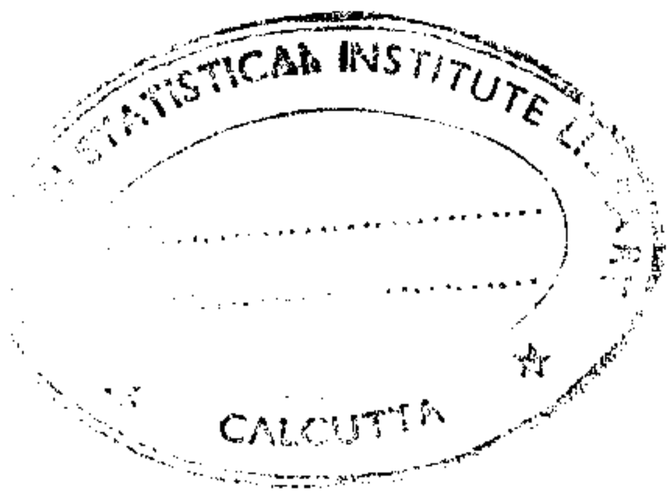
HEURISTIC ALGORITHMS FOR DETERMINING FEASIBLE ROUTING ORDER IN NONSLICIBLE FLOORPLANS

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE M.TECH.(COMPUTER SCIENCE) DEGREE OF THE INDIAN STATISTICAL INSTITUTE

BY

RAJAN GANGADHARAN



UNDER THE SUPERVISION OF

Dr. BHARGAB B. BHATTACHARYA

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisors, Dr.B.B.Bhattacharya and Mrs.Susmita Sur-Kolay for their helpful guidance, supervision and encouragement. I would also like to extend my gratitude to Mr.Subhas.C.Nandy for helping me whenever I turned to him.

I would like to acknowledge the help and support provided by my friends Ashutosh Kumar Jha,Jaydeep Das,Suresh M, Milind Kamble and M.Deviprasad.

G Rajan

Rajan Gangadharan

C O N T E N T S

ABSTRACT 1

1. INTRODUCTION

1.1 Introduction 2

1.2 Scope Of This Dissertation 4

2. DESCRIPTION OF CHANNEL ROUTING PROBLEM

2.1 Channel Graph And Its Properties 6

2.2 Maximal Rectangular Hierarchy 9

2.3 The Minimum Feedback Vertex Set Problem 12

3. IMPLEMENTATION OF HEURISTICS FOR THE MFVS PROBLEM

3.1 Floorplan generation 24

3.2 Channel Graph Extraction 25

3.3 Extraction of strongly connected components 30

3.4 Construction of Component Graph 32

3.5 Experimental Results 35

4. CONCLUSIONS 39

REFERENCES 44

HEURISTIC ALGORITHMS FOR DETERMINING FEASIBLE ROUTING ORDER IN NONSLICIBLE FLOORPLANS

ABSTRACT

In this dissertation we make an experimental study of the algorithms suggested in [SB91] for determining a feasible channel routing order in nonslicible VLSI floorplans by reserving channels. Since the problem is NP-complete, a comparative study of the different heuristics based on time complexity and quality of the solution has been done.

KEYWORDS :

VLSI layout design, Floorplanning, Channel routing, Slicing structures, Algorithms, NP-completeness, Nonslicible floorplans.

CHAPTER 1

INTRODUCTION

1.1 Introduction

In recent years, advances in VLSI (Very Large scale Integration) technology have overwhelmingly influenced every sphere of life where electronics is being used. These tiny chips encapsulate hundreds of thousands of electronic components with remarkable miniaturization and are realized as geometrical patterns in layers of different materials deposited on monolithic crystalline semiconductor wafers.

The major categories of representing these chips are behavioral, structural and physical. Behavioral representations describe the function of the circuit. Structural representations are characterized by the composition of circuits in terms of primitive circuit components and their interconnections. Physical representations provide information needed from the viewpoint of fabricating the VLSI chip.

The design of a chip [PL88] consists of the following phases.

a. Design specification.

b. Functional design.

c. Logic design.

d. Circuit design.

e. Physical design.

f. Fabrication on semiconductor chips.

Layout design is essentially the phase of physical design when behavorial and structural representations of an electronic system are translated to geometric shapes to be used during fabrication. The cost and performance (area,speed) of VLSI chips are influenced by the quality of the layout. Also the size and compexity of the electronic circuits in the VLSI chips have neccesitated the use of computer automation for layout design. One of the most successful design techniques for layout design is the hierarchical layout design.

In hierarchical layout design , a hierachical subdivision of the physical structure of the circuit is first created by the partitioning process. The next important step is to obtain a floorplan of the chip, given a set of functional modules and their topological neighborhood relation based on the interconnections in the logic design of the circuit. The problem of allocation of positions to these modules on the chip is known as floorplanning. Based on the floorplan guidelines the later phases of actual placement and wire interconnection are carried out.

Most of the existing literature deal with a special type of floorplans known as slicings or slicible floorplans[OT83,LGW83]. The preference for slicings is for their computational ease. It is possible to represent a slicing in a number of concise ways which facilitate recursive divide and conquer processing on the floorplan. It is known that constructing a slicing from a slicing tree [OT82] as well as optimal wiring of a single net in a

slicible floorplan with minimum overall area can be done in polynomial time[LSW87]. Optimal orientation of modules is solvable in polynomial time for slicings, but is NP-complete for general floorplans [ST83]. The more general occurrence of floorplans are the nonslicible ones. The geometric and graph theoretic attributes of nonslicing patterns are relatively unexplored to date.

1.2 Scope of this dissertation

The main objective of this dissertation is the implementation of MFVS [Minimum Feedback Vertex Set] algorithm as suggested in [SB91], for determination of feasible channel routing order for nonslicible floorplans. Various heuristics have been proposed for this purpose.

CHAPTER 2

DESCRIPTION OF CHANNEL ROUTING PROBLEM

The main objective of this dissertation is to determine a feasible routing order for nonslicible floorplans. It is necessary to determine an order in which channels are to be routed during the channel routing step, so that successful termination occurs. The order depends entirely on the topological structure of channels in the floorplan [PV79],[SS83], [KA83]. The assumptions are :

a. We consider a rectangular floorplan with rectangular modules and horizontal or vertical cutlines as straight channels with two channels forming only T-junctions.

b. If two channels form a T-junction, then the channel having one of its endpoints at the T-junction is known as the base and the other channel is the crosspiece.

It is evident that for channel routing, the base of a T-junction should be routed before the crosspiece. This type of dependency among the channels is usually depicted by a digraph called the channel graph. It was proved in [SS83] that the channel graph for any floorplan is acyclic if and only if it is slicible. Thus determining feasible routing order in slicible floorplans is equivalent to topological traversal of an acyclic digraph. The problem of feasible routing order crops up in a nonslicible floorplan. It was suggested in [PV79], that cyclic dependency among a set of channels forming a cycle can be broken

by reserving any one channel, without loss of generality as the last in the routing order. The width of this reserved channel is estimated generously thereby enabling successful routing of the other channels in the cyclic dependency.

2.1 Channel Graph and its properties

Before going through the properties of the channel graph, we recapitulate a few important definitions.

A floorplan is a rectangular dissection of an enveloping rectangle by isothetic line segments, called cuts, into a finite number of indivisible non-overlapping rectangles which correspond to the functional modules in the VLSI floorplan. A floorplan is slicible or a slicing, if its rectangular dissection can be obtained by recursively dividing rectangles into smaller rectangles until each non-overlapping rectangle is indivisible corresponding to a module. A floorplan with no through-cuts at some stage of dissection is called a nonslicible floorplan.

The channel graph is a digraph C = (V,E), where there is a distinct vertex in V for each channel and there is an arc (i,j) from i to j in E if and only if there is a T-junction of which i is the base and j the crosspiece (Fig 2.1). For a given floorplan, the channel graph is unique. The following two theorems were proved in [SS83].

Theorem 2.1 (4-Cycle Theorem) :

A channel graph has a directed cycle if and only if it has a directed cycle of length 4 [Fig 2.2].

Theorem 2.2 (Slicing Theorem) :

A channel graph is acyclic if and only if the floorplan has a slicible floorplan [Fig 2.1].

The important properties of a channel graph are :

a. It is connected.

b. It is planar, but not grid embeddable.

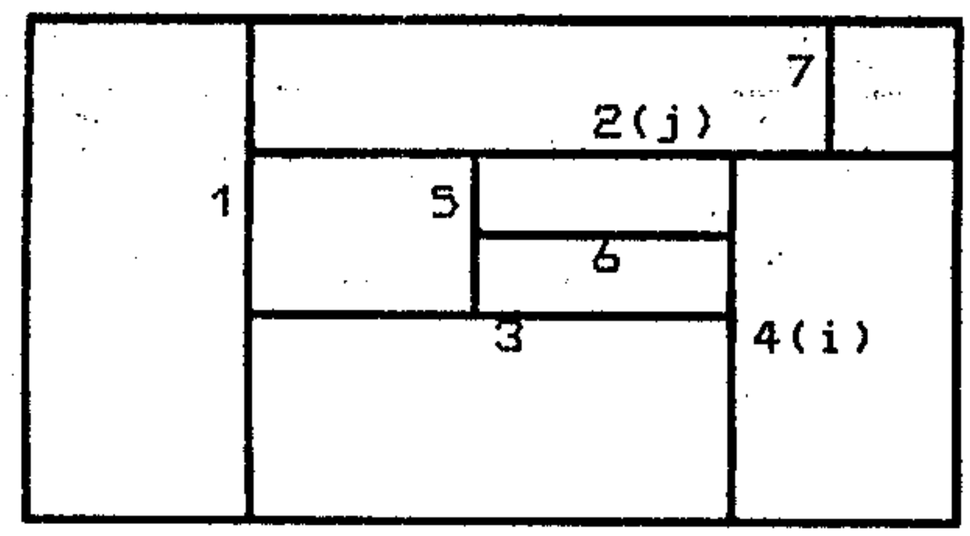
c. It is bipartite.

d. The outdegree of any vertex is atmost 2.

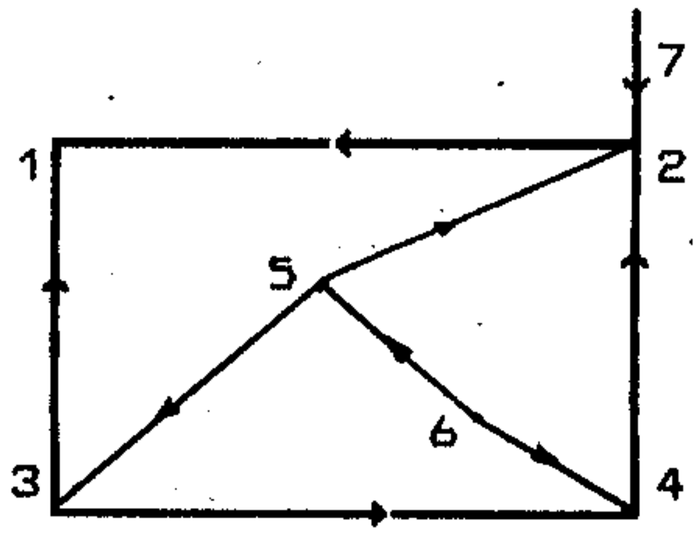
e. The indegree can be any non-negative integer.

Every internal cut (i.e neither of its two endpoints lie on a bounding rectangle) has outdegree 2. For through-cuts in the floorplan, the corresponding vertex in C has outdegree 0. Since C is bipartite, all cycles are of even length and minimum length is 4. Two adjacent 4-cycles cannot share more than one edge [Fig 2.2].

The seminal 4-cycle theorem on channel graphs proved in [SS83] says that the channel graph of a nonslicible floorplan has a directed cycle of length 4. In order to determine a feasible routing order of channels, some method to break cycles is required and this necessitates finding MFVS(Minimum feedback vertex set) efficiently and determining a routing order based on that set of vertices.

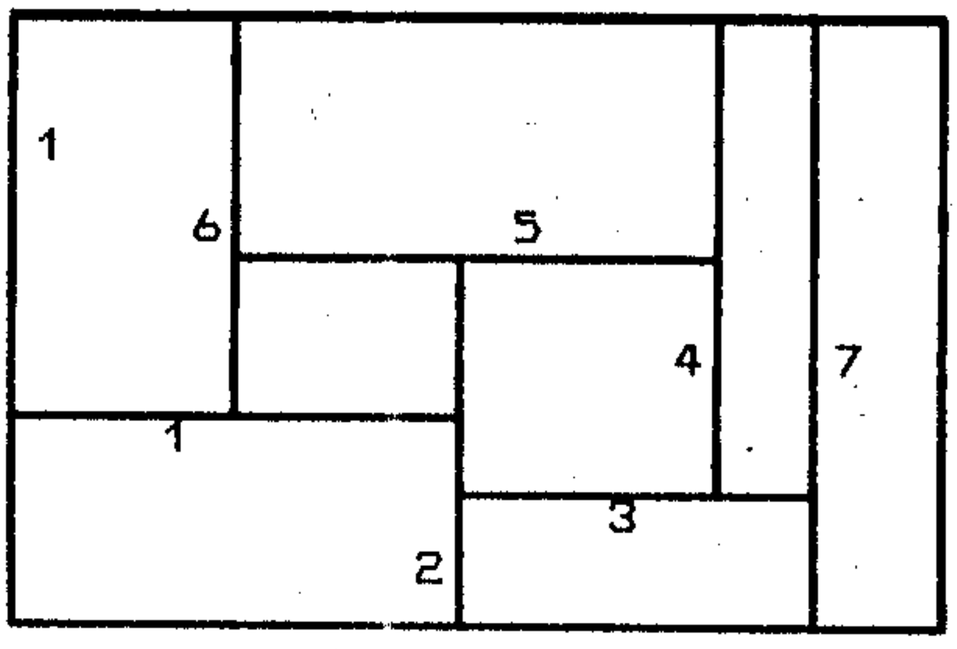


Floorplan

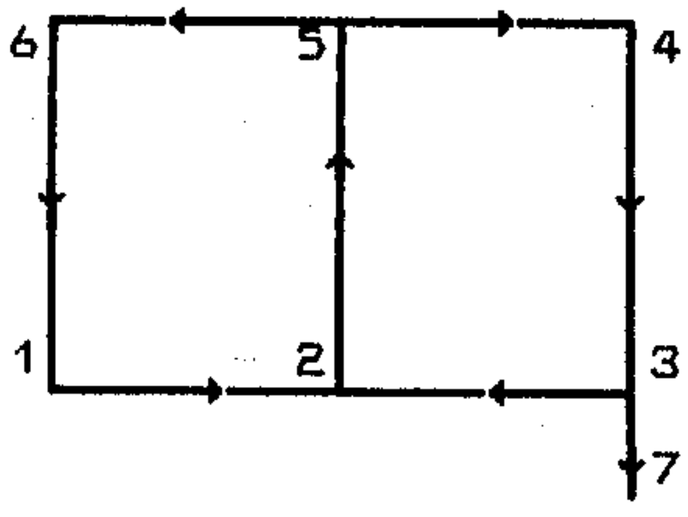


Channel Graph

Fig 2.1



Floorplan



Channel Graph

Fig 2.2

2.2 Maximal Rectangular Hierarchy

A floorplan can be hierarchically decomposed in order to facilitate divide-and-conquer. This is also a generalized representation of a layout which is not necessarily a slicing structure. For slicible floorplans several schemes such as slicing trees, series-parallel graphs are used. The general decomposition is based on the notion of maximal rectangles. In a floorplan, a maximal rectangle can be defined as one which is not contained in any rectangle in the floorplan other than the enveloping one. For a nonslicible floorplan we have the following lemmas.

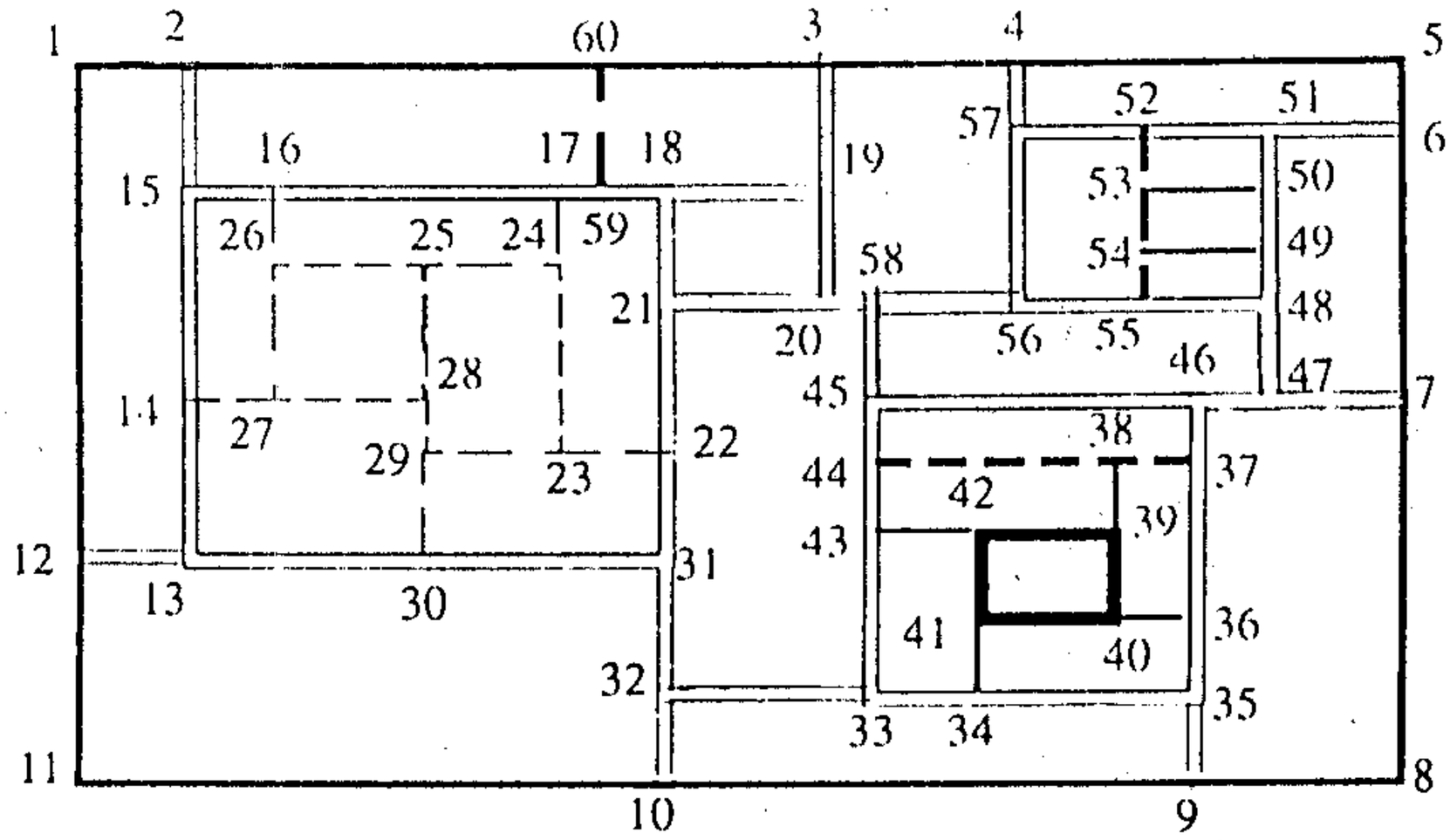
Lemma 2.1 : A non slicible floorplan can be decomposed into a nonempty set of mutually exclusive (non-overlapping) and collectively exhaustive maximal rectangles.

Maximal rectangles can be defined recursively as illustrated by heavy lines in Fig 2.3 to produce a hierarchy of maximal rectangles.

Lemma 2.2 : Given a floorplan, the vertices in its channel graph which correspond to the set of cuts defining the maximal rectangles at any level of the MRH (Maximal rectangular hierarchy), form a strongly connected component of the channel graph.

Since all the cycles of the channel graph appear exactly in one of the strongly connected components, one can process members of the MRH in parallel [Fig 2.4].

(a)



b) Rectangles at Level 0 : 1-5-8-11,

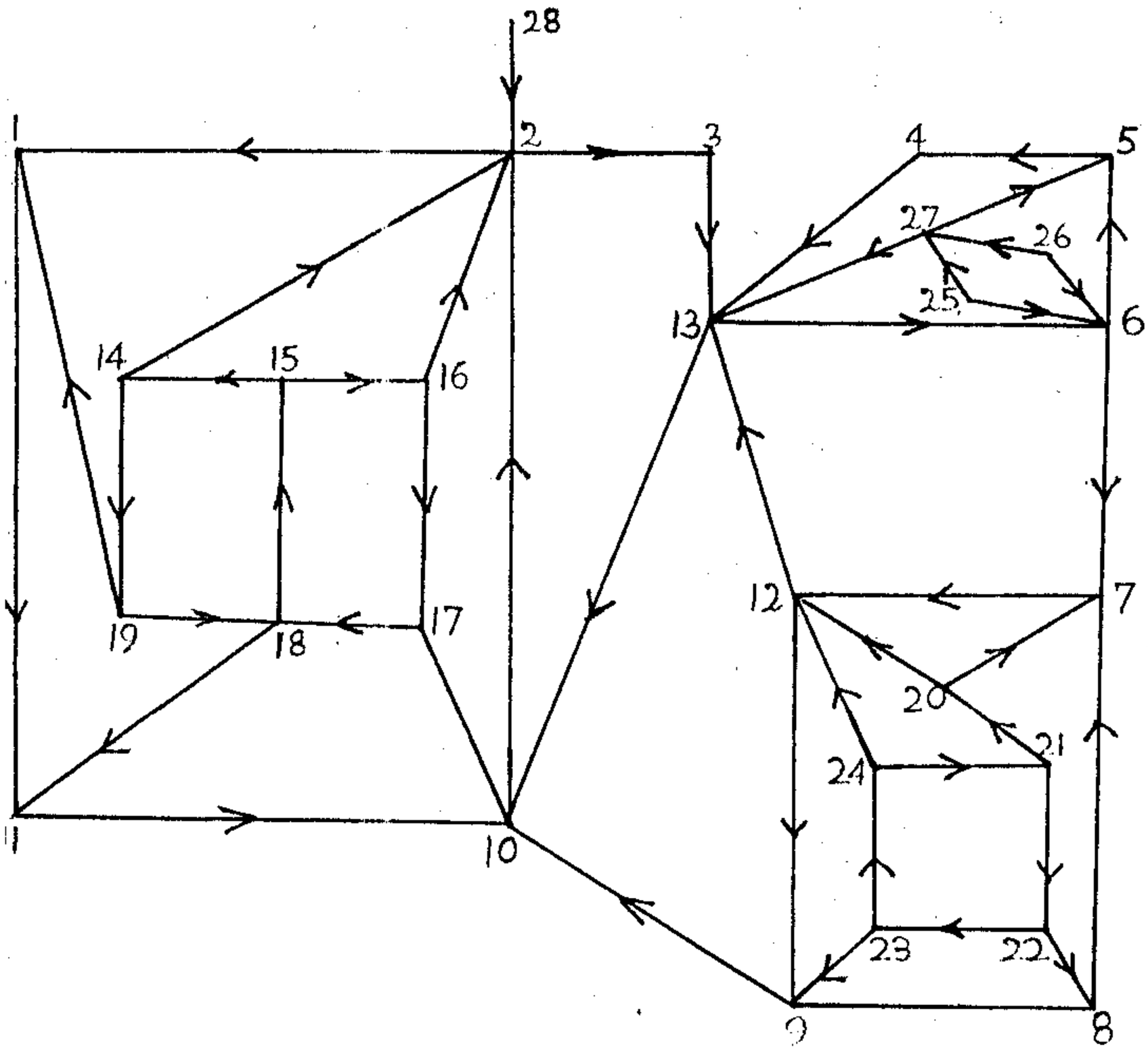
Rectangles at Level 1 : 1-2-13-12,2-3-19-15,3-4-56-58,
4-5-6-57,57-51-48-56,6-51-47-7,7-46-9-8,10-32-35-9,
11-12-31-10,13-15-18-31,18-19-58-21,21-58-33-32,33-45-46-35,
45-58-48-47

Rectangles at Level 2 : 2-60-59-15,60-3-19-59,57-52-55-56,
52-51-48-55,16-27-14-15,16-17-24-26,17-18-22-23,30-29-22-31,
13-14-28-30,27-26-25-28,29-25-24-23,45-46-37-44,44-37-35-33

Rectangles at Level 3 : 52-51-50-53,53-50-49-54,54-49-48-55,
44-38-39-43,38-37-36-40,41-36-35-34,43-42-34-33,42-39-40-41

Fig 2.3 : a) Maximal rectangles in a floorplan

b) Maximal Rectangular Hierarchy



) S.C.C at Level 1 {1,2,3,4,5,6,9,8,10,11,12,13}

Level 2 {14,15,16,17,18,19},{20},{27},{28}

Level 3 {21,22,23,24},{25},{26}

Fig 2.4 a) Channel Graph of floorplan in fig 2.3

b) Strongly connected components

2.3 The Minimum Feedback Vertex Set (MFVS) Problem

Given a digraph G = (V,E), the objective of MFVS problem is to find a subset V' of V such that V' contains at least one vertex from every directed cycle in G and |V'| is minimum [GJ79]. This problem was proved to be NP-complete in general, by using polynomial time transformation from the Vertex-Cover problem. MFVS remains NP-complete for the class of channel graphs as proved in [CW90]. So the main aim is to design an efficient heuristic algorithm for finding MFVS in a channel graph.

The concept of maximal rectangular hierarchy is used for finding MFVS. Since any directed cycle necessarily belongs to only one strongly connected component (S.C.C) of a digraph , the MFVS for each S.C.C can be solved independently.

To solve the MFVS the following two cases are considered :

Case 1 : Every vertex on any long cycle also lies on some 4-cycle.

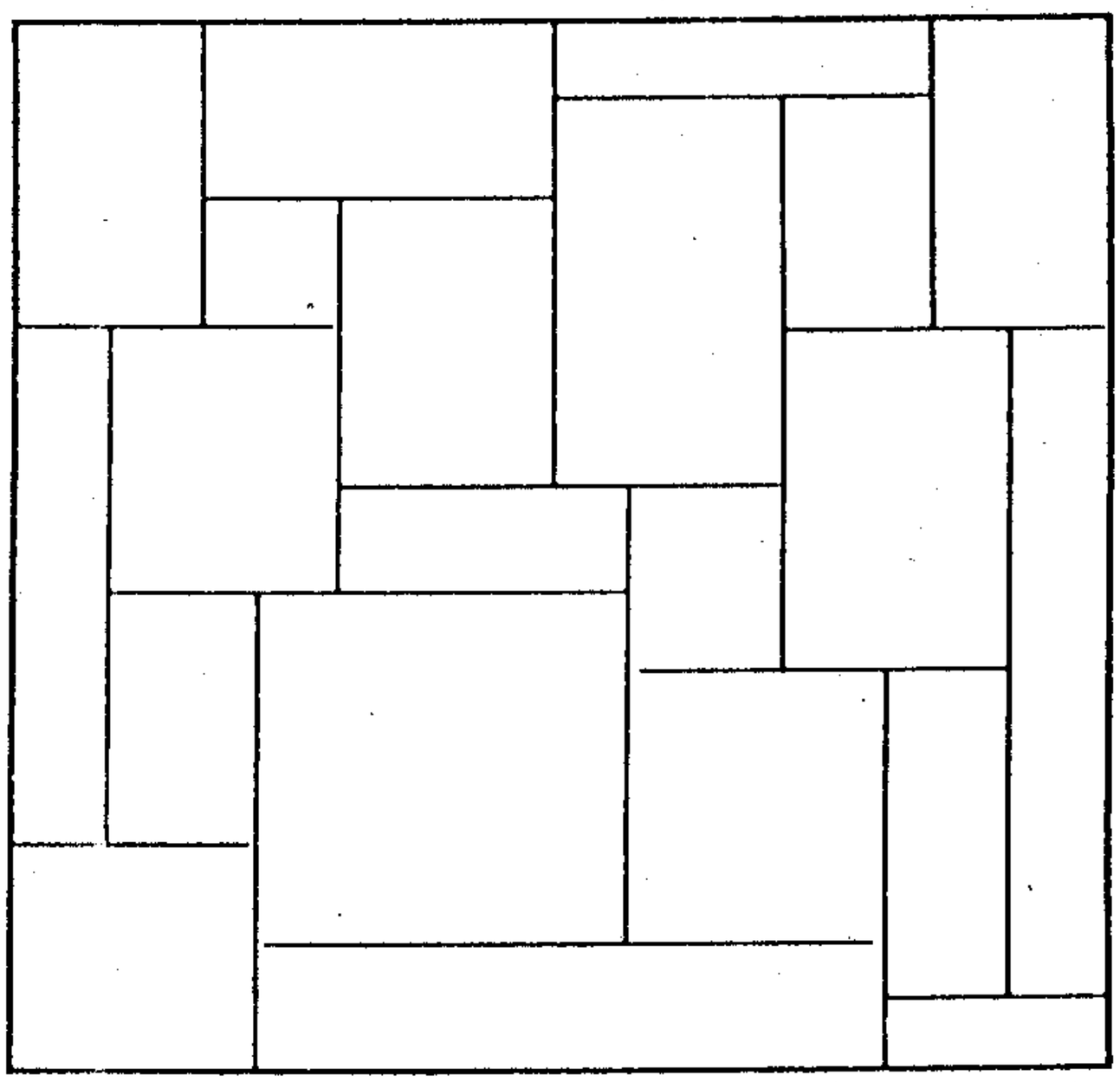
Case 2 : There exist some vertices lying on long cycles which do not appear on any 4-cycle.

The fact observed in the following theorem justifies the need for an efficient heursitic for solving the MFVS problem.

Theorem 2.3 :

a. The number of directed cycles of length 4 in a floorplan with n blocks is atmost O(n) [Fig 2.5].

b. The total number of directed cycles in a floorplan with n blocks is $O(2^n)$ in the worst case [Fig 2.6].



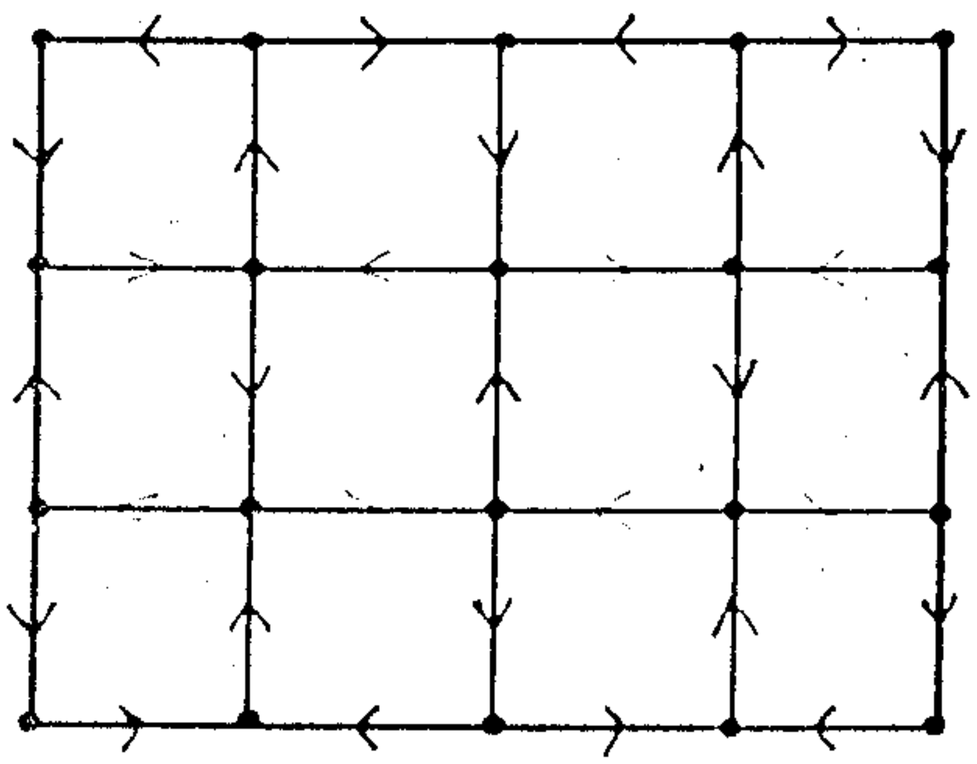


Fig 2.5 A floorplan with n blocks and its channel graph with O(n) 4-cycles

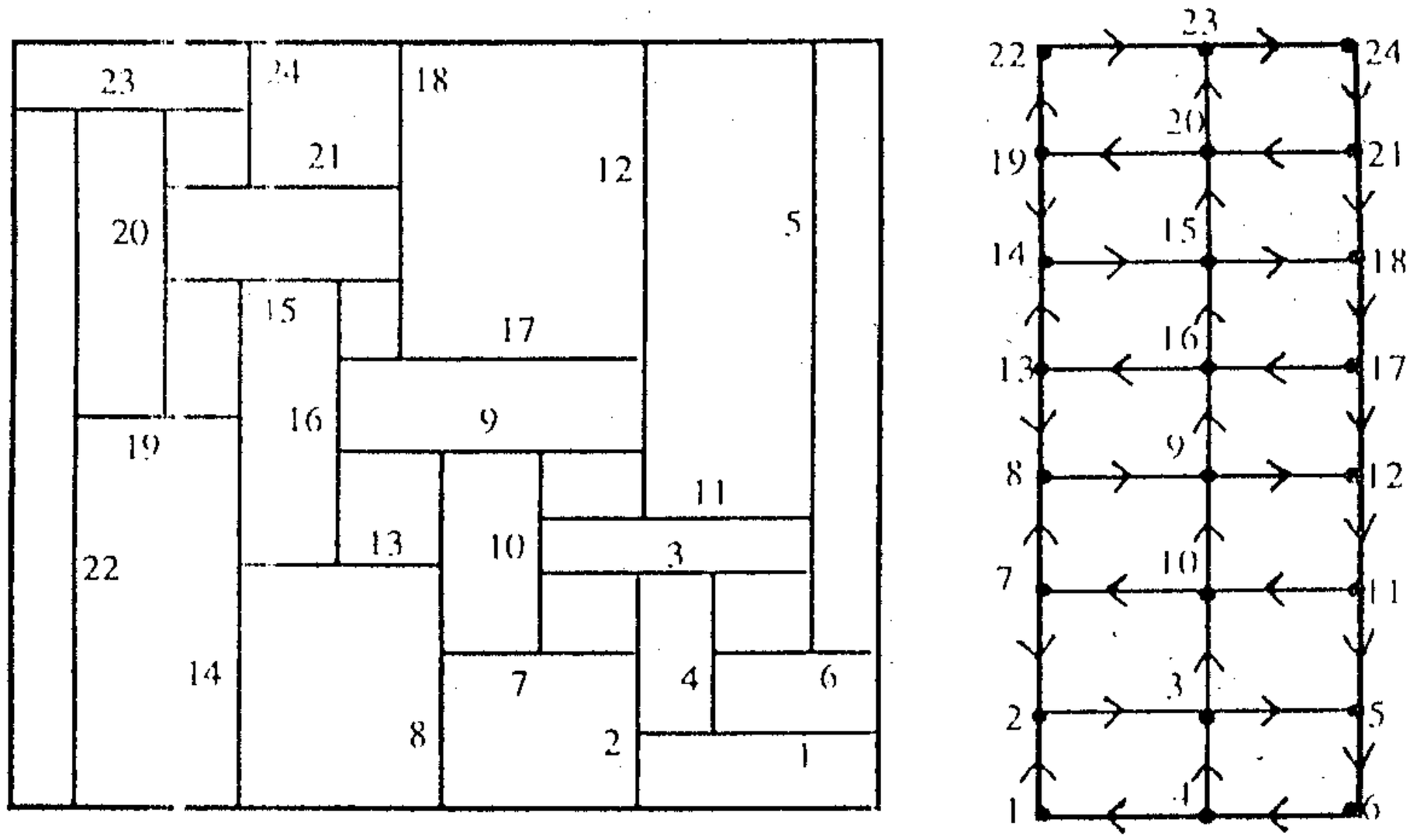


Fig 2.6 A floorplan with n modules and its channel graph with $O(2^n)$ directed cycles

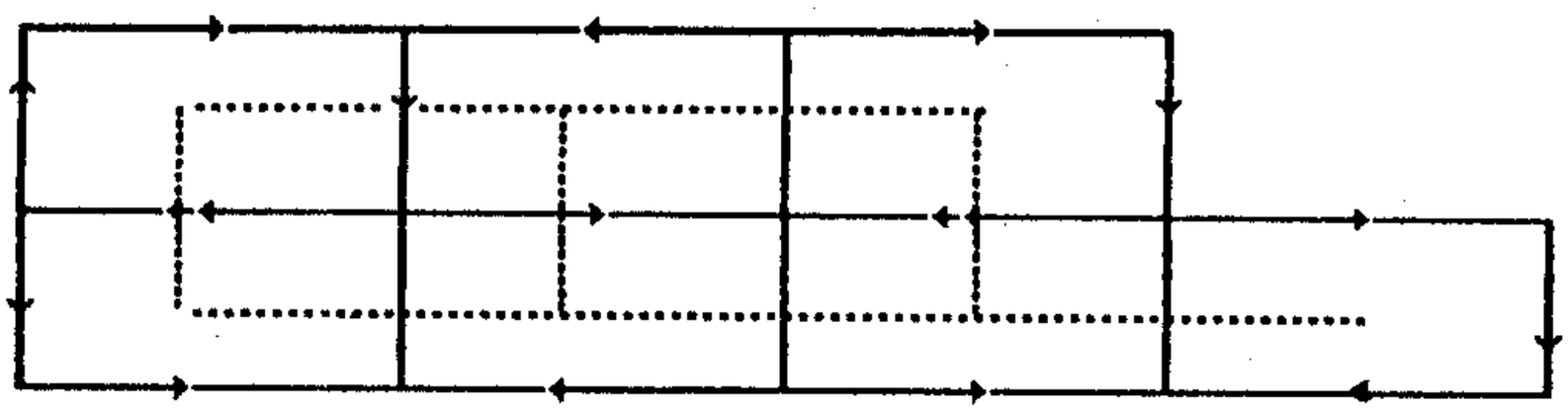


Fig 2.7 Channel graph and its facial-cycle-dual

Case 1 : By theorem 2.3(a) the number of directed cycles is O(n).

Definition 2.1 :

For a planar digraph G = (V,A) its facial-cycle-dual C_d is constructed by placing a vertex in each face of G which is a directed 4-cycle and by placing an edge between two vertices if the corresponding facial 4-cycles of G share at least one vertex [Fig 2.7].

Definition 2.2 :

A clique of a graph is a maximal complete subgraph and the clique cover of a graph is a subset of cliques such that every vertex of the graph belongs to at least one of the cliques in the subset.

The minimum clique cover of C_d is found by a greedy method based on the classical cover table approach [MC87], with the cliques on the vertical axis and vertices on the horizontal axis. First, all essential cliques are selected, then the table is reduced by using row dominance followed by column dominance. If there are no essentials, then ties are broken arbitrarily.

Each k-clique of C_d corresponds to a unique vertex in C, through which k directed 4-cycles pass. Thus the vertices of C corresponding to the minimum clique cover of C_d belong to the MFVS of C. For channel graphs, the size of a maximum clique in C_d can be at most 4. Thus the restricted case of finding 4-cliques or smaller can be done in linear time by keeping count of the number of 4-cycles passing through each vertex in C.

The MFVS can be obtained for case 1 by the following heuristic :

Heuristic 1

Step 1 : Construct C_d, the facial-cycle-dual graph of C.

Step 2 : Find minimum clique cover of C_d using Cover Table approach.

Step 3 : MFVS = Null set. For every clique C in the minimum clique cover, do

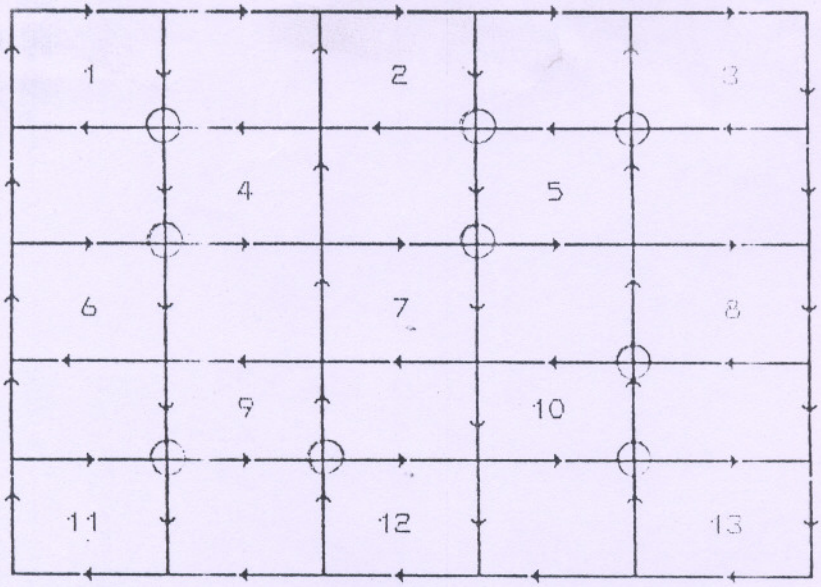
```
Begin
    V <- vertex in C corresponding to clique C.
    if V U MFVS has a directed 4-cycle then
       MFVS <- MFVS  U { Neighbor of V within cycle in C
                         which is covered by C only }
    else
       MFVS <- MFVS U V.
    Delete the new vertex added to MFVS from C along its
     arcs.
    Delete one by one any vertex in C whose
    in or out-degree = 0.
End
```

Theorem 2.4 :

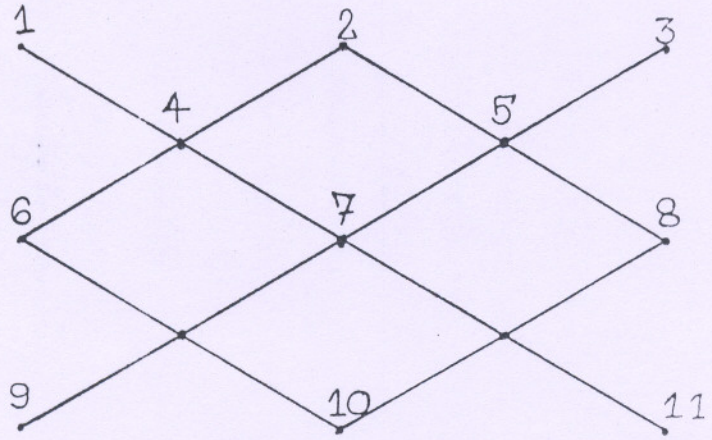
The above algorithm converges taking polynomial time. It produces an optimal solution if removal of MFVS solution leaves the original graph acyclic.

Proof : The complexity of step 1 is O(n). For minimum clique cover, the number of iterations in step 2 for selecting essentials and reducing dominance is as many as the worst case size of the minimum clique cover i.e O(n). Due to the special nature of C_d, each row and each column of the cover table has at most 4 entries. Such a sparse table can be represented by linked lists. During each iteration, row dominance is found by comparing a clique with those of larger size and sharing at least one vertex, thus only 9 row comparisions are sufficient. Since there can be O(n) number of cliques each iteration requires O(n) time. Lastly, step 3 is once again related to size of minimum clique cover and also vertices of C are removed one by one; hence it needs O(n) time. Therefore, the entire algorithm runs in $O(n^2)$ time.

Example : The graph in Fig 2.8(b) is the facial-cycle-dual of the channel graph in Fig 2.8(a). The facial-cycle-dual is constructed in step 1 of the heuristic. The cliques obtained n step 2 are shown in Fig 2.8(c). The minimum clique cover has nine cliques as shown in Fig 2.8(c). In nine successive iterations of step 3, a clique from the minimum clique cover is chosen. But still after running the above algorithm a long cycle along the boundary remains. Then another pass of constructing the corresponding facial-cycle-dual and determining clique cover is required and so on, whereas the solution obtained is no longer minimum.



(a)



(b)

Cliques : {1,4},{2,4},{2,5},{3,5},{4,6},{4,7},{5,7},{5,8},
{6,9},{7,9},{7,10},{8,10},{9,11},{9,12},{10,12},{10,13}

Min Clique cover : {{1,4},{2,5},{3,5},{4,6},{5,7},{8,10},
{9,11},{9,12},{10,13}}

Fig 2.8 A long cycle remains after removing MFVS obtained in Heuristic 1.

Case 2 :

In this case, the MFVS may include vertices through which no directed 4-cycle passes, but only longer cycles pass.

Example : In Fig 2.9, MFVS = {6,14,21} and there are no 4-cycles through vertex 14. Now by theorem 2.3(b), number of cycles can be $O(2^n)$. This calls for a heuristic algorithm to determine MFVS.

The first step of MFVS is to detect clusters of adjacent 4-cycles and to apply algorithm of case 1 on them. Isolated 4-cycles are however left alone, because with respect to this cycle any of the four vertices may be chosen. But if there is a longer cycle sharing a few vertices with this isolated 4-cycle, then for optimality, one of the shared vertices must be chosen from the 4-cycle. Since longer cycles have not yet been considered, a correct choice cannot be guaranteed. Having removed the clusters of 4-cycles from the channel graph C, all pendant vertices (in-degree or out-degree = 0) are removed. For choosing the MFVS from the bigger cycles, two heuristic algorithms are mentioned below.

Heuristic 2 :

Step 1 : Locate clusters of directed 4-cycles (not isolated ones) and solve them with heuristic 1.

Step 2 : Find any cycle using breadth first search.

Step 3 : Select vertex on this cycle having maximum product of in-degree and out-degree.

Step 4 : Remove from C the vertex added to MFVS most recently and delete one by one all vertices which effectively have either

indegree or outdegree = 0.

Step 5 : Repeat from step 2 until no more vertices remain or if only isolated 4-cycles remain in which case obtain the MFVS naively.

Heuristic 3 :

Replace step 3 (i.e the criterion for selecting the vertex to be included in the MFVS solution) of Heuristic 2.

Step 3' : Compute the weight for each vertex on the cycle, where weight of a vertex equals the number of arcs removed as the result of deleting that vertex and select the vertex with the maximum weight.

The time complexity of heuristic 2 is once again $O(n^2)$ because the worst case time for steps 2 and 3 are each O(n). Heuristic 3 has complexity of an order higher than that of heuristic 2 since the computation of weights is quite involved. It depends on weights of some of its successors and predecessors and thus step 3' itself is of $O(n^2)$.

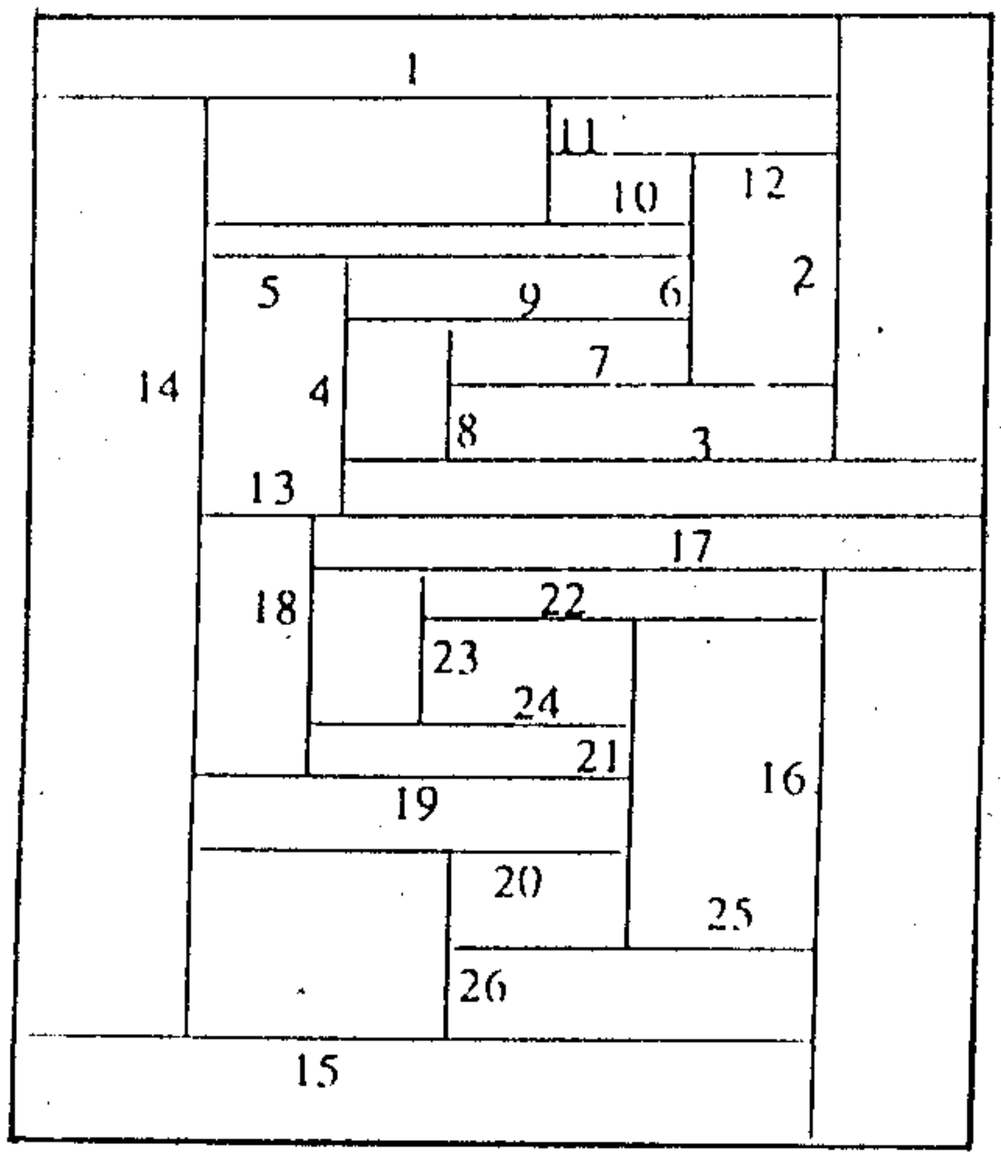
Example : For the channel graph in Fig 2.10(a), after step 1 of Heuristic 2 the MFVS is still a null set. In step 2 suppose the cycle traversed is {10-1-2-3-4-5}. Vertices 10,1 and 3 have the same product of indegree and outdegree. So let 1 be added to the MFVS.The modified channel graph is shown in Fig 2.10(b) after step 3. In the next iteration, let the cycle chosen be {3-4-5-6}. Let 3 be added to MFVS. The modified channel graph is shown in Fig 2.10(c). In the next iteration the cycle {9-10-11-12-13-14} is chosen and the vertex added to MFVS is 9. So the MFVS will be

{1,3,9} by Heuristic 2.

For heuristic 3, in the first iteration in step 3', vertex 3 is chosen since the weight associated to it is more than any of the other vertices in the cycle (i.e w(1) = 8, w(10) = 4, w(3) = 9). The modified channel graph is shown in Fig 2.10(d). In the second iteration, suppose the cycle chosen is {1-2-7-8-9-10}; the vertex added to MFVS will be 10. Thus the MFVS will be {3,10} by Heuristic 3.

Details of the experimental results of the above three heuristics are given in the next chapter.

(a)



(b)

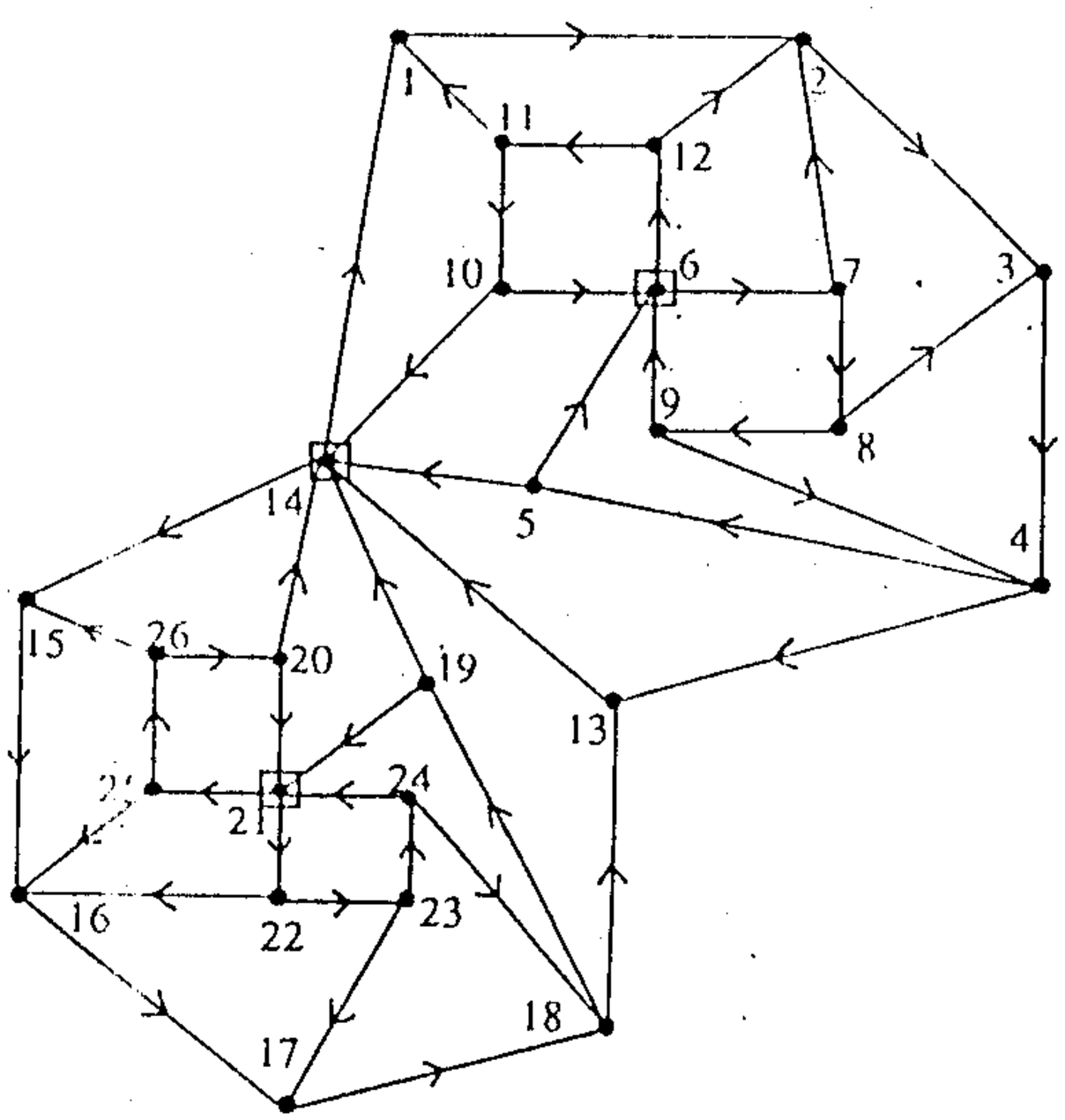


Fig 2.9 a) A Floorplan and b) its channel graph having some directed cycles none of whose vertices are present on any 4-cycle; MFVS = {6,14,21}.

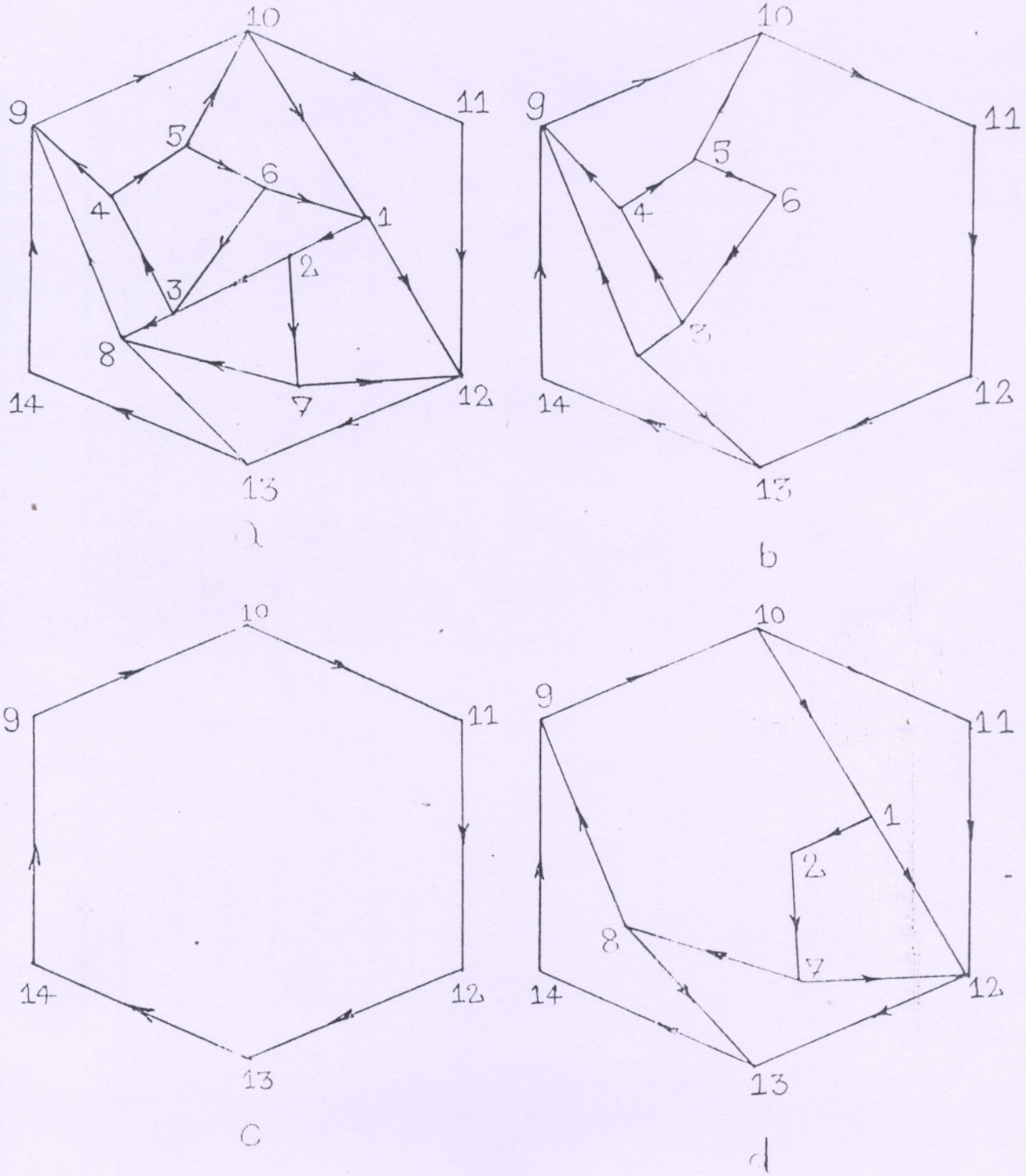

10
9
11
5
6
4
1
2
3
8
7
14
12
13
a
10
9
11
5
6
4
3
14
12
13
b
10
9
11
14
12
13
c
10
9
11
1
2
8
7
14
12
13
d

Fig 2.10

CHAPTER 3

IMPLEMENTATION OF HEURISTICS FOR THE MFVS PROBLEM

To judge the performance of the proposed heuristics,we have made an experimental study. This involves the following steps :

1) Random floorplan generation.
2) Channel graph extraction.
3) Extraction of strongly connected components of the channel graph.
4) Implementation of the heuristic algorithms and determination of MFVS.

3.1 Floorplan generation

Sample floorplans are generated randomly. The user specifies height, width, number of modules on the chip and a random seed. The left end point and length of horizontal intervals are two random variables which are generated randomly, the number depending on the number of blocks specified for the floorplan. Then for each horizontal interval, a vertical line is extended from both its endpoints till another horizontal interval or the boundary intercepts it (Fig 3.1). An endpoint of a vertical line lies on the first horizontal interval blocking it if the abcissa of the vertical line lies strictly between the two endpoints of the horizontal interval. This ensures that two perpendicular lines always meet at a T-junction. A vertical line

can be extended to the boundary if there are no horizontal lines blocking it. If a horizontal interval has two horizontal intervals b and c below it respectively so that the abcissa of one endpoint of b and one endpoint of c are same and the two vertical lines from these two endpoints are blocked by a, then a cross junction occurs. This is taken care of by adjusting one of the horizontal intervals towards the right if the vertical lines are drawn from left to right.

3.2 Channel Graph Extraction

The floorplan generated can be considered to be a grid graph with T-junctions and the four corners as the grid points or vertices. The channel graph is extracted by depth-first traversal. A description of the algorithm used is given below.

INPUT : A set of 5-tuples (one 5-tuple corresponding to each grid point) wherein each 5-tuple contains the following information

a. Address of the grid point.

b. The four directional neighbors of the grid point (North,East,South,West).

OUTPUT : A channel digraph of the given floorplan.

Before describing the main algorithm we describe in brief, the idea behind the algorithm. Corresponding to each grid point, a marker is initialised to 0. Each external grid point will be indicated by setting its tag field to 1. All internal T-junctions will lie on the intersection of two channels while an external

T-junction will lie on only one channel. Whenever a node corresponding to a channel is added to the channel graph, all the T-junctions lying on the channel will have their marker incremented by one. One can conclude the channel graph to be complete with respect to vertices, once all the internal T-junctions have marker as 2 while the external T-junctions have marker as 1. For each T-junction, the "aaddr" field stores the label of the channel graph node that was recently added to the channel graph. As discussed in Chapter 2 we know the existence of strongly connected components at each level of the maximal rectangular hierarchy. Also there exists a path from a S.C.C at a lower level to a S.C.C at a higher level which contains the lower level S.C.C. So in each outer iteration we start from any T-junction and add nodes of the S.C.C corresponding to each of the two channels and also the nodes belonging to the S.C.C at higher levels.

Procedures used :

1. All_nodes_visited (T_junction_list,curr) : This function returns the value TRUE if all internal T-junctions have marker = 2 and the external T-junctions have marker = 1. Otherwise, it returns FALSE and the address of the unvisited node.

2. Get_base_channel (start_node,end_node) : This procedure returns the start and the end addresses of the base channel corresponding to a T_junction.

3. Get_cross_channel (middle_node,start_node,end_node) : This procedure returns the start and the end addresses of the cross

piece channel corresponding to a T-junction (middle_node).

4. Add_new_node (start,end,size) : This procedure adds a new node to the channel graph with its label = size and stores the channel start and end addresses. Also it adds an arc if the current channel is incident on an a channel whose corresponding node already exists in the channel graph.

5. Add_to_prev_node (cross_channel_node, base_channel_node) : This procedure adds an arc from the base channel node to the cross channel node in the digraph.

6. Mark_channel (channel_node_label,start,end) : This procedure increments the marker of all T_junctions on the channel. For those T-junctions on the channel other than its endpoints, it also adds an arc if possible (node corresponding to the base channel already exists) from the base channel to the current channel node. Also it updates the aaddr field for all T-junctions on the current channel to channel_node_ label.

ALGORITHM

```
1. Initialize marker for all T-junctions to 0. size = 0.
2. For all internal T-junctions set tag to 1.
3. While not all_nodes_visited (t_junc_list,curr) do
   begin
4.    Initialize stack, over = false.
5.    if curr.marker = 0 then
      begin
6.       Push curr onto stack
7.       get_base_channel(curr,next)
```

```
8.          size = size + 1
9.          mark_channel (size,prev,next)
10.         add_new_node(curr,next)
11.         curr = next
        end
12.     else
        begin
13.         next1_node = curr.aaddr
            { aaddr denotes the label of the channel node already
              present in the channel graph }
14          get_base_channel(curr,next)
15          next2_node = next.aaddr
16.         prev = curr
17.         size = size + 1
18.         mark_channel(size,prev,next)
19.         add_new_node (curr,next,size)
20.         add_to_prev_node (next1_node,curr.aaddr)
21.         if next2_node <> 0 then
22.           add_to_prev_node(next2_node,curr.aaddr)
23.         curr = next
        end
24.     while not over do
25.         if curr.mark = 2 or curr.tag = 0 then
26.             if stack is empty then
27.                 over = true
28.             else
```

```
29.             pop(curr)
30.       else
          begin
31.          get_cross_channel_end_pts(curr,prev,next)
32.          if (prev.mark = 1) and (prev.tag <> 0) then
33.             next1_node = prev_aaddr
34.          if (next.mark = 1) and (next.tag <> 0) then
35.             next2_node = next.aaddr
36.          get_base_channel (prev,next)
37.          size = size + 1
38.          mark_channel (size,prev,next)
39.          add_new_node (prev,next,size)
40.          if next1_node <> 0 then
41.             add_to_prev_node(next1_node,curr.aaddr)
42.          if next2_node <> 0 then
43.             add_to_prev_node(next2_node,curr.aaddr)
44.          if (next.tag = 0) or (next.marker = 2) then
45.             curr = prev
46.          else
47.             if (prev.tag <> 0) and (prev.marker <> 2) then
             begin
48.                push prev onto stack
49.                curr = next
                end
          end
       end
```

3.3 Extraction of strongly connected components

Before describing the algorithm we need to describe some terms [AHU74] [ST80]. The Depth First Search (D.F.S) of a directed graph G = (V,A) proceeds as follows : Choose a vertex v belonging to G. The start vertex v is called the root of the D.F.S and is now said to be visited. We then select an arc (v,w) wherein the vertex v is called the father of w, denoted by Father(w). The arc (v,w) is now said to be examined and is called a tree edge. During D.F.S when a vertex v is visited for the first time, it is assigned a distinct integer DFN(v) = i, such that v is the ith vertex to be visited during the search. In general, when we encounter an unexamined arc (v,w) the following two cases occur :

1. w has not been visited : In this case (v,w) is a tree edge.
2. w has already been visited :
   a. w is a descendant of v in the D.F.S forest in which case v,w) is called a forward edge.
   b. w is an ancestor of v in the D.F.S forst in which case (v,w) is called a back edge.
   c. If v and w are not related in the D.F.S forest and DFN(w) < DFN(v) then (v,w) is called a cross edge.

For each vertex v in G, we define LOWLINK such that

LOWLINK(v) = min {v} U { w | there is a cross edge or a back edge from a descendant of v to w, and w is in the same strongly connected component as v }

ALGORITHM

Step 1 : G is a directed graph. For every vertex v in G, set

Mark(v) = 0, Father(v) = 0 and Point(v) = 0.

set i = 1 and stack1 = 0.

Step 2 : Select any vertex r with Mark(r) = 0. Set

Dfn(r) = i

Lowlink(r) = i

Mark(r) = 1

Add r to stack1, set Point(r) = 1 and v = r.

Step 3 : If all the edges incident on v have already been labeled 'examined' then go to step 5. Otherwise select an edge (v,w) which is not yet labeled 'examined'. Label it 'examined' and go to step 4.

Step 4 : 1. If mark(w) = 0, set

i = i + 1

Dfn(w) = i

Lowlink(w) = i

Father(w) = v

Mark(w) = 1

Add w to stack1, and set Point(w) = 1 and v = w.

2. If Mark(w) = 1, Dfn(w) < Dfn(v) and Point(w) = 1 then set

Lowlink(v) = min { Lowlink(v), Dfn(w) }

3. go to step 3.

Step 5 : If Lowlink(v) = Dfn(v), then remove all the vertices

from the top of stack1 upto and including v (these vertices form a strongly connected component). Then set Point(x) = 0 for all such vertices x removed from stack1.

Step 6 : If Father(v) = 0, go to step 7, otherwise set

a. Lowlink(Father(v)) = min { Lowlink(Father(v)), Lowlink(v)}

b. v = Father(v) and go to step 3.

Step 7 : If for every vertex x, Mark(x) = 1, then go to step 8 otherwise go to step 2.

Step 8 : Halt.

Fig 3.2 illustrates an example of how the above algorithm works on a directed graph. For the program written to implement the above algorithm the input is an adjacency list representation of the channel graph. The output of the program is the set of strongly connected components of the channel graph.

3.4 Construction of Component Graph

From the various strongly connected components, a component graph is constructed. The component graph is topologically ordered wherein each node corresponds to a strongly connected component. Each node contains information about the number of nodes in that component and has an arc emanating from it to another node if there is an arc from a node in the lower level S.C.C to the S.C.C at the higher level in the original channel graph..Each S.C.C has one or at least four channel nodes.

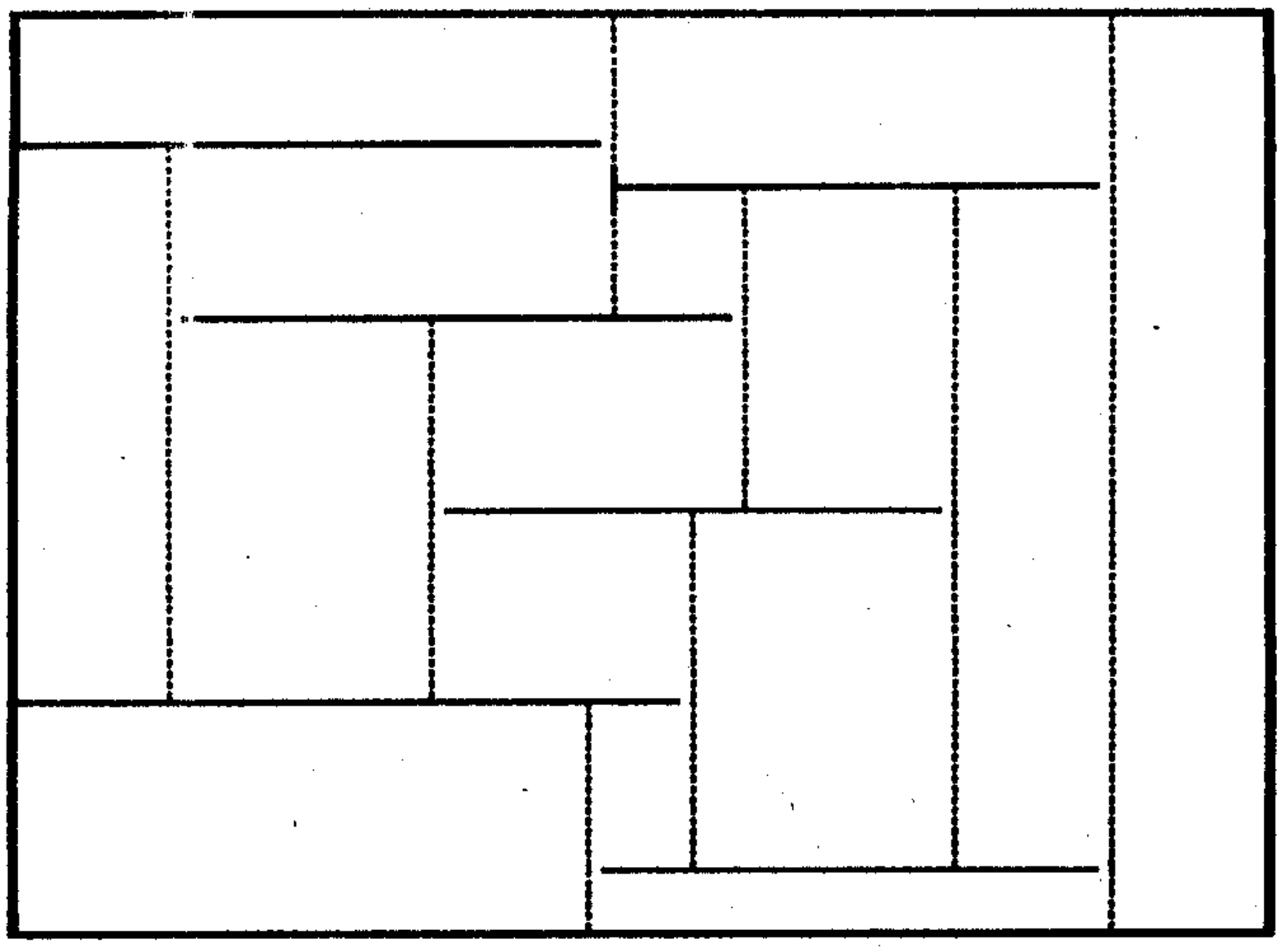


Fig 3.1 Random Floorplan generation

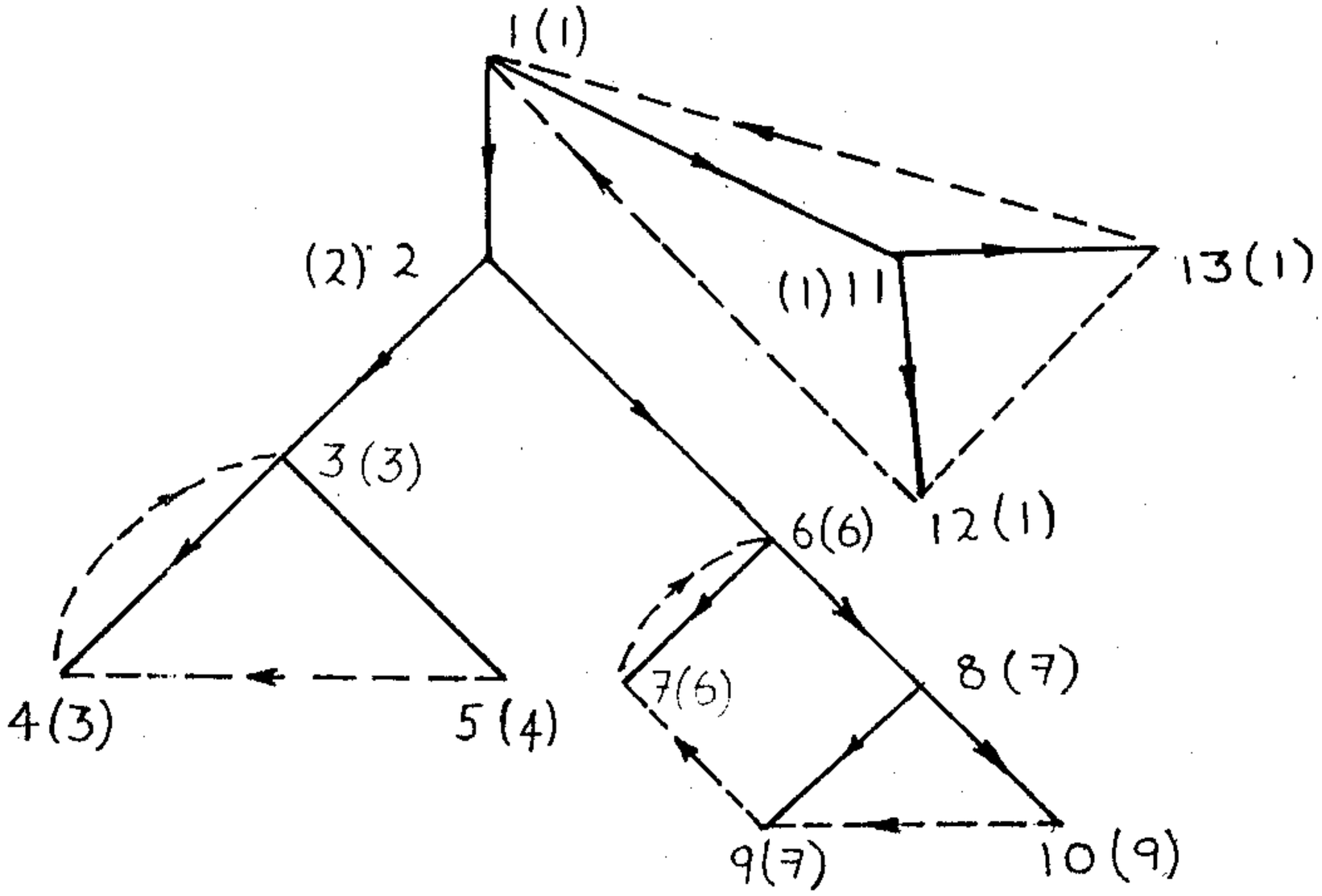
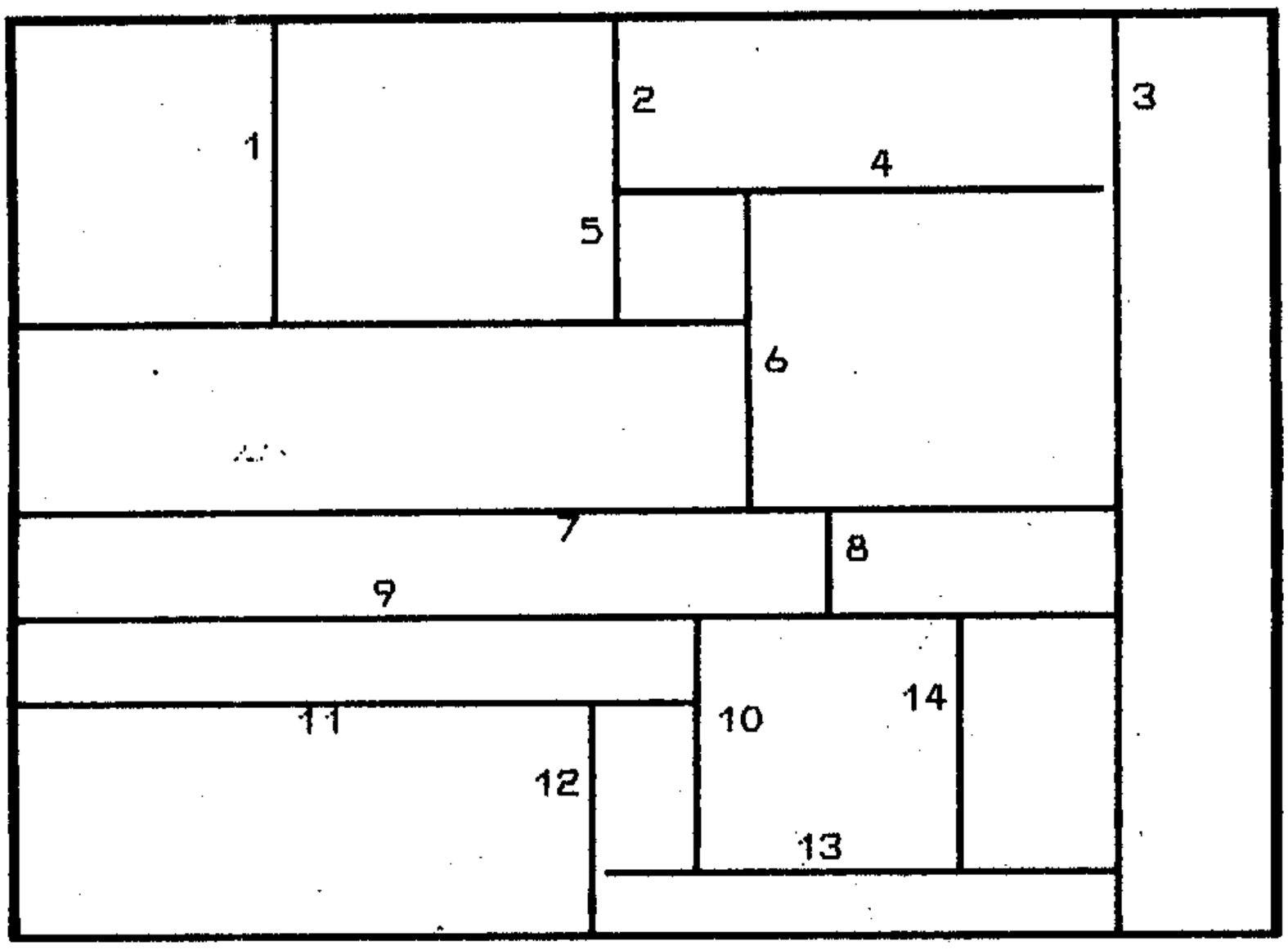
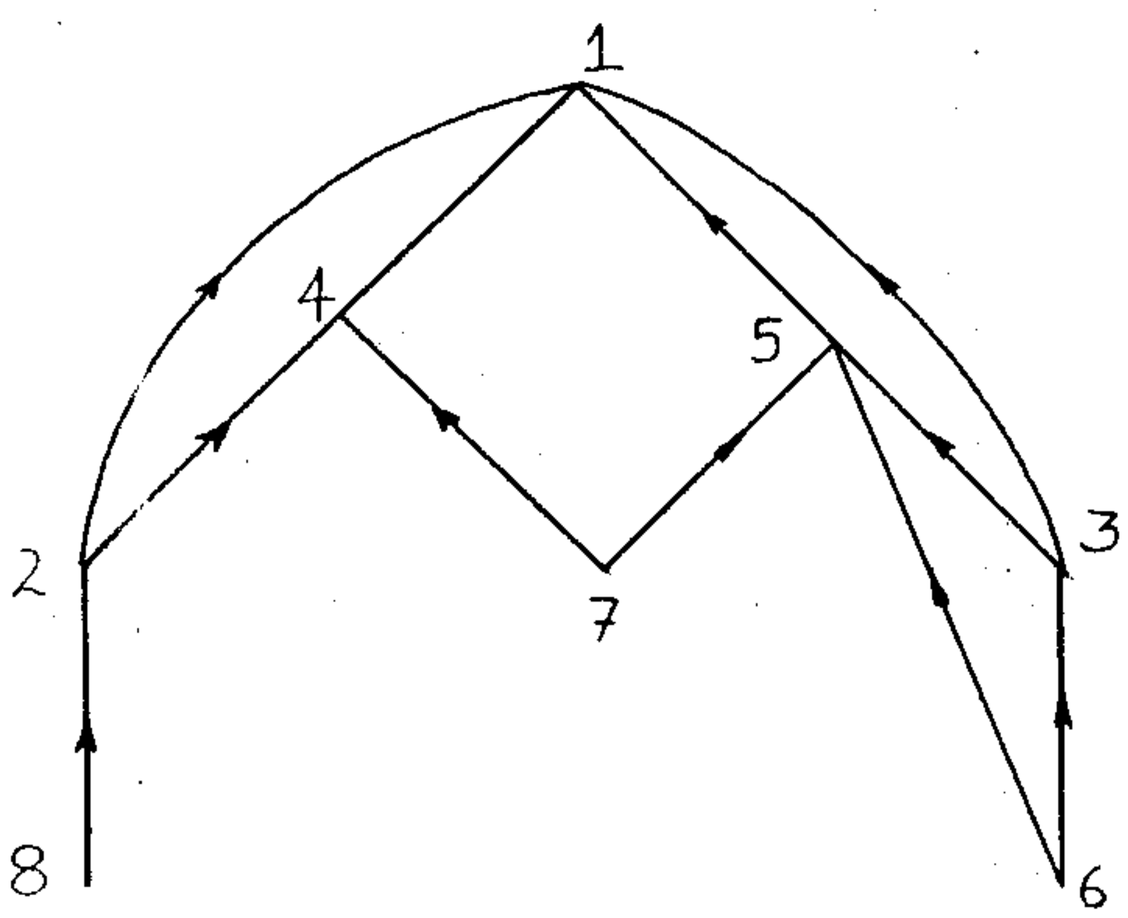


Fig 3.2 Illustration of algorithm in Section 3.3. LOWLINK values are shown in parentheses along with corresponding DFN.

(a)



(b)



(c)Components:1 :{3}, 2 :{2,4,5,6}, 3:{10,11,12,13},

4 : {7}, 5:{9}, 6:{14}, 7:{8}, 8:{1}

Fig 3.3 a) Floorplan b) Component Graph c) Components

The strongly connected components at each level are obtained by checking the component nodes that have zero outdegree. Once a S.C.C is processed, its entry is deleted from the component graph along with its incident arcs.

3.5 Experimental results

The heuristics are run on each of those components that have at least four channel nodes. The results have been tabulated in Tables I and II. Table I gives an idea of the random floorplan generation scheme. The columns show the total number of modules, depth of hierarchy (MRH), width of MRH i.e the maximum number of nodes at any level of the MRH, the number of vertices lying on 4-cycles, the number of vertices lying on longer but not on 4-cycles. Table II presents the C.P.U time on VAX 8651 for the four variations of heuristics for MFVS run on the floorplans in Table I.

The four variations are :

a. Heuristic 2

b. Heuristic 3

c. Heuristic 2 without Heuristic 1 in step 1 (This converts algorithm to that in [Ra82]).

d. Heuristic 3 without Heuristic 1 in step 1.

Exhaustive search for optimal solution took very long time (tens of minutes) for large examples. Among the heuristics we implemented, the best performance was obtained in the first variation, namely Heuristic 2 with heuristic 1 in step 1 at the

beginning. The time taken by the third variation (basically similar to that given by [Ra82]) were often slightly less than the first, but the optimal solution was found by the first. Though Heuristic 3 performed slightly better than Heuristic 2 in terms of the size of the optimal solution, its speed was slower than that of Heuristic 2. In fact the time complexity of Heuristic 3 is of an order higher than that of Heuristic 2.

Sl. No	#Vertices	Depth of hierarchy	Width	#Vertices lying on 4-cycles	#Vertices lying on longer cycles but on no 4-cycle
1	50	4	42	33	9
2	50	4	25	25	4
3	50	6	28	31	5
4	101	4	83	66	17
5	102	4	78	61	17
6	102	6	66	62	12
7	150	6	107	89	22
8	151	6	102	82	24
9	149	6	111	91	27
10	199	6	140	126	28
11	201	6	129	106	27
12	199	6	137	118	27
13	980	8	735	663	130
14	1497	6	1145	985	168

TABLE I : Random generation of nonslicible floorplans

Sl. No	#Vert-ices	Heuristic 2		Heuristic 3		Heuristic 2 w/o step 1		Heuristic 3 w/o step 1	
		|MFVS|	Time secs	|MFVS|	Time secs	|MFVS|	Time secs	|MFVS|	Time secs
1	50	7	0.13	7	0.23	8	0.14	7	0.17
2	50	4	0.11	4	0.20	4	0.14	4	0.16
3	50	7	0.11	7	0.21	7	0.11	7	0.21
4	101	13	0.41	13	0.44	14	0.40	14	0.45
5	102	12	0.30	11	0.33	13	0.32	11	0.43
6	102	11	0.39	11	0.41	12	0.40	12	0.43
7	150	18	0.71	18	0.89	19	0.66	18	0.76
8	151	15	0.67	15	0.75	18	0.72	17	0.79
9	149	18	0.70	18	0.82	22	0.60	20	0.83
10	199	25	1.18	25	1.30	28	0.98	26	1.32
11	201	21	1.13	21	1.27	24	1.11	21	1.20
12	199	24	1.10	23	1.23	26	0.94	25	1.32
13	980	123	27.97	123	30.47	134	22.07	132	30.07
14	1497	184	77.29	181	82.89	203	52.44	192	70.27

Table II Experimental Results of MFVS Heuristic Algorithms

CHAPTER 4

CONCLUSIONS

There are three known methods to determine a feasible channel routing order as given in [PV79],[DAK85] and [GW88]. The MFVS solution can be used in all the three methods to reduce the number of iterations.

Reserved Channels :

In this method [PV79], the straight channels corresponding to the vertices in the MFVS solution are reserved (i.e their width is estimated in advance with some allowance). Since the effort is to minimize the number of such channels, the number of required iterations will be reduced without affecting the completion of successful channel routing.

L-channels :

Feasible routing order with L-channels [DAK85] can also be extracted from the MFVS. For each vertex in the MFVS solution, the channels are redefined by splitting a T-junction opposite to it. For the channel graph in Fig 4.1(b) by using the L-shaped channel we get the modified channel graph as shown in Fig 4.1(c). The advantage of using L-channels is that it can be expanded or contracted for the purpose of completion of routing in it without rerouting other routed channels. Since routing of L-channels is more complex than straight channels, one tries to minimize their

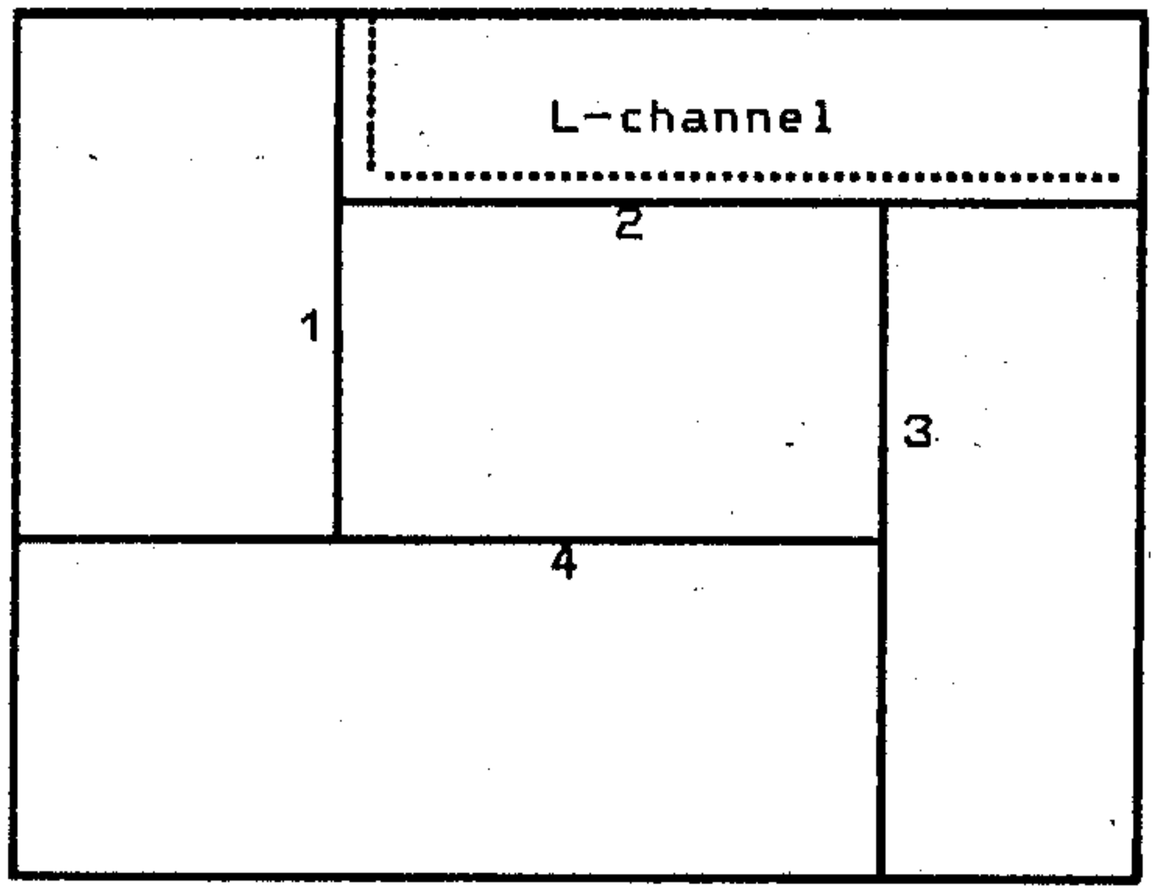
use. For isolated 4-cycles or adjacent 4-cycles one L-channel suffices, while for clusters of 3 or 4 4-cycles, two L-channels are required though |MFVS| = 1. After obtaining the L-channels for all vertices in the MFVS, if two L-channels share a cut, then the MFVS needs to be expanded locally (Fig 4.2). All cycles in the original graph are broken but a few new cycles may emerge in the modified channel graph. It can be shown that this method converges within a few iterations generating new L-channels. An example of a feasible L-channel routing order is shown in Fig 4.3.

Monotone channels :

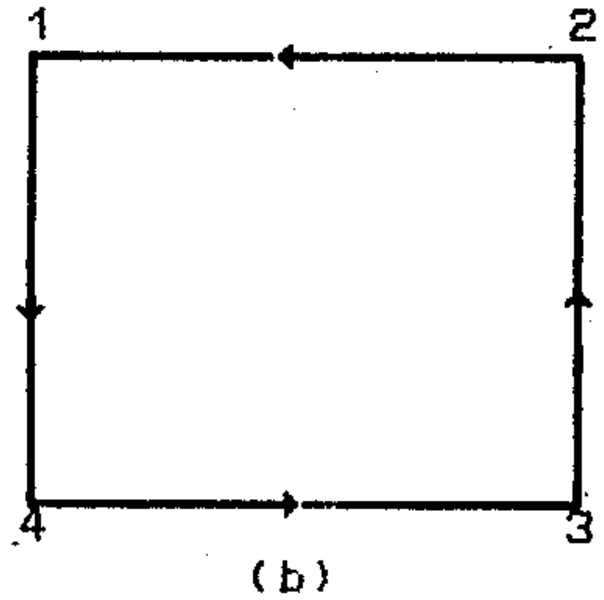
For each vertex in the MFVS the corresponding channel is extended at both ends in a monotone fashion (by the convention of T-junctions) till the floorplan boundary or a previously defined channel is reached (heavy lines in Fig 4.4). By starting from MFVS vertices closest to the floor boundary, it can be guaranteed that a feasible routing order is arrived at.

Scope of Further work :

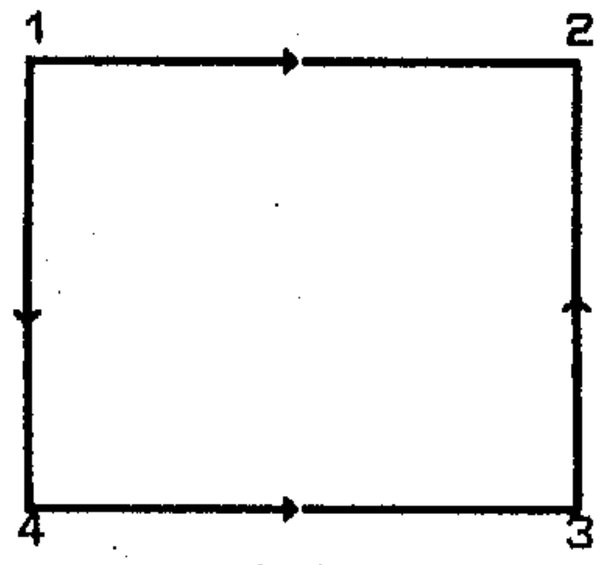
An experimental study of feasible channel routing order based on L-channels and monotone channels needs to be done. Once this is accomplished a comparative study can be made among the three methods.



(a)

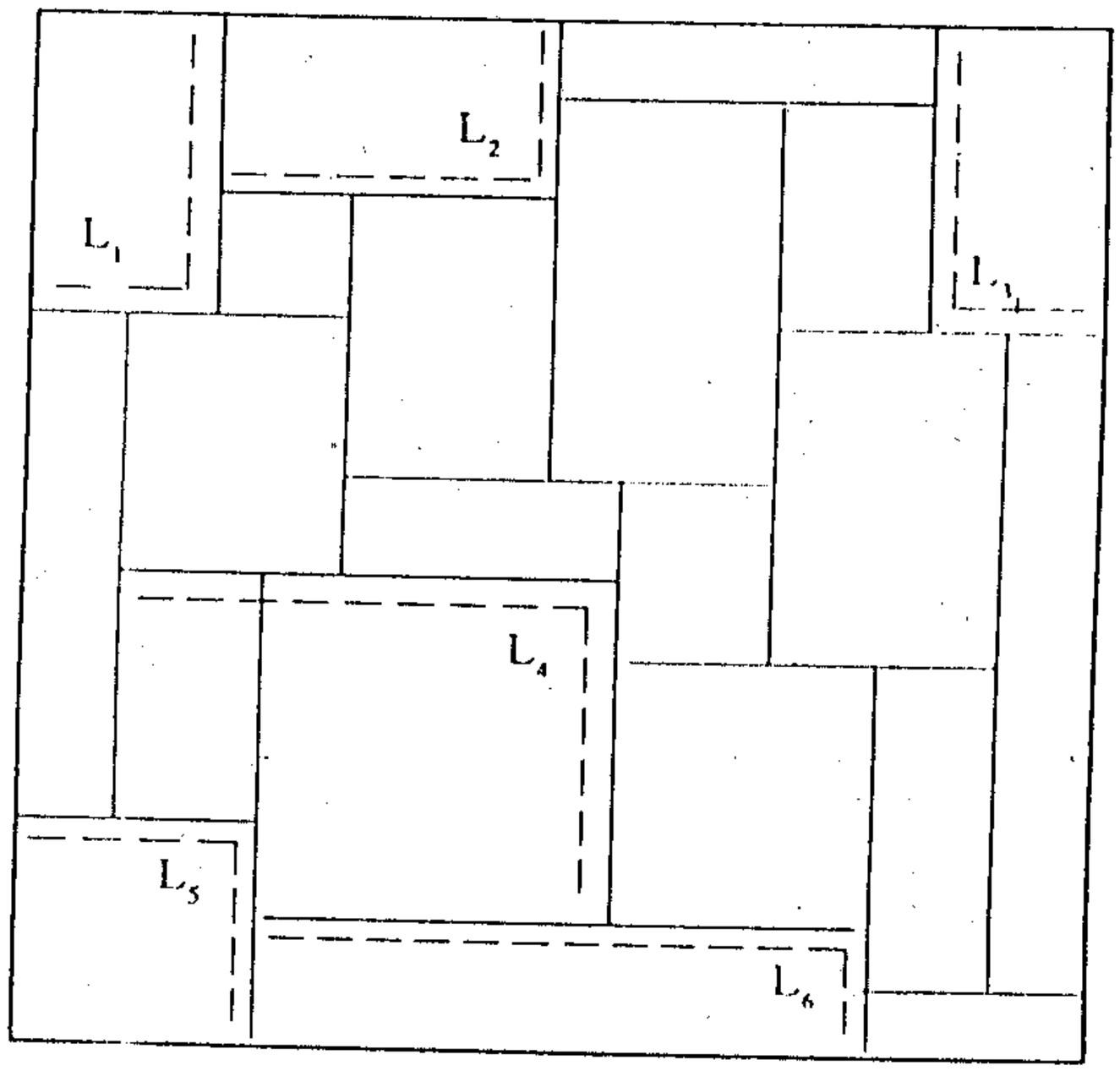


(b)



(c)

Fig 4.1 a) Floorplan b) Channel graph w/o L-channels
c) Channel graph with L-channel



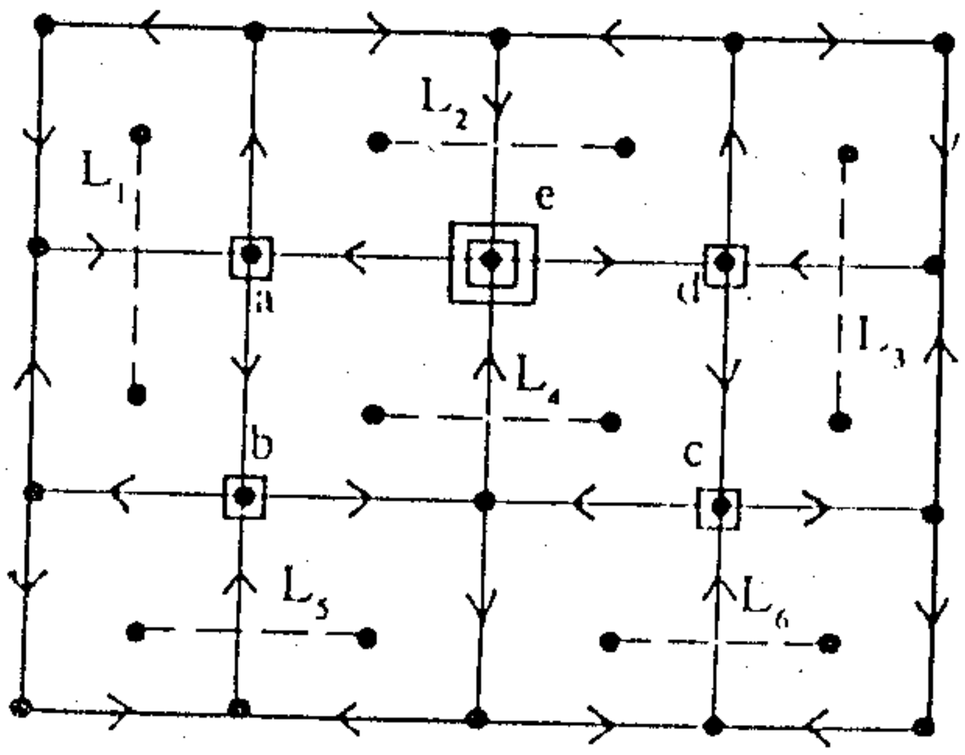


Fig 4.2 Feasible routing order with L-channls based on MFVS = {a,b,c,d} and expanded MFVS = {a,b,c,d,e}.

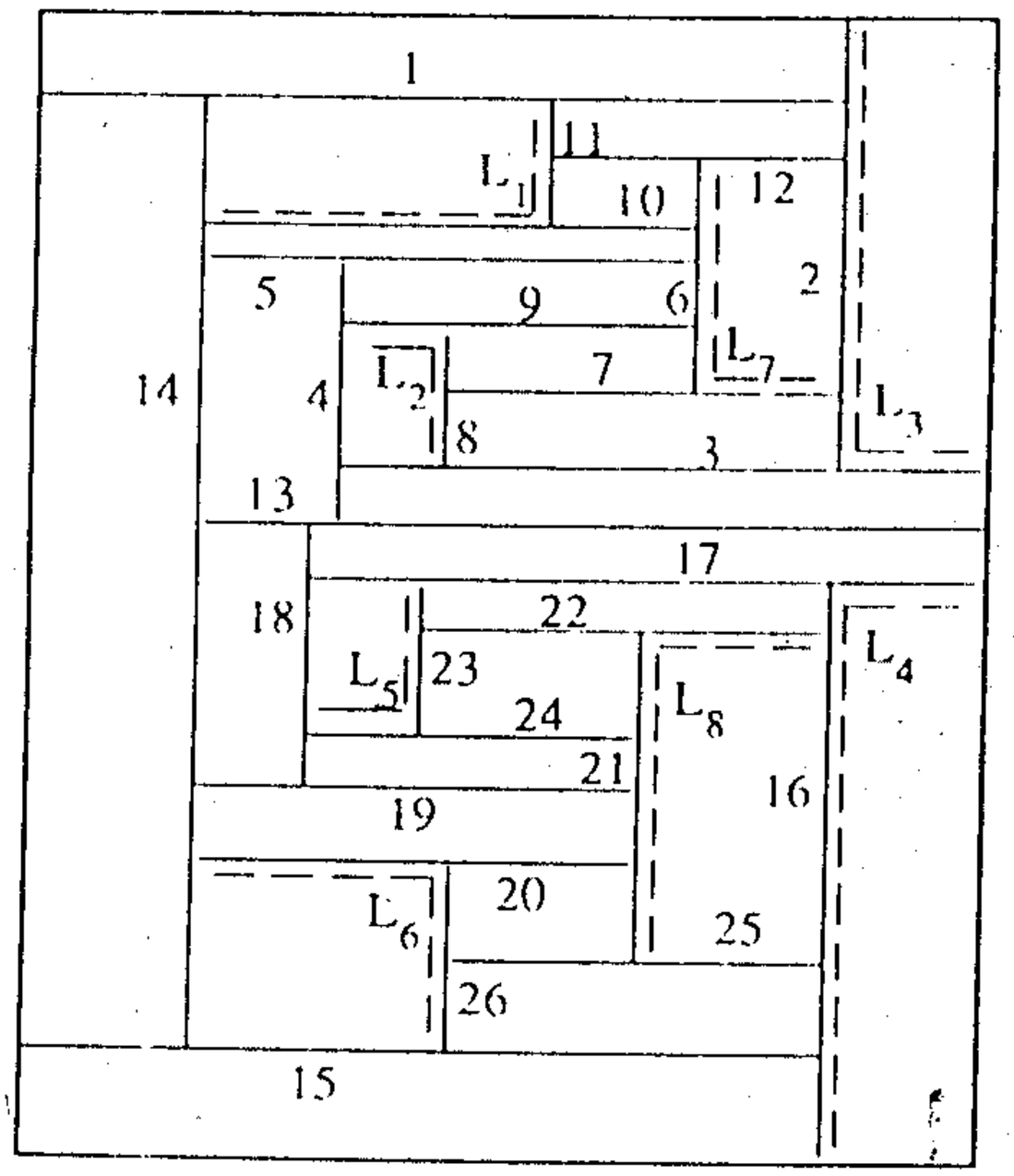


Fig 4.3 Feasible L-channel routing order based on MFVS = {9', 7', L_2, 3', 4, 5, 10', L_7, 12, L_1, 24', 22', L_5, 17, 18, 19, 20', L_8, 25, L_6, 13, 14, 1, 15, L_9, L_4}

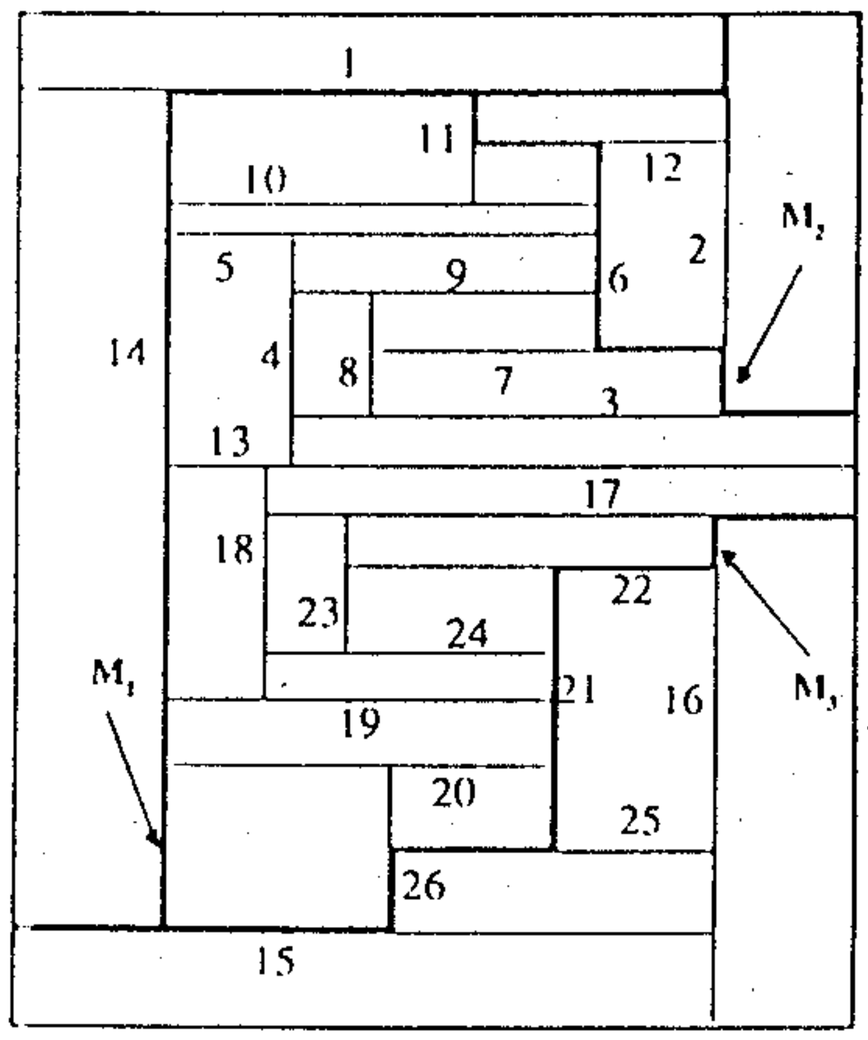


Fig 4.4 Correspondence between minimum set of monotone channels and MFVS ({6, 14, 21})

REFERENCES

[AHU74] Aho,A.V. ,J.E.Hopcroft & J.D.Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Co.,1974.

[CW90] Cai,Y. & I.G.Tollis,"An Optimal Algorithm for Spiral Floorplan Designs," *Proc.of 2nd IEEE Symp.on Parallel & Distributed Processing*, Dallas,December 1990, pp.279-282.

[DAK85] Dai,W.M.,T.Asano & E.S.Kuh,"Routing Region Definition and Ordering Scheme for Building-Block Layout," *IEEE Transactions on Computer-Aided Design*,Vol,CAD-4, No.3,July 1985,pp.189-197.

[GJ79] Garey,M.R. & D.S.Johnson,*Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H.Freeman & Co.,1979.

[GW88] Guruswamy,M. & D.F.Wong, "Channel Routing Order for Building-Block Layout with Rectilinear Modules," *Proceedings of IEEE Intl. Conf. on Computer Aided Design*(ICCAD) 1988,pp.1533-1535.

[KA83] Kajitani,Y. "Order of Channels for safe Routing and Optimal Compaction of Routing Area," *IEEE Transactions on Computer-Aided Design*,Vol,CAD-2,No.4,October 1983, pp.293-300.

[LSW87] Luk,W.K.,P.Sipala & C.K.Wong, " Minimum-area Wiring for Slicing Structures," *IEEE Transactions on*

Computers, Vol.C-36, No.6,June 1987,pp.745-760.

[MC87] McCluskey,E.J.,*Logic Design Principles*, Prentice Hall,1987.

[OT82a] Otten,R.H.J.M.,"Automatic Floorplan Design," *Proceedings of 19th ACM/IEEE Design Automation Conference* (DAC),1982,pp.261-267.

[OT82b] Otten,R.H.J.M.,"Layout Structures," *Proceedings of IEEE Large Scale Systems Symp.* 1982,pp.349-353.

[OT82c] Otten,R.H.J.M.,P.S.Wolfe,"Algorithm for wiring space assignment ins slicing floorplans," *IBM Tecnical Disclosure*, Y0882-0682,1982.

[OT83] Otten,R.H.J.M."Efficient Floorplan Optimization," *Proceedings of IEEE Intl.Conf.on Computer Design* (ICCD), 1983,pp.499-502

[PL88] Preas,B.T. & M.J.Lorenzetti, *Physical design Automtion of VLSI Systems*, The Benjamin/Cummings Publishing Company, Inc.,1988.

[PV79] Preas,B.T. & W.M.vanCleemput,"Placement Algorithms for Arbitrary Shaped Blocks," *Proceedings of 16th ACM/IEEE Design Automation Conference* (DAC), 1979, pp.474-480.

[Ra82] Ramsdell,J.D.,"Finding Minimal feedback vertex Sets," *IEEE Transactions on Circuits and Systems*, Vol.CAS-29, No.9,September 1982.

[SB88] Sur-Kolay.S. & B.B.Bhattacharya, "Inherent Nonslicibility of Rectangular Duals in VLSI

Floorplanning."*Lecture notes in Computer Science*, No.338,Springer Verlag (Berlin), 1988,pp.88-107.

[SB89] Sur-Kolay.S. & B.B.Bhattacharya, "Channel Graphs of Nonslicible Floorplans and an Efficient Heuristic for their Minimum Feedback Vertex Set Problem,"*Tech. Rep.* (P & ES),VLSI-4/89,Electronics Unit,I.S.I., Calcutta,1989.

[SB91] Sur-Kolay.S. & B.B.Bhattacharya, "The Cycle Structure of Channel Graphs in Nonslicible Floorplans and a Unified Algorithm for Feasible Routing Order," *Proc. of Intl.Conf.on Computer Design*(ICCD), Cambridge,MA.,1991,pp.524-527.

[SS83] Supowit,K.J. & E.F.Slutz.,"Placement Algorithms for Custom VLSI," *Proc of 20th ACM/IEEE Design Automation Conference*(DAC), June 1983,pp. 164-170.

[SS91] Sur-Kolay.S,"Studies on Non slicible floorplans in VLSI layout design", PH.D Thesis, Jadhavpur University, Calcutta.

[ST80] Swamy,M.N.S. & Thulsiraman,K.,"*Graphs, Networks and Algorithms*", John Wiley & Sons,1980.

[ST83] Stockmeyer,L.J.,"Optimal Orientations of Cells in Slicing Floorplan designs," *Information and Control*, Vol.57,1983,pp.91-101.