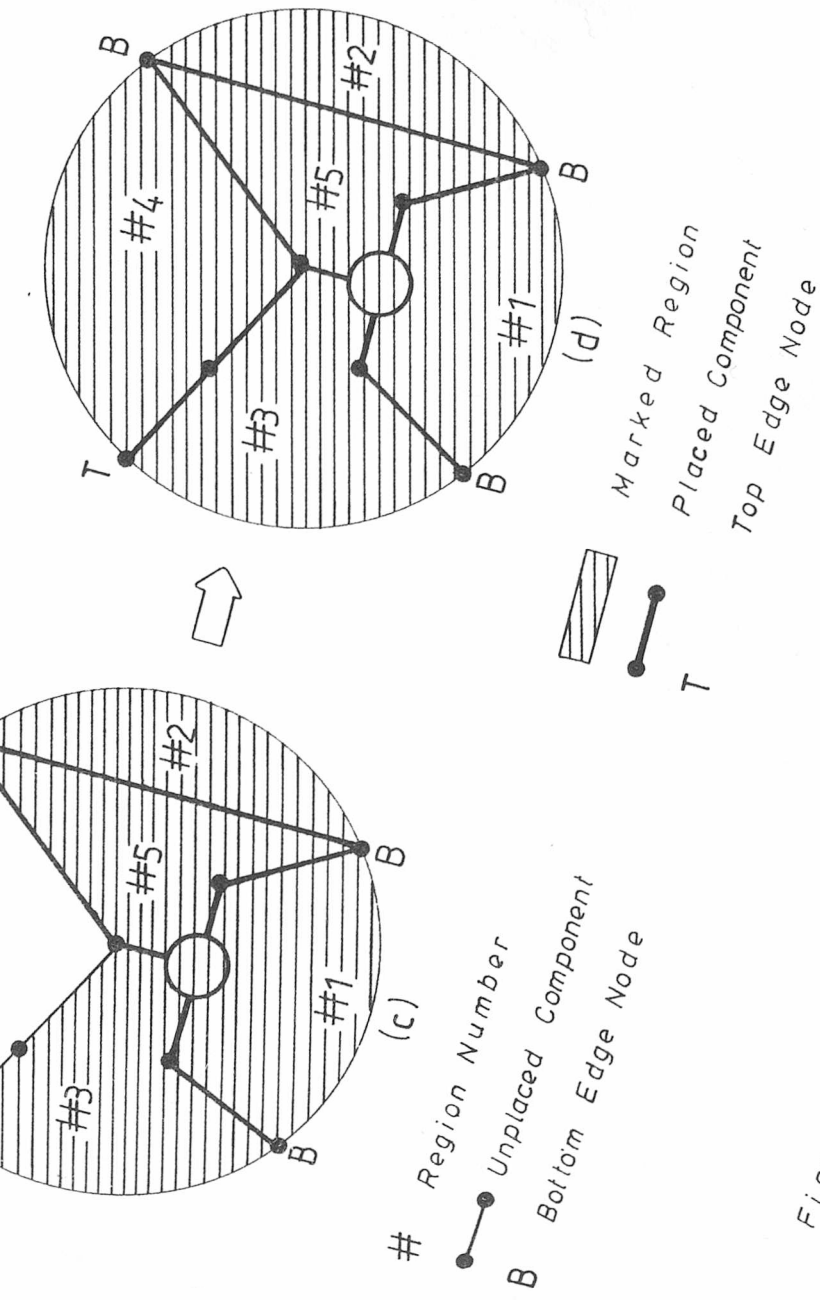
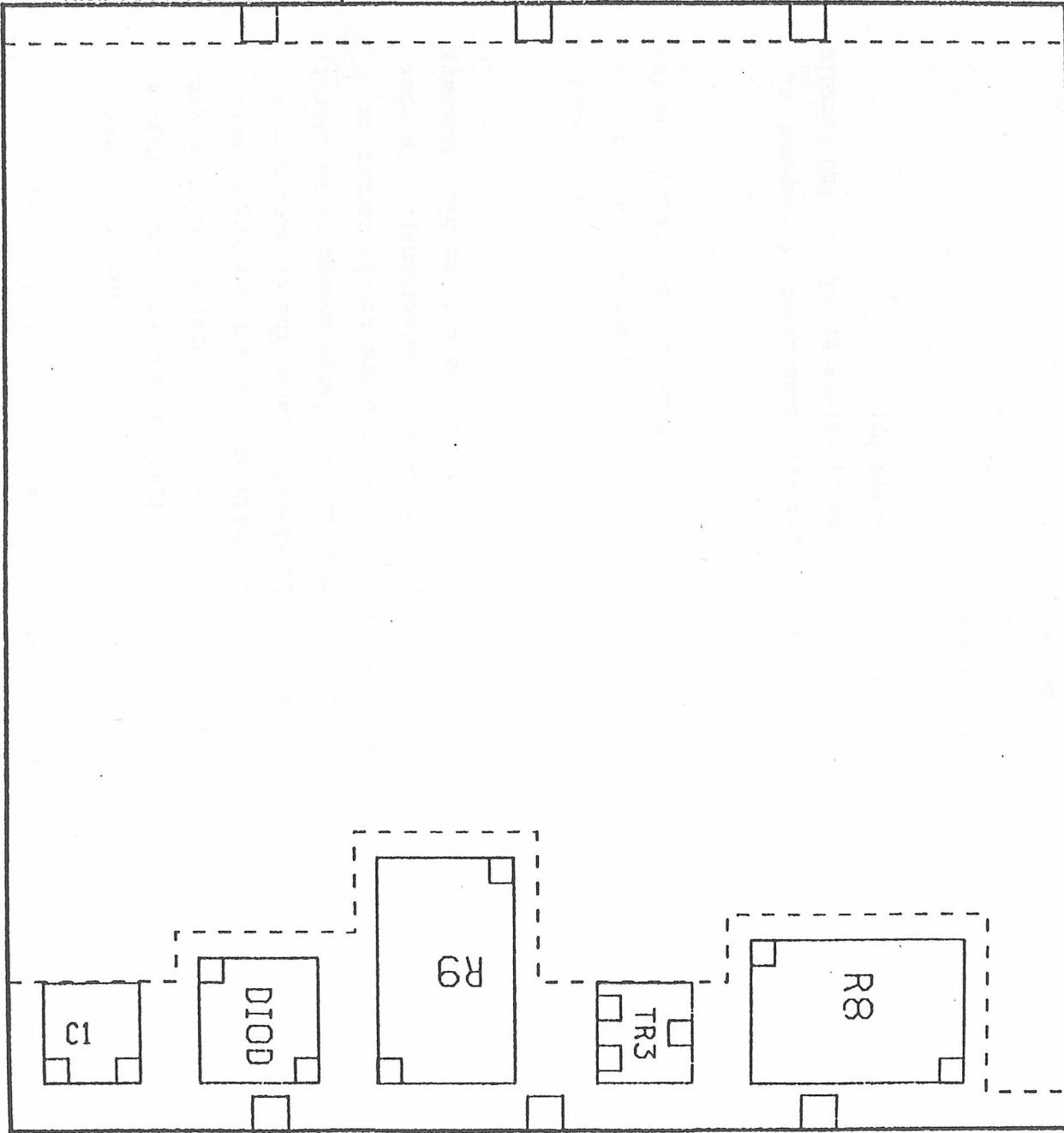


Figure 7.2 Example of Region Selection  
 - 123 -



Thin Film Circuit with pre-defined Resistors



What Next ?

\*SLOTS  
\*

Figure 7.3 Displaying the Slots while running the "LAYOUT" program

a two point horizontal line immediately above the bottom Edge Pads as shown in Figure 7.4 (a). This is deformed to include components by adding points to the array (see diagram (b)). The deformation allows space for connections round the components so it should not, in theory at least, matter whether the connections actually exist at this stage.

Each placement stage uses the Slot Boundary to bring on another string of components, then changes the definition accordingly. Figure 7.4(c) illustrates the main characteristics of the deformation technique. Component "C" on the left has fitted neatly into a slot leaving a small amount of extra space. This would normally cause the formation of a small slot to the right - if the width had been large enough to be useful. The Slot is deemed "Inconsequential" if two conductor tracks cannot be routed into it (corresponding to Input and Output connections). In this example the Slot width is too small to be of any use and the Slot Boundary has ignored it.

Component "D" has not fitted any of the Slots but has been placed as close to the boundary line as possible. This technique is used as a last resort since it wastes valuable space by "Masking" potentially useful slots.

The size of the Slot Array changes with each layer of components. In diagram (c), for example, it fell from 14 points to 10. The formula for the maximum possible array size is given by:

$$N_{max} = 2 \times Bw / (2 \times Cw + Sp)$$

where: Bw = current board width

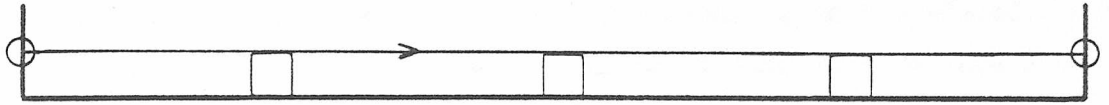
Cw = connection track width

Sp = spacing allowance (constant)

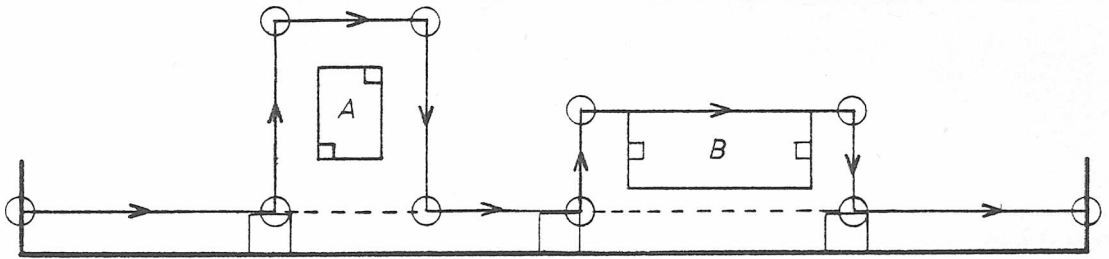
This formula assumes that the Boundary is composed entirely of Slots which are just larger than the "Inconsequential" width of  $2 \times Cw$ . In general the array size reaches only a small fraction of this maximum.

#### 7.4 Placement Techniques

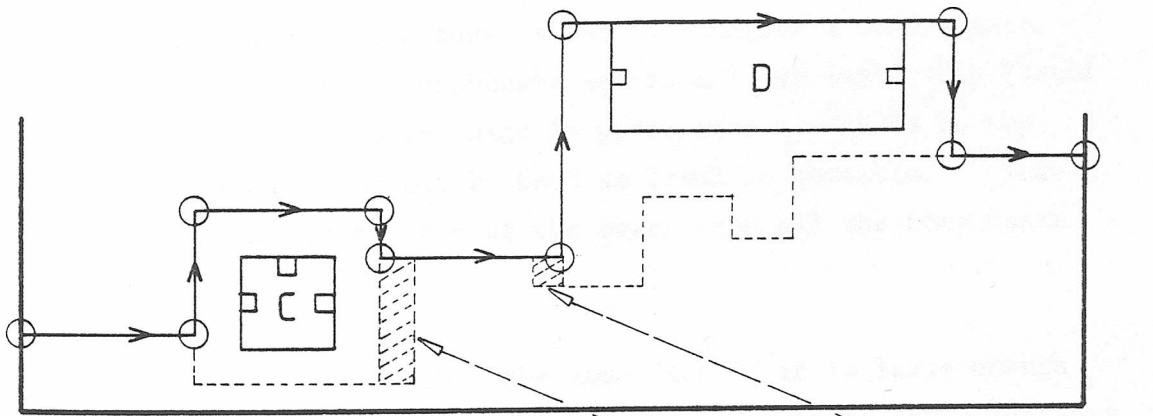
Having set up the Slot Boundary and chosen a Region, the program proceeds to the actual positioning stage. The Region can be thought of as a string of placed components connected across a string of unplaced



(a) Original definition



(b) Initial alteration due to edge string



(c) Subsequent alteration

⊙ new slot co-ordinates      → new slot boundary  
 ----- old slot boundaries

Figure 7.4 Slot Deformation

components as in Figure 7.5. All the unplaced components will eventually lie above the existing ones so it is possible to define an X range on the Slot Boundary into which the new components should be fitted. The range boundaries are set with regard to the two end components in the placed string. In Figure 7.5, for example, the boundary lines are defined from the Incoming pad of component A to the Outgoing pad of component C. The general rule is that the component string should be placed within the two points where it joins the existing components.

The unplaced components are assigned an orientation angle in exactly the same way as the bottom layer (refer to 6.4.1). The string is then fitted onto the Slot Boundary using a combination of the techniques described in the next section. Figure 7.6 shows the flowchart of the process.

#### 7.4.1 Filling The Slots

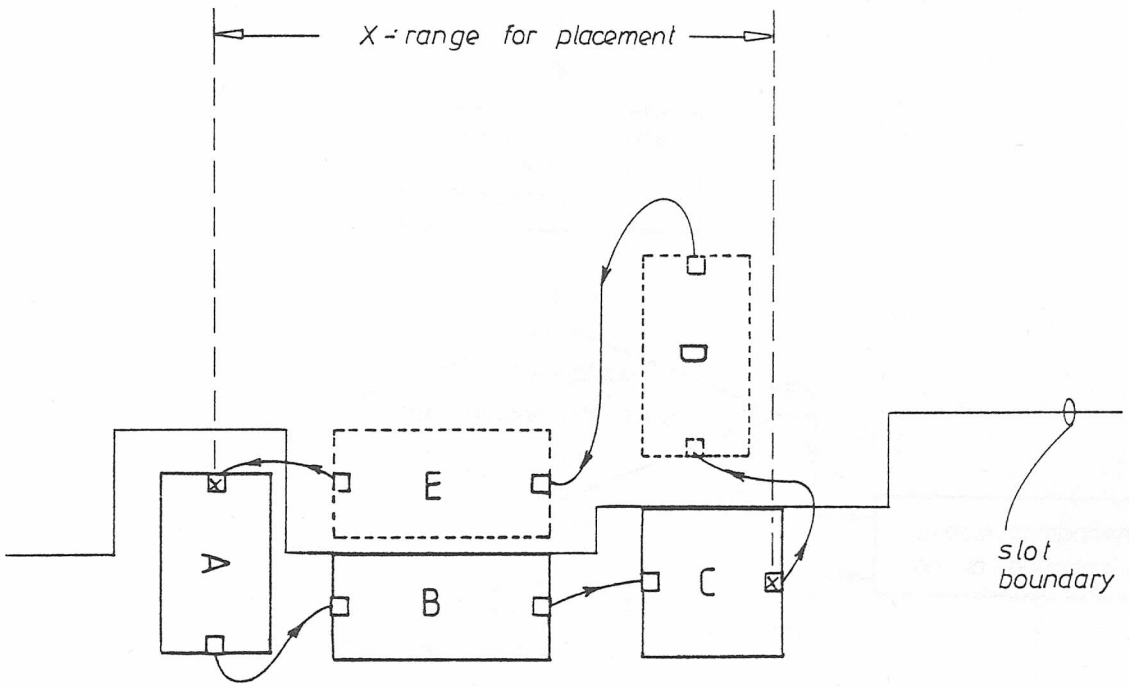
The best placement method in terms of space utilisation and conductor length is to place the components directly into Slots between the X boundaries. This is demonstrated in Figure 7.7.

A Slot is defined as a horizontal section of Boundary in which at least one of its immediate neighbours is set to a higher Y co-ordinate. A horizontal segment with both neighbours set to a lower level than itself is called a "Mound". Slots are used in preference to Mounds on the principle that the Boundary should be kept as level as possible. This reduces the wasted space at the top of the board when all the components have been placed.

Several components may occupy the same Slot if it is large enough after suitable spacing has been allocated around and between the components for conductor tracks.

#### 7.4.2 Filling The Slots/Mounds

If the first method fails to fit all the components between the X boundaries, the Slot - only restriction is relaxed. Components may now be placed on any horizontal line segment (see Figure 7.8).



Region defined  $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$

where A, B and C are placed components

Figure 7.5 Defined Xrange over slot boundary

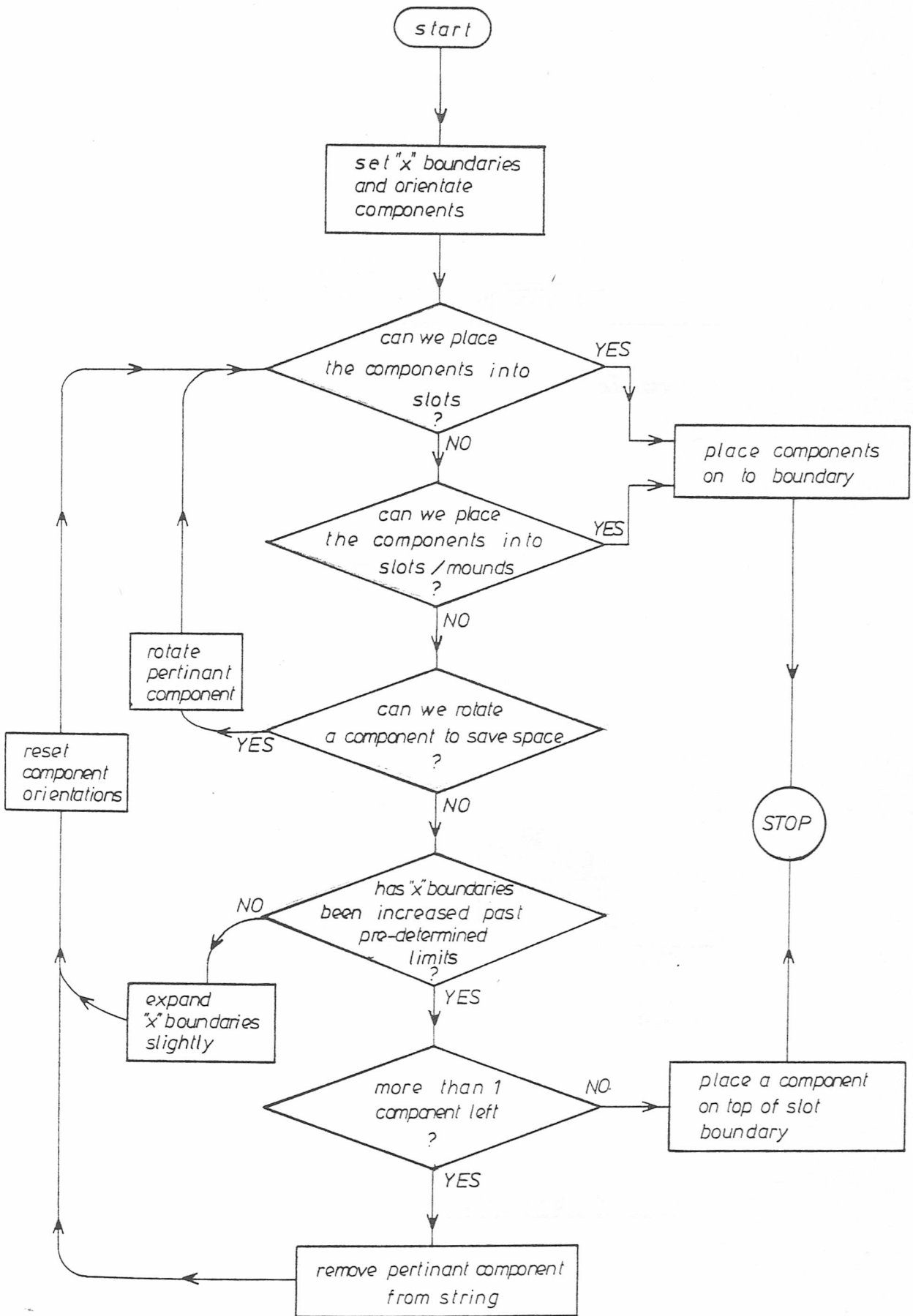


Figure 7.6 Placing a string of components into the Slot Boundary

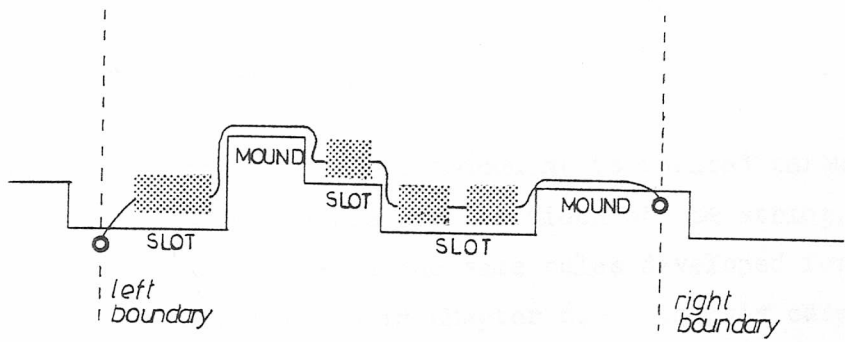


Figure 7.7 Filling the Slots

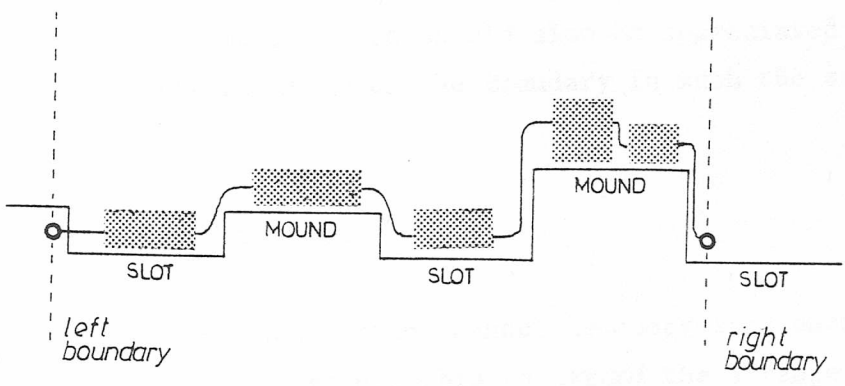


Figure 7.8 Filling Slots and Mounds

While this technique does not actually waste space as such, components placed on Mounds will accentuate the irregularities of the Boundary line. This increases the likelihood of unused space being trapped between the Boundary and the top board edge when the layout is complete.

#### 7.4.3 Component Rotation

When both methods fail, a component is rotated through  $\pm 90$  degrees in order to reduce the overall width of the string. A component is selected for Rotation using the same rules developed for the Initial Placement algorithm described in Chapter 6. In this case, however, it is no longer sufficient that the total component string width be narrower than the available space, since the Slots and Mounds may be of inconvenient sizes. The effect of Rotation merely increases the possibility that a solution will be found - the components appear smaller so they have a better chance of fitting the available Slots.

A single component is rotated each time the two placement methods fail. This continues until a solution is found or the string has been compacted as far as possible without success. This simple loop ensures that Rotation, with its associated increase in conductor length, is kept to an absolute minimum. It should also be appreciated that Rotation will accentuate the irregularity of the Boundary in much the same way as the Mound method.

#### 7.4.4 X Boundary Expansion

When the component string cannot be compressed enough to fit the Slot Boundary, it is often possible to extend the X range involved. The left and right bounding lines are gradually moved to either side until the string can be placed. Figure 7.9 demonstrates this technique using a string of three unplaced components.

The boundary edges are incremented at the same rate unless one side is constricted, in which case the expansion is confined to the other side. It is very difficult to decide when to abandon this approach in favour of other techniques. At present the algorithm is allowed to expand the boundary range by a fixed percentage of its original size. This was



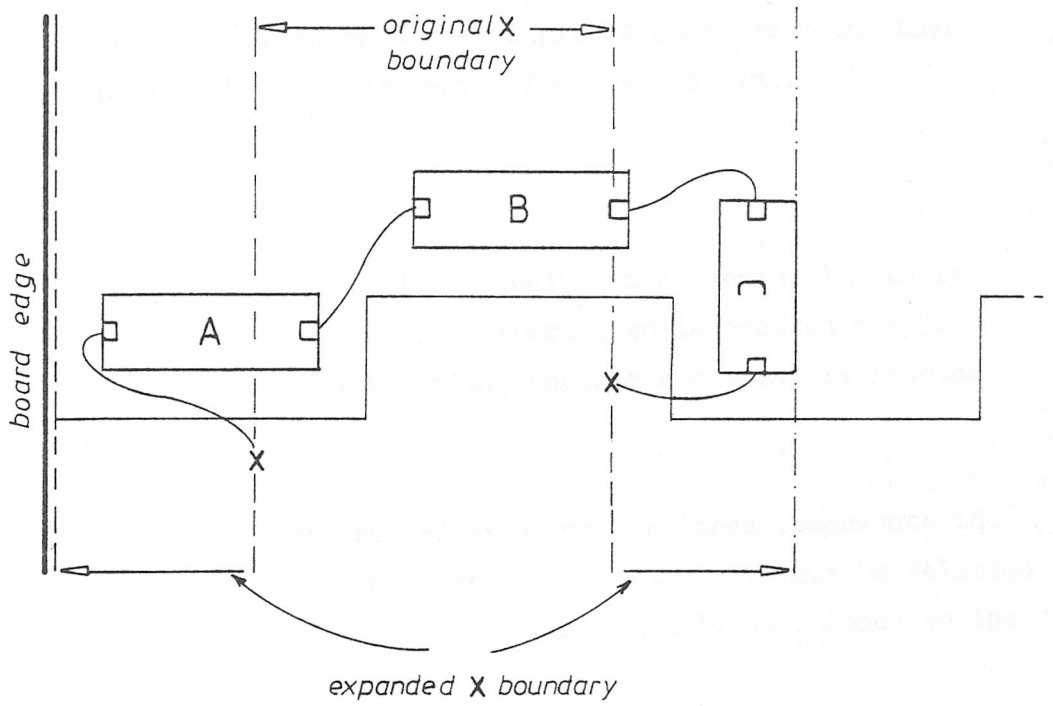


Figure 7.9 Expanded "x" boundary

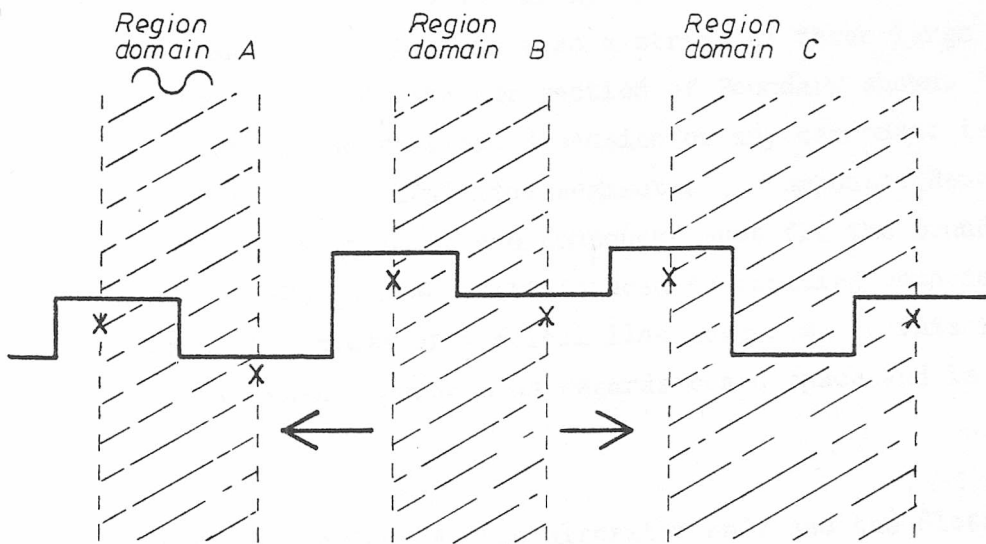


Figure 7.10 Placement domains

intended as a temporary measure until the "Domain" approach shown in Figure 7.10 is implemented. Expansion would then be limited by board edges and other X range boundaries. This prevents the expansion obscuring component connection pads from unplaced components in other Regions, since the X ranges are prevented from overlapping.

#### 7.4.5 Component Removal

This technique is very simple to apply - a component is simply removed from the string using the rules formulated in Section 6.4.3. If this still doesn't produce a solution, another component is removed and the process continues.

Removed components are treated as normal unplaced components in further placement stages. Any particular Region may thus be selected several times before all of its components are finally positioned on the board.

#### 7.4.6 Rough Placement

The positioning methods described in the previous sections try to place components such that they do not lie across vertical line segments in the Slot Boundary. This is not always possible as in the case of Figure 7.11, for example. In this case a string of three large components have to be placed onto the section of Boundary shown. Rotation is of no use here since the smallest dimension of any component is still larger than any of the horizontal line segments. Component Removal is equally useless since at least one component must fit the boundary. A technique known as "Rough Placement" is used to position component "A" onto the Boundary, regardless of vertical line segments. This is the least economical placement approach as regards board space and is used as a last resort.

Having placed component A, the algorithm ends and the Slots are re-defined. The rest of the string are treated as removed components and the Placement Algorithm must be used several times in succession.

#### 7.5 Using The Placement Algorithm

The principle aim of the Algorithm is to bring the next components onto the board ready for human interaction. If the components have been

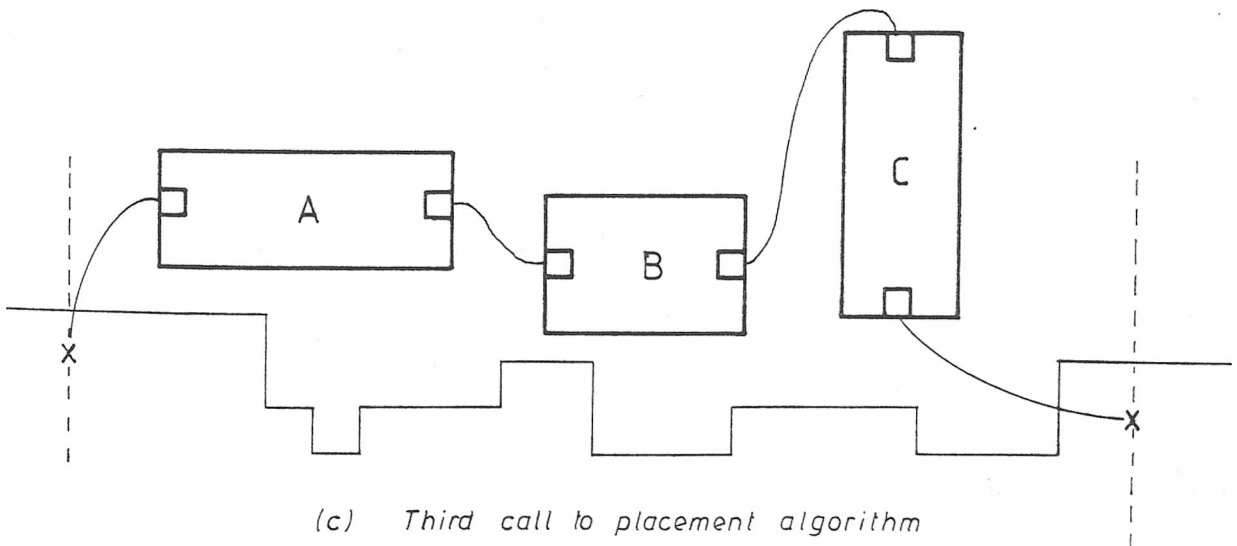
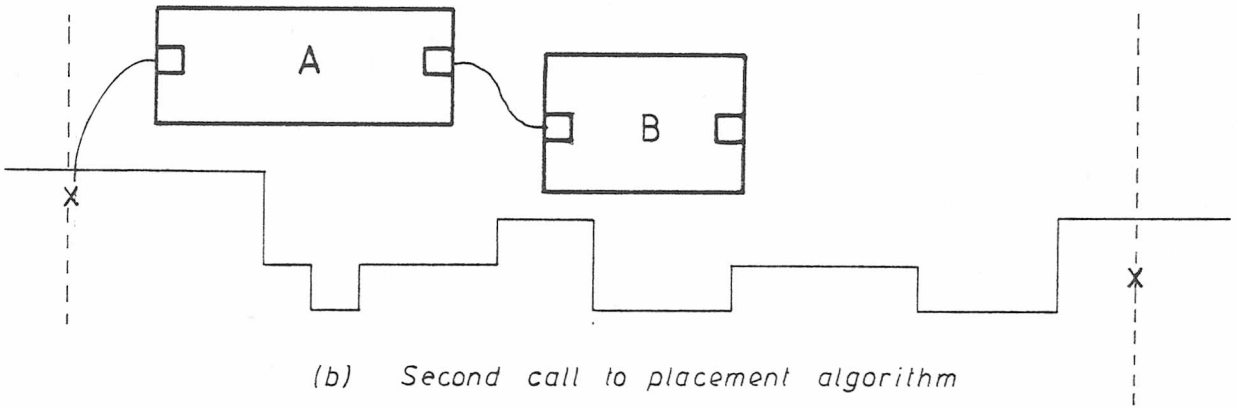
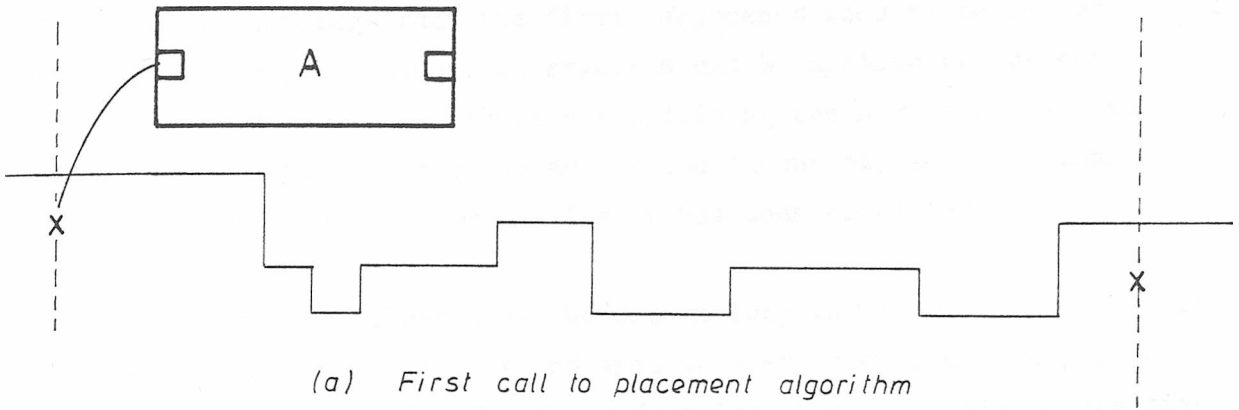


Figure 7.11 Rough placement

positioned satisfactorily, the user can let the process continue automatically, but it is important to give him the chance of vetting the layout at each step. Most Sequential Placement algorithms suffer from the disadvantage that the first components tend to be better placed than later ones. Manual interaction can be applied to correct this degradation at each stage thereby simplifying the next placement task. It is also generally easier to adjust the layout at each step than to attempt major changes after the layout has been completed.

Figure 7.12 illustrates the design loop involved. The process starts with an Edge Placement and ends when every component has been positioned on the board. Chapter 9 describes the available interactive facilities in detail and Appendix 1 demonstrates the Placement Algorithm as applied to a simple circuit.

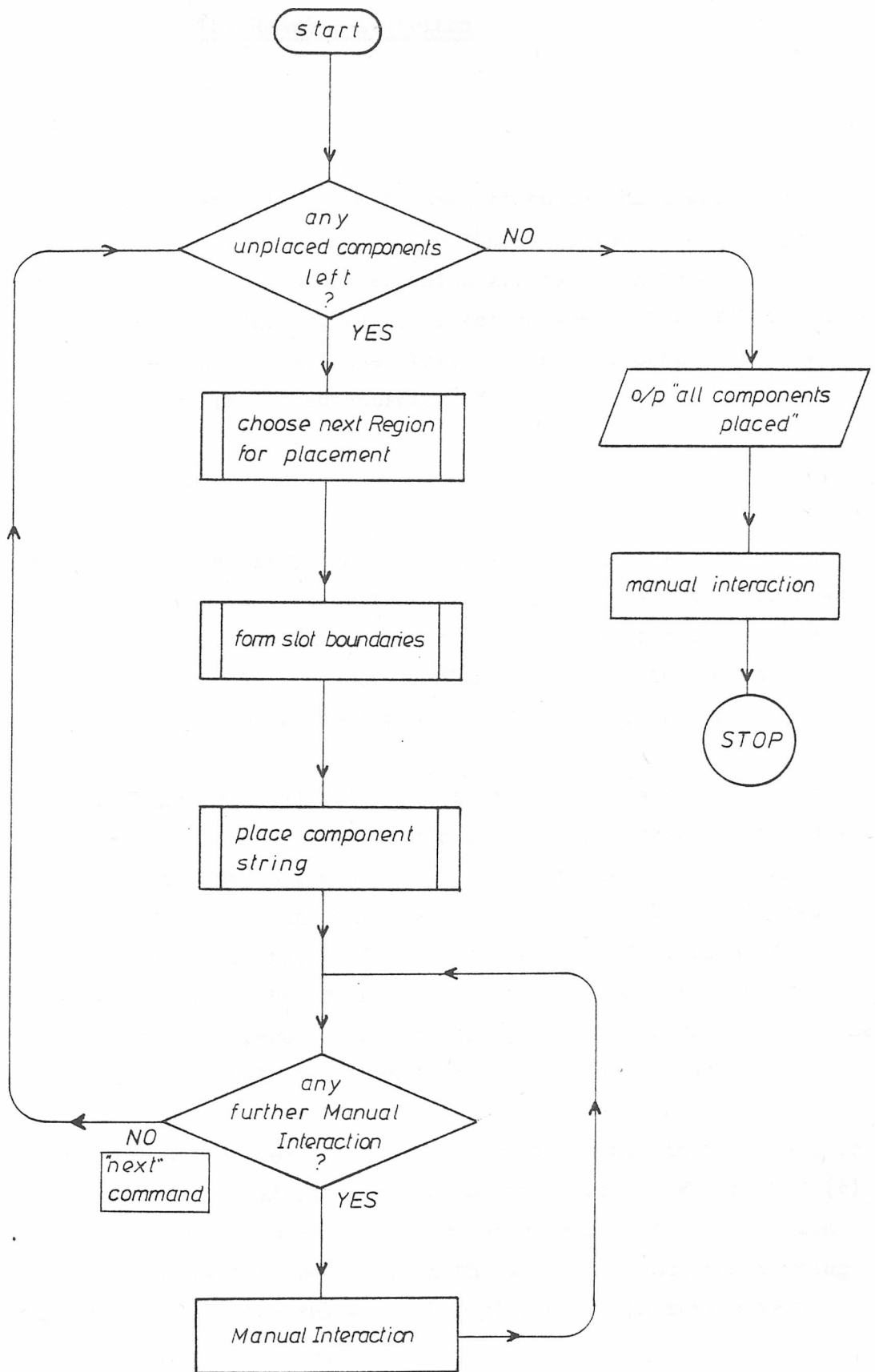


Figure 7.12 Using the Placement Algorithm

## Chapter 8

### The Routing Algorithm

#### 8.1 Definitions

A "ray optic" type algorithm is used to route the interconnections. Each conductor track is constrained to be the same width and must be orthogonal - i.e. made up of segments which are parallel to the X or Y axis. A connection is said to be generated between a "START" component and a "TARGET" component. All the other components, Edge Pads, Board Edges and connections are deemed "Obstacles".

#### 8.2 Basic Algorithm

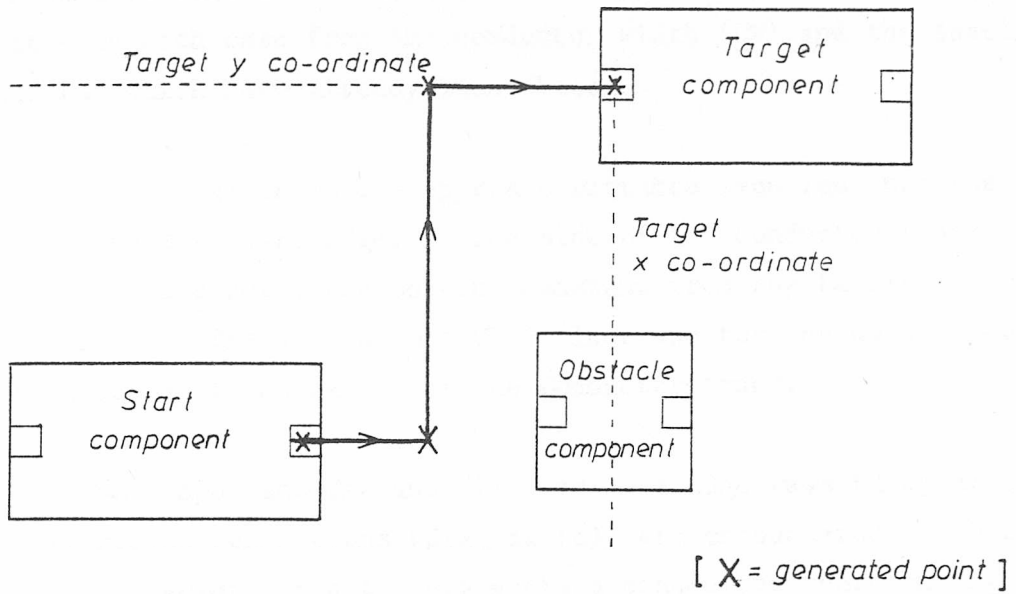
A "Ray" is emitted from the Outgoing Pad in the Start component at right angles to its bounding rectangle. When the Ray is obstructed or reaches the Target co-ordinate, it turns through 90 degrees to lie along the other axis. The connection "hunts" through the Board in this manner until it reaches the correct pad in the Target component.

Consider Figure 8.1 (a) which illustrates the basic idea of the algorithm. The Start pad is positioned on the right here, so the Ray must start in a horizontal orientation. An obstacle is encountered before the Target X co-ordinate can be reached, and the Ray is forced to change direction. It is, in fact, directed upwards because the Target Y co-ordinate is higher than the present value. This time it is possible to reach the required co-ordinate without obstruction and the Ray reverts to its original direction to reach the Target pad.

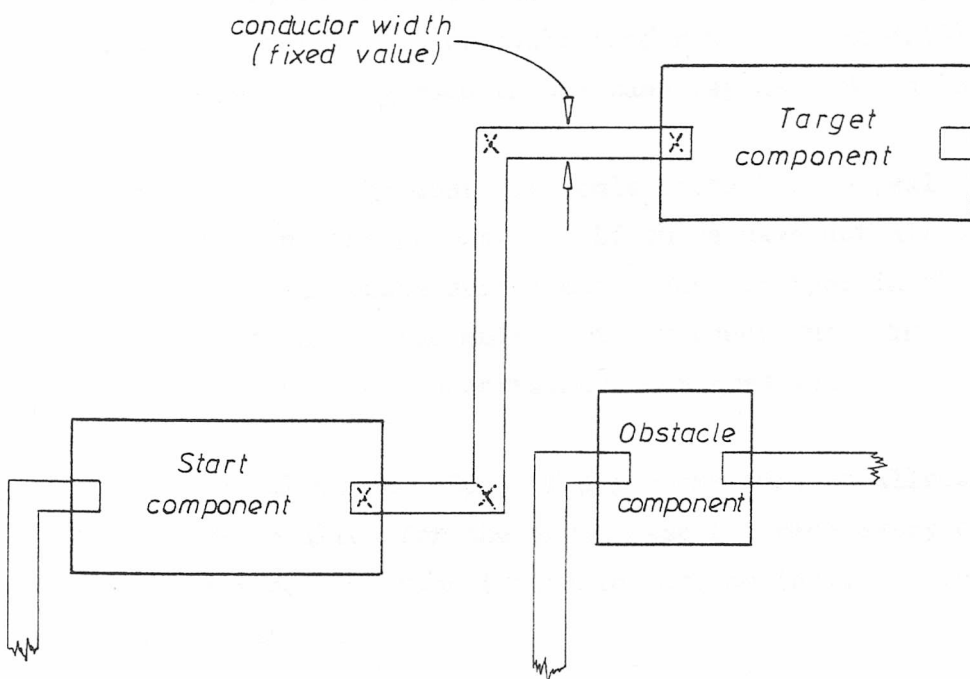
The Algorithm produces a string of turning points in this way, to be stored in a Connection Block in the data structure. Figure 8.1 (b) shows how these co-ordinates can be interpreted to draw the connection on the screen. Since the conductor width is pre-defined, the turning points completely define the shape of the polygon. This is a very economical method of data storage.

#### 8.3 Approach Distances

When the Ray encounters an obstacle of some kind it stops short of the



(a) Conceptual operation



(b) Graphic output

Figure 8.1 Basic routing algorithm

barrier by a certain amount known as the "Approach Distance". This is calculated in each case from the conductor width (CW) and the spacing constant (S) which have already been chosen.

Figure 8.2(a) shows the Approach Distance required when the connection meets a Board Edge. The side of the conductor track is positioned to be exactly one spacing constant from the barrier. This gives an Approach Distance of  $S + CW/2$  since the turning point co-ordinate is defined to be at the centre of the Conductor track.

The same Approach Distance is used when Edge Pads (diagram (b)) or previously routed connections (diagram (c)) are encountered. A special case arises, however, when the Ray meets a connection representing the same Node Number. The two paths are, by definition, at identical circuit voltages so they are allowed to intersect and merge freely.

The connection may sometimes meet the Slot Boundary but it will not always treat this as an obstacle (refer to section 8.10). When it does, the Boundary is regarded in the same way as a Board Edge.

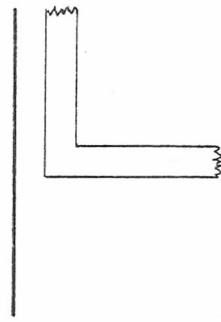
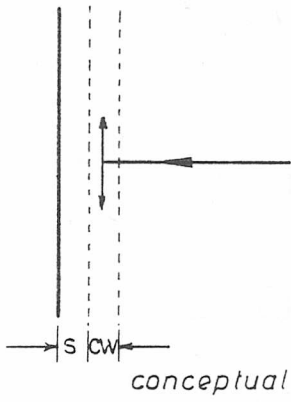
Components are the most difficult obstacles to deal with because they have their own connections. If these have not already been routed, the Algorithm must allocate sufficient space for them in the Approach Distance calculation. The multi-pad component shown in Figure 8.3 demonstrates just how much uncertainty this involves.

Every pad along the "Edge of Approach" must be allocated a channel of width  $S + CW$  to allow for the worst case in which every connection travels in the same direction (shown in diagram (a)). The spacing can thus be calculated from:

$$\text{Spacing} = N_{\text{pad}} (S + CW) + S + CW/2$$

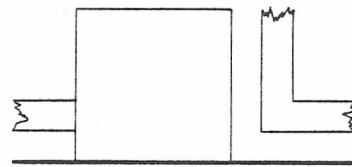
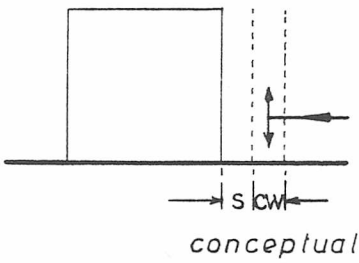
where  $N_{\text{pad}}$  is the number of pads along the edge in question. The Approach Distance can be smaller than this value, however, when the connections leave in opposite directions (diagram (b)) or from different edges (diagram (c)). It is impossible to take advantage of this until all the connections have actually been routed so the maximum spacing is always assumed. This is acceptable for components which have been placed automatically but may cause problems with those that have been





graphic o/p

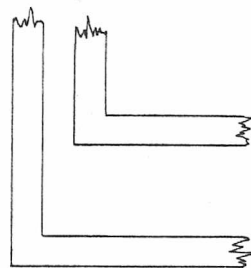
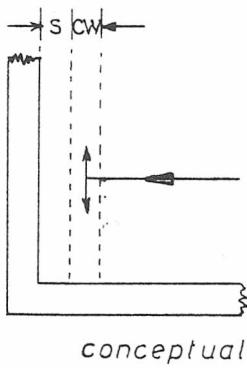
(a) Board edge



graphic o/p

S = spacing allowance  
CW = conductor width

(b) Edge pad



graphic o/p

(c) Connection with different node number

Figure 8.2 Approach distances

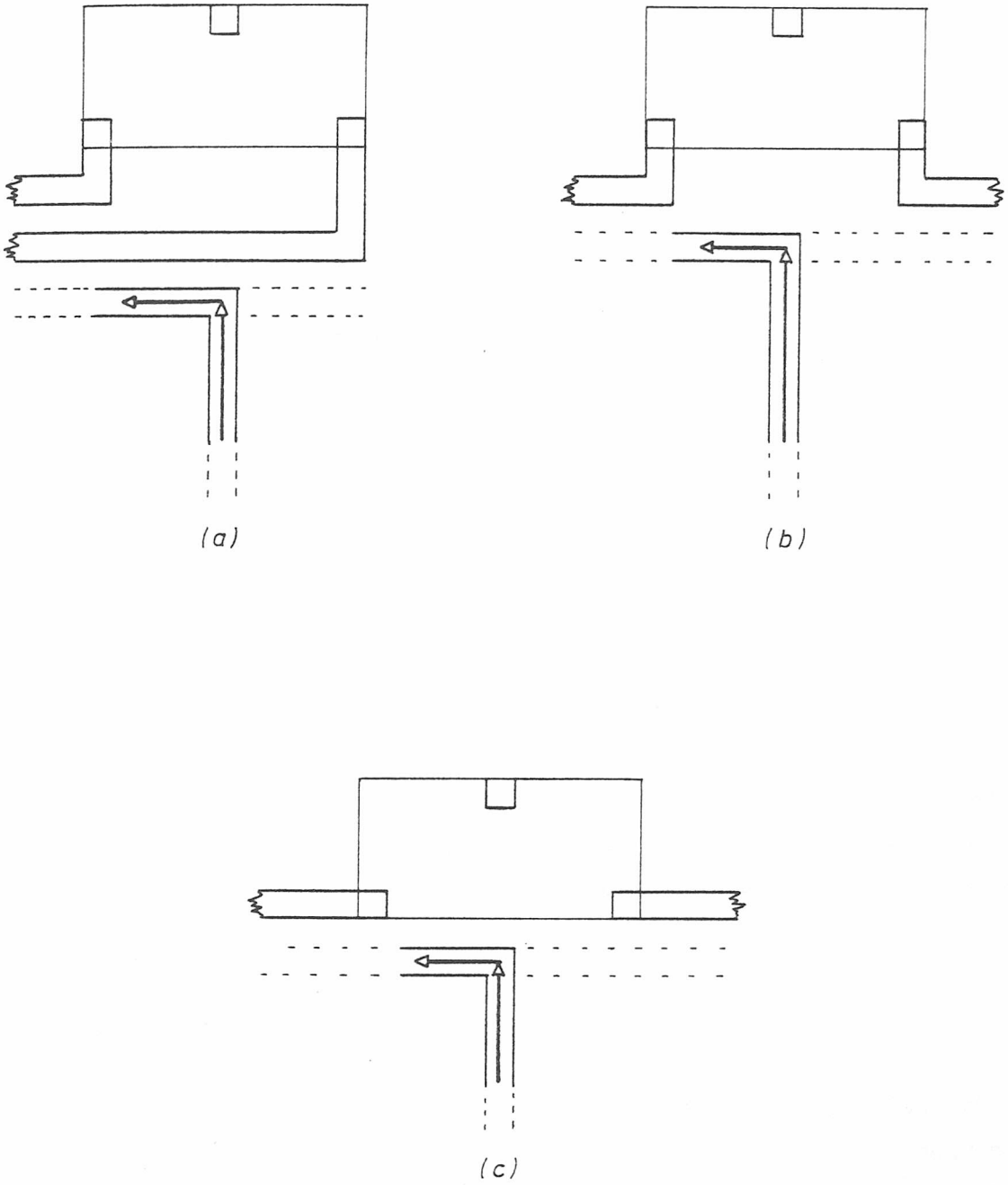


Figure 8.3 Possible approach distances to a component

moved interactively. All the Placement Algorithms set the spacings to their maximum values as a matter of course and it is up to the user to re-position components if he wishes to save space. Should he do so, however, the connection may fail due to lack of space. When this happens, the component connection allowances are temporarily relaxed until the track has been routed.

The technique of relaxing Approach Distances may cause some connection pads to be blocked unnecessarily. When this happens, the designer must use his experience to decide whether to move the obstacle or to re-route the offending connection by hand. The important point is that he is immediately aware of the problem since all connection failures are displayed at the conclusion of the routing process.

## 8.4 Initial Shunting

### 8.4.1 Introduction

A serious disadvantage of the basic algorithm is that the Rays are always sent out TOWARDS the Target pad. The rule is that Rays should be directed to decrease the difference in the X/Y co-ordinates at each step. This will certainly produce minimum length connections but there are many situations when this strategy will simply not work. A number of specialised techniques have had to be included with the algorithm in order to cope with these potential failure conditions, and the first of these is called "Initial Shunting".

### 8.4.2 Single Shunt

Figure 8.4 shows the simplest example where a single "shunt" is required. The Start pad is positioned on the Left of the bounding rectangle here, so the initial connection segment must be horizontal. The algorithm would try to go Right in order to reduce the distance between the two X co-ordinates. This is quite impossible in this case, so the conductor track is shunted to the left before the algorithm is applied.

### 8.4.3 Double Shunts

Two shunts may be necessary in some cases and this introduces a

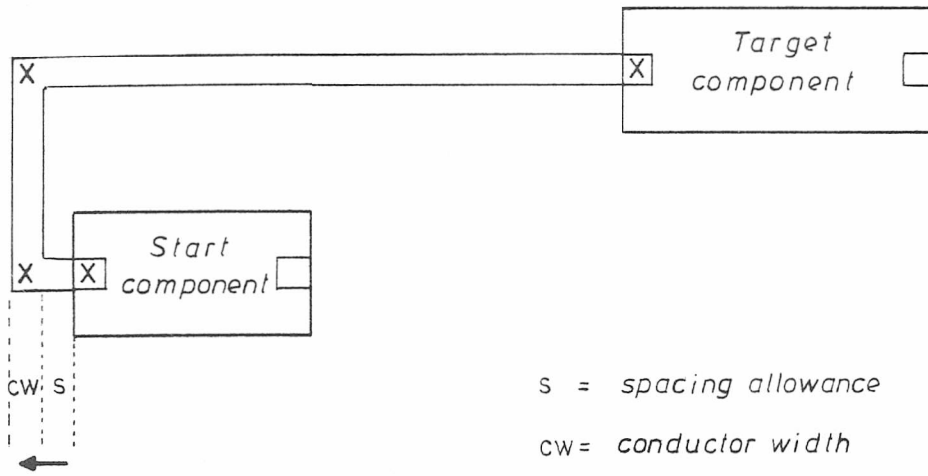
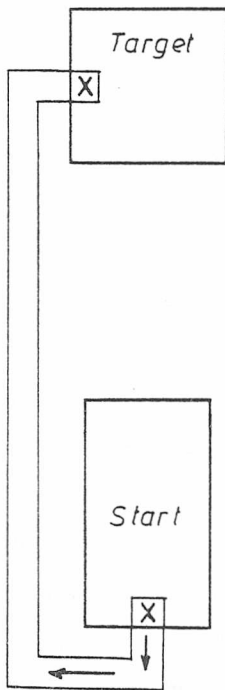
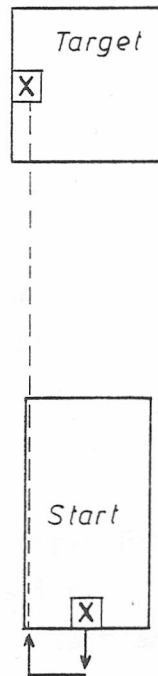


Figure 8.4 Single shunt case (to the left)



(a) Double shunt clockwise  
(down, left)



(b) Single shunt  
(down)

Figure 8.5 Double shunt case

variable into the algorithm - whether to go clockwise or anticlockwise round the component. Figure 8.5 (a) illustrates the former where a Downwards shunt has been followed by a Left Shunt. If the second shunt had been to the Right instead of to the Left, this would have been deemed an anticlockwise case.

In general it is not clear which direction is to be preferred until both have been tried and the results compared.

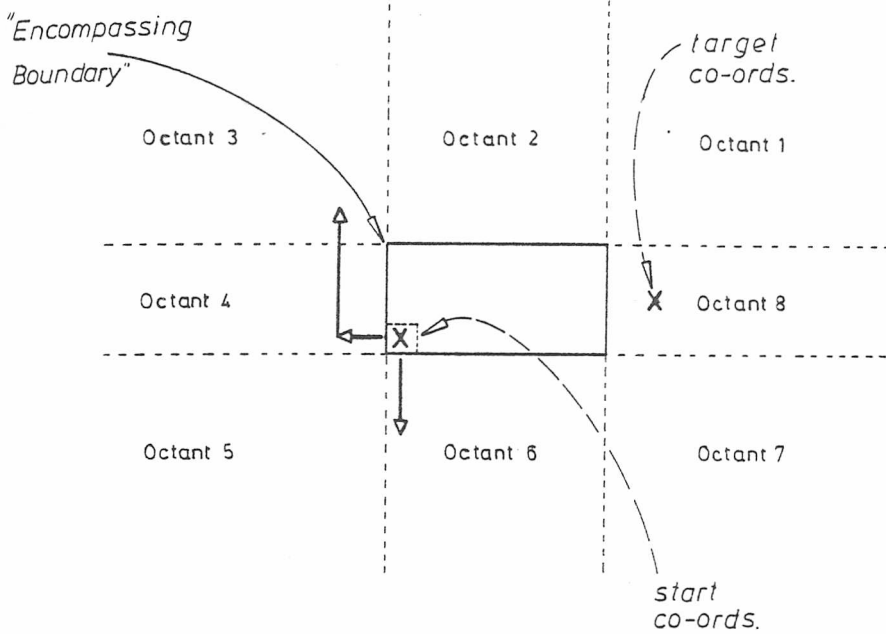
Figure 8.5 (b) justifies the use of the second shunt. In this case the Left shunt has been omitted causing the Algorithm to fail. After reaching the Target X co-ordinate, the vertical Ray was still obstructed by the Start component.

#### 8.4.4 Shunt Selection

A set of rules have been formulated to govern which shunt (or shunts) should be applied in every possible situation. These are listed in Figure 8.6.

The first step is to define an "Encompassing Boundary" round the Start Component. This is initially set to be the Bounding Rectangle. Each pad is then examined in turn and the relevant X or Y co-ordinate is adjusted in the Encompassing Boundary definition to allow for connections. When this has been completed, the four boundary lines are extended to define eight sections or "Octants" round the component (see Diagram (a)). The Target Pad is classified into one of these Octants and it is then possible to read the necessary shunting requirements from the table given in Diagram (b).

An example is shown in (a) where the Target Pad is in Octant 8 and the Start Pad is in the BL (Bottom Left) position. The connection can either be given a single shunt downwards, or a left shunt followed by an upward shunt. These two alternatives are given in the table and represent Anticlockwise and Clockwise solutions respectively. All the corner pad positions produce two alternatives since the connection may start parallel to either board axis. Even if no shunting is required, the connection is attempted using both initial directions to find the best of the two.



(a) Definition of the 8 octants round a component

$TL = \text{top left}$        $T = \text{top}$        $TR = \text{top right}$        $R = \text{right}$   
 $BR = \text{bottom right}$        $B = \text{bottom}$        $BL = \text{bottom left}$        $L = \text{left}$   
 $U = \text{up}$        $D = \text{down}$        $+ = \text{"followed by"}$

		start pad position							
		TL	T	TR	R	BR	B	BL	L
target pad octant	1						D	$\frac{L}{D}$	L
	2				R	$\frac{R}{D+L}$	$\frac{D+L}{D+R}$	$\frac{L}{D+R}$	L
	3				R	$\frac{D}{R}$	D		
	4		U	$\frac{R+D}{U}$	$\frac{R+D}{R+U}$	$\frac{D}{R+U}$	D		
	5		U	$\frac{R}{U}$	R				
	6	$\frac{U+R}{L}$	$\frac{U+R}{U+L}$	$\frac{R}{U+L}$	R				L
	7	$\frac{U}{L}$	U						L
	8	$\frac{U}{L+D}$	U				D	$\frac{L+U}{D}$	$\frac{L+U}{L+D}$

Where two alternatives are given, the upper refers to a CLOCKWISE shunt, and the lower to an ANTICLOCKWISE shunt.

(b) Table of shunts required in all situations

Figure 8.6 Shunt selection

Only half of the possible 64 situations warrant a shunt of any kind.

#### 8.4.5 Shunt Distances

Having chosen the shunt direction, the actual distance which the Ray should travel is calculated from the positioning of the other connector pads round the component rectangle. This is again best illustrated by example. Consider Figure 8.7 which shows a set of possible LEFT shunts from a multi-pin component.

The Target Y co-ordinate is higher than the Start Y co-ordinate in this case so room must be left for connections going to pads immediately above the Start point. A general rule is that the shunt distance be:  
 $\text{dist} = (\text{Npad} + 1) (S + CW)$  where Npad is the number of pads along the same component edge between the Start and Target.

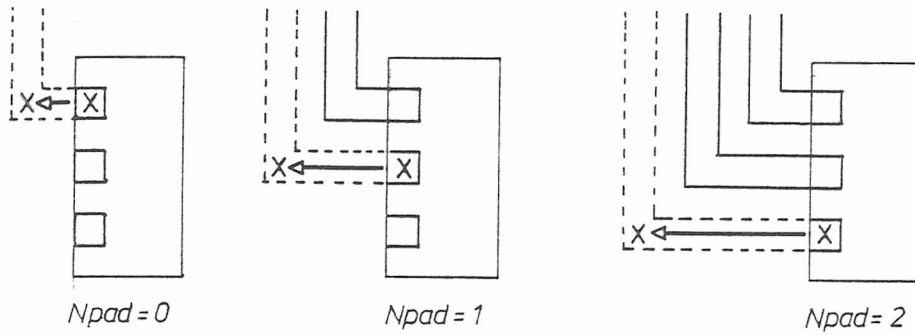
The double shunt situation is rather more complicated. This time the second shunt must allow for connections along TWO sides of the component - Figure 8.8 shows this graphically. The rule here is that the second shunt should continue past the component edge in question ( $Y_{\text{max}}$  in this example) by a distance defined as:  
 $\text{dist} = (\text{Npad} + 1) (S + CW)$   
where Npad is the number of pads along BOTH edges between the Start and Target points. The first shunt is treated in the same way as before.

#### 8.5 Final Shunting

The basic Algorithm has the disadvantage that the Ray must strike the component on the edge bearing the Target pad. If it hits any other edge, the connection will fail since the Ray is blocked by the Target component itself (see Figure 8.9).

A series of up to three "Final Shunts" are applied to route the connection round the component in order to reach the correct edge. This can be done in a Clockwise or Anticlockwise direction and it is not clear which is to be preferred until both have been tried.

The shunt distances are calculated in a similar manner to the



(a)  $Dist = S + CW$

(b)  $Dist = 2(S + CW)$

(c)  $Dist = 3(S + CW)$

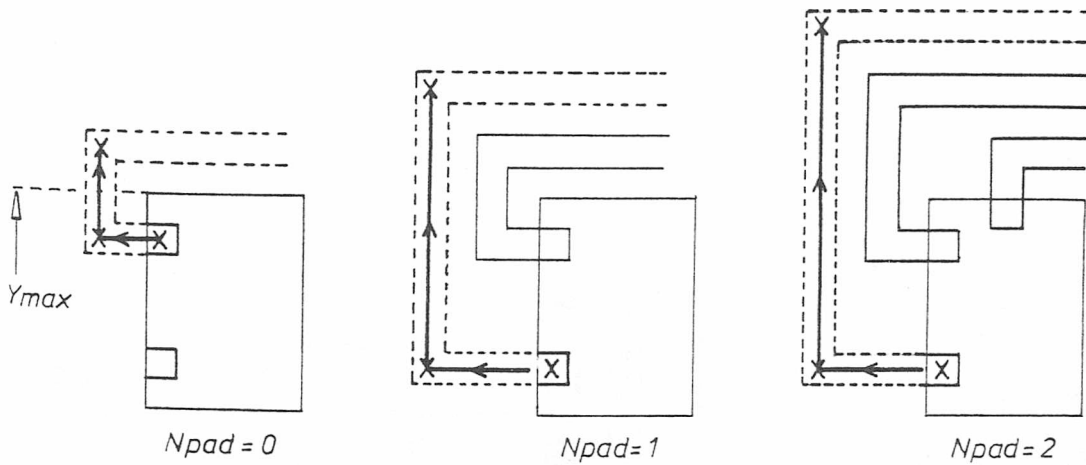
$S$  = spacing allowance

$CW$  = conductor width

$Dist$  = shunt distance from edge

$Npad$  = number of pads in shunt direction

Figure 8.7 Single shunt distances



(a)  $NewY = Y_{max} + S + CW$

(b)  $NewY = Y_{max} + \underline{2(S + CW)}$

(c)  $NewY = Y_{max} + \underline{3(S + CW)}$

Figure 8.8 Double shunt distances



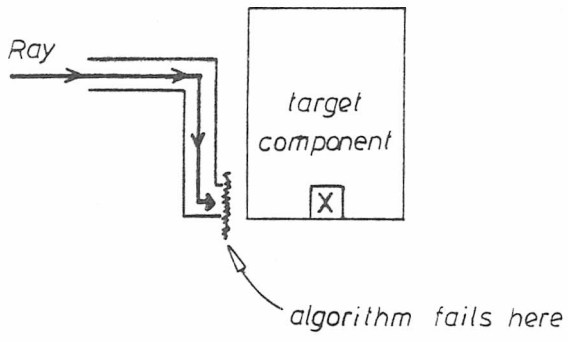
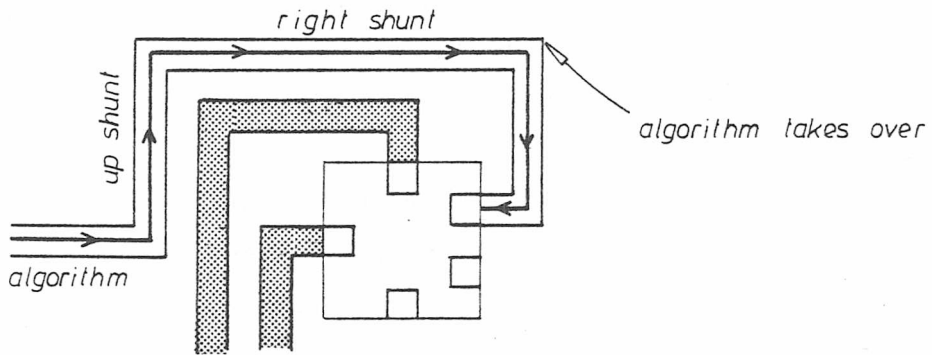
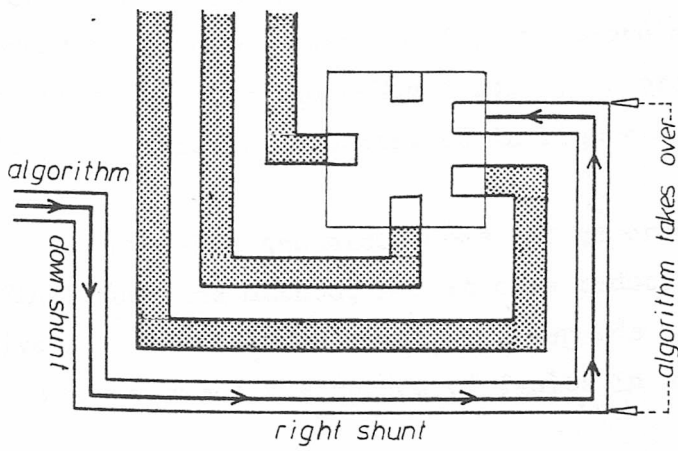


Figure 8.9 Connection fails to end without final shunting



(a) clockwise



(b) anticlockwise

Figure 8.10 Final shunting

Initial Shunt technique except that pads along any of the three sides must be taken into account. Figure 8.10 shows an example of this method.

## 8.6 Under-Run

Final Shunting can sometimes be avoided using a technique known as "Under-Run". The Ray is said to have Under-Run the Target co-ordinate if it prematurely changes direction without encountering an obstacle. This method is used instead of Final Shunting wherever possible because it produces shorter, less complex connections.

The two methods can be compared with reference to Figure 8.11. In this example the Target pad is situated on the lower edge of the component and the Ray is initially directed upwards. Under-run can only be applied when the nearest component edge to the Ray is also the edge containing the Target pad. This is obviously the case here, so the Ray is deflected before the Target Y co-ordinate is reached. The actual magnitude of the Under-Run is calculated from the pad positions along the Target pad edge and is given by:

$$\text{dist} = N_{\text{pad}} (S + CW) + S + P/2$$

where  $N_{\text{pad}}$  is the number of pads between the Ray and Target co-ordinate and  $P$  is the component's pad size.

## 8.7 Over Run

When the Target edge is perpendicular to the Ray, but on the far side of the component, an amount of "Over-Run" can be applied. The Ray is forced to continue past the Target co-ordinate by a specified amount (calculated from the Under-Run formula) in order to eliminate Final Shunting. This is demonstrated in Figure 8.12.

The Over-Run technique does not generate shorter connections than the Final Shunting method, but it does reduce the number of turning points involved. This is important as it speeds up the artwork generation process and reduces the chance of faults in manufacture.

## 8.8 Extrapolation

We have seen that when the Ray meets an obstacle, it immediately

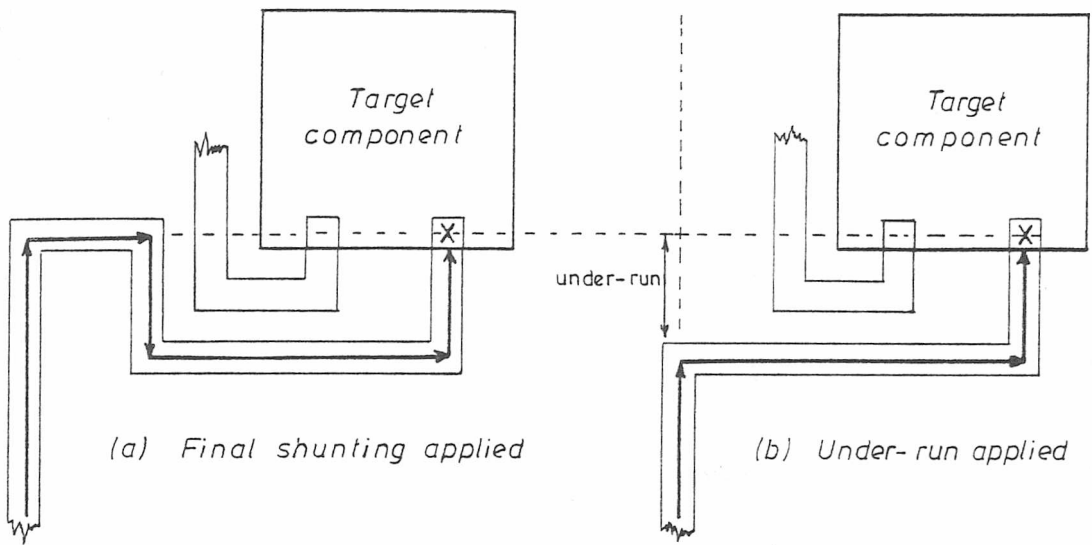


Figure 8-11 The "Under-run" technique

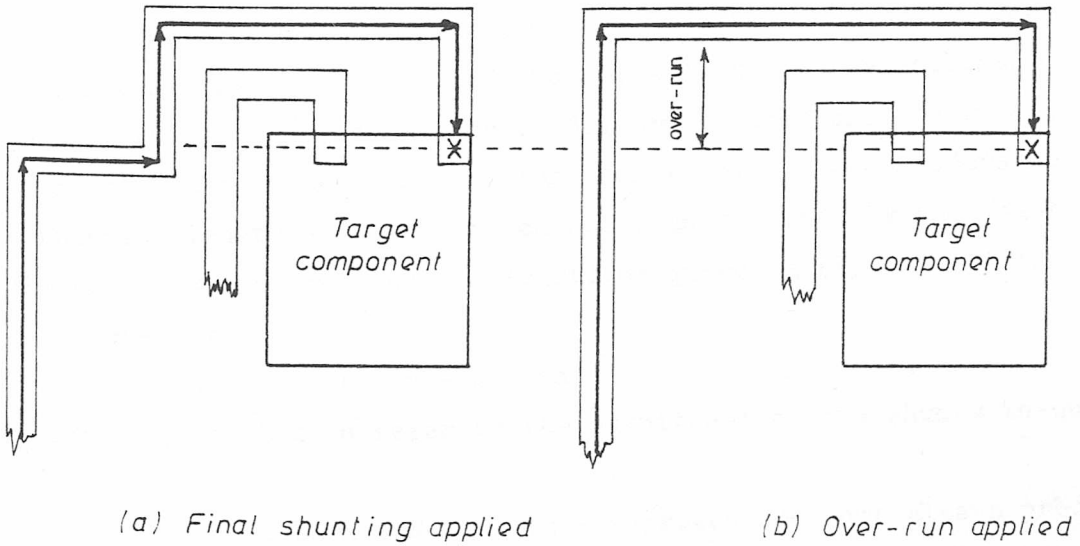


Figure 8-12 The "Over-run" technique

changes direction to run parallel with it. Should the Target co-ordinate be reached before the obstacle is cleared (allowing for Under-Run and Over-Run), the Ray will still be blocked and the Algorithm will fail. This can be avoided if the Ray is shunted or "Extrapolated" to the end of the obstruction as demonstrated in Figure 8.13. The Ray continues past the Target co-ordinate until it has cleared the obstacle and any associated connection space then the Algorithm is re-applied.

## 8.9 Intermediate Shunting

This technique is used when the Ray has reached one of the Target co-ordinates and is then prevented from reaching the other. Figure 8.14(a) gives a typical example of this where a horizontal ray at the correct height has been obstructed by a component.

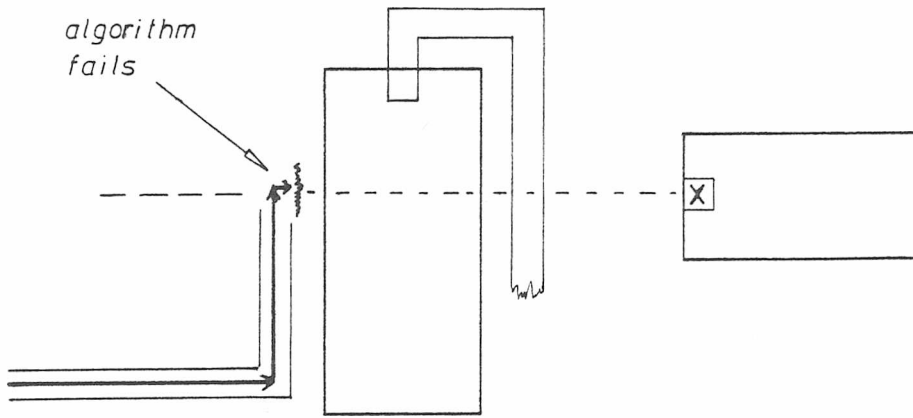
An arbitrary choice must be made here - whether to shunt the Ray upwards or downwards. The former strategy will route the connection round the obstacle in a clockwise manner, while the latter will route it in an anti-clockwise direction. These two alternatives are illustrated in diagrams (b) and (c) respectively.

The shunt magnitudes are chosen such that the new conductor track will not interfere with any of the obstacle's own connections. In general the two shunts will be of different sizes and the rule is that the shorter should be selected. This can result in a considerable saving in connection length since the track has been diverted by the least amount. The actual saving between the solutions given in (b) and (c) can be calculated from:

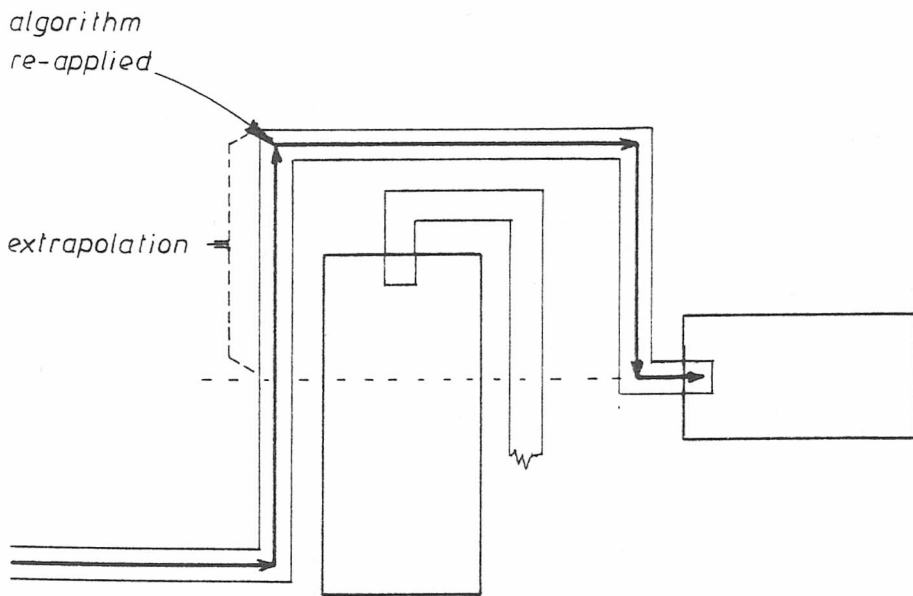
$$\text{Saving} = 2 (S \text{ up} - S \text{ down})$$

where "S up" and "S down" refer to the magnitudes of the shunts in question.

It should be noted that this approach does not always produce the shortest connection since the effect of subsequent obstacles may cancel out this saving. Another disadvantage is that the spacings round the obstruction are normally set such that one direction will fail. If this happens to be the direction which has been chosen, then the connection attempt will fail even though it is possible to route round the component in the other direction.

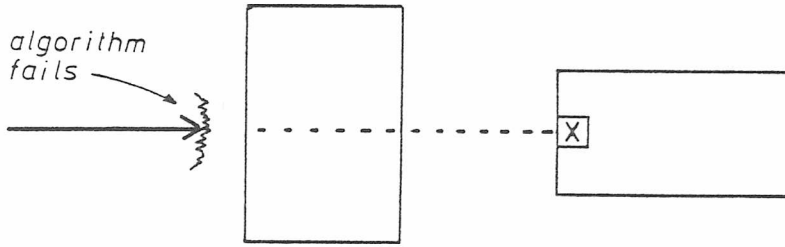


(a) No extrapolation

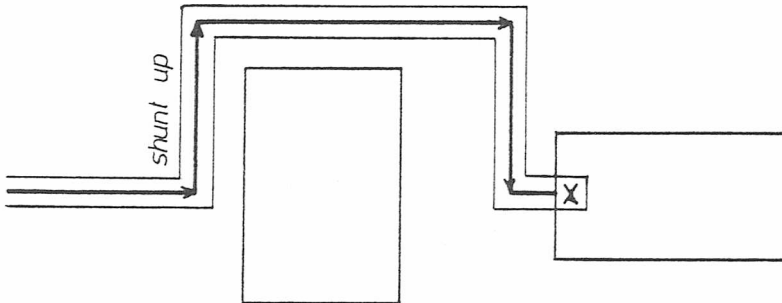


(b) Extrapolation applied

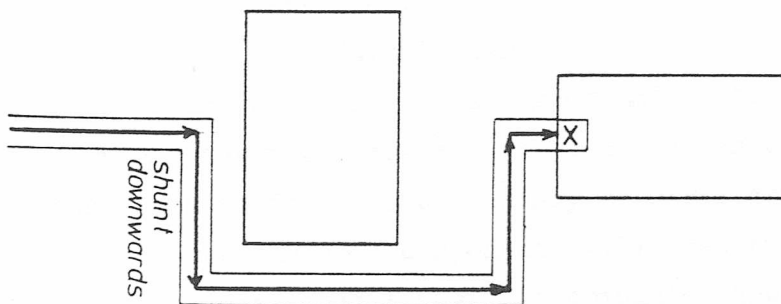
Figure 8.13 Extrapolation technique



(a) Basic algorithm fails because Y co-ordinate already attained



(b) Ray shunted upwards (clockwise)



(c) Ray shunted downwards (anti-clockwise)

Figure 8.14 Intermediate shunting

A safer technique would be to try each direction and then select the successful or shorter solution but this would slow the routing process considerably as there may be any number of such decisions. The Routing Algorithm already incorporates a large number of arbitrary choices and alternatives. Results suggest that these will suffice to produce a solution in the majority of cases. When a connection does fail, the user must interactively route the track by hand.

#### 8.10 Following The Slot Boundary

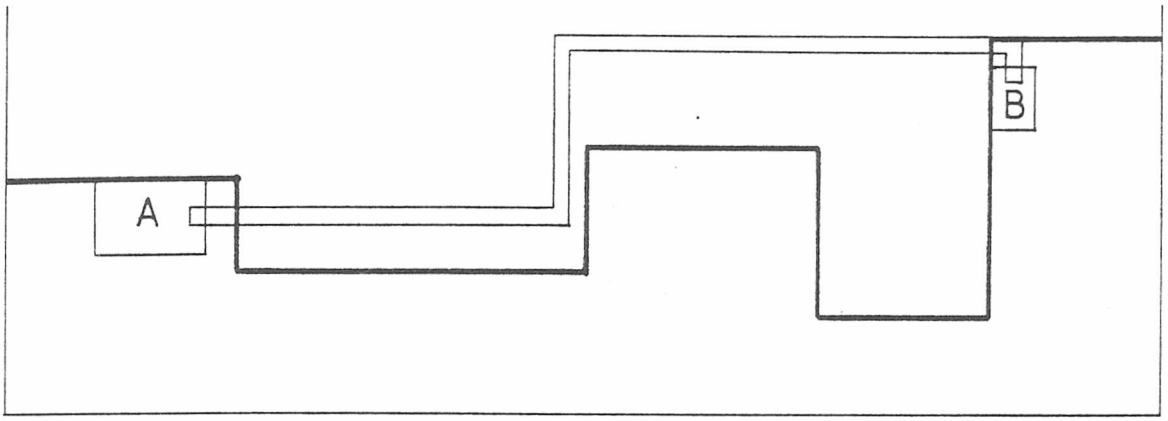
The Routing Algorithm can be applied at any stage in the design but it is best to connect up the components as soon as they appear on the board. There are two good reasons for adopting this approach:

- (a) If a connection fails it must be due to the most recent components which have been placed along the top edge of the existing layout. It is much easier to re-position them at this stage while they are easy to access, than to wait until they have become imbedded within subsequent circuitry. In the latter case it may be necessary to move a great many other components in order to get at the string in question.
- (b) The Slot Boundary can be used in conjunction with the Routing Algorithm to produce a much higher packing density than would otherwise be achieved.

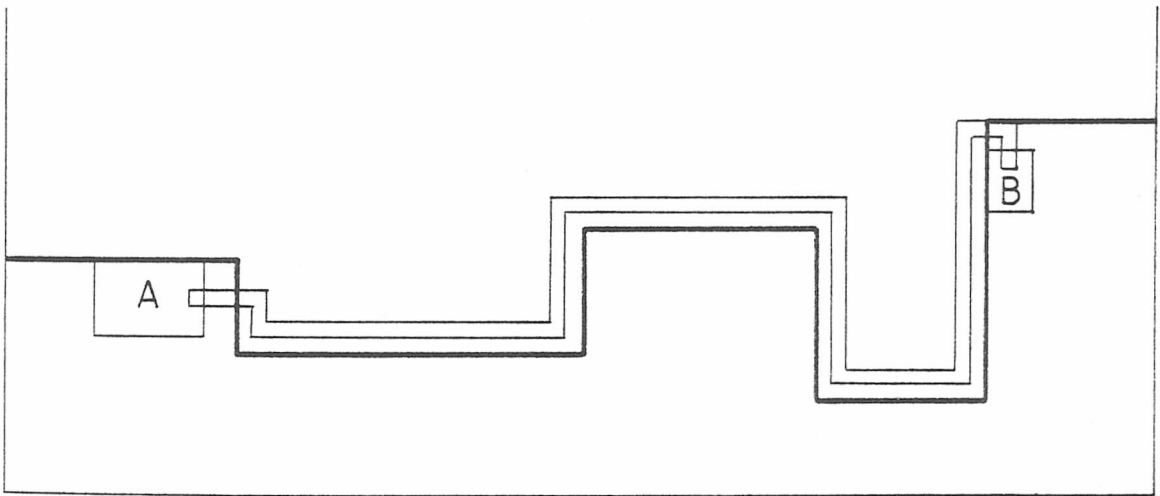
The second reason can be justified with reference to Figure 8.15 which shows the results obtained in a typical situation.

The connection illustrated in Diagram (a) was produced using the normal Routing Algorithm. This is certainly the shortest solution but it wastes a large amount of board space corresponding to the gaps between the Slot Boundary and the track edges. The space can be conserved by forcing the connection to follow the Boundary wherever possible as shown in Diagram (b). Since the next components will have the opportunity of using this extra space, it follows that the complete layout should fit into a smaller board area.

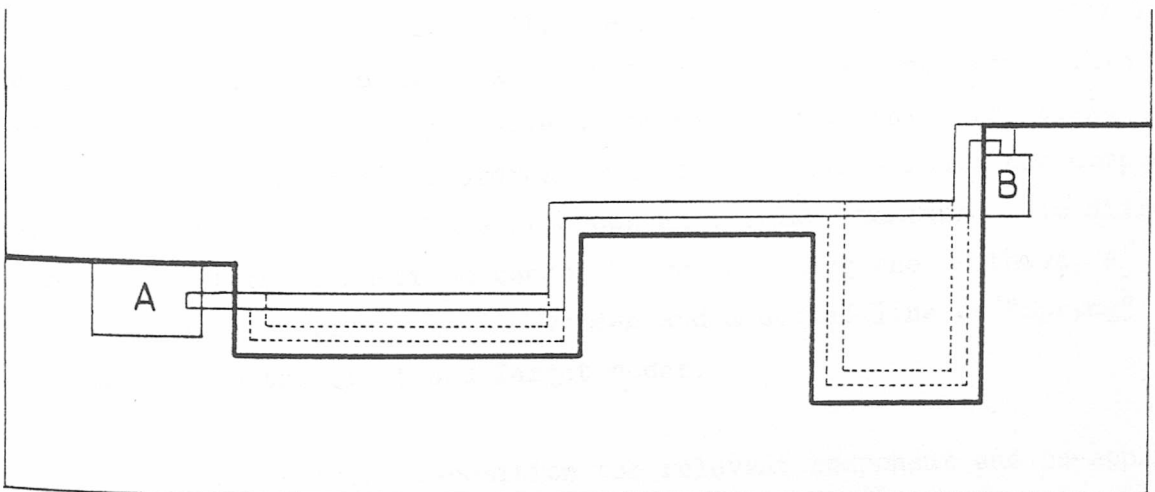
The increased packing density is offset by the fact that the



(a) Normal algorithm



(b) Following the boundary



(c) Solution (b) after connection reduction

Figure 8.15 Using the slot boundary to assist the routing algorithm



connection is rather longer and changes direction more frequently than before. In general it is better to sacrifice connection length to produce a compact layout but there are obviously cases where the extra space is never actually used. The Connection Shortening Algorithm (described in Chapter 9.3.22) must therefore be employed at the end of the design to ensure that every connection is in its shortest possible state. Diagram (c) shows the effect of this on the given example, assuming that the extra space was not required. The connection is just as short as the version obtained using the normal algorithm but incorporates a greater number of turning points. This is not a serious problem considering the potential space that can be saved. The user has the opportunity to alter the connection interactively in any case.

### 8.11 Using The Algorithm

The Algorithm does not make an exhaustive search for every possible connection route so there is a finite chance that it will fail in any given situation. Initial results suggest that the percentage failure rate is acceptably low and is confined to the longer, more complex, connections. The emphasis here is on speed of execution rather than 100% routeability

Any particular connection may require a number of attempts before the best solution can be selected. The flowchart shown in Figure 8.16 shows how these alternatives can be generated.

Only one attempt is necessary should the Basic Algorithm be successful. Initial and Final Shunting can increase this by three, corresponding to clockwise and anti-clockwise selection at either or both ends of the connection. If the Algorithm is still unsuccessful, the "Slot" Boundary is neglected and the whole process repeated - this gives a total of  $2 \times 4 = 8$  possible attempts. When this also fails there is the option of disregarding connection spacings round the components (refer to Section 8.3). This produces an absolute maximum of 16 different attempts. If the connection cannot be routed using one of these, a failure message is written to the screen and a dotted line or "Spring" is drawn between the Start and Target nodes.

The user can then re-position the relevant component and re-apply the Algorithm. He also has the option of routing the Connection manually.

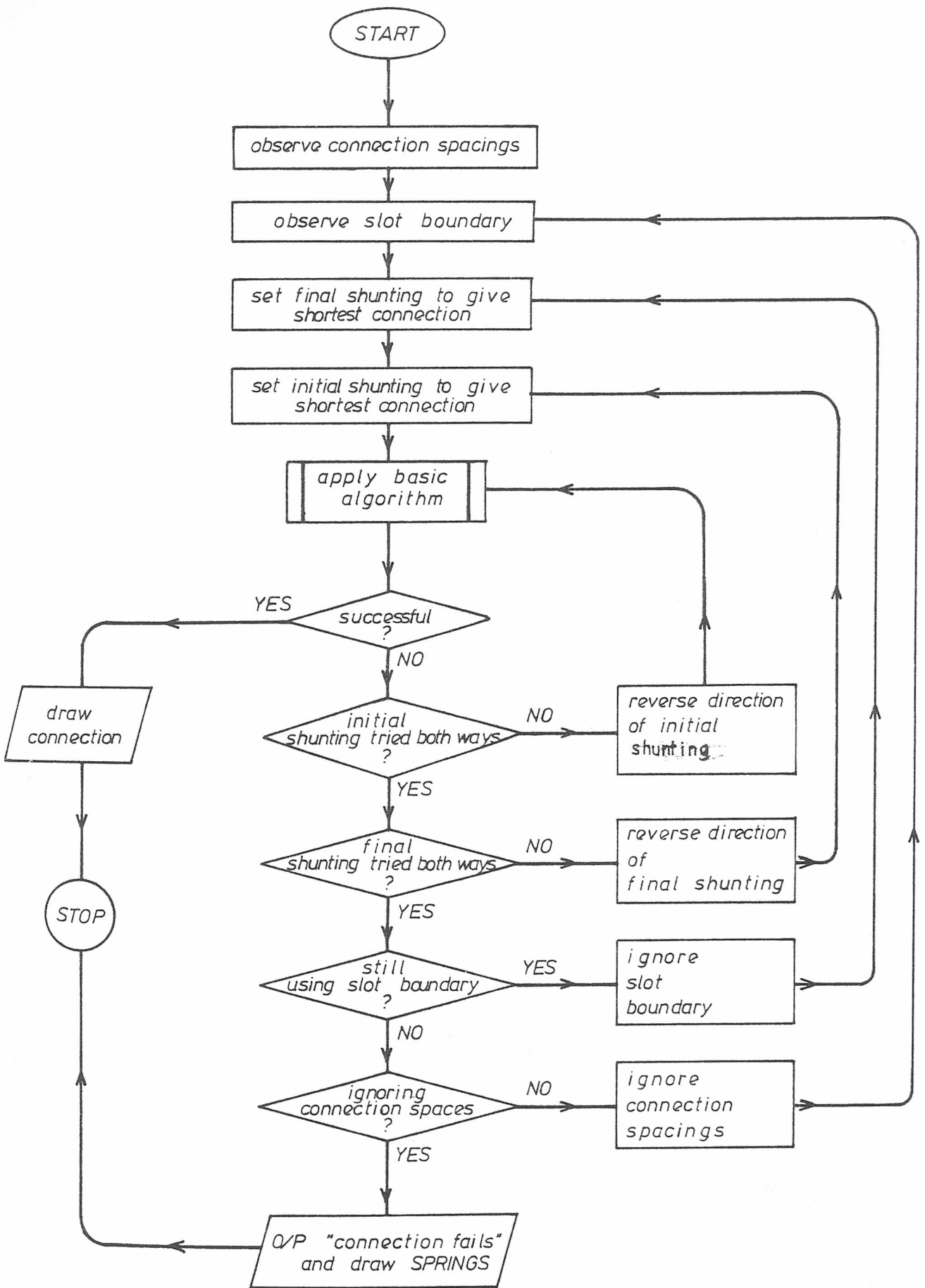


Figure 8.16 The Routing Algorithm

In general the Algorithm is quite fast - the actual speed is determined by the number of attempts which need to be made. It is unfortunate that there is no way of recognising potentially un-routable connections without using the full 16 variations.

The Algorithm has only recently been developed so there has been little time for rigorous testing. There are also several other techniques which could be added but their effect on speed and reliability can only be guessed at this stage. They are described in Chapter 11.

## Chapter 9

### Interactive Design Program - LAYOUT

#### 9.1 Introduction

The data structure generated by the Initial Placement program is now used by the main interactive design program, which is called LAYOUT. The latter incorporates both the Placement and Routing algorithms discussed in Chapters 7 and 8, and the user can apply them at any stage. In addition, a number of manual intervention facilities allow him to adjust the layout interactively.

The process can be suspended at any time and the current board geometry written to the same or another data file. A sequence of files representing the layout at various stages can be built up in this way. These enable the user to abort his present layout attempt without having to restart from scratch.

#### 9.2 Command Modes

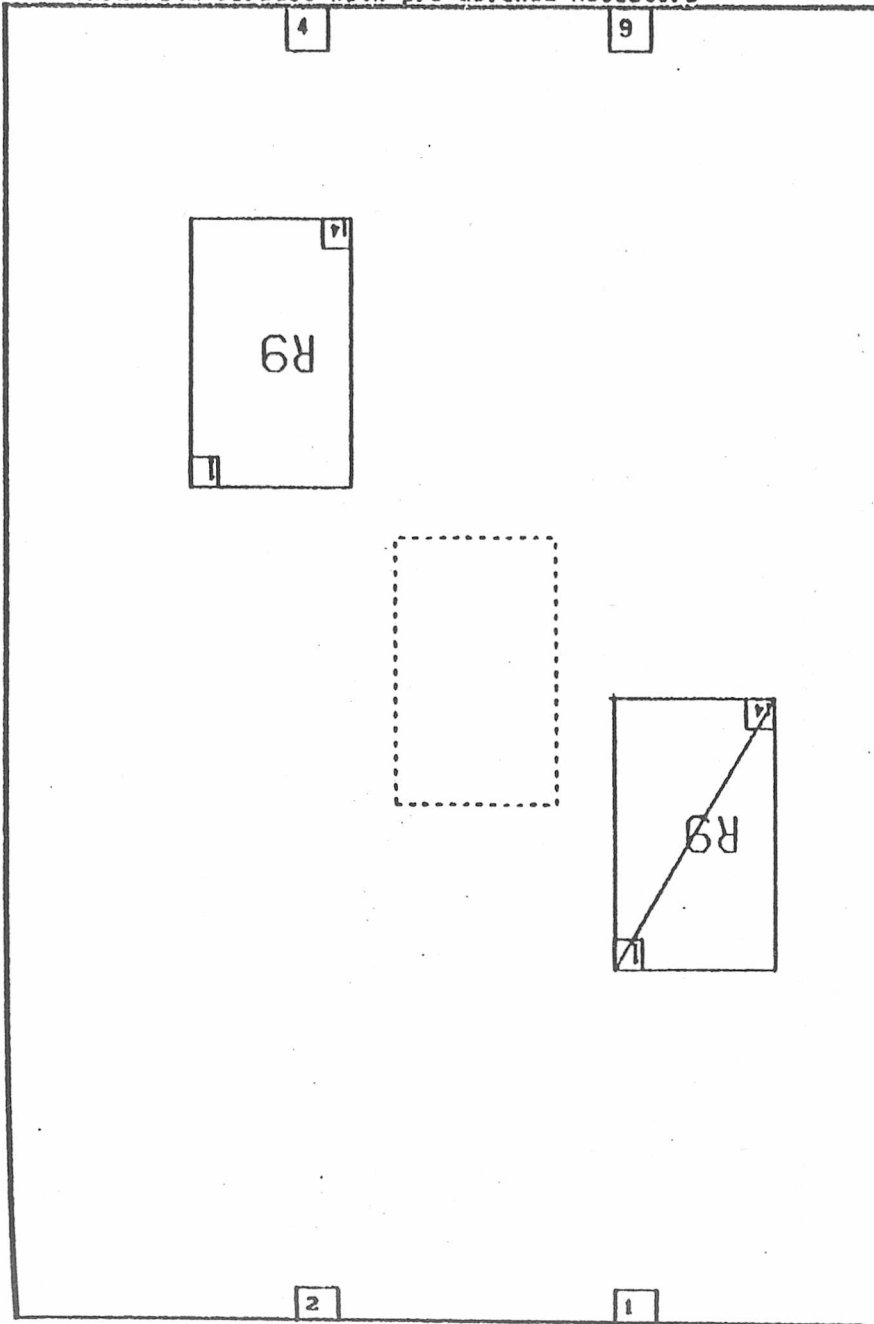
##### 9.2.1 Menu Mode

There are two distinct methods of controlling the program flow. These are called "Menu Mode" and "Cursor Mode". The default is the former, and refers to a section of display area down the right hand edge of the screen called the "Menu".

Figure 9.1 gives an example of the use of Menu Mode to move a placed component to a new position. The title of the circuit (specified in the original description file) is displayed along the top of the screen, and the message "What Next ?" written at the top of the Menu.

Any command string typed in via the terminal keyboard will automatically appear after the Command Prompt "#" in the menu area. In the example given, the code-word "MOVE" has been input causing the message "Indicate Component and New Position" to be generated. The Command Prompt is used to distinguish between command strings and program messages.

Thin Film Circuit with pre-defined Resistors



12

What Next ?

#MOVE

Indicate Component

And New Position

What Next ?



15

Figure 9.1 Example of "Menu Mode"

There is only room for 34 lines of text in the Menu Area, so the layout must be completely redrawn at intervals to allow the output of more text. This is a completely automatic process in that the program initiates the redraw facility when the number of remaining free lines is insufficient for the command type requested.

A comprehensive range of operating commands is given in Figure 9.2. This list can be displayed on the screen by typing the code-word "HELP" in Menu-Mode. The information is stored in a simple text file called LAYOUT.HLP, so it can be listed on the lineprinter to provide a reference document.

### 9.2.2 Cursor Mode

The time taken to redraw a layout is related to its size and complexity. In any event it is better to avoid this time-consuming process as much as possible. One way of doing this is to restrict the use of the menu area to an absolute minimum, thus reducing the frequency with which it overflows. An alternative method of program control has been added with this philosophy in mind. It is called "Cursor Mode".

When "CURSOR" (or simply "CUR") is typed in the menu area, a string of "Facility Boxes" are drawn along the top of the screen. Each box contains a verbal description of the function it represents and can be selected using the X/Y cursor.

Since all the boxes are positioned at the same height on the screen, it is sufficient to ensure that the vertical cursor line intersects with the box in question. In Figure 9.3, for example, the "WINDOW" option would be selected when a character is typed.

The characters inside the boxes are hardware generated and are written as a continuous line of text. Since the terminal is normally working at 120 characters per second, the extra time incurred when redrawing the screen is negligible. The boxes take slightly longer to draw but are ergonomically pleasing to the user.

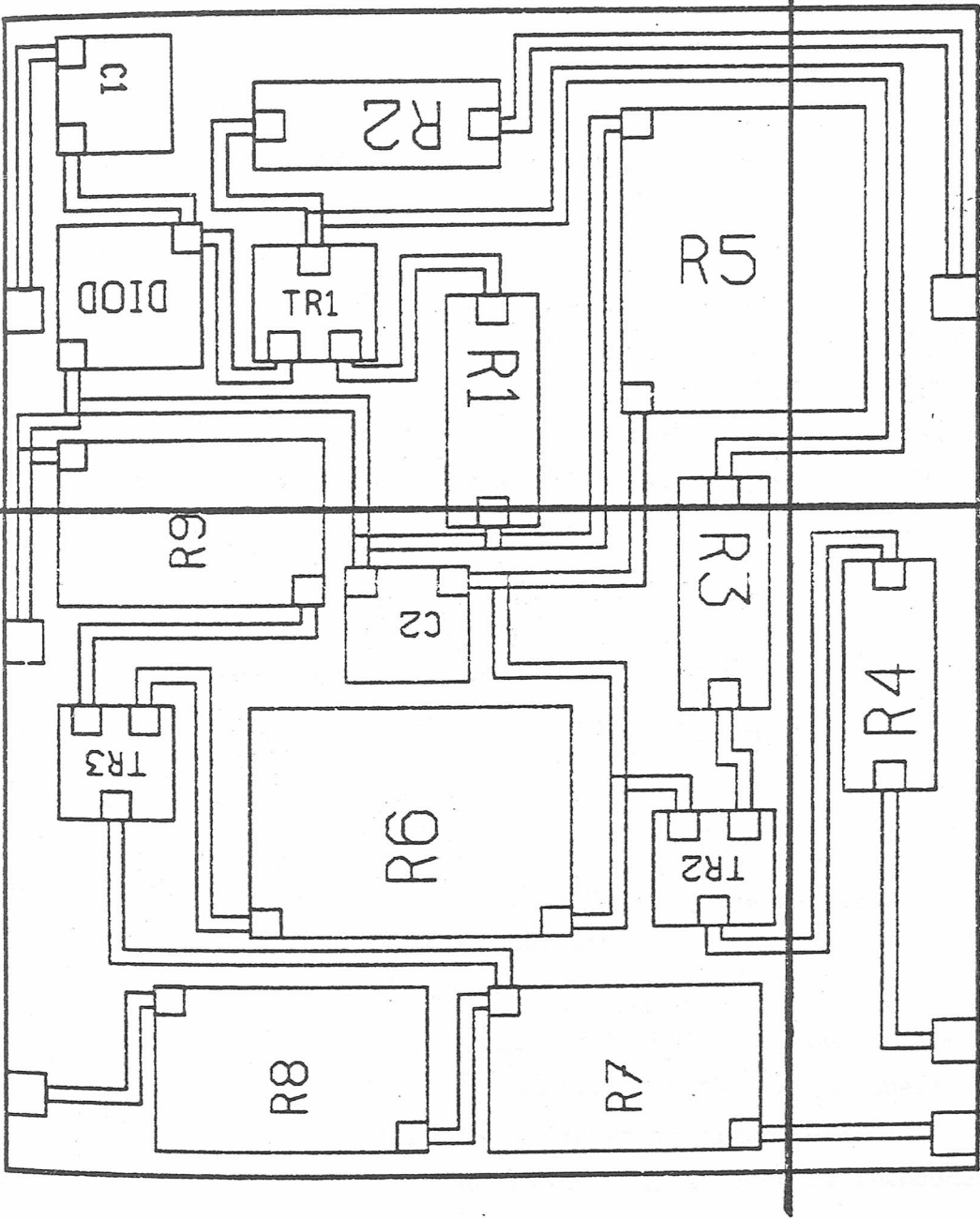
The program compares the X co-ordinate returned from the cursor with a list of X ranges corresponding to the facilities. It is a relatively simple programming task to change the selection when the need arises.

"LAYOUT.FOR" Operating Commands

.....  
AUTOMATIC...(AUT)..... Initiate an Automatic Connection  
BOARD.....(BOA)..... Insert extra board space interactively  
CONNECT UP...(CON)..... Make any possible connections  
CURSOR.....(CUR)..... Move into CURSOR mode  
END..... Close files and exit from program  
EXPAND.....(EXP)..... Expand circuit in a given quadrant  
HARD COPY...(HAR)..... Produces a hard copy of screen  
HELP.....(HEL)..... Prints out this HELP file  
IDENTIFY...(IDE)..... Component is identified  
MANUAL.....(MAN)..... Requests a MANUAL connection  
MASK.....(MAS)..... Changes the o/p format to second mask  
MOVE.....(MOU)..... Move a component  
NEXT.....(NEX)..... Brings on the next components  
NO CON.....(NOC)..... Forget all the connections  
NO NODES...(NON)..... SUPPRESSES O/P OF NODE NOS (OR ENABLES)  
PLACE.....(PLA)..... Place a component on the substrate  
QUERY.....(QUE)..... as SPR but for one component  
REDRAW.....(RED)..... Redraw the complete circuit  
REMOVE.....(REM)..... Remove - specify COMponent / CONNecTion  
RESTART.....(RES)..... Close files and restart program  
ROTATE.....(ROT)..... Rotate a component  
SHORTEN.....(SHO)..... Shorten a connection as far as possible  
SHRINK.....(SHR)..... Shrink the board area as specified  
SLOTS.....(SLO)..... Draw out the SLOTS  
SPRINGS.....(SPR)..... Draw out connections as dotted lines  
WINDOW.....(WIN)..... Reset the screen window dimensions  
ZOOM.....(ZOO)..... Zoom in or out to a specified point on scree

Figure 9.2 LAYOUT operating commands





— cursor lines

Figure 9.3 Example of Cursor Mode

Menu Mode is re-entered when the cursor is positioned such that it does not intersect with any facility box. The easiest way of doing this is to move the vertical cursor line into the Menu Area itself before pressing a keyboard character.

Cursor Mode is much quicker to use than Menu Mode since the need for typing command strings is reduced. Each box must be of a reasonable size, however, so this restricts the selection to a maximum of about twelve different facilities.

### 9.3 Facilities

This section gives a brief description of each facility which is available to the user. In each case it is assumed that Menu Mode has been selected. If this is not the case, then most prompt messages will be suppressed (to save Menu area), and the onus is on the user to know what input data is expected.

It is not necessary to type in the complete facility code in each case - these can be shortened to any extent, provided that the code string given is unique to one facility name.

#### 9.3.1 "AUTOMATIC" Connection

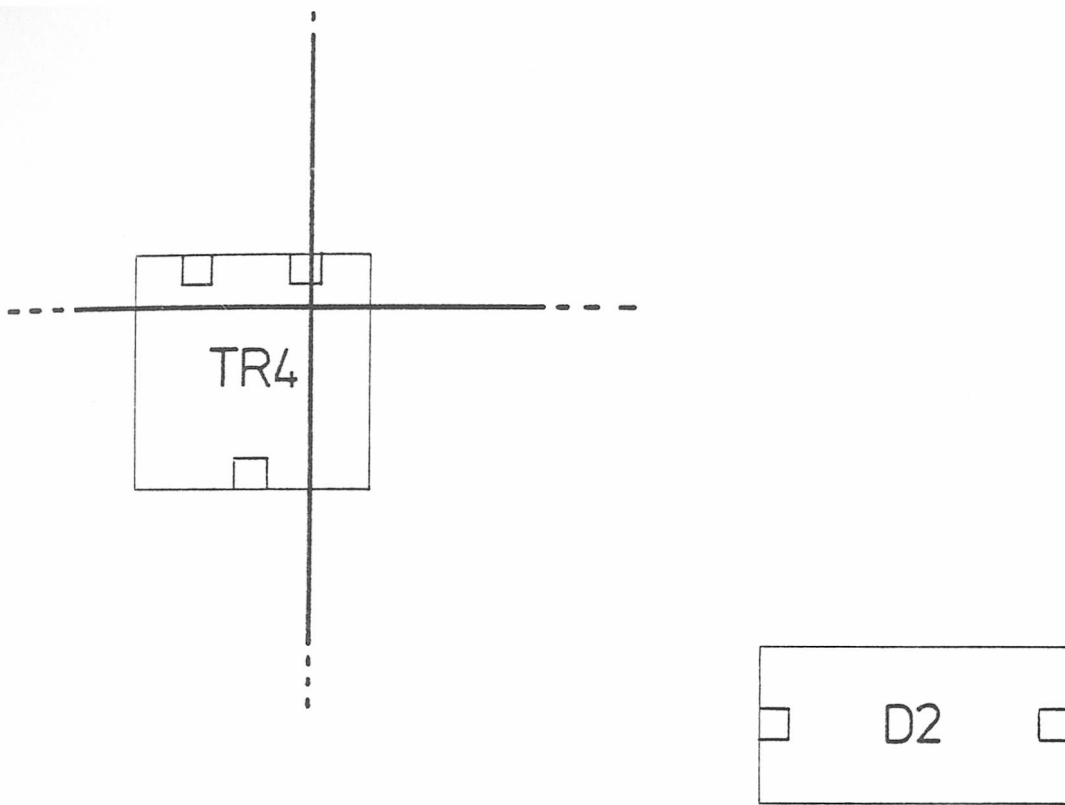
When this facility is selected, the message:

Indicate START and END pads

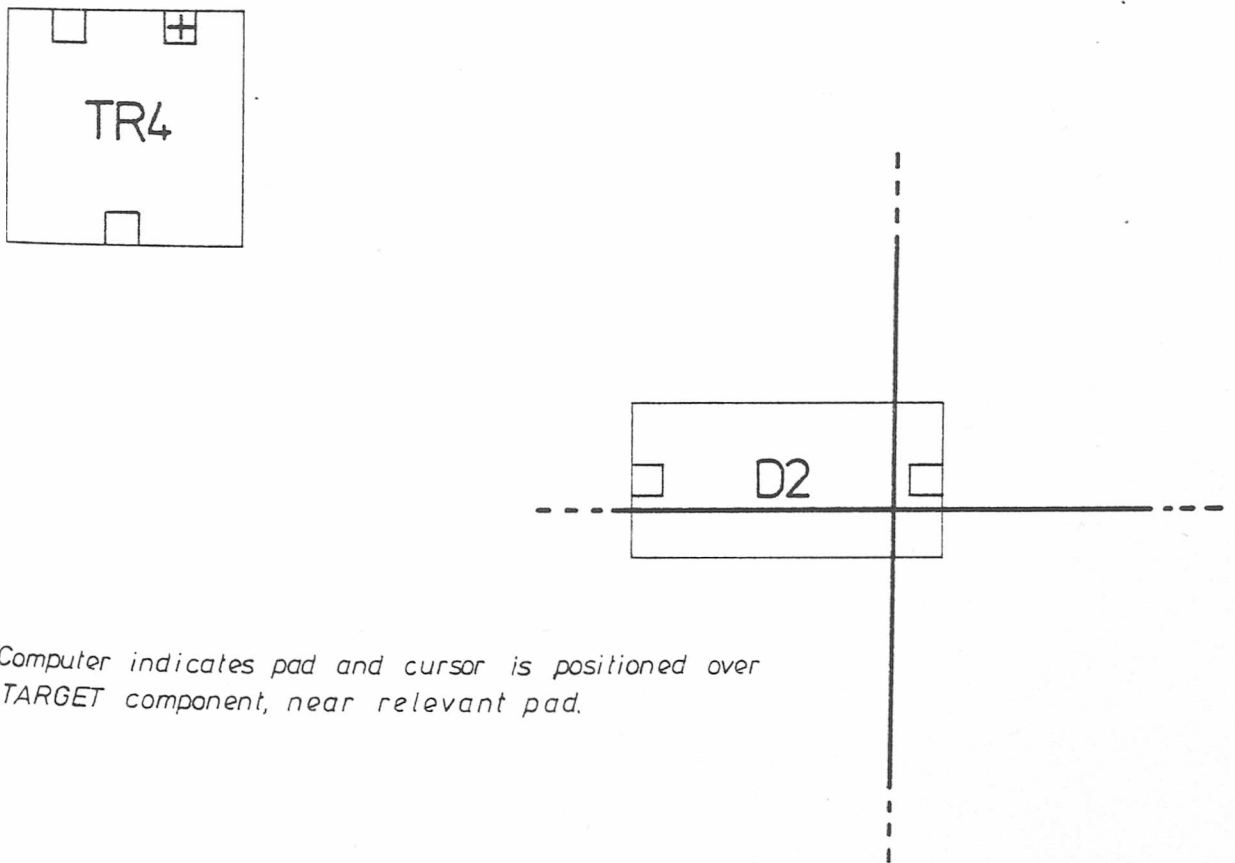
is given and the cursor appears.

The user must now point to the pad at which the connection should start. Assuming that the cursor is positioned within a component boundary, the nearest pad will be selected as shown in Figure 9.4(a). Failing this, an error message will be output and the attempt is abandoned.

A cross is drawn at the START pad position to indicate that the program now requires the location of the second pad. This is defined as before (diagram (b)), and the program proceeds to apply the automatic routing algorithm. If this is successful, the connection is drawn on the screen and the "What Next ?" prompt given. Should the algorithm fail, a message is written to this effect and the attempt is abandoned.

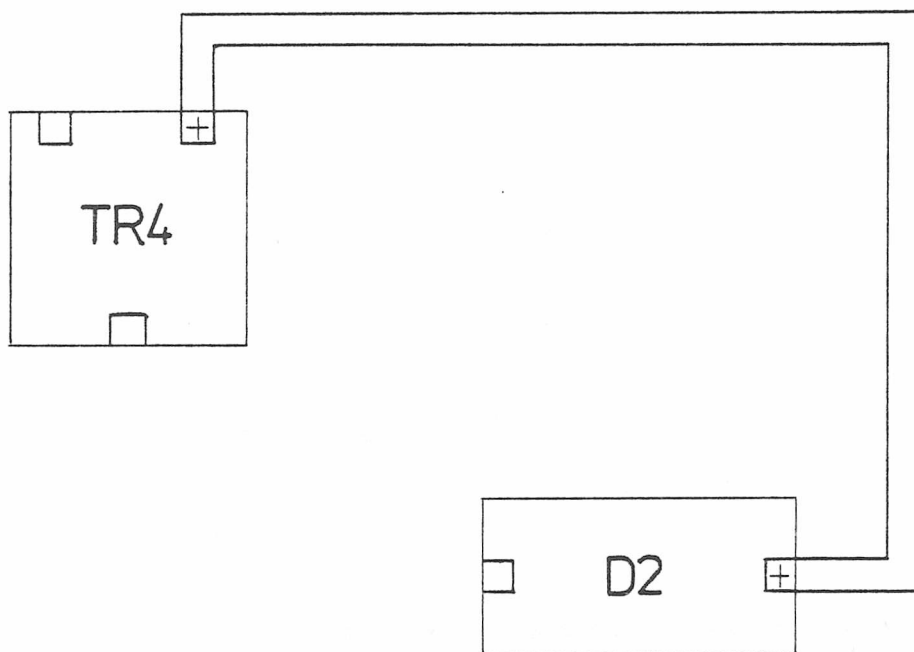


(a) *Cursor is positioned near required pad in first component and a character pressed.*



(b) *Computer indicates pad and cursor is positioned over TARGET component, near relevant pad.*

Figure 9.4 Instigating an automatic connection



(c) Computer indicates TARGET pad and routes connection

Figure 9.4 Instigating an automatic connection (continued)

### 9.3.2 "BCARD" - Increase Board Area

This facility allows the insertion of horizontal and vertical "strips" of board space - effectively increasing the size of the layout area. It is best explained with reference to Figures 9.5 and 9.6 which show the same circuit before and after this technique has been applied.

The cursor is positioned at the desired location and a character is pressed. Thick lines are drawn along the current cursor segment positions as a reference for the next stage, which is to move one of the thumbwheel controls. If a horizontal strip is to be inserted, then the horizontal cursor segment must be moved. Vertical strips can likewise be defined using the vertical control. The thickness of the strip to be inserted is indicated by the distance between the new and old positions of the relevant cursor segment.

The layout is next adjusted and the graphical output re-scaled to fit the screen - as shown in Figure 9.6. A vertical strip has been inserted in this example.

A very similar technique is used to remove unwanted strips of board area. This involves the use of the "SHRINK" facility which will be discussed in section 9.3.23.

### 9.3.3 "CONNECT" Up Components

This command causes the Routing Algorithm to be applied in a general fashion. It is equivalent to using the "AUTOMATIC" (Automatic Connections, section 9.3.1) command between every pair of terminal pads bearing the same Node Number. Figure 9.7 gives the strategy required to do this in the form of a flowchart.

The first task is to find a list of component pairs which require to be connected together. The most convenient way of doing this is to consult the Planar Graph of the circuit. Since each "Region" in the graph is actually an interconnected loop of components, every component must be connected to its immediate neighbours in the definition.

It is therefore quite straightforward to follow round the Region definitions, picking out adjacent pairs to add to the list. Each Region

What Next ?  
#BOARD

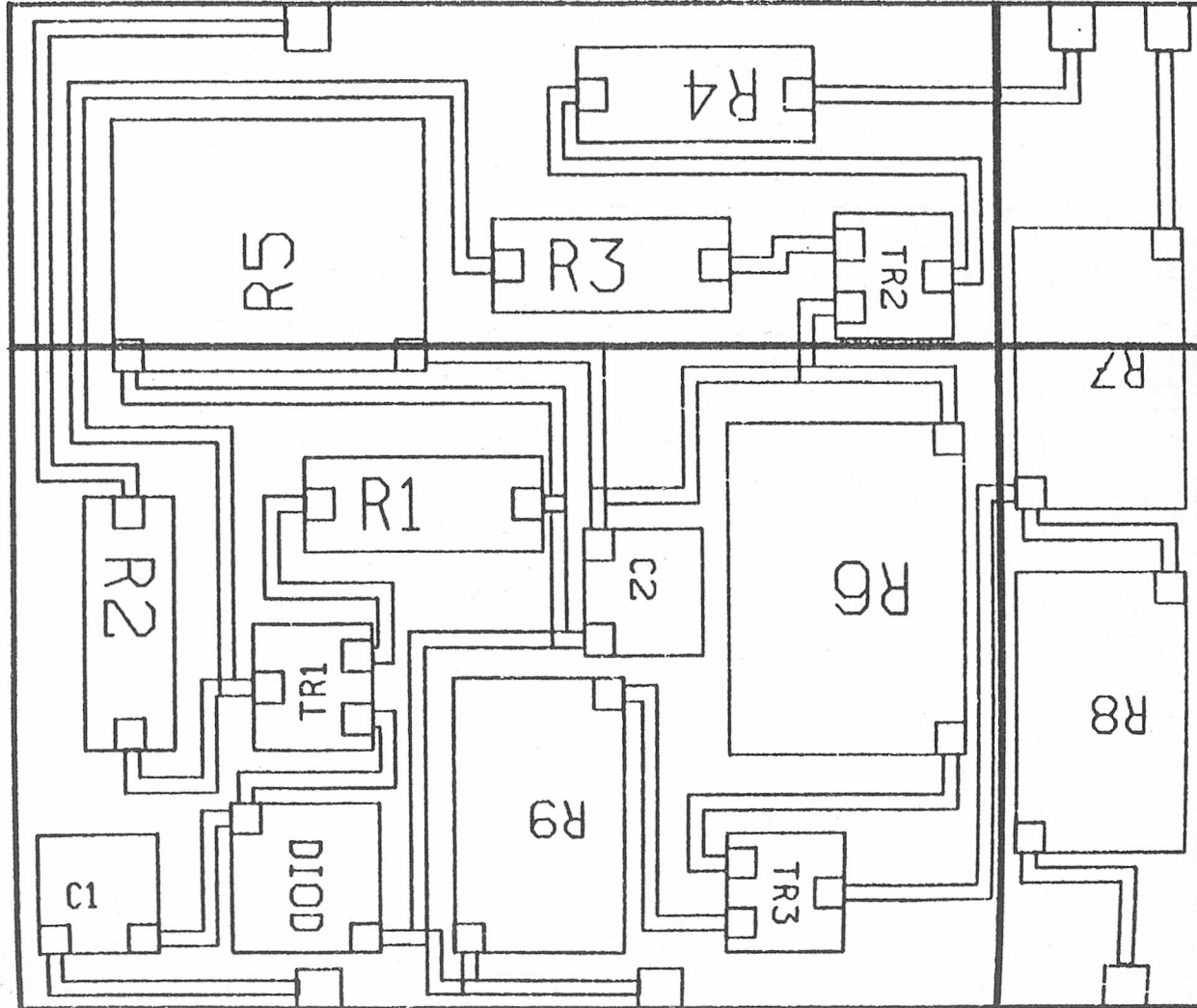
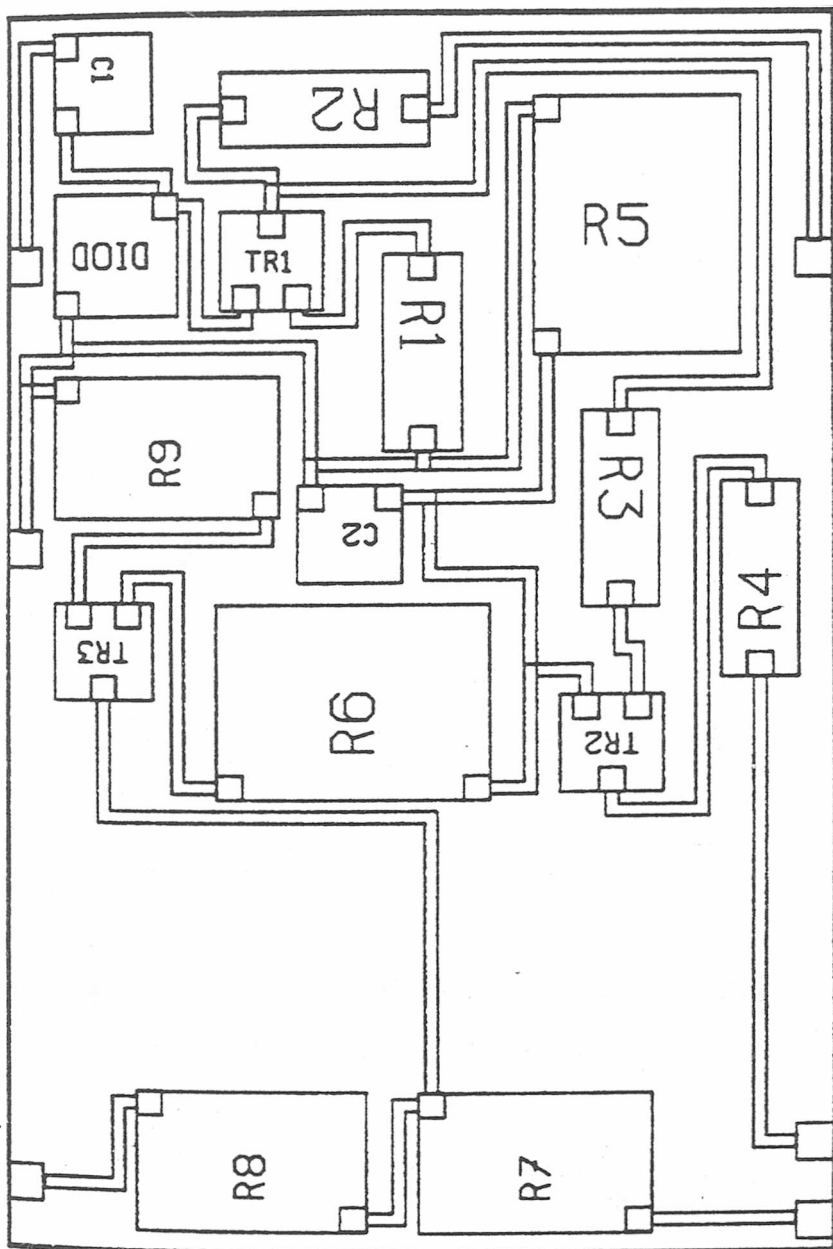


Figure 9.5 Preliminaries to the "board" facilities



• What Next ?

Figure 9.6 Results of "Board" facility

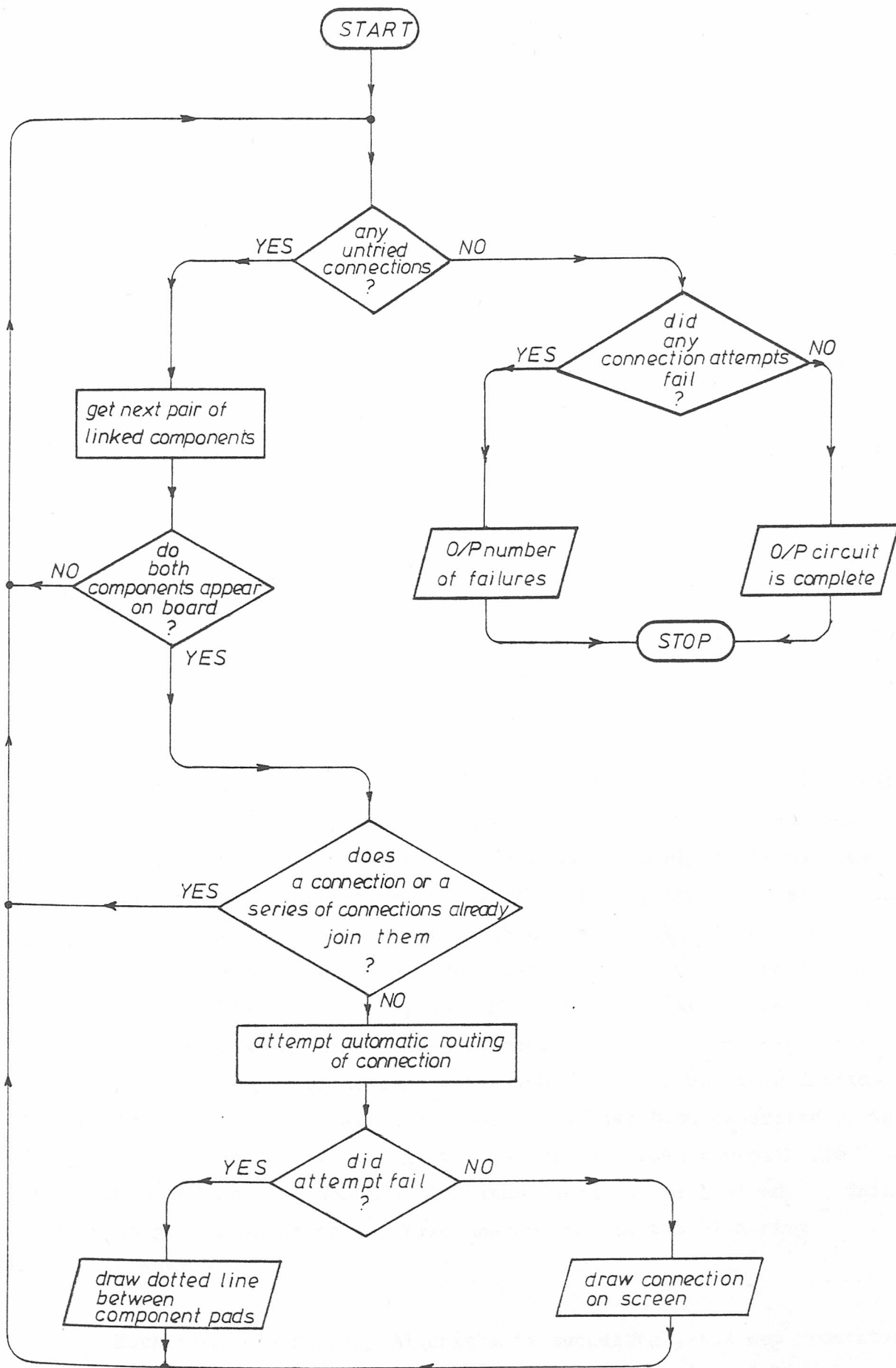


Figure 9.7 Flowchart of the "CON" facility algorithm



will produce "n" pairs if it contains "n" Branches. The only exception to this is Region number 1 which links all the Edge Pads together to simulate the allowable board area. Since each pad represents a different Node Number, it is meaningless to use this Region as a basis for interconnection.

An alternative method of generating the pair list is to carry out an exhaustive search for terminals of similar Node Number. This would involve picking one component pad then looking at every other in turn to find a match - obviously a much slower technique. It can, however, be used in conjunction with the Planar Graph approach to significantly reduce connection length. This is discussed further in Chapter 11.5.2, but has not been implemented due to the increased execution time involved.

The "CONNECT UP" facility can be used at any time throughout the design process, so it is not certain whether every component has actually been placed on the Board. Having found a legitimate pair, the algorithm must first check that they both appear on the substrate before applying the routing process.

Given that this is so, there still remains the possibility that they have already been connected together. This presents a problem, since they may be indirectly linked in a "Component String" through any number of intermediate components. In Figure 9.8, for example, components A and B are electrically connected through C and D although there exists no direct link between them. A "Tree Search" of all the Connection Blocks is used to check this. The program starts at the first component pad and computes the pads which have been connected to it. It then starts from each of the new positions generating a new list of possible destinations. The process continues until all the connections have been exhausted or the target pad is reached. If the latter is the case then the pair are already connected and the Routing Algorithm need not be applied. This technique is very similar to the Tree Search used in the Planarity Algorithm (Chapter 5.2).

Each time the Routing Algorithm is successful, the new connection is immediately drawn on the screen. When it fails, a dotted line is drawn between the two component pads in question (refer to the "SPRINGS" facility in section 9.3.25).

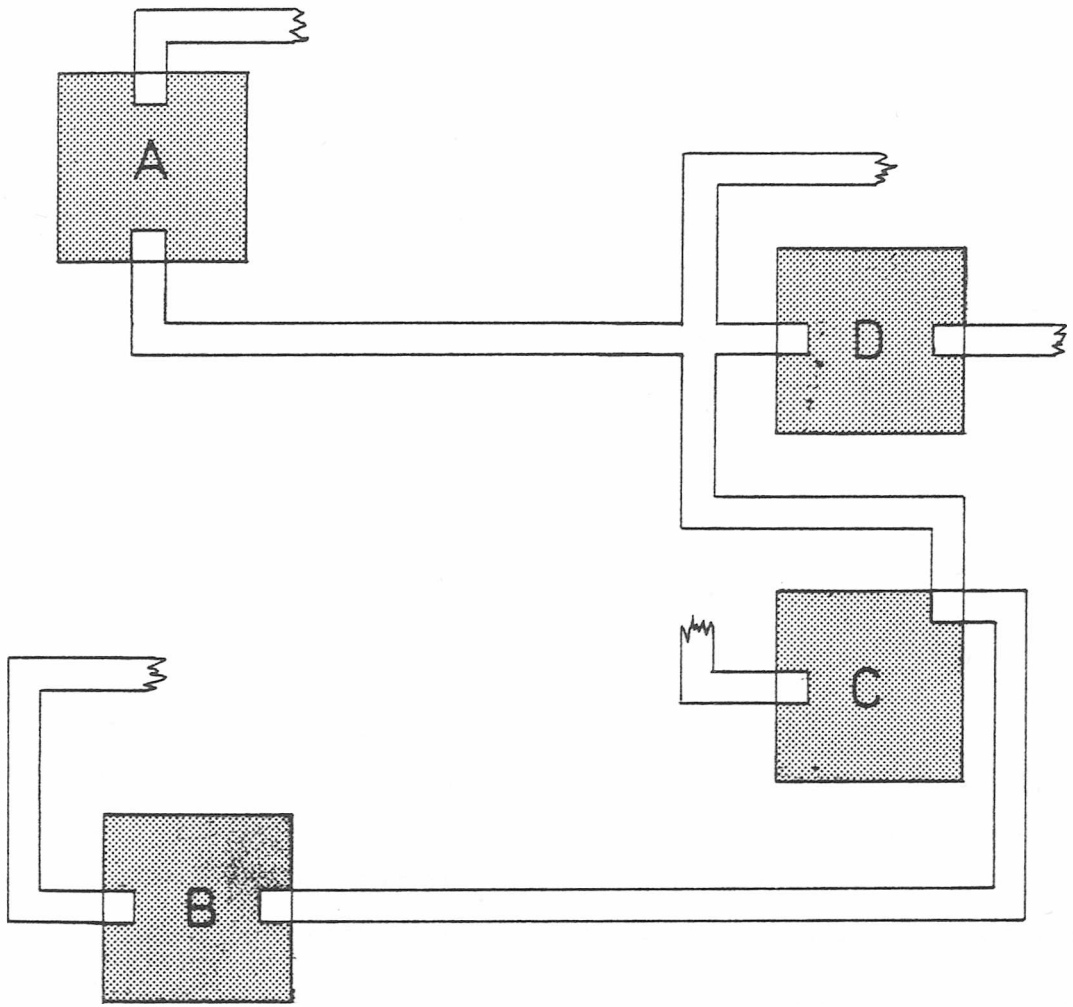


Figure 9-8 Connection string

After every connection pair has been attempted, the program writes a message in the menu area and requests another command. If every component has been placed on the board and every connection pair is now linked, the message:

#### THE CIRCUIT IS COMPLETE

is output. If every connection attempt succeeded, but there are still some unplaced components, the following will appear:

#### NO FAILURES

A count is kept of any failures which may have resulted during the process and this may also be displayed as, for example:

3 HAVE FAILED !

#### 9.3.4 "END" Program

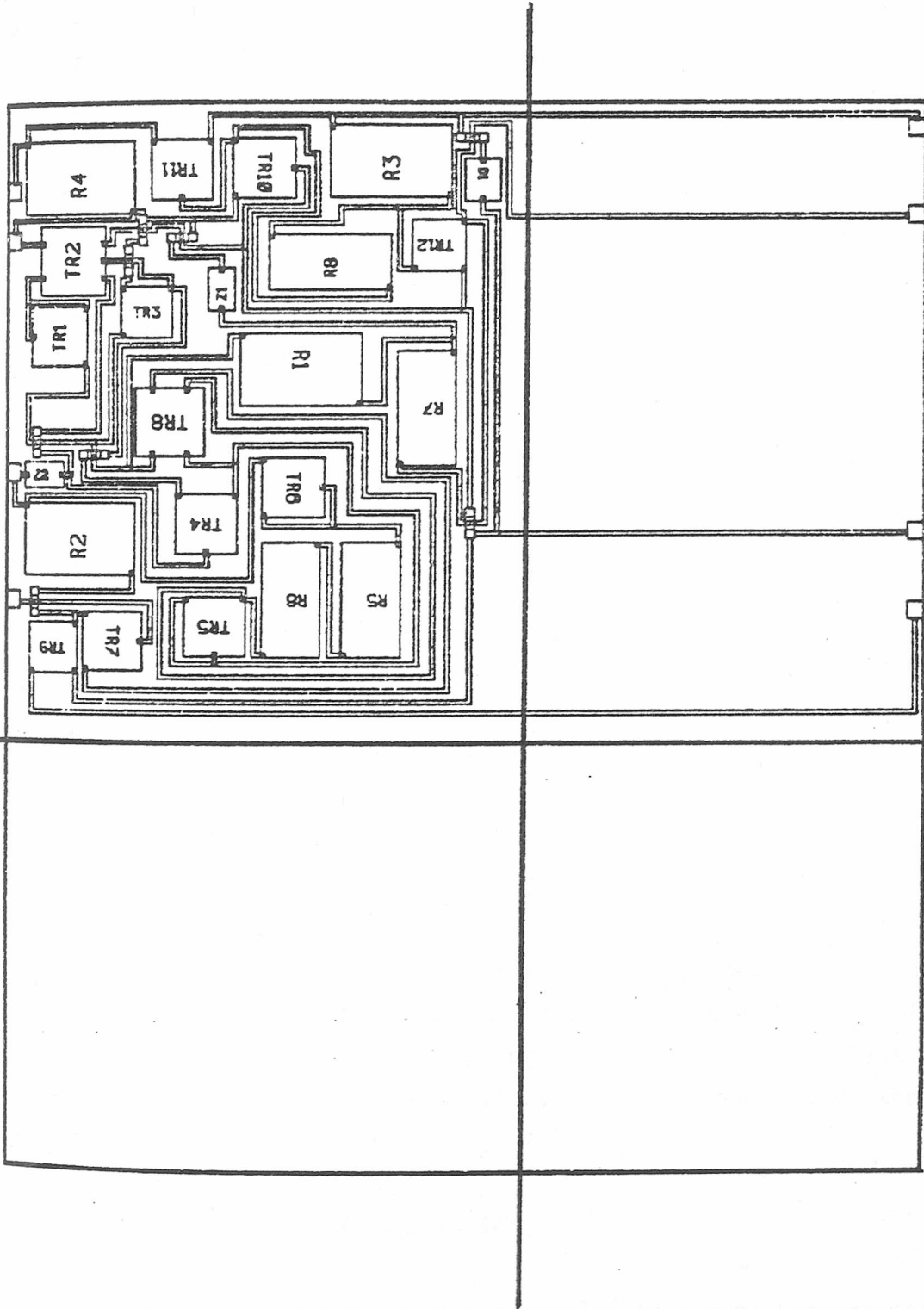
As the name suggests, this facility causes the program to terminate and update the data structure file with the latest information. An alternative to this is to use the job control language halt command (CNTRL/C), which stops the program running. The data file would not then be altered, and the user could start from the point at which he last filed the layout data.

The "RESTART" facility (section 9.3.20) has the same effect as the "END" facility, in that it updates the layout file. Instead of terminating, however, the user is then invited to enter a new data file name and the program is restarted.

#### 9.3.5 "EXPAND" Layout

The "Expand" facility automatically increases spaces, and alters connection tracks, in order to map a small rectangular window of the layout onto a larger. It is best illustrated by example.

Figure 9.9(a) shows a layout in which the substrate has been increased interactively using the "Board" facility (9.3.2). The cursor is used to define the smaller rectangle. Any of the four quadrants generated by the intersection of the two cursor lines can be selected.



What Next ?

Figure 9.9(a) Use of the "Expand" facility

In this case the bottom left (BL) has been chosen. The others are referred to as BR (Bottom Right), TL (Top Left) and TR (Top Right).

Having defined the larger rectangle in a smaller manner, the program indicates both rectangles using dotted lines (see Figure 9.9(b)) and asks the operator if he still intends to expand the circuit with the rectangles shown. He can abort the process by typing "NO" at this point.

Assuming that he allows the program to proceed, the adjusted layout is redrawn as in Figure 9.9(c).

Connections crossing the boundary between the two rectangles are deleted, since it is not clear where the corner points on either side should go.

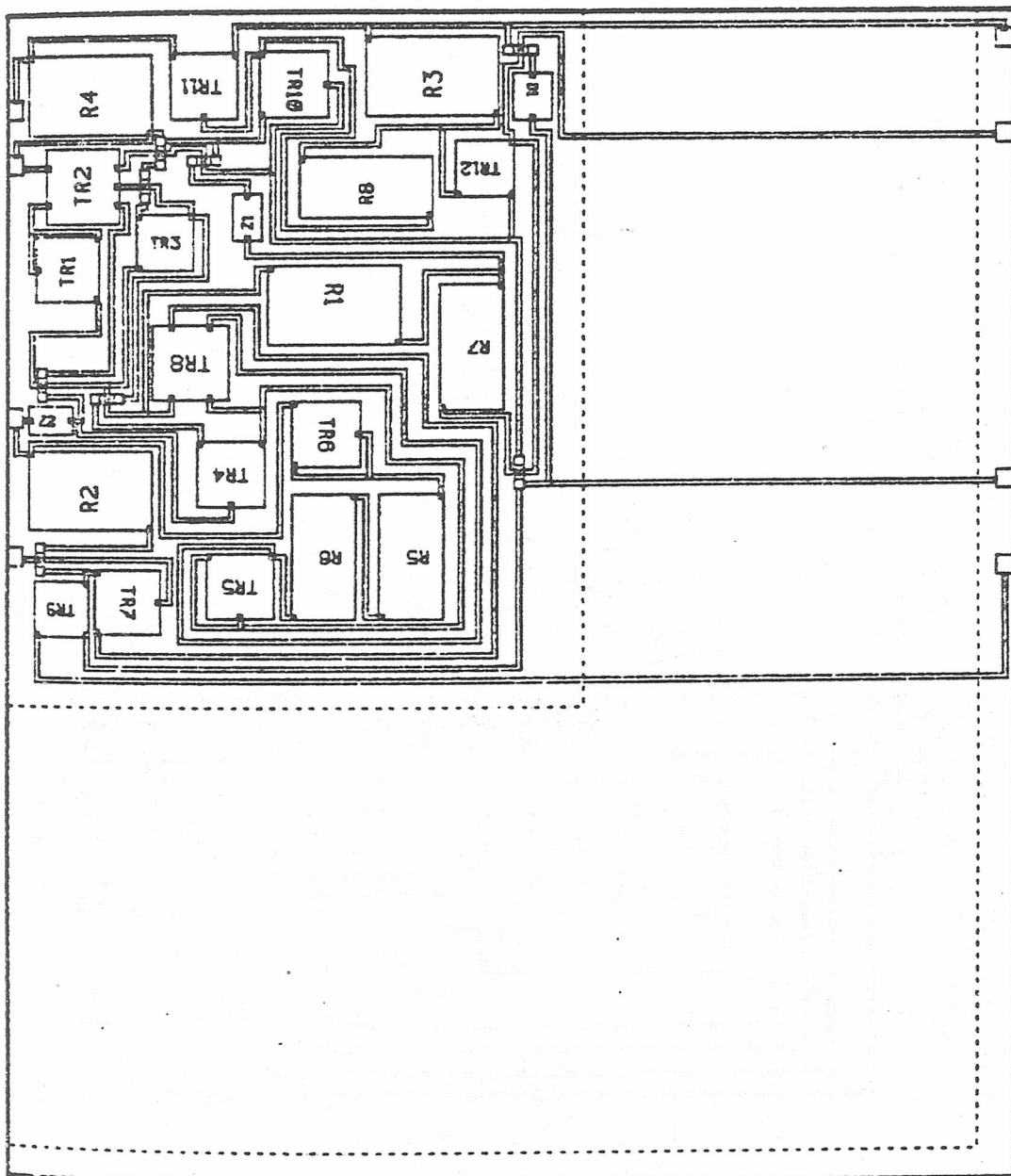
Edge Pads cannot be moved in a Y direction since they are constrained to lie along the top and bottom edges. Connections to Edge Pads which lie within the smaller rectangle are therefore deleted.

It is now up to the user to move Edge Pads and reform any connections which may have been deleted. The latter task may be manual or automatic as required. Figure 9.9(d) shows the final expanded layout after subsequent alteration.

The "EXPAND" facility is also very useful when applied to the Initial Placement string along the bottom of the Board. If it is known that the board width is not going to be reduced, it is best to expand the components to take up the available space. This eases the task of the Placement Algorithm.

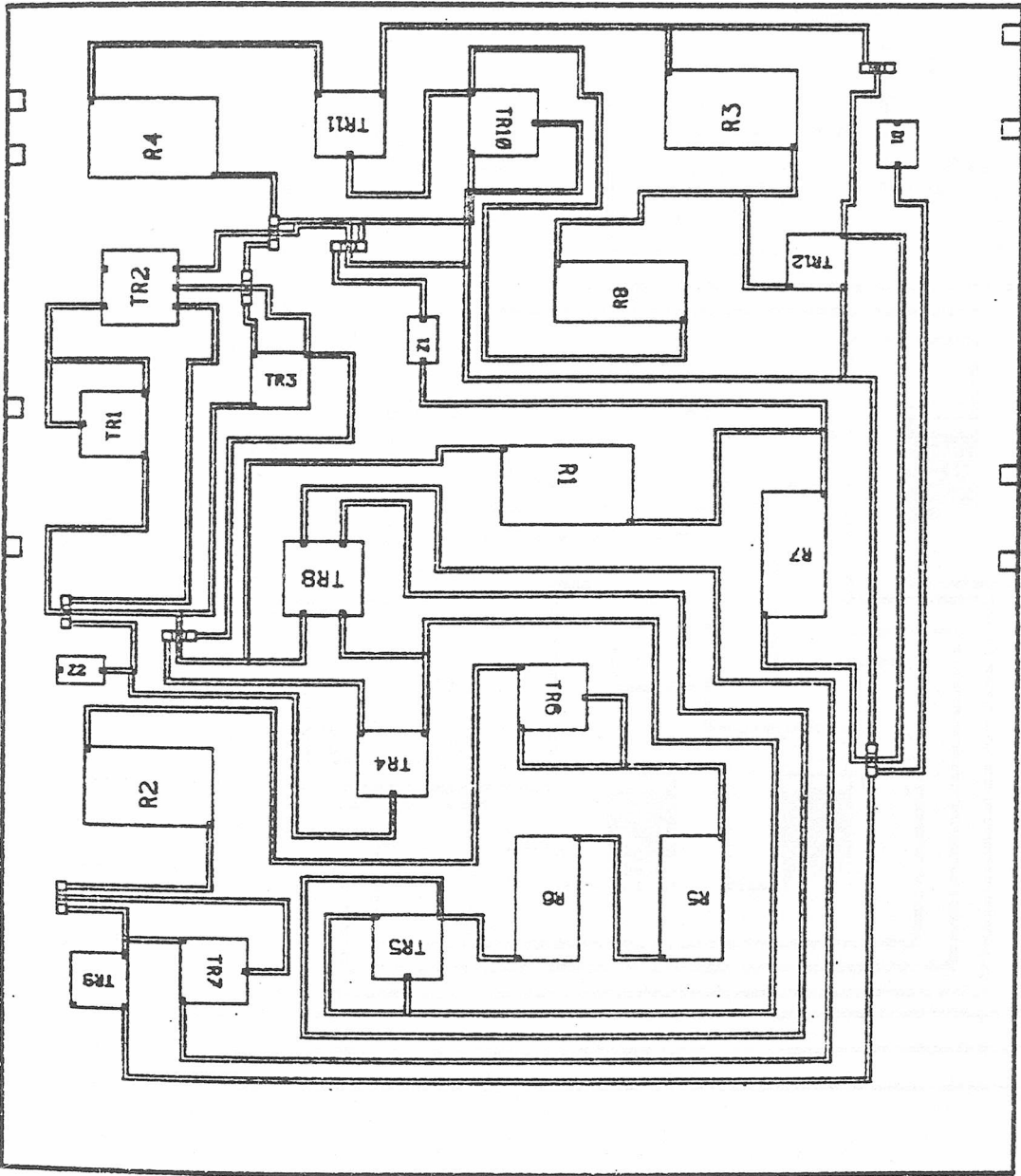
#### 9.3.6 "HARD COPY" Generation

The terminal used was coupled to a Hard Copy device which takes a permanent copy of the drawings on the screen. The paper diagram which is obtained gives a very good representation of the layout and is a quick substitute for a pen-plotter drawing. Typing "HARD COPY" will cause a copy to be produced. The duplicator can also be operated using a push button.



What Next ?  
\$EXPAND  
Expand -- BL, TL, TR or BR ?  
Shall I carry on ? YES

Figure 9.9(b) Use of the "Expand" facility



What Next ?

Figure 9.9 (c) Use of the "Expand" facility

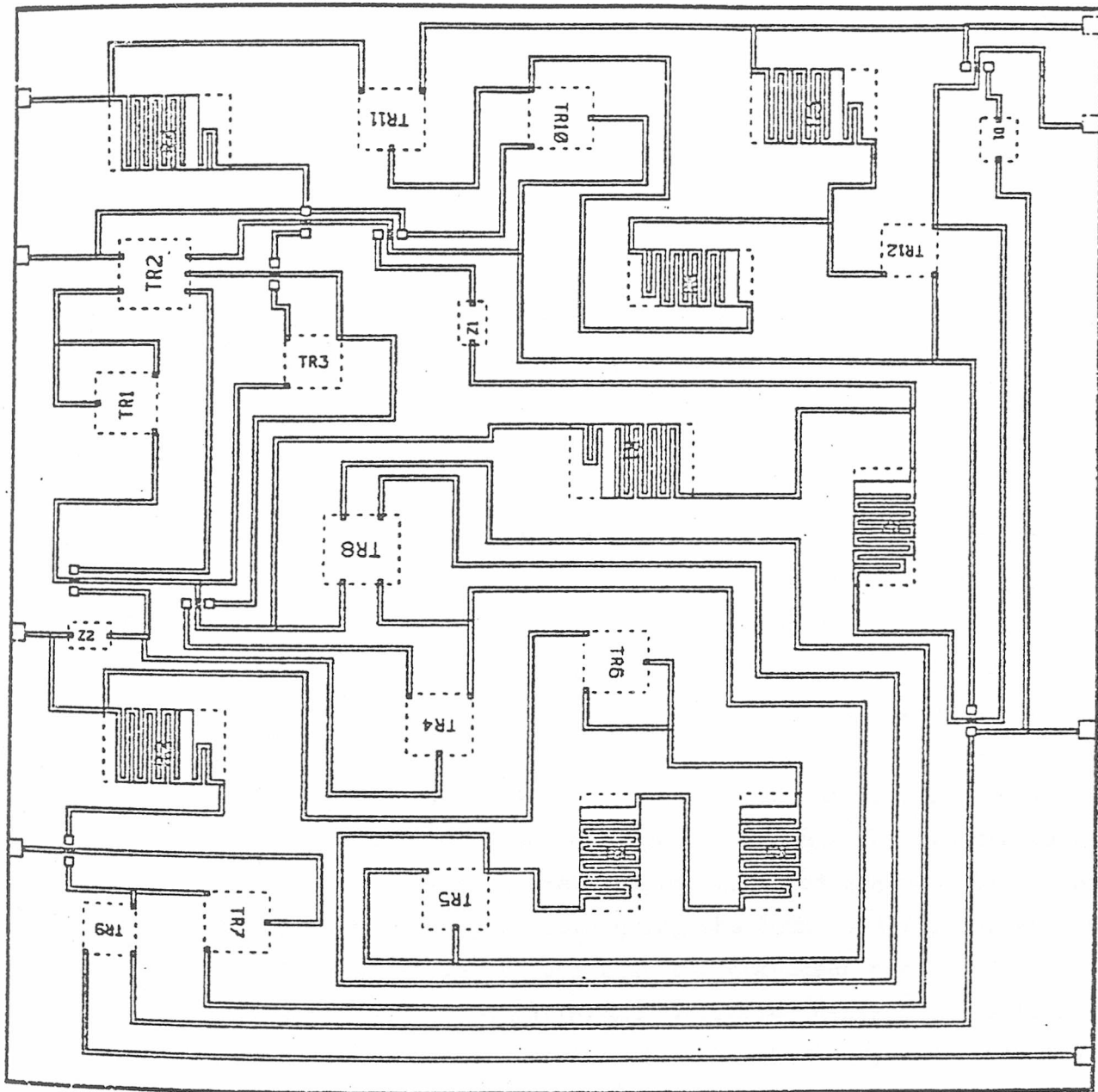


Figure 9.9(d) Use of the "expand" facility



The advantage of this facility lies in the fact that the computer job control language is buffered. This means that the user can type in several commands without having to wait until each has been executed. It is therefore perfectly possible to copy several layouts without having to sit idly waiting for each to be drawn before pushing the "copy" button.

### 9.3.7 "HELP" List

A complete list of the facilities available to the user can be displayed by typing the word "HELP" in the menu area. After erasing the screen, the program will list the commands as shown in Figure 9.2.

### 9.3.8 - "IDENTIFY" Component

Two distinct methods of character output are used on the graphics screen. The first is called "hardware generation", which means that the terminal automatically writes the character using a dot matrix. An alternative is to use "software generated" output. In this mode the characters are drawn using a series of straight lines set up in the program.

The former is a very speedy way of generating text and is used to write all command prompts and messages. It has the disadvantage, however, that only certain discrete character sizes are possible. Since the layout may be scaled up or down using the ZOOM (9.3.27) and WINDOW (9.3.26) facilities, it is obvious that the component names and Node Numbers will also have to be infinitely variable in size. Software generated characters, being a series of straight lines, may be scaled in an identical manner to the component shapes and are necessary in this case.

Software generated characters are literally hundreds of times slower to draw than their hardware generated equivalent. The time needed to re-draw the layout can therefore be significantly reduced by omitting character names and Node Numbers ("NO NAMES" section 9.3.14 and "NO NODES" section 9.3.15). If this is the case then the user must be provided with a means of finding out the name of a particular component without having to re-draw the complete circuit. The facility to do this is called "IDENTIFY".

After typing the code mnemonic, the computer responds with:

"Indicate Component"

and sets up the cursor segments. The user must now position the cursor within the boundaries of the component rectangle which he wishes to identify. One of several responses may now be made. If the cursor has not been positioned inside a component definition, the message:

COMPONENT NOT FOUND

will be given and the attempt terminated. If a component is found, its name and rotation angle are supplied e.g.:

COMPONENT "TR6"ANGLE 270

The "IDENTIFY" facility is also very useful when the scaling is such that the component names cannot easily be read.

#### 9.3.9 "MANUAL" Connection

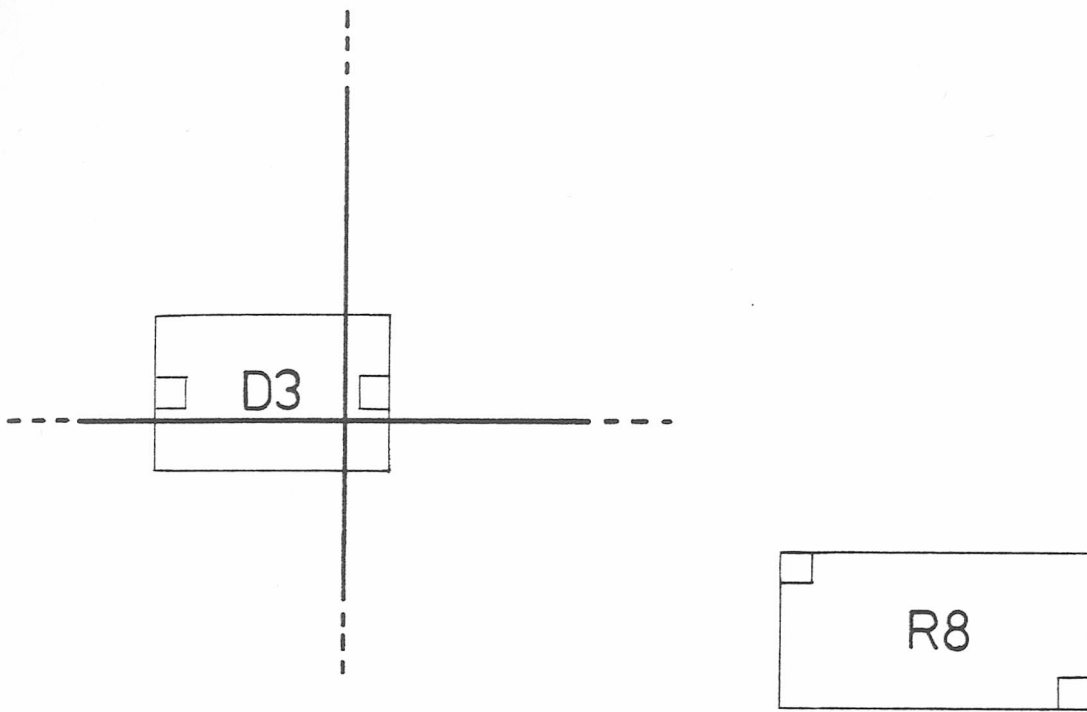
When the Routing Algorithm fails, or the user wishes to improve the result, he can define a connection path interactively using the "MANUAL" facility. This is demonstrated with reference to Figure 9.10.

The first stage is to indicate the component pad at which the connection is to start. This need not be a very exact operation, since the nearest pad will be selected providing that the cursor is actually inside a component definition. Failing this, the message:

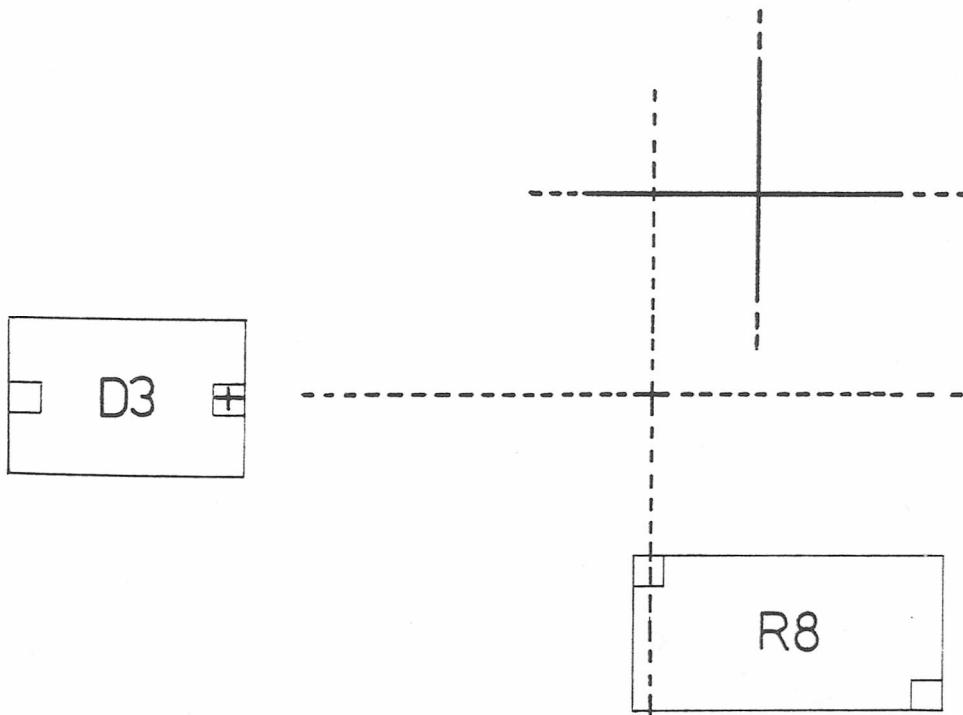
COMPONENT NOT FOUND

will be displayed and the connection attempt aborted.

The pad is marked with a cross as shown in diagram 9.10(b), and the first corner point can then be defined. Connections are restricted to be made up of horizontal and vertical segments, so the program checks to see which co-ordinate has changed the most. It then assumes that the other remains unchanged. In this case the X change is greater than the Y, so a horizontal segment is assumed. The Y co-ordinate is reset to be the same as the last value and another cross drawn on the screen as in Figure 9.10(c).

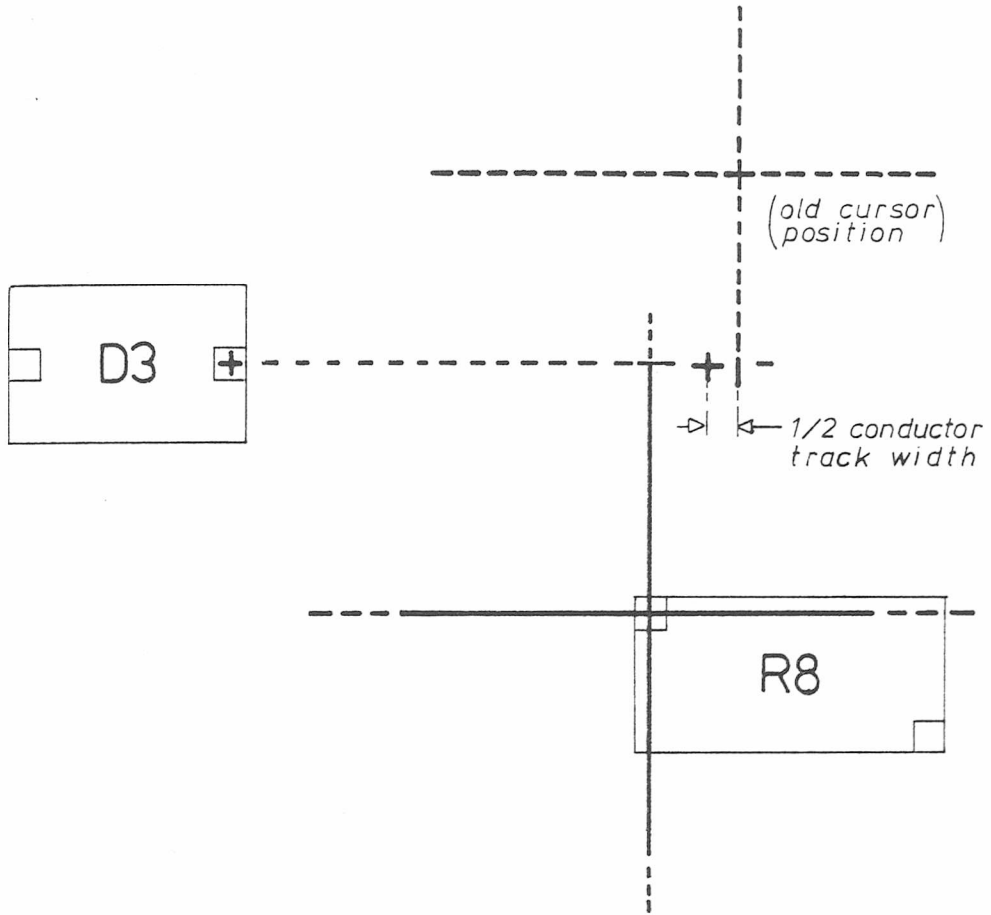


(a) *Cursor is positioned near pad in first component.*

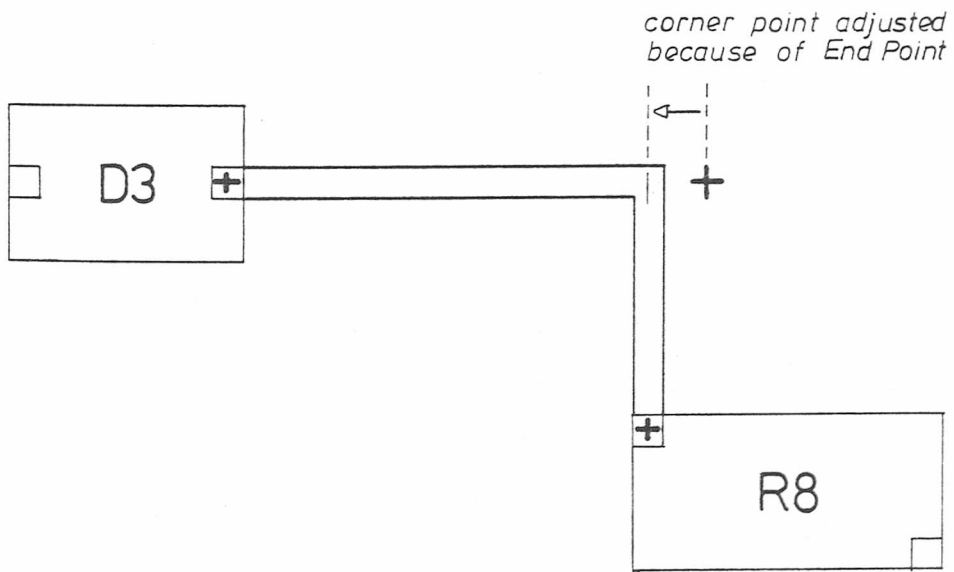


(b) *First corner point is then indicated*

Figure 9.10 Defining a connection manually



(c) TARGET pad is finally indicated (approximately)



(d) Connection is drawn on screen and stored in data structure

Figure 9.10 Defining a connection manually (continued)

To aid the user in positioning the edge of the connection, the true corner position is set to be half a track width back from the cursor position in the direction of movement. In Figure 9.10(c), the direction is to the RIGHT, so the corner position will be set slightly to the left of the cursor.

When all the corner points have been defined, the user can terminate the connection by positioning the cursor near a component pad and typing the letter "F" (for FINISH). The program first ensures that the last corner point is in a legal position to give a non-sloping end segment. In Figure 9.10(d), the last direction of movement is along the Y axis, so the penultimate X co-ordinate has been adjusted.

The connection is then drawn onto the screen and the process terminated.

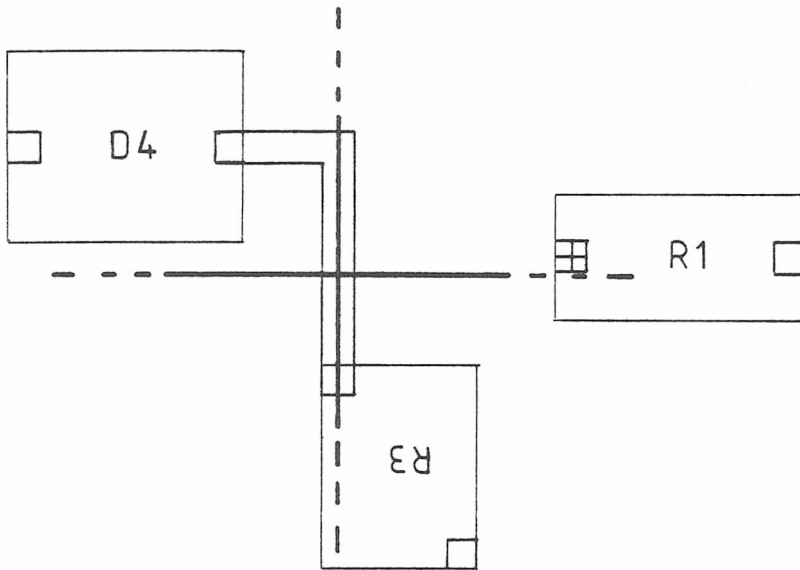
An additional option is to abort the process at any stage in the connection attempt. This is achieved by typing an "X" (for EXIT) character while defining a Start, Corner or End point. In this event, the process is terminated immediately.

It may be convenient, in some situations, to use part of an existing connection while defining the new connection. This is called "Latching", and can be employed by typing the character "L". If the cursor is positioned within a connection boundary, the relevant cursor co-ordinate(s) are adjusted to those of the connection. Consider the situation shown in Figure 9.11(a).

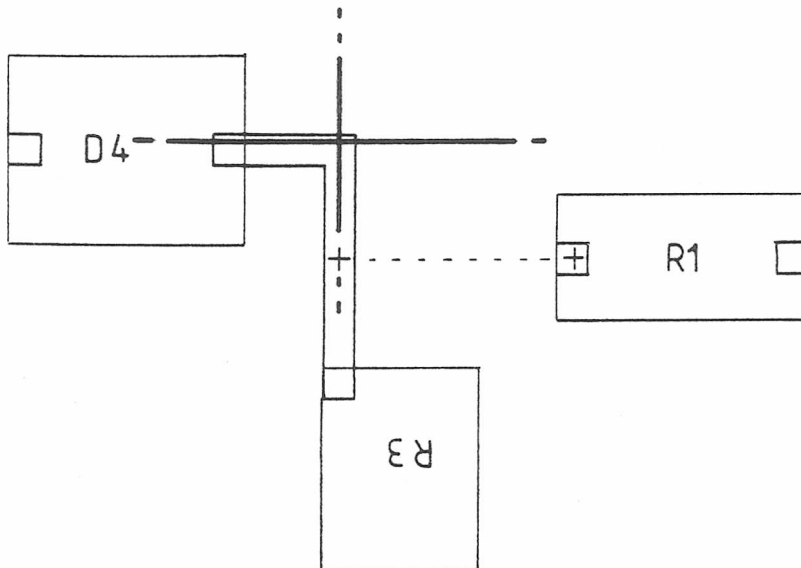
A connection is to be made between components R1 and D4. Having specified the start pad in R1, the user has latched onto an existing connection between D4 and R3. He then latches onto an existing corner point in Figure 9.11(b) by placing the cursor within the corner "square". When the connection is drawn onto the screen, both conductor tracks will be co-incident for some way. This saves in plotting time and, of course, reduces the effective overall conductor length (Figure 9.11(d)).

#### 9.3.10 "MASK" Selection

Components and crossovers are generally represented by a rectangle encompassing a number of terminal pads. It is also possible to view the

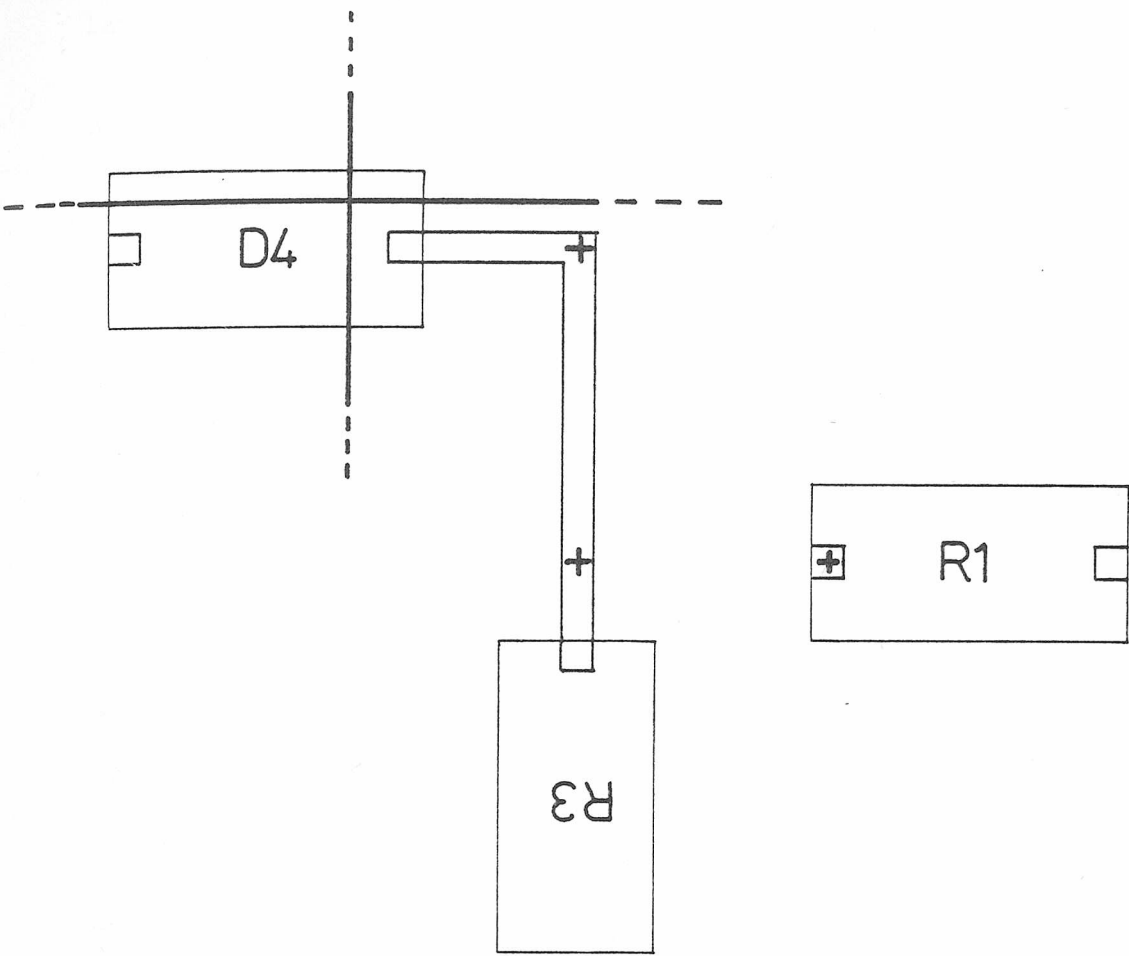


(a) *Cursor positioned within connection boundary after start pad has been indicated*

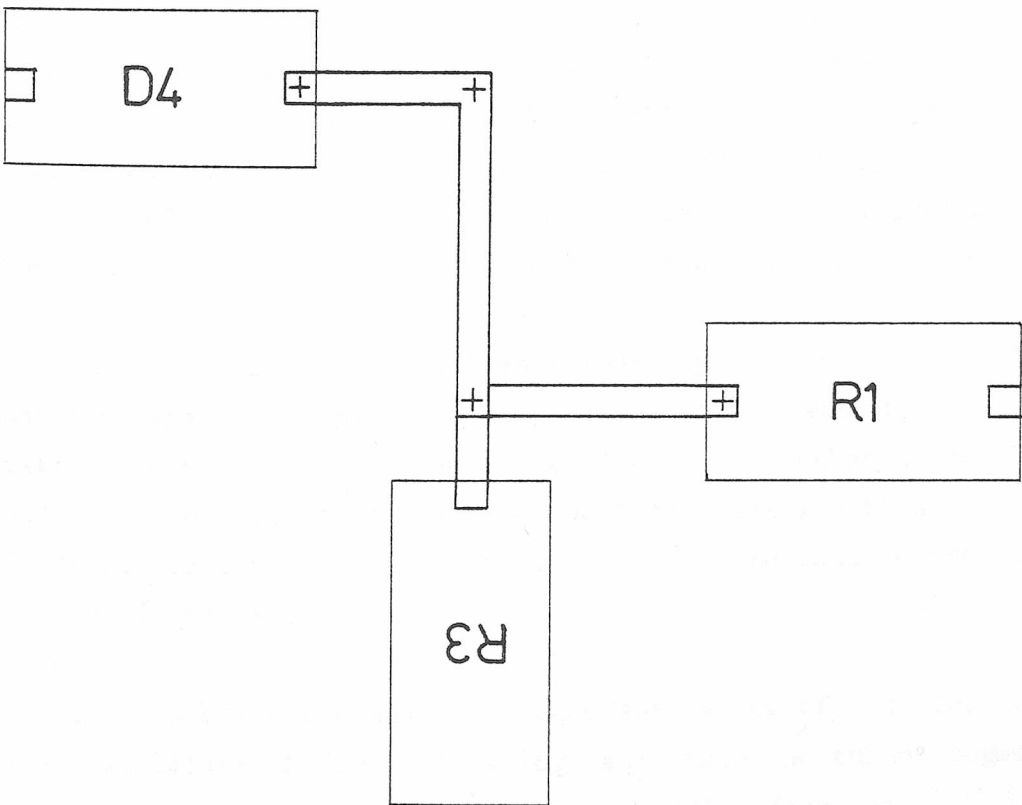


(b) *Corner point is adjusted to lie within connection and to have the same y-co-ord. as the START pad*

Figure 9.11 "Latching" on to an existing connection



(c) Second corner point is marked and cursor is positioned over TARGET pad.



(d) Connection complete.

Figure 9.11 "Latching" on to an existing connection (continued)

layout as a set of two superimposed photolithographic masks which will be produced to fabricate the circuit. This mode of output is selected by typing "MASK", and rejected by a further call to this facility.

In "Mask Mode", lines drawn in solid are on Mask 1 and dotted lines are on Mask 2 (Refer to Chapter 1.2 for further information). The only exception to this are Component Names, Node Numbers and perimeter boundaries round attached devices which will not appear on either mask. Figures 9.9(c) and 9.9(d) shown the same layout in normal and "Mask Mode" respectively.

#### 9.3.11 "MOVE" Component

The user can reposition components and Edge Pads with the "Move" facility. The latter are restricted to lie along horizontal board edges, however, so their Y co-ordinate cannot be changed.

After receiving the command string, the program gives the prompt message:

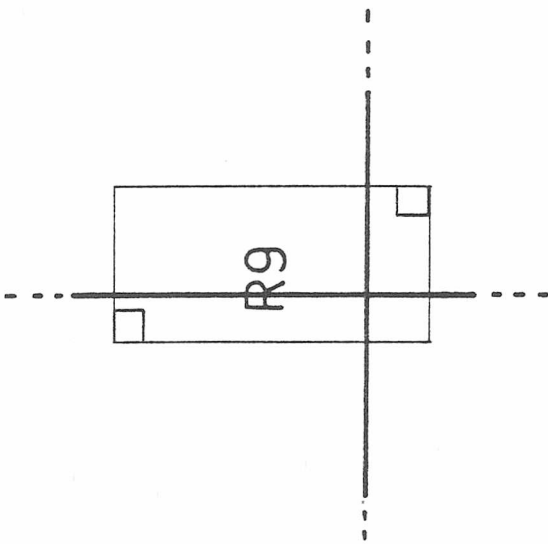
INDICATE COMPONENT AND NEW POSITION

and sets the cursor lines up. The user must now position the cursor inside a component or Edge Pad rectangle and hit a character button. This is demonstrated in Figure 9.12(a). The program then selects the nearest rectangle corner and marks it with a cross as shown in 9.12(b). This point can now be re-defined with the cursor using any button except the "space" button. A dotted outline is drawn to represent the component's new position and the cursor re-appears as in Figure 9.12(c).

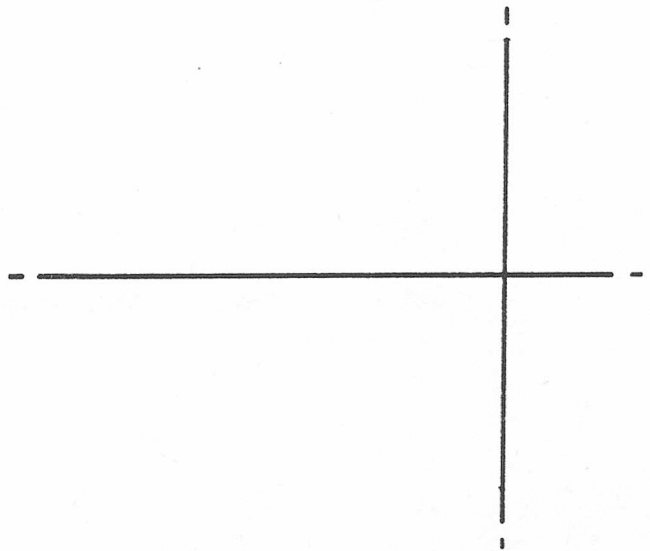
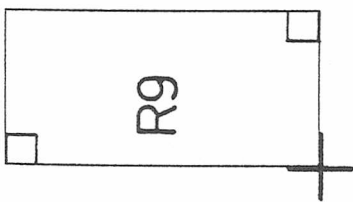
Any number of trial positions may be specified in this way until a "space" is typed. On receiving this command, the old definition is marked with an angled line to show that it has been deleted, and the component is re-drawn at the last trial location. In Figure 9.12(d), for example, three trial attempts were made before the space button was pressed.

A component may be "picked up" by any of its four corners. This simplifies the re-positioning task since the cursor segment lines can be regarded as two of the bounding rectangle edges.



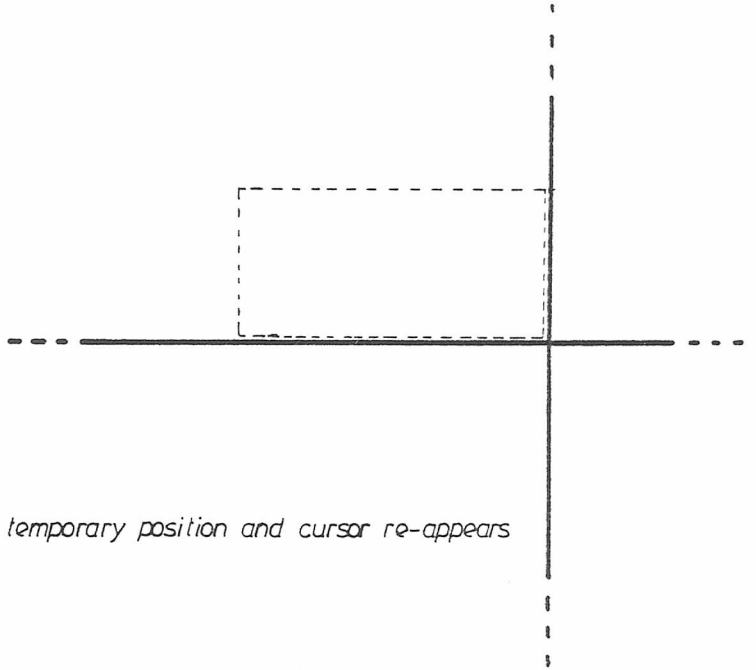
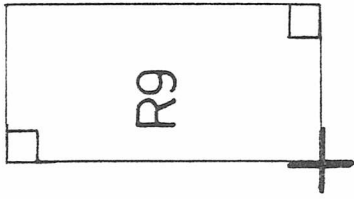


(a) *Cursor is positioned over a component and a character pressed*

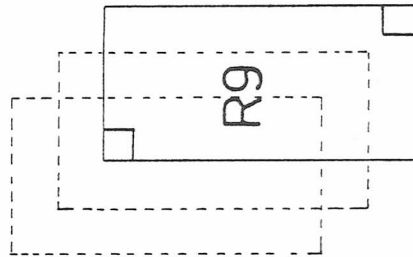
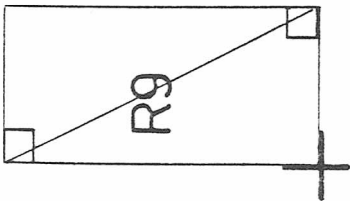


(b) *Computer responds by marking nearest corner and cursor is re-positioned*

Figure 9.12 Moving a component interactively



(c) Dotted outline marks temporary position and cursor re-appears



(d) Last temporary position is confirmed when the space button is pressed.  
The old definition is deleted and the component re-drawn in the new position.

Figure 9.12 Moving a component interactively (continued)

### 9.3.12 "NEXT" Components

The "NEXT" Command is used to initiate the Placement Algorithm described in Chapter 7. If all the components already appear on the board then the message:

#### NO REGIONS

is printed and the placement attempt abandoned. Should there be any unplaced components then a pertinent Region is selected and the Placement Algorithm applied. The new components are then drawn on the screen.

### 9.3.13 "NO CONNECTIONS"

When "NO CONNECTIONS" is typed, all connection data is deleted. This facility can be used in conjunction with the "CONNECT UP" facility (9.3.3) to ensure minimum connection length after a board expansion using "EXPAND" (9.3.5).

There appears to be no effect on the layout, but all the connections have been deleted and will not appear when the circuit is re-drawn.

Once this facility has been used, the connection data is irretrievably lost.

### 9.3.14 "NO NAMES"

It has already been mentioned that all component names are "Software" generated and, as such, take a long time to draw out. The "NO NAME" facility suppresses all component names and so reduces the time taken to redraw the layout. A further call of "NO NAMES" re-sets the option such that they re-appear.

### 9.3.15 "NO NODES"

Node Numbers are also "Software" generated and can be suppressed using the "NO NODE" command. As with the "NO NAMES" facility (described in section 9.3.14), it can be reset with a second call and the Node Numbers will re-appear at the next re-drawing phase.

### 9.3.16 "PLACE" Component

The user can choose to bring an un-placed component onto the board at any stage in the design. He simply types the command "PLACE" and receives the message:

NAME AND ORIENTATION ?

He must then enter a component name and rotation angle (in any order) separated by a space or comma. The computer may respond with:

COMPONENT ALREADY PLACED !

in which case the attempt is terminated. Alternatively, the cursor segments are set up and the user indicates the position which he wishes the centre of the component rectangle to occupy. This is not crucial since he can later re-position the component with the "MOVE" facility (9.3.11).

When a character is pressed, the component is drawn onto the screen in the desired orientation and its new co-ordinates entered into the data structure.

If an invalid component name or rotation angle is entered, the computer will inform the user of his error. Another feature is that the rotation angle is taken as zero if it is not specified by the user. This saves time when using the facility.

### 9.3.17 "QUERY" A Component's Connections

Typing "QUERY", and indicating a component with the cursor, will cause all unformed connections to and from the component to be shown as dotted lines. This aids the user in deciding where to position the component to give a minimum connection length. The "SPRINGS" facility (section 9.3.25) can be regarded as automatically applying "QUERY" to every component on the board. Figure 9.16 demonstrates the output obtained in such a case.

### 9.3.18 "REDRAW" Layout

Since the screen cannot be selectively erased, it is necessary

to redraw the layout at intervals to free Menu area. This is done entirely automatically, but the user also has the option of refreshing the screen by typing the command "REDRAW" at any stage in the design. This facility is used to erase out of date information in order to unclutter the screen.

#### 9.3.19 "REMOVE" Component/Connection

The user can remove a component at any time by typing "REMOVE COMPONENT" and indicating the relevant bounding rectangle with the cursor. A diagonal line is drawn across the rectangle and the component will disappear when the circuit is next redrawn. This is demonstrated in Figures 9.13(a) and (b).

The component may be brought back onto the board at a later stage using the "PLACE" (9.3.16) or "NEXT" (9.3.12) facilities.

It is also possible to delete a conductor track by typing "REMOVE CONNECTION" and placing the cursor anywhere within the connection's boundaries. In this case (illustrated in Figure 9.14), a line is drawn up the centre of the track to indicate that it has been deleted. The connection is now irretrievably lost and its data space will be re-used by the next connection to be formed.

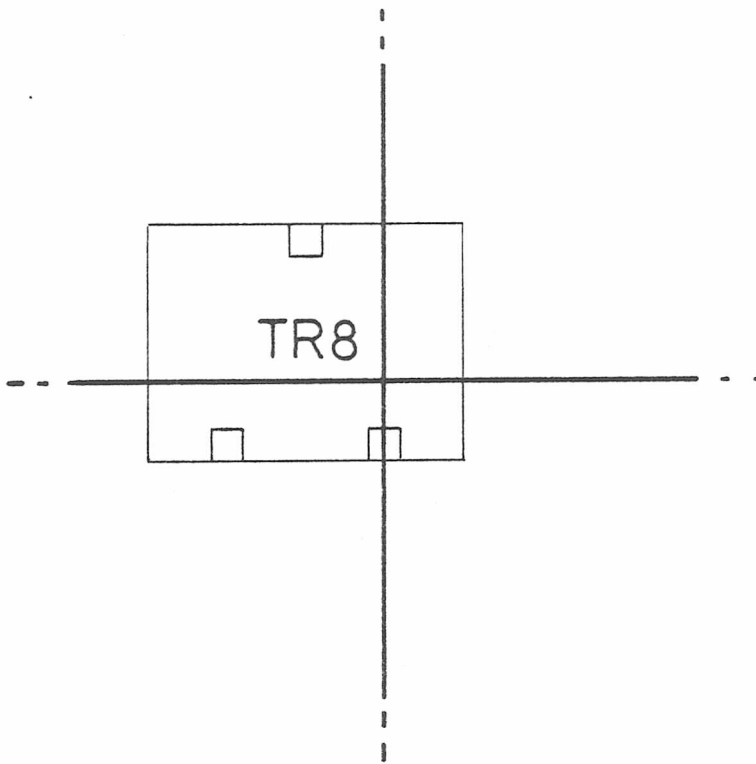
#### 9.3.20 "RESTART" Program

Like the "END" command (9.3.4), this facility updates the data structure file representing the layout. Instead of terminating the program, however, it requests the name of a new data file. Upon receipt of this data it copies the new layout into store and re-enters command mode. The user has effectively switched to another layout.

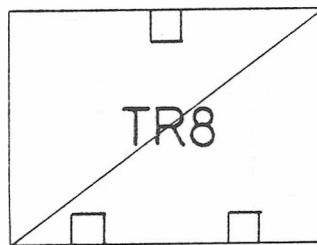
#### 9.3.21 "ROTATE" Component

Any component appearing on the board may be re-orientated using the "ROTATE" facility. Upon receipt of this command, the computer responds with:

ORIENTATION/NAME ?

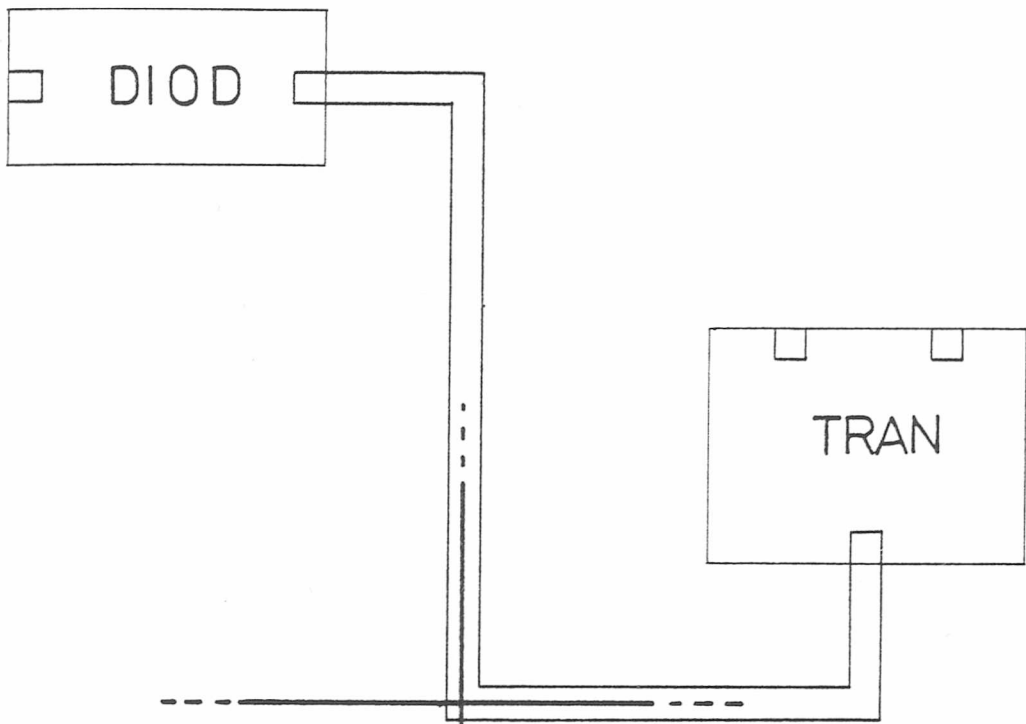


(a) *Cursor is positioned within component's bounding rectangle*

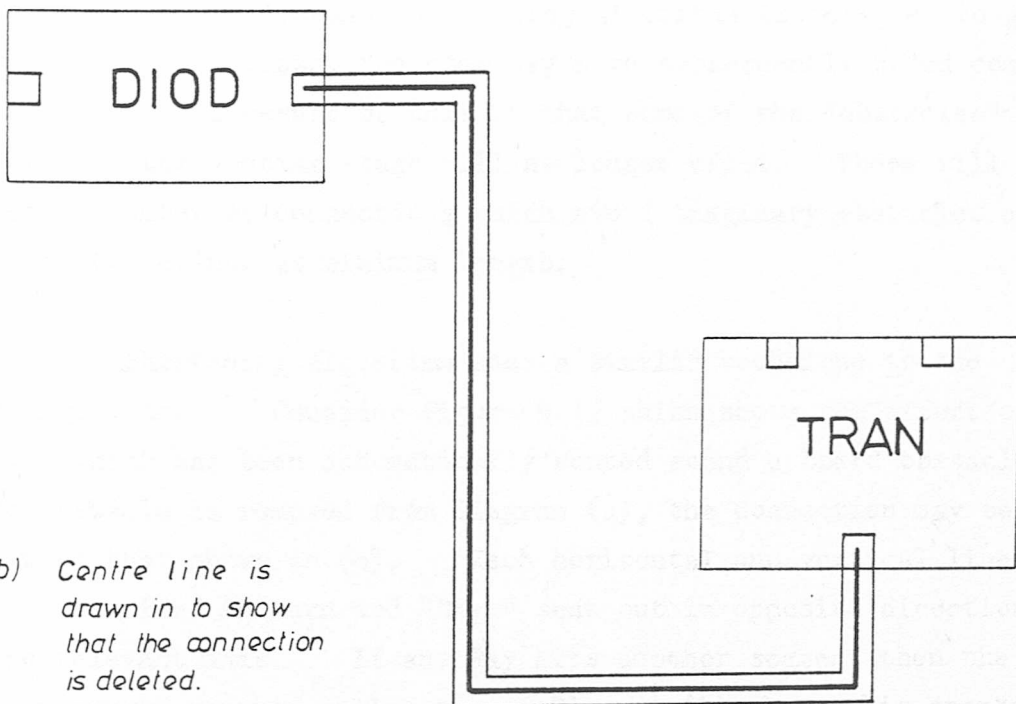


(b) *Transverse line is drawn in to indicate that component has been deleted*

Figure 9.13 Removing a component



(a) *Cursor is positioned anywhere within the connection outline.*



(b) *Centre line is drawn in to show that the connection is deleted.*

Figure 9.14 Removing a connection

and the user must enter the name of the component together with a rotation angle. Since the component must have all its sides perpendicular to the axes, only 0,  $\pm 90$ ,  $\pm 180$  and  $\pm 270$  degrees are valid. The computer will also check that the component name is valid, and appears on the board, before the user is allowed to proceed.

The cursor is next used to indicate the component's new location. When a character is pressed, the old component definition and any associated connections are immediately deleted. The component is then drawn in the desired orientation with its centre positioned at the point marked by the cursor.

### 9.3.22 "SHORTEN" A Connection

An algorithm has been developed to automatically reduce the overall length of a connection where possible. There are two distinct reasons for the existence of in-efficient connections:

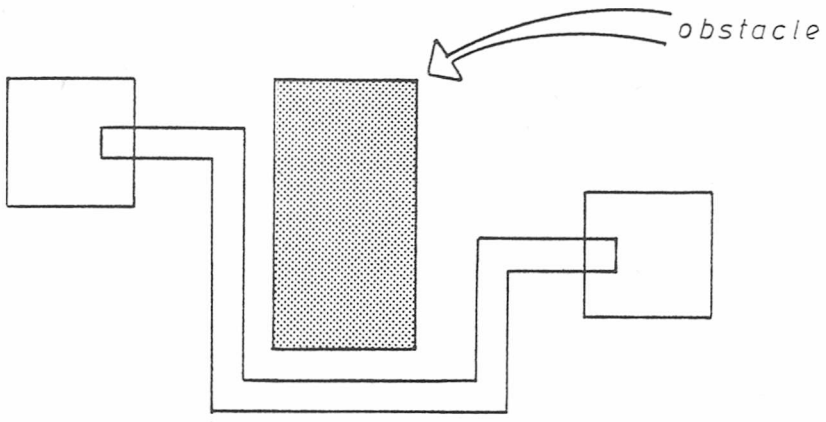
(a) The user has the ability to define conductor tracks interactively using the "MANUAL" facility (9.3.9). There is no guarantee that his effort is the shortest solution.

(b) Although the Routing Algorithm is designed to generate minimum length connections, the user may have subsequently moved components on the board. The result of this is that some of the "obstacles" encountered in the routing stage will no longer exist. There will be, in effect, a number of connections which avoid imaginary obstacles and as such cannot be defined as minimum length.

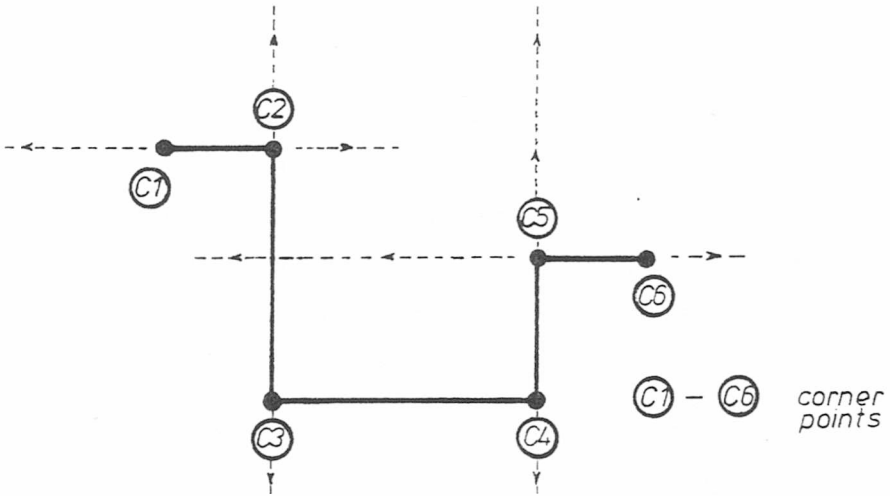
The Shortening Algorithm uses a similar technique to the Routing Algorithm. Consider Figure 9.15 which shows the effect on a connection which has been automatically routed round a board obstacle. When the obstacle is removed from diagram (a), the connection may be shortened to that shown in (c). Each horizontal and vertical line segment is examined in turn and "Rays" sent out in opposite directions along the relevant axis. If any Ray hits another segment then the connection can be reduced in length. Diagram (b) shows this graphically.

In this example, the Ray emitted to the left from corner C5 intersects with the vertical line segment between C2 and C3. It is now

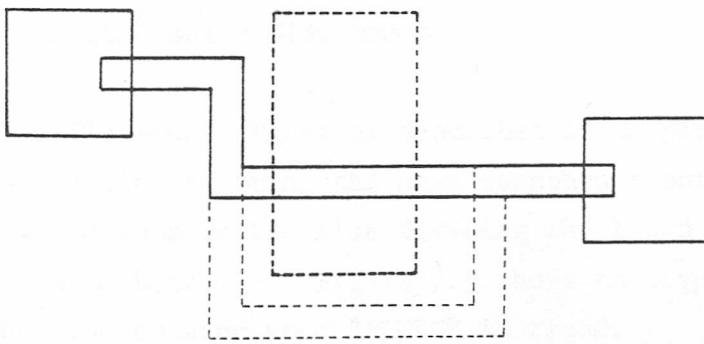




(a) original definition



(b) algorithm execution



(c) shortened definition

Figure 9-15 Shortening algorithm

possible to re-define the conductor track as shown in Diagram (c) by omitting corners C3 and C4.

The algorithm is continually re-applied until none of the Rays intersect with the connection. While this doesn't guarantee an absolute minimum length connection, it does provide a quick and fairly satisfactory solution.

To use this facility, the user types in the command "SHORTEN" and the cursor is immediately set up. He can then indicate the connection of his choice by positioning the cursor within the boundaries of the conductor track in question. When the algorithm has been applied, the connection is re-drawn over its original definition.

#### 9.3.23 "SHRINK" Board Area

Unwanted strips of board area can be deleted using the "SHRINK" facility. The user does this by positioning the cursor, pressing a character, then moving a single cursor control wheel and pressing another character. If the X control is selected, then the vertical strip of board between the two X co-ordinates is deleted and the layout re-drawn. Horizontal strips can be deleted in a similar manner using the Y control wheel.

The program will only delete a strip if there are no components, connection turning points or Edge Pads contained within it. When a legal strip has been defined, the board size is reduced in the relevant dimension. Each component, Edge Pad and connection is then examined and re-positioned if necessary before the layout is re-drawn.

#### 9.3.24 "SLOTS" Display Slot Boundary

The Placement Algorithm described in Chapter 7 uses a co-ordinate "Slot Boundary" line to bring the next components onto the board. This can be displayed as a dotted line dividing the board into free area above and occupied area below. Figure 7.3 shows the type of graphical output obtained when the command word "SLOTS" is typed.

#### 9.3.25 "SPRINGS" On Components

Each unrouted connection is displayed as a single dotted line

between the two components when "SPRINGS" is typed. Figure 9.16 gives an example of this.

The "SPRINGS" facility is very useful throughout the design. It gives a good guide to manual positioning and overall placement efficiency. It is also convenient in that it provides a speedy method of finding any un-routed connections that may otherwise have been overlooked.

### 9.3.26 "WINDOW" Layout

When the circuit is displayed on the terminal, the co-ordinates involved are automatically scaled up or down such that the whole layout fits the screen. For very large circuits it is necessary to view only a small portion of the layout at a time if the detail is too small to be seen. This can be achieved using the "WINDOW" or "ZOOM" facilities.

When the command word "WINDOW" is typed, the program clears the screen and returns with a message of the type:

```
"WINDOW DEFINED AS:  
BOTTOM LEFT CORNER 0,0  
TOP RIGHT CORNER 5000, 6000  
RE-DEFINE WINDOW OR RETURN FOR ORIGINAL SETTINGS"
```

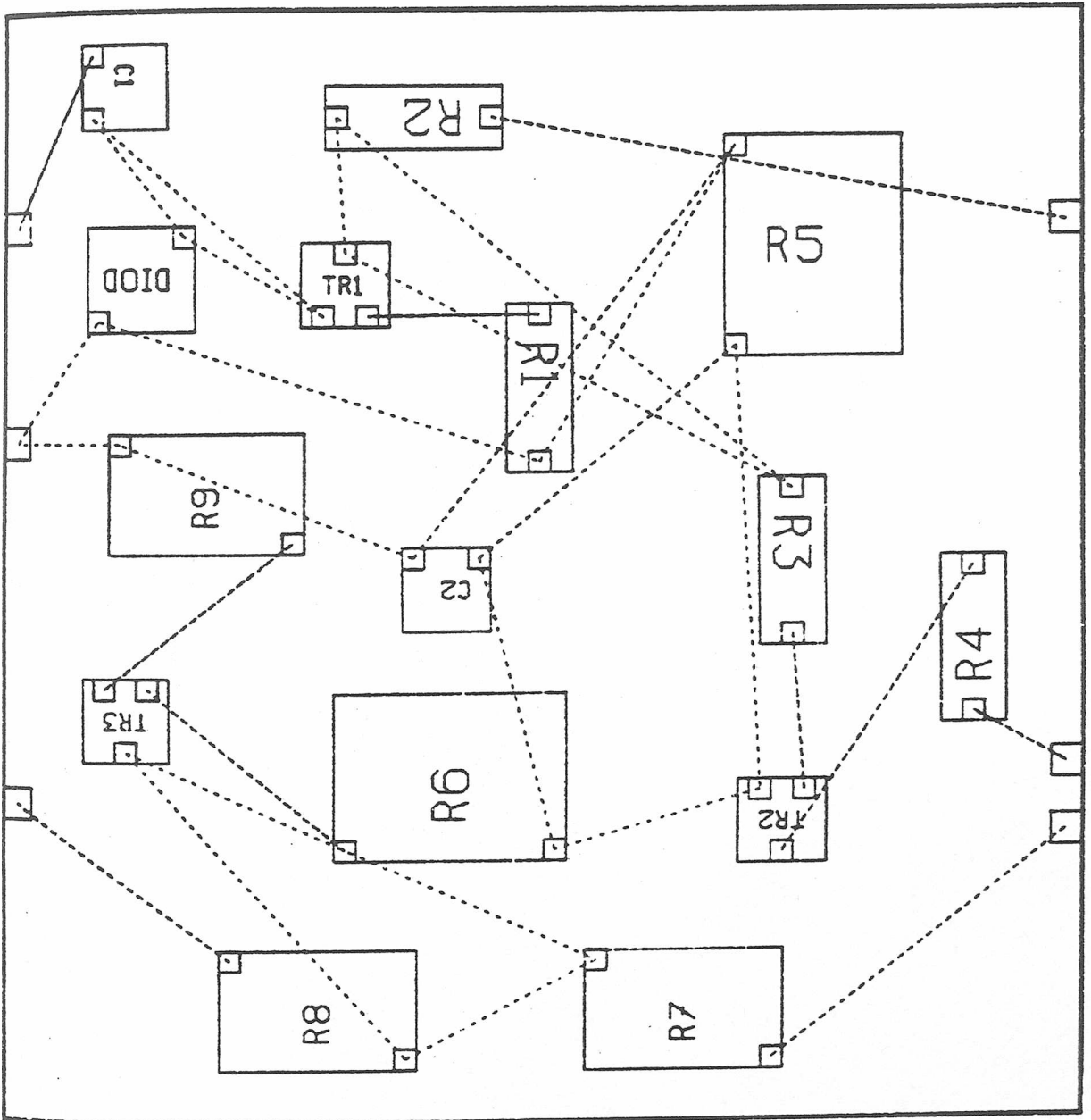
The figures given are in screen co-ordinates, each of which represent (un-scaled) 0.00005". The user can specify new co-ordinates or type a return. In the latter case, the window is set such that the whole layout is displayed.

Upon receipt of the new window values, the layout is re-drawn and control returns to command mode.

### 9.3.27 "ZOOM" IN OR OUT

The window can also be reset using the "ZOOM" facility. When the program receives this command, it responds with:

```
"Enter Zoom Factor"
```



What Next  
\$SPRINGS

Figure 9-16 The "SPRINGS" facility

and the user must type in a positive or negative number. The cursor is then set up and a point indicated by typing any character.

The layout is now re-drawn with the part of the layout which was indicated appearing at the centre of the screen. If the number was positive, the co-ordinates are all scaled up and the components will seem bigger -the user has apparently "ZOOMED IN" on a part of the layout. Conversely, a negative number will cause the co-ordinates to be scaled down and the components will seem smaller. In this case the user has apparently "Zoomed Out" to look at a larger section of the circuit.

Figure 9.17 shows an example using a positive zoom factor. This portion of the layout could also have been selected using the WINDOW facility described in Section 9.3.26.

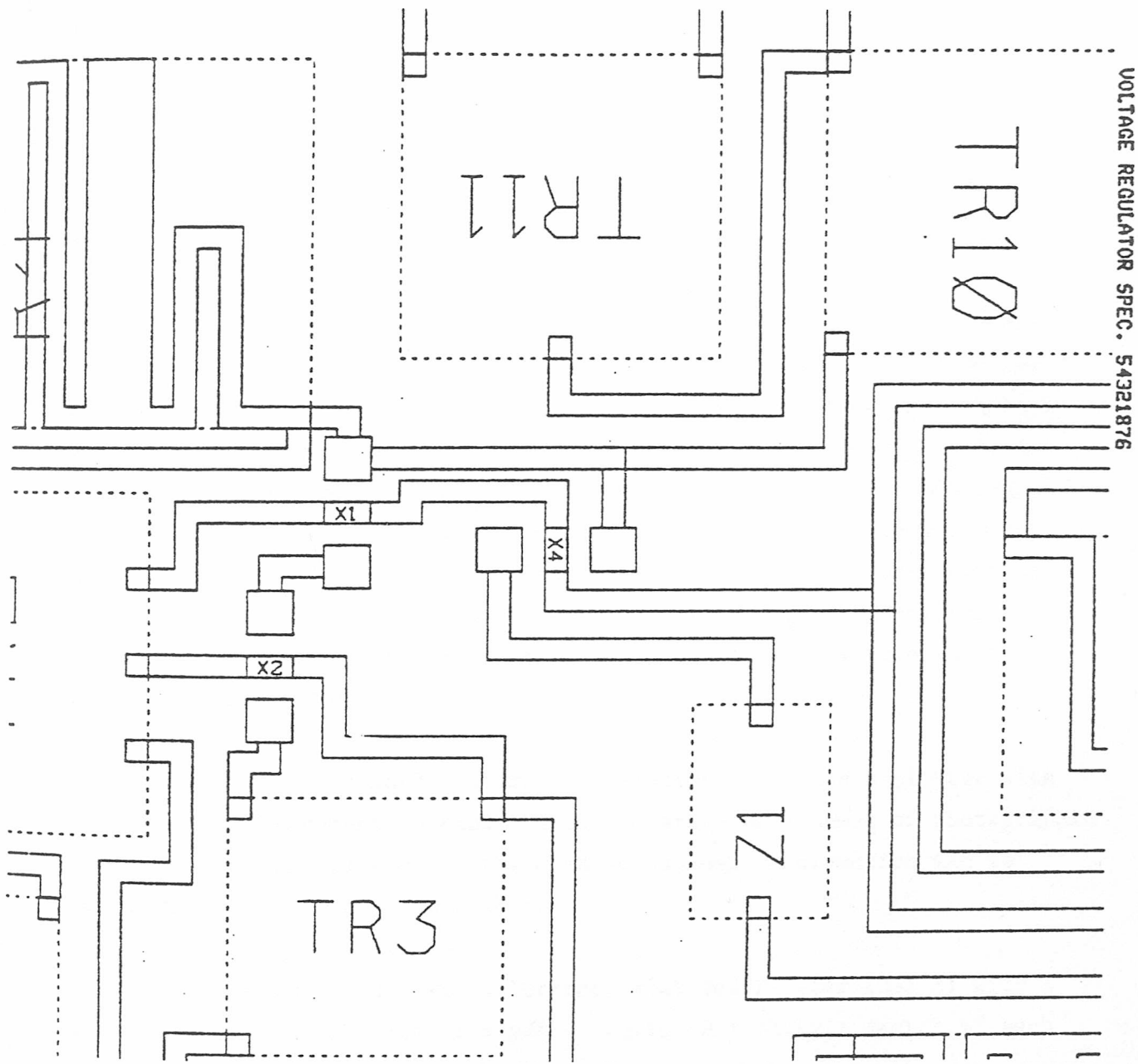


Figure 9-17 Effect of "WINDOW" or "ZOOM" facility when drawing layout

## Chapter 10

### Producing Photolithographic Masks Using A Tape-Controlled Photoplotter

#### 10.1 Introduction

A suite of programs to design thin film circuit layouts has already been described in previous chapters. The layouts produced by this method are stored as sets of co-ordinates in computer memory, and there remains the problem of fabricating the corresponding photolithographic masks needed to actually make the circuits. There are two main methods of doing this under computer control - the first uses a tape controlled co-ordinatograph with a knife attachment to cut out the required shapes in opaque plastic sheeting. This is a fairly simple programming task and there exist several plotter programs which ensure that the knife is correctly orientated at each cutting stroke. The major disadvantage with this method is that the unwanted strips of plastic must be manually removed, and there is also the possibility that the strips have not been entirely freed by the knife. Sometimes very fine strips are produced which are extremely difficult to remove by hand. This is, in any case, a time-consuming task better avoided.

An alternative method of mask generation is to use a plotter with a light beam attachment to expose selected areas on a sheet of photographic material. This eliminates the need for manual intervention and is the technique adopted in this case.

The equipment used was a Ferranti flat Bed Plotter (Ref 8) with a light beam fitted in place of a pen - Figure 10.1 shows a sketch of such a system.

A sheet of light-sensitive material is placed onto the flat plotting surface and exposed to light in the desired places. When developed, a translucent plastic sheet is obtained with black opaque patches where it has been exposed to the light. The size and shape of the beam can be altered at will by selecting one of the sixty-three apertures contained in a rotatable cartridge. Shapes may either be "flashed" in a particular location (as in the case of letters and numerals for example), or moved with the shutter open. Moving the light source with an open shutter, however, may cause problems of over-exposure for some aperture shapes.

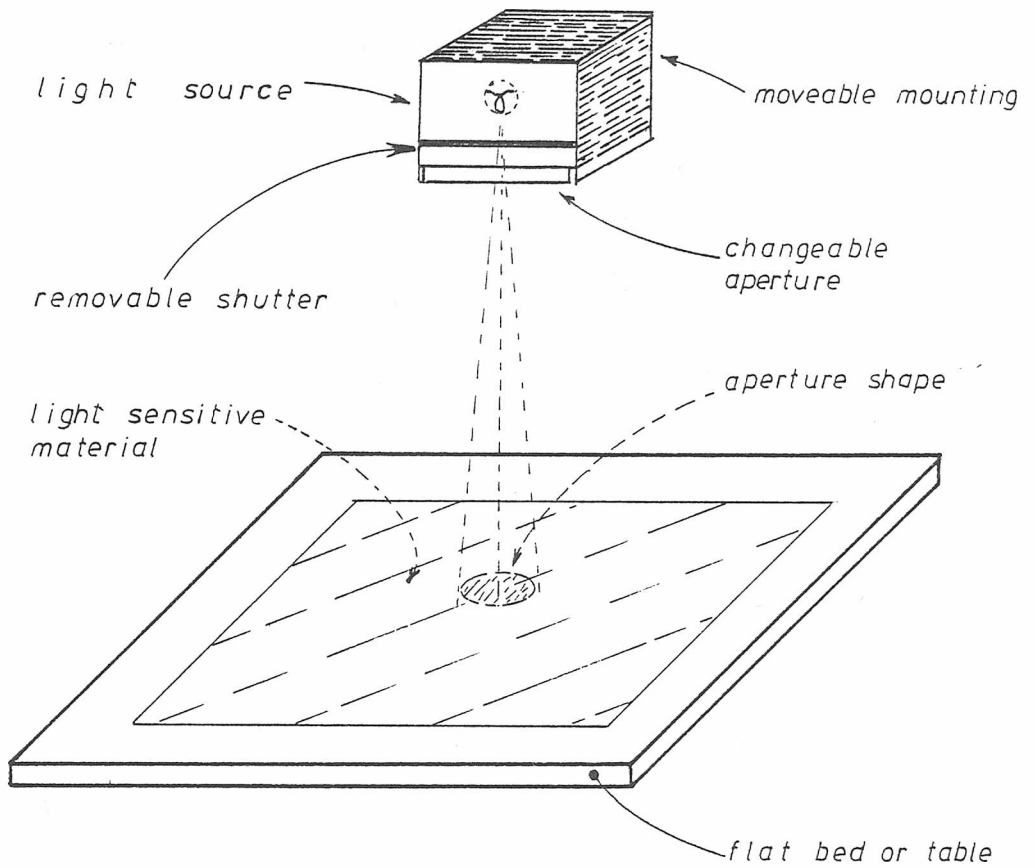


Figure 10.1 Elements of a Photoploter

HEXADECIMAL DIGIT	PRINTING CHARACTER
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	:
11	;
12	<
13	=
14	>
15	?

Figure 10.2 Table of co-ordinate printing characters



The plotter can be driven on-line by means of a dedicated support computer, or off-line by either 7/9 track magnetic tape or 8 hole paper tape. The latter method is used in this case, the coded paper tape being produced by the programs described in this chapter.

## 10.2 Co-ordinate Representation

The plotter has a co-ordinate range of -32,763 to +32,763 units. Each unit represents a thousandth of an inch, but the largest sheet of photographic material available is 37" x 50". Masks drawn on the plotter are usually photographically reduced to a twentieth of their original size, so the plotter unit is effectively 1/20th of a thou'. For this reason, the screen unit of the resistor design and automatic layout programs has also been chosen as 1/20th of a thou' so the resulting co-ordinate data can be used to drive the plotter directly.

The origin or "datum" of the co-ordinates may be positioned anywhere on the drawing area, but is usually set to the bottom left hand corner. Co-ordinates are represented on tape by a four digit hexadecimal number. In the hexadecimal system, the numbers in each column are in the range 0 to 15 instead of the usual 0 to 9. This enables the large co-ordinate numbers to be represented with only four digits. Single printing characters representing the numbers 10 to 15 are used according to the table in Figure 10.2. The conversion from a decimal co-ordinate to a four digit coding is demonstrated in Figure 10.3. This process is carried out automatically in the paper tape generating program to be described in Section 10.4.

## 10.3 Plotter Driving Codes

This section describes the code strings needed to drive the photoplotter. These are simply punched onto paper tape and fed into the machine.

All plotting commands consist of three characters and may or may not be followed by X and Y co-ordinate data. The first character is always a backslash "\" to indicate the beginning of a fresh set of instructions.

## Negative Number

e. g. -5768

Step 1 : Decrease modulus by 1

$$5768 \rightarrow 5767$$

Step 2 : Find Remainders

$$16/\underline{5767}$$

$$360 \quad r \quad \underline{7}$$

$$16/\underline{360}$$

$$22 \quad r \quad \underline{8}$$

$$16/\underline{22}$$

$$\underline{1} \quad r \quad \underline{6}$$

Step 3 : Subtract remainders from 15 to get hexadecimal equivalent

$$15 - 1 \rightarrow 14 \text{ (MSB)}$$

$$15 - 6 \rightarrow 9$$

$$15 - 8 \rightarrow 7$$

$$15 - 7 \rightarrow 8 \text{ (LSB)}$$

From Table in Figure 10.2 :

$$\text{CODING NEEDED} = \underline{\underline{879>}}$$

## Positive Number

e. g. + 600

Step 1 : Find Remainders

$$16/\underline{600}$$
$$37 \quad r \quad \underline{8}$$

$$16/\underline{37}$$
$$\underline{2} \quad r \quad \underline{5}$$

Step 2 : From Table in Figure 10.2 :

$$\text{CODING NEEDED} = \underline{\underline{0258}}$$

Figure 10.3 conversion of a Decimal Number to a four-digit hexadecimal coding

The next character determines whether the plot is to be ABSOLUTE or INCREMENTAL - i.e. whether the co-ordinate codes represent an actual point on the film, or whether they are simply increments in the X and Y directions. If the character is a "B" the plot is ABSOLUTE, and if it is an "@" the plot is INCREMENTAL.

The last character controls the light source. If it is "0", the light is OFF during the move. If, however, the character is a "2", the light is left ON. There is one more case-if it is an "=", then the light will be "FLASHED" after the move has been completed. The four digit co-ordinate codes are then added to the three control characters.

The final command string to be explained is the "change symbol" code. "\P2" followed by a symbol causes the aperture cartridge to be rotated until the desired shape is reached. A table of aperture codes can be found in Figure 10.4 together with the command strings explained above.

#### 10.4 Plotter Code Producing Program - "PHOTO"

Compiling a paper tape to drive the plotter is a long and tedious task if done by hand. A program called "PHOTO" has been written to automatically produce a data file containing the relevant plotting codes. A simple plotting language is used as a starting point for the translation to plotter code. Although these commands can be typed in during the program execution, it is better to make up a data file with the same information. In this way, should any mistakes be detected by the program, it is a simple matter to edit the data file. The commands used in the data file are explained below:-

ABS,X,Y DARK	Move to point X,Y
ABS,X,Y,LIGHT	Draw to point X,Y
ABS,X,Y,FLASH	Flash at point X,Y
REL,X,Y,DARK	Move by X,Y displacement
REL,X,Y,LIGHT	Draw by X,Y displacement
REL,X,Y,FLASH	Flash by X,Y displacement
SYM,N	Change to symbol no. N
COM	Comment on next line
BRD,X,Y	Definition of Board Size
END	End of Plot

command code string	meaning
\BO xxxx yyyy	<u>DARK MOVE</u> to point xxxx, yyyy
\B2 xxxx yyyy	<u>BRIGHT MOVE</u> to point xxxx, yyyy
\B= xxxx yyyy	<u>FLASH</u> at point xxxx, yyyy
\@O xxxx yyyy	<u>DARK INCREMENTAL</u> move of xxxx, yyyy
\@2 xxxx yyyy	<u>BRIGHT INCREMENTAL</u> move of xxxx, yyyy
\@=xxxx yyyy	<u>DARK INCREMENTAL</u> move of xxxx, yyyy, then <u>FLASH</u>
\P2 c	change aperture to that with coding "c" (see Table below)

SYMBOL SIZE	TYPE	CALL	POSITION	SYMBOL SIZE	TYPE	CALL	POSITION	SYMBOL SIZE	TYPE	CALL	POSITION
.187	Target Dowel	\P2@	0	U	Reverse Char. .070 High	\P2U	21	.100	Lands Square	\P2*	42
A	Reverse Char. .070 High	\P2A	1	V		\P2V	22	.020	Track	\P2+	43
B		\P2B	2	W		\P2W	23	.025	Track	\P2.	44
C		\P2C	3	X		\P2X	24	.030	Track	\P2-	45
D		\P2D	4	Y		\P2Y	25	.040	Track	\P2.	46
E		\P2E	5	Z		\P2Z	26	.050	Track	\P2/	47
F		\P2F	6	.005	\P2C	27	0	Reverse Char. .070 High	\P20	48	
G		\P2G	7	.010	\P2\	28	1		\P21	49	
H		\P2H	8	.0125	\P2□	29	2		\P22	50	
I		\P2I	9	.015	\P2↑	30	3		\P23	51	
J		\P2J	10	.030	\P2←	31	4		\P24	52	
K	\P2K	11	.050	\P2	32	5	\P25		53		
L	\P2L	12	.060	\P2!	33	6	\P26		54		
M	\P2M	13	.075	\P2"	34	7	\P27		55		
N	\P2N	14	.100	\P2#	35	8	\P28		56		
O	\P2O	15	.125	\P2\$	36	9	\P29		57		
P	\P2P	16	.150	\P2%	37	.060	Track	\P2:	58		
Q	\P2Q	17	.200	\P2&	38	.075	Track	\P2;	59		
R	\P2R	18	.050	\P2'	39	.100	Track	\P2<	60		
S	\P2S	19	.060	\P2(	40	.125	Track	\P2=	61		
T	\P2T	20	.075	\P2)	41	.150	Track	\P2>	62		
						.200	Track	\P2?	63		

e.g. **B** character      ● circular land      ○ track      ■ square land      ⊕ target dowel

Figure 10.4 Tables of command strings and aperture codes

If a line is encountered within the data file which fits none of the above commands, a message of the type:-

#### ERROR IN LINE NUMBER 7

is printed and the program halts. This prevents a typing mistake producing an error in the resulting paper tape. In this example the user would check line 7 for an error.

A typical data file is shown in Figure 10.5 together with the code file produced. It is obvious that the compilation of even this simple code file would be time consuming without the use of the program. The on-line method is shown in Figure 10.6 for comparison. Note that much more computer time is used compared to the data file method.

When the required code file has been produced, a paper tape can be made by simply listing the file with the tape punch switched on. An "end of plot" character (CNTRL/D) is automatically added at the end of the tape to tell the plotter that the drawing is finished.

### 10.5 Straight Line Polygons On The Photoplotter

When making a thin film mask using the photoplotting technique, it is necessary to produce quite complex geometric shapes. This should be achieved by systematically exposing the shape to light employing the minimum number of aperture changes in the fastest possible time. Since there are an infinite number of possible shapes, it is sensible to break each one into its component triangles, rectangles and trapeziums. These simple shapes can then be individually processed and "filled in" by the light beam in an optimal manner.

### 10.6 Rectangle Forming Algorithm

The rectangle is by far the easiest shape to process when photoplotting, and any "paraxial" polygon may be completely defined by this shape. "Paraxial" implies that all the edges are parallel to either the X or the Y axis like the meandering resistors described in Chapter 3.

A program to automatically break a paraxial polyson into constituent rectangles was written and tested on a wide range of shapes - Figure 10.7 shows a flowchart of the algorithm involved. To illustrate the process

title of plot → EXAMPLE OF PHOTO.FOR OPERATION  
 SYM,36  
 COM  
 THIS IS A COMMENT !  
 ABS,500,600,DARK  
 REL,100,0,LIGHT  
 ABS,100,200,FLASH  
 END

(Data  
Input  
file)

PHOTO-PLOTTER TAPE GENERATOR  
 - COMMANDS ARE :- "END", "SYM", "REL" AND "ABS"

(use of  
Program  
"Photo")

INPUT NAME FOR DATA OUTPUT FILE - TAPE

INPUT NAME OF SOURCE FILE OR RETURN DRIVE

END OF EXECUTION  
 CPU TIME: 0.01 ELAPSED TIME: 00.02  
 EXIT

EXAMPLE OF PHOTO.FOR OPERATION  
 \P2\$\B04?108520\@246000000\B=46008<00

(Data  
Output  
file)

Figure 10.5 Use of "PHOTO.FOR" with a data drive file

PHOTO-PLOTTER TAPE GENERATOR  
- COMMANDS ARE :- 'END', 'SYM', 'REL' AND 'ABS'

INPUT NAME FOR DATA OUTPUT FILE - TAPE

INPUT NAME OF SOURCE FILE OR RETURN

TYPE IN DESCRIPTIVE TITLE  
EXAMPLE OF PHOTO.FOR OPERATION

COMMAND ? SYM  
ENTER SYMBOL NUMBER - 36

COMMAND ? ABS  
INPUT X AND Y VALUES ? 500,600  
DARK OR LIGHT LINE OR FLASH ? DARK

COMMAND ? REL  
INPUT X AND Y VALUES ? 100,0  
DARK OR LIGHT LINE OR FLASH ? LIGHT

COMMAND ? ABS  
INPUT X AND Y VALUES ? 100,200  
DARK OR LIGHT LINE OR FLASH ? FLASH

COMMAND ? FUTTR  
INPUT X AND Y VALUES ?  
DARK OR LIGHT LINE OR FLASH ?

COMMAND NOT UNDERSTOOD

COMMAND ? END

END OF EXECUTION  
CPU TIME: 1.52 ELAPSED TIME: 1:18.11  
EXIT  
®

EXAMPLE OF PHOTO.FOR OPERATION  
^P2\$^B04?108520^0246000000^B=46008<00

Figure 10.6 On-line use of the program "PHOTO.FOR"

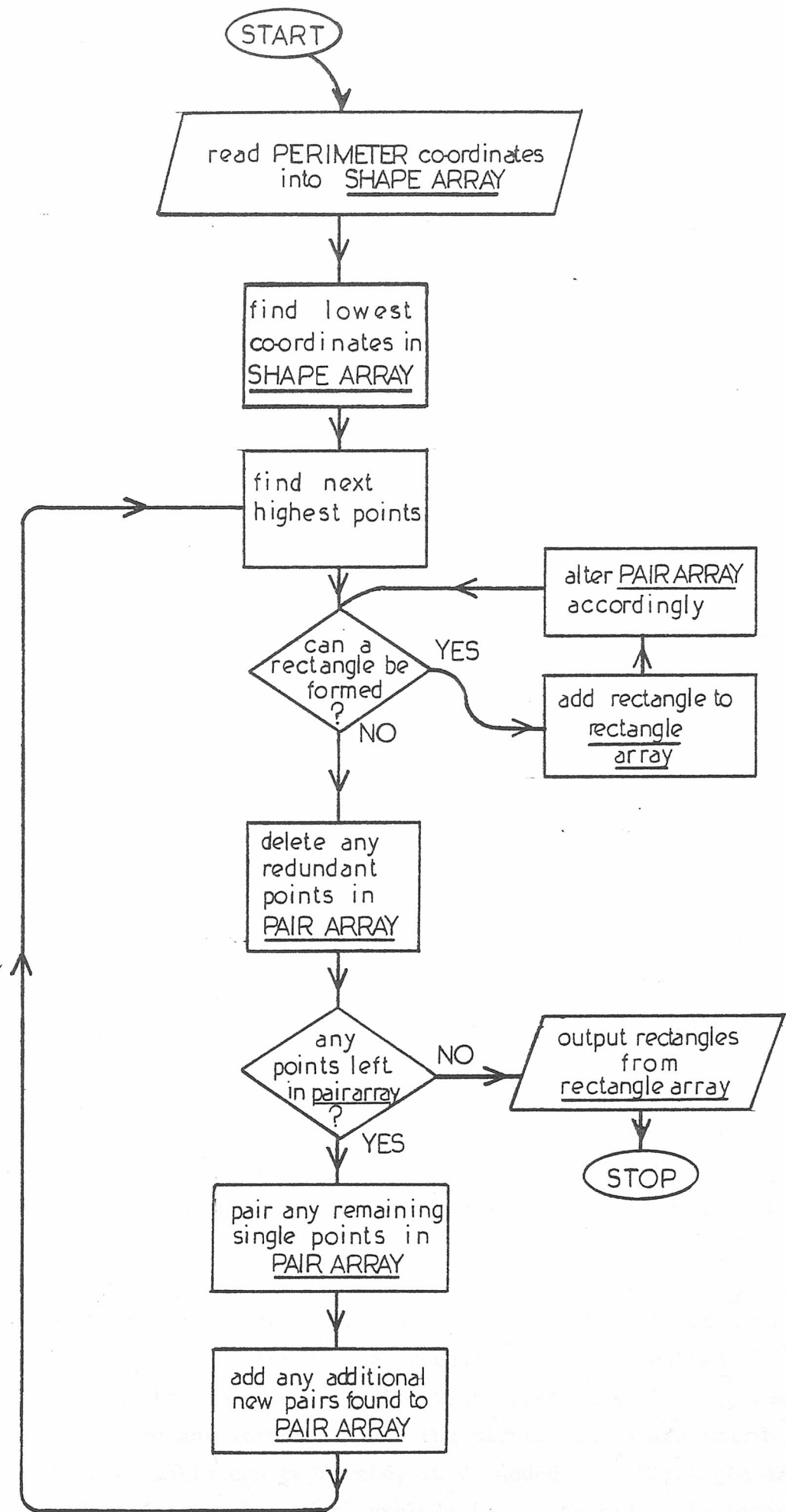


Figure 10-7 Rectangle forming program



an arbitrary shape is shown in Figure 10.8 and redrawn as the rectangles are formed.

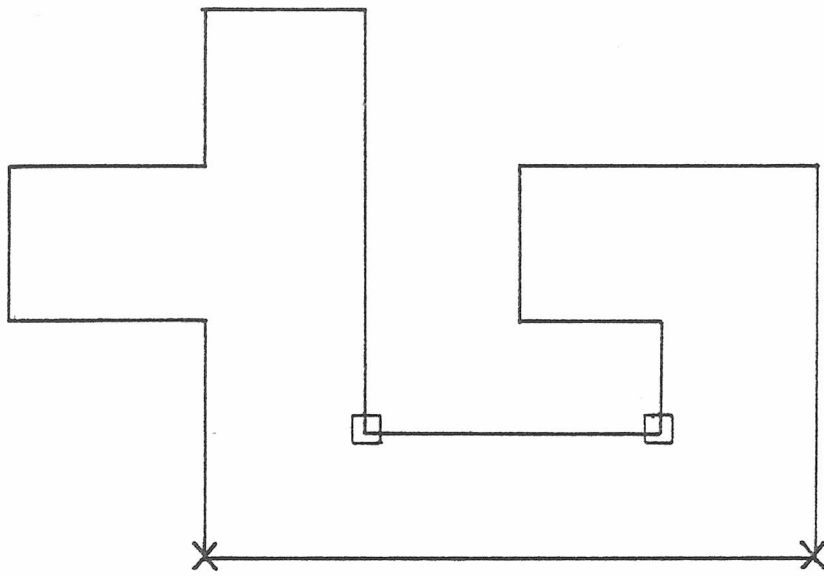
The polygon is initially stored as a set of corner points in the "Shape Array". The next step is to find the lowest pair (or pairs) of co-ordinates in the shape description. Having stored these values in a matrix called "The Pair Array", the next highest points are found. This stage is shown in diagram (a). There are two cases where a rectangle can be formed - the first is when one of the newly found points is directly above a point in The Pair Array (diagram (b)), and the second is when two new points lie completely within two points in The Pair Array (as in (a)). In both cases the rectangle co-ordinates are entered into a matrix called "The Rectangle Array", and The Pair Array is re-defined.

The process repeats itself and new rectangles are formed. Figure 10.8(c) shows the situation where redundant corner points have appeared in The Pair Array. These points must be deleted from the array before continuing. The process halts when all the co-ordinates in The Pair Array become redundant - diagram (f) shows the completed process where all possible rectangles have been formed.

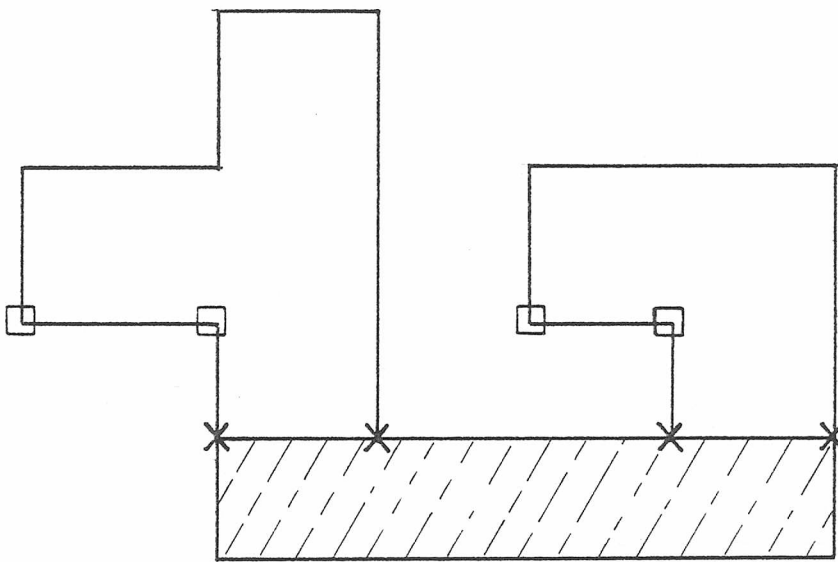
#### 10.7 Straight-Line Polygon Splitter - "Cutter"

The program discussed in the last section can only break up polygons whose sides are parallel to the X or Y axis. Shapes with sloping sides are quite common in thin film masks, so we need to extend the program to deal with any straight-lined polygon. Program "CUTTER" was written for this purpose and tested on a wide range of shapes - concentrating mainly on patterns which are commonly found in thin film circuitry. The program incorporates the rectangle forming method but is overall rather more complicated. The flowchart for the basic splitting process is given in Figure 10.9.

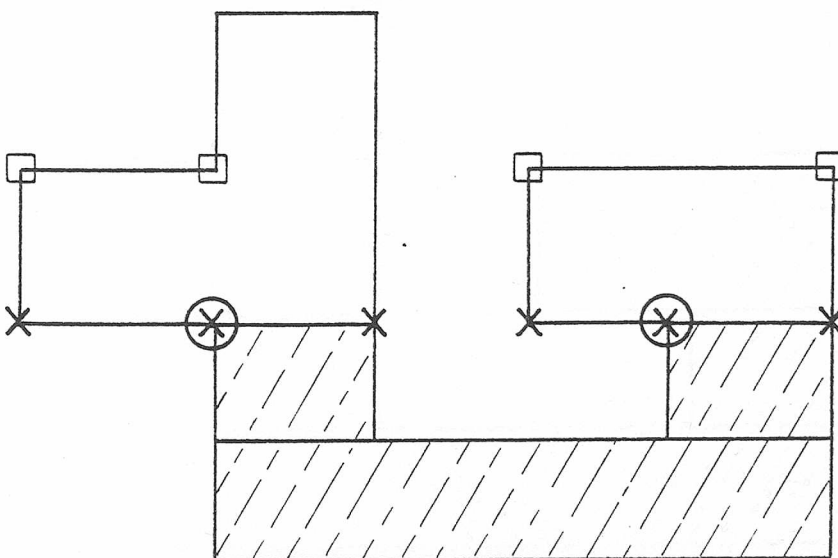
The first step is to read the co-ordinates of the polygon into the "Shape Array" as before. Due to the processing methods adopted, it is necessary to define the polygon in a CLOCKWISE direction, although the starting point can be any vertex. All the sloping lines are examined in turn. If a triangle can be formed, it is added to a "Triangle Array" and the shape description altered to exclude it. Should it be impossible to form a triangle, the sloping line segment is stored and the shape description converted to be paraxial at that point. Figure 10.10(a).



(a)



(b)

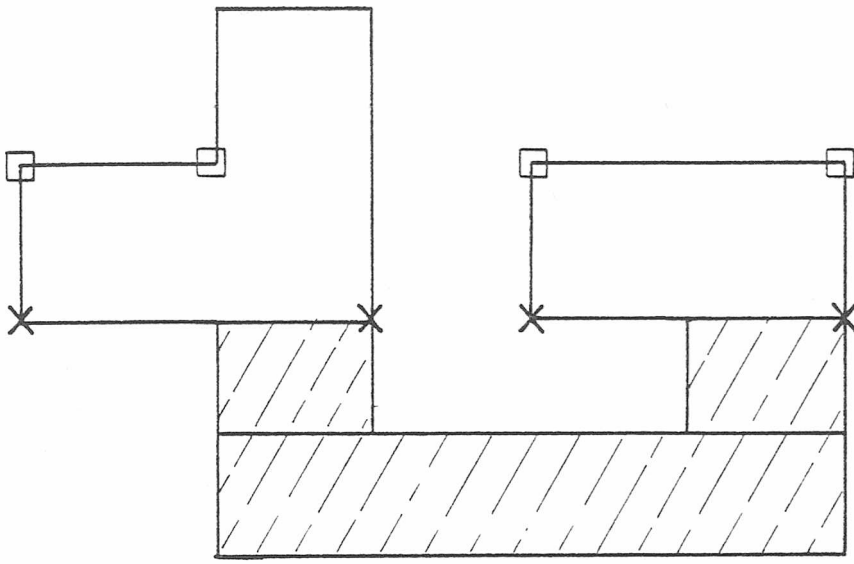


(c)

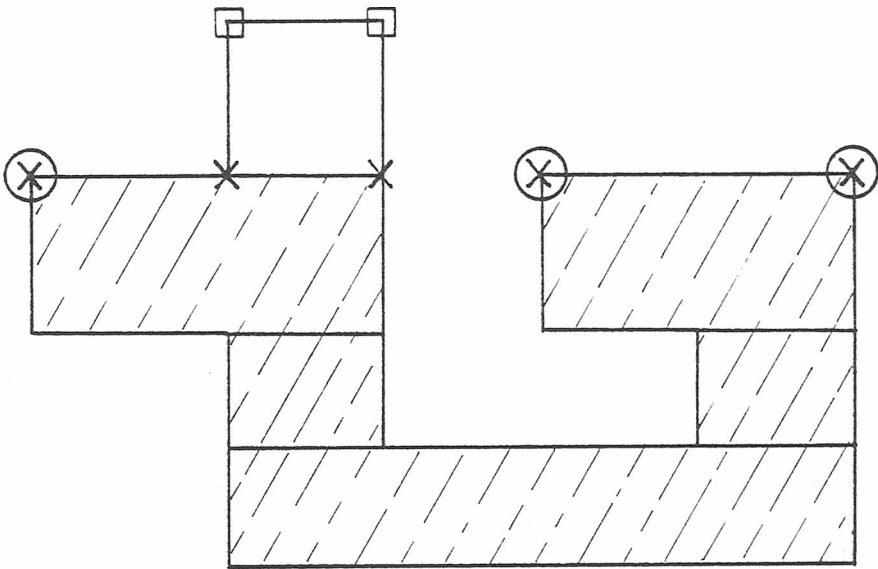
over

- X - Point in "PAIR ARRAY"
- - Next highest points in "SHAPE ARRAY"
- ⊗ - Point in (PAIR ARRAY) which is redundant
- Rectangles generated

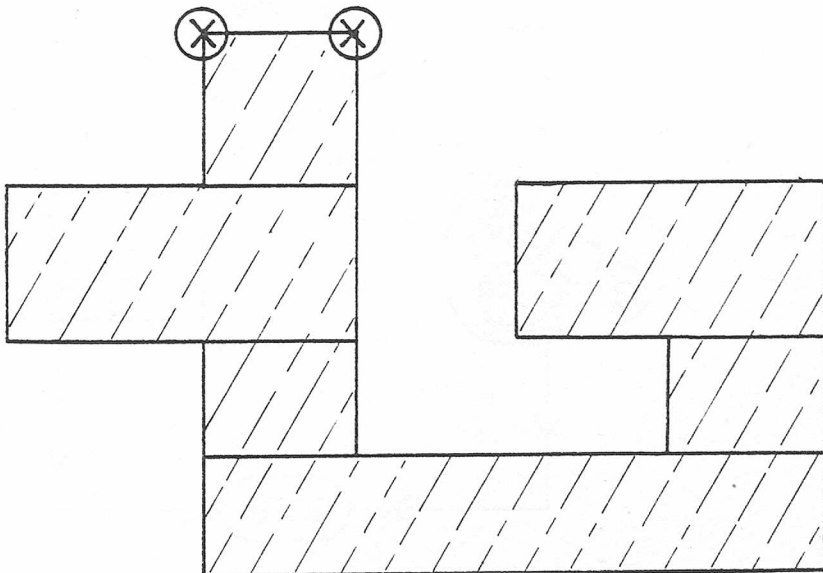
Figure 10.8 Short Polygon Rectangle Generation



(d)



(e)



(f)

Figure 10.8 (continued)

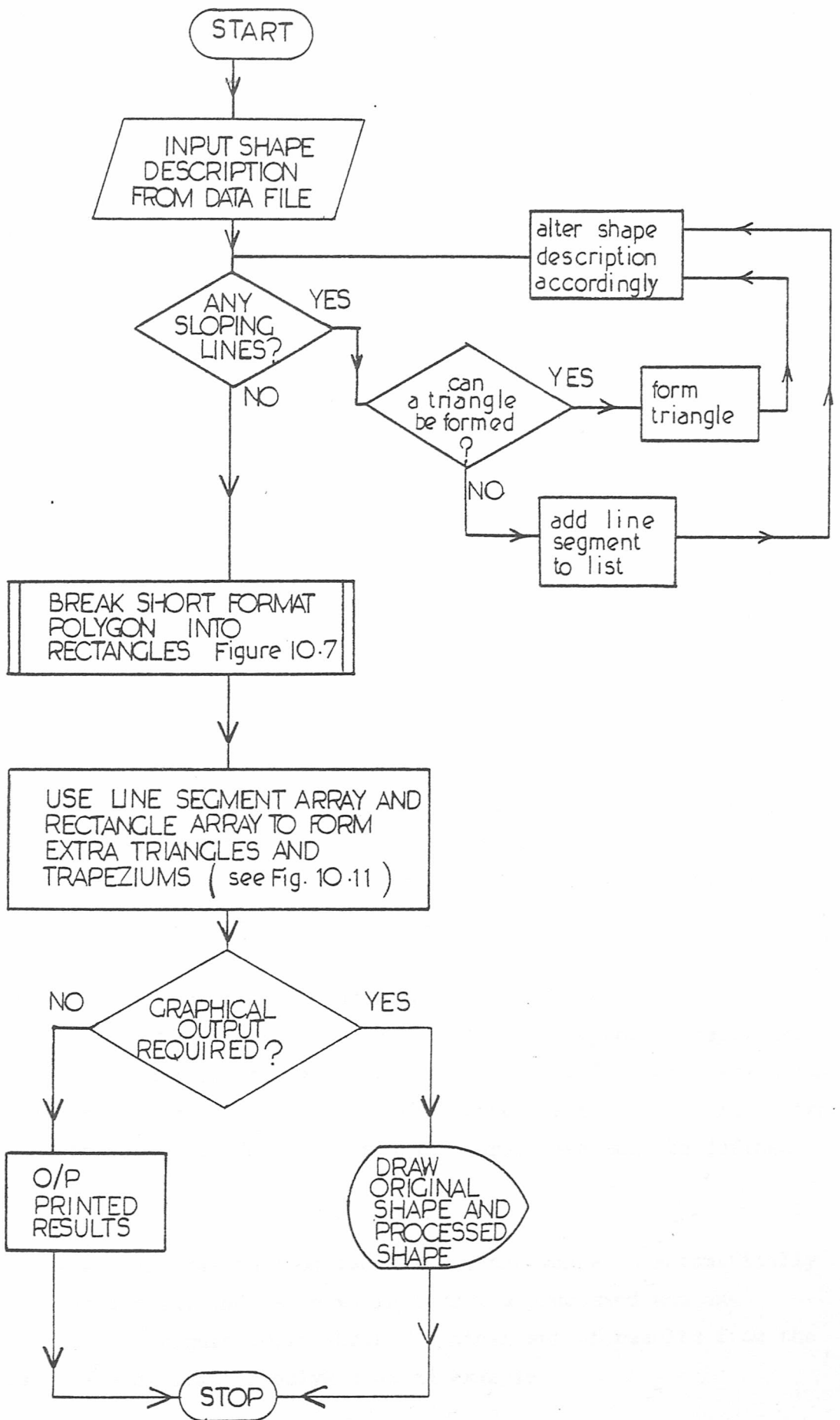


Figure 10.9 Flowchart of basic splitting algorithm

illustrates this technique where triangles are generated or sloping lines are stored for later processing. It can be seen that the converted shape description is now in paraxial form.

The rectangle splitting method is next applied, resulting in Figure 10.10 (b). In this example there are now 1 triangle, 1 sloping line segment and 4 rectangles in storage. To deal with the remaining sloping lines, it is necessary to compare the "Rectangle Array" with the "Line Segment" array. There are 10 ways in which right-angled triangles and trapeziums can be formed from these arrays and they are demonstrated in Figure 10.11. This part of the program is by far the most complex.

Figure 10.10 (c) shows the situation where 4 rectangles and 3 triangles have been created to define the polygon in question. This is an acceptable solution, but a consideration of the "filling" methods to be used suggests that trapeziums are much easier to process than triangles and, indeed, take less plotting time. Further processing is therefore used to combine triangles and rectangles where possible into trapeziums. Figure 10.12 shows four examples where this technique can be applied.

A further process is used to combine stacked or adjacent geometries which are produced as peculiarities of the splitting algorithm. This is demonstrated in Figure 10.13. A flowchart showing the complete splitting program "CUTTER" is given in Figure 10.14.

The user has the choice of "Numerical" or "Graphical" output. In the Numerical output mode (see Figure 10.15), the component shapes are defined by the co-ordinates of their corners. To avoid unnecessary data, the rectangles need only be defined by their bottom left and top right corners. Similarly, the trapezium's top and bottom Y co-ords need only be defined once.

When using Graphical mode on the V.D.U., the shape is automatically scaled to fit the screen, and is shown in both its processed and un-processed forms. Figure 10.16 shows a typical set of results from the algorithm using a non-paraxial polygon as an example.

## 10.8 The Use Of Different Aperture Shapes

Several test tapes were compiled and run on the Ferranti flat bed photoplotter to gauge the usefulness of each available aperture shape.

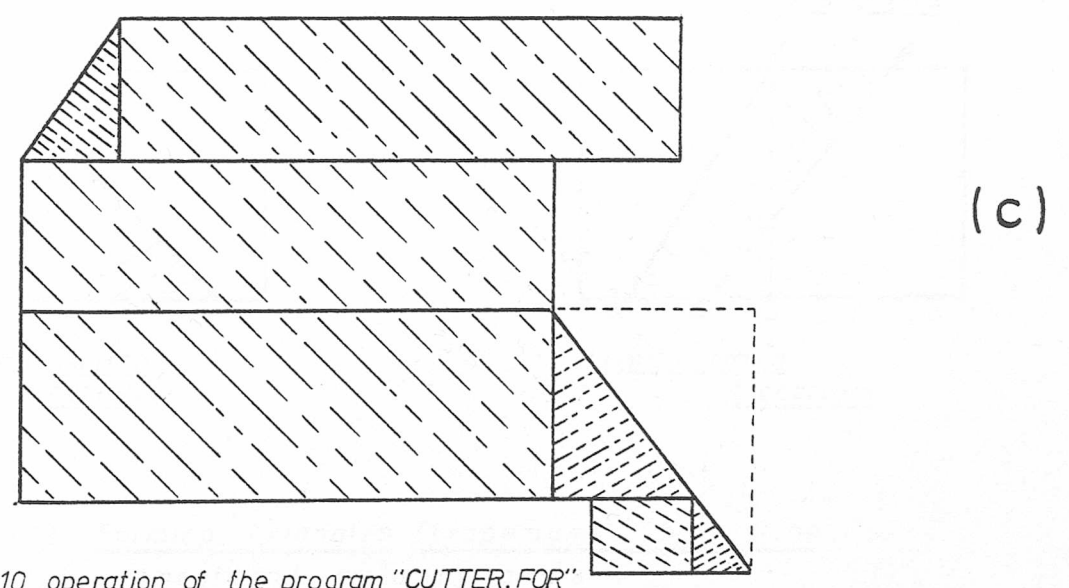
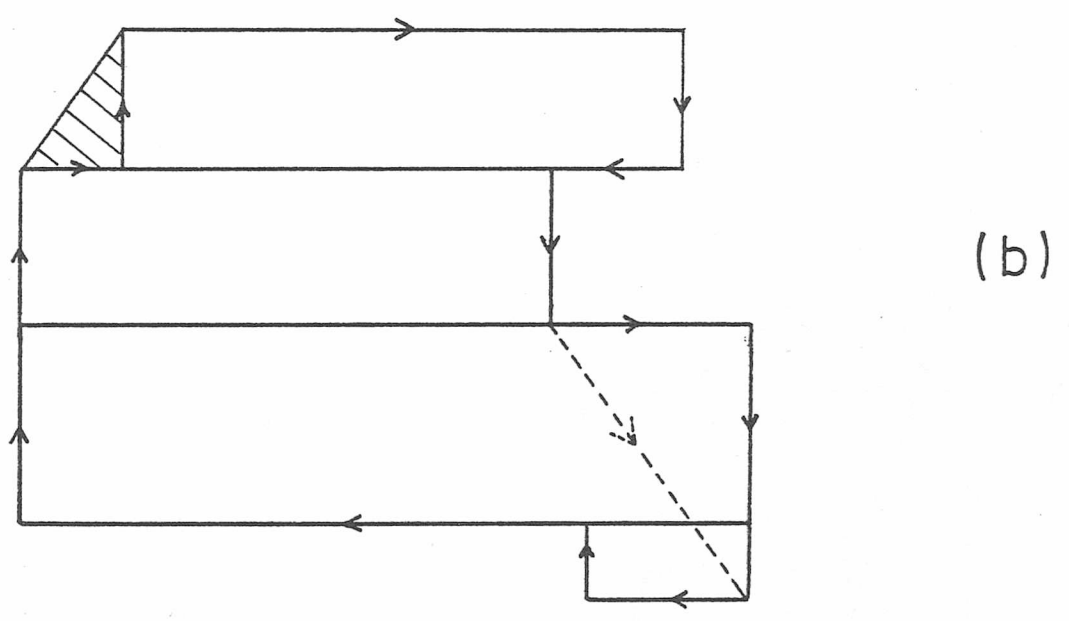
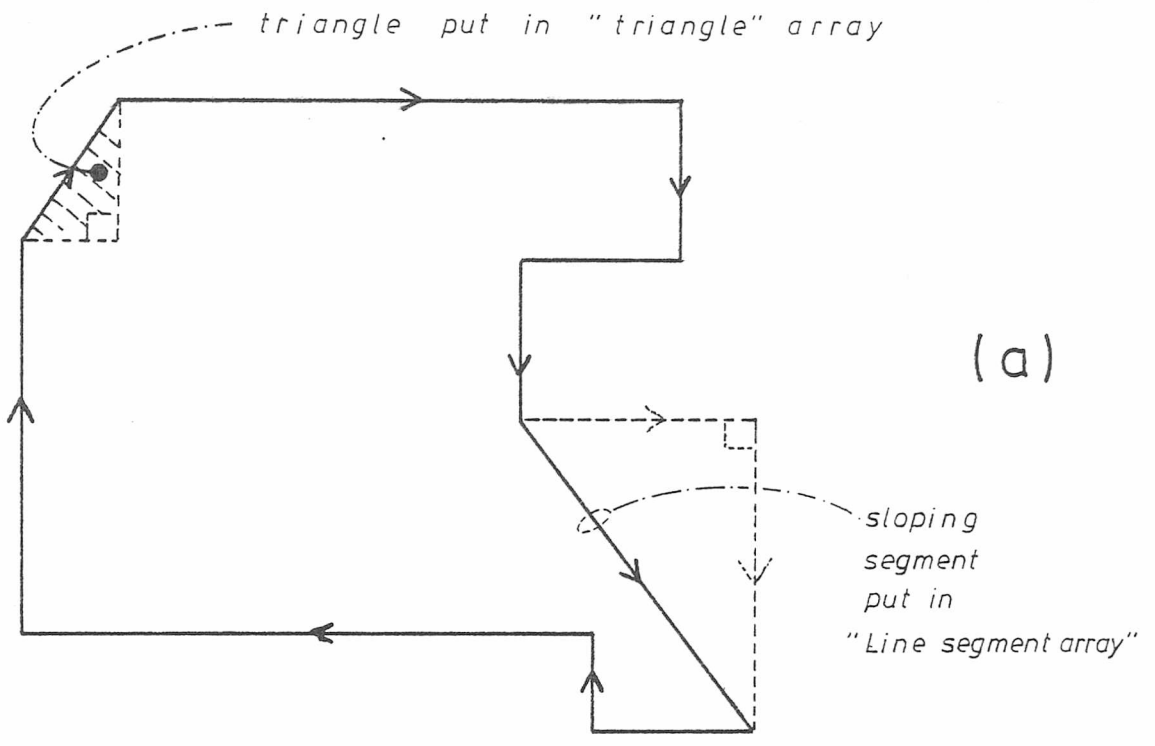
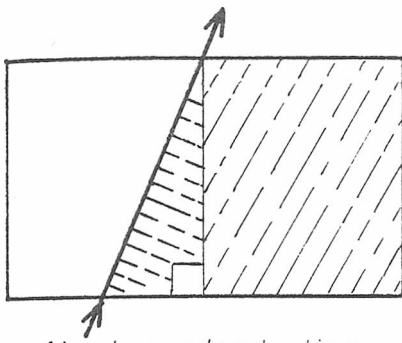
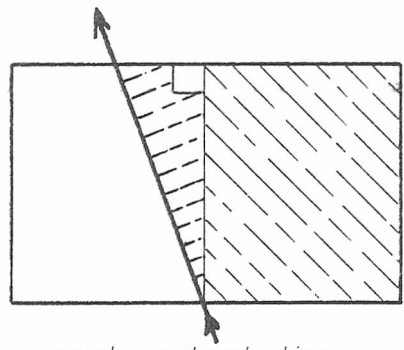


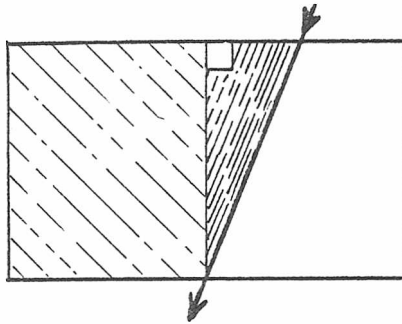
Figure 10.10 operation of the program "CUTTER.FOR"



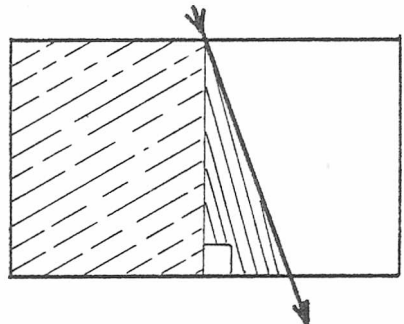
first quadrant line segment



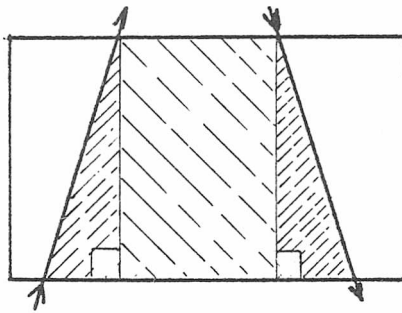
second quadrant line segment



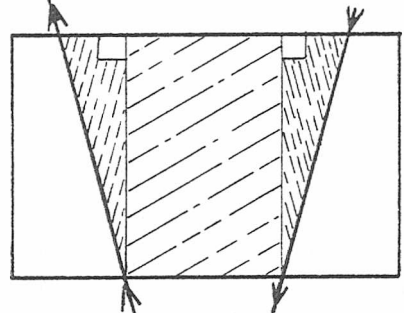
third quadrant line segment



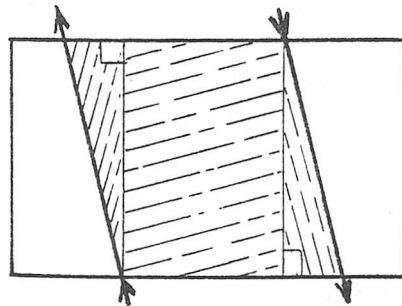
fourth quadrant line segment



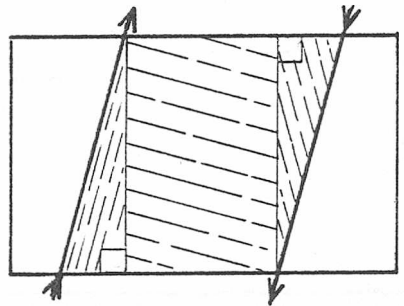
first and fourth quadrant line segments



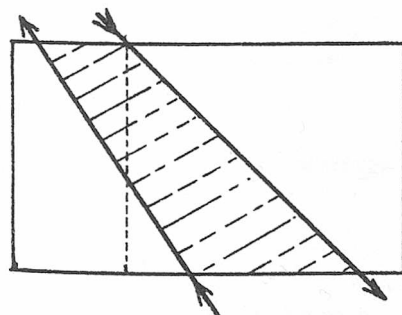
second and third quadrant line segments



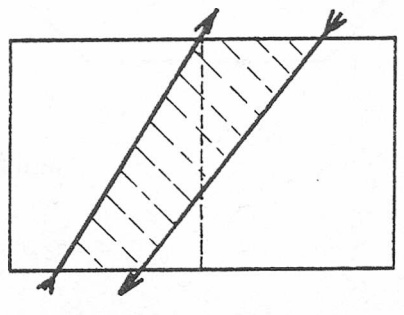
second and fourth quadrant line segments



first and third quadrant line segments

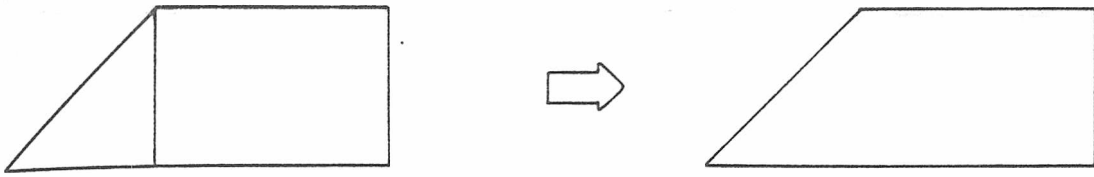


second / fourth trapezium

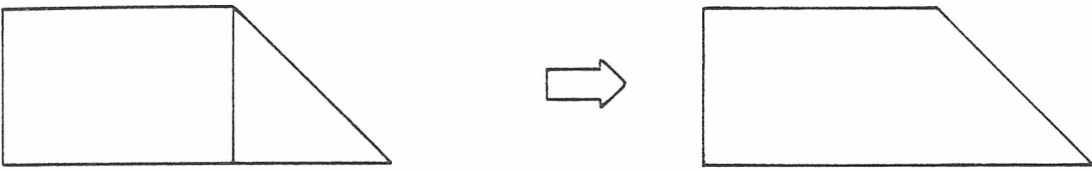


first / third trapezium

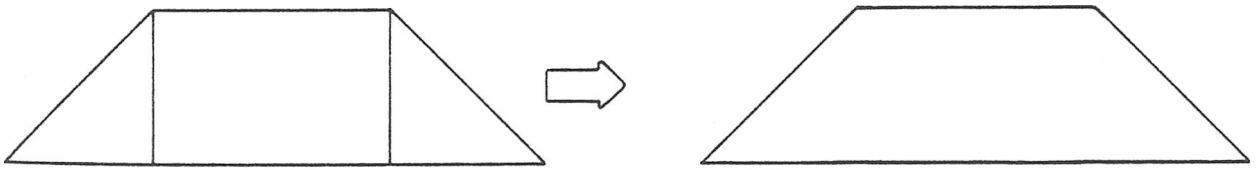
Figure 10.11 Forming triangles / trapeziums from line segment and rectangle arrays



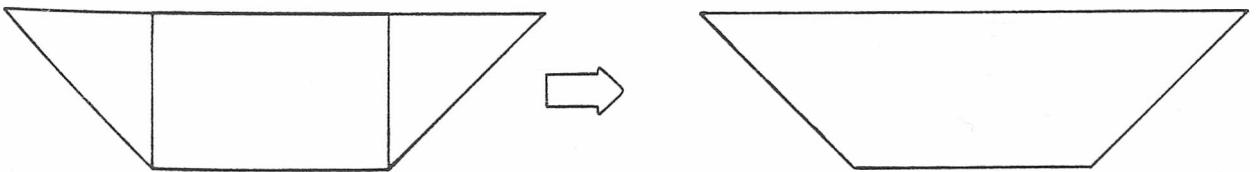
(a) *triangle + rectangle* → *trapezium*



(b) *rectangle + triangle* → *trapezium*



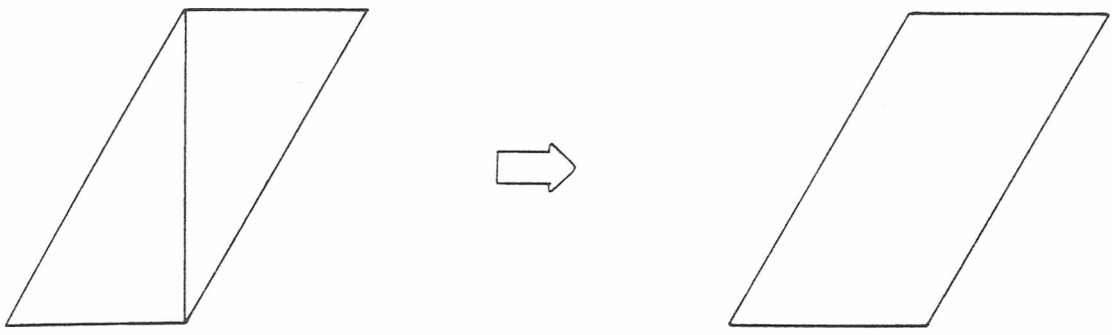
(c) *triangle + rectangle + triangle* → *trapezium*



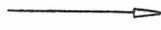
(d) *triangle + rectangle + triangle* → *trapezium*

Figure 10.12 Combining rectangles and triangles

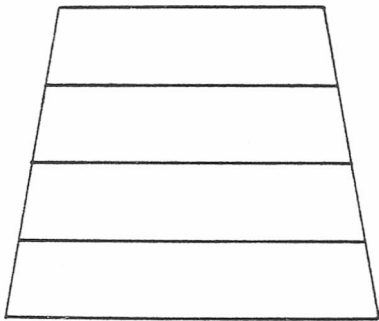




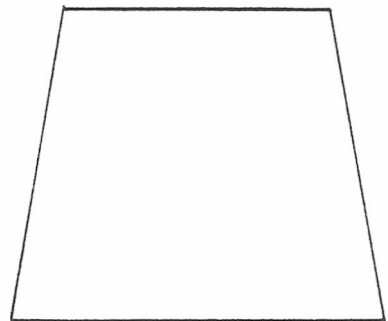
(a) adjacent triangles



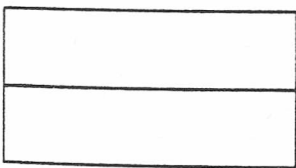
trapezium



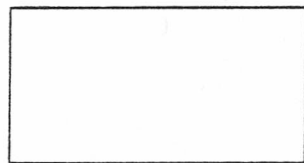
(b) stacked trapeziums



1 trapezium



(c) stacked rectangles



1 rectangle

Figure 10.13 Further processing of geometries

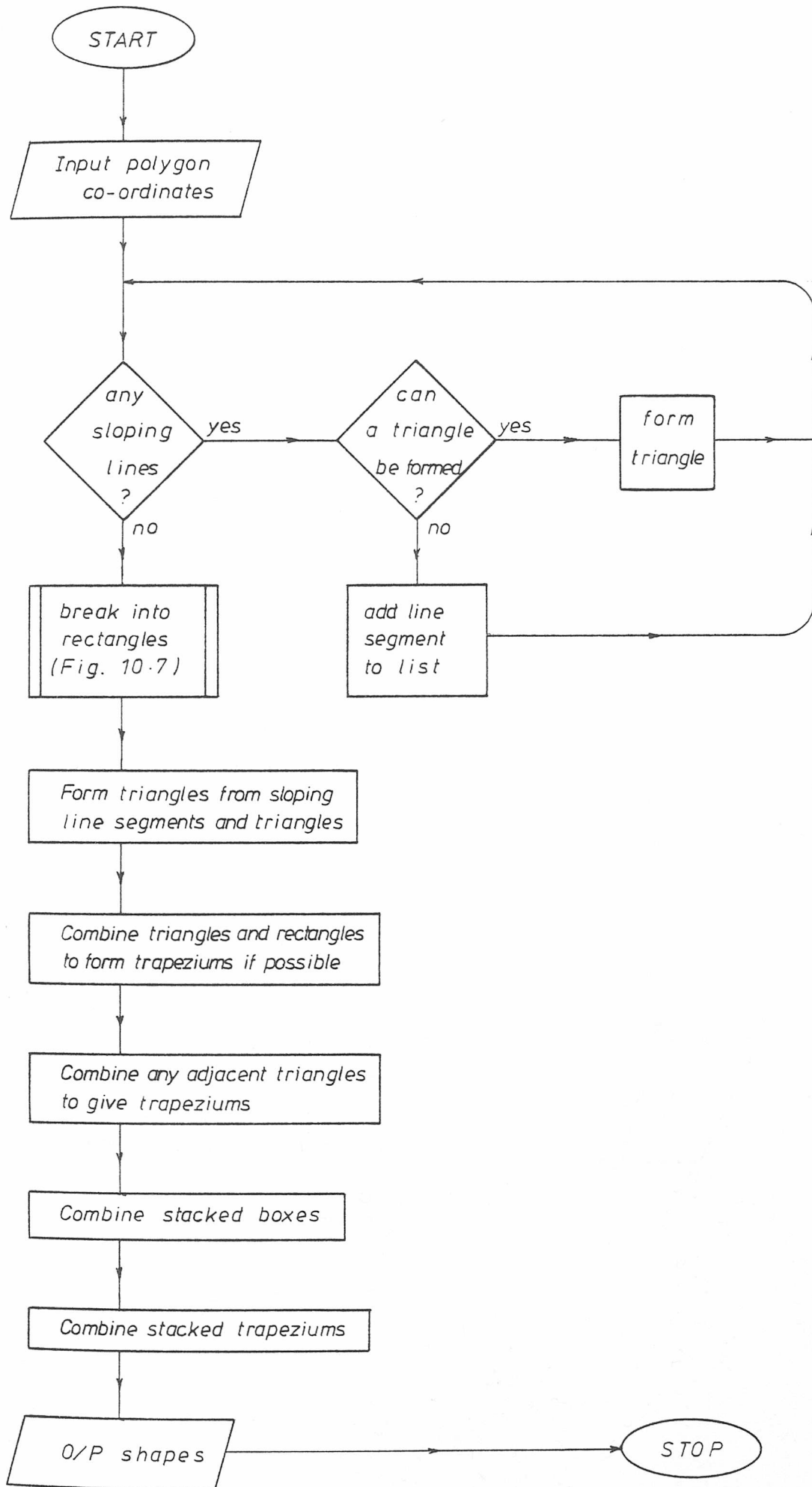


Figure 10.14 Flowchart of program "CUTTER"

DATA FILE "SHAPE.DAT" :-

0,2  
 0,11  
 5,9  
 2,4  
 5,4  
 7,9  
 7,11  
 9,11  
 9,9  
 6,4  
 6,0  
 3,0  
 -1

(co-ordinates of shape's  
 corner points defined in  
 a clockwise direction)

RUN CUTTER

CUTTER.FOR  
 A PROGRAM TO BREAK UP A PERIMETER DESCRIPTION  
 OF A STRAIGHT LINE POLYGON INTO ITS  
 COMPONENT RECTANGLES, TRIANGLES AND TRAPEZIUMS

INPUT NAME OF FILE CONTAINING DESCRIPTION  
 SHAPE

DO YOU REQUIRE A GRAPHICAL OUTPUT ? NO

TRIANGLE( 4) :- ( 0, 9) - ( 0, 11) - ( 5, 9)

TRIANGLE( 3) :- ( 2, 9) - ( 2, 4) - ( 5, 9)

TRIANGLE( 2) :- ( 3, 2) - ( 3, 0) - ( 0, 2)

TRAPEZIUM : ( 5- 6, 4) - ( 7- 9, 9)

RECTANGLE : -( 3, 0)( 6, 2)

RECTANGLE : -( 0, 2)( 6, 4)

RECTANGLE : -( 0, 4)( 2, 9)

RECTANGLE : -( 7, 9)( 9, 11)

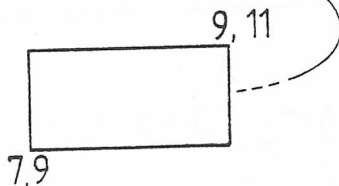
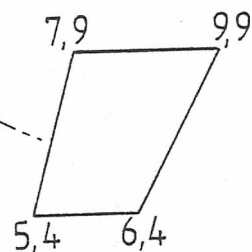
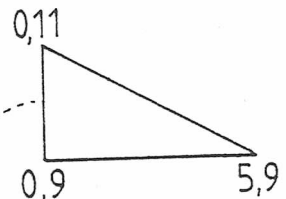
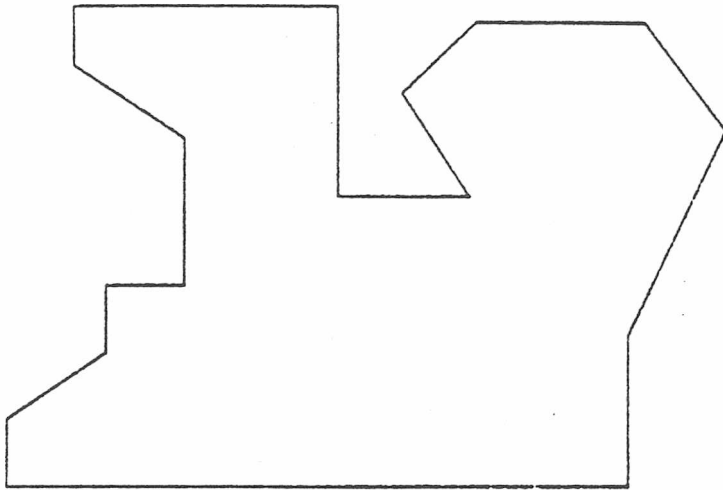
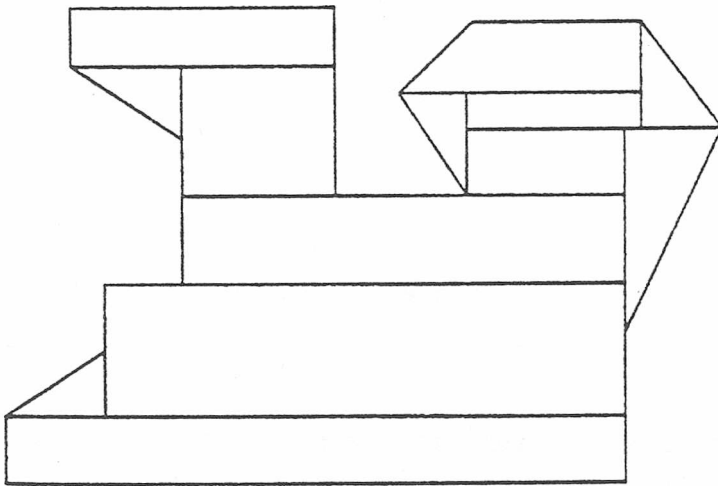


Figure 10.15 Numerical output from "CUTTER.FOR"



original shape



processed shape

Figure 10.16 Graphical output from "CUTTER"

Figure 10.17 shows a series of passes at different angles using a variety of apertures.

Group (a) was produced using a circular "Land" of light. This produced considerable "Flaring" (over-exposure) at the edges of the track. The Flaring becomes more pronounced if more than one pass is made over the same area of film. Clearly this shape is useless for anything but "Flashed" circular pads.

Group (b) is the result of using a circular "Track" aperture - this is similar to the Land, except that the middle is blanked out. A significantly lower concentration of light results from this aperture and Flaring is consequently reduced. Indeed, no Flaring could be detected under a 300X microscope. The only problem with this shape is that the aperture must be moved further than its own diameter - if it were moved less, a space would be left in the track due to the blanked-off centre.

A square Land was tested in group (c) and exhibited an unacceptable amount of flaring. An interesting phenomenon is that passes at an angle of 45 degrees produce very little flaring compared to those at 0 or 90 degrees. This can be explained by the fact that the sharp corners of the square effectively reduce the light concentration and hence the resulting Flaring. The corners will have minimum effect when the direction of motion is parallel with one of the axes. This aperture is again only useful when used in a "Flashed" mode.

Group (d) has been produced using the symbol "#1" to simulate a small split-shaped aperture. This shape has a small enough area to prevent flaring and has the advantage that it can reach into right angled corners.

The conclusion drawn from these tests is that only circular Track and slit-shaped apertures can be used in motion.

## 10.9 Filling In Rectangles

### 10.9.1 Method(a)- Using Circular Track only

Figure 10.18 shows the result of moving a small diameter track aperture round the inside of a rectangular perimeter. This causes an unavoidable rounding of the outside corners, the severity of which depends

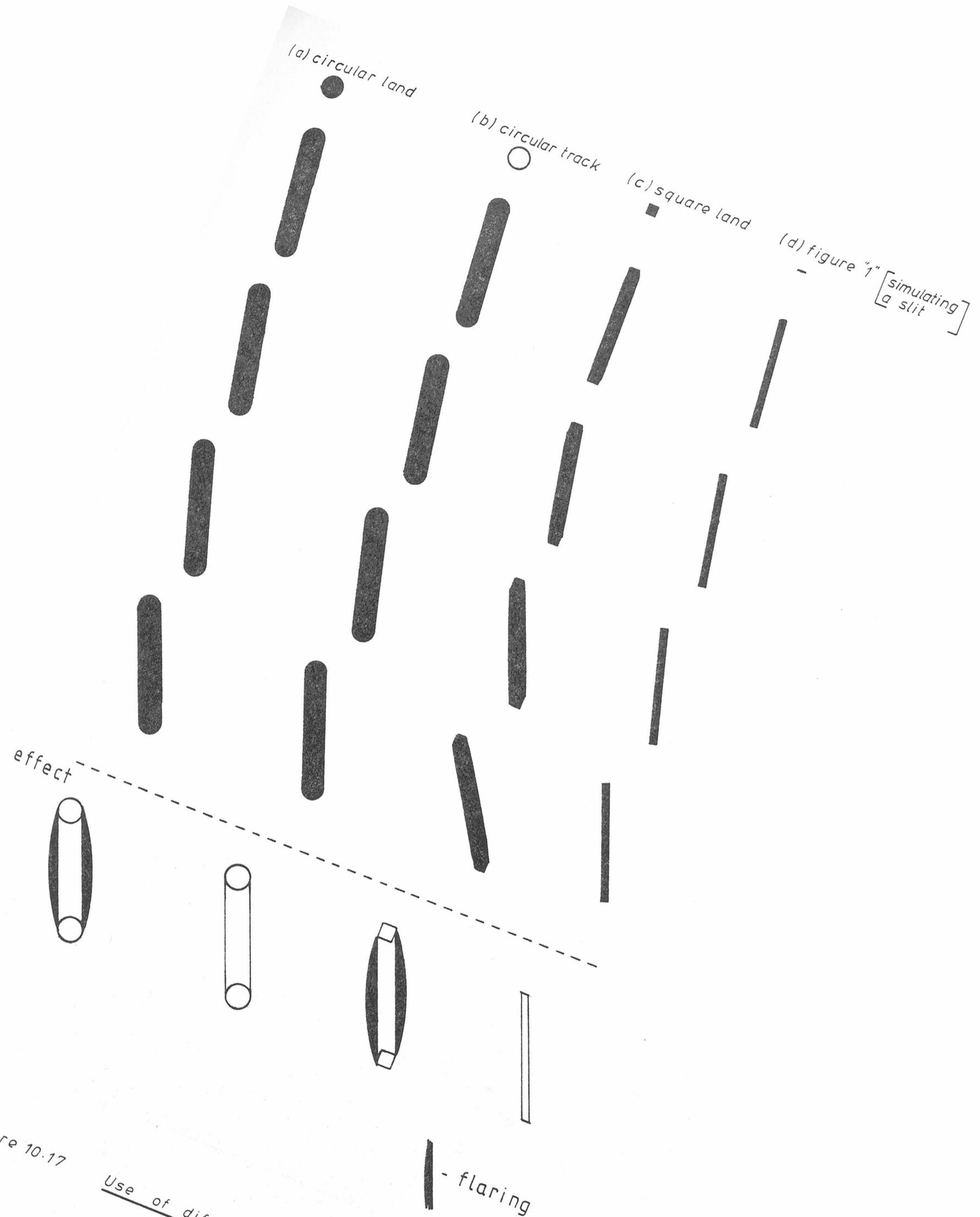


Figure 10.17

Use of different aperture shapes

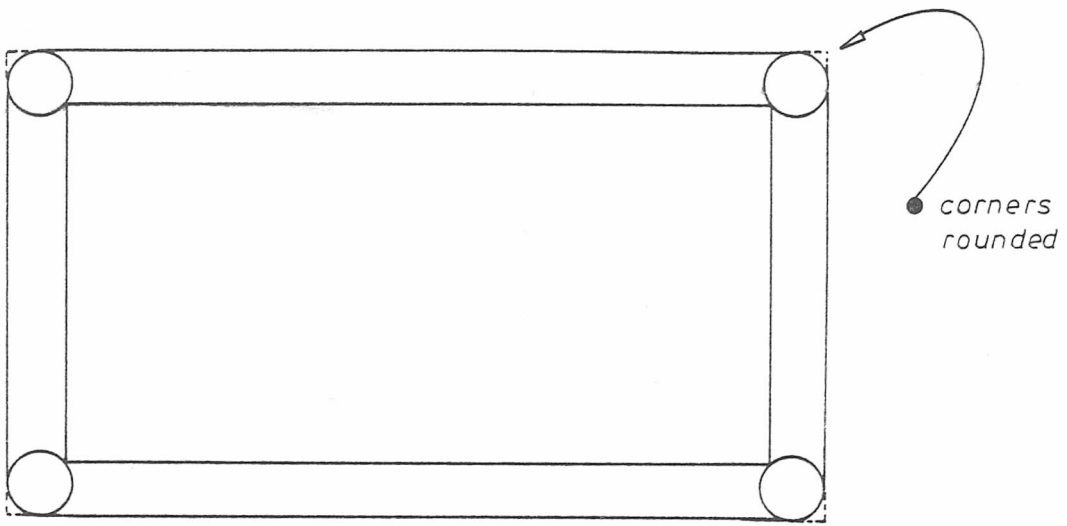


Figure 10-18 Circular Track used for Rectangular filling

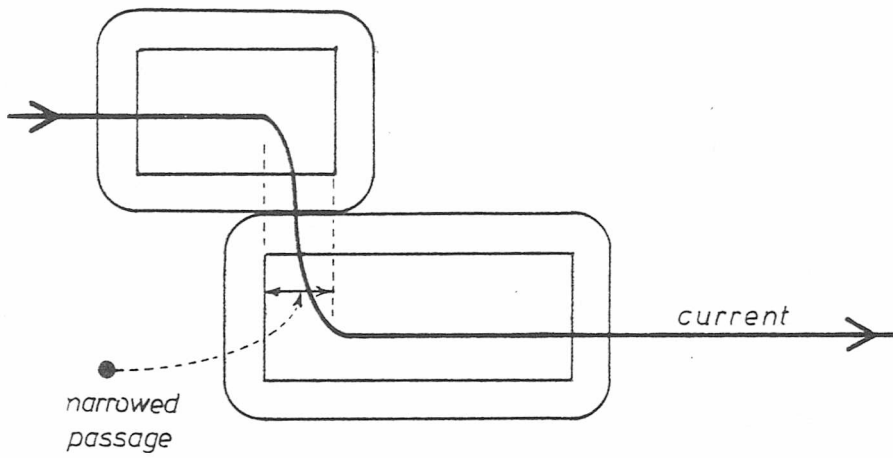


Figure 10-19 Stacked rectangles

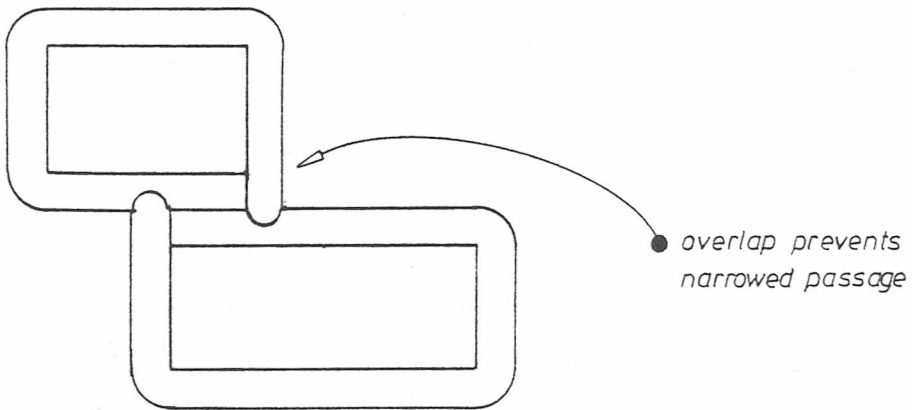


Figure 10-20 Stacked rectangles with filling overlap

upon the diameter selected. In general a little rounding is quite acceptable since there is little current flow in sharp vertices in any case. Trouble can result, however, when rectangles are stacked together - as in the case of a meandering resistor pattern. Figure 10.19 illustrates such a case where the rounding phenomenon has effectively increased the resistance by creating a bottle-neck in the current flow. This can be eliminated in practice by overlapping the filling of one rectangle into an adjacent one as in Figure 10.20. Such a system ensures that only true perimeter corners are left in a rounded state but it significantly increases the computer processing time required. This could be significant for very large plots.

Having plotted the rectangle perimeter, the next stage is to fill in the area left in the centre. One method is to use a series of straight adjacent passes as demonstrated in Figure 10.21. The passes are made parallel to the rectangle's largest dimension to minimise plotting time, and the same aperture is used as before. Each pass must overlap at each end by a value equal to the aperture radius in order to prevent gaps between pass ends (refer to diagram). The overlap, then, restricts the aperture diameter to an absolute maximum equal to twice that of the perimeter pass. Since a small aperture will initially be selected to minimise the rounding effects, this method tends to be rather slow.

A better approach is to use successively larger diameters to save plotting time. Figure 10.22 shows the result of such an approach, and it can be seen that the diameter increase is directly related to the diameter of the initial perimeter aperture. This example needed only one change, but larger rectangles may require a steady increase to the largest aperture size.

The geometry associated with reaching the inner right angle is given in Figure 10.23. From the right-angled triangle shown in (b) it is obvious from Pythagorean theorem that:

$$(R - L)^2 + (R - L)^2 = R^2$$

where R = new aperture radius

L = overlap required to reach into corner

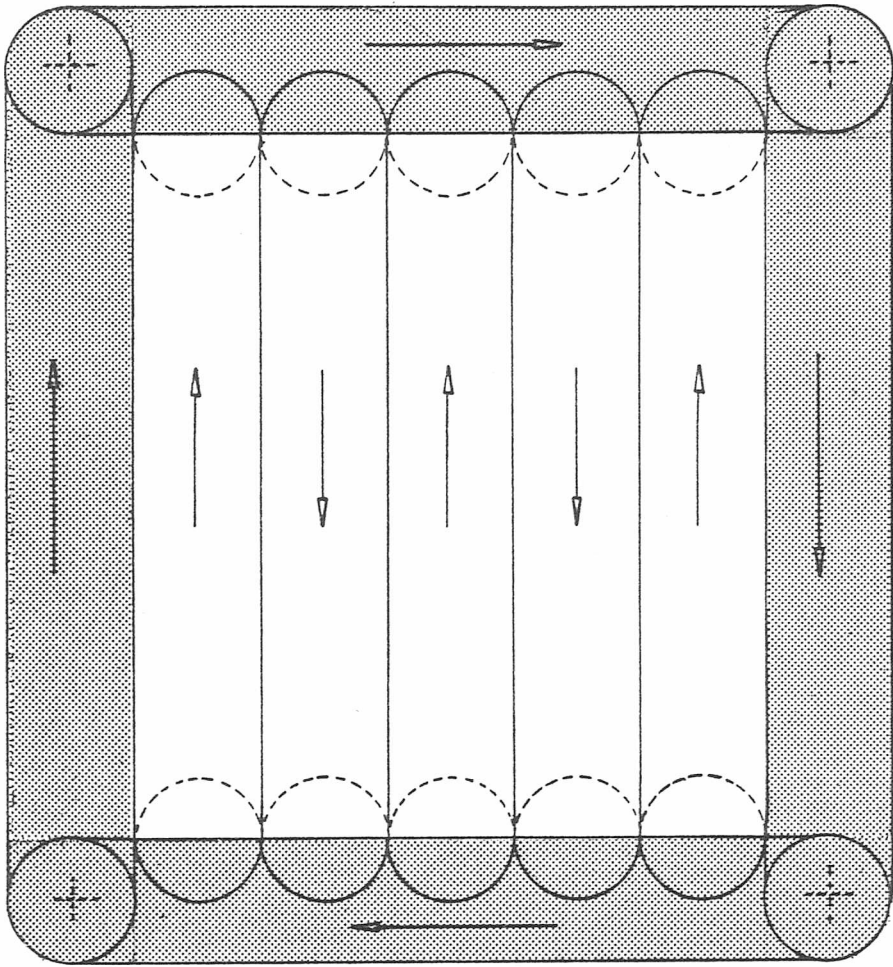
i.e.  $R^2 - 4RL + 2L^2 = 0$

Thus  $R = \frac{4L \pm \sqrt{16L^2 - 8L^2}}{2}$

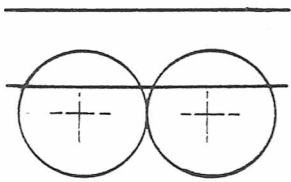
$\Rightarrow R = 2L + \sqrt{2}L$  (since  $R > L$ )

$\Rightarrow L = R/3.414214$



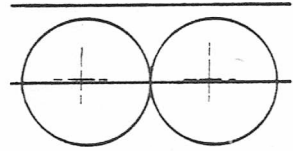


*Initial Pass perimeter*



*overlap < radius*

—> *gap results*



*overlap > = radius*

—> *no gap*

*Figure 10-21 Filling using adjacent passes*

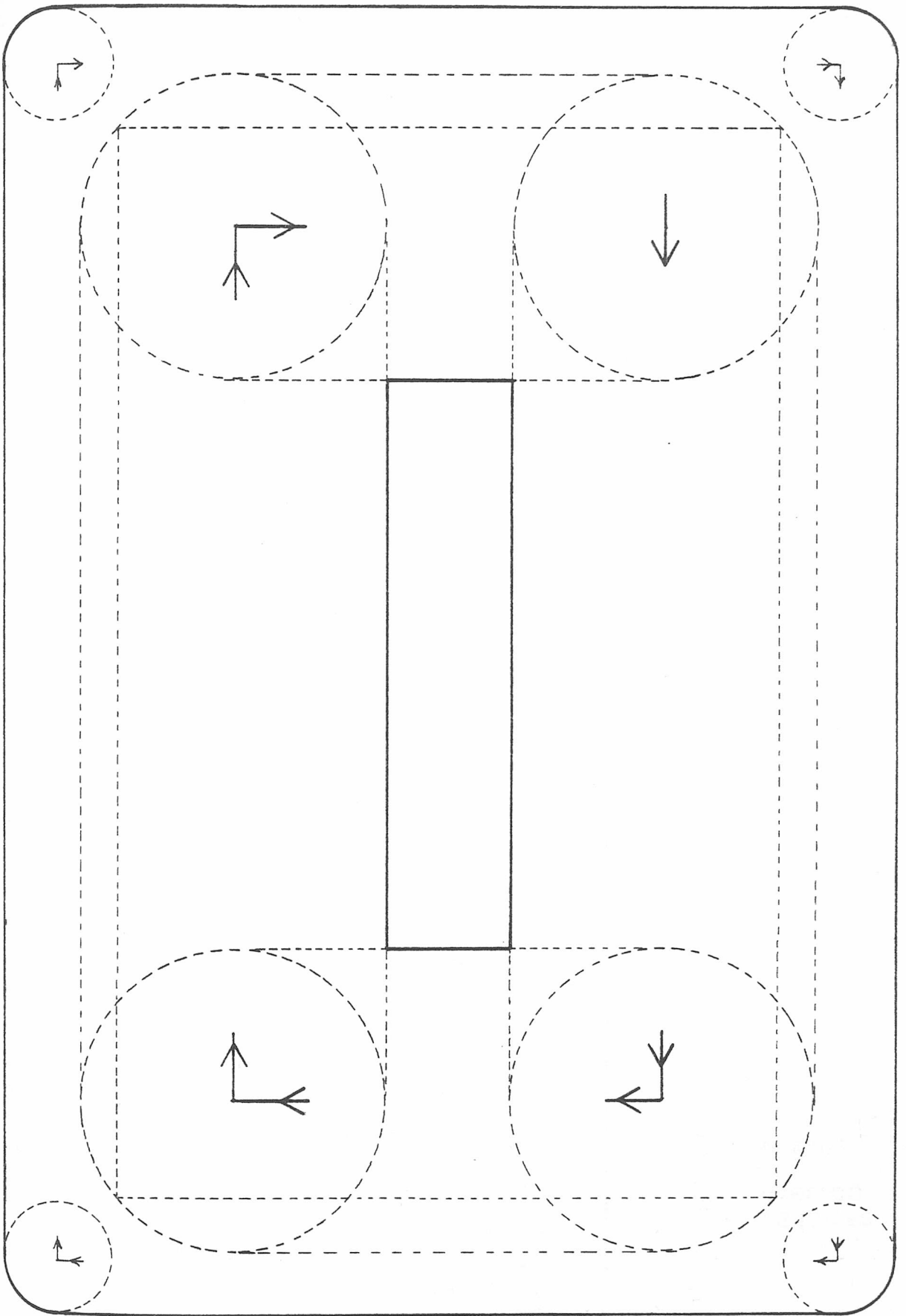
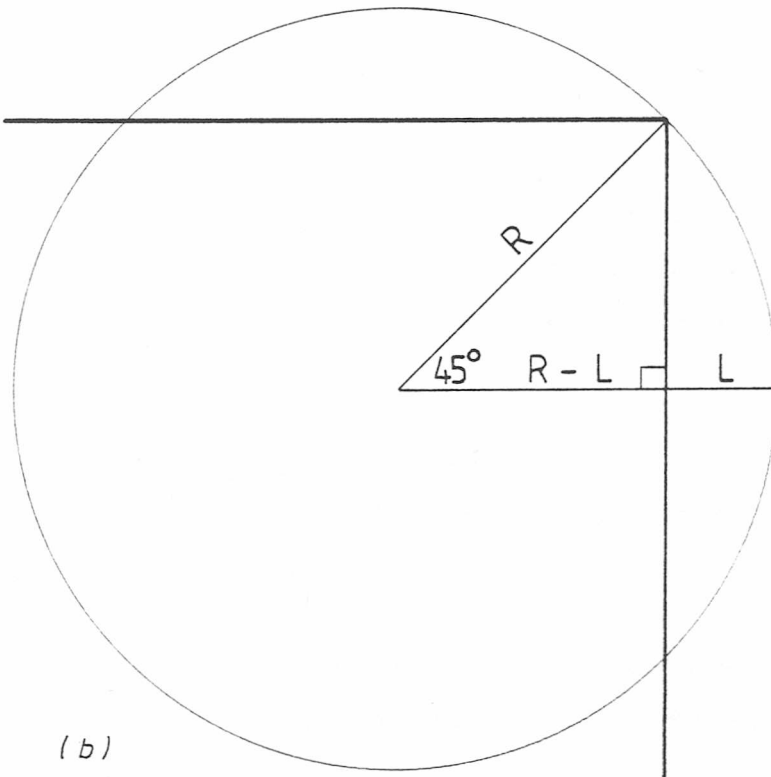
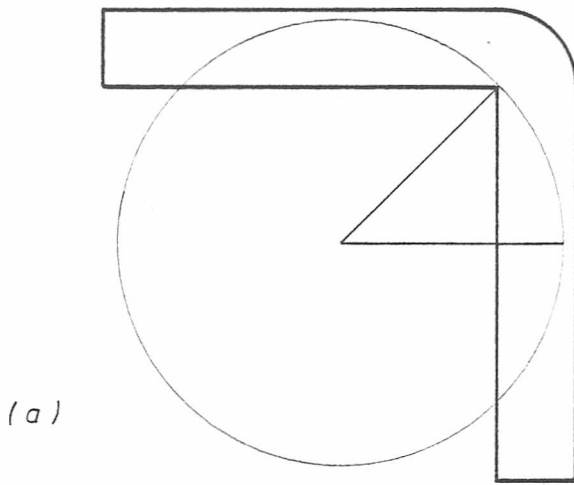


Figure 10 · 22

Filling in using progressively larger diameter track apertures



R = radius  
L = overlap required

Figure 10.23 Geometry of corner

The amount of overlap required for any particular aperture diameter can now be calculated. A table showing the available aperture diameters with their corresponding overlaps is given in Figure 10.24.

Assuming an initial selection of 0.005", which is the smallest available aperture size, it is possible to immediately jump to 0.015" by allowing a 0.0022" overlap. This avoids the use of intermediate sizes 0.010" and 0.0125" which would merely prolonge the filling process. Using the same argument the next logical steps are 0.060" and finally 0.200" which is the largest symbol diameter in the plotter cartridge. It is apparent that any rectangle can be processed using a maximum of four different aperture sizes. This method is generally faster than that proposed in Figure 10.21, and the difference is more pronounced in larger rectangles. Tests were carried out using both techniques and equally good definition results were obtained - plotting time apart. No flaring could be detected, but the rounding effect was clearly visible under a microscope.

#### 10.9.2 Method (b) - Using Track And Square Apertures

An improvement to the methods described above is to use square Land apertures to form the rectangle corners. Figure 10.25 shows how the land can be flashed at each corner to give a good representation of the required rectangle. The square size must be greater than, or equal to, the outside track radius to ensure that no gaps are left. Since the smallest square available is 0.05", rectangles with dimensions smaller than this cannot be treated in this way.

A large diameter track can now be used for the perimeter pass which means that the simple adjacent pass method is better suited than the increasing diameters method.

The technique of overlapping passes in adjacent rectangles is still used in preference to the flashed square method where applicable.

#### 10.9.3 Method (c) - Using Slit Shaped Apertures

Methods (a) and (b) have had problems in reaching the right-angled corners of the rectangle due to the circular aperture shape involved. The use of horizontal and vertical slits solves this problem and greatly reduces the processing time needed in each case.

	aperture diameter (ins)	required overlap (ins)
*	0.005	0.0007
	0.010	0.0015
	0.0125	0.0018
*	0.015	0.0022
	0.030	0.0044
	0.050	0.0073
*	0.060	0.0088
	0.075	0.0110
	0.100	0.0146
	0.125	0.0183
	0.150	0.0220
*	0.200	0.0293

Figure 10-24 Table of Aperture overlaps showing a possible set of increment jumps

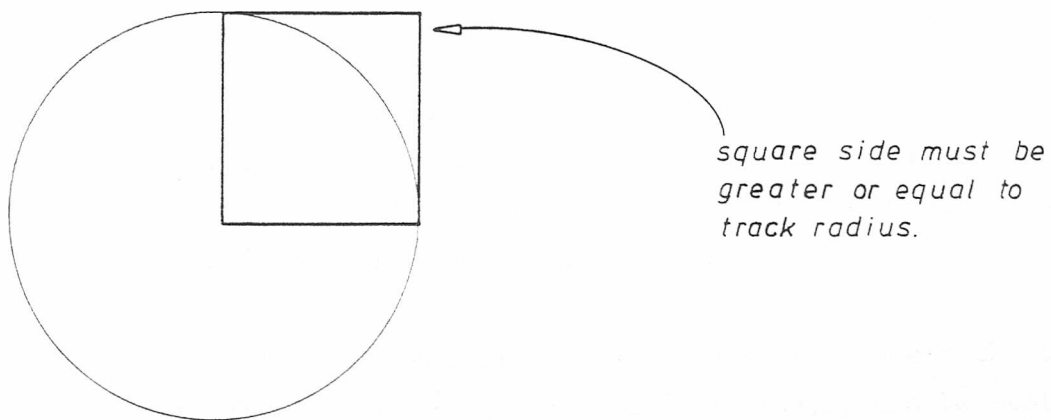
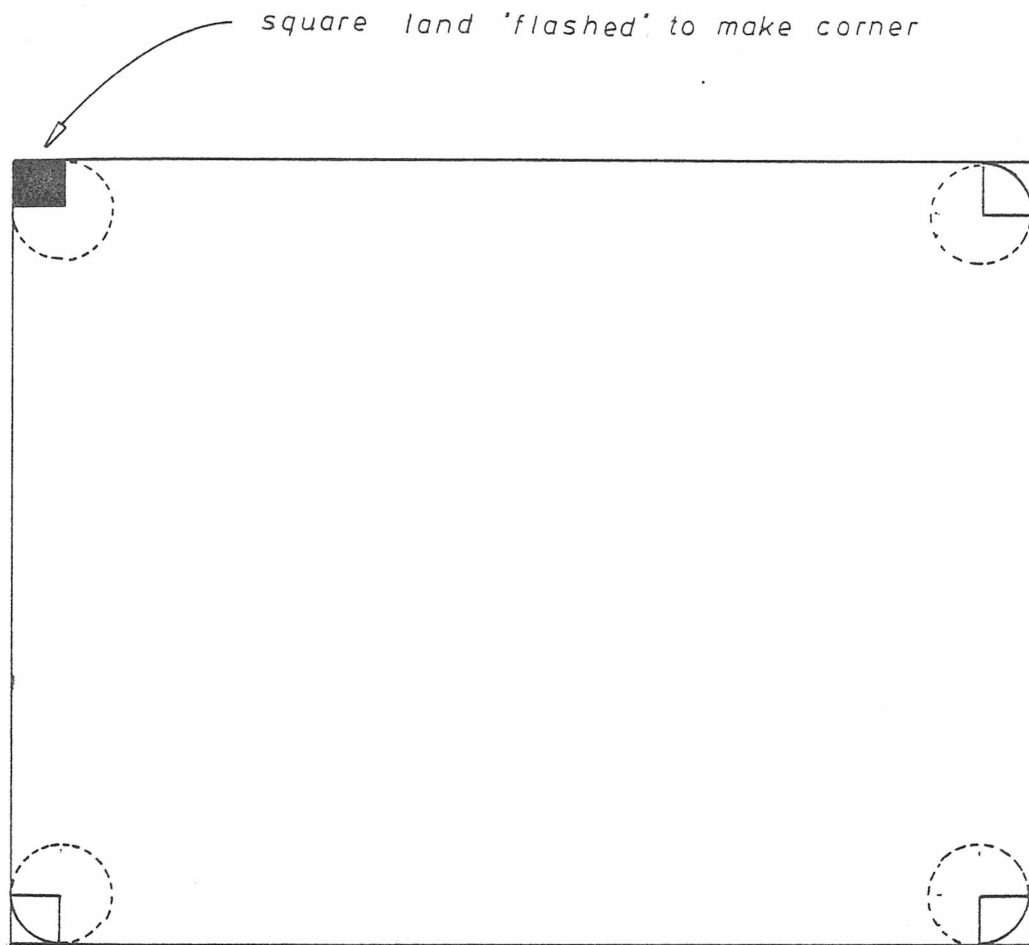


Figure 10.25 Filling using circular track and square land

In Figure 10.26, a rectangle has been processed in two distinct ways - first using a horizontal slit and then a vertical slit. The largest horizontal slit to fit the rectangle width is used in (a) to produce a number of adjacent passes. If the rectangle is not an integer number of passes wide, the last pass is made to overlap the penultimate one. A similar technique is adopted for the vertical slit, case shown in (b).

It is not immediately clear which method is best since a small vertical slit may be very much faster than a large horizontal slit, or vice-versa. The algorithm which has been developed simulates both ways for each rectangle using the largest possible slit in each case. It then chooses the faster of the two. An additional consideration is the time taken to change the aperture. It may be more expedient to use the same, or a closer aperture than to select a distantly placed aperture which will fill the rectangle in a shorter time.

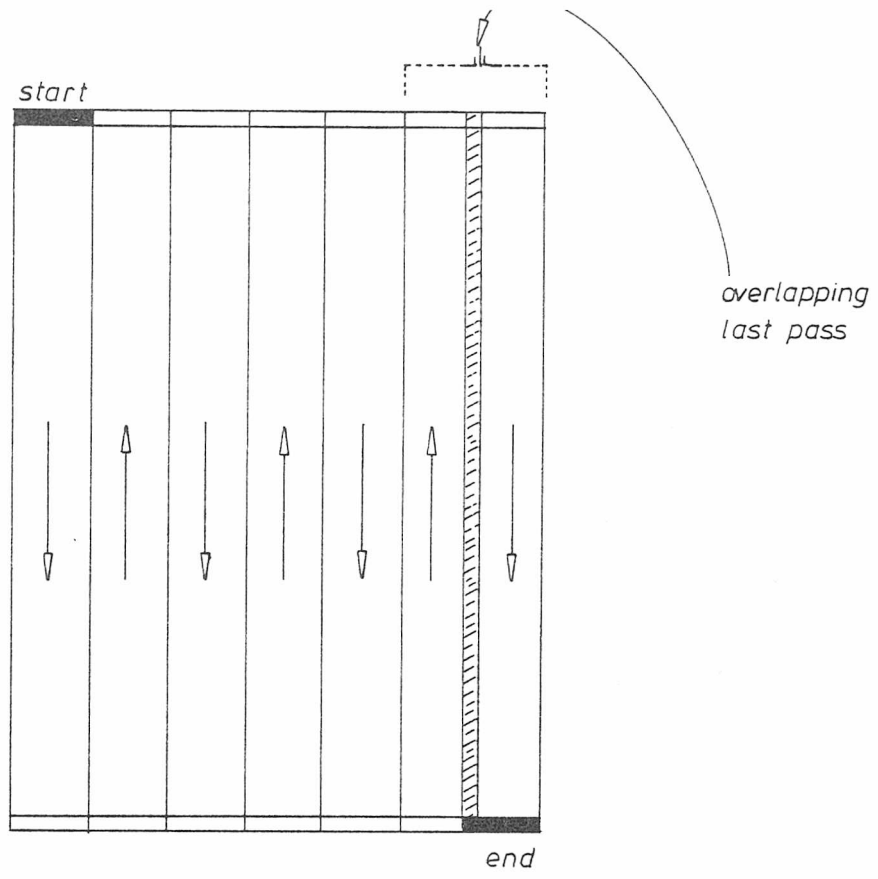
In the example given there is more chance of mis-alignment error in (b) than in (a). This is due to the increased number of positional moves required. Case (a) is called the "Best Contour" representation for this reason - this may or may not be the fastest of the two depending upon the slit sizes that are available.

#### 10.10 Filling In Triangles

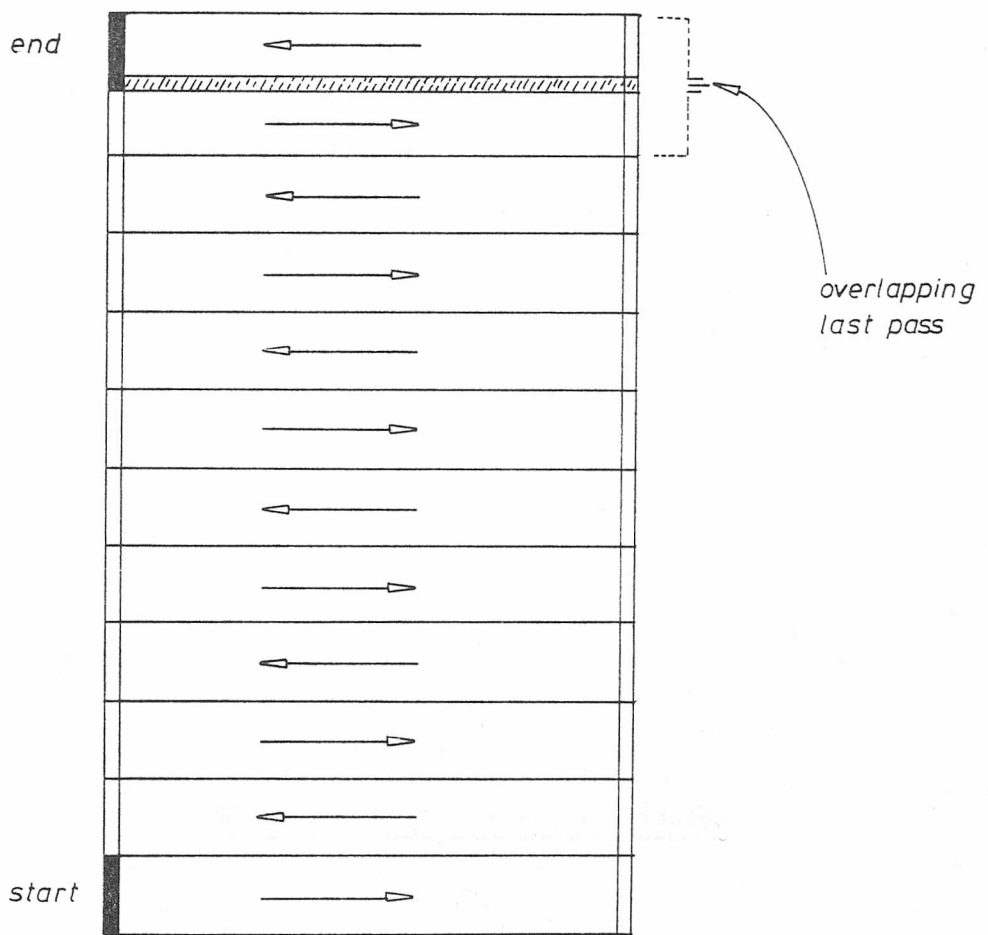
The right-angled triangles generated in the polygon splitting algorithm have one side parallel to the X axis and one side parallel to the Y axis. No satisfactory filling technique could be implemented using circular apertures due to the shape's inherent sharp vertices. A slit based approach met with considerably more success.

Both Horizontal and Vertical slits are used to process the triangle, although the overlapping technique mentioned previously can be employed to eliminate one orientation in a limited number of cases.

The first step is to plot overlapping passes along the hypotenuse using both orientations as demonstrated in Figure 10.27 (a). This gives a very good representation of the angled line, but necessitates a certain amount of overshoot to reach the sharp vertices. The slit width has been



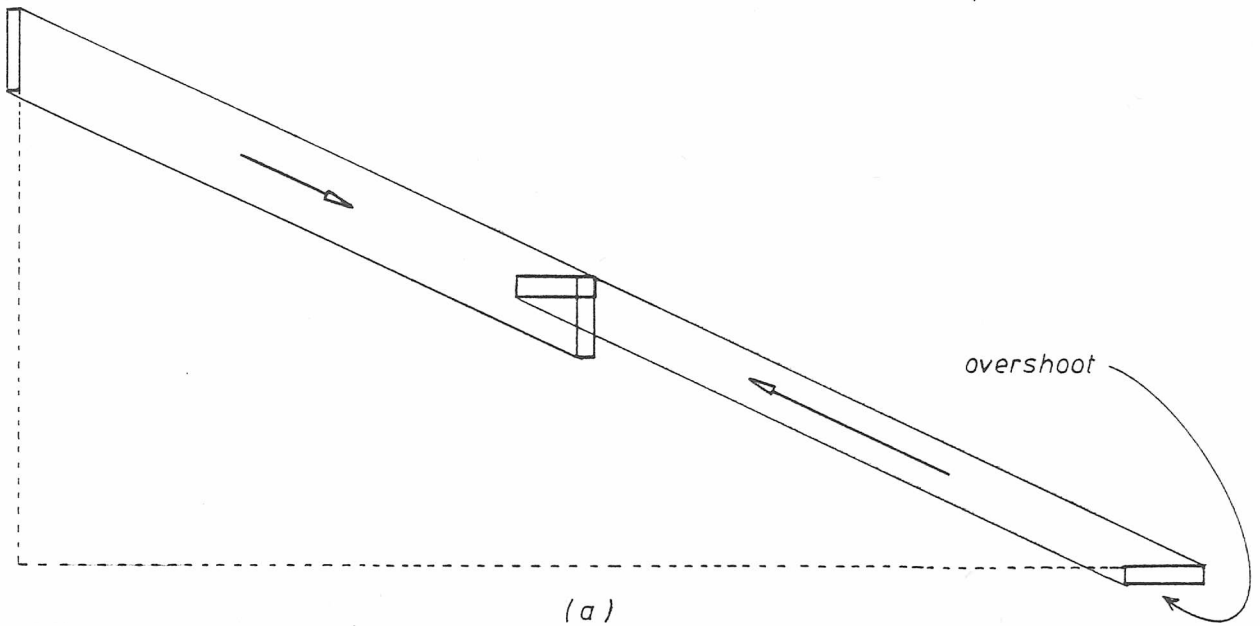
(a) using horizontal slit



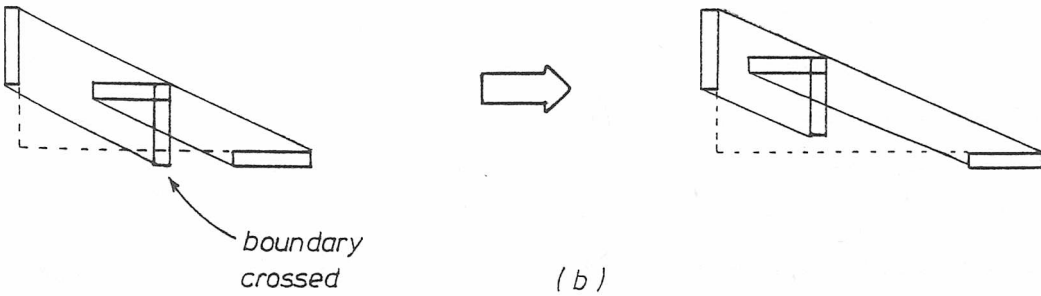
(b) using vertical slit

Figure 10.26 Filling using Slit Apertures





(a)



(b)

Figure 10.27 Filling triangles - Angled passes

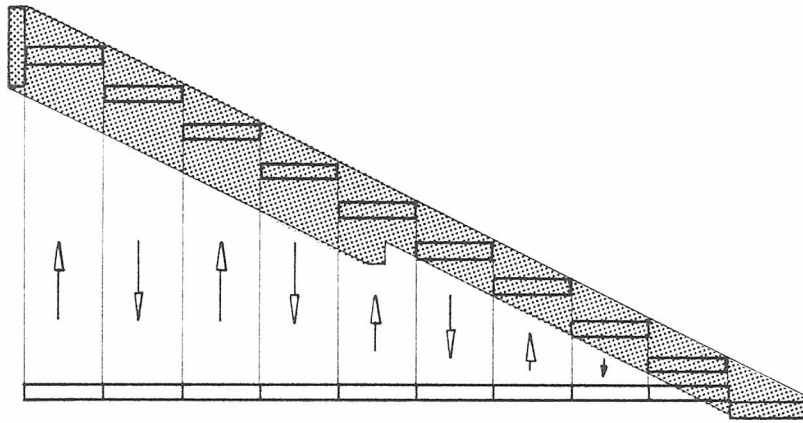
emphasised to illustrate the filling method but the overshoot is negligible in practice. There is also the probability that there will be an adjacent rectangle present and the overshoot would not matter at all in that case.

The point where the two orientations overlap is usually set to the middle of the hypotenuse but this may have to be varied for acute angles to prevent the slit intersecting the other triangle boundaries. This problem is illustrated in Figure 10.27 (b). It may be impossible to find a suitable position - in which case the slit size is deemed too large, and the process fails. The triangle filling algorithm initially selects the largest slit size and changes to a smaller one each time it fails. This ensures that the plotting time is kept to a minimum. The smallest allowable triangle is closely related to the smallest slit size for this reason.

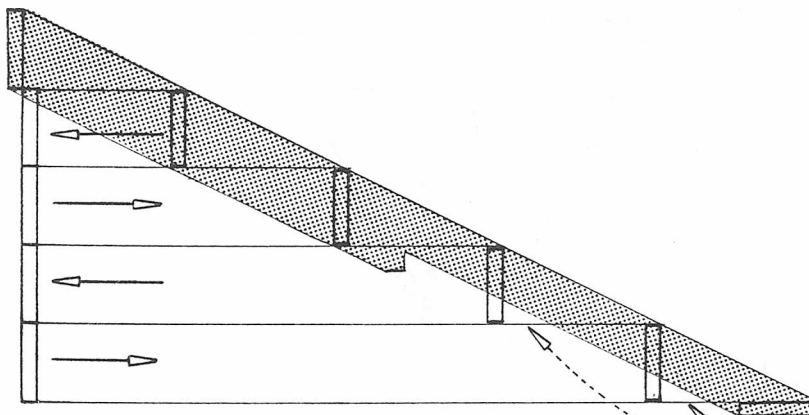
The next stage is to fill in the main bulk of the triangle as efficiently as possible. Adjacent passes are made perpendicular to the longest paraxial side employing the same slit size. In this method, shown in Figure 10.28 (a), the number of passes is equal to the ratio of side length to slit size. One would assume, then, that by using the small side in preference to the larger, a smaller number of passes would be needed. This saves plotting time by eliminating un-necessary dark positioning moves. The problem with this approach is that the slit size is always larger than the width of the hypotenuse pass (measured in the same orientation). This causes spaces to be left in the triangle which cannot be reached without overshooting the hypotenuse. A smaller slit must be used to eliminate this effect - the size calculated from the angles involved. The choice between methods (a) and (b) is made on the grounds of speed in each particular case.

An amount of overlap is allowed between adjacent passes to prevent the slit vertices crossing the hypotenuse due to slight positional errors in the photoplotter. This overlap can be set up by the user as required.

Figure 10.29 illustrates the algorithm using triangles in each of the four possible orientations. The program deals with first quadrant cases only, but uses reflection in the X and Y axes to process the other orientations.



(a) Filling along longest side



(b) Filling along shortest side

○ spaces left

Figure 10.28 Triangle filling - processing main bulk

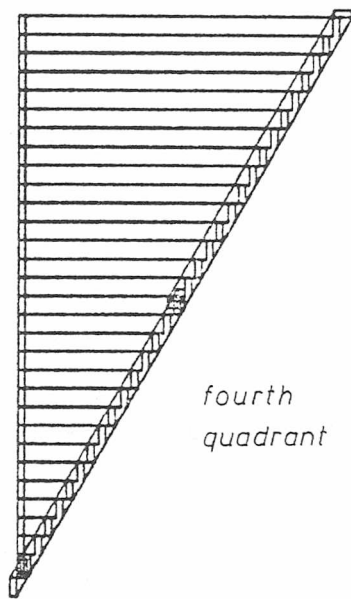
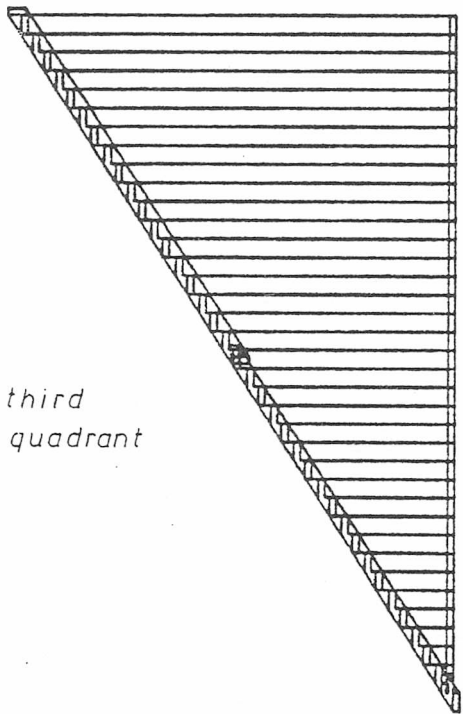
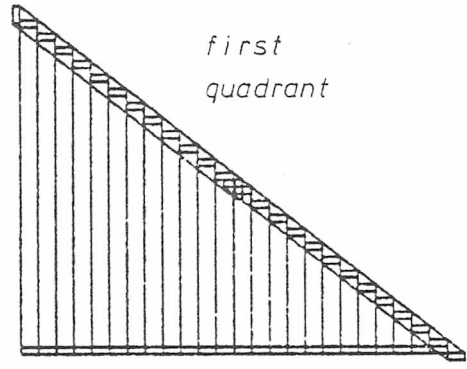
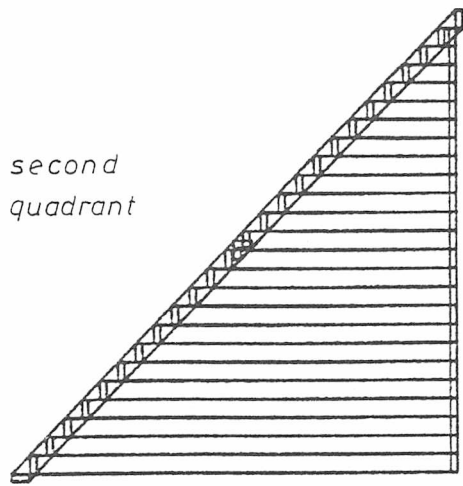


Figure 10.29 Examples of triangle filling

## 10.11 Filling In Trapeziums

The trapeziums produced in the splitting algorithm are defined as having two edges parallel to the X axis. Like the triangle case, it is difficult to process the trapezium using circular apertures due to the acute angles involved. The shape lends itself, however, to a slit filling method.

Figure 10.30 (a) demonstrates the technique, which uses a Horizontal slit. The idea is to have a step size equal to the slit width for the long horizontal edge and a suitably smaller step size for the short horizontal edge. This produces a certain amount of overlap which is directly proportional to the ratio between the two edge sizes. There will be a critical ratio where flaring becomes a problem along the shorter edge - this ratio is difficult to predict but should become evident in practical tests.

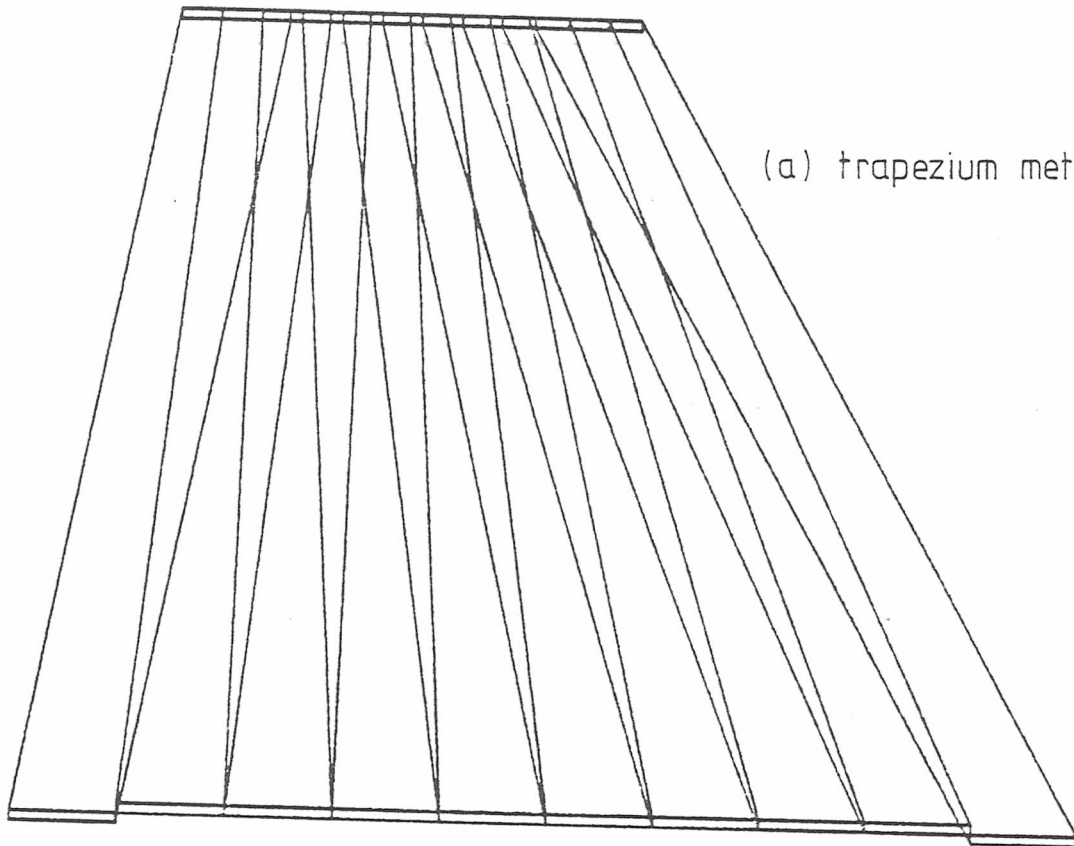
The trapezium can be split into a number of triangles and a rectangle. Figure 10.30 (b) shows how these shapes can be processed individually to fill in the trapezium. This is a much slower technique but produces less overlap. Should the ratio between horizontal edges be unacceptably large, then this method can be employed to guard against flaring. The mechanics of this, in practice, is that the splitting algorithm will not form trapeziums with a ratio larger than a figure specified by the user.

Figure 10.31 shows the range of trapezium types which can result. Note that a slight degree of overshoot is required in each case to reach the acute angled vertices.

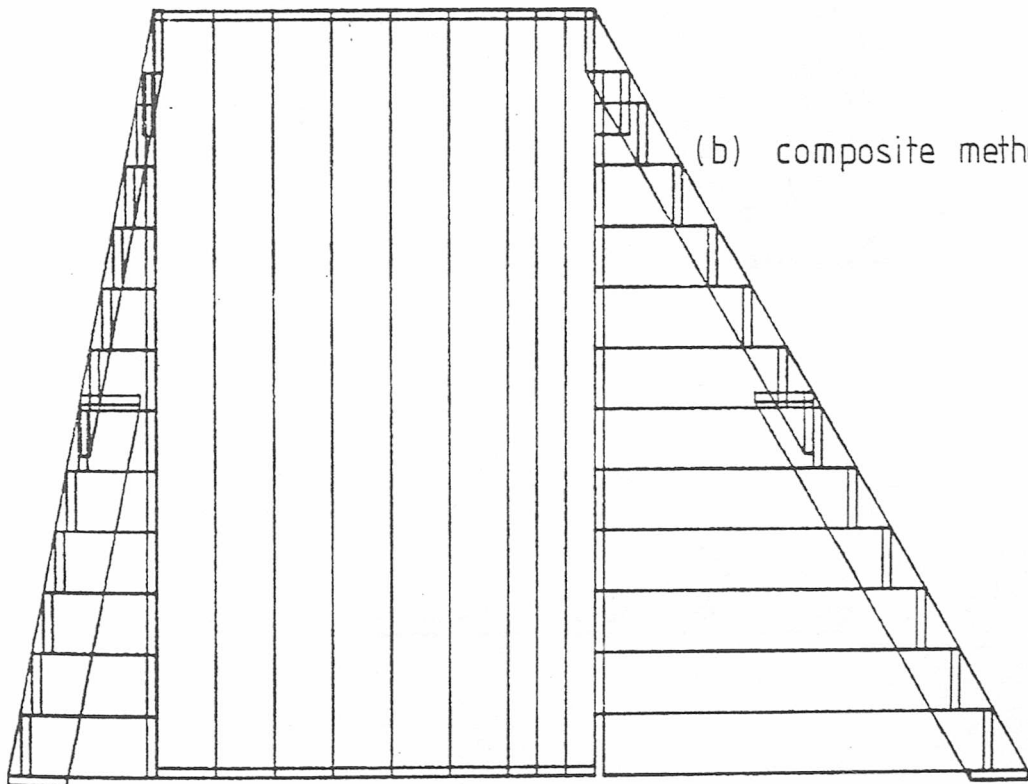
## 10.12 Conductor Track Filling

Rather than treat each conductor track as a separate polygon, it is more efficient to process them in a slightly different way. Consider Figure 10.32 which shows two methods using slits and circular track respectively.

The slit method shown in (a) must continually alternate between horizontal and vertical orientations. This involves a considerable amount of aperture selection time and also requires re-positioning moves



(a) trapezium method



(b) composite method

Figure 10.30 Filling in a trapezium

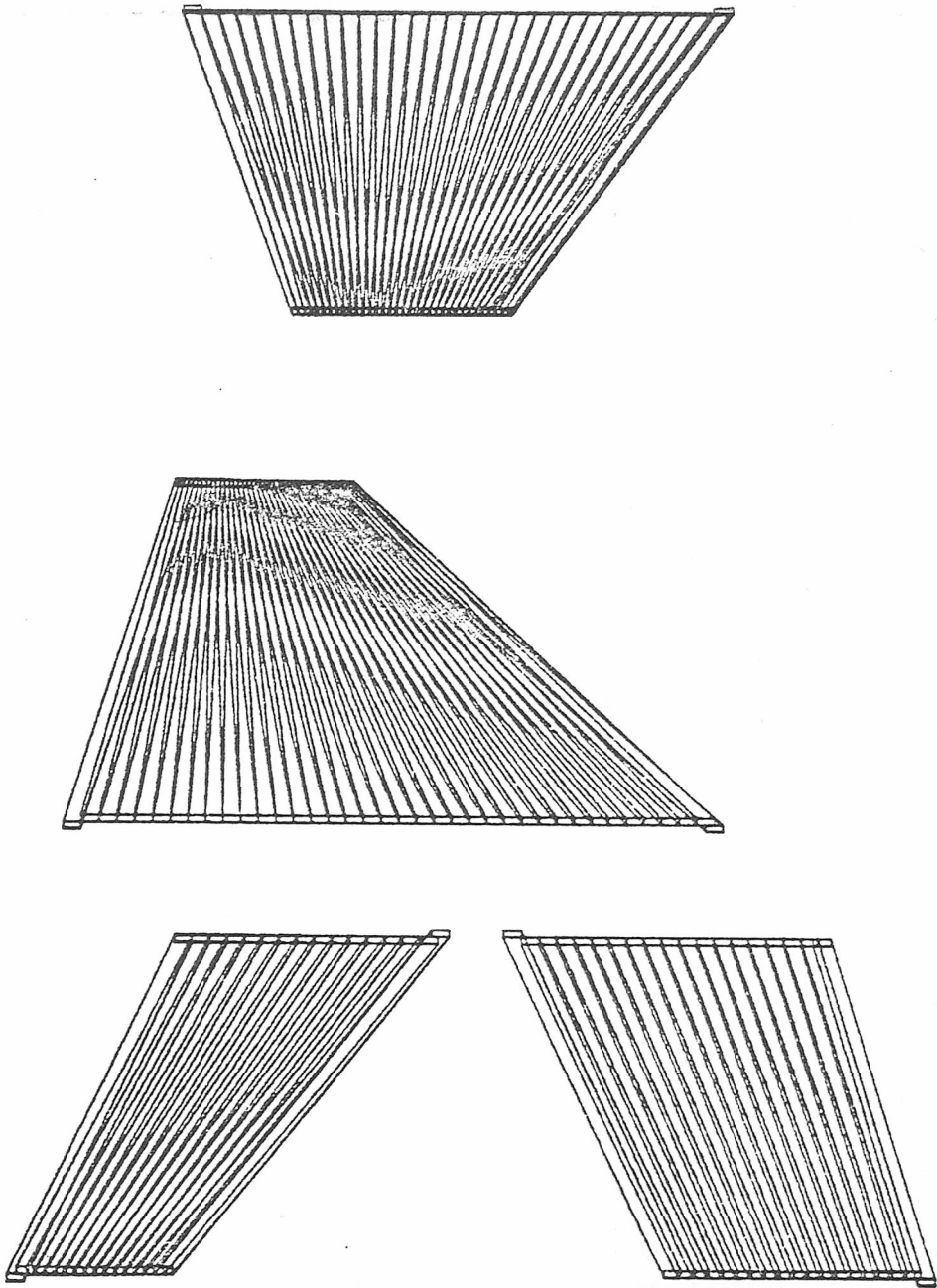
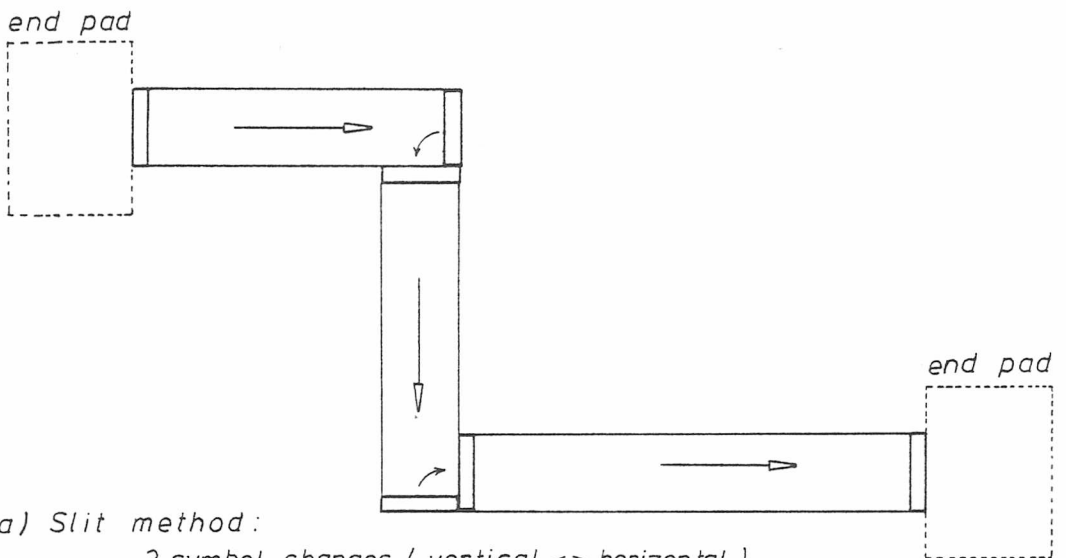
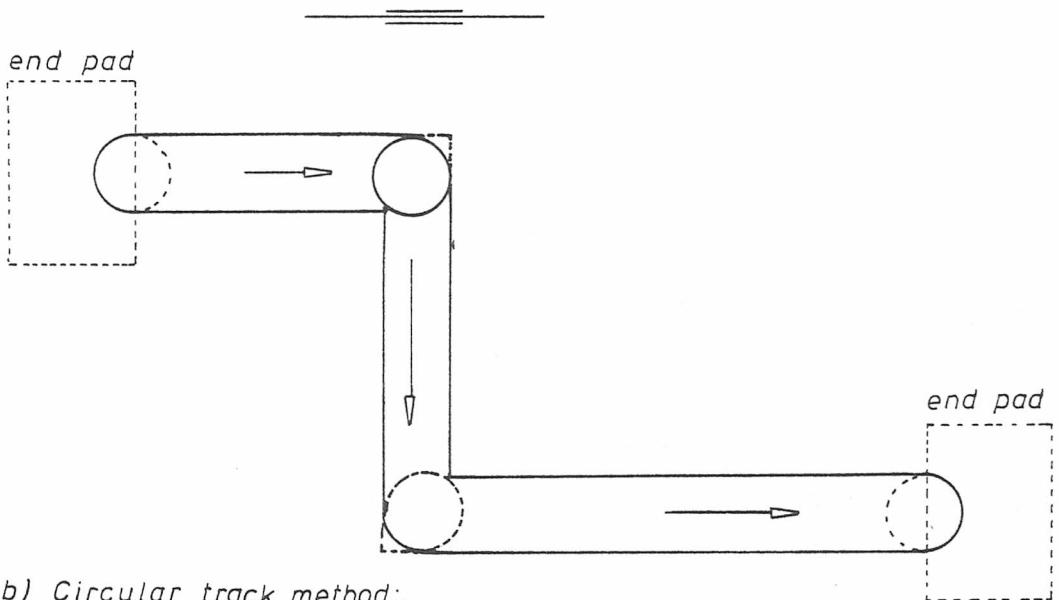


Figure 10-31 Examples of trapezium filling



(a) Slit method:

- 3 symbol changes ( vertical  $\leftrightarrow$  horizontal )
- + 3 dark moves ( to start point and repositing slit )
- + 3 light passes
- 
- = 9 operations



(b) Circular track method:

- 1 symbol change ( to select track aperture )
- + 1 dark move ( to start point )
- + 3 light passes
- 
- = 5 operations

Figure 10-32 Track filling methods



between each pass. A much faster technique is to use a circular track aperture to eliminate re-positioning and selection times. In the example given, only 5 operations are needed using the circular track method compared to the 9 required in the slit equivalent. The plotting time difference between the two methods depends upon the relative position of the apertures in the cartridge and the length of the conductor track under consideration.

The outside corners are rounded slightly in (b) but this is actually preferable in conductor tracks. Another effect of using a circular aperture is that it must overshoot the track ends by an extent equal to its own radius. Since all conductors are constrained to end in terminal pads, this is of little importance.

The conductor width in this example was chosen to coincide with one of the aperture dimensions. This saves plotting time since only one pass need be used for each conductor track. It is perfectly possible, however, to process any track width using two or more overlapping passes. Since the choice of conductor width in the layout stage is not too critical, it is obviously good sense to favour the nearest aperture size.

### 10.13 Mask Production Programs

Two programs have been developed using the splitting and filling algorithms which have just been described. The first is compatible with the "GAELIC" suite of programs (Ref 6) and is called "PHOT". It operates directly from a GAELIC ring data structure and produces data files containing plotting language commands - one data file for each mask required. The second program, "SHADE", is used with the Layout and Interconnection suite itself. This program creates a similar set of plotting files from a "LAYOUT" data structure.

Figure 10.33 shows how these plotting files can be used to produce visual checking diagrams (on the V.D.U. /Pen Plotter) or to produce tapes to drive the photoplotter itself.

The program "VIDEO" shows the filling methods by drawing each aperture pass separately. Figure 10.34 shows a typical line drawing of this type. This is used as a final check before the expensive process of photoplotting is initiated and can be displayed on the terminal

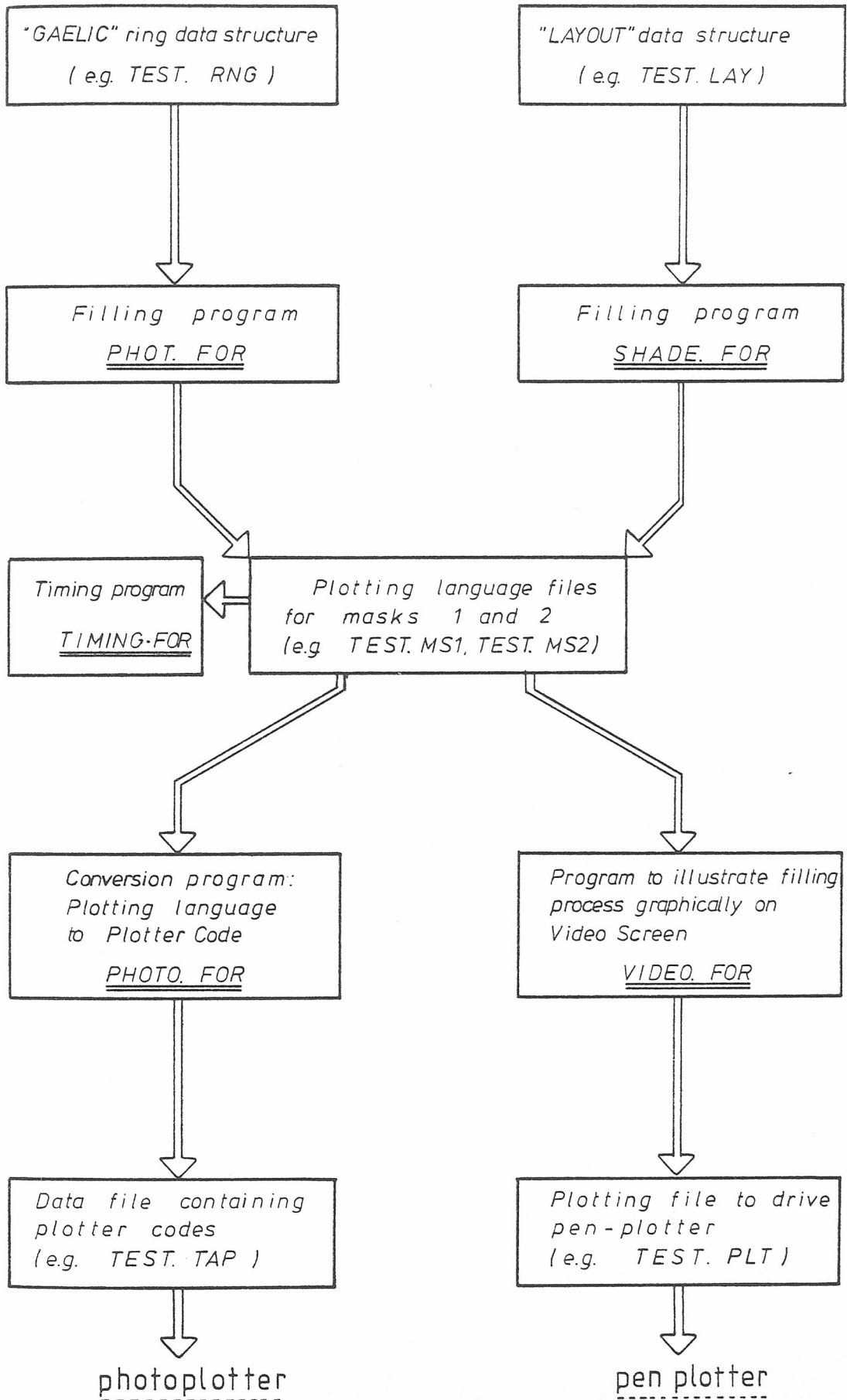


Figure 10.33 The Mask Production Suite of Programs

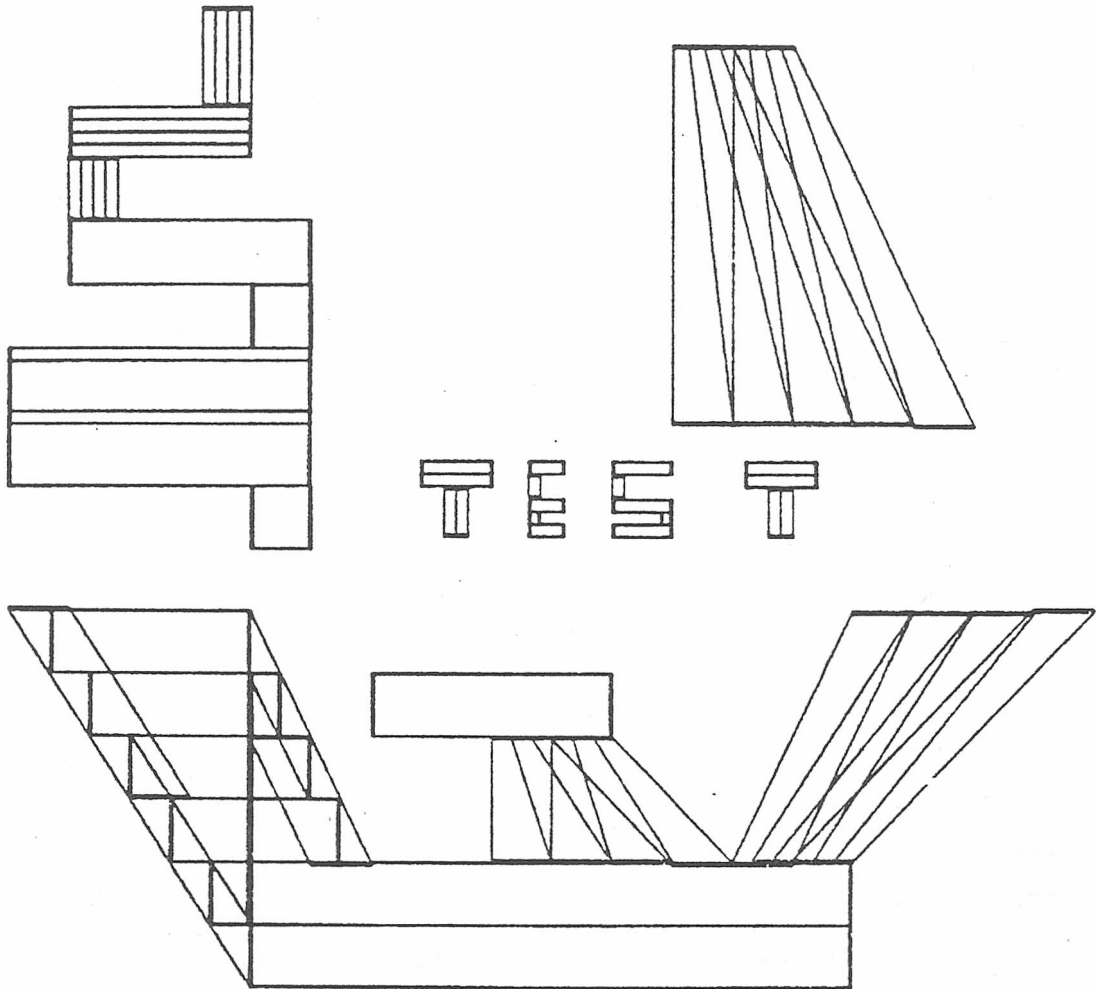


Figure 10.34 Photoplotter testing drawing produced by "UIDEO"

screen or drawn by a pen-plotter. It is also useful to compare this drawing with the actual photoplotter output in order to determine which passes, if any, resulted in flaring.

#### 10.14 Evaluation Of Various Slit Ranges And Filling Techniques

The slit apertures discussed in previous sections are not, as yet, available. Ferranti Ltd expressed interest, however, in fabricating a set of slit apertures and provided a typical thin film circuit description in GAELIC language for examination. The circuit is given in Figure 10.35 together with its associated packaging. The two masks needed to fabricate this circuit (shown separately in Figure 10.36) are totally paraxial. This allows a comparison to be made between the various rectangle filling algorithms with regard to plotting time and quality of reproduction.

A program has been written to calculate the total plotting time required by any data file containing a set of plotting commands (as described in Section 10.4). The program, called "TIMING", gives the aperture selection time and plotting time separately for evaluation purposes.

The slit width should be as small as possible to minimise the overshoot effects discussed in previous sections. Practical tests with circular land apertures, however, have indicated that apertures with dimensions less than 0.010" do not give a satisfactory sheet exposure.

The range of slit sizes to be adopted is a compromise between expense of fabrication and the possible saving in plotting time that can be achieved. To examine the effect of different slit ranges, the test circuit was processed in a number of ways. The table given in Figure 10.37 shows a selection of results, (calculated by "TIMING") using five different ranges. In addition, each range was tested in four distinct ways.

Column (a) shows the results of the fastest slit filling method, where the computer determines whether horizontal or vertical filling is best for any particular rectangle. The number given after the word "SIZE" refers to the size of the plotting file produced. This is

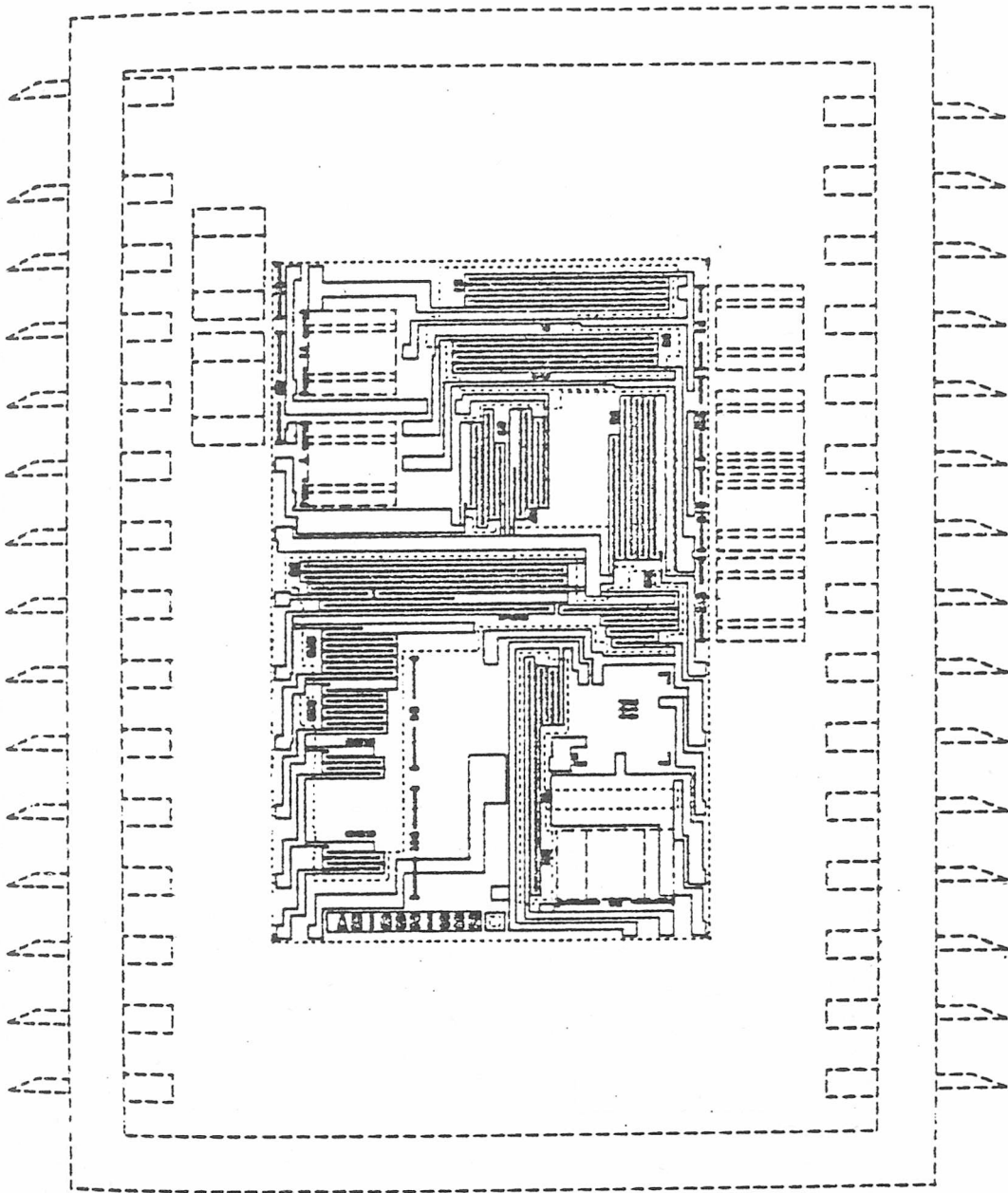
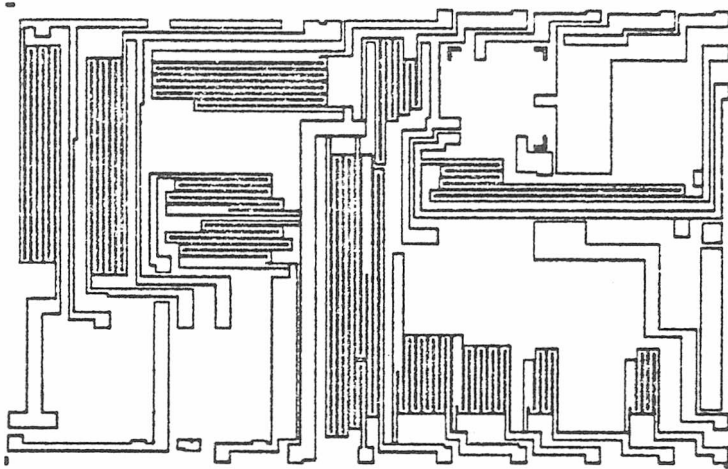
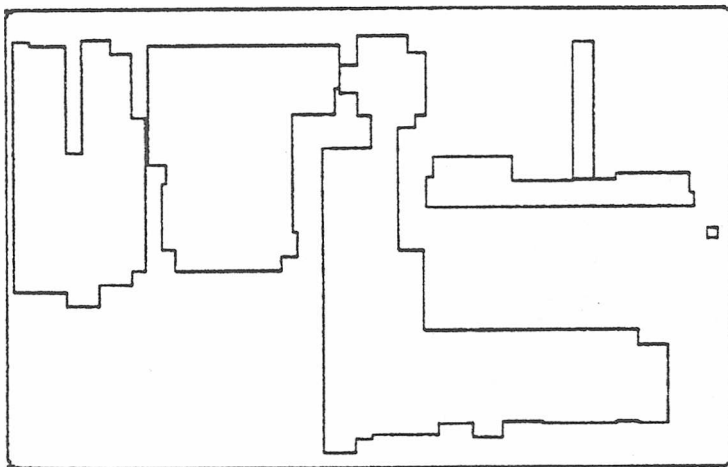


Figure 10.35 Typical industrial thin film circuit



Mask 1



Mask 2

Figure 10.36 Masks required to fabricate circuit shown in Figure 10.35

	(a) FASTEST METHOD	(b) ONLY HORIZONTAL SLITS	(c) ONLY VERTICAL SLITS	(d) BEST CONTOUR
5 SLITS 200, 100, 50, 30, 20 x 10 [thous']	SIZE = 16 pages SYMBOL = 21s <u>TIME = 13m20.56s</u> 10 apertures	SIZE = 26 pages SYMBOL = 6.78s <u>TIME = 14m30.76s</u> 5 apertures	SIZE = 27 pages SYMBOL = 11.25s <u>TIME = 14m55.65s</u> 5 apertures	SIZE = 14 pages SYMBOL = 23.94s <u>TIME = 13m52.38s</u> 10 apertures
4 SLITS 200, 100, 50, 20 x 10	SIZE = 16 pages SYMBOL = 19.84s <u>TIME = 13m19.46s</u> 8 apertures	SIZE = 26 pages SYMBOL = 6.72s <u>TIME = 14m30.76s</u> 4 apertures	SIZE = 27 pages SYMBOL = 9.72s <u>TIME = 15m5.68s</u> 4 apertures	SIZE = 14 pages SYMBOL = 22.47s <u>TIME = 13m51.3s</u> 8 apertures
3 SLITS 200, 100, 20 x 10	SIZE = 32 pages SYMBOL = 22.31s <u>TIME = 15m7.03s</u> 6 apertures	SIZE = 28 pages SYMBOL = 6.53s <u>TIME = 18m3.38s</u> 3 apertures	SIZE = 29 pages SYMBOL = 7.25s <u>TIME = 17m18.59s</u> 3 apertures	SIZE = 17 pages SYMBOL = 20.19s <u>TIME = 19m31.97s</u> 6 apertures
2 SLITS 200, 20 x 10	SIZE = 39 pages SYMBOL = 14.22s <u>TIME = 16m11.25s</u> 4 apertures	SIZE = 38 pages SYMBOL = 3.06s <u>TIME = 20m18s</u> 2 apertures	SIZE = 34 pages SYMBOL = 3.53s <u>TIME = 20m58s</u> 2 apertures	SIZE = 24 pages SYMBOL = 18.66s <u>TIME = 24m28.4s</u> 4 apertures
1 SLIT 20 x 10	SIZE = 31 pages SYMBOL = 18s <u>TIME = 29m41.6s</u> 2 apertures	SIZE = 177 pages SYMBOL = 0.03s <u>TIME = 42m41s</u> 1 aperture	SIZE = 182 pages SYMBOL = 0.19s <u>TIME = 43m9.81s</u> 1 aperture	SIZE = 31 pages SYMBOL = 18s <u>TIME = 29m41.6s</u> 2 apertures

Figure 10-37 Timing results for test circuit

directly proportional to the length of paper tape produced to drive the photoplotter, so large plotting files will result in long tapes. The timing program does not take into account the time needed to physically read the tape, but the length of tape must be considered as an important factor in gauging the efficiency of a plotting method. The aperture selection time is given separately from the overall processing time.

All slit sizes must lie within the boundaries 0.200" - 0.020". The latter figure is governed by the tolerances which can be achieved in circuit fabrication, while the former is the largest aperture size that will fit the machine. If we assume that the 0.200" aperture is available then the largest rectangle which could not be processed would have a dimension just less than this figure. The next smallest slit size should be able to fill this rectangle using no more than two overlapping passes. It should also be as small as possible, such that it can process a maximum number of rectangle sizes. The obvious choice, then is 0.100". Using the same reasoning we can choose the other aperture sizes to be 0.050" and 0.025". After due consideration it was decided that a slit of 0.020" would be used instead of the 0.025" value to allow the plotter to process the smallest rectangle dimensions that can arise in practice (Ferranti standards). This means that rectangles in the range 0.040" to 0.050" (exclusive) will need three passes instead of the usual two overlapping passes. Adding a slit size of 0.030" to the range did not reduce the plotting time significantly and actually increased the aperture selection time due to the increased number of slits.

The same slit ranges were tested in column (b), but only the horizontal orientation was allowed. A similar set of results were produced in column (c) using vertical slits. It can be seen that the plotting times are within 6% of the fastest method when large slit ranges are available, but are much longer for small ranges. It must be remembered that only half the number of apertures are needed for this method, and this will save the cost of slit manufacture. Much longer tapes are produced, however, and this could be significant. There is also the problem that triangles could not be processed using a single orientation.

Column (d) is the result of the "BEST CONTOUR" method discussed earlier. Each rectangle is filled using passes parallel to the longest



sides thus minimising the number of possible mis-alignment errors which could be produced. The plotting times needed in this technique are again comparable to the fastest method for larger slit ranges. Small ranges on the other hand produce significantly slower plots. The 2 slit case, for example, takes 50% more time than the fastest method.

An anomaly is found in the single slit case where both the fastest and BEST CONTOUR methods take exactly the same time, but this is easily explained. Since only one slit size is available, it is always faster to fill a rectangle using passes along the longest sides. This is, in effect, the technique used for the "BEST CONTOUR" result so it is not surprising that identical plots are produced.

Since the aim of the "BEST CONTOUR" approach is to minimise unnecessary positioning moves, it follows that fewer plotting commands are generated. This makes for shorter data files and correspondingly shorter tapes than in any of the other methods.

The circuit was also processed using circular track and square land apertures, but the fastest plotting time that could be achieved was 24 minutes and 15 seconds. When we compare this with the results using slits we see that a range of only two sizes is still faster. It is also important to notice that 19 apertures are being used in the first case compared to only 2 slit apertures.

The results indicate conclusively that a range of slit apertures would be the fastest and therefore cheapest way of producing masks on the photoplotter. Ferranti Ltd agreed with this and suggested that an experimental set should be made. A range of four slit sizes was chosen as the optimal case from the test results.

The photoplotter itself was used to produce transparencies of the slit shapes which will be photographically reduced to form the aperture masters. Circular track and square lands were used to plot the slits which were chosen to be 0.200", 0.100", 0.050" and 0.020" X 0.010". At the time of printing, the slits are still in the fabrication stage.

Conclusions And Possible Improvements

This chapter indicates the degree of success achieved in each of the separate design stages described in Chapters 3 to 10. It also outlines the areas in which the programs could be extended and improved.

11.1 Component Design

11.1.1 Conclusions

The resistor design programs outlined in Chapter 3 have been fully tested in all but fabrication. This is not a serious omission since it is possible to check the resistance of the final resistor patterns to a reasonably high accuracy without actually making them. The basic formulae used to derive the resistor equations, of course, have been verified by practical experiment. (Ref 2).

Run times have proved to be very variable - they depend upon the number of iterations needed in each case which is rather unpredictable. Invariably they have been very much less than one minute of c.p.u. time on a DEC system 2040.

11.1.2 Possible Improvements

At present the suite is limited to resistor design. The automatic design of active components such as capacitors and inductors is an obvious area of interest. It would also be useful if the user could define component geometries himself - perhaps using a data description file similar to the output from the resistor programs. This would allow him to design in-slice components by traditional methods but still have the advantage of computer aided placement and routing. An additional benefit of this is the ability to use crossover sites as capacitors - two conductor tracks being separated by a dielectric. This is often preferable to the jump wire technique and generates an in-slice component at the same time.

Another area of interest concerns the trimming of resistors after fabrication. At present the laser beam or equivalent mechanism is positioned by hand (or co-ordinate data prepared by hand) prior to each

trimming pass. It would be much more efficient to produce this information directly from the data structure. It would also be possible to provide information regarding the maximum trimming distance available in each block. This would eliminate human errors such as overtrimming, trimming from the wrong side of the block etc., and would reduce the overall fabrication time.

When the circuit is being designed manually, the engineer is often left with irregular shapes in which to place his in-slice resistors. The automatic design programs can only cope with rectangular boundaries so they may consequently waste useful space. Radley (Ref 18) has attempted to ease this problem with his program to design resistors inside any straight-edged polygon. There is no provision for the addition of trimming blocks, however, and each corner square has been approximated to a constant supplied by the user. It may be useful to implement a version of his algorithm incorporating formulae to allocate trim blocks and calculate corner allowances individually.

## 11.2 Describing And Storing The Circuit

### 11.2.1 Conclusions

The system of rings and pointers which has been used to represent the circuit in the data structure was chosen to speed the interrogation time at the sacrifice of memory storage. It has proved versatile in all aspects of processing and was changed very little from its original form during the development phases. It is based on the structure developed by Rose (Ref 19) in his original Planarity Algorithm program.

### 11.2.2 Possible Improvements

It would be very useful to have separate files containing the Master Component Libraries and Topological data respectively. Since the former can be common to a number of layouts while the latter is particular to one, it would be much better to segregate the two. This would slow the Planarity program, however, since it would cause the library file to be searched each time a new component type is encountered. Another problem is that rare component types would have to be added to the library even though they might never be used again. A compromise would be to

allow the user to define his own Master descriptions in the normal way but cause the program to refer to the library for any missing descriptions.

It has already been mentioned in section 11.1.2 that it would be useful if the user could define his own in-slice components. This would require a special data description format not unlike the GAELIC manual input language (Ref 6) which could be stored in a separate file.

The proposed system, then, would be driven from four separate data files:

- (a) A Master Description library file which is only altered when a new component type comes into general useage.
- (b) A Topology file containing the circuit topology together with additional Master Descriptions and a list of nodes representing Edge Pads.
- (c) A Resistor Description file generated by the automatic design programs described in Chapter 3.
- (d) An In-slice Component data file - compiled by the user himself.

### 11.3 Planarity

#### 11.3.1 Conclusions

The Planarity Algorithm described in Chapter 5 and incorporated into the layout suite was written by Rose (Ref 19) to design single-sided printed circuit boards. Alterations have included adjustments to the description/storage formats and the addition of automatic crossover insertion algorithms.

The basic algorithm is very fast indeed but the crossover insertion stage can be rather slow as it involves a number of tree searches and design loops. At present the algorithms do not necessarily find the solution with the minimum number of crossovers and all the improvements listed in the next section are concerned with this problem.

## 11.3.2 Possible Improvements

### 11.3.2.1 Node Splitting

The crossover insertion algorithms have been implemented using the principle that it is possible to reach one Region from another if they have a common Branch in the topological graph. In general it is also possible to cross between Regions which have a common Node.

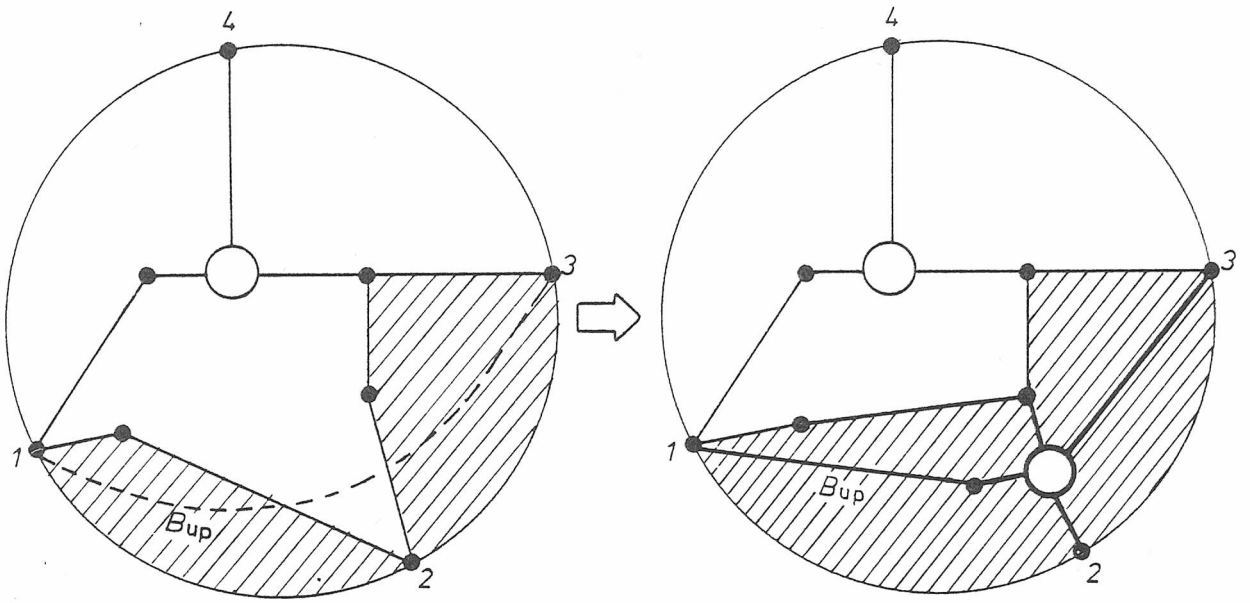
Figure 11.1 (a) shows a situation where component "Bup" has to be inserted between nodes 1 and 3. Two crossovers would normally be inserted here since there exists no common Branch to both Regions. A technique known as "Node-Splitting" can be applied to reduce this to one, however. In this method the two Regions need only have a common Node - either an Edge Pad or a circuit Node as in (b). Using this approach in conjunction with the existing algorithm will guarantee a solution with the absolute minimum of crossover sites.

### 11.3.2.2 Re-sequencing the Edge Pad List

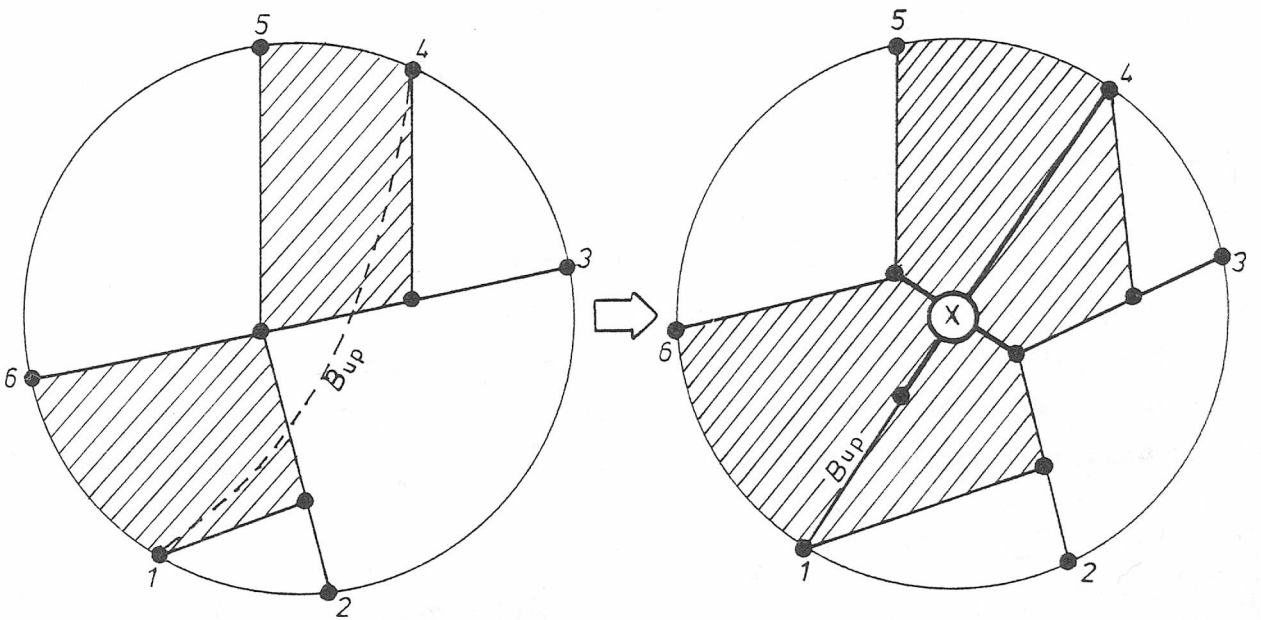
If the circuit requires many crossovers it is advisable to run the planarity program as a batch job overnight, say, due to the extended processing time required. Should this be the case it would then be possible for the program to investigate the effect of re-sequencing the Edge Node list. This would enable the user to select the order which resulted in the fewest number of crossovers (Refer to section 5.6).

It is important to note that this is different from simply starting the cyclic order description at a different place. It is true that this will cause different graphs to be generated but, assuming an efficient algorithm, the number of crossovers needed should be the same. Altering the cyclic order, on the other hand, affects the planarity of the circuit directly.

Each Edge Node in the Topography description could be marked in some way to indicate whether re-ordering is allowed. The computer would then generate a series of graphs by swopping moveable pairs.



(a) Edge pad common



(b) Single node common

Figure 11.1 Crossover insertion using Regions which touch at a POINT

### 11.3.2.3 Component Pad Swopping

The technique described in the last section could be taken a stage further by introducing pad-swopping in multi-terminal devices. Consider Figure 11.2 (a) which shows a graph with an unplanar branch Bup. If pads 3 and 2 are interchanged then the crossover can be avoided (diagram (b)). This technique can be applied to devices which have electrically identical terminals such as the inductor shown in (c). Note that pin 1 could not be swopped with either of the other two, but that pins 3 and 2 are completely interchangeable.

A similar philosophy could be adopted with SETS of pins. Figure 11.2 (d) illustrates an I.C. containing four identical three-pin devices. As before there are cases where two pin groups may be swopped - 1 and 2, 4 and 5 etc. In addition to this there is the possibility of interchanging complete sets such as Pins 1, 2, 3 and Pins 4, 5, 6.

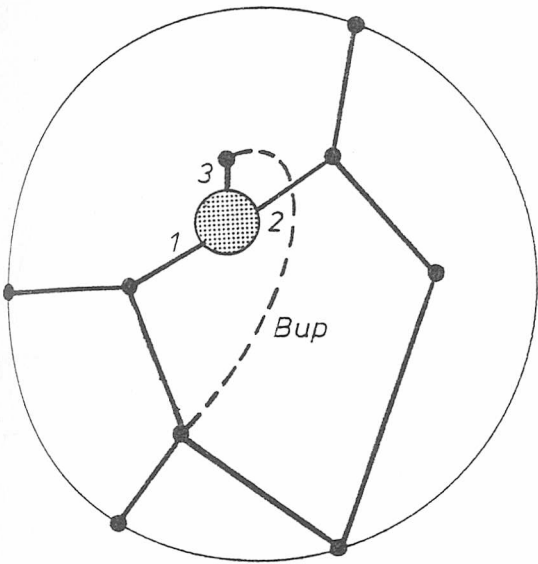
### 11.3.2.4 Routing Outside The Edge Pads

During fabrication, the Edge Pads are connected to package leads using jump wires. It is sometimes possible to take advantage of this physical characteristic in order to avoid a crossover.

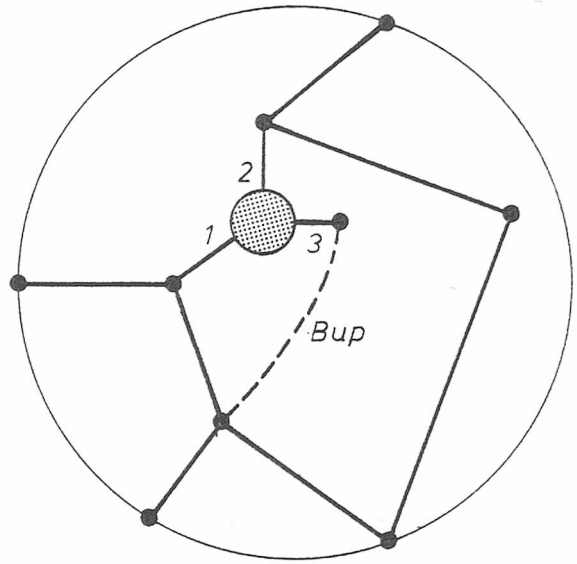
Consider Figure 11.3 (a) which shows a graph containing a single unplanar branch (R2). The important thing to notice here is that both ends of the component are attached to Regions containing Edge Pseudo Branches. If the corresponding Edge Pad(s) between the two Regions are moved away from the board edge (as shown in (b)) then it is possible to route a connection topologically outside the graph boundaries. The inevitable crossover is then formed when the Edge Pads are connected to the outside world.

### 11.3.2.5 Multiple Crossover Jumps

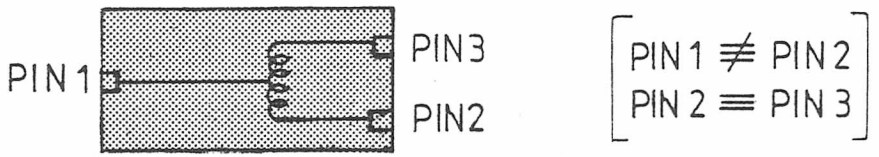
The definition of a crossover has so far been taken as the insertion of a jump wire across a single conductor track, but there is no reason why the wire should not bridge several tracks at once. It would be very difficult to program this technique into the Planarity Algorithm directly, but the addition of a post-processor to combine single jumps would be much easier to implement. Consider Figure 11.4 which illustrates how this could be carried out.



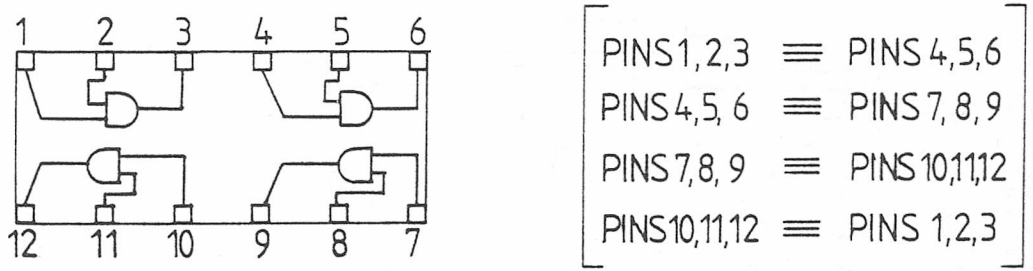
(a) original definition



(b) after pad swapping



(c) multi-pin device specification

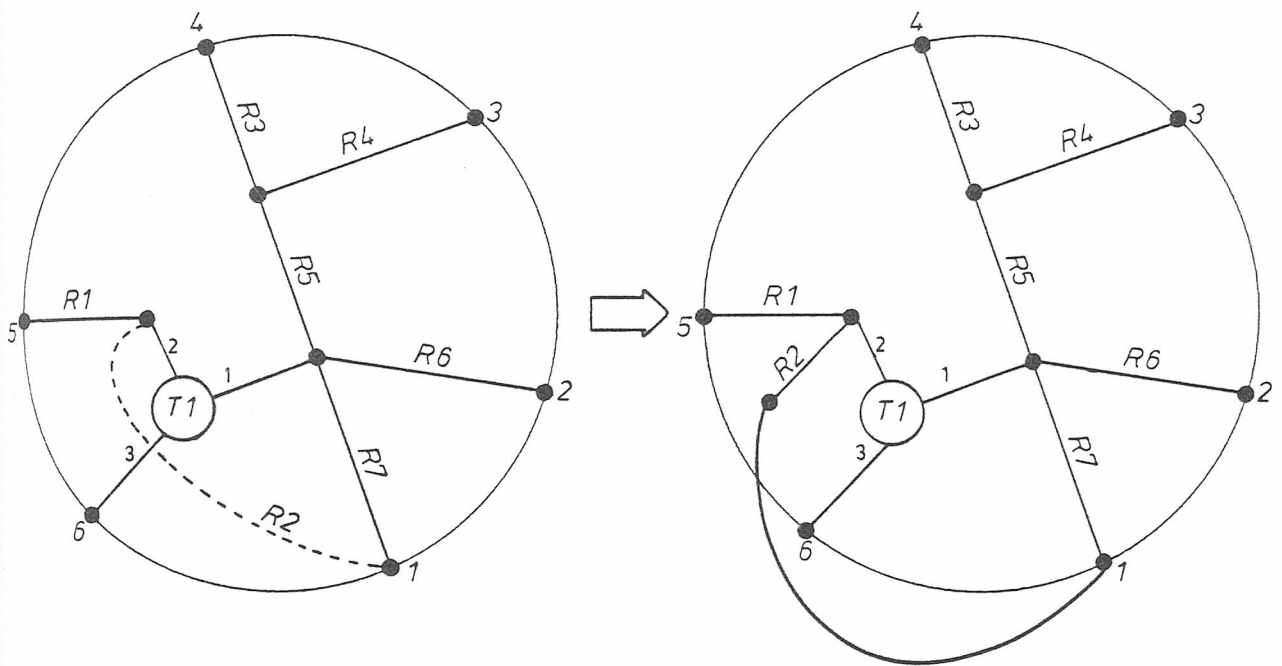


(d)

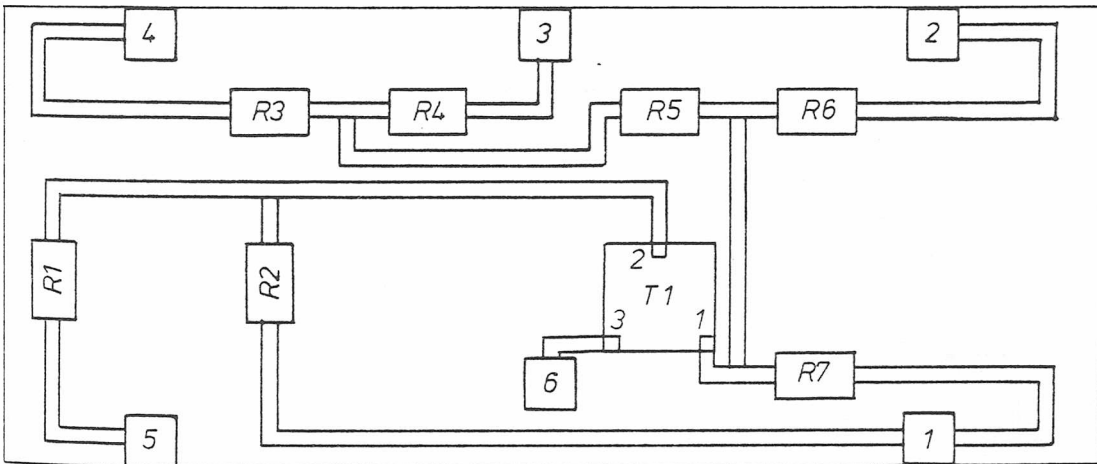
$\neq$  electrically dissimilar  
 $\equiv$  electrically identical

Figure 11.2 Topological pad swapping



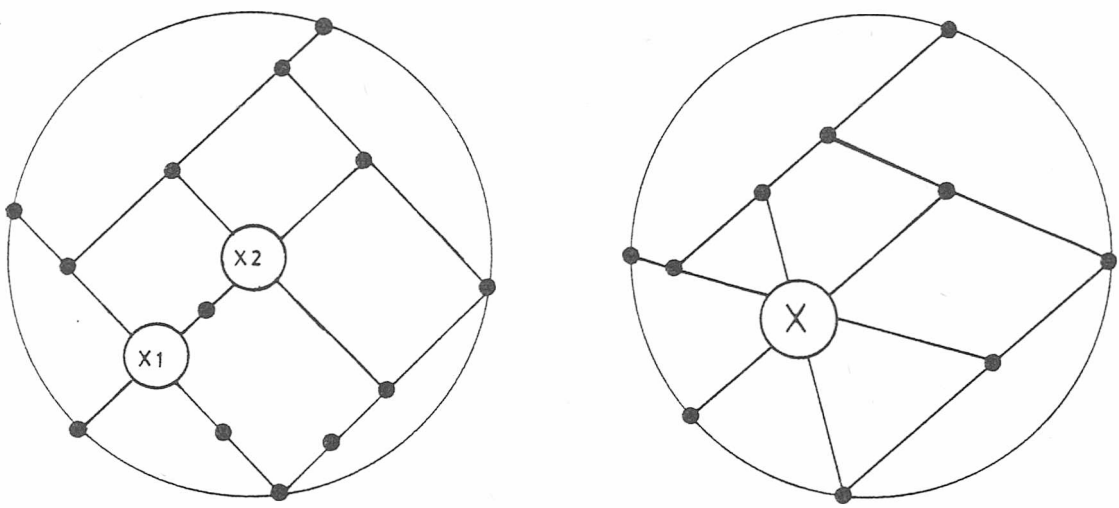


(a) Topological representation



(b) Layout equivalent

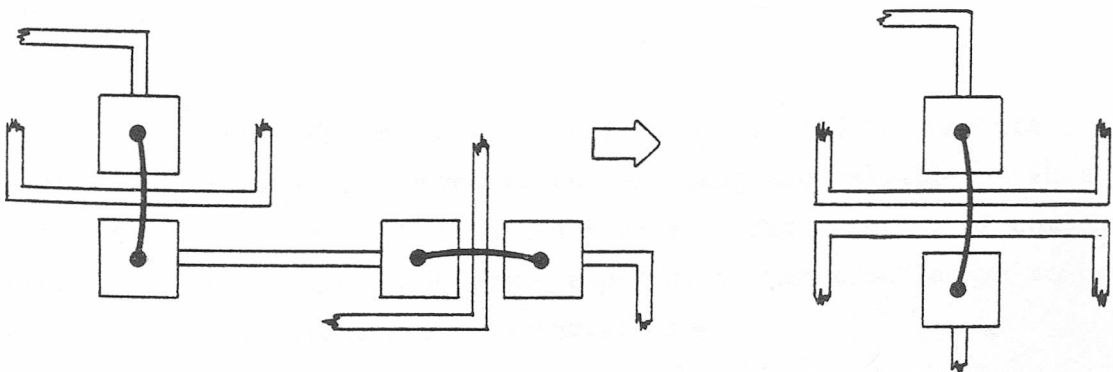
Figure 11.3 Routing round edge pads



(a) *Topological replacement*



(b) *Programmed representation*



(c) *Physical characteristics*

Figure 11.4 Replacing a string of crossovers by a single Multi-crossover device

The two crossovers "X1" and "X2" shown in (a) are simply replaced by a single multi-crossover device "X". This is similar to a normal crossover except that it incorporates more "shorted pairs" of pads. When the wire is attached (diagram (c)), both crossings are formed simultaneously. There is also a saving in board area and connection length.

#### 11.3.2.6 Routing Underneath Attached Components

A further technique to avoid crossovers is to route conductors through the bounding rectangles of attached devices. The component would then be placed immediately on top of the tracks. This may not be possible or desirable in every case because of interference problems or simply due to the component's physical characteristics. The designer would have to indicate this in the Master Description file.

Figure 11.5 demonstrates how this technique would be achieved in the topological graph. In this example the component Bup can be inserted without using a crossover if an extra pair of "shorted" pads are added to the device called TRAN. This is equivalent to incorporating pads into the Master Description as demonstrated in Figure 11.6.

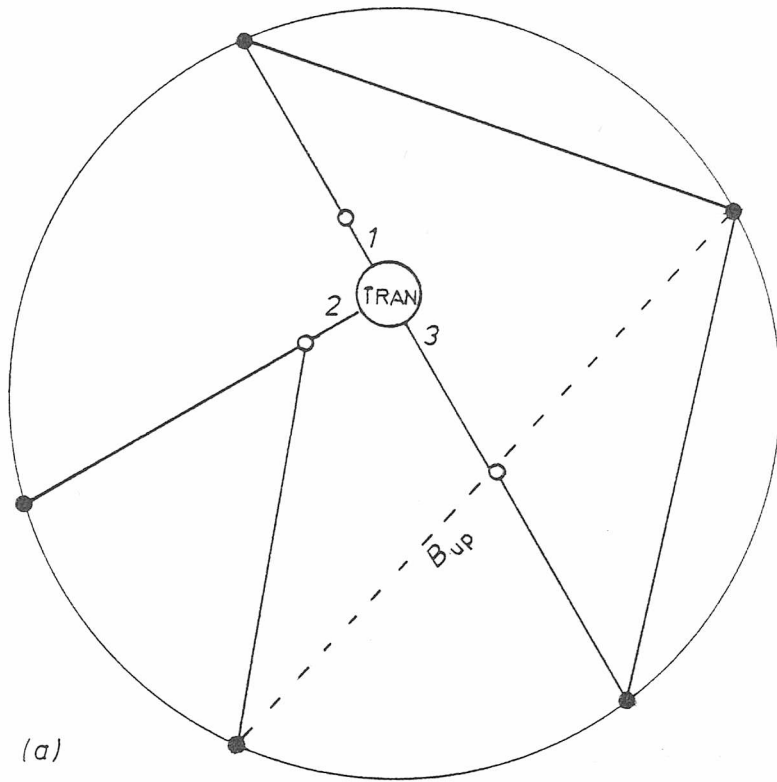
The number of tracks which can be routed under any particular component depends upon the conductor width relative to the size of the bounding rectangle and terminal pads. This could be calculated automatically.

### 11.4 The Placement Algorithms

#### 11.4.1 Conclusions

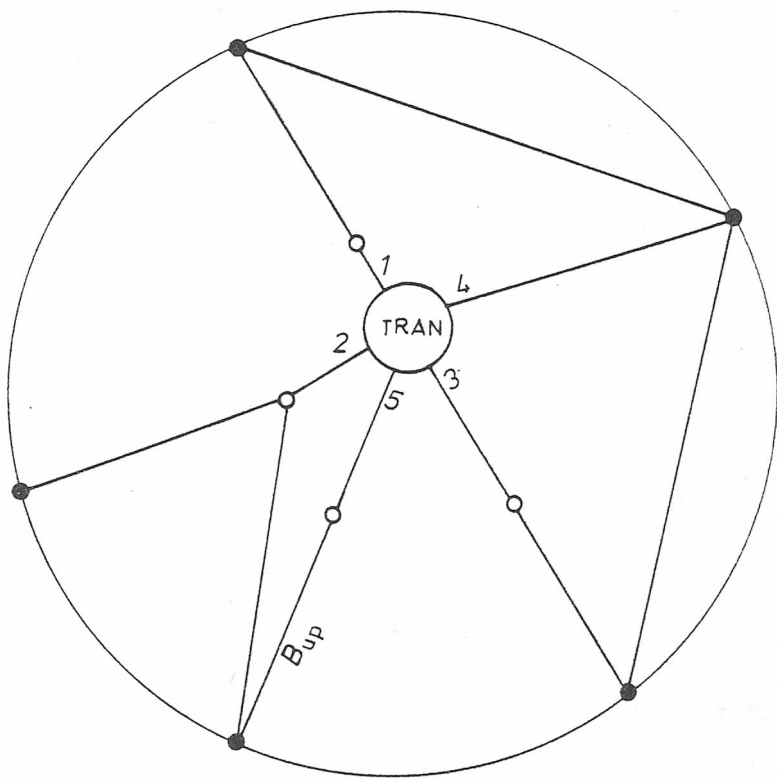
The Initial Placement Algorithm which is used to generate a base layer of components has proved to be very fast and reliable. This was to be expected since it is relatively easy to fit a string of components onto a level section of boundary - especially when troublesome members can simply be discarded for later processing.

The main Placement Algorithm has not been quite so successful. This was one of the final areas to be tackled and has consequently been subjected to fewer tests. The result is not always predictable and a certain amount of fault-finding is still required before the algorithm could be used to its full potential.



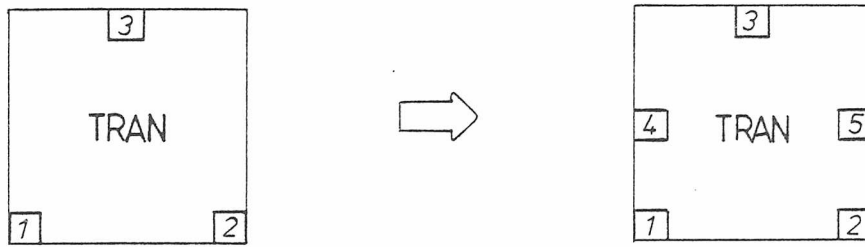
(a)

( $B_{up}$  = unplanar branch)

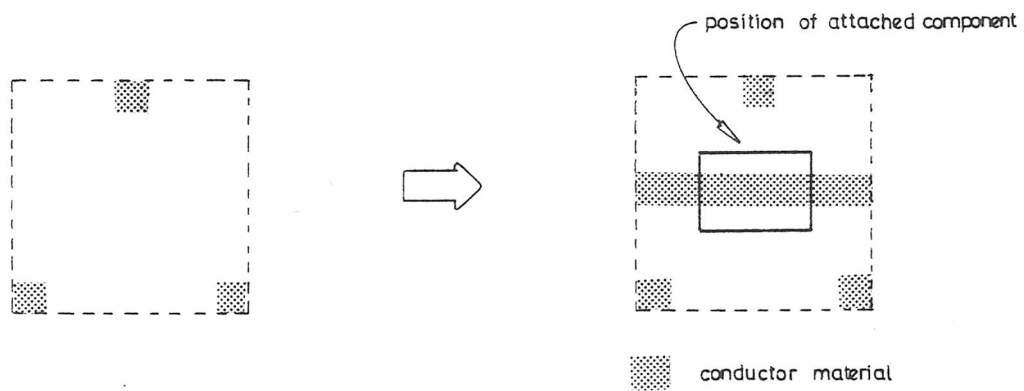


(b)

Figure 11-5 Going underneath an attached device — topological representation



(a) Program representation



(b) Physical characteristics

Figure 11.6 Going underneath an attached device

Although the algorithm may not always produce a totally satisfactory solution, it serves at the very least to bring the next components onto the board in the correct order (refer to Chapter 7). Any improvement will simply reduce the amount of human interaction at each stage.

Run times have proved to be very variable but are consistently low enough for "hands-on" design.

#### 11.4.2 Possible Improvements

##### 11.4.2.1 X Boundary Expansion Limits

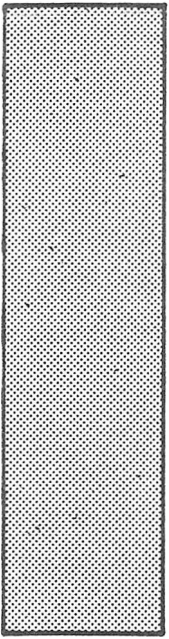
One of the techniques used by the main Placement Algorithm is that of X Boundary Expansion (section 7.4.4). It is difficult to decide when to abandon this approach in favour of other methods, and at present the expansion is limited by an arbitrary factor supplied on the intuition of the user. A much more satisfactory limit has been postulated in section 7.4.4 which guarantees to prevent the expansion obscuring pads belonging to previously placed components. The implementation of an algorithm to make this decision would be highly advantageous.

##### 11.4.2.2 Edge Pad Placement

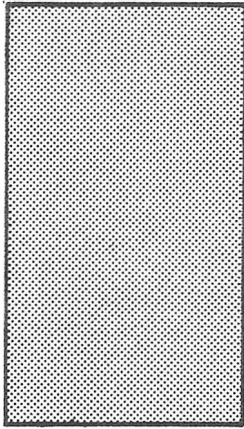
When using the Initial Placement program PLACE, the operator is required to specify which Edge Pads are to lie along the bottom edge of the board (refer to section 6.3). Since the number of components in the base layer depends entirely upon this decision, the horizontal width of the layout is largely determined at this stage. Consider Figure 11.7 which shows several possible layout dimensions which could result from the same circuit given different Edge Pad assignments. The number of alternatives is limited by the number of Edge Pads in the layout definition.

The total board area (Ba) must be at least as great as the total component area (Ca). Clearly some additional area must be allocated for conductor tracks and spacings - this is represented by a Packing Factor (Pf) in the equation:

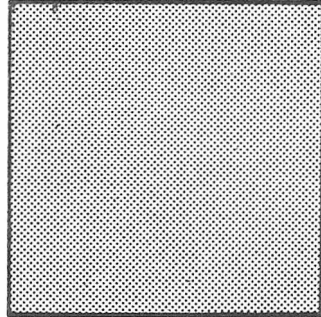
$$Ca = Ba \times Pf \quad (Pf < 1)$$



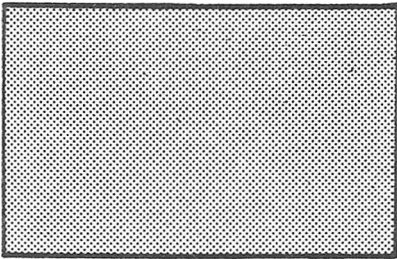
(a)



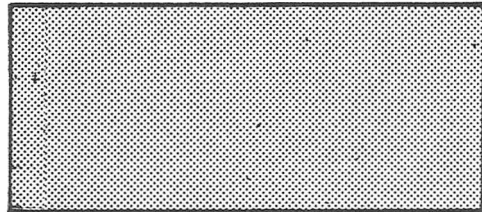
(b)



(c)



(d)



(e)

$$Ca = Pf(H \times W)$$

Ca = total component area

Pf = packing factor (<1)

W = board width

H = board height

ie. 
$$H = \frac{Ca \times Pf}{W}$$

Figure 11.7 Effect of edge pad assignment on board dimensions

The board area can also be calculated from its horizontal width (W) and vertical height (H) using the simple formula:

$$Ba = H \times W$$

If W is known (from an initial placement in which none of the base components have been rotated or removed) we can thus calculate the expected height to be:

$$H = \frac{Ca}{W \times Pf}$$

The Packing Factor will, of course, vary from layout to layout but an approximate value can be assumed.

This formula was applied to all three of the test circuits documented in Appendices 1 and 2. The results which were obtained are tabulated in Figure 11.8.

The layouts were found to have Packing Factors of 0.4063, 0.2940 and 0.3228 respectively. The average value (0.341) was then used to estimate the board height from the component area and board width. It can be seen that the % error ranges from + 19.14% to - 13.77%. This is quite encouraging but there is no guarantee that this is a typical range of Packing Factors.

The user, then, has the ability to choose an Edge Pad assignment which appears to produce the most promising board shape. A list of possible board sizes can be generated automatically and listed for his consideration but this would mean using the Initial Placement Algorithm many times. It is not clear whether this would increase the run time sufficiently to necessitate a batch processing approach.

A formula has been developed relating the number of iterations required to the number of Edge Pads in the layout (the derivation is illustrated in Figure 11.9). If Npad Edge Pads are present, the Algorithm will have to be applied  $Npad^2 - 2Npad + 1$  times. In general the number of Edge Pads will seldom exceed 30, which necessitates 841 iterations. This would certainly require the program to be run in a batch mode.



○ circuit under consideration

	(a)	(b)	(c)	
width of layout	160.5	158.5	160	(W)
total component area	8803.25	6105	7387.25	(Ca)
final height of layout	135	131	143	(H)
calculated packing factor	0.4063	0.2940	0.3228	$\left(\frac{Ca}{W \cdot H}\right) = Pf$
estimated height using packing factor Pf = 0.341	160.84	112.95	135.40	$\left(\frac{Ca}{Pf \cdot W}\right) = H2$
% out on height estimation	+19.14%	-13.77%	-5.31%	$\left(\frac{H2-H}{H}\right) \%$

(c) Voltage regulator  
(Appendix 2)

(b) Low voltage audio power  
amplifier (Appendix 2)

(a) Simple test circuit  
(Appendix 1)

Figure 11-8 Results from packing factor estimation experiments

Number of edge pads on bottom edge	Variations	Number of combinations
0	not allowed	—
1	not allowed	—
2	AB,BC,CD,DE, EA	Npad
3	ABC, BCD, CDE, DEA, EAB	Npad
4	ABCD, BCDE, CDEA, DEAB, EABC	Npad
5(=Npad)	ABCDE	1

Assumptions: 5 edge pads ( Npad = 5 )  
named A,B,C,D,E, F

⇒ Number of combinations

= Number of calls to initial placement algorithm

= ( Npad - 2 ) Npad + 1

=  $Npad^2 - 2 Npad + 1$

Figure 11.9 Equation relating the number of calls to the initial placement algorithm with the number of edge pads in the layout

It would be unusual, however, to allocate a large number of Edge Pads to the bottom edge and a small number to the top edge (or vice-versa). If the user were to specify a maximum ( $N_{max}$ ) and a minimum number ( $N_{min}$ ) of allowable bottom Edge Pads, the computation time would be cut considerably. It can be shown that the number of iterations ( $N_{it}$ ) is now given by the formulae:

$$N_{it} = \frac{(N_{max} - N_{min} + 1) N_{pad}}{N_{pad}} \quad \text{if } N_{max} < N_{pad}$$

and

$$N_{it} = \frac{(N_{pad} - N_{min}) N_{pad} + 1}{N_{pad}} \quad \text{if } N_{max} = N_{pad}$$

Taking the example of 30 Edge Pads, and max/min values of 18 and 12 respectively, the number of iterations falls to 210.

#### 11.4.2.3 Choice Of Placement Technique

The Main Placement Algorithm described in Chapter 7 uses a number of different placement techniques in a rigid order of preference. It is clear that filling the Slots (section 7.4.1) will produce the most compact layout. The next best methods are to use mounds as well as slots (7.4.2), or to perform component rotation (7.4.3).

When all three fail it is not at all clear which of the remaining three (X Boundary Expansion, Component Removal and Rough Placement) should be attempted next. An improvement to the layout suite would be to allow the user to select the approach which would appear to be the most appropriate in each situation. It may be possible to develop an algorithm to do this selection automatically but this is, in itself, a complex problem.

### 11.5 The Routing Algorithm

#### 11.5.1 Conclusions

The Routing Algorithm has proved to be fairly successful and generally very speedy. This was one of the later areas of research and has not been in use long enough to locate every minor fault. Nevertheless, it serves in its present state to free the designer from the simpler connections and gives him consequently more time to spend on the longer more complex ones.

### 11.5.2 Possible Improvement

Even if we assume that the Algorithm will always find the best path between any two components, there is still no guarantee that the layout will be constructed with an absolute minimum conductor length. This can be explained with reference to Figure 11.10.

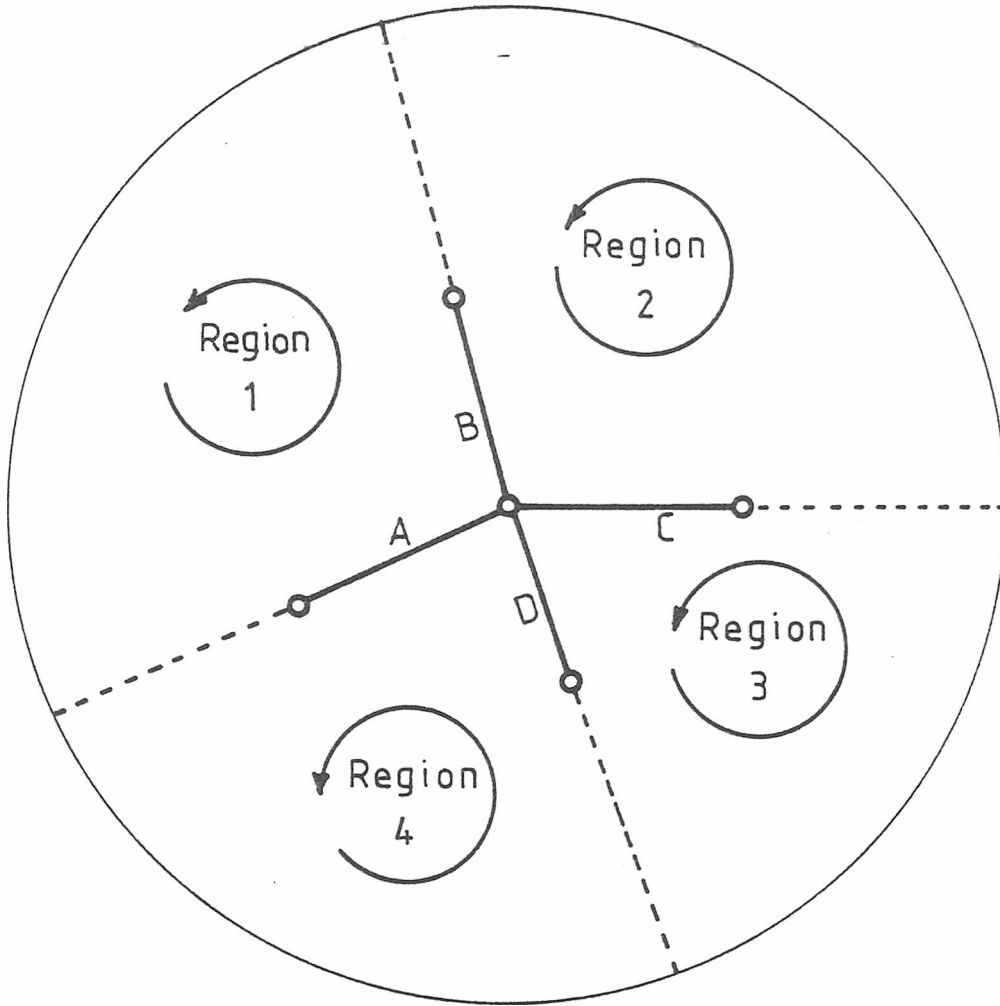
Diagram (a) shows a small section of a larger topological graph. Using the Region process of generating connection pairs (see section 9.3.3), the Routing Algorithm would be applied between components A and B then B and C etc as shown. Assuming that the first three attempts are successful, the final connection between D and A already exists as a string through B and C. This could give a physical layout as shown in diagram (b).

It can be seen that each connection is of minimum length but the overall string could be improved simply by selecting a different set of component pairs.

A better way of generating component pairs would be to process the components sequentially rather than following the Regions. The program would be designed to ensure that each component was connected to its nearest neighbour in the string.

This would involve a good deal more processing time than the simple Region method but would produce much simpler metallisation patterns. Diagram (c) shows the improvement which would result in this example. One major problem, however, is that "isolated groups" can be formed. Consider diagram (d) in which component C has been moved such that it is now closer to A than component B. Only two connection attempts would now be generated namely D to B and A to C. In this example every component is indeed connected to its nearest neighbour but the string is not continuous.

The solution would be to apply the Region method in conjunction with the "nearest neighbour" method. Having produced the network shown in diagram (d), the Region method would join components A and B. The other three attempts (namely B to C, C to D and D to A) would then be abandoned since the paths already exist directly or indirectly.

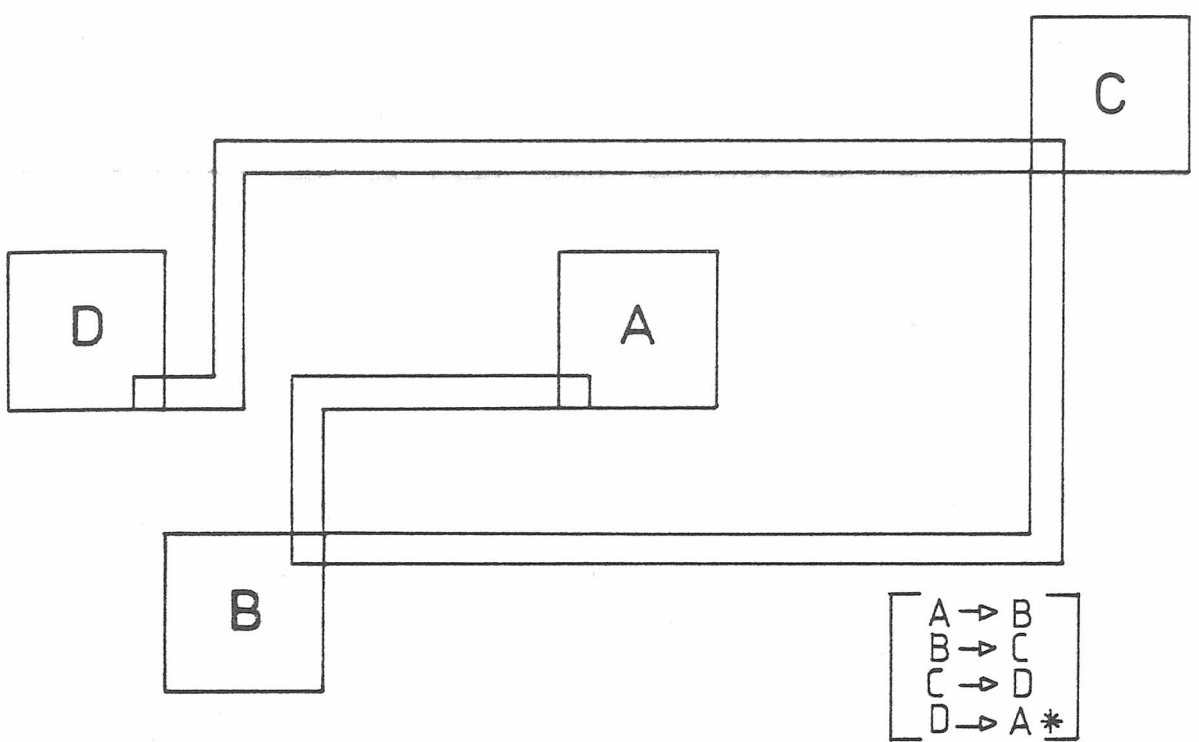


Connections:

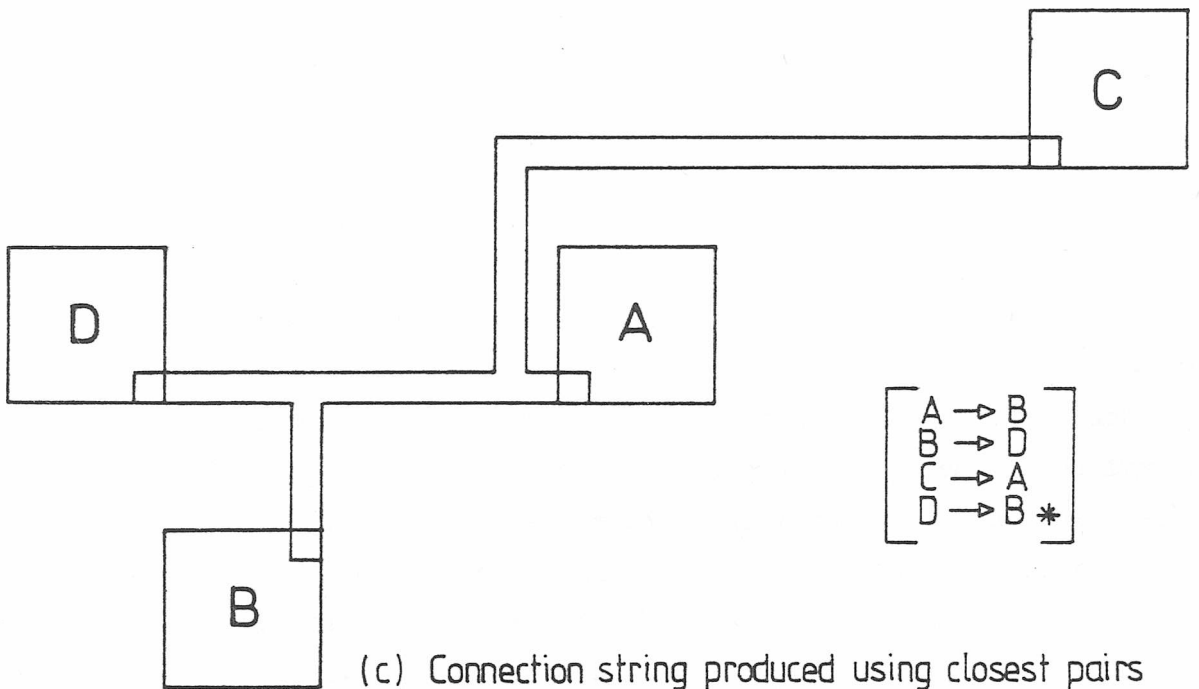
- Region 1 : Component A to Component B
- Region 2 : Component B to Component C
- Region 3 : Component C to Component D
- Region 4 : Component D to Component A

(a) Section of Topological Graph

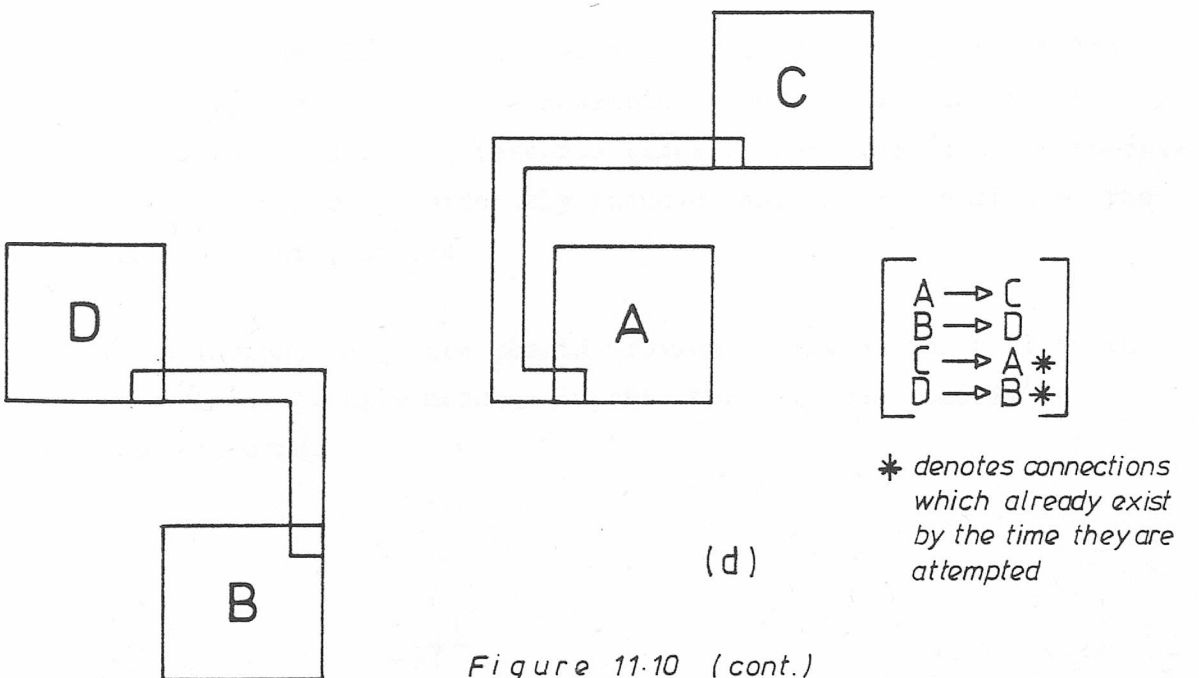
Figure 11.10 Minimum connection net techniques



(b) Connection string produced by region method



(c) Connection string produced using closest pairs



(d)

## 11.6 Photoplotting

Although the photoplotting algorithms have not been used to any great extent due to the delay in manufacture of slit apertures, they have been exhaustively tested using a graphical simulation program (refer to section 10.13). The suite also includes a timing program which can calculate the total plotting time of a particular layout and thus evaluate the merits of different filling strategies without actually having to do the plotting.

## 11.7 Final Conclusions

From the results described in earlier chapters it is clear that a feasible method of designing thin film circuits by computer has been developed. The layout examples given in Appendices 1 and 2 compare favourably with manually generated solutions, but the design process is very much faster. There is also the advantage that numerically controlled fabrication devices can be driven directly from the data structure. This eliminates the process of coding up the layout from a production drawing.

One improvement would be to use a refresh graphics terminal rather than the storage terminal. This would enable the user to move components and connections dynamically without having to redraw the layout at each step. It is also possible to obtain flashing and different intensity lines. This would be much better ergonomically speaking, but the cost of refresh equipment is much higher than the storage tube equivalent. There is also the problem that the refresh screen cannot display nearly so much detail as the storage terminal.

If a dedicated mini-computer were to be used instead of a multi-access mainframe, then it would be possible to operate at much higher transmission speeds and faster response times. The time taken to re-draw the screen would then be considerably reduced, and the ergonomics of the system would therefore improve.

In conclusion, the suite should provide a very useful tool to the designer. Being totally modular in structure, it can readily be extended and improved.

## Appendix 1

### Example Of Layout Design Using

#### A Simple Test Circuit

The use of the "LAYOUT" suite as a design tool is best illustrated by example. Consider Figure A1.1 which shows a simple circuit comprising three transistors, nine resistors, two capacitors and a diode. This circuit was specifically designed with the object of testing the design programs, and contains the complete range of circuit types i.e.:

- (a) Multi-pad attached components (TR1, TR2, TR3).
- (b) Two-pad attached components without a marked pin of polarity (R1, R2, R3, R4, C1, C2).
- (c) Two-pad component with a marked pin of polarity (DIOD).
- (d) In-slice meandering resistors (R5, R6, R7, R8, R9).

Several of the resistors were chosen to be in-slice components, so the first task was to design them using the program "DESRES". A full description of this program is given in Chapter 3.2.

Two distinct resistor patterns were generated and stored in a temporary data file (called MR1 and MR2 respectively). The track width was set to 20 screen units and resistances of 30 and 60 squares were requested. This gave patterns measuring 115 x 170 and 150 x 208 units respectively.

Physical sizes and pad positions were then chosen for the discrete components which were called "RES", "TRAN", "CAP" and "DIODE". These are illustrated in Figure A1.2.

The next step was to assign arbitrary Node Numbers to each electrical node in the circuit as shown in the circuit diagram in Figure A1.1.



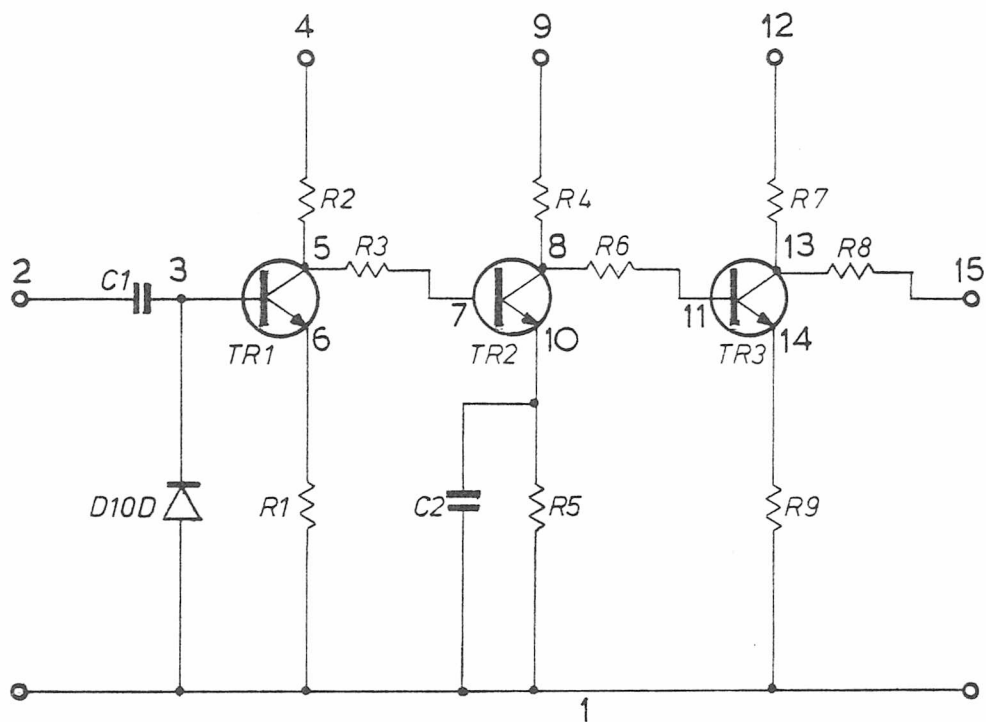
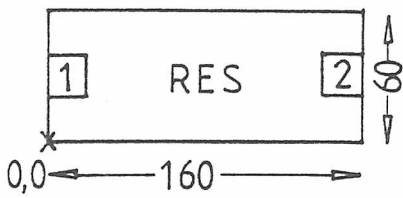
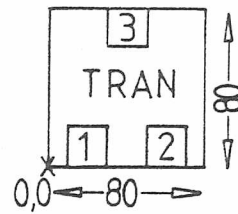


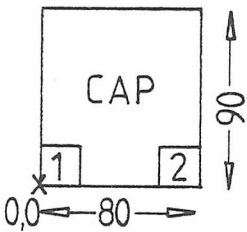
Figure A1-1 Simple test circuit



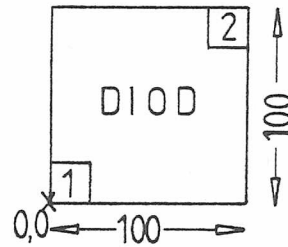
`RES`  
 width = 160  
 height = 60  
 PAD1 at (10,30)  
 PAD2 at (150,30)



`TRAN`  
 width = 80  
 height = 80  
 PAD1 at (20,10)  
 PAD2 at (60,10)  
 PAD3 at (40,70)



`CAP`  
 width = 80  
 height = 90  
 PAD1 at (10,10)  
 PAD2 at (70,10)



`DIOD`  
 width = 100  
 height = 100  
 PAD1 at (10,10)  
 PAD2 at (90,90)

Figure A1.2 Master component definitions for attached devices

A data file was then formulated to describe the component geometries and circuit topology as illustrated in A1.3. The compilation of such a description is explained in Chapter 4 sections 2 and 3. Note that components R5, R6, R7, R8 and R9 have been assigned to the in-slice resistor descriptions "MR1" and "MR2" which are stored in a separate data file.

The program "PLANAR" (see Chapter 5.7) is used to generate a topological graph of the circuit from the two data files mentioned above. An optional print out of this graph is given in Figure A1.4. Each "Region" is listed in turn, giving the component names (and associated Node Numbers) in the order in which they appear. This can be used to draw a graphical representation of the topology as shown in Figure A1.5 - a very useful technique for trapping program errors while the suite is being developed. With medium and large scale circuits it would be unusual to produce such a drawing unless the program was malfunctioning in some way.

The program "PLACE" was next used to generate an initial board placement (refer to Chapter 6 for more details). At this stage it was necessary to decide upon values for board width and height, conductor width, spacing allowance and Edge Pad size. Values of 1000, 1000, 20, 20 and 30 were chosen respectively. It was also necessary to decide where to position the Edge Pads. After a number of trial runs it was found that the most promising results were obtained by placing Edge Pads 4, 9 and 12 along the top edge, and Pads 2, 1 and 15 along the lower. Figure A1.6 shows the layout obtained in this way.

At this stage the main design program called "LAYOUT" was employed. Working directly from the initial placement, it incorporates a number of facilities designed to carry the design to a successful conclusion (refer to Chapter 9).

The design can now be split into a number of distinct stages. A brief description has been given in each case.

Stage 1 (Figure A1.7)

- (a) All possible connections have been routed using the "CONNECT" facility.

Thin Film Circuit with Pre-defined Resistors

MASTER RES,2,160,60  
10,30  
150,30

MASTER TRAN,3,80,80  
20,10  
60,10  
40,70

MASTER CAP 2,80,90  
10,10  
70,10

MASTER DIODE,2,100,100  
10,10  
90,90

MASTER MR1,2

MASTER MR2,2

STOP

RES  
R1,6,1  
R2,4,5  
R3,5,7  
R4,9,8

MR1  
R5,10,1  
R6,10,11

MR2  
R7,12,13  
R8,13,15  
R9,14,1

TRAN  
TR1,3,6,5  
TR2,7,10,8  
TR3,11,14,13

CAP  
C1,2,3  
C2,10,1

DIODE  
DIOD,3,1

EDGE  
1 15 12 9 4 2

STOP

PLANAR GRAPH FOR MCIR1.CIR  
 THIN FILM CIRCUIT WITH PRE-DEFINED RESISTORS

.....  
 REGION NUMBER 1  
 .....

EDGE PSEUDO BRANCH - NODES (1) TO (2)  
 EDGE PSEUDO BRANCH - NODES (2) TO (4)  
 EDGE PSEUDO BRANCH - NODES (4) TO (9)  
 EDGE PSEUDO BRANCH - NODES (9) TO (12)  
 EDGE PSEUDO BRANCH - NODES (12) TO (15)  
 EDGE PSEUDO BRANCH - NODES (15) TO (1)  
 .....

REGION NUMBER 2  
 .....

EDGE PSEUDO BRANCH - NODES (12 TO (15)  
 2 PIN COMPONENT "R7" NODES (12,13)  
 2 PIN COMPONENT "R8" NODES (13,15)  
 .....

REGION NUMBER 3  
 .....

2 PIN COMPONENT "R7" NODES (12,13)  
 EDGE PSEUDO BRANCH - NODES (9) TO (12)  
 2 PIN COMPONENT "R4" NODES (9,8)  
 PIN 3 OF "TR2" (NODE 8)  
 LINK BRANCH : TR2 PIN 2 TO PIN 3  
 PIN 2 of "TR2" (NODE 10)  
 2 PIN COMPONENT "R6" NODES (10,11)  
 PIN 1 of "TR 3" (NODE 11)  
 LINK BRANCH : TR3 PIN 3 TO PIN 1  
 PIN 3 of "TR3" (NODE 13)  
 .....

REGION NUMBER 4  
 .....

LINK BRANCH : TR3 PIN 3 TO PIN 1  
 LINK BRANCH : TR3 PIN 1 TO PIN 2  
 LINK BRANCH : TR3 PIN 2 TO PIN 3  
 .....

REGION NUMBER 5  
 .....

LINK BRANCH : TR2 PIN 2 TO PIN 3  
 LINK BRANCH : TR2 PIN 3 TO PIN 1  
 LINK BRANCH : TR2 PIN 1 TO PIN 2  
 .....

REGION NUMBER 6  
 .....

EDGE PSEUDO BRANCH - NODES (15) TO (1)  
 2 PIN COMPONENT "R8" NODES (13,15)  
 PIN 3 OF "TR 3" (NODE 13)  
 LINK BRANCH : TR3 PIN 2 TO PIN 3  
 PIN 2 of "TR3" (NODE 14)  
 2 PIN COMPONENT "R9" NODES (14,1)  
 .....

Figure A1.4 Print-out of Planar Graph

REGION NUMBER 7

.....  
2 PIN COMPONENT "R9" NODES (14,1)  
PIN 2 of "TR3" (NODE 14)  
LINK BRANCH : TR3 PIN 1 TO PIN 2  
PIN 1 of "TR3" (NODE 11)  
2 PIN COMPONENT "R6" NODES (10,11)  
2 PIN COMPONENT "C2" NODES (10,1)  
.....

REGION NUMBER 8

.....  
2 PIN COMPONENT "C2" NODES (10,1)  
2 PIN COMPONENT "R5" NODES (10,1)  
.....

REGION NUMBER 9

.....  
2 PIN COMPONENT "R5" NODES (10,1)  
PIN 2 OF "TR2" (NODE 10)  
LINK BRANCH : TR2 PIN 1 TO PIN 2  
PIN 1 OF "TR2" (NODE 7)  
2 PIN COMPONENT "R3" NODES (5,7)  
PIN 3 of "TR1" (NODE 5)  
LINK BRANCH : TR1 PIN 2 TO PIN 3  
PIN 2 OF "TR1" (NODE 6)  
2 PIN COMPONENT "R1" NODES (6,1)  
.....

REGION NUMBER 10

.....  
LINK BRANCH : TR1 PIN 2 TO PIN 3  
LINK BRANCH : TR1 PIN 3 TO PIN 1  
LINK BRANCH : TR1 PIN 1 TO PIN 2  
.....

REGION NUMBER 11

.....  
2 PIN COMPONENT "R3" NODES (5,7)  
PIN 1 OF "TR2" (NODE 7)  
LINK BRANCH : TR2 PIN 3 TO PIN 1  
PIN 3 OF "TR2" (Node 8)  
2 PIN COMPONENT "R4" NODES (9,8)  
EDGE PSEUDO BRANCH - NODES (4) TO (9)  
2 PIN COMPONENT "R2" NODES (4,5)  
.....

REGION NUMBER 12

.....  
EDGE PSEUDO BRANCH - NODES (1) TO (2)  
2 PIN COMPONENT "DIOD" NODES (3,1)  
2 PIN COMPONENT "C1" NODES (2,3)  
.....

REGION NUMBER 13

.....  
2 PIN COMPONENT "DIOD" NODES (3,1)  
2 PIN COMPONENT "R1" NODES (6,1)  
PIN 2 OF "TR1" (NODE 6)  
LINK BRANCH : TR1 PIN 1 TO PIN 2  
PIN 1 OF "RE1" (NODE 3)  
.....

Figure A1.4 (continued)

REGION NUMBER 14

.....  
LINK BRANCH : TR1 PIN 3 TO PIN 1  
PIN 3 OF "TR1" (NODE 5)  
2 PIN COMPONENT "R2" NODES (4,5)  
EDGE PSEUDO BRANCH - NODES (2) TO (4)  
2 PIN COMPONENT "C1" NODES (2,3)  
PIN 1 of "TR1" (NODE 3)

Figure A1-4 (continued)

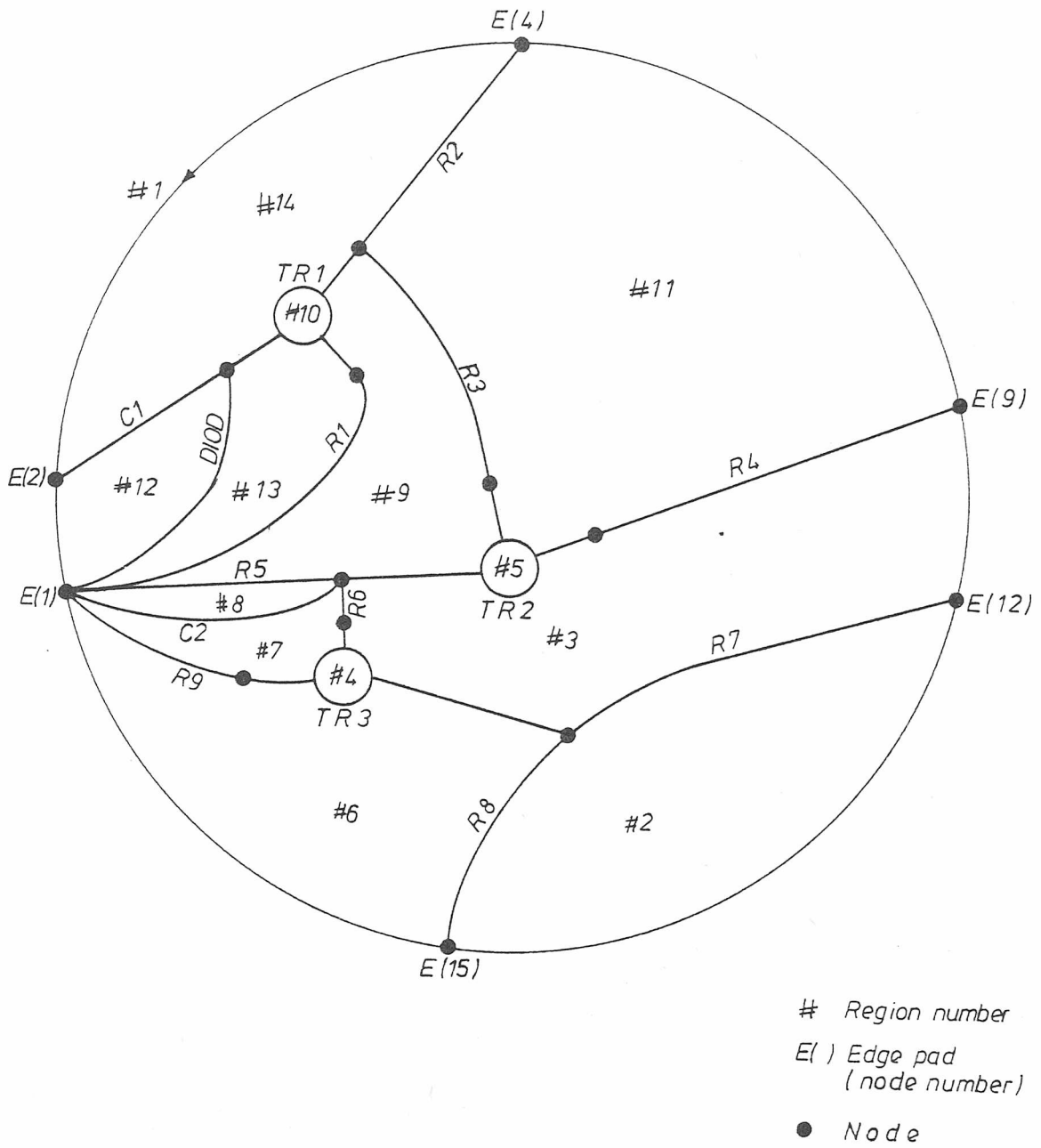
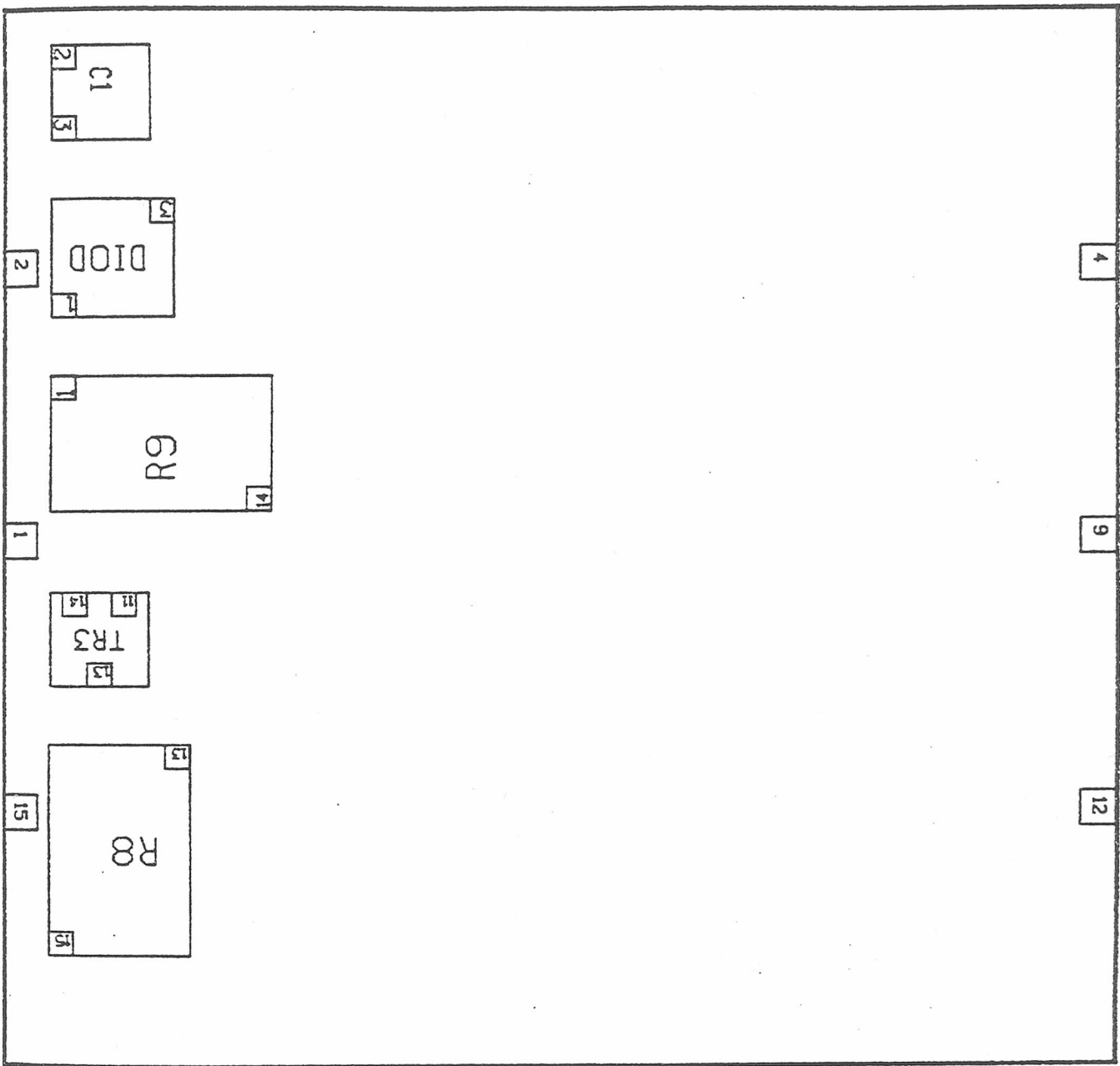


Figure A1.5 Planar graph of circuit





TOP  
BOTTOM

Figure A1-6 Output from program "PLACE"

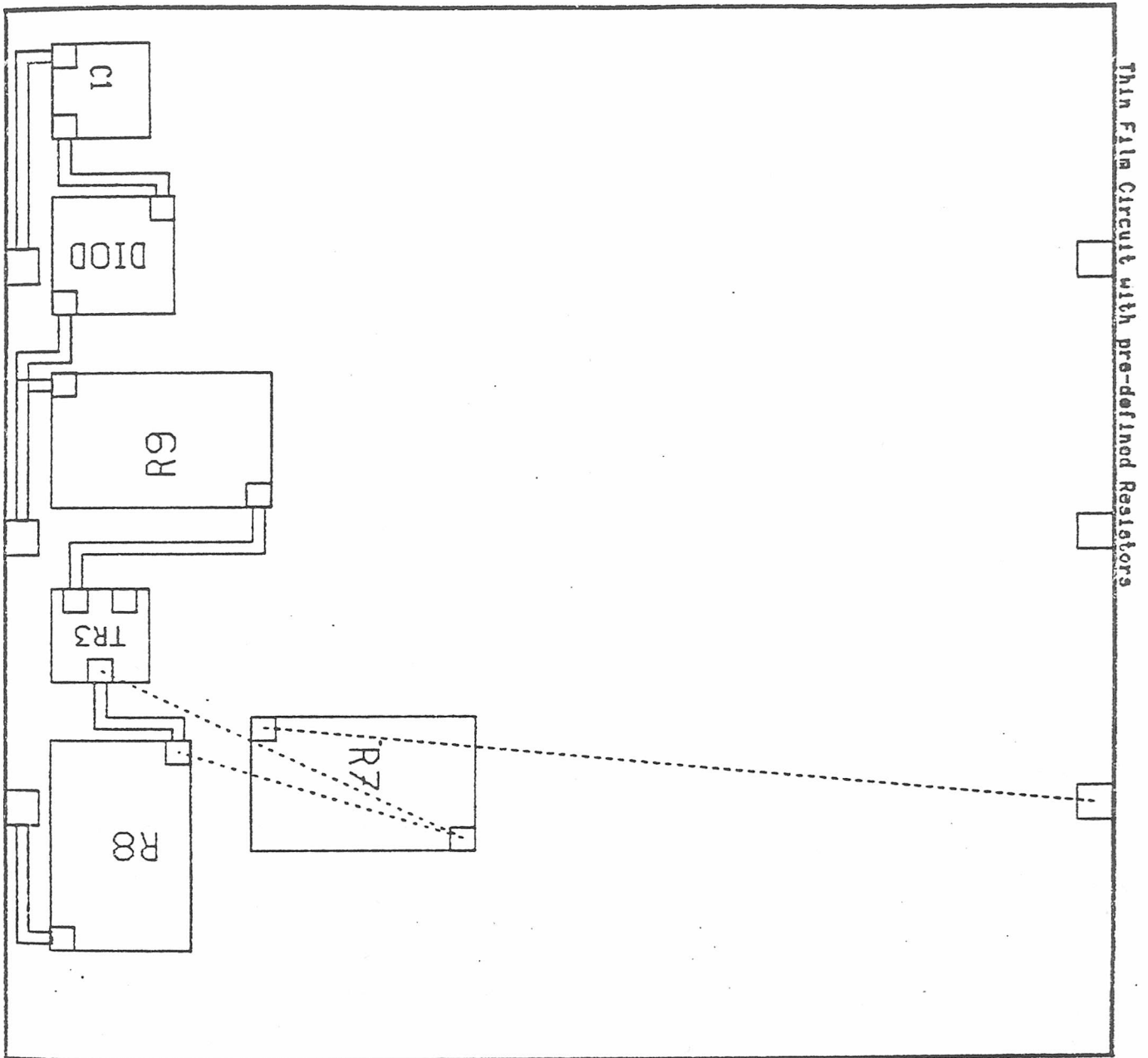


Figure A1-7 Stage 1 in "LAYOUT"

- (b) The next component (R7) has been brought onto the board using the "NEXT" facility.
- (c) Un-formed connections have been indicated as dotted lines as a guide to interaction ("QUERY" facility).

Stage 2 (Figure A1.8)

- (a) R7 rotated and re-positioned to improve layout.
- (b) One connection auto-routed. The other has been left since it goes to an Edge Pad on the top of the board (to avoid congesting further calls to the placement algorithm).

Stage 3 (Figure A1.9)

- (a) Next components brought onto board (R6 and C2).
- (b) Connections indicated by "SPRINGS" facility.

Stage 4 (Figure A1.10)

- (a) Vertical strip inserted between components R9 and R8 using the "BOARD" facility such that R6 will fit slot.
- (b) R6 and C2 rotated and re-positioned.
- (c) Connection auto-routed.

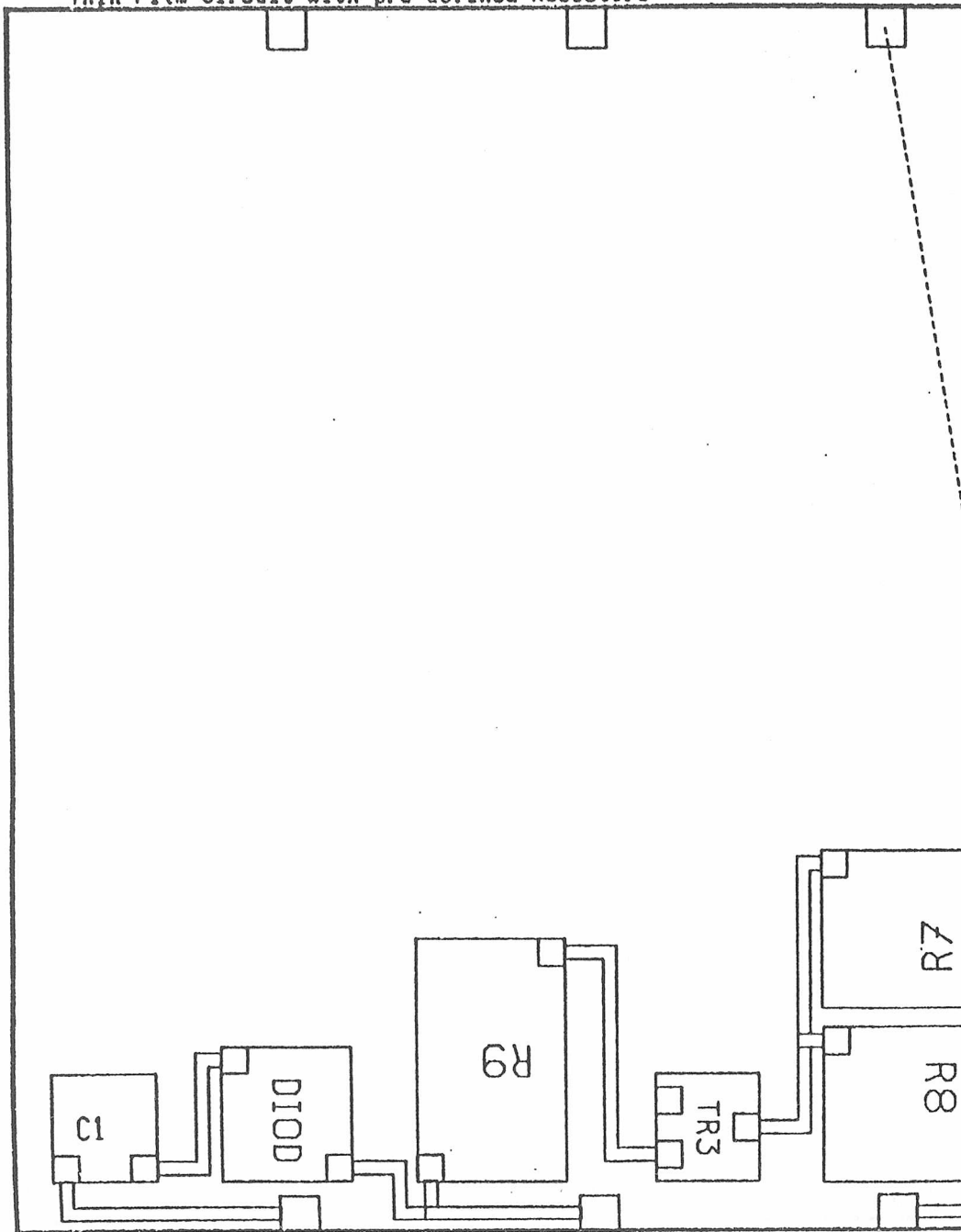
Stage 5 (Figure A1.11)

- (a) Next components brought onto board (TR2 and R4).
- (b) Connections indicated by "SPRINGS".

Stage 6 (Figure A1.12)

- (a) Component R4 re-positioned.
- (b) Component TR2 rotated and re-positioned.
- (c) One connection routed, one left (to avoid interference with future placements).

Thin Film Circuit with pre-defined Resistors

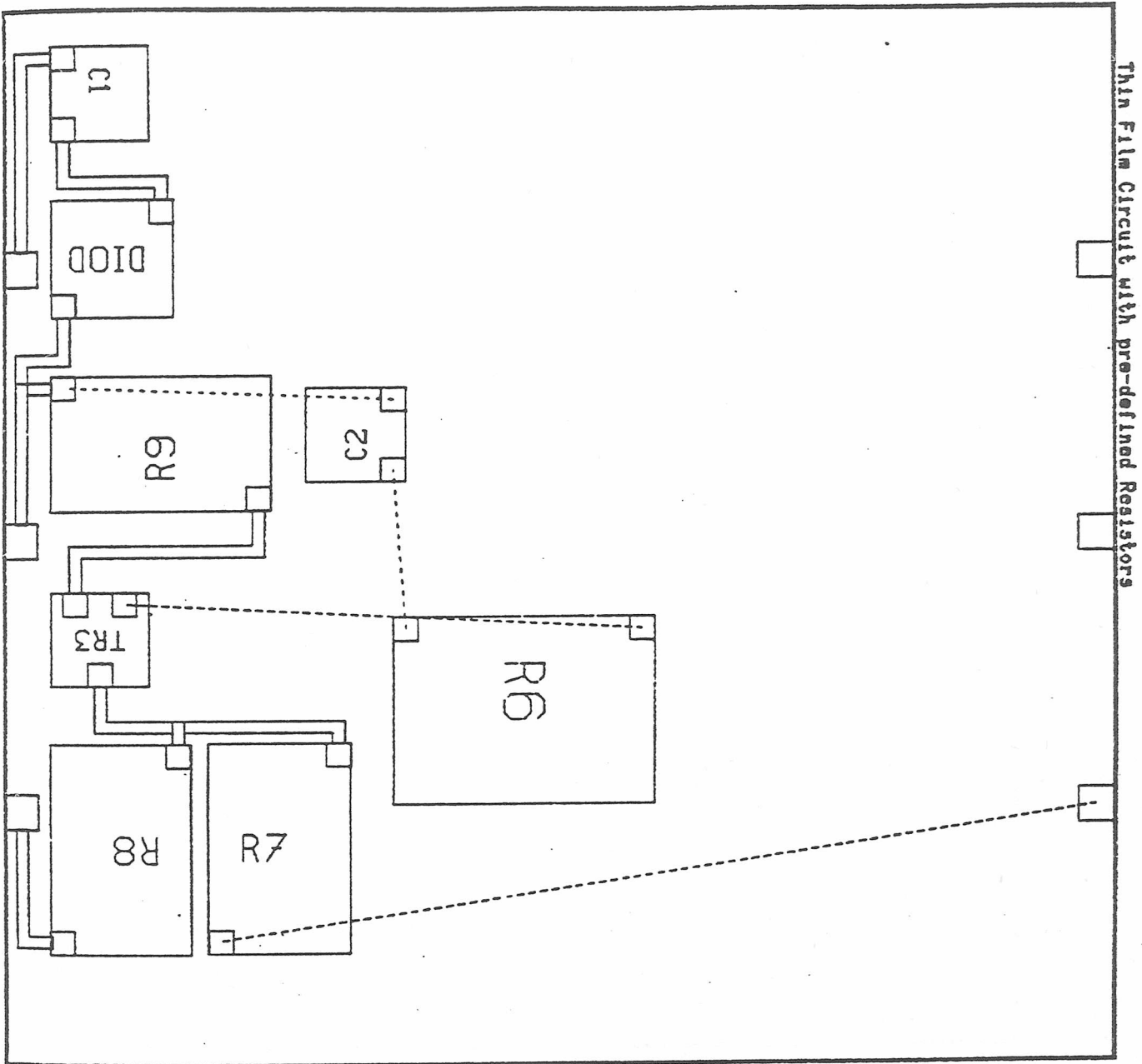


What Next ?

\$SPR

\*

Figure A1.8 Stage 2 in "LAYOUT"



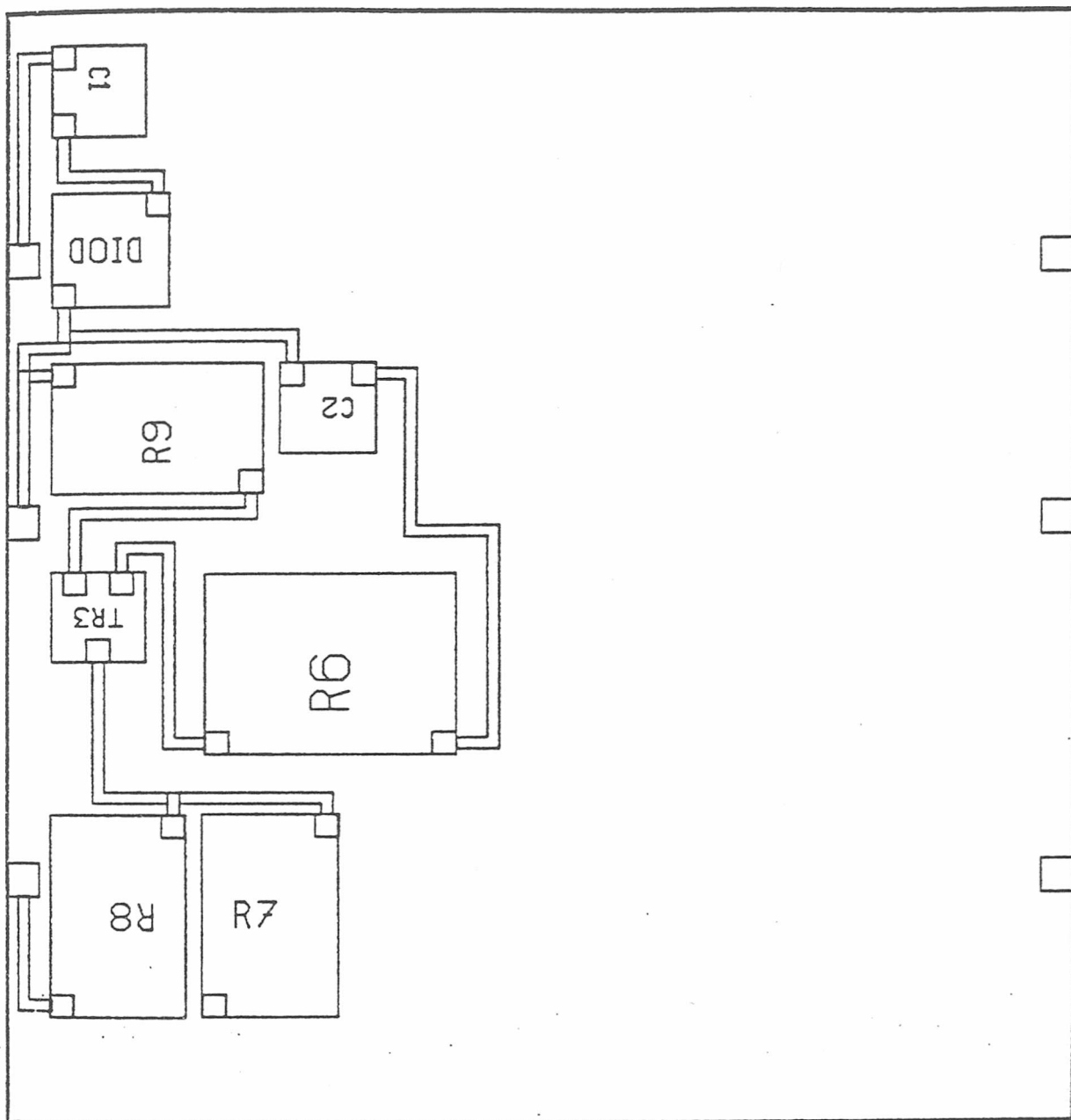
Thin Film Circuit with pre-defined Resistors

What Next ?

\$SPR

\*

Figure A1-9 Stage 3 in "LAYOUT"



What Next ?

Figure A1.10 Stage 4 in LAYOUT

What Next ?

‡SPR

‡

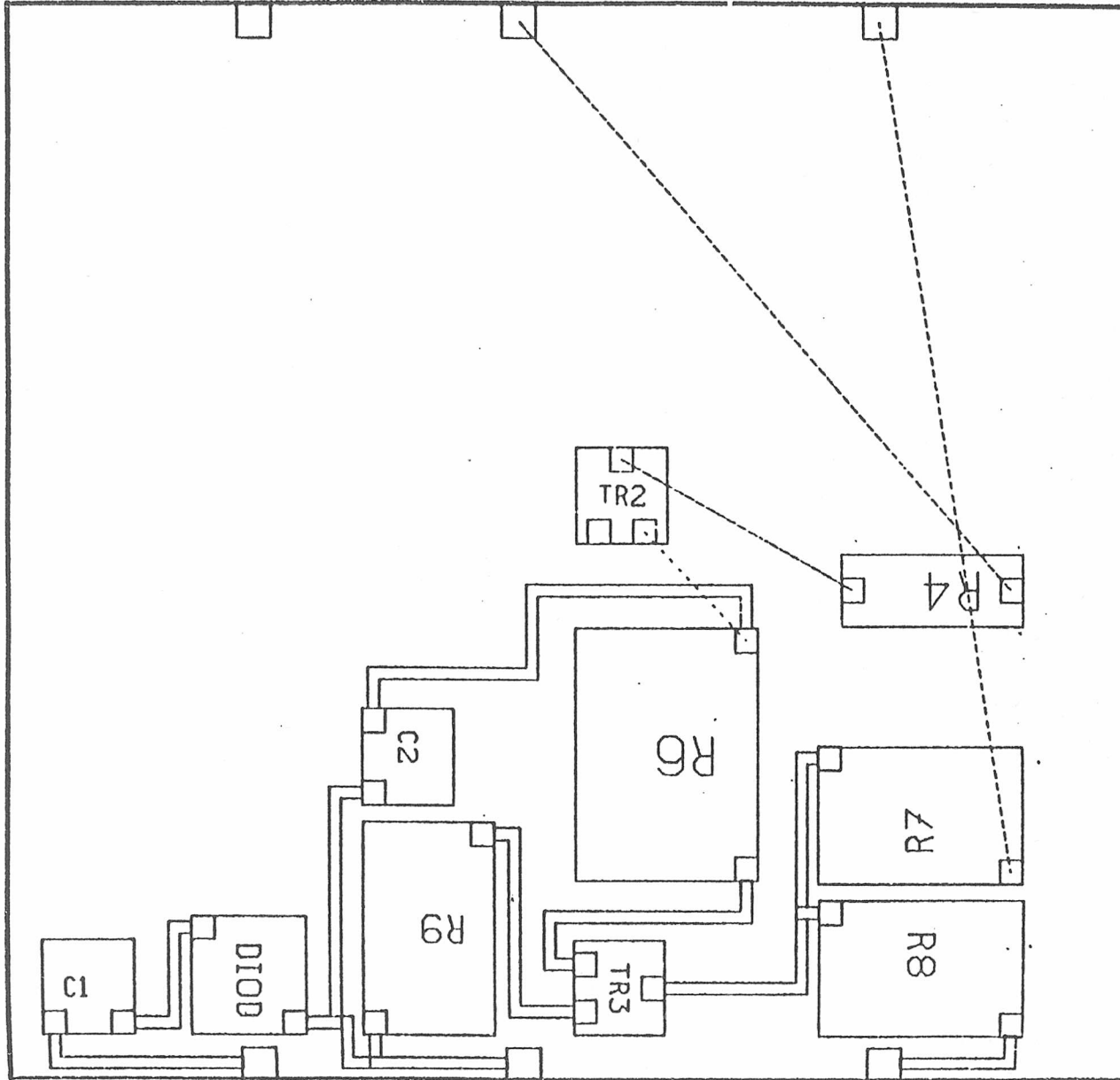


Figure A1.11 Stage 5 in "LAYOUT"



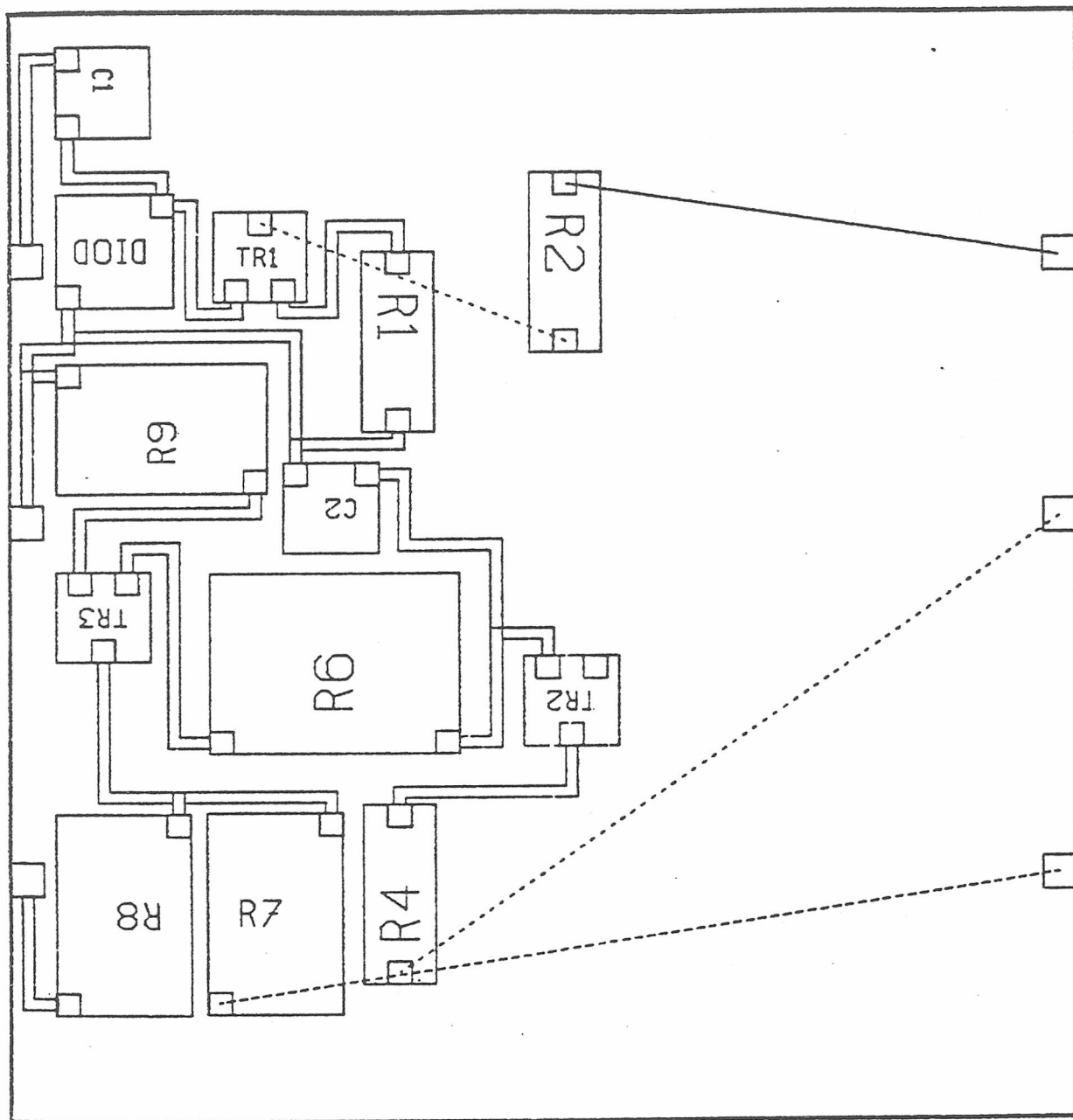


Figure A1-12 Stage 6 in "LAYOUT"

(d) Next component brought onto board (R2).

(e) Connections indicated with "SPRINGS"

Stage 7 (Figure A1.13)

(a) R2 rotated and re-positioned.

(b) One connection routed, one left (to avoid congestion).

(c) Next component brought onto board (R5).

(d) Connections indicated by "SPRINGS" facility.

Stage 8 (Figure A1.14)

(a) Component R5 rotated and re-positioned.

(b) Connections auto-routed.

(c) Next component brought onto board (R3).

(d) Connections indicated by "SPRINGS".

Stage 9 (Figure A1.15)

(a) Component R5 re-positioned.

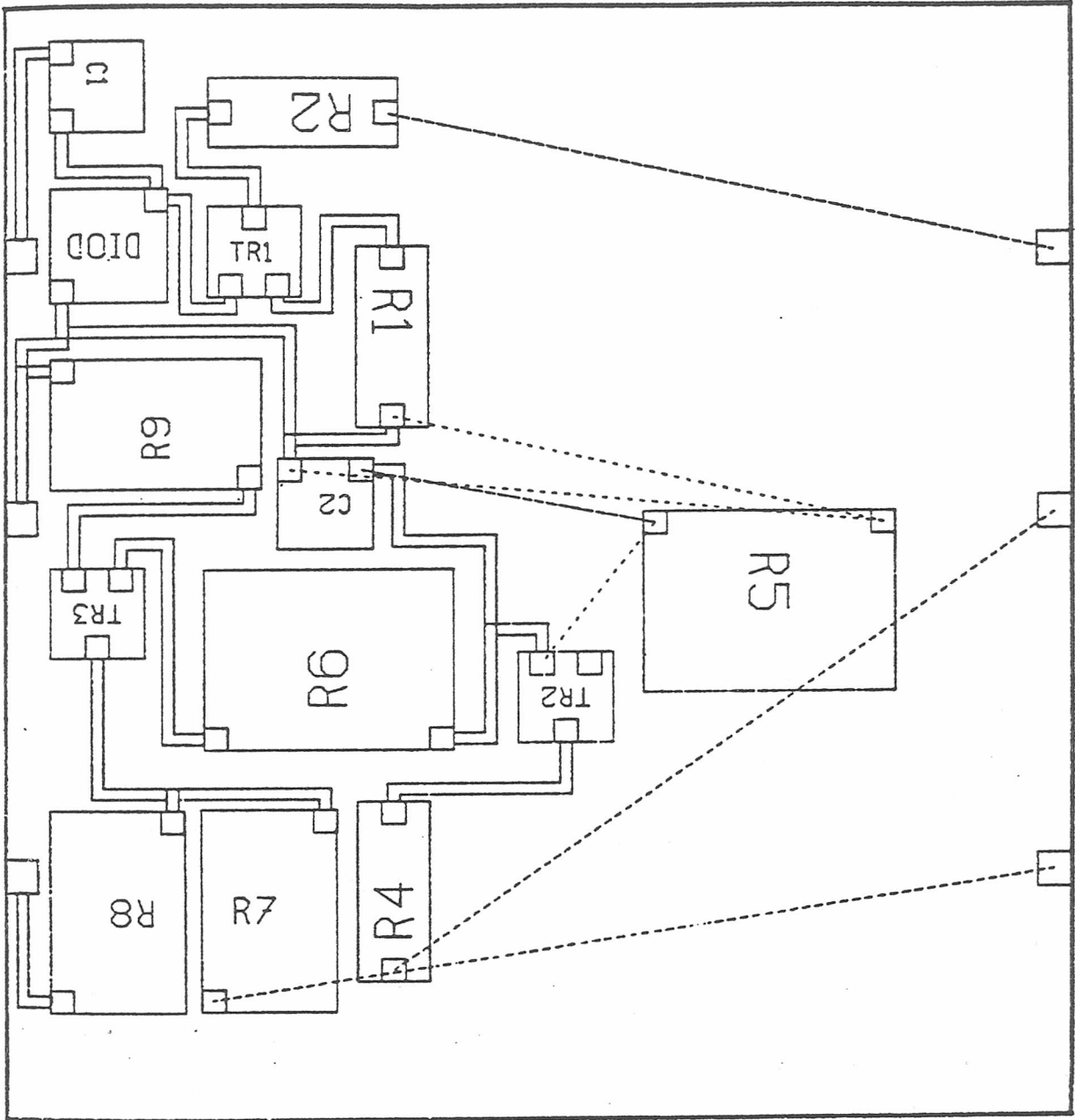
(b) Component R3 re-positioned.

(c) All connections routed.

The circuit is now complete - all components appear on the board and every necessary connection has been made.

Stage 10 (Figure A1.16)

(a) Horizontal strip of wasted space removed from top of board (using "SHRINK" facility).



What Next ?  
\*SPR

Figure A1.13 Stage 7 in "LAYOUT"

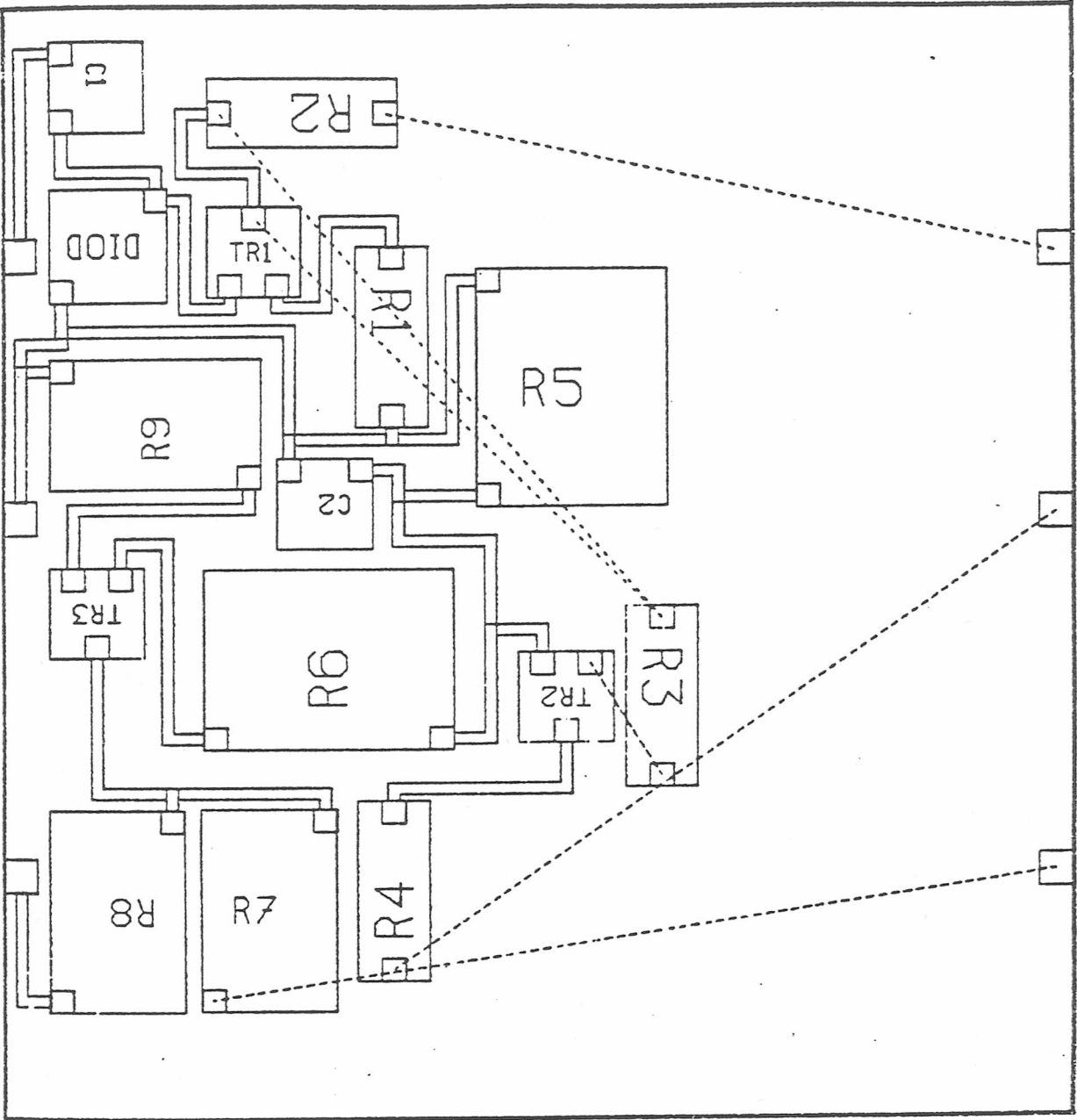
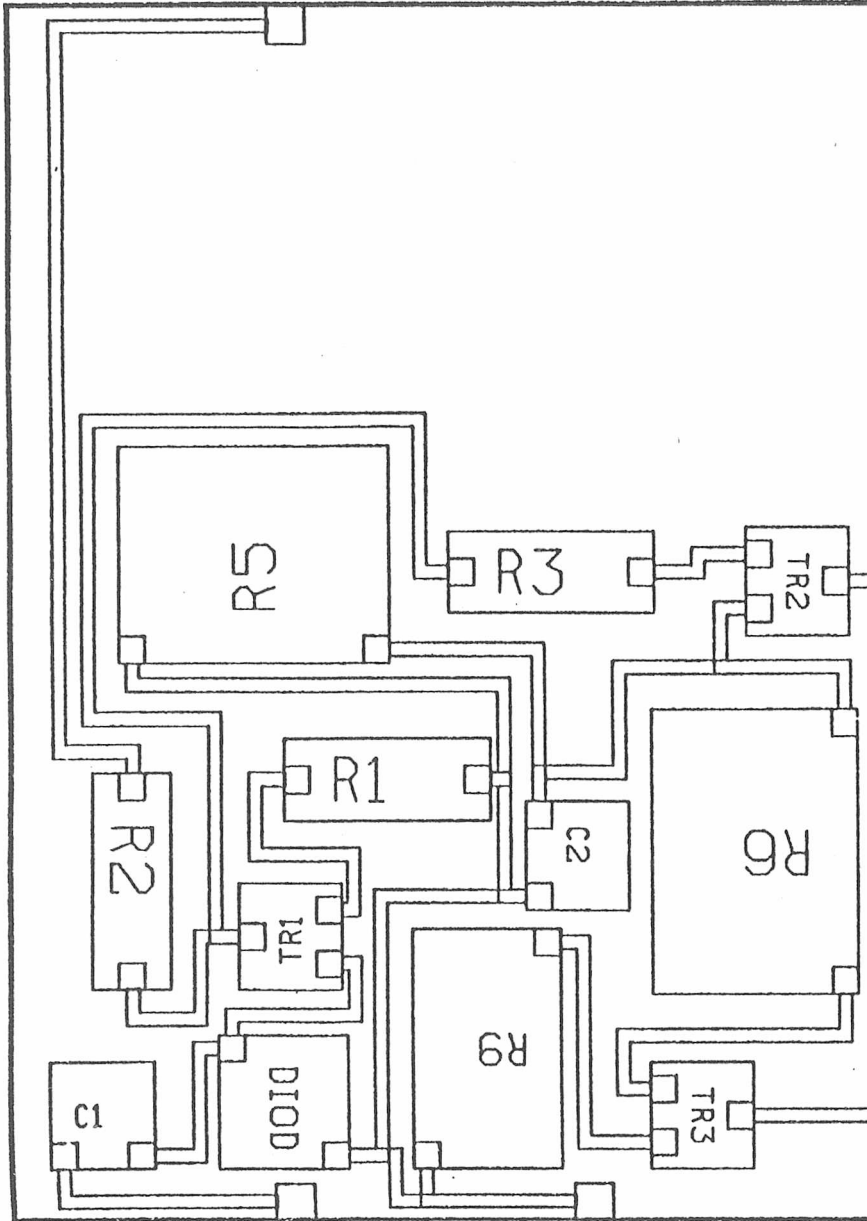


Figure A1-14 Stage 8 in "LAYOUT"

Thin Film Circuit with pre-defined Resistors



What Next ?

‡SPR

CIRCUIT IS COMPLETE

‡

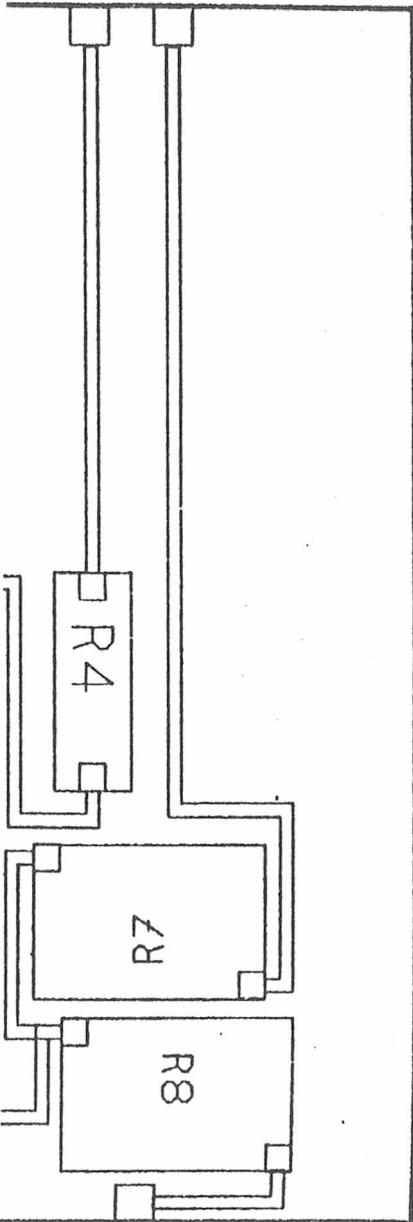
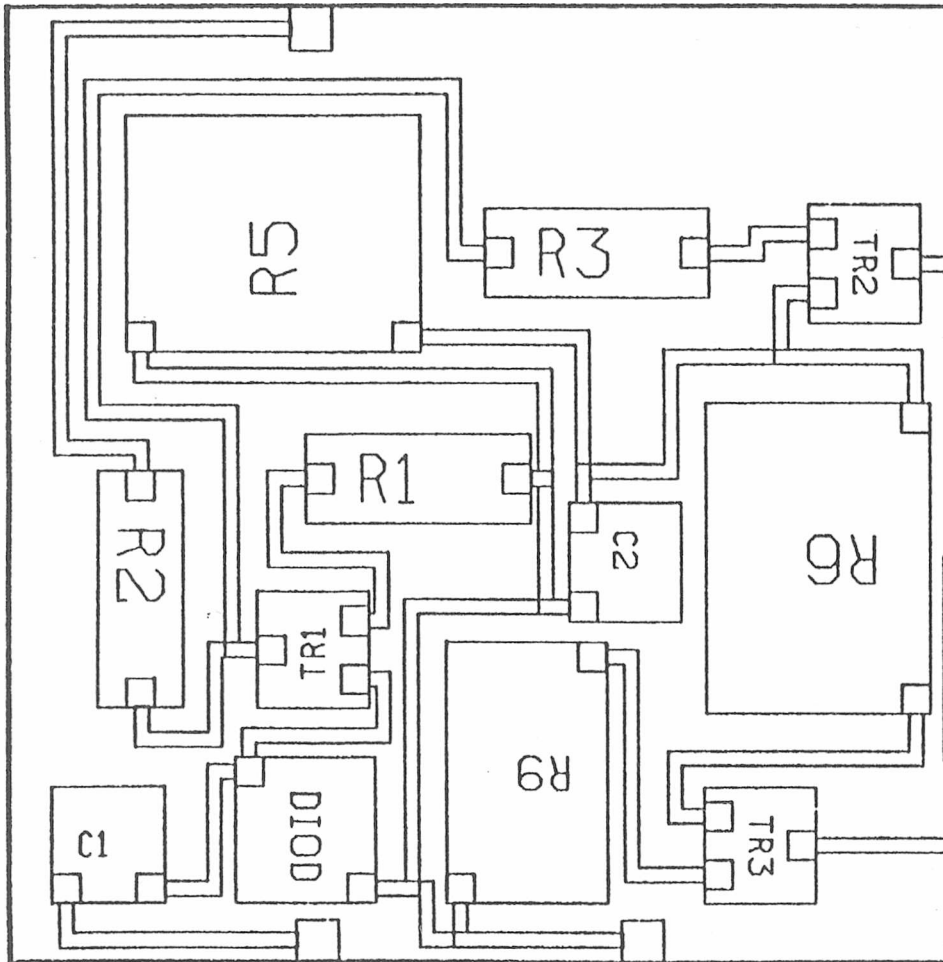


Figure A1.15 Stage 9 in LAYOUT

Thin Film Circuit with pre-defined Resistors



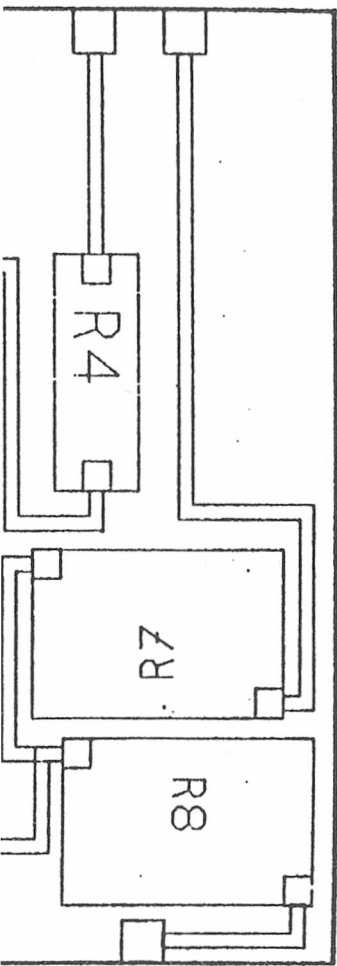


Figure A1.16 Stage 10 in "LAYOUT"



- (b) Vertical strip of wasted space removed from right hand edge of board, again using "SHRINK" option.

Stage 11 (Figure A1.17)

- (a) Drawing mode changed to show in-slice components("MASK" command).
- (b) Node Numbers restored ("NO NODES" facility).
- (c) Components R7 and R8 rotated to save space.
- (d) Vertical strip of wasted space removed from right hand edge of board.
- (e) R4 repositioned.

Stage 12 (Figure A1.18)

The layout is now complete and can be scaled to fit any size of board or left in its most compressed form.

- (a) Horizontal strip inserted at right hand edge of board ("BOARD" command).
- (b) Vertical strip inserted at top of board.
- (c) Circuit scaled up to fit new board ("EXPAND" facility).
- (d) Any broken connections re-formed.
- (e) Node Numbers suppressed.

The layout file can now be passed to the photoplotting suite for manufacture (Chapter 10).

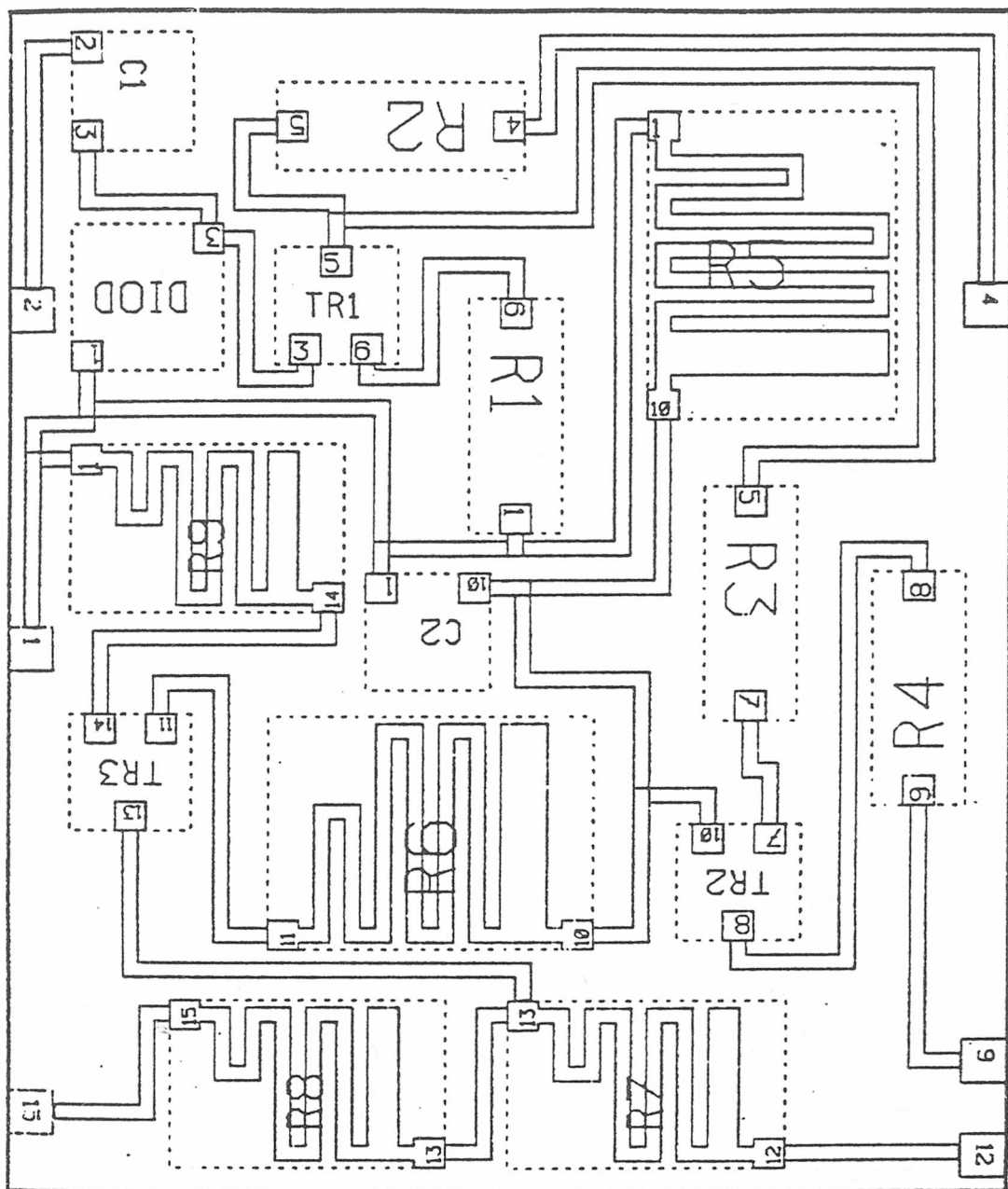


Figure A1-17 Stage 11 in 'LAYOUT'  
(HIGH DENSITY FORM)

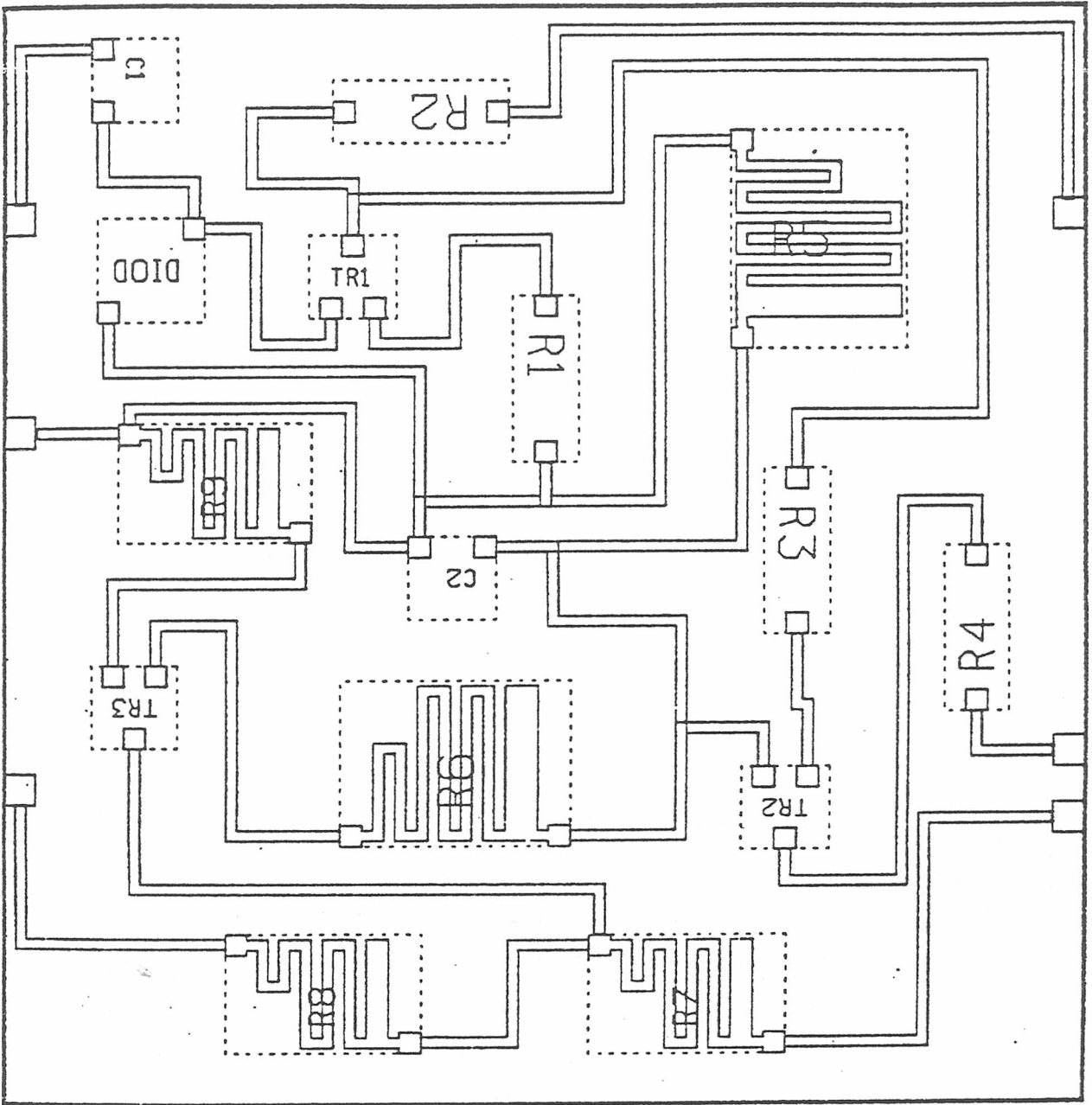


Figure A1-18 Stage 12 in "LAYOUT" (Expanded form)

## Appendix 2

### Results Obtained From Practical Circuits

This appendix illustrates the results which can be obtained when practical circuits are considered. Both of the examples are inherently un-planar, and are therefore difficult to design. The first layout has been achieved by inserting a set of crossovers into the topology file before implementing the programs. In contrast to this, the second circuit has been left un-planar and a set of crossovers have been generated completely automatically.

#### Circuit 1 - A Low Voltage Audio Power Amplifier

The circuit diagram for this example is shown in Figure A2.1. Four crossovers have been included to achieve planarity and these have been named CR1, CR2, CR3 and CR4.

After assigning appropriate Node Numbers and defining the necessary component geometries, the description file shown in Figure A2.2 was compiled. Notice that the crossovers are treated as normal four-terminal devices.

The final layout produced for this example is given in Figure A2.3. A comparable solution would take a considerable time to achieve using a manual layout approach. It would then, in any case, have to be transcribed into computer memory for fabrication.

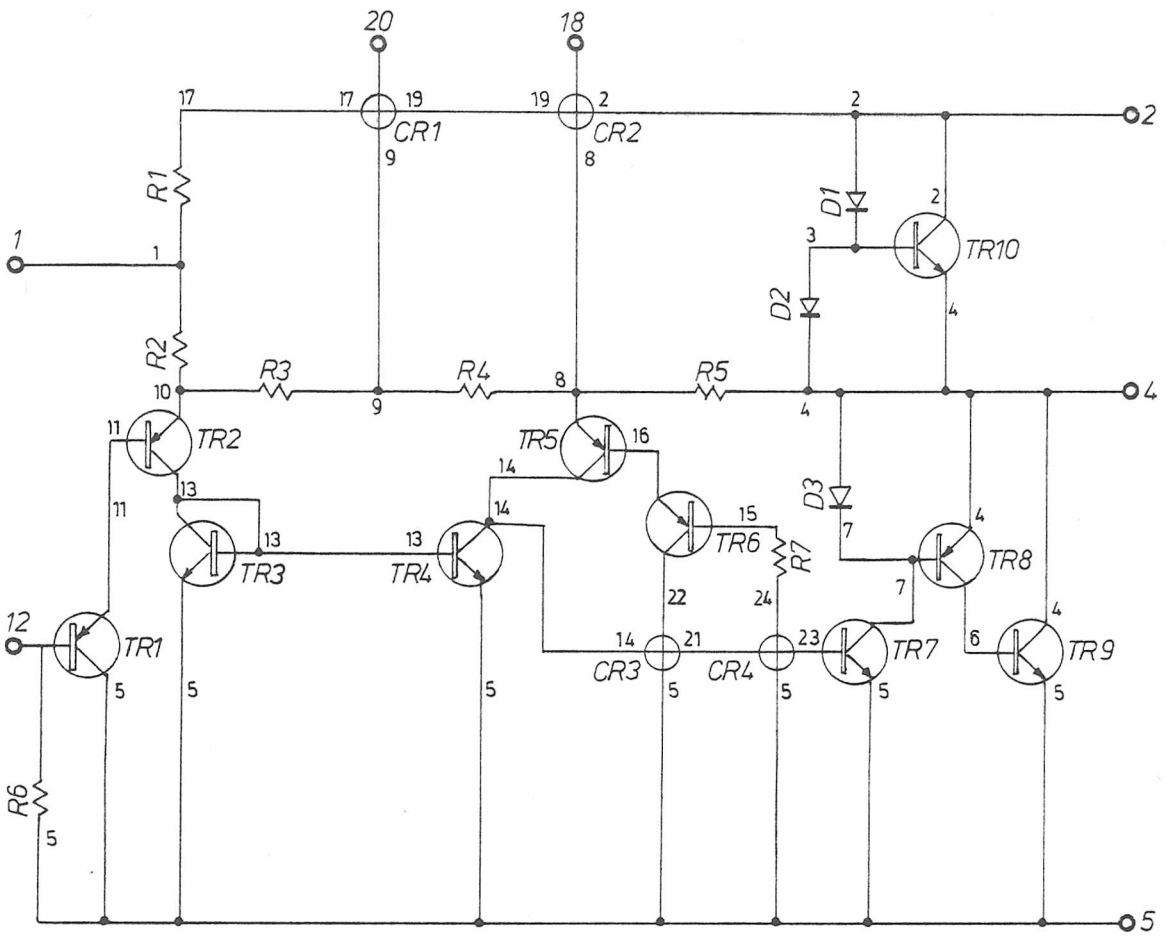
#### Circuit 2 - Voltage Regulator

This network (shown in Figure A2.4) is again inherently un-planar. This time the crossovers were inserted automatically - a total of eight were needed. When the auto-insertion algorithm creates a crossover device, it names it "Xn" (where n is the total number of crossovers used to date). The set X1 to X8 have been generated in this example.

The definition file given in Figure A2.5 contains a "Master Crossover" description which defines the bonding pad size involved.

Figure A2.6 shows the layout produced for this example. This is more complicated than circuit 1 but has achieved a high packing density.

A pencil and paper approach is clearly a poor substitute in this case.



ref.- Linear Data Book National (1976)  
 circuit LM386  
 low voltage audio power amplifier  
 page 10 - 55

Figure A2.1 Circuit example 1

TY AMP.CIR  
LOW VOLTAGE AUDIO POWER AMPLIFIER

DIODE 2

D2,3,4

D3,4,7

MASTER RES 1,2

MASTER RES 2,2

MASTER RES 3,2

MASTER RES 4,2

MASTER TRAN 1,3,400,200

200,175

25,25

375,25

CROSS

CR1,9,19,20,17

CR2,8,2,18,19

CR3,14,5,21,22

CR4,21,5,23,24

MASTER TRAN 2,3,600,500

300,460

40,40

560,40

EDGE

12,5,4,2,18,20,1

STOP

MASTER DIODE 1,2,800,200

50,100

750,100

MASTER DIODE 2,2,600,160

40,80

560,80

MASTER CROSS, 100

STOP

RES 1

R1,17,1

R2,1,10

RES 2

R3,10,9

R5,8,4

RES 3

R4,9,8

R6,12,5

RES 4

R7,15,24

TRAN 1

TR1,12,5,11

TR2,11,13,10

TR5,16,8,14

TR6,15,16,22

TRAN 2

TR3,13,13,5

TR4,13,5,14

TR7,23,5,7

TR9,6,5,4

TR10,3,4,2

DIODE 1

D1,2,3

Figure A2.2 Description file for Circuit 1

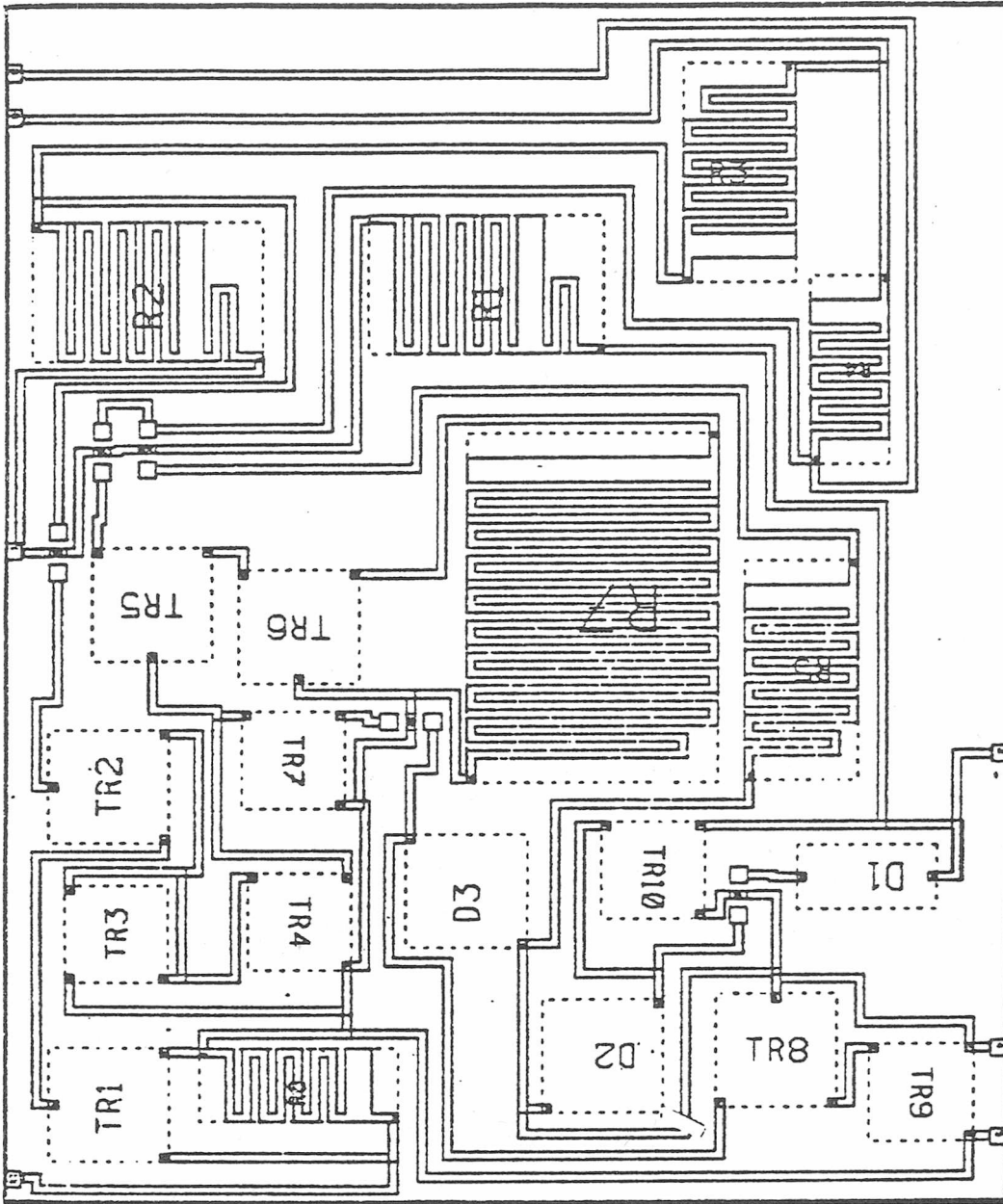
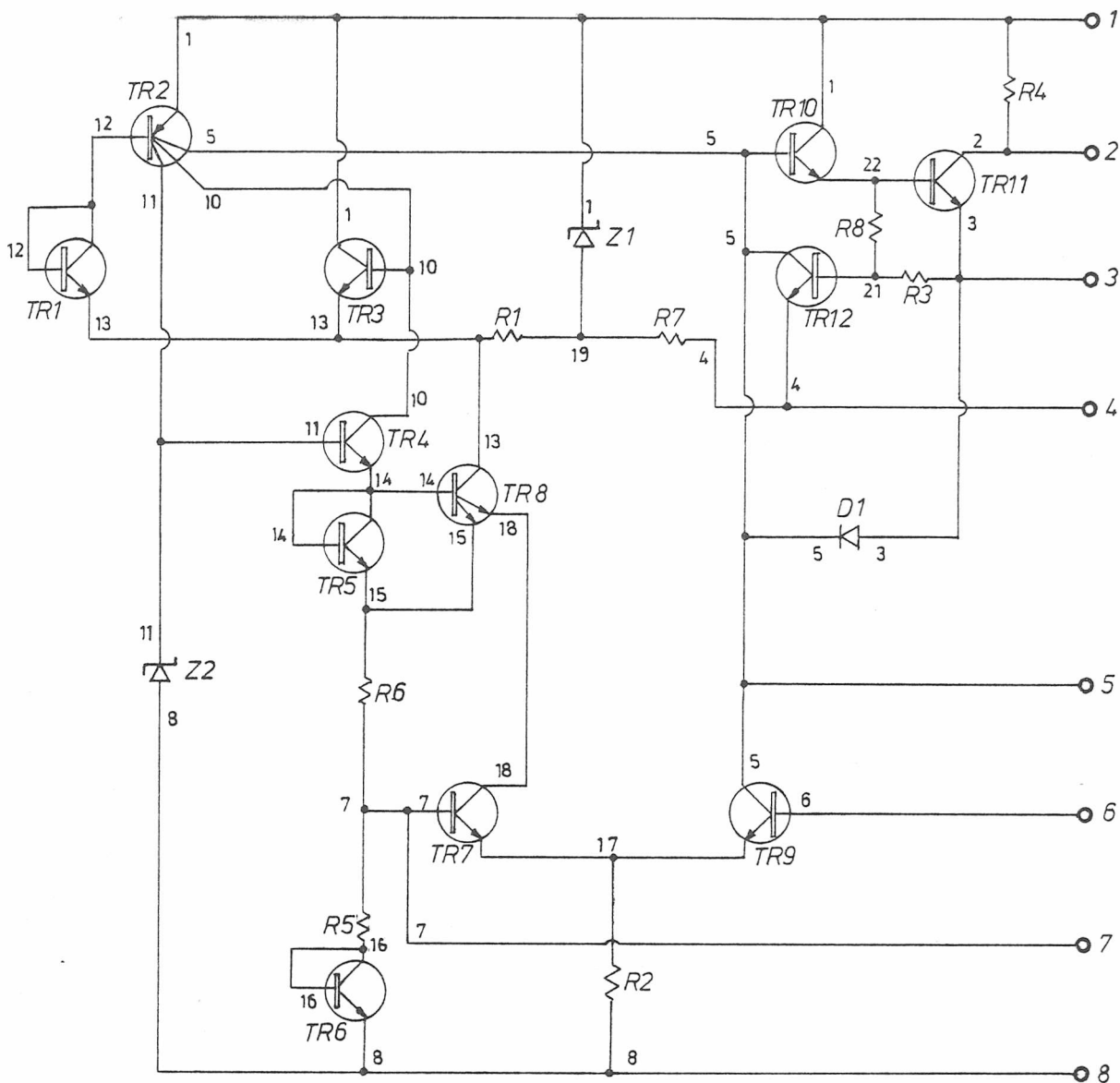


Figure A2-3 Layout produced for Circuit 1

What Next ?  
 SHARD COPY





ref. Linear Data Book National (1976)  
 circuit LM100 voltage regulator  
 page 1-1

Figure A2-4 Circuit example 2

TY C2.CIR  
VOLTAGE REGULATOR SPEC. 87654321

MASTER RES1,2

MASTER RES2,2

MASTER ZEN,2,500,300

25,150  
475,150

MASTER DIOD,2,500,400

25,200  
475,200

MASTER MTRN1,3,700,700

25,25  
675,25  
350,675

MASTER MTRN2,3,600,600

25,25  
575,575  
25,575

MASTER MTRN3,5,800,800

200,25  
400,25  
600,25  
600,775  
200,775

MASTER MTRN4,4,800,800

200,25  
600,25  
600,775  
200,775

MASTER CROSS,100

STOP

MTRN1

TR1,13,12,12  
TR4,14,10,11  
TR5,15,14,14  
TR6,8,16,16  
TR7,17,18,7  
TR10,22,1,5  
TR11,3,2,22

MTRN2

TR3,13,10,1  
TR9,17,6,5  
TR12,4,21,5

MTRN3

TR2,11,10,5,1,12

MTRN4

TR8,15,18,13,14

RES1

R1,19,13  
R2,8,17  
R3,3,21  
R4,1,2

RES2

R5,7,16  
R6,15,7  
R7,4,19  
R8,22,21

ZEN

Z1,19,1  
Z2,8,11

DIOD

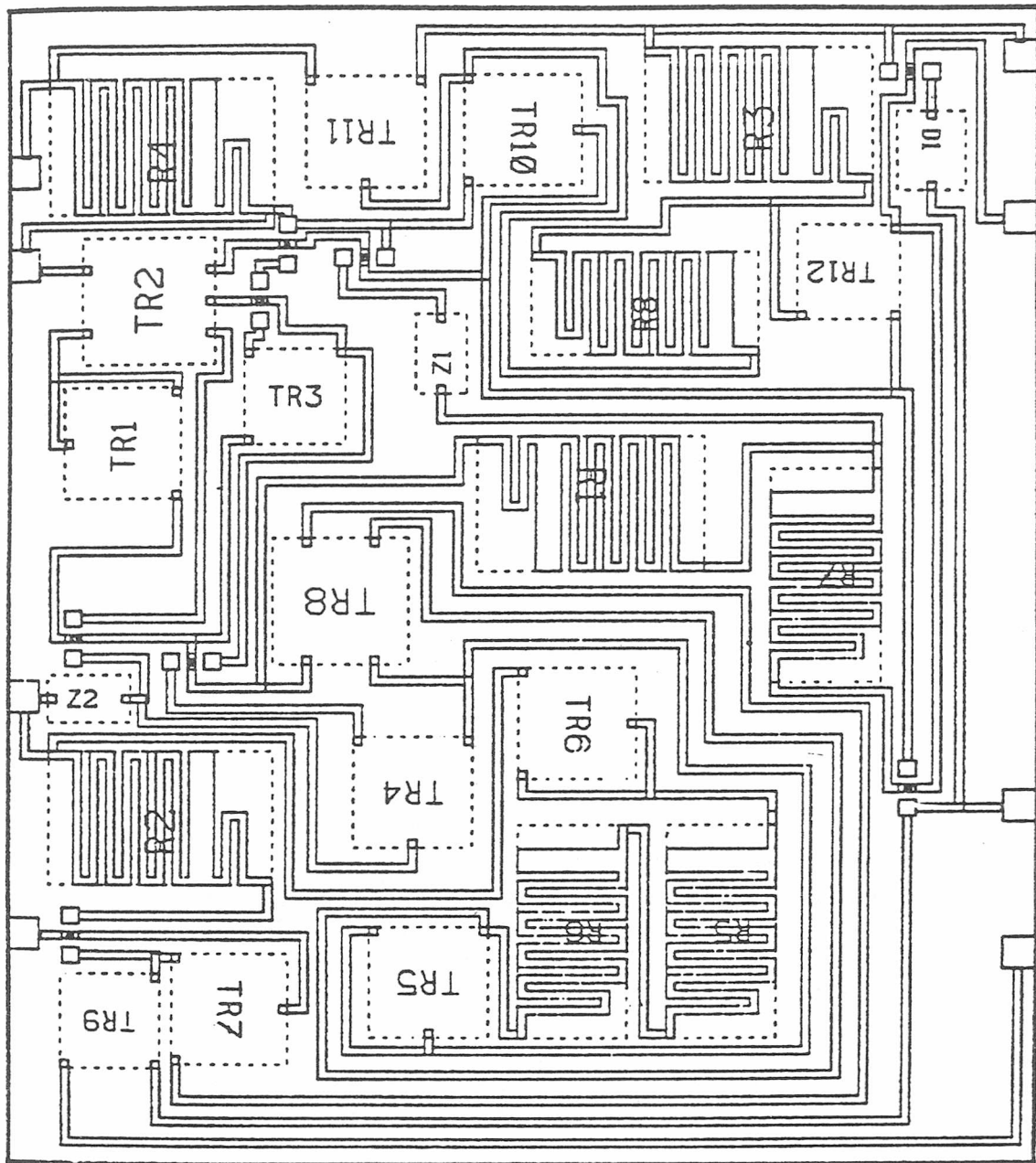
D1,3,5

EDGE

8 7 6 5 4 3 2 1

STOP

Figure A2-5 Description File for Circuit 2



What Next ?

Figure A2-6 Layout produced for Circuit 2

## References

- 1) P. Agrawal and M.A. Breuer  
"Some Theoretical Aspects Of Algorithmic Routing"  
Dept. Electrical Engineering, Research Report,  
University Of Southern California.
- 2) Bell Telephone Laboratories  
"Integrated Device And Connection Technology",  
Chapters 1, 2, 3 and 4.
- 3) F. Capoccacia and A.L. Frisiani,  
"An Algorithm For The Placement Of Large Scale  
Integrated Circuit Elements",  
ALTA FREQUENZA Vol 39 Part 2 1970.
- 4) C. Crum  
"GADOR - Graphic Aided Design Of Resistors",  
Research Report, Edinburgh University, August 1974.
- 5) R.C. Doig,  
"Automatic Routing Of Multi-Layer Printed Circuit Boards",  
Internal Report, GEC Computers Ltd,  
Borehamwood, Herts.
- 6) J.D. Eades, J.R. Jordan, R.G. Kelly and J. Murray,  
"Application of GAELIC To The Design Of A Large Scale  
Integrated Circuit",  
International Conference on C.A.D.,  
I.E.E. Publication 111, 8 - 11th April 1974.
- 7) "Heat Dissipation In Thin Film Circuits",  
Internal Report, Ferranti Ltd, Edinburgh.
- 8) "Programming Formats For Flat Bed Plotters",  
Ferranti Ltd, Plotter Handbook.
- 9) C.J. Fisk, D.L. Caskey and L.E. West,  
"ACCEL : Automated Circuit Card Etching Layout",  
Proc. I.E.E.E., Vol 55, No. 11, pp. 1971 - 1982, November 1967.

- 10) A.J. Fletcher,  
"Computer Programs For The Layout Of Integrated Circuits",  
Ph.D. Thesis, May 1970 (UMIST).
- 11) D.W. Hightower,  
"A Solution To Line Routing Problems On The Continuous Plane",  
Proc of the 6th Annual Share ACM.,  
I.E.E. Design Automation Workshop, June 1969.
- 12) D.J. Kinniment and L.J. Weston,  
"An Evaluation Of Conventional And Backtracking Algorithms  
For Routing Printed Circuit Boards",  
Proc. International Conference CADMECCS,  
I.E.E. pub 175, July 1979.
- 13) C.Y. Lee,  
"An Algorithm For Path Connections And Its Applications",  
IRE Trans. on Electronic Computers, vol. 10,  
no. 3, pp 346-365, September 1961.
- 14) D.F.A. Leever, s,  
"Automated Printed Circuit Board Design Using A Graphical Display",  
Proc. IEE Conference on Computer Science and Technology, June 1969.
- 15) J.S. Mamelak,  
"The Placement Of Computer Logic Modules",  
Journal of the Assoc. of Computing Machinery,  
Vol. 13, No. 4, October 1966.
- 16) C.E. Nugent et al,  
"An Experimental Comparison Of Techniques for The Assignment  
Of Facilities To Locations", Operations Research Journal, 1966.
- 17) K. Patel,  
"Computer - Aided Decomposition Of Geometric Contours  
Into Standardised Areas", Computer Aided Design Magazine,  
Vol. 9., No. 3, July 1977.
- 18) P.E. Radley,  
"The Automatic Layout Of Electronic Circuits",  
Ph.D. Thesis, University of Cambridge, August 1972.

- 19) N.A. Rose,  
"Computer Aided Design Of Printed Wiring Boards",  
Ph.D. Thesis, University of Edinburgh,  
April 1970.
- 20) A. Rowles and J. Toome,  
"Allocation, Placement And Routing By Computer",  
Microelectronics in Equipment Conference, November 1966.
- 21) R.A. Rutman,  
"An Algorithm For Placement Of Interconnected Elements  
Based On Minimum Wire Length",  
Proc. Spring Joint Computer Conference, 1964.
- 22) L. Steinberg,  
"The Backboard Wiring Problem; A Placement Algorithm",  
SIAM Review, Vol. 3, No. 1, January 1961.
- 23) J.D. Tait,  
"Design And Fabrication Of A Thin Film Microcircuit",  
Honours Thesis,  
Robert Gordons Institute Of Technology,  
Aberdeen.

## Bibliography

- 1) M. Adamovicz and A. Albano,  
"Nesting Two - Dimensional Shapes In Rectangular Modules",  
Computer Aided Design magazine Vol. 8, No. 1 January 1976.
- 2) E.M.P. Amitirigala and G. Rzevski,  
"On The Computer-Aided Assessment Of Overheating Of  
Electronic Systems",  
International Conference On C.A.D.,  
I.E.E. Publication No. 175,  
3rd - 6th July 1979.
- 3) J. Atiyah,  
"An Interactive Graphics Based Schematic Drawing System",  
International Conference on C.A.D.,  
I.E.E. Publication No. 111,  
8th - 11th April 1974.
- 4) R. Beatty,  
"Handbook Of Materials And Processes For Electronics"  
(Chapter 11 - Thin Filas).
- 5) D.L. Broster,  
"CELLMOSS - An Automated Custom LSI Design Technique",  
International Conference On C.A.D.,  
I.E.E. Publication No. 175, 3rd - 6th July 1979.
- 6) R.C. Doig,  
"Computer Aids To The Design Of Thin Film Circuits",  
Interim Report, Robert Gordons Institute Of Technology,  
Scotland. September 1977.
- 7) R.C. Doig,  
"The Fabrication Of Masks For Thin Film Circuits Using A  
Ferranti Masterplotter With Slit Shaped Photographic Apertures",  
Internal Report, Robert Gordons Institute Of Technology,  
Scotland. February 1978.
- 8) R.C. Doig,  
"Computer Aided Design Of Thin Film Circuits",  
International Conference On C.A.D.,  
I.E.E. Publication No. 175, 3rd - 6th July 1979.

- 9) A.J. Fletcher,  
"EUREKA - A System For Laying Out Circuits Using A Single Layer Of Connections",  
International Conference on C.A.D.,  
I.E.E. Publication No. 86, 24th - 26th April 1972.
- 10) A.J. Fletcher,  
"EUREKA - A System For The Automatic Layout Of Single - Sided Printed Circuit Boards", International Conference on C.A.D.,  
I.E.E. Publication No. 111, 8th - 11th April 1974.
- 11) W.E. Hillier,  
"Practical Multilayer Printed Circuit Board Layout Using Interactive Graphics",  
International Conference on C.A.D.,  
I.E.E. Publication No. 111, 8th - 11th April 1974.
- 12) K.H. Hosking and P.G. Moulton,  
"A Program Suite For The Layout Of Double-Sided Printed Circuit Boards Using A Very Fast Routing Algorithm",  
International Conference on C.A.D.,  
I.E.E. Publication No. 111, 8th - 11th April 1974.
- 13) R.C. Levine,  
"Computer-Aided Design Of Thin Film Amplifiers, Filters And Distributed RC Networks"  
inc "Computer - Aided Integrated Circuit Design"  
Editor G.J. Herskowitz, McGraw - Hill Books.
- 14) J. Mittleman,  
"Computer - Aided Design : Part 11  
Short Cuts To IC's And P - C Boards",  
"Electronics" magazine 4th September 1967.
- 15) K.G. Patterson,  
"Tracking Performance Parameters Of Printed Circuit Boards With Regular Layout",  
International Conference on C.A.D.,  
I.E.E. Publication No. 111, 8 - 11th April 1974.
- 16) S. Pimont,  
"New Algorithms For Grid - Less Routing Of High Density Printed Circuit Boards", International Conference on C.A.D.,  
I.E.E. Publication No. 175, 3 - 6th July 1979.



- 17) L.S. Pugh,  
"An Improvement In Printed Circuit Board Routability  
Using A Maze - Running Algorithm", Electronics Letters  
Vol. 14, No. 1, 5th January 1978.
- 18) P.E. Radley,  
"The Automatic Layout Of Electronic Circuits By Computer",  
International Conference On C.A.D.,  
I.E.E. Publication No. 86, 24 - 26th April 1972.
- 19) R.C. Sloan,  
"Computer Aided Layout Of Thick Film Hybrid Circuits",  
International Conference on C.A.D.,  
I.E.E. Publication No. 111, 8 - 11th April 1974.
- 20) T.F. Smith and B.J. Woods,  
"Poligon - An Interactive Graphics Design Tool",  
International Conference on C.A.D.,  
I.E.E. Publication No. 175, 3rd - 6th July 1979.
- 21) A Spitalny and M. Goldberg,  
"Drawing Board For I.C.'s",  
"Electronics" magazine, 4th September 1967.
- 22) N. Sugiyama and K. Saitoh,  
"Electron Beam Exposure System AMDES",  
"Computer Aided Design" magazine,  
Vol. 11, No. 2, March 1979.
- 23) N. Sugiyama, H. Kawanishi, T. Ohtsuki and H. Watanabe,  
"An Integrated Circuit Layout Design System",  
"Computer Aided Design" magazine,  
Vol. 6, No. 2, April 1974.
- 24) N. Sugiyama and M. Hirano,  
"An Automated Circuit Layout Design Program Of A Systematic  
Chip Using Building Blocks",  
International Conference on C.A.D.,  
I.E.E. Publication No. 86, 24 - 26th April 1972.
- 25) "High - Speed Laser Trimming Of Hybrid Circuits",  
"New Electronics" magazine,  
November 28th 1972.