A DIAGRAMMATIC REASONING APPROACH TO AUTOMATIC WIRE ROUTING

by

Arthur Lo

B.Sc., Simon Fraser University, 1982

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

Computing Science

SIMON FRASER UNIVERSITY

April, 1984

APPROVAL

Name: Arthur Lo

Degree: Master of Science

Title of thesis: A Diagrammatic Reasoning Approach to Automatic Wire Routing

Examining Committee:

Chairperson: Dr. Arthur L. Liestman

Dr. Brian V. Funt
Senior Supervisor

Dr. Nick J. Cercone

Dr. Richard F. Hobson

Dr. Gordon McCalla
External Examiner

Date Approved: April 19, 1984

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

A DIAGRAMMATIC REASONING APPROACH TO AUTOMATIC WIRE ROUTING

Author: _____
(signature)

ARTHUR LO
(name)

Feb 12, 1985
(date)

Abstract

Diagrams have been known to be an important tool in problem-solving. The usefulness of diagrams comes from the fact that humans can extract information contained in a diagram, make changes to the diagram if necessary, and interpret the results. ROUTER is a problem-solving system implementation that uses a "see-and-compute" approach to solve routing problems that arise in hierarchical integrated circuit layout design. ROUTER's components include a high level reasoner which knows about the routing constraints, a simulated parallel processing "retina" to "look at" its diagram, and a diagram which can be modified by the high level reasoner. Detailed algorithms and estimation on algorithm complexity are presented. The results of simulation runs seem to suggest that diagrammatic representation of information can be effectively utilized by a computer problem-solving system.

Acknowledgements

I wish to express my many thanks to my thesis supervisor Brian Funt for his invaluable guidance of my research and his precious time spent in proofreading the thesis drafts. I also wish to express my gratitude to Austine Chan, Stanley Wong and Ed Bryant for their spiritual support during the many hours I spent in the terminal room.

Table of Contents

List of Figures

1. Introduction

Diagrams have been known to be invaluable tools for many intellectual activities. Computer scientists use flow charts and hierarchical module diagrams for program documentation. Architects use diagrams extensively in all phases of a design. Students use pencil and paper in solving mathematical problems. The usefulness of diagrams comes from the fact that humans can extract information contained in a diagram, make changes to the diagram if necessary, and interpret the results. In other words, a diagram can be viewed as a simulation which is cheap to perform on a piece of paper. We can also say that diagrams are "idealizations" of complex real world situations. But can a computer use diagrams effectively to solve problems like humans can? If yes, then what kind of special equipment is needed for such a computer? This thesis will give some insight to these questions.

1.1 Research Motivation

A problem that is non-trivial and can be presented naturally in the form of a diagram is needed and has been identified for this research project. This is the wire routing problem in hierarchical integrated circuit (IC) layout design. This problem is to electrically connect pins that belong to some rectangular blocks (the logical sub-circuitry of a complex system) according to a connection

list specified by a circuit designer subject to certain design constraints. For a human designer, the routing problem is a very tedious task. Fortunately, automatic wire routers such as the Lee's (Lee (1961)) router have been developed to help the designer solve the routing problem. (Other types of routers will be discussed in the next chapter.) However, due to the sequential nature of Lee's algorithm, it is very likely to get stuck and it often needs help from the designer to guide it around obstructions. The designer usually has some tricks or intuition on how to move wires around to avoid blockages. It is the intuition that is often hard to capture and then incorporate into computer programs.

The use of diagram and intuition by a computer system merge into a good Artificial Intelligence (AI) project. Some previous research work on the use of diagram and the application of AI techniques to computer aided design (CAD) systems are worth mentioning and will be presented in the next section.

1.2 Previous Research

Gelernter (1963) made use of diagrams in his classic geometry-theorem proving machine. The main use of diagrams is reflected in one of his heuristics "Reject as false any statement that is not valid in the diagram." By using diagrams to reject false hypotheses, the search space can be

considerably limited.

Sussman and Stallman (1975) developed a circuit analysis program called EL which employed heuristic "inspection" methods to solve rather complex dc bias circuits. The heuristics "give EL the ability to explain any result in terms of its own qualitative reasoning processes. EL's reasoning is based on the concept of a 'local one-step deduction' augmented by various 'teleological' principles and by the concept of a 'macro-element'." Later Sussman and Stallman (1977) further extended their project. They designed and implemented a problem-solving language called Antecedent Reasoning System (ARS). They also developed two methods in their computer-aided circuit analysis system. "One is a method of electrical network analysis [they] call analysis by propagation of constraints. The other is the technique of efficient combinatorial search by dependency-directed backtracking." Sussman (1977) himself outlined the problem of intelligent failure recovery in a problem-solver for electrical design. His solver was able to learn from the mistakes it made. Sussman's idea was that "many bugs are just manifestations of powerful startegies of creative thinking-- that creation and removal of bugs are necessary steps in the normal process of solving a complex problem."

Sandini and Tagliasco (1980) proposed a retina-like structure for scene analysis. Their aim was "to demonstrate that intrinsic economy of a human retina-like structure, in

terms of an optimum compromise among large visual field, acceptable resolution, and information reduction, in the scene analysis of man-oriented environments." They suggested that "the technology of solid-state video cameras and appropiate scanning procedures could allow the actual implementation of the proposed retina, especially in the field of industrial scene analysis."

Chaikin (1981) suggested visual modes of processing by computers (he called them drawing machine) should be closely investigated. His "Drawing Machines" are "devices which utilize the spatial organization of information in a concrete, non-symbolic manner much like seeing or drawing." He described and proposed several kinds of drawing machines such as conformal mapping devices, magnetic bubble arrays, and a massively parallel processor.

1.3 An extension to WHISPER

WHISPER, a problem-solving program which explored the role of diagrams in a computer, was implemented by Funt (1976, 1980). His system consisted of a High Level Reasoner (HLR) which had a limited knowledge of qualitative physics, and a simulated parellel processing "retina" to "observe" a diagram. The diagram in the system was simulated by a two dimensional array which could be modified by the HLR. The problem WHISPER was designed to solve was a stability problem-- given a diagram of a blocks world structure in two

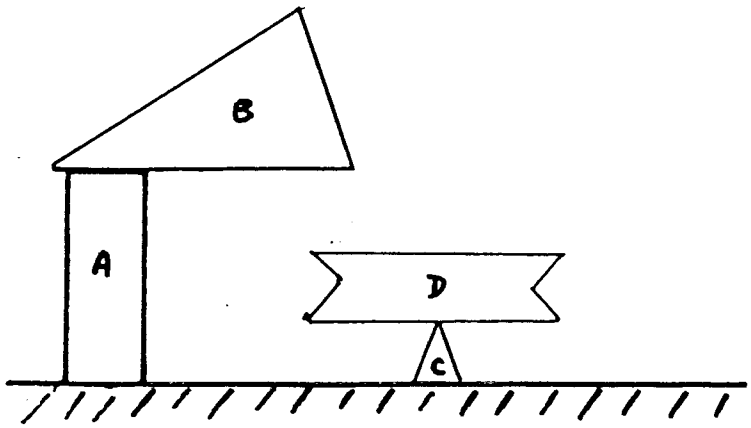


Figure 1

A typical problem WHISPER can solve. (Adapted from Funt (1980))

dimensions, WHISPER detected instabilities and generated a sequence of diagrams representing how the structure would collaspe (Figure 1).

WHISPER can form the basis for a problem-solving computer model which makes use of diagrammatic reasoning. Therefore, it makes sense to further investigate WHISPER's model on a different problem domain outside of the blocks world. In fact, the system (ROUTER) I have implemented is an extension to the above work. It uses WHISPER's techniques to solve automatic wire routing problems.

This thesis is organized as follows: First, the problem domain is formally described with a justification of its choice. Then an overview of the structure and organization

of ROUTER will be presented. Finally, a discussion of the simulation of the system with a detailed presentation of algorithms and results will follow.

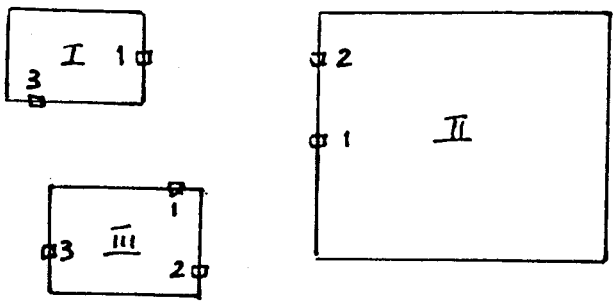
2. Problem Domain: Automatic Wire Routing

The problem to be solved by ROUTER is a simplified version of the routing problem in hierarchical integrated circuit (IC) layouts. This problem is suitable for ROUTER because a diagram is one of the main tools used in solving such a problem. This chapter will start with a definition of the automatic wire routing problem. Then the commom methods of solving the problem will be reviewed. Following that will be a discussion of the choice of the problem. Finally, a simplified version of the routing problem that ROUTER faces will be described.

2.1 Definition of Automatic Wire Routing

Akers (1972) characterizes a wire routing problem as follows: "Given an interconnection diagram (or the information therefrom), and a circuit IC on which the elements in the diagram have been previously placed, lay out the necessary conductor paths on the IC to achieve the indicated electrical connections subject to the imposed constraints" (see Figure 2.) Obviously, the more constraints, the harder the problem. No conductor can have a width more than 25 mils; conductors cannot cross each other; feedthroughs(vias) and conductor turns should be minimized; a specified minimum spacing must be maintained between conductors-- all these are good examples of the various constraints that make the problem interestingly

Components Layout



Connection List

((I-1 II-1 III-1)
(I-2 II-2)
(I-3 III-3))

Figure 2

Example of a routing problem.

difficult. (Akers 1972)

According to Preas and vanCleemput (1979), hierarchical IC layout is the process of hierarchically decomposing the system function of highly complex chips into tractable module sizes. "The starting point for a layout is a set of components, represented by rectangular blocks of arbitrary size and shape and their interconnections. Their external connections or pins are at fixed positions around the peripheries of the blocks." The problem is to place and interconnect the blocks "in such a way that total area of the IC is minimal, subject to the constraint that the routing must be 100% completed."

2.2 Common Approaches in Wire Routing

Two common approaches are prevalent in attacking the routing problem-- sequential and global. Each of them has advantages and disadvantages. A sequential router uses Lee's algorithm (Lee 1961) to connect one path at a time. According to Heinisch (1981), simplicity and versatility are its advantages since various design rules can easily be accommodated. As pointed out by Dees et al. (1981), however, the totally serial nature of Lee's approach usually causes a larger number of routing failures as the problem complexity increases. Nevertheless, continuing research (e. g. "A minimum-impact routing algorithm" by Supowit (1982)) has shown that there are ways to improve the performance of

a sequential router.

The second approach, a global or channel router, was first introduced by Hashimoto and Stevens (1971). According to Heinisch (1981), the channel router starts with the definition of a channel partition by splitting every layer into parallel channels and then goes through a series of processes-- channel assignment, channel routing and channel coupling. The major advantage of the channel approach is that "it splits the initial problem [into] a hierarchy of smaller size ones, which themselves are often reducible to combinatorial problems over a small set of data." As a result, higher routing completion may be achieved. In terms of drawbacks, the channel approach usually involves "more complex algorithms than sequential routing" and they "are more sensitive to device features." (Heinisch (1981))

2.3 Justification of the Problem Domain

Wire routing was chosen as the problem domain because (1) routing is a real world problem (unlike the blocks world situation in WHISPER) complicated enough to gain much attention and continuing research effort; and (2) diagrams are one of the important tools available to human in solving routing problems.

Many papers have been written on the subject of wire routing over more than two decades. One of the subproblems in wire routing is to break down an interconnection list

into a wire list which defines single-pin to single-pin connections. As noted by Akers (1972), depending on the constraints and the way wire list is generated, this subproblem is equivalent to well-known problems such as the Travalling Salesman Problem and the Rectilinear Steiner Tree Problem which are NP-complete. (Garey and Johnson 1977, 1979). This gives us some feeling about the complexity of the wire routing problem.

A graphics package is an essential component in a circuit design system. It allows a circuit designer to enter a design through the monitor screen into the system. Without being able to see his design (say in the form of schematics), it will be very difficult for the designer to analyse his circuit. In this regard, the monitor screen serves as a diagram which is a vital tool in the design process.

2.4 A Simplified Version of The Routing Problem

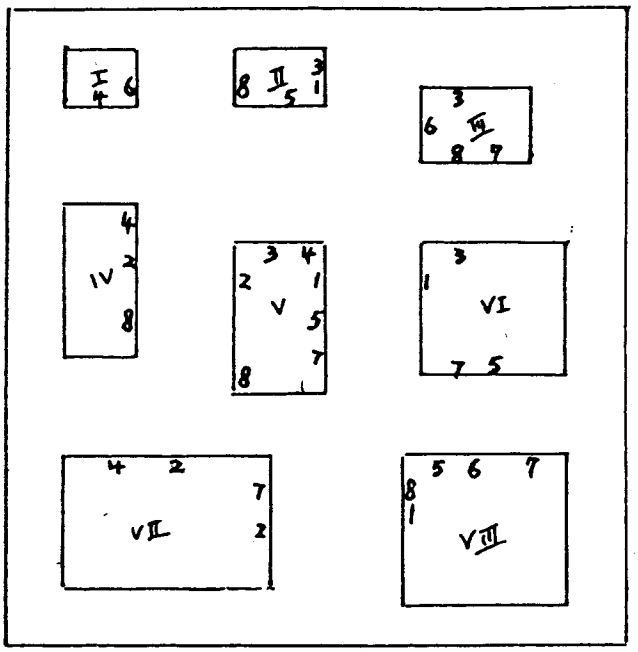
Since ROUTER depends on a software simulation of parallel processing, the routing IC is limited to a two-layer 33 x 33 grid IC in order that the execution times be acceptable. Rectangular blocks of arbitrary size and shape with pins around their peripheries are assumed to be placed by a person, or by some placement algorithms such as the one by Preas and vanCleemput (1979).

The simplified routing problems that ROUTER will attack are subject to the following constraints:

1. Conductors cannot cross one another.
2. Total conductor length should be minimized.
3. Vias should be minimized.
4. There should be at least unit spacing between conductors.
5. No conductor can have a width more than one unit.

In reality, the width of a via is usually larger than the width of a conductor. However, for simplicity, they are assumed to be the same. The constraint that routing must be 100% completed is relaxed here since some essential rip-up and reroute techniques have not be incorporated into the project. (Dees and Karger (1982) have written an excellent survey on rip-up and reroute techniques.) Constraints 4 and 5 can be dealt with quite easily by restricting wires to run on the grid lines. Figure 3 shows a typical kind of problem ROUTER is to solve.

(a) A problem



(b) A solution

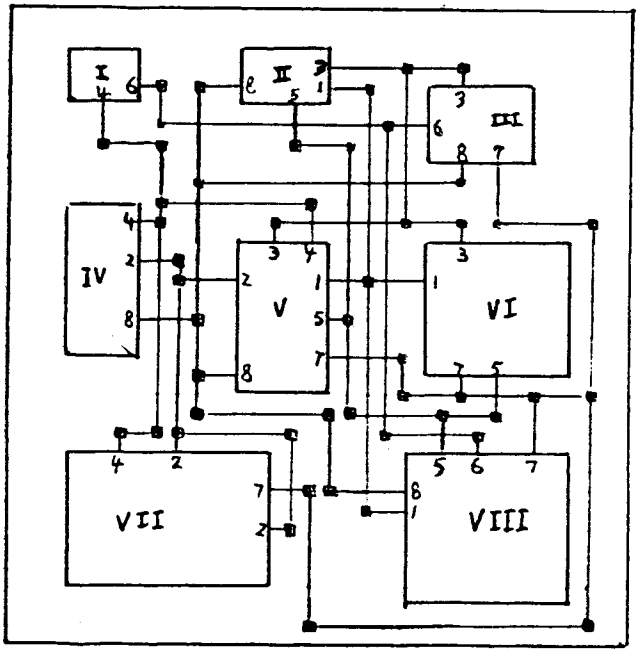


Figure 3

A typical problem ROUTER is to solve. (a) A problem. (b) A solution.

3. Structure and Organization of ROUTER

In this chapter, the structure and organization of ROUTER will be detailed. An overview of the system structure and its components will be described first. Then a detailed description of each component will be presented.

3.1 Overview

Figure 4 shows an overview of the structure of ROUTER. It consists of three major components: the high level reasoner, the parallel processing retina, and the diagram. The high level reasoner is a traditional problem-solving program with the exception that it can direct a retina to "observe" a diagram and make changes to the diagram when necessary. The retina is responsible for extracting information from the diagram on which the wire routing problem will be represented.

3.2 The High Level Reasoner

The high-level reasoner (HLR) is the major problem-solving component which has knowledge of how to make connections and can accommodate the constraints imposed on the routing problem. The HLR consists of many specialists which can interpret features extracted from a diagram through the retina. Each specialist concerns a certain feature such as "whether a track is alright to be used" or "whether a wire will definitely block other unrouted nets."

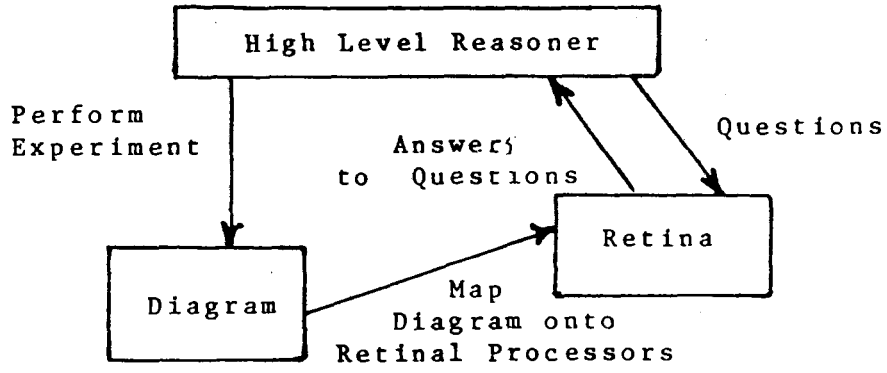


Figure 4

An overview of the structure of ROUTER. (Adapted from Funt (1980).)

Note that the HLR is a domain dependent system. So each new problem domain will need a new HLR.

3.2.1 Redrawing Mechanism

Pencil, paper and eraser are readily available for people in their daily problem-solving activities. Likewise, the redrawing mechanism is available to the HLR in its problem-solving. The redrawing mechanism can make changes to the diagram under the control of the HLR. This allows ROUTER to perform experiments in an organized trial-and-error basis.

3.3 The Retina

ROUTER's retina is roughly modelled after the human eye. It is a collection of parallel processors simulated by software. Each identical processor has its own independent memory and input device called a receptor. ROUTER has two analogous properties to the human eye:

1. The retina can move freely over a diagram and is able to fixate at any particular location.
2. The retina has declining resolution from its centre to its periphery.

Without fixation, ROUTER would not be able to examine a diagram in detail due to the declining resolution. Figure 5 shows the geometrical arrangement of the retinal receptors. The central area is called the "fovea" and has the highest

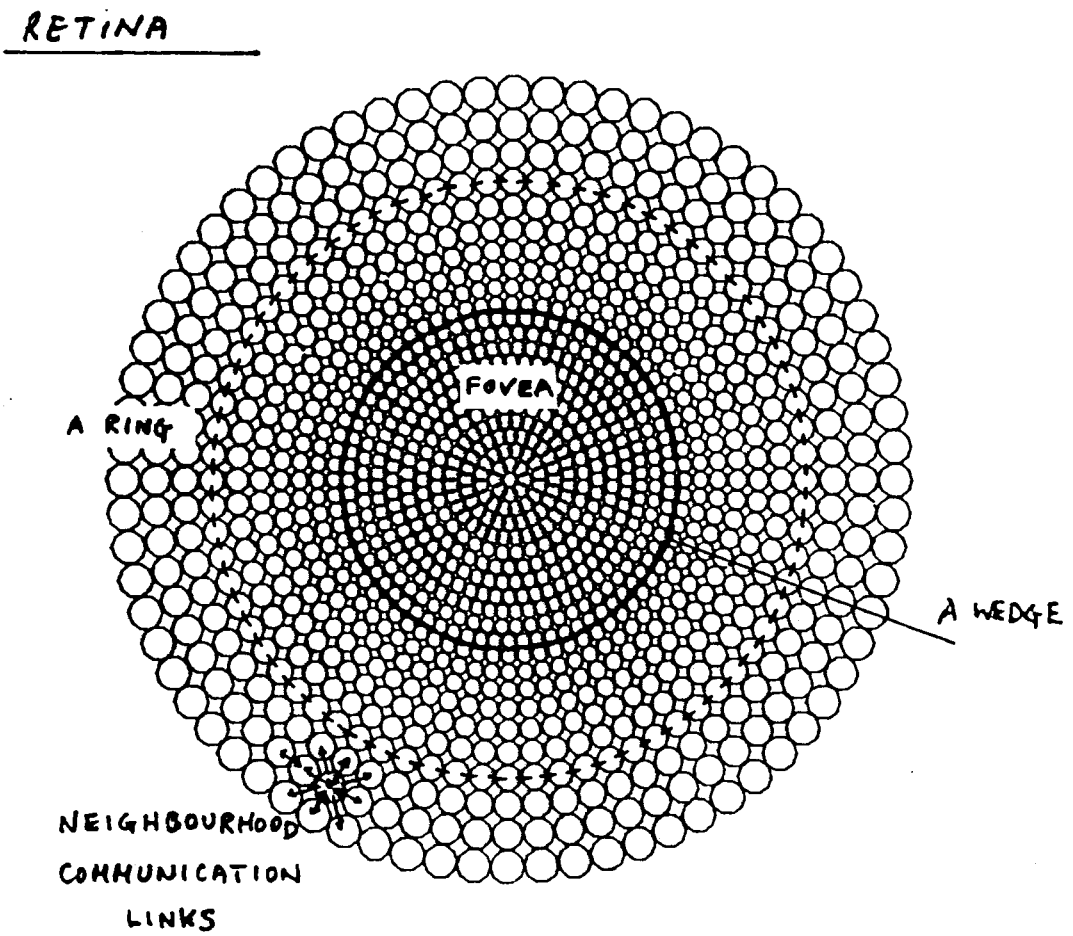


Figure 5

The Retina

resolution, while the remaining area is called the periphery. The retina is arranged in rings and wedges for easy addressing. Each receptor (bubble) represents a processor and the diameter of the smallest bubble equals the length (or width) of one pixel in the diagram. Each bubble can communicate with its neighbors and also with the retinal supervisor which broadcasts the procedures to be executed by the parallel processors. Such a simple communication network will help to ensure possible future hardware implementation. Note that the retinal supervisor is responsible for a certain amount of sequential processing.

3.3.1 ROUTER's retina vs. WHISPER's

Note that the retina of ROUTER differs from that of WHISPER in that ROUTER has a fovea. Without the fovea, the very central portion of the retina cannot be filled with bubbles that are getting smaller and smaller in size. (Eventually, these bubbles will have infinitely small size.) In other words, there will always be a central hole that cannot be filled by the very small bubbles. This singularity problem was not addressed in WHISPER. The fovea of ROUTER provides a solution to the problem. Instead of using bubbles that are decreasing in size to fill the central part, bubbles of unit diameter are used. (Note that the innermost peripheral bubbles also have unit diameter.) Despite the fact that the number of bubbles in a particular foveal ring is usually less than that in a peripheral ring,

the structure of the fovea does not conflict with the wedge organization of the retina since each foveal bubble is logically assigned to one or more wedges. Therefore, scaling and rotation of an image on the retina are supported both on the fovea and the periphery.

3.3.2 Other possible retinal topologies

The present retinal topology is not the only possible one. The geometrical arrangement of a sunflower's seeds seems to be a good alternative. A similar structure is also observed in the central portion of a crysanthemum (see Figure 6). Interestingly, Weiman and Chaikin's (1977) logarithmic spiral grid for image processing and display is very similar to the flower's central part. Sandini and Tagliasco (1980) also have a retina-like structure similar to Weiman and Chaikin's (see Figure 7). However, these authors have not solved the singularity problem of a retina-like structure.

3.3.3 The Perceptual Primitives

Each perceptual primitive detects a particular domain independent feature in the diagram. The HLR asks the retina about a certain feature, and the retina uses the appropriate perceptual primitive to answer the question. In other words, the perceptual primitives are sequential/parallel algorithms chosen by the HLR for the retina to execute. Examples of perceptual primitives are: find the bubble

Figure 6

The central portion of a flower

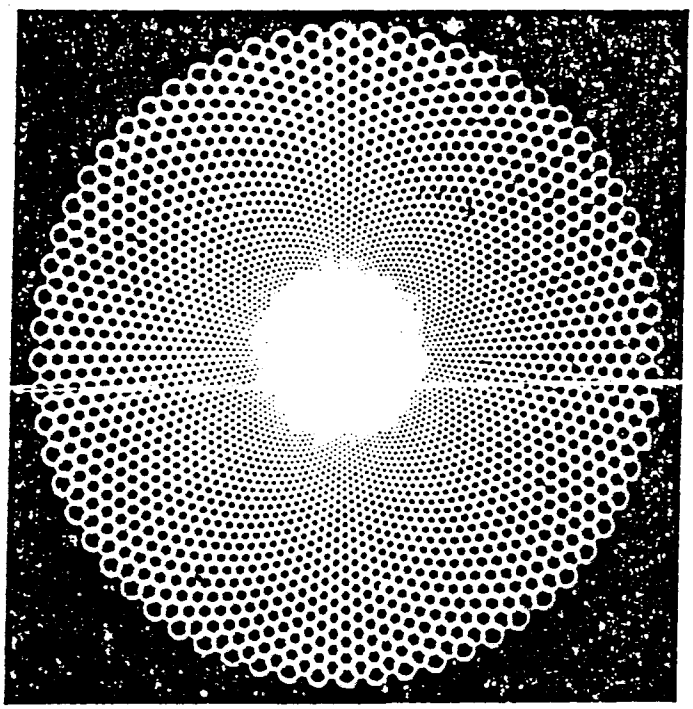


Figure 7

Sandini and Tagliasco's (1980) retina-like structure.

closest to the retinal centre satisfying a given condition; scale the retinal image; rotate the retinal image; and find if there is a straight path between two points. The following procedure illustrates how to find a clear straight path between two points, say A and B.

The retina is first fixated on one of the points arbitrarily (say A) , and then the retinal supervisor directs each retinal bubble to execute the following steps:

Step 1. If the bubble value is empty then stop.

Step 2. If the bubble is not on the same wedge as point B then stop.

Step 3. If the bubble has ring number smaller than that of point B, then send a "NO" message to the retinal supervisor.

Step 4. Stop.

So if the retinal supervisor has not received a message after a certain time delay, then there is a clear straight path between points A and B.

3.3.3.1 Retinal Scaling

It is extremely easy to scale the retinal image on the periphery by employing neighbourhood communication. For example, to expand the retinal image, each peripheral bubble simultaneously sends its value as a message to its outer wedge neighbour (Figure 8). In order to bring about the desired scaling, the message passing process has to be repeated sequentially. A proof of the scaling property of

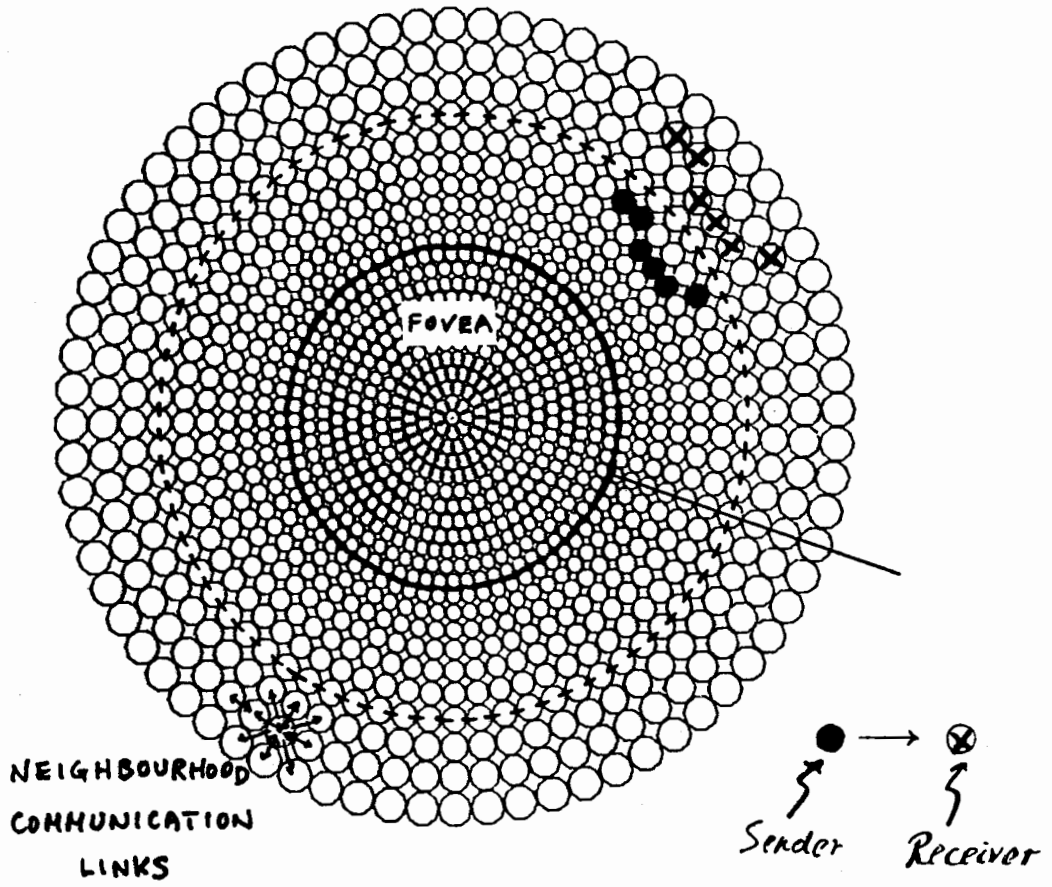


Figure 8

Each bubble sends its value to its outer wedge neighbour.

the periphery has been given by Funt (1976). Although scaling on the fovea is not as easy as on the periphery, it is not too difficult. Since each scaling step involves a constant scaling factor, what each foveal bubble has to do is to determine if it should pass its value to its logical wedge neighbour. More precisely, each foveal bubble must simultaneously execute the following:

1. Let D = the distance from the centre of the retina to the centre of the bubble.
2. Let D = D x scaling factor.
3. If D lies outside of the bubble's area, then pass its value as a message to its logical wedge neighbours.

It turns out that after a few scaling steps, some foveal bubbles may receive a message and yet their values have not been passed. The resulting values of those bubbles will contain all colours currently covered by them. There is still one more complication. When scaling objects on the fovea, gaps will occur since some bubbles will have passed their values and yet have not received a message from their neighbours (Figure 9). The solution to this problem is to let each foveal bubble record the colour its inward or outward wedge neighbour is looking at. Then after the scaling, each appropiate foveal bubble can check and fill the gaps that should not be there.

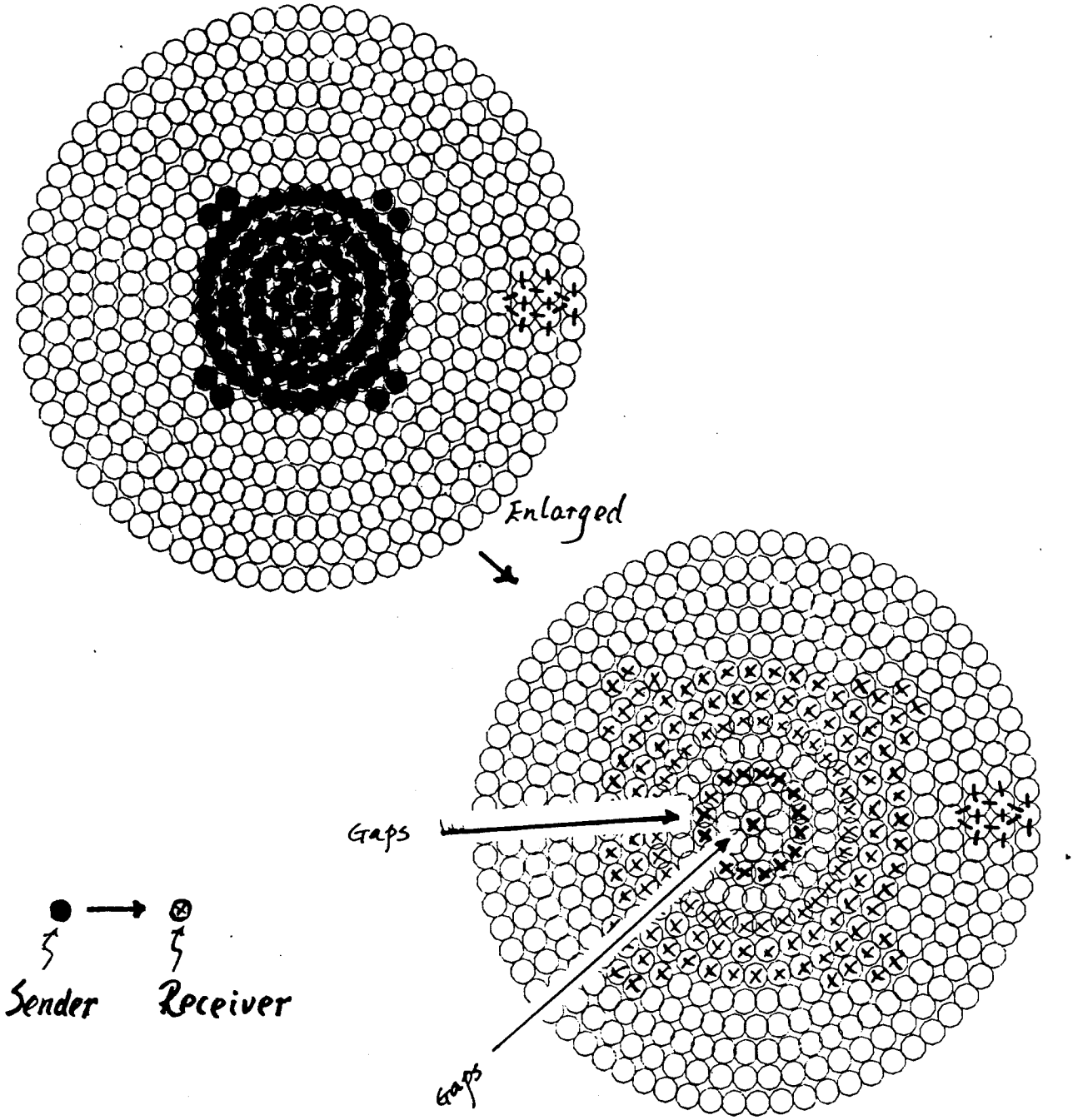


Figure 9

Gaps occurred in the fovea as an object was enlarged.

3.3.3.2 Retinal Rotation

This is very similar to the retinal scaling. For the periphery, each bubble passes its values as a message to its clockwise ring neighbour or anticlockwise ring neighbour depending on the rotational direction. For the fovea, each bubble determines if it has to pass or not. Actually, retinal rotation is simpler than the retinal scaling since for each foveal ring, either all bubbles in that ring pass their values or none do.

3.3.4 Implementation of the Retina

The retina is simulated by a two-dimensional array. Figure 10 shows the logical-to-physical mapping of the retina. Each element of the array represents a bubble which has a name of the form P*w*-*r* where *w* denotes the wedge number and *r* denotes the ring number. For example, P8-15 is the name of the bubble with wedge number 8 and ring number 15. Each bubble has a 16-bit value representing the current contents of the bubble. Each bit represents a colour, so each retinal bubble is capable of distinguishing 16 different colours. The primary advantage of this encoding scheme is that it saves memory space and also allows the retina to be filled quickly.

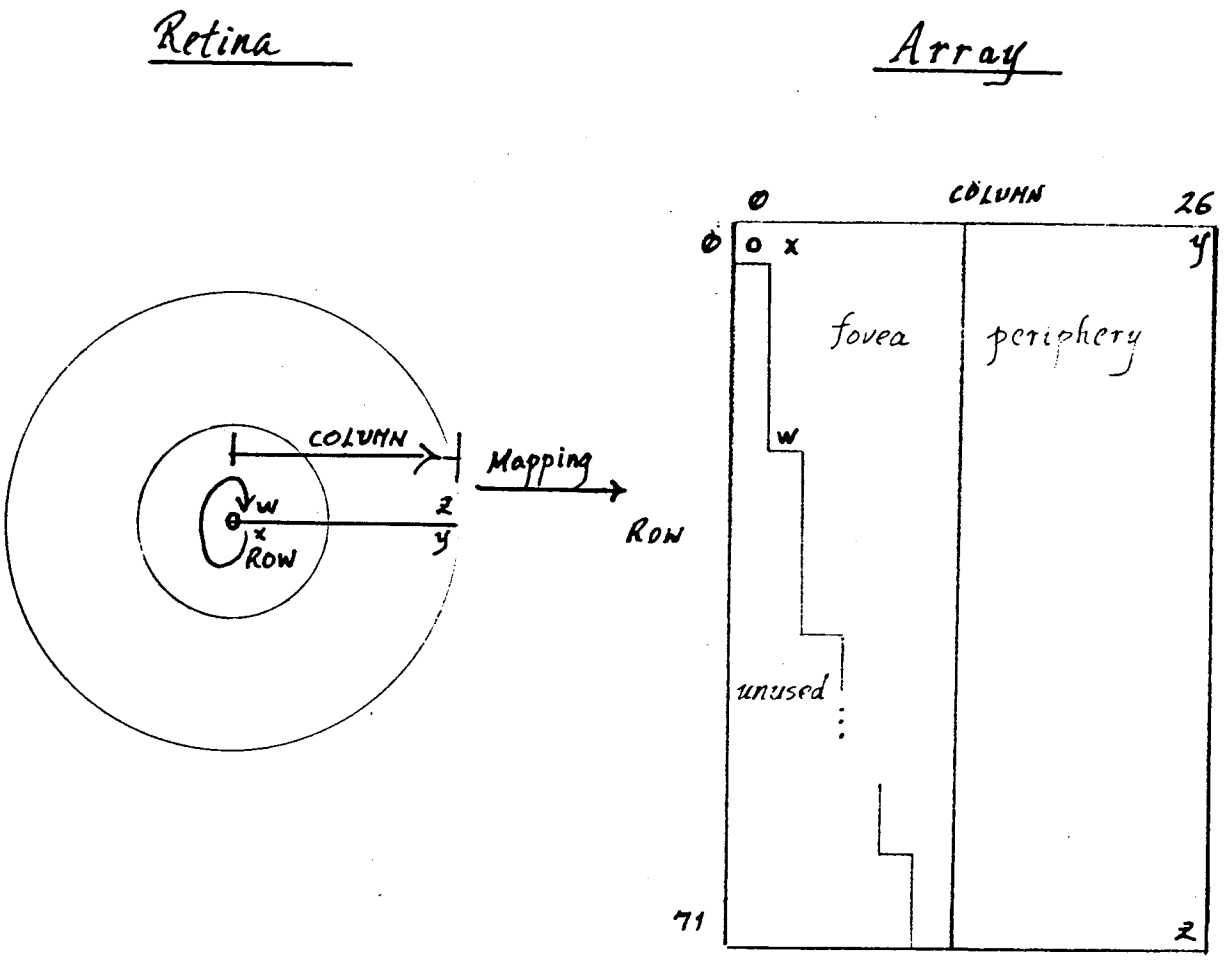


Figure 10

The logical to physical mapping of the retina.

3.4 The Mapping from Diagram to Retina

The diagram of the IC is simulated by a 70 x 70 array. The mapping from the array diagram to the retina can be visualized as overlaying the retina on top of the diagram and filling the bubbles accordingly. The grid lines of the routing IC will fall on the even numbered rows and columns. The width of a pixel in the array diagram is assumed to be one unit.

4. See and Compute

In the previous chapter, I described the overall structure of ROUTER. Now it is time to present the algorithms used to solve the routing problem.

4.1 Methods

A flowchart is given in Figure 11 showing an overview of the flow of control of the system. First, ROUTER reads the sizes of each of the rectangular functional blocks and their positions on the two-layer IC. Then it reads an interconnection list which specifies a set of electrical connections or nets. Boxes 2 and 3 represent the most important parts of the system and they will be detailed in the sections to follow.

4.2 Grouping Similar Nets

It is interesting to note that many of the nets on a routing IC are similar. Given a set of similar nets, can a person use it to facilitate routing? It seems that if he could route a net from that set, then the rest of nets could also be routed "like" the first one. But could ROUTER do it too? This will be investigated after the section where the actual routing algorithms will be detailed. Let us first describe how ROUTER groups similar nets.

First, all nets with the same number of pins are grouped together. Then for each group, the following is

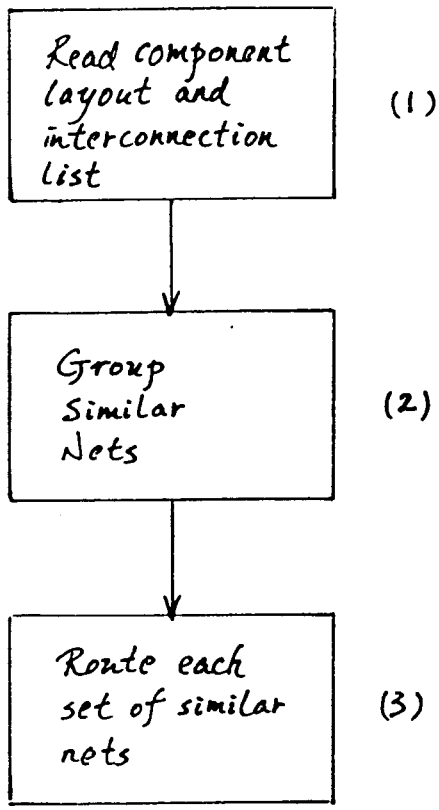


Figure 11

An overview of the flow of control of ROUTER.

executed:

1. For all possible pairs of nets (denoted by net-A and net-B)
   a. if net-A and net-B pass both a preliminary test and a similarity test
      DO
      1) Put them into the same set.
      2) Record the scaling and rotational information.

The preliminary test calculates the relative distances of pins to the centre of gravity of the nets.

How well ROUTER can apply similarity in routing depends on many factors. One is the accuracy of the similarity test. However, similarity is quite subjective and a completely accurate similarity test is, if not impossible, very difficult to find. Nevertheless, a similarity test based on the location of pins, the block that each pin belongs to, and the obstructions within the net has been developed. Given net-A and net-B with their centres of gravity, the similarity test can be loosely described as follows:

1. The retinal supervisor directs the retina to fixate at the centre of gravity of net-A. Note that the area of interest will be specially coloured so that the retina will not be confused during this phase.
2. Each bubble in the retina remembers what it has seen.
3. The retina is fixated at the centre of gravity of net-B.
4. A scaling factor is calculated.

5. The retinal image is scaled.
6. For n = 0 to 3

   DO

   a. Rotate the retinal image by 90 degrees in the clockwise direction
   b. For each bubble

      DO

      1) Compare the colour a bubble is looking at to the colour in its memory cells. (use the Excuse Mechanism)
      2) Report match or no match to the retinal supervisor.
7. The retinal supervisor finds the best fit. (Best fit will be defined later.)
8. If no best fit, return no and stop.
9. The scale and rotation amounts are returned.
10. Stop.

4.2.1 Excuse Mechanism

It is rarely the case that two nets have exactly the same configuration. In other words, the colour seen by a bubble seldom matches the colour it is currently looking at. Thus we need a mechanism-- the excuse mechanism-- which allows us to match things that are close together. The excuse mechanism works as follows:

If a bubble is looking at something green, say, and its memory cell records something red, then the bubble asks its immediate neighbours if they have red recorded in their memory cells. If there is at least one such neighbour, a match is found.

4.2.2 Comparison

The comparison phase is based on three factors: pin-locations, blocks that pins belong to, and obstructions within the net. Note that in order to avoid false matches as illustrated in Figure 12, we have to compare net-A to net-B and then net-B to net-A. This is analogous to the definition of set equality:

if Set A is a subset of Set B and if Set B is a subset of Set A, then Set A equals Set B.

The following definitions are necessary for understanding how the best fit is found.

Definitions

Let

- t1 be the average percentage of match of pin-locations,
- t2 be the average percentage of match of the blocks that associate with the pins, and
- t3 be the average percentage of match of obstructions within the nets.

Two nets will pass the similarity test if

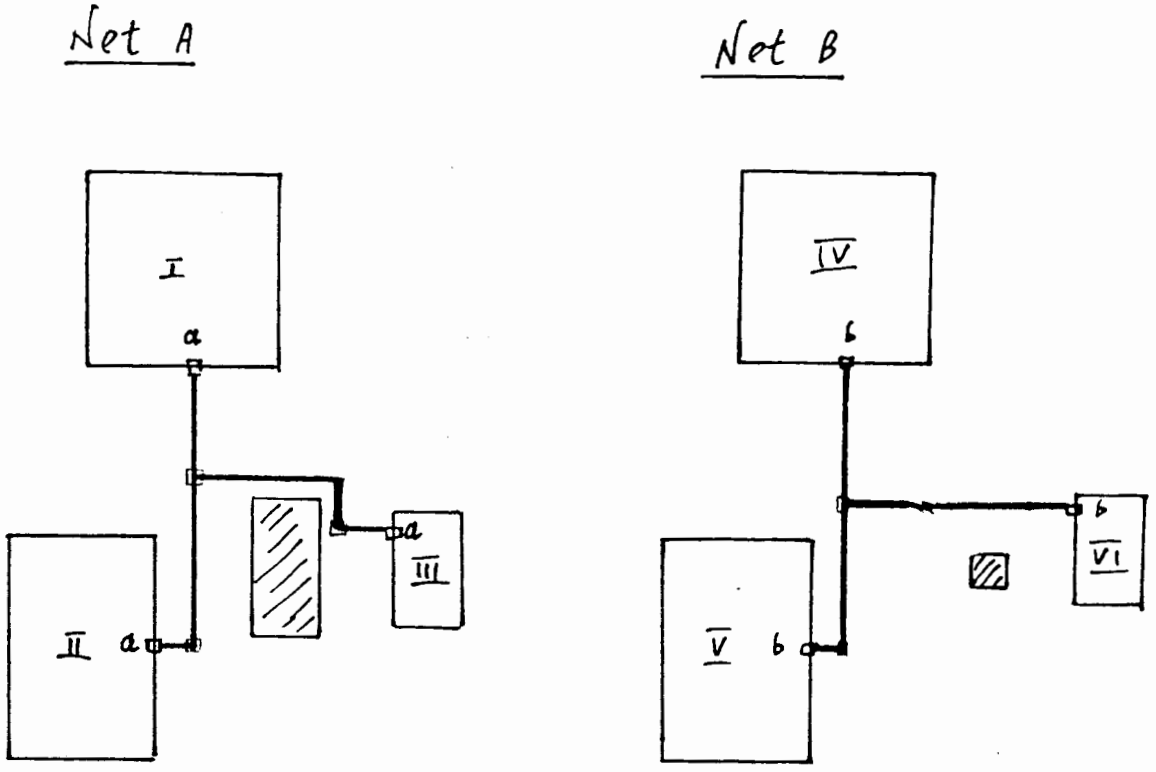


Figure 12

A false match will occur if net-A is compared to net-B, but net-B is not also compared to net-A.

$$t1 > 0.8 \text{ and } t2 > 0.7 \text{ and } t3 > 0.6$$

The best fit is defined as the highest score computed from the sum of t1, t2 and t3. The parameters in the similarity decision function may not be the best ones. An example is shown in Figure 13 where net-1 and net-2 are recognized as similar. There are, however, some cases where two nets are similar and yet are not recognized as such by ROUTER. (See Figure 14)

4.3 Route each set of similar nets

After similar nets have been grouped together, ROUTER routes each set of similar nets one at a time. Routing nets with the largest number of pins first seems to be a reasonable choice. This routing phase is the central part of the system. It contains a preprocessor, two Lee's routers, and an analysis phase. The preprocessor makes use of likely paths and similarities between nets. The two Lee's routers are responsible for wire connections. The analysis phase is an optimization phase.

4.3.1 Preprocessor

Given a net to be routed, the preprocessor draws the likely paths of the net on the IC (See Figure 15). The reason those paths are called likely paths is that when a net is routed, a lot of wires are likely to fall on these paths. The idea here is to move wires that lie on the

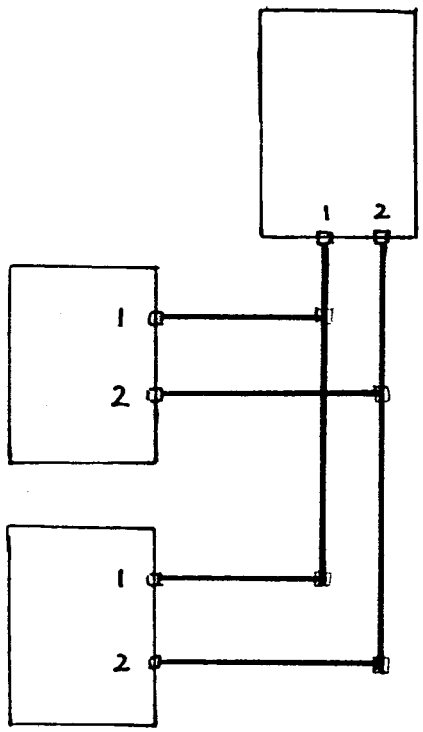


Figure 13

Experimental results of the similarity test.

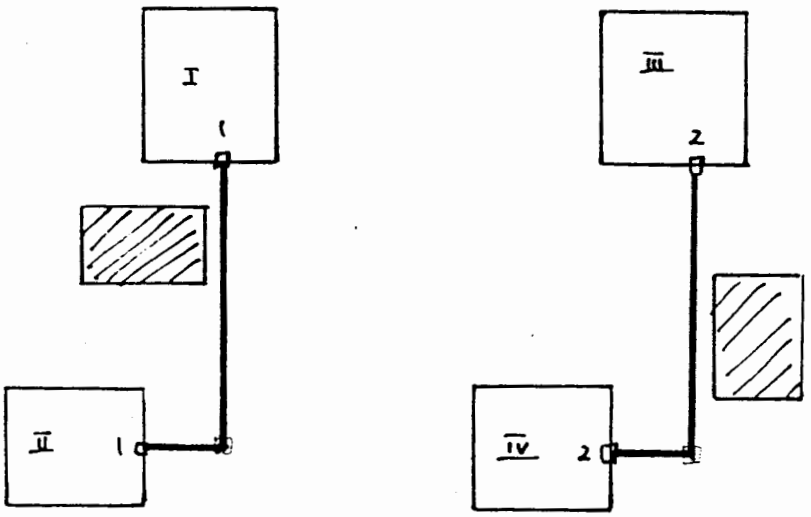


Figure 14

Cases that ROUTER does not recognize as similar.

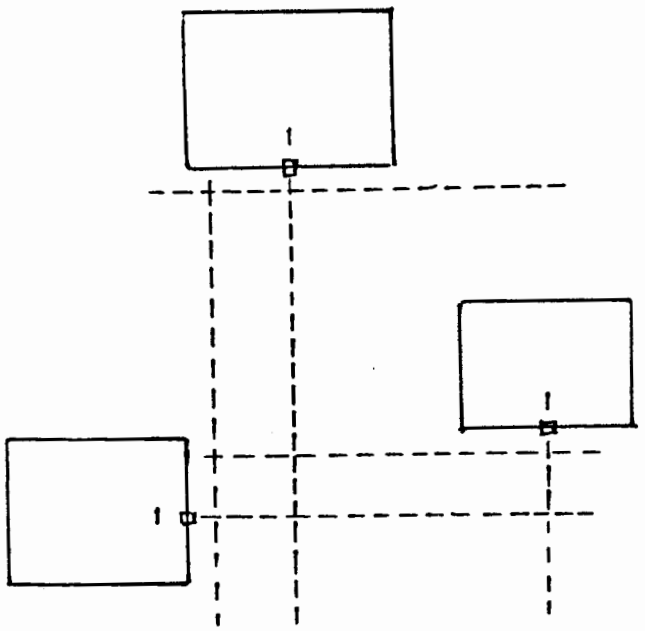


Figure 15

Likely paths are denoted by dotted lines.

likely paths to empty tracks such that the total length of each wire and its number of vias does not increase. One type, the "Z" wire, has been identified that can be moved in two directions without increasing its length or number of vias as shown in Figure 16. Other types of wires can also be moved without penalty but have not been included in this thesis as shown in Figure 17.

4.3.1.1 How ROUTER makes use of likely paths

Having drawn the likely paths, the retina fixates at the centre of the diagram. It detects all segments of wires that are lying on the likely paths. Then ROUTER finds all the Z wires and moves them aside if possible.

4.3.1.2 Use of Similarity

After moving aside type Z wires, a net is ready to be routed. ROUTER first checks if this net is similar to any other nets that have been routed. For example, if net-B is the net under consideration and net-A is a routed net that is similar to net-B, then ROUTER will execute the following:

1. The retinal supervisor directs the retina to fixate at the centre of gravity of net-A and record the solution of net-A. (i. e. the wires of net-A)
2. Apply the necessary scaling and rotation to the retinal content. (These parameters are readily available since net-B stored this information when it was compared to net-A in the phase of Grouping Similar Nets.)
3. Move the retina to the centre of gravity of net-B.

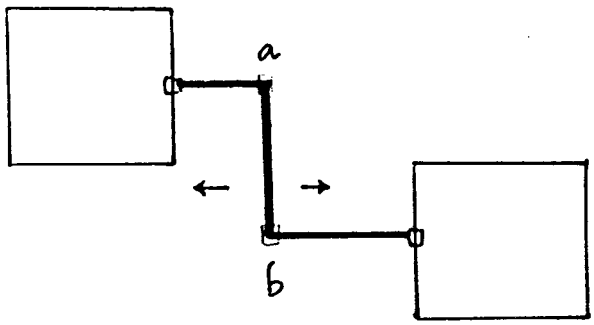


Figure 16

Type Z wire.

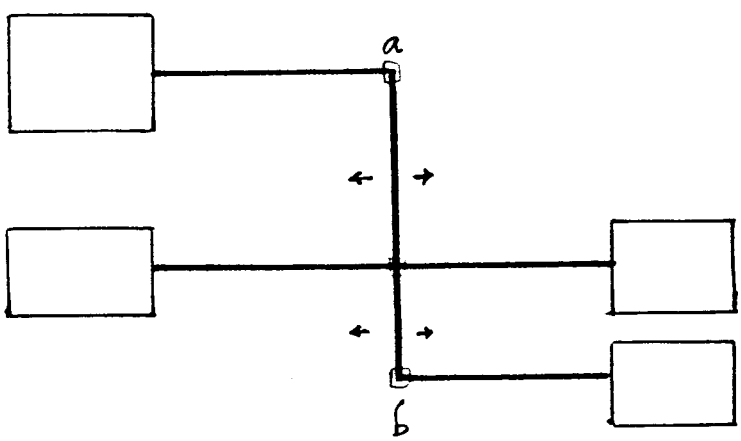


Figure 17

Wire ab can be moved in the direction indicated without penalty.

4. For each bubble

   DO:

   a. If a bubble records a part of the solution THEN

      use a special colour to colour the area that it is looking at

5. Stop.

Note that if a bubble is 3 units in diameter, then it will paint the area it is over with a circle of 3 + "BLUR" units in diameter. "BLUR" was set to 4 in the program. This is necessary since there are some little gaps between bubbles. If those gaps are not filled, then they may affect the Lee's routers which will be presented in the next section. The claim here is that the specially coloured area will very likely contain a solution to the net we want routed. But what about a net that is not similar to any other or is the first one from a group to be routed? In this case, the easiest solution is to let Lee's router search the entire IC or a suitably large rectangle that contains the net to be routed. Another solution may be to use the algorithm by Rubin (1974).

4.3.2 Lee's Router

Lee's algorithm (Lee 1961) is a well-known method of finding the shortest path between two points in a maze. Given two points say A and B on a 2-D grid IC, Lee's algorithm can be briefly described as follows:

1. Initialize each grid point to zero.

2. Pick a point, say A, label it as 1.
3. Let n = 2.
4. For each grid point n-1, label its neighbours, if possible, as n.
5. Increment n.
6. If B is now labelled, then stop.
7. If the whole IC has not been traversed, goto 4.
8. Stop.

Figure 18 shows how Lee's router works. When point B finally gets a label, in this case 4, the algorithm stops. Finding the shortest path simply requires tracing back from point B to point A.

4.3.2.1 Using the retina to apply Lee's algorithm

An effort has been made to implement Lee's algorithm on the parallel processing retina. The first approach was to fixate the retina on, say pin A and then use the retina's communication links to broadcast the labels until a label reaches the bubble representing pin B. However, this method has two problems. First, bubbles in the periphery do not have the resolution to distinguish if the areas they are looking at can be labelled. For instance, consider Figure 19 in which a big bubble is over two tracks. One of them is occupied and the other is free. This bubble, however, will not see the empty track. Second, the grid of the IC does not correspond to the positions of the bubbles in the retina.

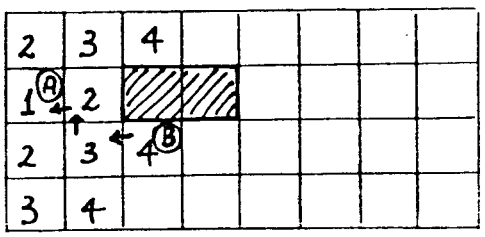


Figure 18

An example showing how Lee's algorithm works.

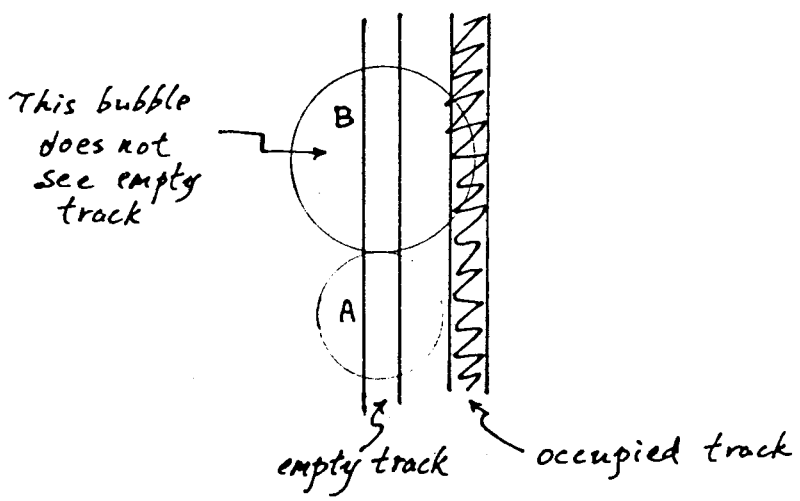


Figure 19

This bubble could not see the empty track.

Rather than fixating at one of the pins, a second approach was to move the retina midway between the two pins. This method, however, still suffers from the same problems. Since the peripheral bubbles have rather poor resolution, they are deactivated during the labelling process. Since the foveal bubbles' diameter equals the width of a grid line, when the retina fixates on a grid point, some foveal bubbles will fall on each grid line of the IC. The foveal bubbles that fall on the grid lines will be marked as special bubbles called the Lee's bubbles. The neighbouring Lee's bubbles will then be called the Lee's neighbours. The Retinal Lee's Algorithm can be loosely described as follows:

1. Initialize a labelling array to zero.
2. Use a "special colour" to mark one pin. (say pin A)
3. The element of the labelling array that corresponds to the location of pin A is labelled as 1.
4. Let n = 2.
5. Move the retina to the area such that the fovea is able to see the special colour.
6. If the retina sees no special colour, goto 9.
7. For all foveal bubbles that see the special colour
   DO
   a. For all its Lee's neighbours (at most 4)
      1) if one sees an area which is OK to be labelled, send its name to the retinal supervisor.
   b. If at least one Lee's neighbour says OK
      DO

1) Label the grid point as n.

2) Erase the special colour in the area covered by this foveal bubble from the IC.

8. Goto 5.

9. Since the retinal supervisor has all the names of the bubbles that say OK, colour those areas on the IC with the special colour and increment n.

10. Repeat step 5.

The termination criteria are: (1) if a bubble labels pin B, then stop. (2) if no more of the special colour exists on the IC, stop. (This is the case when there is no path between the two pins.)

Note that the periphery is primarily responsible for performing Step 5. In the current implementation, the position of the farthest bubble away from the retinal centre that sees the special colour will be the location for the next fixation. Only about 8 foveal bubbles will actually label the IC in each fixation.

One advantage of this algorithm is that it can make use of the property of similar nets. As mentioned in a previous section, a certain area of a net will be coloured so that the retina can concentrate on that area of interest. This approach could significantly reduce the search space often traversed by the ordinary Lee's algorithm.

Since true parallel processing is not possible on the VAX-11/750, and the Retinal Lee's Algorithm was originally implemented in LISP, it took roughly three minutes of

elasped time to route a single wire. It has been estimated that the number of fixations ROUTER has to make is of the order of k ** 3 (i.e. k to the power of 3) where k is the length of the wire to be connected. For these reasons, a Lee's algorithm that does not use the retina has been used in testing the system. However, this algorithm can also make use of the property of similar nets by only searching the smallest rectangular area of the IC that contains the special colour.

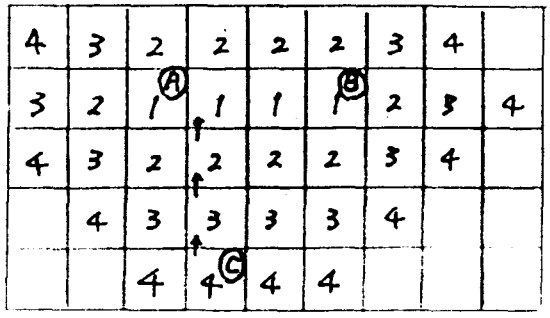
4.3.2.2 Routing a Net

The Lee's algorithm we have just described can only route two pins at a time. However, what is needed is a router that can route nets with more than two pins. It turns out that it is relatively easy to modify Lee's algorithm to do just that as was suggested by Akers (1972). The idea is to first route any two pins. Then in a second pass of the Lee's router, the whole area under the routed wire will be labelled as 1. The labelling process broadcasts the labels until one of the unconnected pins of that net is labelled. This process is illustrated in Figure 20.

4.3.3 Analysis

After a wire has been laid out, control is passed to the analysis phase. This phase is responsible for detecting definite blockages, and minimizing vias and wire lengths.

Pin A is first connected to pin B



Routed net:

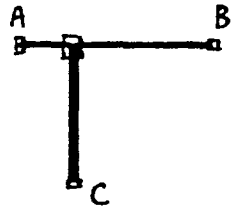


Figure 20

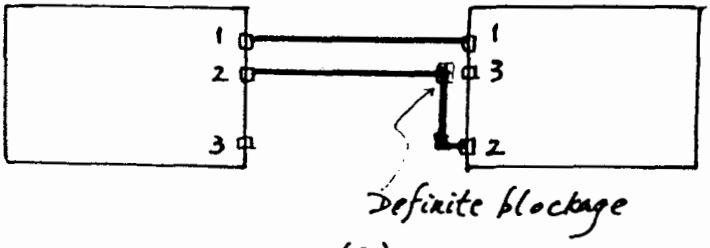
Using modified Lee's algorithm to route a net with three pins.

4.3.3.1 Definite blockages

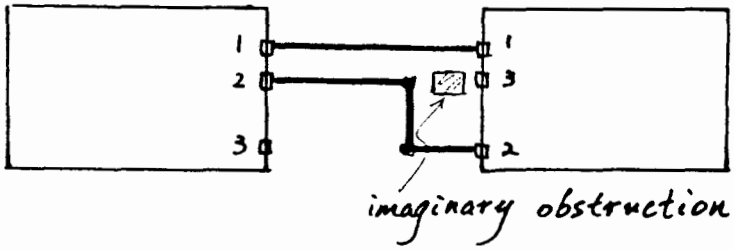
One of the biggest problems with a sequential router is that wires already laid out may block other unrouted nets. Consider Figure 21a; net 2 makes it impossible to reach pin 3. One solution is to put in an imaginary obstruction as in Figure 21b, and then reroute 2. ROUTER detects definite blockages with a specialist called definite-blockage-detector. Given the pins of a net and the location of a via, the definite-blockage-detector performs the following:

1. Fixate the fovea on the via.
2. If one or more of p0-2, p4-2, p8-2 and p12-2 sees a pin that does not belong to the net under consideration
   a. Return definite blockage found.
   b. Stop.
3. Return no definite blockage found.
4. Stop.

If definite blockages are found, then imaginary obstructions are put in their places and the wires are rerouted. Note that removing definite blockages does not guarantee the routability of other unrouted nets. What ROUTER has achieved here is that it can recognize obvious blockages. Preventing blockages remains an area for further research. (Supowit 1982)



(a)



(b)

Figure 21

(a) Pin 3 is definitely blocked. (b) Imaginary obstruction placed by ROUTER and net-2 rerouted.

4.3.3.2 Minimizing Vias and Wire Length

In routing a net, there are many situations where vias and wire length can be minimized. Consider Figure 22a, the solid path wxyz is routed first, and the dotted path mn is then connected to path wxyz. But if the segment xy can be moved downward as in Figure 22b, the total length of the net will be decreased. This situation is termed a type-1 defect. As in the earlier case, path wxyz is a Z wire.

A type-2 defect is illustrated in Figure 23. In this case, by moving the Z wire as shown in Figure 23b, not only could we save one via, but we could also shorten the net and thereby free a track for other nets.

4.3.3.3 How does ROUTER rectify the defects

Let us again consider Figure 21. First of all, when a wire has been routed to the net it belongs to, the last segment (here only one segment namely mn) of that wire and the point of contact are registered. Then ROUTER finds the segment xy that is perpendicular to mn. From then on, ROUTER determines whether xy is actually part of a Z wire. In this case, it is. ROUTER then determines the defect type.. Note that the retina was not used in the determination of the Z wire even though it is theoretically possible. Rather, the retina is used to check if the newly moved wire is still within the routing constraints. Two specialists-- track-checker and definite-blockage-detector- are responsible for the checking. Since

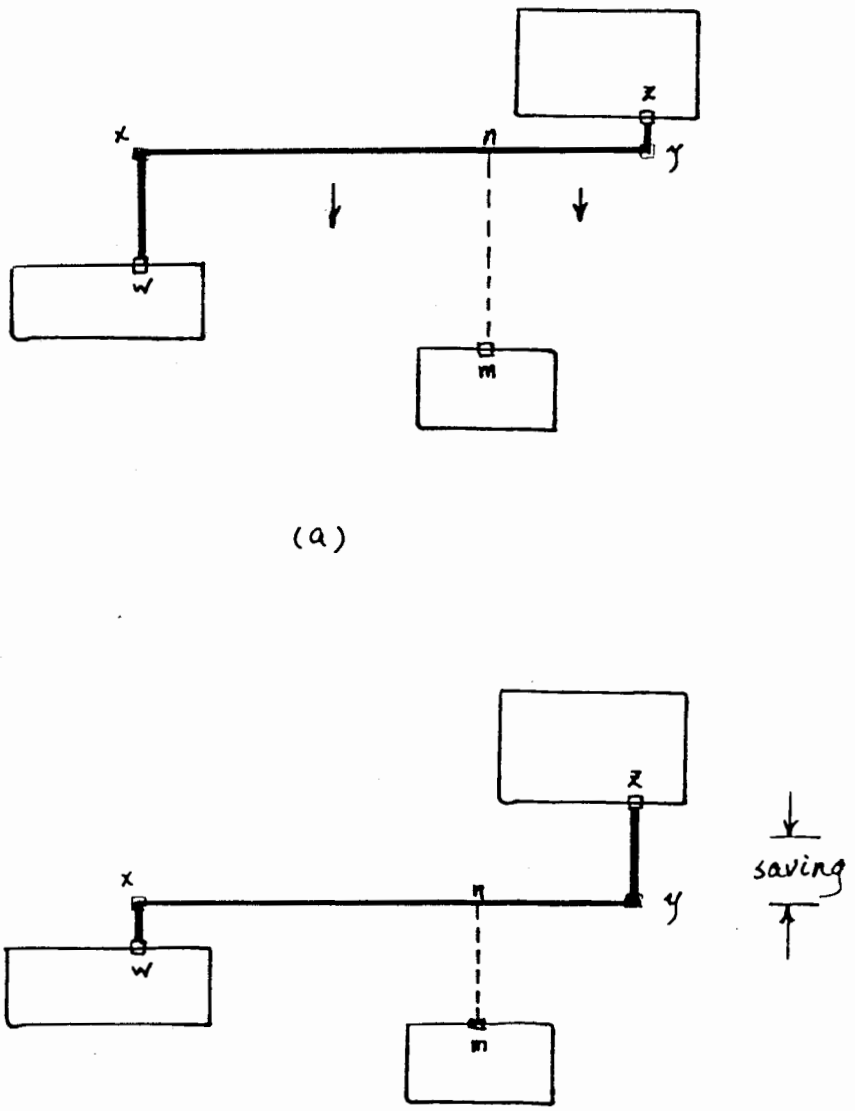
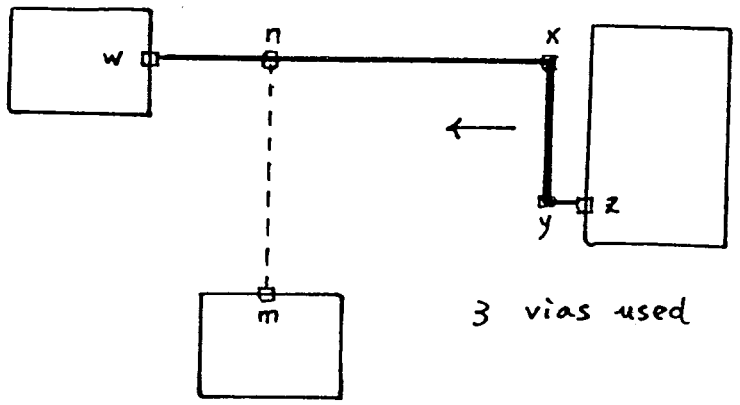
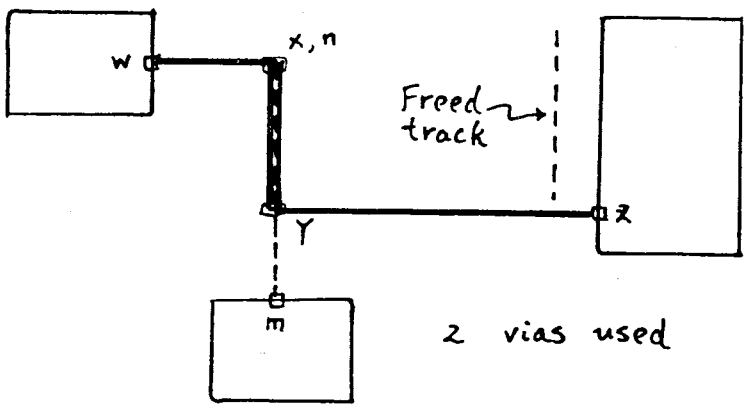


Figure 22

Type-1 defect. (a) wxyz is a Z wire. (b)wxyz moved downward.



(a)



(b)

Figure 23

Type-2 defect. (a) wxyz is a Z wire. (b) wxyz moved left.

definite-blockage-detector has been described previously, only track-checker will be elaborated here.

If we want to check if a track is free, it is first specially coloured. Then the retinal supervisor directs the retina to fixate at one of the two ends of that track. Since the retina also knows the orientation of the track, the procedure to check a track is:

1. For each bubble

   DO

   a. If the bubble is over a special colour

      DO

      1) If it sees an obstruction colour or a via colour that does not belong to the current net

         DO

         a) If it is a foveal bubble THEN

            Return its position on the IC and the message "track not OK"

         ELSE

            Send an ALARM to the retinal supervisor

      2) If it sees a route colour

         a) Send "saw route colour" to supervisor

2. Supervisor checks if there is a message saying "track not ok"

   a. Return "track not ok" and Stop.

3. If all bubbles that saw the special colour sent the message "saw route colour"

   a. Return "track not ok" and Stop.

4. If there is an alarm from the periphery

   DO

   a. Move the retina to that location

   b. Goto 1.

5. Return "track ok" and Stop.

Although two types of defects have been identified and can be handled by ROUTER, I suspect more defect types actually exist.

4.4 Some Examples

The first example ROUTER worked on was taken from Ciesielski and Kinnen (1982). There were ten nets to be connected. Nets 5 and 8 each had three pins and the rest of nets each had only two pins. Considering Figure 24a, a type 2 defect was discovered when net 5 was connected. However, it could not be resolved. Nets 8, 1, 2, 3, were routed without any problem. When net 4 was preprocessed, wires from nets 5 and 2 were found lying on the likely paths of net 4. This is illustrated in Figure 25a and was resolved as in Figure 25b. After net 6 had been routed successfully, wires from nets 1 and 2 were found lying on the likely paths of net 7 (see Figure 26a). Those wires were moved as illustrated in Figure 26b. Note if net 1 (or net 6) was not moved, then net 7 could not be connected. Besides, the length of net 7 was also shortened.

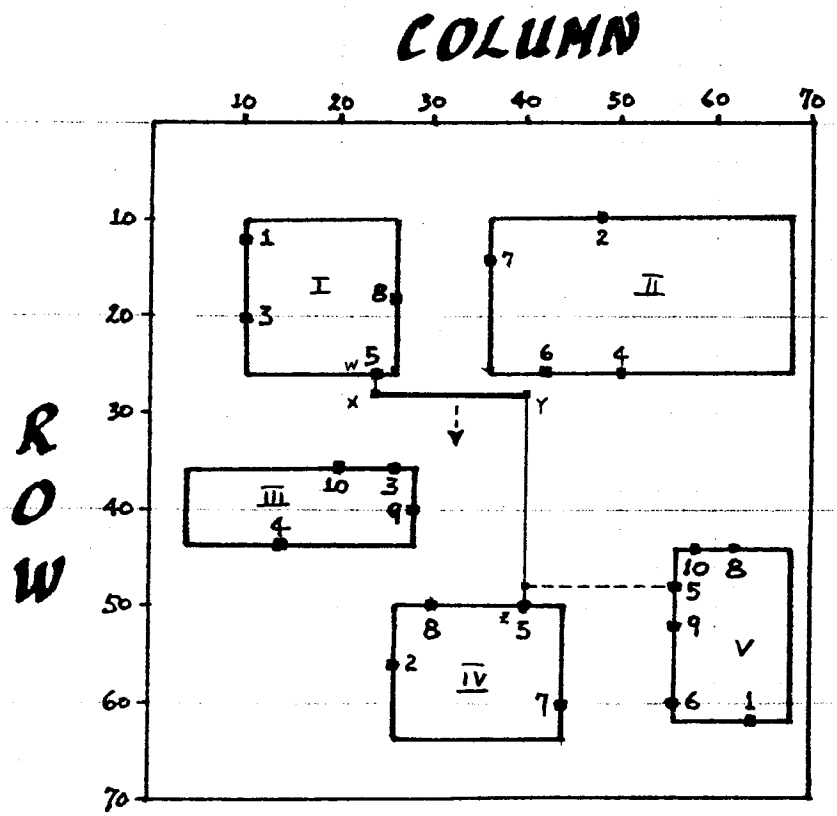


Figure 24

A type 2 defect was found but could not be resolved. wxyz is a Z wire. Wire xy should be moved downward but is block by

III

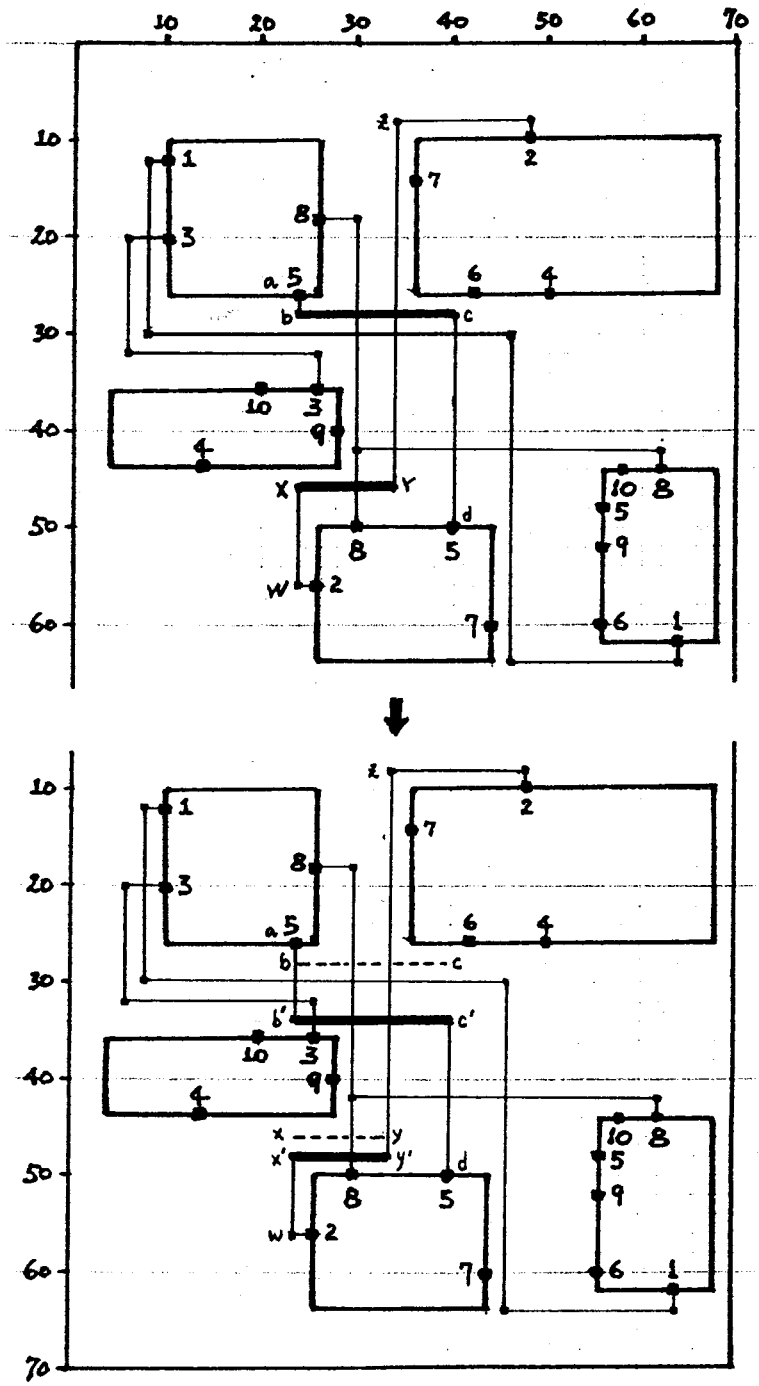


Figure 25

Two Z wires were found when net 4 was preprocessed. Wire bc moves down 3 tracks and wire xy moves down 1 track. The tracks previously occupied by wires bc and xy are now free.

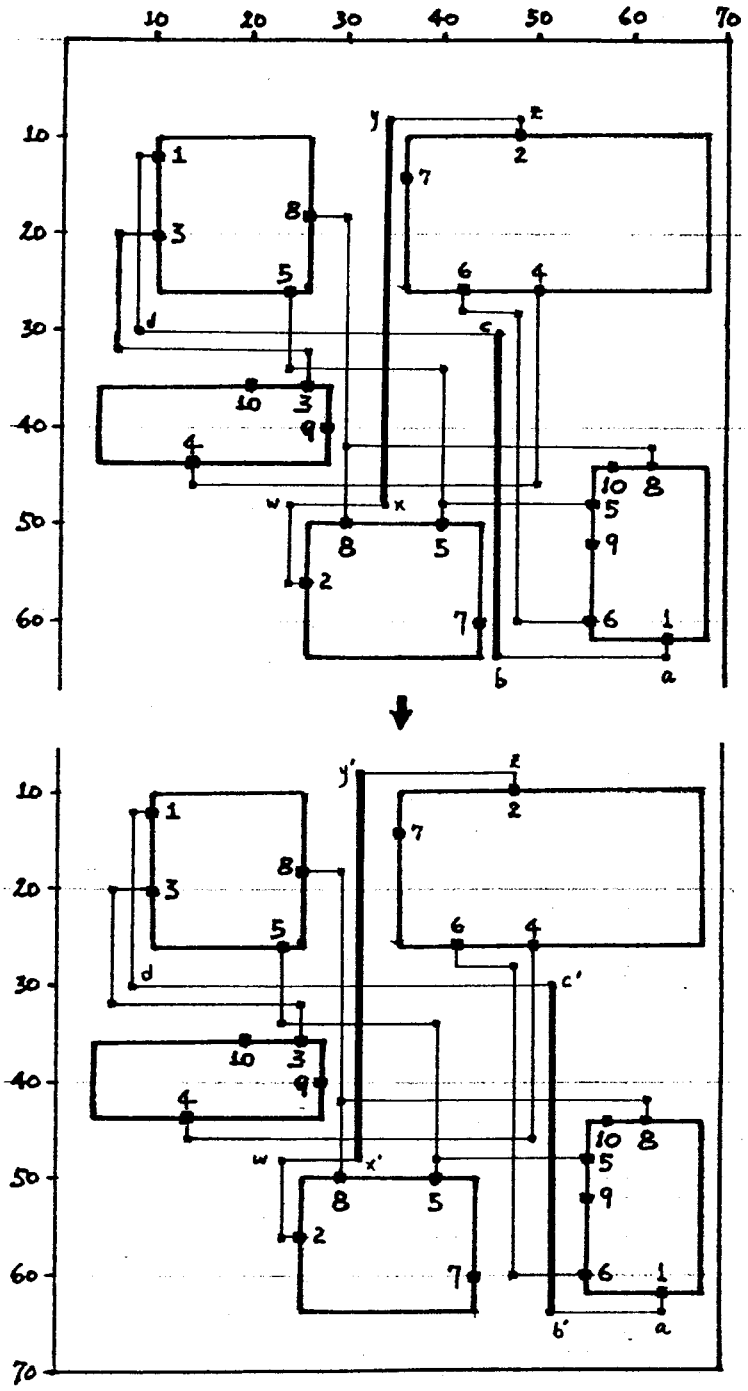


Figure 26

Two Z wires were found when net 7 was preprocessed. Wire bc moves right 3 tracks and wire xy moves left 1 track.

There was no problem with net 9. However, when net 10 was preprocessed, a Z wire from net 5 was found as in Figure 27a. It was moved aside as in Figure 27b. The finished IC is shown in Figure 28. The total wire length was 612 units and the total number of vias used was 32.

It is interesting to see how an ordinary Lee's algorithm will solve the problem presented in this example. Note the routing order was maintained for comparison. Lee's router had no trouble routing nets 5, 8, 1, 2, 3, 4 and 6. However, it could not connect net 7 since it was blocked by net 1 and net 6. (Figure 29.) After manually moving net 1 aside, the router managed to finish routing the IC with a total wire length of 624 units and a consumption of 38 vias. (See Figure 30.)

The second example is taken from The Giant Book of Electronics Projects by the editors of 73 magazine (1982) page 141. Although this is actually a part of the component layout of a display IC circuit, it helps demonstrate ROUTER's capability of handling similar nets. Of the 18 nets to be routed, ROUTER found four groups of similar nets. They are grouped as follows: 7 and 14; 1, 2, 8, and 9; 3, 4, 10, and 11; 5, 6, 12 and 13. First, net 7 was routed followed by its similar net 14 according to the solution of net 7. Second, net 1 was routed followed by its similar nets 2, 8, and 9 according to the solution of net 1. The rest of the nets were routed in the same fashion. A completed IC is illustrated in Figure 31.

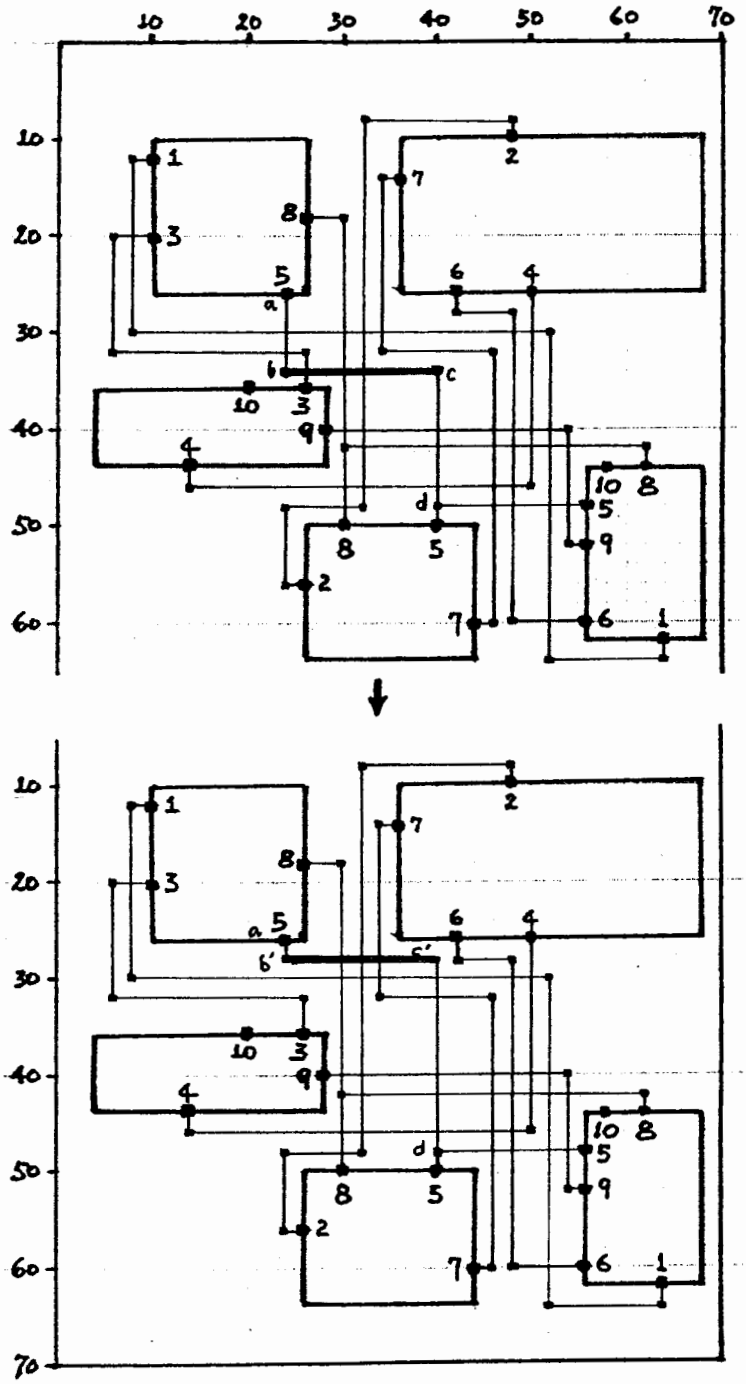


Figure 27

A Z wire was found when net 10 was preprocessed. Wire bc will move up 3 tracks.

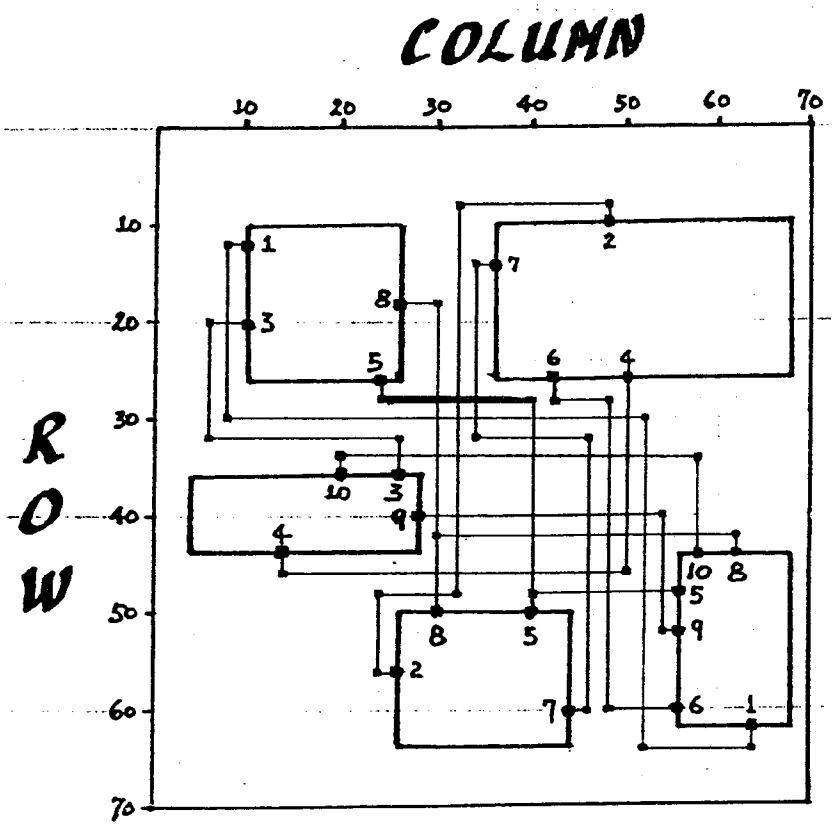


Figure 28

The finished IC for example 1.

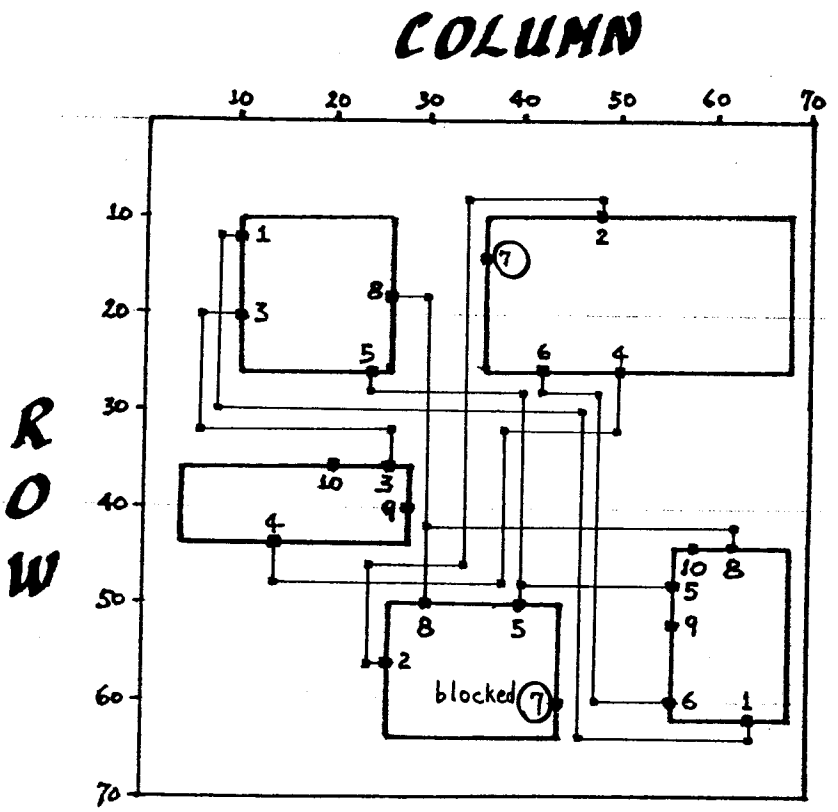


Figure 29

An ordinary Lee's router was blocked when routing net 7.

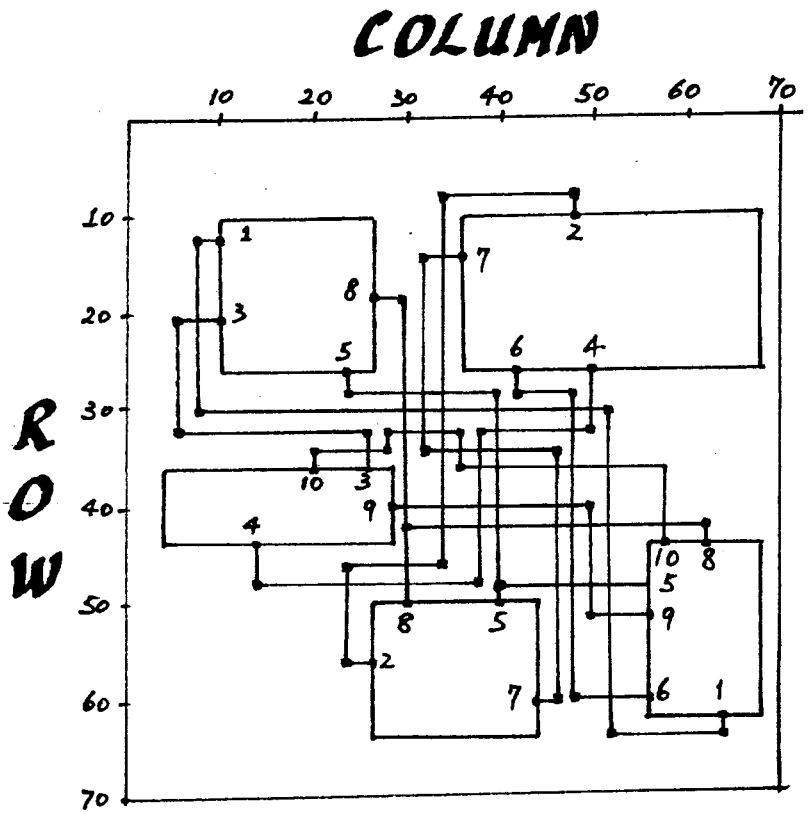


Figure 30

The finished IC produced by an ordinary Lee's router after manual intervention.

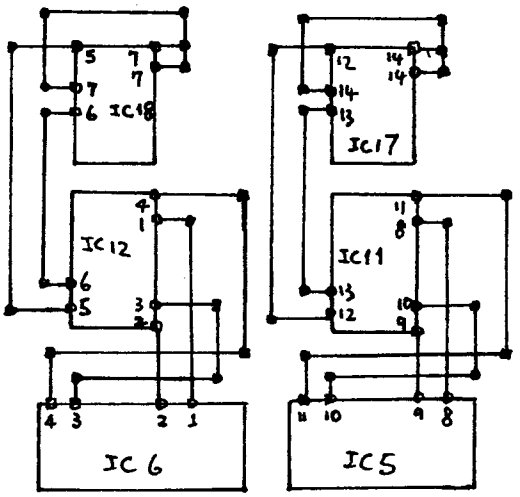


Figure 31

A completed IC.

In this example, a lot of exactly similar (identical) nets were found and if their solutions could be duplicated exactly, a lot of computational effort would be saved. Another point to note in this example is that there are actually two identical groups of components (IC6, IC12 and IC18; IC5, IC11 and IC17). If ROUTER could recognize them, then the solution to the first group would also be a solution to the second one.

The last example ROUTER worked on was the one illustrated in Figure 3a. This example shows the kind of complexity ROUTER can handle. There were eight nets to be routed. Nets 7 and 8 had the most pins (5), and nets 1, 2, 3, 4 and 5 each had four pins. Net 6 had only three pins.

Of all these nets, only nets 4 and 5 passed the preliminary test. However, they failed the similarity test. After net 7 was routed, net 8 was preprocessed. ROUTER found that a Z wire from net 7 was on a likely path (Figure 32a). That wire was moved as illustrated in Figure 32b. Then net 8 was routed and followed by net 1. In routing net 2 (Figure 33a), a type 2 defect was found and was resolved as in Figure 33b. Nothing of particular interest happened when net 3 was routed. When routing net 4, a type 2 defect was found. (See Figure 34a.) However, this defect could not be resolved. The finished IC is shown in Figure 35a. The total wire length was 680 units and the total number of vias used was 53. A finished IC produced by an ordinary Lee's algorithm is illustrated in Figure 35b for comparison. (Note that the routing order is the same in both cases.) In this case, the total wire length was 692 units and the total number of vias was 55.

4.5 Eye Movement Protocols

It is an interesting experience to study how the retina focuses on the different parts of the diagram in order to solve a problem. In many instances, ROUTER has to determine if laying a wire on a track will violate the routing constraints. As illustrated in Figure 36, point 2 is supposed to connect to point 2'. Assume that the retina is first fixated at point A. Since the retina does not see any

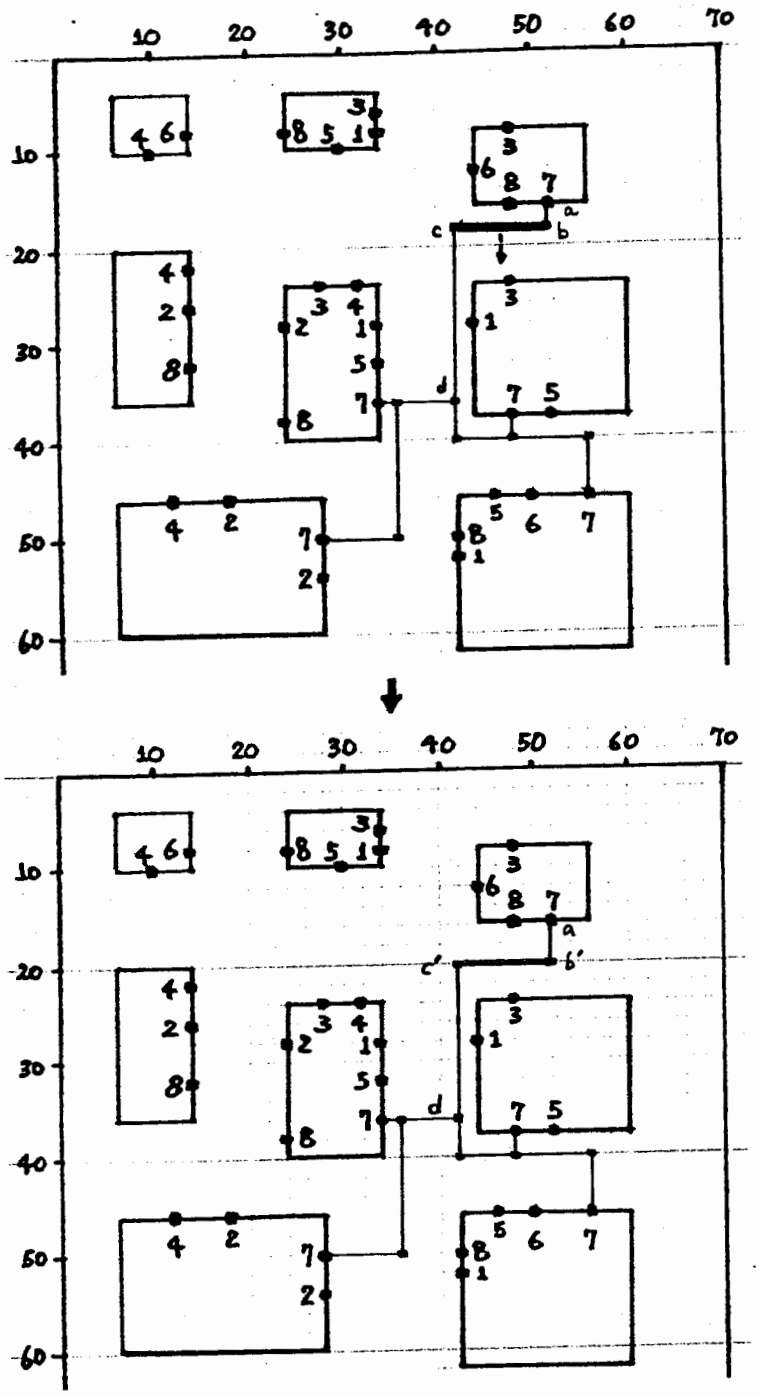


Figure 32

A Z wire was found when net 8 was preprocessed. Wire bc moved down 1 track.

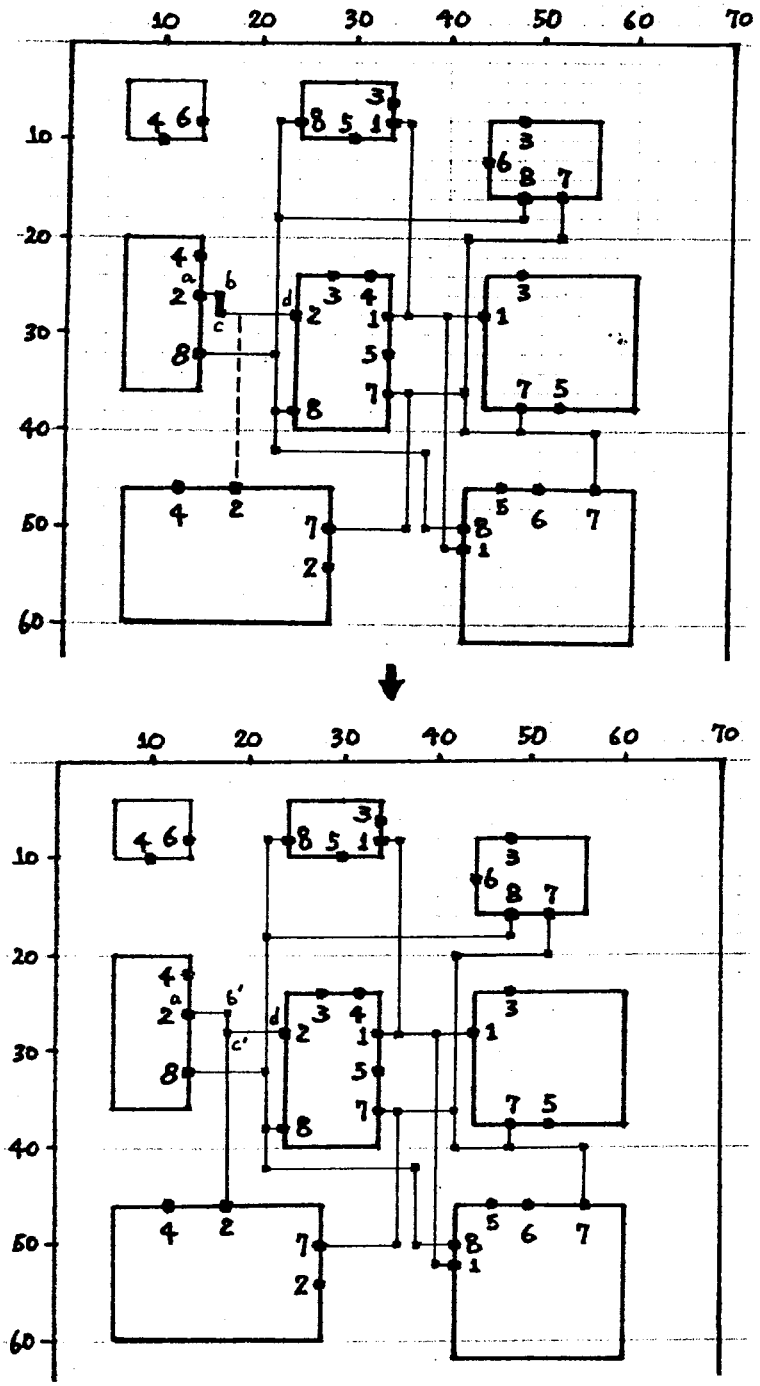


Figure 33

A type 2 defect was found when net 2 was routed. abcd is a Z wire. The dotted wire which joined the Z wire abcd made it a type-2 defect.

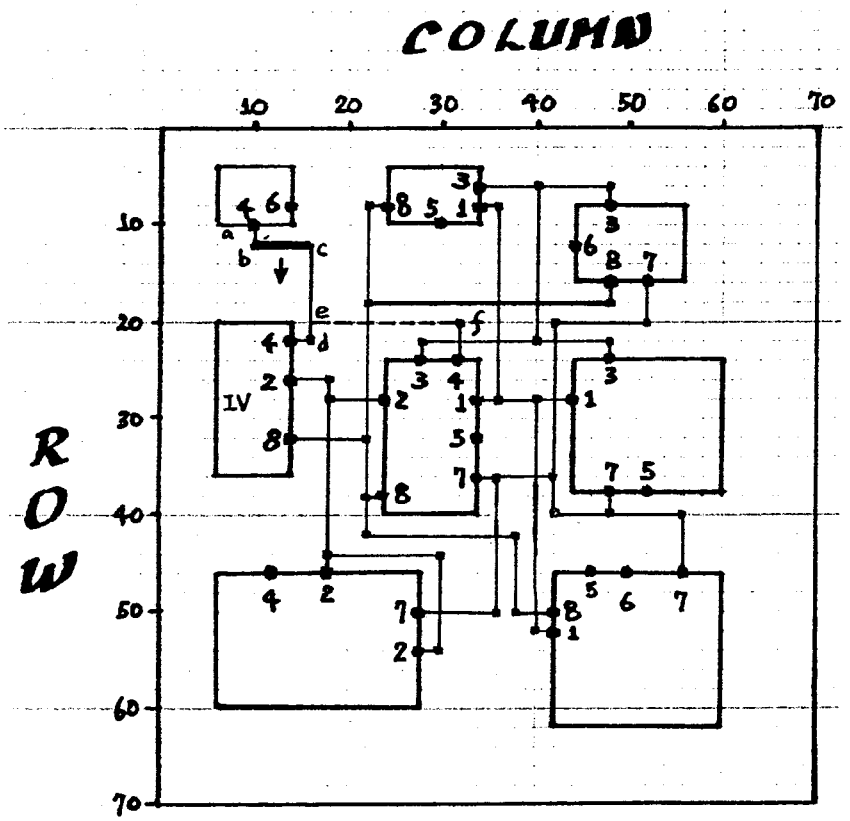


Figure 34

A type 2 defect was found when net 4 was routed. Wire bc should move down 4 track. But it could not since it was blocked by rectangle IV. abcd is the Z wire when wire ef is joined to it.

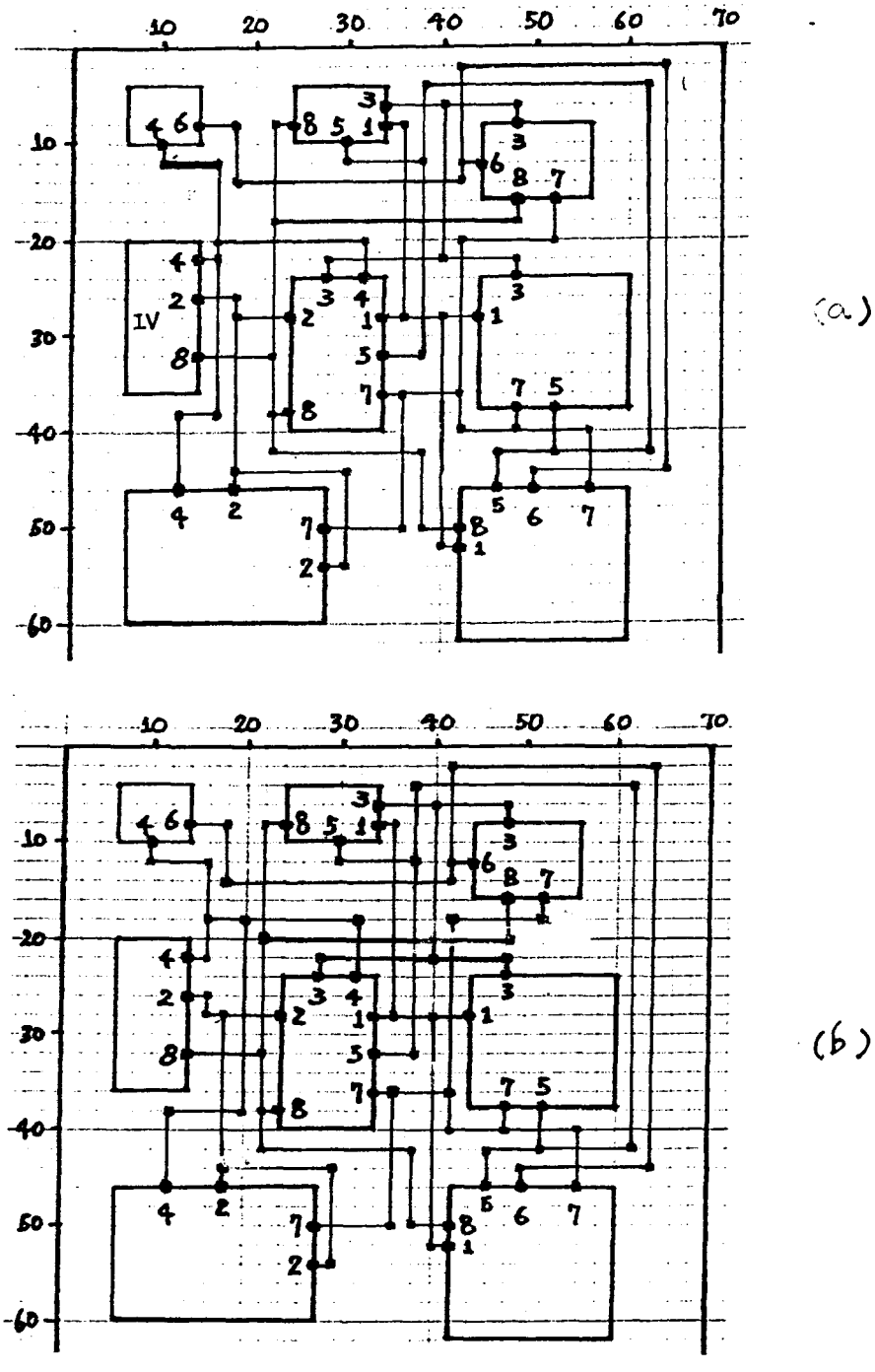


Figure 35

Finished ICs. (a) By ROUTER. (b) By ordinary Lee's algorithm

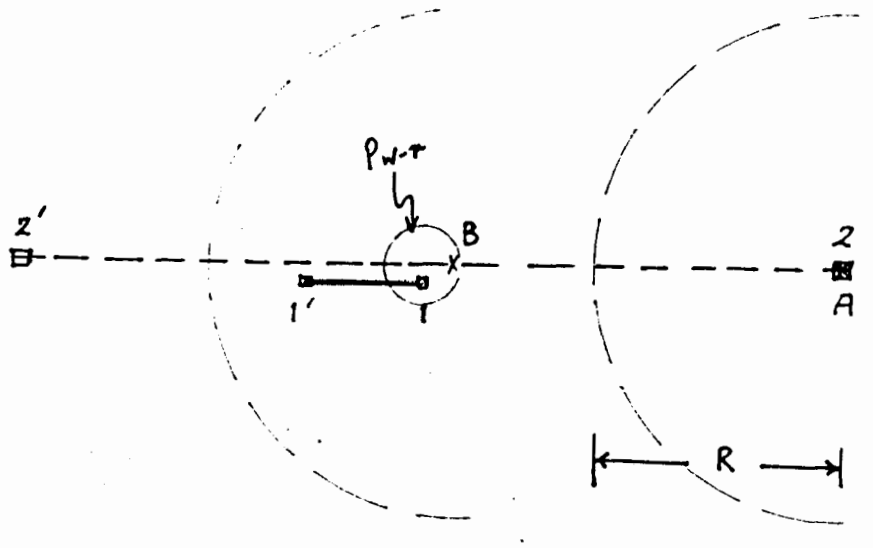


Figure 36

Eye movement of a linear scanning process

constraint violations, ROUTER asks the periphery if any violations occur. With its limited resolution, bubble Pw-r detects a possible violation-- the tentative wire may cross a via. Upon receiving the alarm, ROUTER directs the retina to fixate at point B so that the fovea can decide if the alarm should be honoured. Of course the tentative wire does not cross the via and the fovea can also tell that via 1' does not pose any problem. This process repeats until either no alarm is generated (which means the tentative wire does not violate the routing constraint) or wire 2-2' should not be laid.

Another example is illustrated in Figure 37. This time ROUTER is to find all the vias that lie on the likely paths.

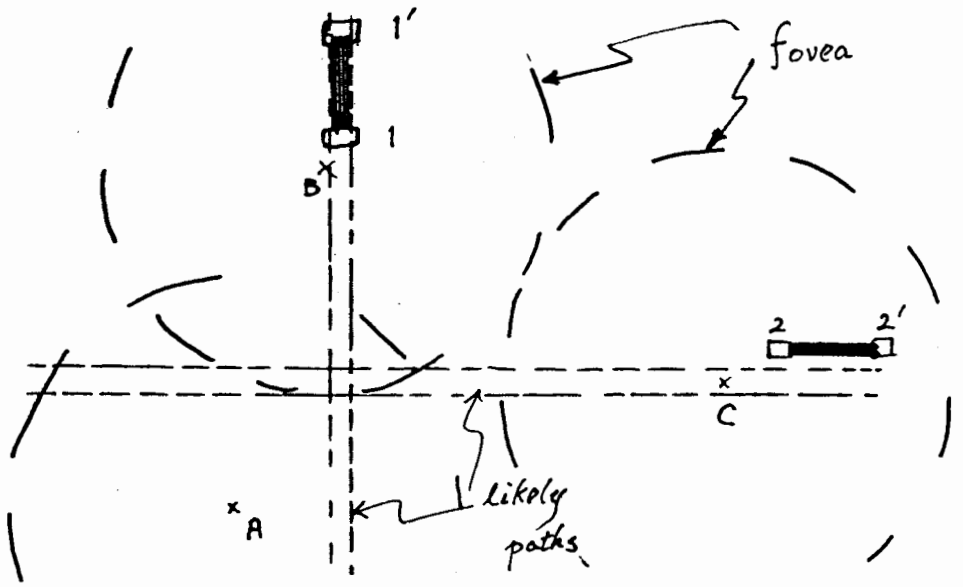


Figure 37

Eye movement protocol of the process of picking points of interest

Assume the retina is first fixated at point A where the fovea detects no violation. Then ROUTER asks the periphery if any violations occur. In this case, four peripheral bubbles detect vias near the likely paths. The position of the bubble that is closest to the fovea becomes the next fixation point. ROUTER finds that vias 1 and 1' both lie on the likely paths. Two bubbles in the periphery have detected possible via candidates and the retina is moved to focus on point C. This time the fovea can see that vias 2 and 2' do not lie on the likely paths.

The advantage of the two processes is that there is no need to search each cell in sequence in order to pick up points of interest. This advantage is more prominent in the

second example where a few fixations can replace a search through the whole area.

The eye movements in the first example seem to be analogous to the scanning process of a human being while the second example appears to be similar to the process of picking points of interest.

4.6 How colour can be applied to highlight areas of interest

There are many situations where special colours are needed in order to facilitate problem solving. For example, ROUTER may want to know if a specific track is free. The first thing ROUTER does is to paint some colour on that track in order to highlight it. But the time taken to sequentially paint each pixel on that track is equivalent to sequentially checking if each section of the track is free. So it seems that colouring the diagram may be a waste of time. However, there is another way to colour a diagram by using the parallel processing retina. This is how it works: The retinal supervisor broadcasts the area (by means of range) where it should be highlighted. Since each bubble can calculate its current location on the diagram, it will also be able to test if it is actually looking at the area specified by the supervisor. Therefore, those bubbles that are looking at the area of interest will mark in their memory cells the special colour assigned by the retinal supervisor for highlighting purposes. Remember, all the

calculations needed here are done in parallel. This method of "seeing things" is much faster than sequentially painting each pixel of interest.

4.7 Languages used

The languages used in ROUTER's implementation were FRANZ LISP and C. In the early phases of the project, FRANZ LISP was mainly used because of its interactive environment. As the project progressed, more and more functions were written in C. Although programs written in FRANZ LISP can be compiled, they are still inefficient compared to the same versions written in C. In particular, programs that were executed by the retina were mostly written in C for efficiency.

4.8 Complexity Analysis

The task of estimating the complexity of ROUTER's algorithms is complicated by the amount of pins and blocks a chip has. So only rough estimates will be given in this section. For the purpose of analysis, one addition is assumed to consume one unit u of cpu time. One multiplication/division takes 10u. Time taken to pass one byte of information from one to ones neighbour is assumed to be m units. Also we will assume the retina has a fixed number of rings r and a fixed number of wedges w. The fovea is assumed to have the same structure as the periphery.

Fixation will take f units to complete.

4.8.1 Scaling

On average, messages will need to be passed r/2 times for a scaling. Therefore, the cost for scaling is

mr/2 u

4.8.2 Rotation

Most rotations of the retinal image are by 90 degrees. Then the cost for rotation is

mw/4 u

4.8.3 Similarity

Two fixations are required for a similarity test (2f). To find a scaling factor we need approximately 2p additions (p stands for the number of pins in a net) and 3 divisions:

2p + 30 u

One scaling process is needed (mr/2). Three 90 degree rotations are required (3*(mw/4)). Assuming each bubble will only compare to its previous memory content and to its 8 nearest neighbours' contents, then the cost for comparison in parallel is

9c u

where c units of time is taken for each comparison. Now if 20% out of rw bubbles report a match, then the cost of passing messages to the retinal supervisor is

rw * 0.2m u

The retinal supervisor also needs time to calculate a score which takes

rw * 0.2 u

Therefore, the approximate cost of comparing net-a to net-b is

2f + 2p + 30 + mr/2 + 3mw/4 + 4(9c + 0.2rw(1+m))

Note that all the terms here can be regarded as constants except 2p. Therefore, the operation of similarity test is linear in the number of pins in the net.

4.8.4 Preprocessing

Let v be the number of vias already placed on a chip. So in the worst case, we need v fixations to discover if they lie on the likely paths of a net. However, using the method of picking points of interest, only a small percentage of vias will be picked up as if they are near the likely paths. Assuming on the average only 20% of vias are close to the likely paths, then the cost would become 0.2v u. if those vias cluster together, then additional saving will be achieved since one fixation may pick up a few of those vias.

4.8.5 Identifying Z wires

The retina could theoretically be used to identify Z wires. However, for simplicity, ROUTER uses a very simple method to recognize Z wires.

4.8.6 Lee's algorithm

The complexity analysis of Lee's algorithm and some of its variations can be found in Rubin's paper (Rubin (1974)).

4.8.7 Track-checker

As can be seen in the section on "Eye Movement Protocol," there is no need to sequentially search each pixel to see if it is already occupied. The number of fixations needed depends on the number of wires that cross the track and the number of vias that are near it. If the track of interest is k units long, then in the worst case, we need k/R fixations in order to scan the track. (R denotes the radius of the fovea.) Of course, the best case would be just one fixation.

5. Conclusion

ROUTER's success in problem solving depends on its abilities in extracting and interpreting information from diagrams. In particular, information that is visually obvious such as similarity, the Z wires, and the two defect types has been exploited. This thesis has shown that visually represented information is a suitable interface for a computer system equipped with a retina-like input device and some kind of redrawing mechanism.

The redrawing mechanism of the HLR is one of the important aspects of ROUTER that makes hypothesis testing possible. It allows wires to be drawn and later erased from the diagram. Without this capability, it would be very difficult to move wires around. The redrawing mechanism is analogous to human use of pencil, eraser and paper.

ROUTER's fovea has provided a solution to the singularity problem on the retina-like structure devices. It allows the whole retina (fovea and periphery) to work in unison. Scaling and rotation can be done by simply passing information between neighbours. The periphery can work as a global detector while the fovea can resolve any ambiguities that exist.

5.1 Future Research Direction

5.1.1 Construction of the parallel processing retina

The ultimate research effort is to build a prototype of the parallel processing retina. But before such a prototype can be built, many hardware design problems must first be worked out. In particular, problems concerning the actual representation of the diagram and the mechanism of filling the retina must be solved. Yet to be determined is the optimum number of bubbles needed in the retina. Problems concerning communication protocols and the message passing mechanism between the processors also need to be worked out in great detail.

5.1.2 A Harder Problem

It is not uncommon to see an automatic wire router get stuck and request help from a person. Usually, that person will look at the layout diagram representing the current state of the routing phase and will tell the router how to get around the problem. To use ROUTER to automate this phase successfully may prove to be a real challenge.

ROUTER is still at its infancy in solving automatic wire routing problems. To build a robust wire routing system based on ROUTER, we need to incorporate rip-up and reroute techniques, and be able to route wires on less congested areas. Nonetheless, ROUTER has demonstrated that a computer can use diagrams effectively in solving a

non-trivial problem, and that WHISPER's model is a particularly good approach to diagrammatic representation of information.

References

Akers, Sheldon B. (1972). "Routing," in : M. Breuer (Ed.), Design Automation of Digital Systems: Theory and Techniques. Vol. 1: Hardware, (Prentice Hall, Englewood Cliffs, 1972), pp. 283-333.

Braccini, C., G. Gambardella, G. Sandini and V. Tagliasco (1981). "Borrowing from the eyes to create robot vision algorithms," Sensor Rev. (GB), Vol. 1 (1981), No. 2, Pp. 68-72.

Chaikin, George M. (1981). "DRAWING MACHINES," La Jolla Institute Conference Proceedings 1981 Advanced Computer Concepts, pp. 57-65.

Ciesielski, M. J. and E. Kinnen (1982). "AN ANALYTICAL METHOD FOR COMPACTING ROUTING AREA IN INTEGRATED CIRCUITS," 19th Design Automation Conference (1982), pp.30-37.

Davis, Philip J. (1974). "VISUAL GEOMETRY, COMPUTER GRAPHICS AND THEOREMS OF PERCEIVED TYPE," Proceedings of Symposia in Applied Mathematics, 20(1974), pp. 113-127.

Dees, William A. and Patrick G. Karger (1982). "AUTOMATED RIP-UP AND REROUTE TECHNIQUES," 19th Design Automation Conference (1982), pp. 432-439.

Dees, W., K. Parmar, A. Goyal, R. Tsui, B. Rathi, and R. Smith, II (1981). "A computer-aided VLSI layout system," AFIPS Conference Proceedings 1981 National Computer Conference (Vol. 50), pp. 11-18.

Eastman, Charles M. (1973). "Automated Space Planning," Artificial Intelligence, 4(1973), pp. 41-64.

Eastman, Charles M. (1970). "Representations for Space Planning," Communications of the ACM, 13, No. 4 (1970), pp. 242-250.

Funt, Brian V. (1976). "A Computer Implementation Using Analogues in Reasoning," Tech. Report 76-9, Department of Computer Science, University of British Columbia (1976).

Funt, Brian V. (1980). "Problem Solving with Diagrammatic Representation," Artificial Intelligence, 13 (1980), pp. 201-230.

Garey, Michael R. and David S. Johnson (1979). Computers, Complexity, and Intractability: A Guide to the Theory of

NP-Completeness, (Mathematical Sciences Ser.), 1979, W H Freeman

Garey, Michael R. and David S. Johnson (1977). "The Rectilinear Steiner Tree Problem is NP-Complete," SIAM J. Appl. Math., 32, 1977, pp. 826-834.

Gelernter, H. (1963). "Realization of a Geometry-Theorem Proving Machine," in : E. A. Feigenbaum and J. Feldman (Eds.), Computer and Thought, (McGraw-Hill, New York, 1963) pp. 134-152.

Hashimoto, Akihiro and James Stevens (1971). "Wire Routing by Optimizing Channel Assignment Within Large Aperture," 8th D. A. Conference (1971), pp. 155-169.

Heinisch, J. (1981). "Aiming at a General Routing Strategy," 18th Design Automation Conference, (1981), pp. 668-675.

Lee, C. Y. (1961). "An Algorithm for Path Connections and Its Applications," IRE Trans. on Electronic Computers, Vol. EC-10, No. 3 (Sept 1961), pp. 346-365.

Preas, B. T. and W. M. vanCleemput (1979). "Placement Algorithm for Arbitrarily Shaped Blocks," Design Automation Conference, 1979, pp. 474-480.

Pylyshyn, Zenon W. (1976). "IMAGERY AND ARTIFICIAL INTELLIGENCE," Research Bulletin No. 376, University of Western Ontario (1976).

Pylyshyn, Zenon W. (1975). "Representation of Knowledge: Non-Linguistic Forms-- Do We Need Images and Analogues?" Theoretical Issues in Natural Language Processing. Ed. R. Schank and B. L. Nash-Webber. Conference Proceedings Cambridge, Mass., June 10-13, 1975, pp. 174-177.

Rubin, Frank (1974). "The Lee Path Connection Algorithm," IEEE Transactions on Computer, 23, No. 9 (1974), pp. 907-914.

Sandini, Giulio and Vincenzo Tagliasco (1980). "An Anthropomorphic Retina-like Structure for Scene Analysis," Computer Graphics and Image Processing, 14 (1980), pp. 365-372.

Schenker, P. S., E. G. Cande, K. M. Wong, and W. R. Patterson III (1981). "NEW SENSOR GEOMETRIES FOR IMAGE PROCESSING: COMPUTER VISION IN THE POLAR EXPONENTIAL GRID," Proceedings of the 1981 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1144-1148.

Strothotte, Thomas (1981). FAST RASTER GRAPHICS USING PARALLEL PROCESSING, Burnaby, B. C.: Simon Fraser University, M. Sc. Thesis, 1981.

Supowit, Kenneth J. (1982). "A Minimum-Impact Routing Algorithm," 19th Design Automation Conference (1982), pp. 104-112.

Sussman, Gerald Jay and Richard Matthew Stallman (1975). "Heuristic Techniques in Computer-Aided Circuit Analysis," IEEE Transactions on Circuits and Systems, November (1975), Pp. 857-865.

Sussman, Gerald Jay and Richard Matthew Stallman (1977). "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," Artificial Intelligence, 9, No. 2(1977), Pp. 135-196.

Sussman, Gerald Jay (1977). "ELECTRICAL DESIGN -- A PROBLEM FOR ARTIFICIAL INTELLIGENCE RESEARCH," 5th International Joint Conference on Artificial Intelligence, Aug. 22-25 (1977), Pp. 894-900.

The Editors of 73 Magazine (1982). THE GIANT BOOK OF ELECTRONICS PROJECTS, TAB BOOK Inc., 1982.

Weiman, Carl and George Chaikin (1977). "LOGARITHMIC SPIRAL GRIDS FOR IMAGE PROCESSING AND DISPLAY," Tech. Report 77-3, OLD DOMINION UNIVERSITY (1977).