

Copyright © 1983, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.


GLOBAL ROUTING FOR GATE ARRAY

by

J-T. Li and M. Marek-Sadowska

Memorandum No. UCB/ERL M83/60

23 September 1983



GLOBAL ROUTING FOR GATE ARRAY

by

J-T. Li and M. Marek-Sadowska

Memorandum No. UCB/ERL M83/60

23 September 1983

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Global Routing for Gate Array

Jeong-Tyng Li and Malgorzata Marek-Sadowska

Department of Electrical Engineering and Computer Sciences
Electronics Research Laboratory
University of California, Berkeley, CA 94720

ABSTRACT

We propose a new approach to global routing of gate arrays which can handle any channel capacities and pin distributions on the chip. Our approach first identifies the unique connection patterns, then takes advantage of the channel capacity in the outer area before that of inner area. The route of each net grows cautiously from outside toward inside in order to achieve high completion rate. We also develop a backtracking method to remove the wrong connections made in the outer area. Finally, we give a special case which the maximal network flow technique can solve in polynomial time.

September 23, 1983

Global Routing for Gate Array

Jeong-Tyng Li and Malgorzata Marek-Sadowska

Department of Electrical Engineering and Computer Sciences
Electronics Research Laboratory
University of California, Berkeley, CA 94720

1. Introduction

The purpose of global routing is to provide a loose initial routing of nets on a chip to guide the subsequent detailed wiring. It is a crucial part in gate array layout design.

A gate array wafer is preprocessed up to and before the interconnection level and custom logic functions will be realized by the final interconnections. Usually the gate array layout design is divided into two phases, "placement" and "routing". However, it is quite difficult to determine whether or not the placement results are satisfactory until the routing is done. On the other hand, it is also hard to say which phase needs improvement if routing fails: when a poor routing method is used, we may not complete the interconnections for otherwise good placements.

The wiring space in gate arrays is limited. Channel widths are fixed, giving rise to the "routing problem for gate arrays," that of fitting the desired interconnections into the available channels. In order to obtain 100% wire completion, we need to distribute the wires over the chip such that no overflow occurs. This problem is not trivial if (1) each wiring channel is nearly saturated, or (2) the channel capacities and pin distribution are not uniform due to the physical constraints (e.g., complicated macro design, internal blockage and macro intrusion).

For the gate array wiring, many algorithms have been developed. For instance, [1, 2] route each net independently using either approximate Steiner tree method or simply maze-running. If overflows occur, then they reroute some nets from the congested spots in order to reduce the congestion. Ting and Tien explore the independent rerouting issue [2]. The incremental assignment

method [3] uses the "smallest-enclosing-rectangle-first-connected" manner for the net ordering to be connected. After one net is routed by the shortest path, the edge weights on a graph reflecting cell adjacency and channel capacity are updated. A hierarchical routing approach [4] solves the global wiring problem as an integer programming problem, but with the crucial assumption of a uniform wiring substrate. Therefore, the choice of cut lines is of little importance. Karp, et. al. [5] provide an upper bound on the channel capacities and a guaranteed algorithm, but their bound is still too high for the practical cases.

In this paper, we deal with more general cases where both channel capacities and pin distributions can be non-uniform. In the following sections we give a problem formulation, discuss the existence and properties of solutions and propose a heuristic algorithm based on our analysis. We also discuss briefly the computational complexity.

2. Formulation of the Problem

Let us assume that the whole chip is divided into a rectangular array of cells, each of which contains some wiring grid lines and circuit pins. The interconnections and exact pin locations within each cell are ignored [1]. Each net is identified with a set of cells where interconnection pins are located. Each cell is surrounded by four boundaries. Channel capacity of the boundary is the estimated number of tracks available for net crossing. Each cell thus has four channel capacities associated with it.

The global routing problem is to construct wire routes for each net in such a way that for any cell boundary the number of nets crossing it is less or equal to the corresponding channel capacity. The number of cell boundaries the wire route crosses is taken to be the length of the wire route. Fig. 1(a) shows a global routing example for gate array in which the pins of 6 nets are distributed in 3×3 cell array and channel capacity is indicated on each boundary. Fig. 1(b) shows a wiring result.

We assume here that two interconnection layers are available for routing, one of which is used for horizontal wiring, another for vertical.

3. Existence and Properties of Solutions

Let P be a global routing problem for gate array. We say P has a solution if there exist routing patterns for P such that all the nets are completed and, for any cell boundary, the number of nets crossing it does not exceed the associated channel capacity. Suppose P has m feasible solutions in total, and $\{ S_j \}$, $j=1, 2, \dots, m$, is a set of all the solutions of P. A route R_j^i of net N_i is said to be a feasible route if there is a solution S_j in which net N_i takes the route R_j^i . Therefore, $S_j = \{ R_j^1, R_j^2, \dots, R_j^n \}$, where n is the total number of nets in P. If $U = \bigcap_{j=1}^{j=m} S_j \neq \phi$, then U contains the unique portions of solutions, i.e., some nets have unique routes no matter how others are connected. For instance, if N_i has a unique route, then $R_1^i = R_2^i = \dots = R_m^i$. If N_i has a part of solutions which is unique, then $\bigcap_{j=1}^{j=m} R_j^i \neq \phi$. If we are unable to identify these unique routing patterns, finding a solution of P may require an exhaustive search. In fact, simplified formulations by [5, 6, 7] have shown the global routing problem to be NP-complete. Therefore, we aim at a heuristic algorithm which will wire each net very cautiously to eliminate most of the rerouting steps.

Consider an example in Fig. 2(a) which has two solutions shown in Fig. 2(b)-(c). Fig. 2(d)-(e) show the set of feasible routes for net $N_1 : \{ R_1^1, R_2^1 \}$, (f)-(g) the set of feasible routes for net $N_2 : \{ R_1^2, R_2^2 \}$. If N_2 were connected as in Fig. 2(h), then N_1 can not be connected in that case, since there is no capacity left on cell (a, 1)'s boundaries with which to connect the pins of N_1 in cells (a,1) and (a,2). In other words, there does not exist a solution to the problem Fig. 2(a) in which N_2 takes shape shown in Fig. 2(h). Now, if N_1 is wired as in Fig. 2(i), then net N_2 has only one feasible route: $\{ R_1^2 \}$. If a portion of net N_2 is connected as in Fig. 2(j), then feasible solutions for uncompleted nets are: $\{ R_1^1 \}$ for N_1 , and $\{ R_1^2 \}$ for N_2 .

Therefore, in order to find a solution for P, we hope to have an algorithm with the following capabilities:

- (1) It can select a correct route for a net in each routing step. It does not need to find an entire route for that net, i.e., it only needs to connect some pins and go on to other nets.

(2) After each wiring step, it will be able to update the feasible routes for not yet completed nets as in Fig. 2(j).

Intuitively, it is clear that we can not build an algorithm having the capability (2) and solving P without an *a priori* knowledge of all solutions of P. However, the capability (2) can be expressed in terms of feasible routing area.

Consider a global routing problem P and the current situation on the chip.

Definition 1. A set F_i of cells forms a **feasible routing area** for net N_i , if, for any cell $C_a \in F_i$, there is a feasible route R_j^f that reaches C_a , for some solution S_j of P.

For example, originally feasible routing areas for N_1 and N_2 are $\{(a,1), (a,2), (b,1), (b,2)\}$. Suppose now that N_2 is partially connected as in Fig. 2(j). Then the feasible routing areas are $\{(a,1), (a,2)\}$ for N_1 and $\{(b,1), (b,2)\}$ for N_2 . Hence, a feasible routing area for a net is determined by the current situation. In other words, once a net is partially connected, the feasible routing areas of other nets may be changed.

The feasible routing area can be also interpreted as follows. Let C_j^f denote a set of cells at which a route R_j^f can arrive. The feasible routing area for net N_i is:

$$F_i = \bigcup_{j=1}^{j=k} C_j^f, \text{ where } k \text{ is the total number of solutions in the current situation.}$$

Definition 2. A cell is said to be a **non-pass-through (NPT)** cell if the number of unconnected pins of different nets inside it is equal to or less by one than the total remaining channel capacities on its four boundaries.

For example, cell (a,1) in Fig. 1(a) is an NPT cell, since there are 3 unconnected pins and total channel capacities are 4. It thus can not be crossed by nets N_4 and N_5 which have no pins in (a,1), otherwise at least one of three pins in (a,1) can not be connected outward.

We can extend definition 2 to describe a non-pass-through region. A non-pass-through region is a region whose boundary can not be crossed by a net with no pins inside this region. Cells (a,1) and (b,1) in Fig. 1(a) form an NPT region, since the total channel capacities on the region boundary are 6, and there are 5 different pins to be connected outward. Therefore net N_6 can not pass through that region.

Lemma 1. Suppose P has a solution. If there is a 2-pin net N_i with pins in two adjacent cells with non-zero channel capacity on the common boundary, then there exists a solution S_j of P in which N_i is wired through the common boundary of such two adjacent cells.

Proof: Suppose the two pins of N_i are in cells C_a and C_b and they are not wired by the shortest path in a solution S_1 . We know S_1 exists, since P has a solution. There are two possible cases:

- (1) In S_1 , the channel capacity on the common boundary of C_a and C_b is greater than the crossing demand there. Then the current route of N_i can be removed and replaced simply by the segment between C_a and C_b .
- (2) In S_1 , no channel capacity is left on C_a and C_b 's common boundary. It means that there is at least one other net N_j which crosses the common boundary (see Fig. 3(a)). We can reroute N_j by taking the route of N_i , and connect N_i by the segment between C_a and C_b . Fig. 3(b) illustrates this point. •

Lemma 2. If P has a solution, and the total channel capacities on a closed region boundary are μ and there are ν pins inside, $\nu > \mu$, then at least $\nu - \mu$ of them should be connected within this region.

Proof: By contradiction. Suppose only γ pins are connected inside the region, $\gamma < \nu - \mu$. In the best case, only those $\nu - \gamma$ unconnected pins need to pass through the region boundary. Thus, there should be more than $\nu - (\nu - \mu)$ channel capacities to accommodate them, which contradicts the assumption. •

Fig. 4(a) shows a closed region (outlined in boldface) with channel capacities 5 along its boundary and 8 pins of 5 different nets. These 5 nets need to be connected outward. Hence there are 6 pins which must be connected within the region.

Lemma 3. Suppose P has a solution. If there are two adjacent NPT cells, C_a and C_b , with non-zero channel capacity μ on the common boundary, and there are μ or fewer nets with pins in both C_a and C_b . Then there is a solution in which these pins are connected by the shortest path between C_a and C_b .

Proof: Let $N_1, N_2, \dots, N_j, j \leq \mu$, be the nets with pins in both C_a and C_b . If there

are nets with exactly 2 pins among N_1, \dots, N_j , then we can apply Lemma 1 to connect such nets by the shortest paths. If all j nets are connected, we are done.

Otherwise, we consider the remaining nets, say $N_1, \dots, N_k, 1 \leq k \leq j$. Note that the channel capacity left on the C_a and C_b 's common boundary becomes $\mu - (j - k)$. Since C_b is an NPT cell, the nets with pins in C_a , but not in C_b , can not cross C_b . Only N_1, \dots, N_k can pass through the common boundary. Since channel capacity $\mu - j + k \geq k$, each of N_1, \dots, N_k has a route through the common boundary. This completes the proof. ■

Fig. 4(b) shows two NPT cells with capacity 2 on the common boundary. Net 1 and 4 have routes through the common boundary, shown in Fig. 4(c).

The following theorem describes the situation when the interconnection has a unique pattern.

Theorem 1. The wire route of net N_i has a unique pattern if and only if N_i is forced by the current situation on the chip (i.e., channel capacity constraints, previous routed patterns, etc.) to take the unique shape.

Proof: First, if the situation on chip will force a net N_i to take a unique route or part of it, then, in any feasible solution, N_i 's route is unique or part of it is unique.

Next, suppose N_i has a unique solution or part of it is unique. Since the connection pattern (or part of it) is unique now, no matter how the remaining nets are routed, this particular connection can not take a different shape. In other words, no matter how the connections are made for other nets at each step, the feasible routing area (or part of it) for N_i is the same at each time. If this is not the case, N_i can take at least 2 different shapes, which contradicts our assumption. Thus, a unique pattern of N_i is forced by the current situation on the chip. This completes the proof. ■

Before going on, we need the concept of "barrier".

Definition 3. A barrier is a boundary of a closed non-pass-through region.

Definition 4. A boundary of a closed region is said to be a barrier for net N_i if the route of N_i can not cross this region.

In Fig. 5, the boundary of the region formed by cells (a,1) and (b,1), outlined in boldface, is a barrier for net N_3 , since N_3 can not cross this closed region.

Definition 5. A connection pattern is said to be a generalized routing for a net N_i if it connects all the pins of N_i and may contain some redundant segments.

Fig. 6(a) shows a route for net N_1 . Fig. 6(b) shows a generalized routing for N_1 , with an example of redundant segments in the upper right quadrant. The generalized routing will be used in proving the following theorem.

Theorem 2. Suppose a global routing problem P has a solution and a net N_i is to be connected. The feasible routing area F_i of N_i is a region B_i bounded by the barriers for net N_i .

Proof: Consider a generalized routing for net N_i . Suppose the theorem is not true, then there are 3 possible cases:

- (a) $F_i \supset B_i$ and $F_i \neq B_i$.
- (b) $F_i \cap B_i \neq \phi$, but $F_i \not\supset B_i$ and $F_i \not\subset B_i$.
- (c) $F_i \subset B_i$ and $F_i \neq B_i$

If either (a) or (b) holds, then, from the definition of barrier, P does not have a solution, since N_i can cross its barrier. This contradicts the assumption of the existence of a solution.

Suppose (c) holds. Consider any cell C_a in $B_i - F_i$. Since C_a is not in the feasible routing area of N_i , generalized routings of N_i can not reach C_a . Therefore, the boundary of C_a is a barrier for N_i . Hence it should be part of the boundary of B_i . It implies $B_i - F_i = \phi$, which contradicts the assumption $F_i \neq B_i$. This completes the proof. ■

By Theorem 2, in order to find a solution for P, we need to find the barriers first to guide the net connections, i.e., check all possible dissections of the routing area. This amounts to an exhaustive search.

4. Graph Model and Definition

We define a cell graph $G=(V, E)$, where each vertex $v_i \in V$ corresponds to a cell, and there is an edge $e_{ij} = (v_i, v_j) \in E$ if the corresponding cells of v_i and v_j are adjacent with non-zero channel capacity on the common boundary. Each

edge e_{ij} has an associated capacity which is the corresponding channel capacity on the common boundary of v_i and v_j . Fig. 7 shows a cell graph for Fig. 1(a).

Definition 6. In a particular planar drawing of the cell graph G (see Fig. 7), with the boundary of the exterior face of the G 's planar drawing corresponds to the cells abutting the boundary of the chip, we define two types of mesh.

- (1) **Outermost mesh** which bounds the particular drawing of G , i.e., the circuit separates the exterior face from G .
- (2) **Inner meshes** which bound the areas in the drawing, where edges are missing.

Fig. 8(a) shows an example of a cell array where the cell boundaries with 0 channel capacity are marked with heavy lines. Fig. 8(b) shows the plane graph G with the outermost and inner meshes drawn in dotted lines. Sometimes the outermost and inner meshes may overlap somewhere. See Fig. 8(c)-(d).

5. Heuristic Algorithm

Since the gate array global routing problem is NP-complete, we must develop a heuristic algorithm. It first embeds the unique routes forced by the NPT cells. Then it connects nets along the outermost and inner meshes, which we will explain in detail. The outermost (inner) mesh will become smaller (larger) as the channel capacities on the mesh are used up. The routed nets will gradually grow from outside to inside. Hence, competition for the internal routing resource is lessened. It makes chips more routable and eliminates most of the rerouting steps. In the following subsections, we explain each routing step.

5.1. Two-Pin-Adjacent-Cell Rule

We first search for the nets containing only 2 pins which are located in two adjacent cells. By Lemma 1, these nets have the routes simply passing through the common boundary of two adjacent cells, respectively. If the channel capacity on the common boundary is exceeded by the number of nets requiring passage through the boundary, then we arbitrarily select the exact amount of nets equal to the channel capacity to be connected by the shortest paths.

5.2. Finding NPT Cells and Barriers

After we make the 2-pin-adjacent-cell connections, we update the boundary channel capacities. Since finding NPT regions and barriers for nets is time consuming, we only locate the barriers composed of the NPT cells. In other words, we detect in the algorithm the following barriers from the cell graph G .

- (1) A cutset whose elements are NPT cells which can be grouped like a wave-front in G (see Fig. 9(a)-(b)).
- (2) G is disconnected (see Fig. 9(c)-(d)).

5.3. Unique Routes

If there is a net whose pins can not be connected due to the barriers or graph disconnection, report no solution for the given problem. Otherwise we consider the following cases.

- (1) Some nets with pins in the neighborhood of NPT cells are forced to take unique routes. For instance, cell (c,2) is an NPT cell and net N_1 does not have a pin in (c,1). N_1 thus should have a route from cell (c,1) up to (b,1), because N_1 can not cross cell (c,2). (see Fig. 10(a))
- (2) By Lemma 3, the nets which have 2 pins in two adjacent NPT cells have connections simply through the common boundary of adjacent NPT cells. Fig. 10(a) shows cells (a,4) and (b,4) are NPT cells which both contain a pin of N_2 . We make a connection for N_2 through the common boundary of (a,4) and (b,4).
- (3) If there is a net with two pins located in an NPT cell and its non-NPT neighbor, respectively, we connect them by a segment through their common boundary. Fig. 10(a) presents a connection pattern for N_3 which has two pins in cells (c,2) and (c,3), where (c,2) is an NPT cell and (c,3) is not.

If there is no detected unique route, we go to section 5.4. Otherwise, we embed unique routes, update the channel capacity on cell boundary and continue to consider the following cases.

5.3.1. Updating the Cell Status (From non-NPT To NPT)

We find that a non-NPT cell may become an NPT cell because it contains end points of net routes. To make it clear, we consider an example in Fig. 10(a). Cell (b,1) now has an end point of N_1 's route which is forced by the NPT cell (c,2). The remaining channel capacities on (b,1)'s boundaries are 2. Therefore, if some other net passes through (b,1), N_1 has no way to connect to (a,1). Cell (b,1) is thus a new NPT cell. This indicates that the status of cell (b,1) needs to be updated to be an NPT cell.

Let us consider an end point of a net's route in a cell to be a new pin in that cell. We use the following rule to change a cell status from a non-NPT cell to an NPT cell.

A cell status is updated to be an NPT cell if the number of pins (including new pins) of different uncompleted nets inside it is equal to or less by one than the remaining channel capacities on its four boundaries.

Suppose there are new NPT cells created after we embed the unique routes. We continue to check the neighborhood of new NPT cells in order to find the unique routes. For instance, in Fig. 10(a), cell (b,1) is a new NPT cell. Hence, a pin of N_4 is forced to move from (a,1) to (a,2). Also case (3) in detecting unique routes can be applied to N_1 which thus has a route from (b,1) up to (a,1). Therefore, N_1 is finished with route (c,1) \rightarrow (b,1) \rightarrow (a,1). (see Fig. 10(b))

5.3.2. Extension of 2-Pin-Adjacent-Cell Rule

If there is a net which needs one more segment of unit length to complete its connection, then we add the segment, where unit length means crossing only one cell boundary. For instance, in Fig. 10(a), net N_3 needs a segment from (c,3) to (b,3) to complete its connection. Its final route is shown in Fig. 10(c), (c,2) \rightarrow (c,3) \rightarrow (b,3). We can view this step as an *extension of 2-pin-adjacent-cell rule*.

After we embed routes due to new NPT cells and extension of 2-pin-adjacent-rule, we repeatedly check the new NPT cells' neighbors and apply the extension rule to applicable nets until no further unique routes can be found.

5.4. Mesh Routing

We update the channel capacities after imbedding the detected unique routes. If cell graph G is disconnected, we treat each maximal connected sub-graph, i.e., component of G , as a sub-problem. Suppose now, without loss of generality, G is connected. Since there are no detected unique connection patterns, we create the outermost and inner meshes of G and connect nets along the meshes. The reason is to push the wires to the chip boundary in order to reduce the congested spots in the center of the chip. We can view inner meshes as the boundaries of obstructions in the chip. When the channel capacities are used up along the meshes, the outermost (inner) mesh will become smaller (larger). We then keep on working on the smaller (larger) meshes until we finish all the nets.

5.4.1. Connecting a Pair of Pins Along Meshes

Suppose we have a global routing problem, shown in Fig. 11(a), whose outermost mesh is in Fig. 11(b). For each cell on the outermost mesh, we associate with the pins which are contained in it. For instance, the vertex corresponding to cell (a,1) is associated with 2 pins of nets 1 and 5 (see Fig. 11(b)). We create a table in which the nets with least pins appearing on the mesh are at the top. Fig. 11(c) shows a sorted table. We compute the shortest distances for those nets with 2 pins on the mesh. If none exist, we then consider those of 3 pins on the mesh, and so on. For instance, the two pins of net N_1 are three units apart (a,2) \rightarrow (a,1) \rightarrow (b,1) \rightarrow (c,1) and 5 apart for N_4 . Therefore, net N_1 is to have such a connecting path on the mesh, provided the following things do not happen.

(1) Disconnection Trouble

Cell graph G will be disconnected so that there is a net which can not be connected due to this separation.

For instance, net N_4 is to be wired along a mesh by a wavy path shown in Fig. 12(a). Unfortunately, the capacities on edges (d,2)-(d,3) and (b,4)-(c,4) are 1, respectively. Therefore, the cell graph will be disconnected so that net N_4 can not be connected (Fig. 12(b)). This indicates that N_4 can not take that route.

(2) Barrier Trouble

The NPT cell barriers will separate cell graph G into several disjoint regions so that a net can not be connected.

For instance, net N_i is to be wired by a wavy path shown in Fig. 12(c). Since no capacity will be left on edge (a,2)-(a,3), NPT cell barrier $\{ (b,3), (c,2) \}$ will separate G into two disjoint regions so that net N_j can not be connected (Fig. 12(d)). Therefore, N_i should not take that route.

After the above look-ahead check for net N_1 in Fig. 11(a)-(c), if no immediate trouble occurs, then we will connect N_1 along the mesh by the shortest path (a,1) \rightarrow (b,1) \rightarrow (c,1). Otherwise, we choose another pin pair of the next shortest distance to be connected so as to avoid trouble.

5.4.2. Handling The Vertex of Degree 1

When we exhaust the channel capacities on the mesh, some vertices may have only one degree. For instance, Fig. 13(a), (b) shows that cell C_i has only one degree, since N_1 's route passes through C_i and N_2 's route is forced by NPT cells to reach C_i . In order to handle a cell C_i of degree 1 in G , we consider the following possible cases.

- (1) A pin is originally in C_i given a global routing problem. For instance, net N_3 has a pin in C_i , which is specified by the problem input.
- (2) A new pin is forced to be in C_i , e.g., net N_2 is forced to reach C_i so that C_i has a new pin of N_2 .
- (3) A net whose route passes through C_i . For example, net N_1 passes through C_i .

The rule for moving pins or extending routes from C_i to its neighbor C_j is: original pins first, new pins second and extension of routes last.

If the channel capacity between C_i and C_j is equal to or larger than three, then N_1 , N_2 and N_3 can each have a route $C_i \rightarrow C_j$. Otherwise, we move N_3 's pin first and N_2 's second.

5.4.3. Mesh Expansion

In the following cases, we can not imbed a connection along meshes. The outermost mesh thus needs to be expanded inward, inner meshes outward.

- (1) No net has more than 1 pin appearing on the mesh so that no connection can be made.
- (2) No net can be wired on the mesh because its connection will create disconnection and barrier troubles explained above.

Fig. 14 illustrates an inward expansion of an outermost mesh.

5.4.4. Wire Route Pruning

When we complete a net wire route, it may contain redundant segments, namely, it is a generalized routing. For instance, N_1 , N_2 and N_3 have routes ended in cell C_j (see Fig. 13(c)). Suppose, we afterwards complete net 1 by a path shown in Fig. 13(c). Therefore, net 1 has a redundant segment, $C_i \rightarrow C_j$, which should be deleted.

The procedure to prune the wire route of a completed net is:

- (1) Delete the redundant segments from the wire route of the completed net.
- (2) The returned channel capacities may have the following effects.
 - (2.1) Add vertices to the outermost or inner meshes.
 - (2.2) Merge several components of the cell graph into one component.
 - (2.3) Reverse NPT cells to become non-NPT cells, i.e., the cell status is reversed. For instance, in Fig. 13(c), cell C_j becomes a new NPT cell after we move pins of nets 2,3 and extend net 1's route from C_i to C_j , since there are 3 pins of uncompleted nets and the remaining capacities are 3 on C_j 's 4 boundaries. After net 1 is finished and a redundant segment (c,1)-(c,2) is deleted, cell C_j contains 2 pins of unfinished nets and its channel capacities around it rise to 4. Hence, cell C_j is no longer an NPT cell.

5.4.5. Backtracking

Since our algorithm wires connections along the mesh with only one-step-look-ahead ability, it can not foresee the barriers hidden deeply in the center (Fig. 15). It is possible to make connections in the outer area such that N_i can not be connected. Therefore, we develop a backtracking method which can handle the graph disconnection type problem to remove these poorly placed wires.

Suppose there are two components G_1 and G_2 which are mutually disconnected. Let us assume that nets N_1, N_2, \dots, N_k need to be wired from G_1 to G_2 .

First, we select a set of nets $\{N_i\}$ which satisfy the following conditions.

- (1) N_i crosses G_1 's boundary more than once.
- (2) N_i has no pins in G_1 .
- (3) N_i has a connecting path from G_1 to G_2 , which does not reach any other components (see Fig. 16(a)).

We delete the routes of N_i 's until there are not less than k capacities returned to the G_1 's border (since there are k nets that need to be wired to G_2). The channel capacities gained from wire deletion will merge two components G_1 and G_2 into one, accompanied with the enlargement of routing meshes. Nets N_1, \dots, N_k then are connected from G_1 to G_2 . After that, we update the cell graph and routing meshes and resume the mesh routing for each component. For instance, Fig. 16(a) shows a net N_i crosses G_1 's boundary twice with no pin in G_1 . After N_i 's route is removed, we find a connecting path for N_1 to reach G_2 's border. Then we resume the mesh routing on the two updated components (see Fig. 16(b)-(c)).

5.5. Example

Consider the global routing problem in Fig. 17(a) in which each channel capacity is 2.

Nets 1 and 8 are first connected by the 2-pin-adjacent-cell rule (Fig. 17(b)). Cells (a,2) and (d,1) are NPT cells (Fig. 17(c)). A pin of net N_3 is forced to move down to (b,1) which becomes a new NPT cell, and two pins of N_9 in (c,1) and (d,1) are connected according to the case (3) in section 5.3. Then we add one more

segment from (c,1) to (c,2) to finish N_9 (Fig. 17(d)). The updated cell graph G is shown in Fig. 17(e), whose outermost mesh is in Fig. 17(f). Note that no inner mesh exists at this time. We can create a table (Fig. 17(g)) listing the nets having pins appearing on the mesh, e.g., N_3 has a pin in (b,1).

We sort the table in such an order that the net with least pins on the mesh is first. We first compute the shortest lengths for those nets with 2 pins on the mesh. If none exist, we consider those of 3 pins on mesh, and so on. We find that the pin pairs of N_4 and N_{10} have the shortest distance along the mesh, i.e., (a,2) \rightarrow (a,1) \rightarrow (b,1) for N_4 and (d,1) \rightarrow (d,2) \rightarrow (d,3) for N_{10} . We arbitrarily choose one pair, say N_4 's, to be wired on the mesh. Then we find that neither disconnection nor barrier troubles will occur after N_4 takes that connecting path. Therefore, N_4 is finished. The updated cell graph G and outermost mesh are shown in Fig. 17(h) and (i), respectively.

N_{10} is the next one to be wired along the current mesh. We then apply the extension of 2-pin-adjacent-cell rule to finish N_{10} by adding a segment from (d,2) up to (c,2) (see Fig. 17(j)). N_{12} is the next one to have a route, (d,2) \rightarrow (d,3) \rightarrow (d,4) \rightarrow (c,4), on the mesh. Since cell (d,2) has a pin of unfinished net N_{12} and its remaining channel capacities on boundaries are 2, then (d,2)'s status is updated to be an NPT cell. N_{13} is forced to have a route (d,1) \rightarrow (c,1) (see Fig. 17(k)). Cell (c,1) is thus a new NPT cell.

Now cell (d,1) has only one degree (see Fig. 17(l)). Since N_9 and N_{10} are completed, only N_{13} can have a path extended to cell (d,2). But because cell (d,2) is an NPT cell, N_{13} 's route can not reach (d,2). Therefore, cell (d,1) is deleted from G. Cell (d,2) then has degree 1 so that N_{12} has a route extended to cell (c,2). After that, N_{12} is completed by adding a segment from (c,2) \rightarrow (c,3). The current cell graph G and its outermost mesh are shown in Fig. 17(m)-(n).

By the same procedure, we finally obtain a result shown in Fig. 17(q).

5.6. Summary

Let us summarize the routing steps in our algorithm.

- <1> Make 2-pin-adjacent-cell connections.
- <2> Find NPT cells and barriers.
- <3> Find the unique routes due to the NPT cells, and apply the extension of 2-pin-adjacent-cell rule to the applicable nets.
- <4> Wire pin pairs along the meshes.
- <5> If no disconnection and barrier troubles occur, we update cell graph and create new NPT cells. Repeat checking the neighborhood of NPT cells and applying the extension of 2-pin-adjacent-cell rule to applicable nets. Otherwise, use backtracking method to remove wrong wires.
- <6> Prune wire route of the completed nets.
- <7> Go back to step <4> until no more connections can be imbedded.

To conclude this section, we point out some features in our algorithm.

- (1) In each iteration, we deal with a problem of smaller size. The detected unique routes are identified prior to any connections along meshes.
- (2) We do not complete one net at a time. The wire route of each net grows from outside toward inside gradually in order to even out the wire distribution in the style of spanning tree.
- (3) The rip-up and shove aside rerouting [8, 9] are done automatically.
- (4) The look-ahead ability helps eliminate wrong connections made in the outer area. A simple backtracking method removes the wrong wires and connect those "difficult" nets.

6. Complexity and a Special Case

The main step in our algorithm is to search the shortest path for the pin pairs along the meshes and check if the graph disconnection or NPT barriers will separate the connection of some net. Let us assume that (1) the dimension of the given cell array is $p \times q$, (2) the total number of nets is n , (3) the maximum number of pins in each net is σ . Since we connect a pair of pins each time, it needs $(\sigma - 1)$ steps to complete one net. In each routing step we do the following things.

First, we sort the nets having pins on the meshes in increasing number of pins order. We will need $n \times \log n$ comparisons at most. Second, the complexity of calculating the shortest path between pin pairs along the meshes is $O(n \times (p+q))$. The effort to find the shortest one among them is $O(n)$. The one-step-look-ahead requires $O(p \times q)$ effort for each net. In worst case we need to check n nets. Therefore, in each routing step the complexity to connect a pair of pins on the mesh is $O(n \times \log n + n \times (p+q) + n \times p \times q)$. The approximate total computation effort is $O(n^2 \times (\log n + p \times q))$.

Besides showing the general global routing problem to be NP-complete, we find a special case and its extension which can be solved by the max-flow min-cut technique.

Two Pin Shortest Wire in Manhattan Geometry (2-pin SWMG)

Instance: Assume there are 2-pin nets distributed in an $M \times N$ cell array in such a way that for each net, the left hand pin is not below the right hand pin or both pins are on a same vertical line. The channel capacity on each boundary is also given.

Question: Can each net on the chip be routed by the shortest wire in Manhattan geometry such that all the channel capacity constraints are satisfied?

Fig. 18(a) shows the possible distribution of pins and their allowable wire layouts. Fig. 18(b) shows some wire layouts which are not allowed in the special case. We find that the special case can be formulated as a network flow problem so that it is polynomially solvable.

Lemma 4. 2-Pin SWMG can be solved in the polynomial time.

Proof: Let us construct a directed graph $H = (V_h, E_h)$, where each vertex v_i in $V_h - \{S, T\}$ corresponds to a cell and S, T represent the source and sink, respectively. The edge $(v_i \rightarrow v_j)$ is directed from v_i to v_j if v_i and v_j are two adjacent cells with v_i to the left of v_j or above v_j . A capacity which is equal to the channel capacity on the corresponding cell boundary is assigned to each edge.

An edge $(S \rightarrow v_i)$ is created between source S and v_i if, in the corresponding cell of v_i , we have the left-hand (or top) pins of some nets. A capacity is assigned to $(S \rightarrow v_i)$, which equals the number of left-hand and top pins in v_i . We create an

edge ($v_i \rightarrow T$) between v_i and sink T if there are the right-hand (or bottom) pins in v_i and the number of such nets equals the capacity assigned to this edge. Fig. 18(c)-(d) shows an example with a constructed digraph.

The maximum flow in the digraph H can not be greater than the sum of capacities of the edges directed to the sink. Since each routing of a net defines a flow from source to sink, the maximum flow in H is equal to the number of completed nets. Therefore, the routings can be realized for all the nets in the array if and only if the maximum flow in H is equal to the sum of capacities of the edges directed to sink. Thus this special case can be solved in polynomial time (see [10]).

For the multi-pin case, we also find that if the nets can be decomposed into 2-pin subnets such that the pin distribution follows the same constraint as above, then it is solvable by max-flow method.

For instance, net 1 has five pins which can be decomposed into four 2-pin subnets (Fig. 19). For each subnet, (1) the left hand pin is not below the right hand pin or both are on a vertical line and (2) the smallest enclosing rectangle intersects those of other subnets at no more than one point. If every multi-pin net can be decomposed into such subnets, then the problem is transformed into a 2-pin SWMG problem. Therefore, we can apply max-flow method to this multi-pin case.

7. Experimental Results

Table 1 shows some results. The first example is randomly generated. The second one is a realistic example. Fig. 20(a) shows the pin distribution of the second example. Fig. 20(b) shows the wire routes of 464 nets. From that, we can see that the outer area has higher density than the inner area.

From the time complexity analysis, the most time consuming step is the one-step-look-ahead check. If the pin distribution and channel capacities are uniform, then we may not need the look-ahead check in order to speed up the process.

The heuristic algorithm is implemented as part of our gate array layout system.

8. Conclusion

We analyze the gate array global routing problem and point out the intrinsic difficulties of finding an optimal solution. A heuristic algorithm is developed using a new "mesh routing" method as well as some rules for identifying the unique routes. The algorithm can handle non-uniform situations on the chip so that it is suitable for gate array chip with built-in circuits (ROM, RAM, etc.). We also point out the features of mesh routing which will reduce the congestion on the chip and avoid the one-net-at-a-time ordering problem. This makes chips more routable. Since it is still possible to make wrong connections, we also propose a backtracking method to reroute. Further optimization on the program and the combination of mesh routing and hierarchical approach [4] are under study.

9. Acknowledgment

The authors wish to thank Professor E.S. Kuh for his valuable guidance and encouragement. This work is supported by the National Science Foundation under Grant ECS - 8201580, and the Joint Service Electronics Program F49620-79-C-0178. The first author wishes to thank Bell Labs for their financial support.

10. Reference

- [1] K.A. Chen, M. Feuer, K.M. Khokmani, N. Nan and S. Schmidt, "The Chip Layout Problem: An Automatic Layout Procedure", Proceedings of 14th Design Automation Conference, New Orleans, June, 1977
- [2] B.S. Ting and B.N. Tien, "Routing Techniques for Gate Array", IEEE Trans. on CAD of ICs and Systems, Oct. 1983.
- [3] T. Matsuda, T. Fujita, K. Takamizawa, H. Mizumura, H. Nakamura, F. Kitajima and S. Goto, "LAMBDA: A Quick, Low Cost Layout Design System for Master-Slice LSIs", Proc. 18th Design Automation Conference, 1982, pp. 802-808.
- [4] M. Burstein and R. Pelavin, "Hierarchical Wire Routing", IEEE Trans. on CAD of ICs and Systems, Oct. 1983.

- [5] R.M. Karp, R. Rivest, C.D. Thompson, U. Vazirani and V. Vazirani, "Global Routing in Gate Arrays", to appear.
- [6] R. Raghavan, J. Cohoon and S. Sahni, "Manhattan and Rectilinear Wiring", Technical Report 81-5, March 1981, Computer Science Dept., Univ. of Minnesota.
- [7] D.S. Johnson, "The NP-Completeness Column: An Ongoing Guide", Journal of Algorithms, 3, 381-395 (1982).
- [8] W.A. Dees, R.J. Smith, "Performance of Interconnection Rip-Up and Reroute Strategies", Proceeding of 18th Design Automation Conference, 1981.
- [9] W.A. Dees, P.G. Karger, "Automated Rip-Up and Reroute Techniques", Proceeding of 19th Design Automation Conference, 1982.
- [10] S. Even, "Graph Algorithms", Chapter 5, Computer Science Press, 1979.
- [11] M. Marek-Sadowska and J.T. Li, "Global Router for Gate Array", Proc. of 1st ICCAD Conference, 1983.

	Example 1	Example 2
Total # of nets	267	464
Total # of pins	723	1693
Total # of completed nets	267	464
Estimated Manhattan length of all pin-pair connections (independent of the channel capacities).	935	5132
Total available track length (2 layers)	1620	8960
Actual connection length made by program	1033	5366
CPU time (min)	16	82

Table 1.

Figure Captions

Fig. 1. An example of global routing problem for gate array.

- (a) Pins of 6 nets distributed in a 3*3 array. The numbers in parenthesis show the channel capacities.
- (b) A solution of (a).

Fig. 2. Two solutions for a global routing example.

- (a) A global routing problem.
- (b), (c) Two solutions for (a).
- (d), (e) Feasible routes of net N_1 .
- (f), (g) Feasible routes of net N_2 .
- (h) Infeasible route for net N_2 .
- (i) N_1 's route forces N_2 to take route R_1^2 .
- (j) If the solid line connection is made for N_2 , N_1 has to take pattern R_1^1 .

Fig. 3. 2-pin-adjacent-cell rule.

- (a) A solution for N_i and N_j .
- (b) N_i takes the shortest connection.

Fig. 4. The number of pins asking for passage exceeds the total channel capacities along its boundary.

- (a) There are 8 pins inside the closed region (outlined in boldface) and 5 capacities are on its boundary. Hence there are 6 pins connected inside.
- (b) C_a and C_b are NPT cells with capacity 2 on the common boundary.
- (c) Pins of nets N_1 and N_4 are connected together.

Fig. 5. A barrier for net N_3 .

- (a,1) and (b,1) form a barrier for net N_3 .

Fig. 6. Generalized routing.

- (a) A wire route for N_1 .
- (b) A generalized routing for N_1 .

Fig. 7. Cell graph.

A planar drawing of the cell graph for Fig.1(a).

Fig. 8. Outermost and inner meshes.

- (a) A problem with 0 channel capacity on some boundaries.
- (b) The outermost and inner meshes.
- (c), (d) A problem with overlapping outermost and inner meshes.

Fig. 9. Barriers detected by the algorithm.

- (a), (b) The NPT cells group like a wavefront barriers.
- (c). The cell graph is disconnected.

Fig. 10. Unique routes detected by the algorithm.

- (a) A pin of N_1 in cell (c,1) is forced to move up. Two pins of N_2 in NPT cells (a,4) and (b,4) are connected together. Two pins of N_3 are wired together, where (c,2) is an NPT cell.
- (b) N_1 is completed according to the case (3) of section 5.3. After that, cell (b,1) becomes an NPT cell so that a pin of N_4 in (a,1) is moved to (a,2).
- (c) N_3 is finished according to the extension of 2-pin-adjacent-cell rule.

Fig. 11. Selecting a pin pair to be connected on the mesh.

- (a) A global routing problem.
- (b) Its corresponding outermost mesh.
- (c) A table listing the nets with pins on the mesh by which the table is sorted.

Fig. 12. Disconnection and barrier troubles.

- (a), (b) N_i is to be connected by the wavy path. Hence, cell graph is disconnected so that N_j can not be completed.

(c), (d) N_i wants to take the wavy path. But N_j can not be completed.

Fig. 13. Redundant segment.

(a), (b) N_2 has an unique route due to NPT cells. Therefore, cell C_i has only one degree.

(c) N_1 , N_2 and N_3 's routes reach C_j .

Fig. 14. Mesh expansion.

Outermost mesh is expanded inward.

Fig. 15. A net difficultly to be connected.

N_i has a pin hidden so deep in the center that we can not detect the unique route for it in the beginning.

Fig. 16. Backtracking.

(a) N_i crosses G_1 's boundary twice with no pin in there.

(b) Delete N_i 's route to merge G_1 and G_2 . Then N_1 is connected to G_2 's border.

(c) After that, we have two new components G_1' and G_2' .

Fig. 17. Example.

(a) A global routing problem.

(b) N_1 and N_8 are connected by the shortest routes.

(c) Cell graph and NPT cells.

(d) N_3 has a forced route.

(e) Updated cell graph.

(f) Outermost mesh of (e).

(g) A table of nets which have pins on the mesh.

(h), (i) Updated cell graph and outermost mesh.

(j) The connections already made.

(k) Cell (d,2) becomes an NPT cell so that N_{13} has a forced route.

(l) Cell graph for (k).

- (m) Delete (d,1) from cell graph followed by cell (d,2).
- (n) The outermost mesh for (m).
- (o) Final result.

Fig. 18. 2-pin SWMG special case.

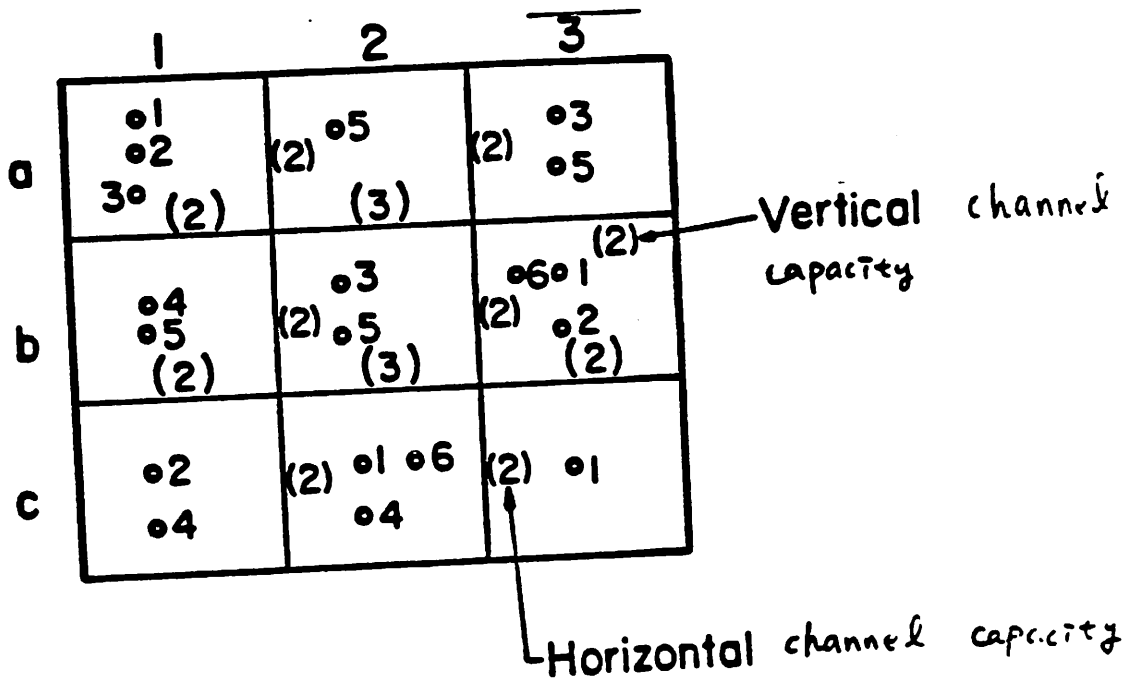
- (a) Allowable patterns in 2-pin SWMG.
- (b) Patterns not allowed.
- (c) An example of 2-pin SWMG.
- (d) The digraph for (c).

Fig. 19. Multiple pin SWMG case.

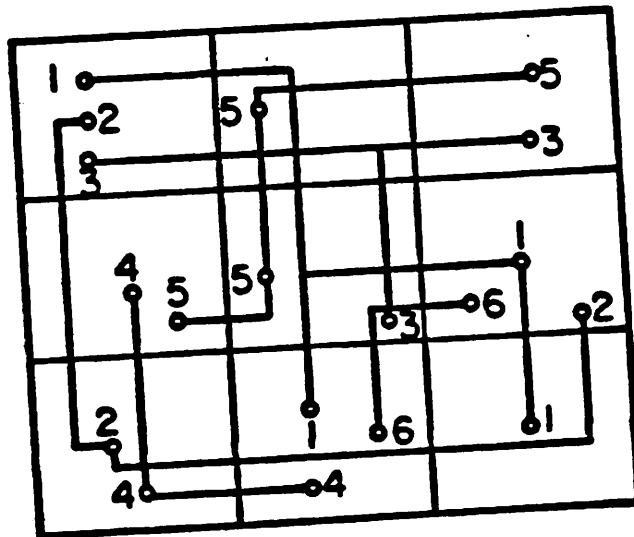
A 5-pin net is decomposed into 4 2-pin subnets.

Fig. 20. A realistic example.

- (a), (b) The pin distribution and final result of Example 2.

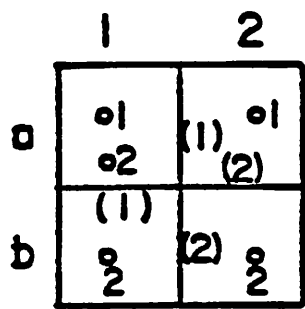


(a)

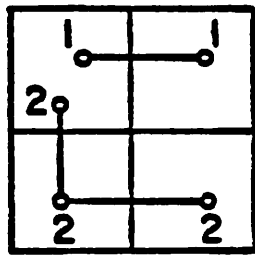


(b)

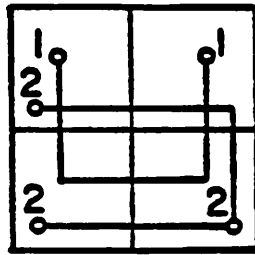
Fig.1



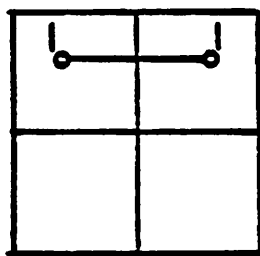
(a)



(b)

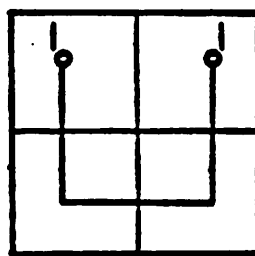


(c)



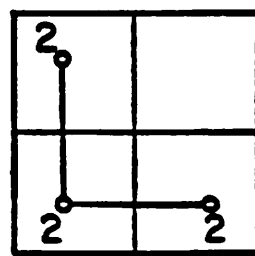
R_1^1

(d)



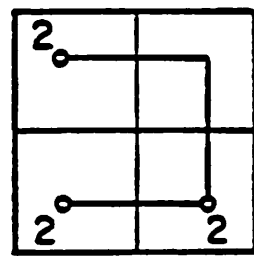
R_2^1

(e)



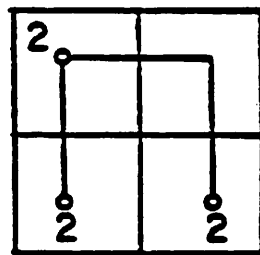
R_1^2

(f)

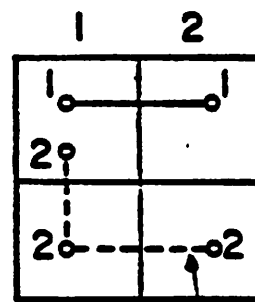


R_2^2

(g)

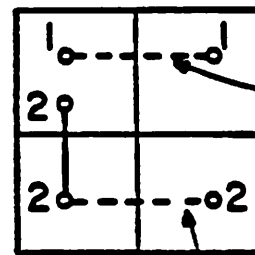


(h)



R_1^2

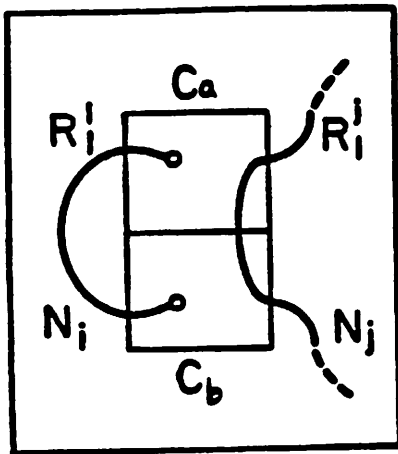
(i)



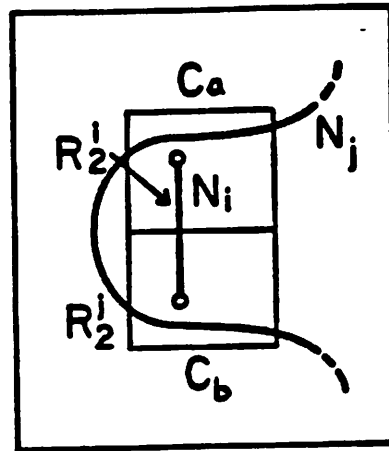
R_1^2

(j)

Fig. 2

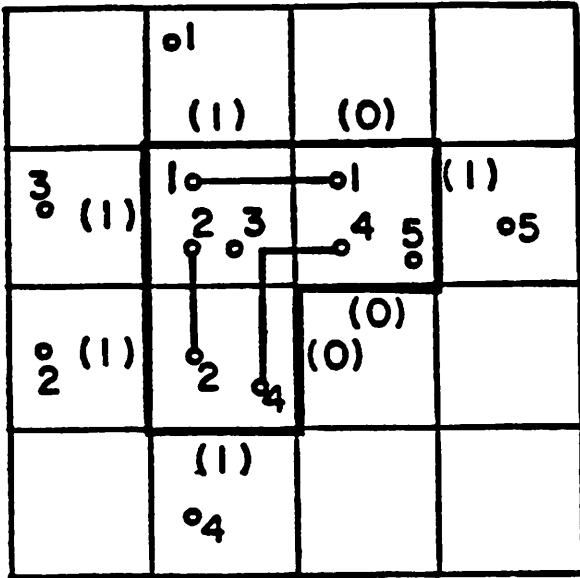


(a)

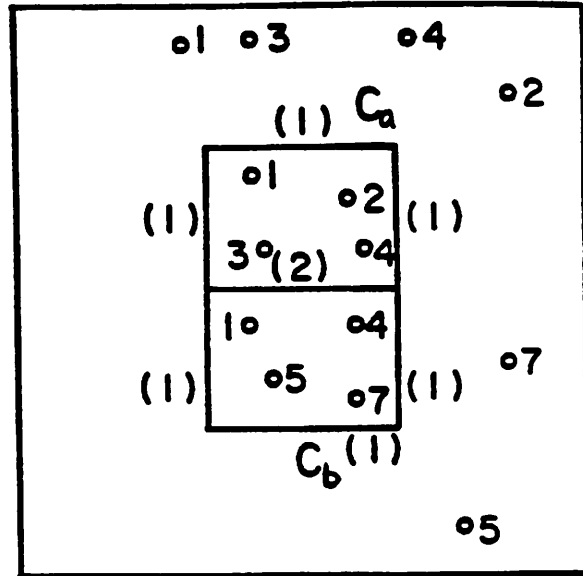


(b)

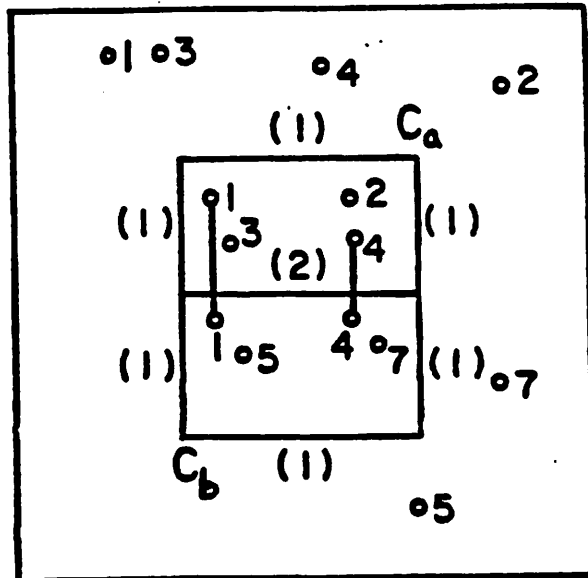
Fig. 3



(a)



(b)

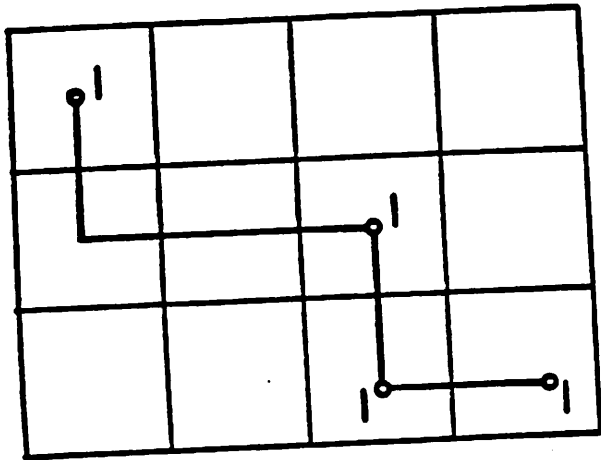


(c)

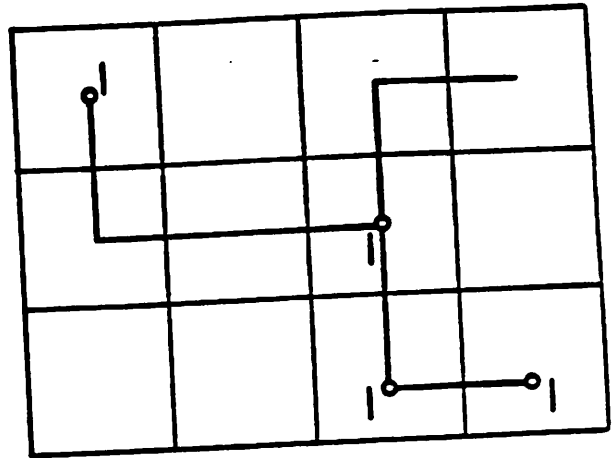
Fig. 4.

	1	2	3
a	01 (5)	(1)	01
b	02	(1) 03	02
c	(1) 03		

Fig 5.



(a)



(b)

Fig. 6.

	1	2	3
a		(2)	(2)
	(2)	(3)	(2)
b		(2)	(2)
	(2)	(3)	(2)
c			(2)
	(3)		

Fig. 1. (a)

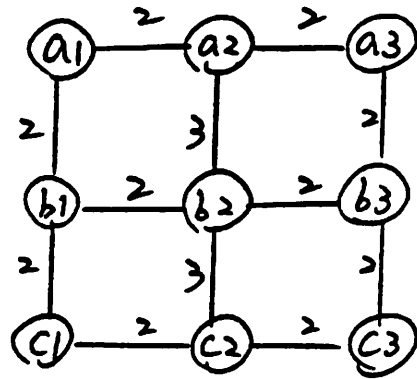
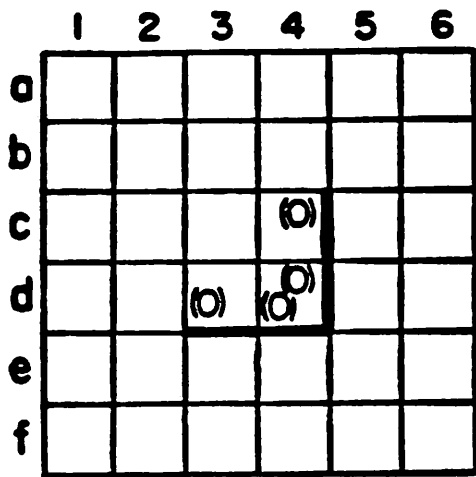
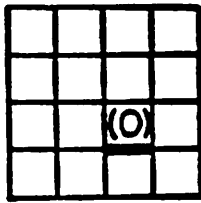


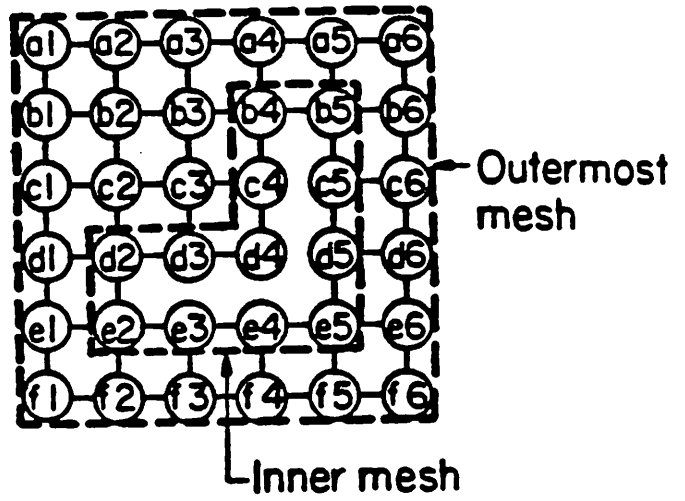
Fig. 7.



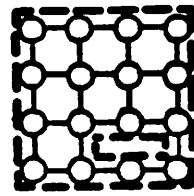
(a)



(c)



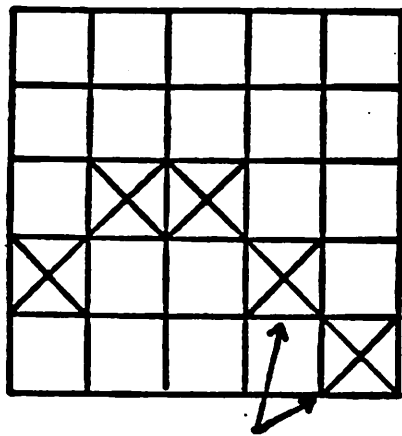
(b)



(d)

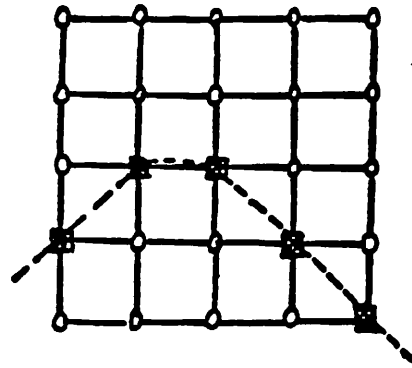
Inner and outermost meshes overlap

Fig. 8.



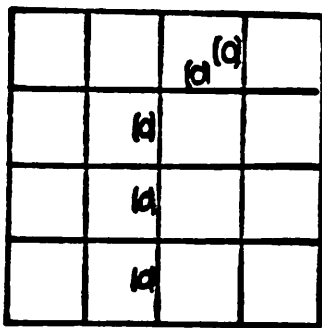
NPT cell

(a)

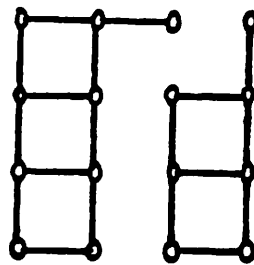


Barrier

(b)

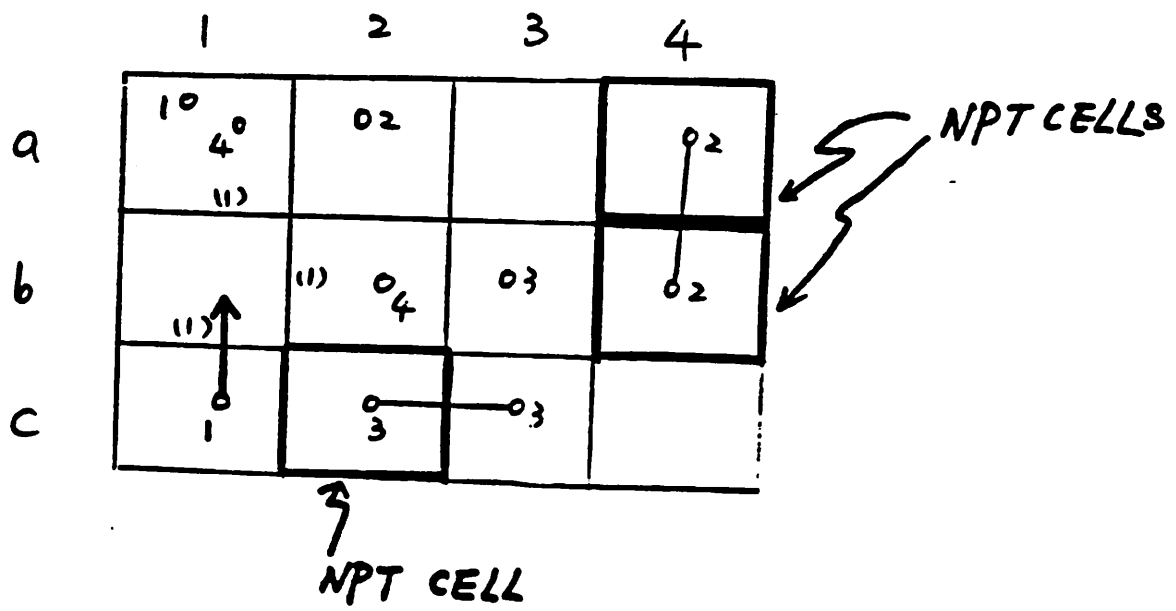


(c)

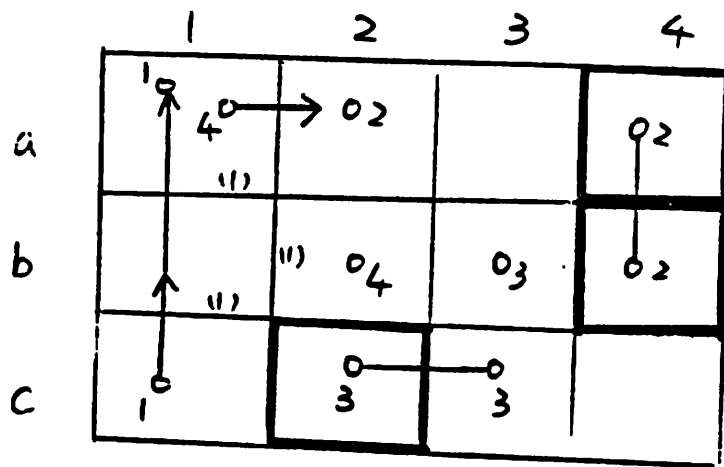


(d)

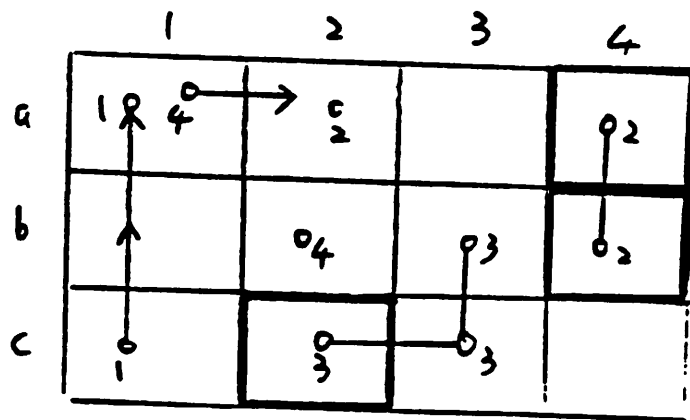
Fig. 9.



(a)



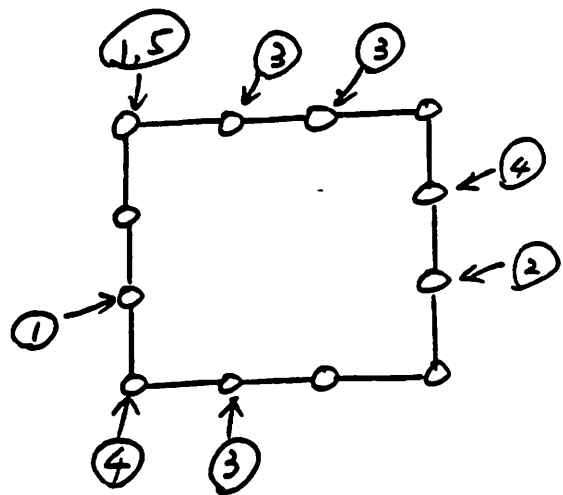
(b)



(c)

Fig. 10

	1	2	3	4
a	01 05	03	03	
b		02	04	04
c	01		05 02	02
d	04	03		



a)

b)

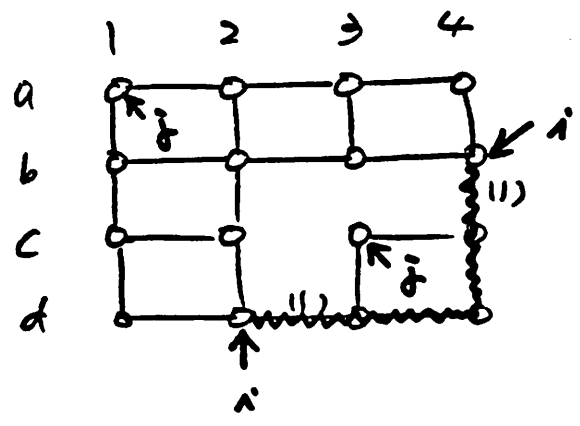
NET	PINs
2	(c, 4)
5	(a, 1)
1	(a, 2) (c, 1)
4	(d, 1) (b, 4)
3	(a, 2) (a, 3) (d, 2)

c)

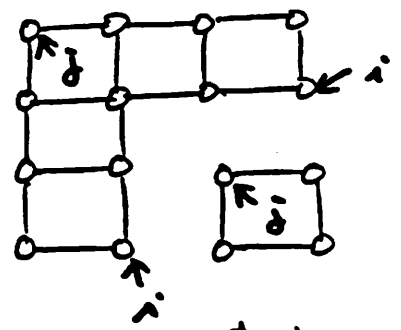
Fig. 11.

row

column

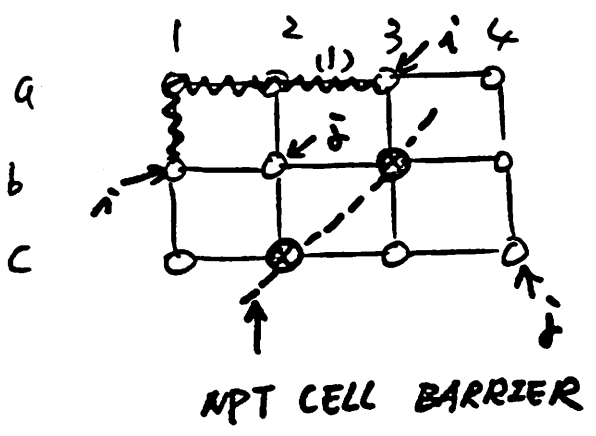


(a)



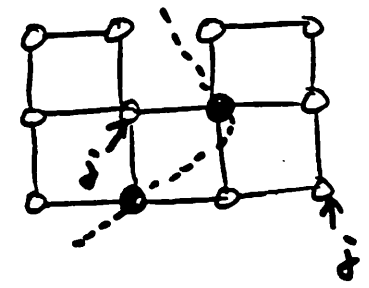
Net j can not be connected

(b)



NPT CELL BARRIER

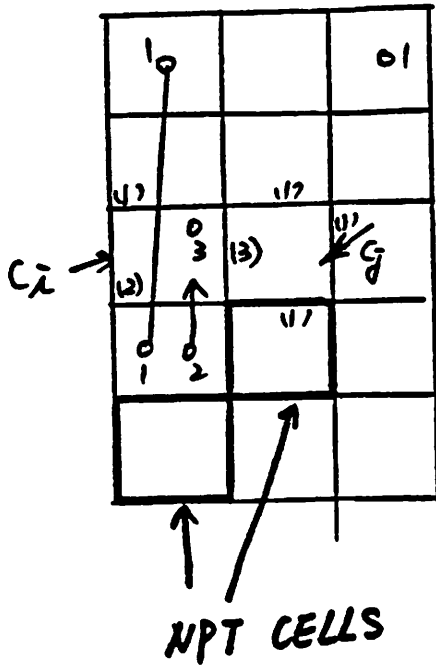
(c)



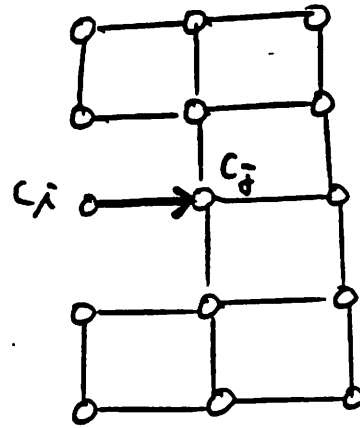
Net j can not be connected

(d)

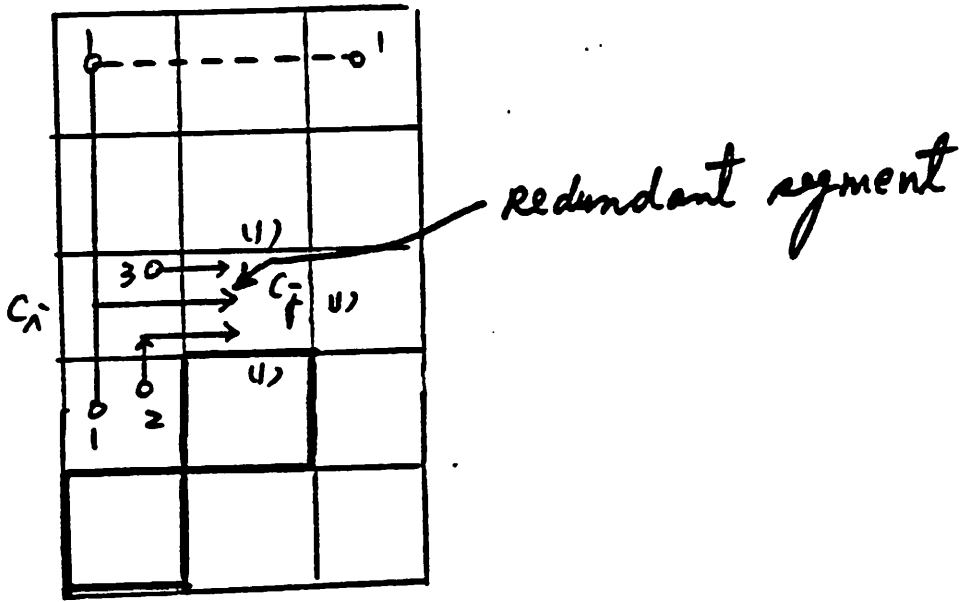
Fig. 12



(a)



(b)



(c)

Fig. 13

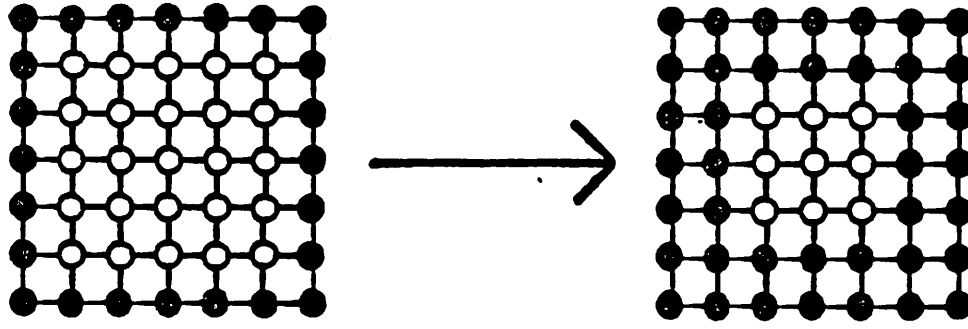
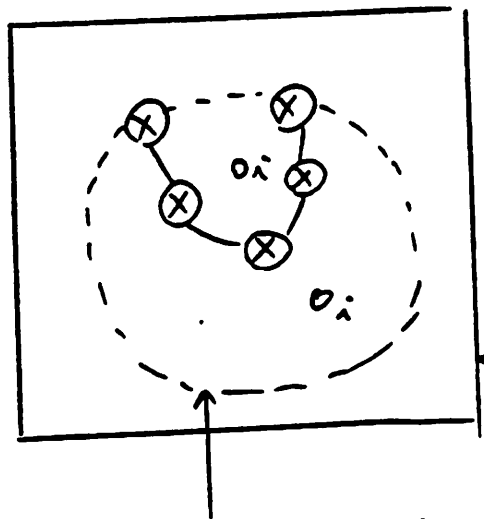


Fig. 14

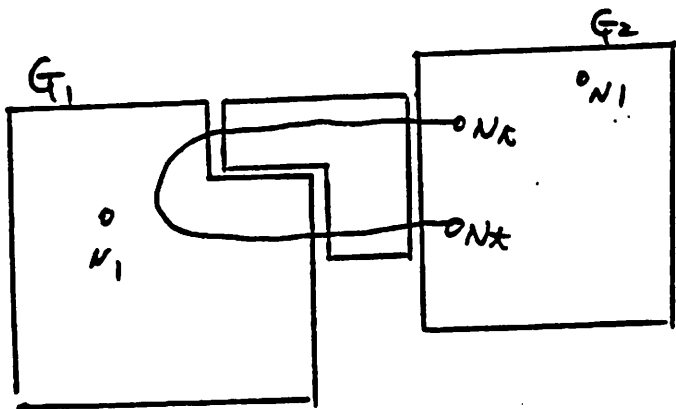


Net i has a pin hidden inside the barrier.

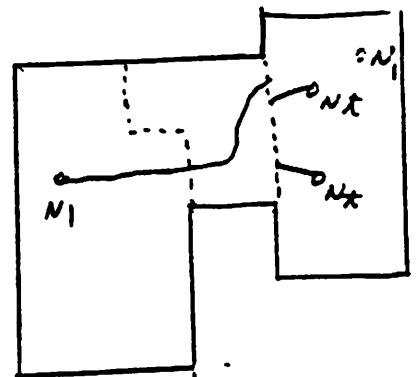
original outermost mesh.

New outermost mesh

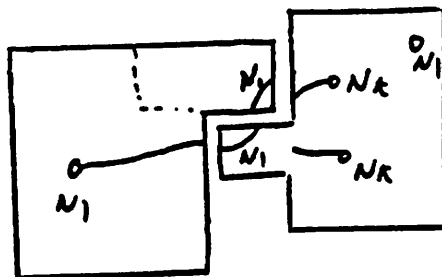
Fig. 15.



(a)



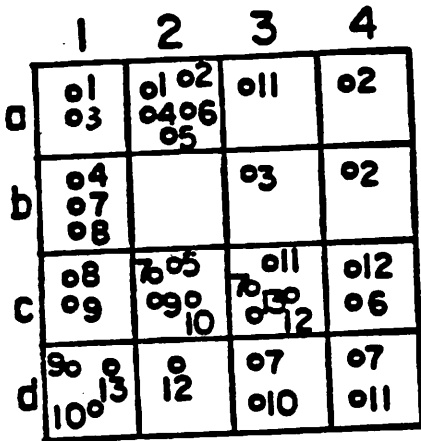
(b)



(c)

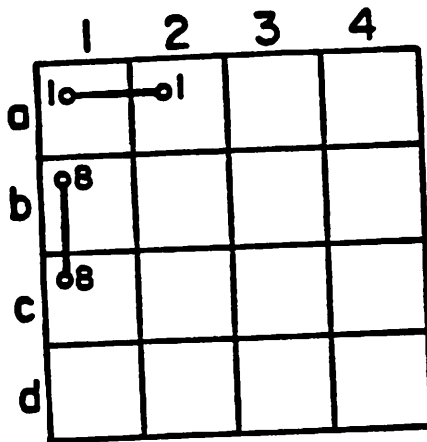
Fig. 16

Fig. 17

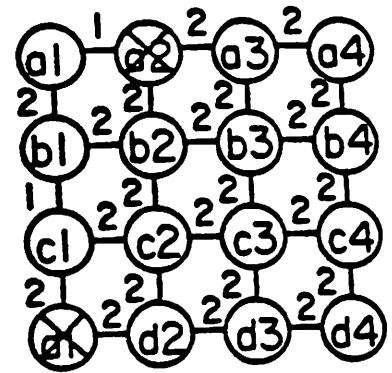


(a)

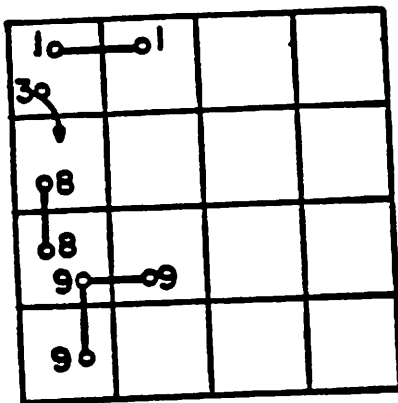
All channel capacities are set to 2.



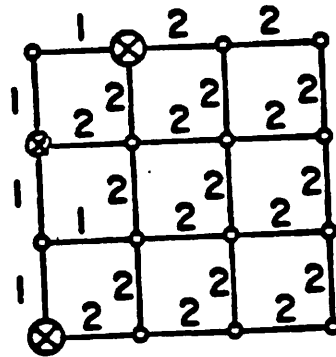
(b)



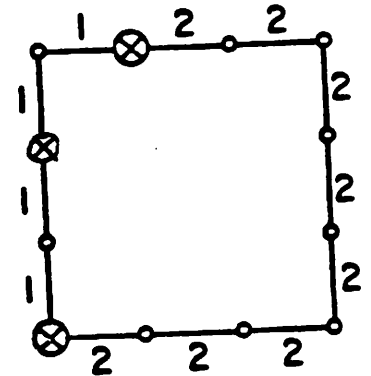
(c)



(d)



(e)



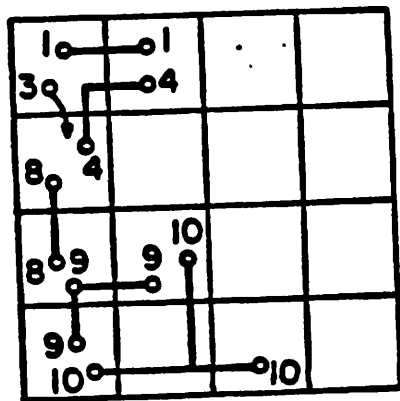
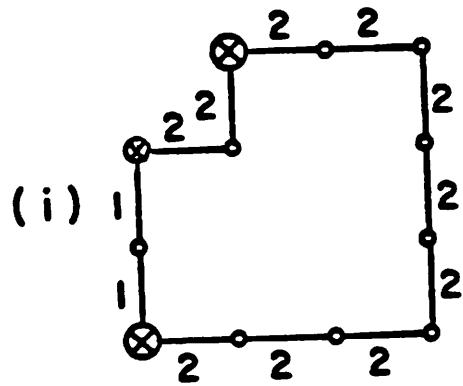
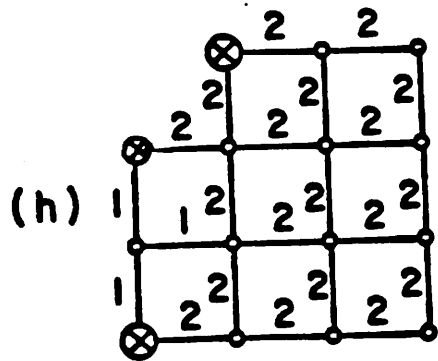
(f)

Fig. 17. (CONTI.)

Net	CELL
3	(b, 1)
5	(a, 2)
13	(d, 1)
11	(d, 4) (a, 3)
4	(a, 2) (b, 1)
6	(a, 2) (c, 4)
10	(d, 1) (d, 3)
12	(d, 2) (c, 4)
2	(a, 2) (a, 4) (b, 4)
7	(b, 1) (d, 3) (d, 4)

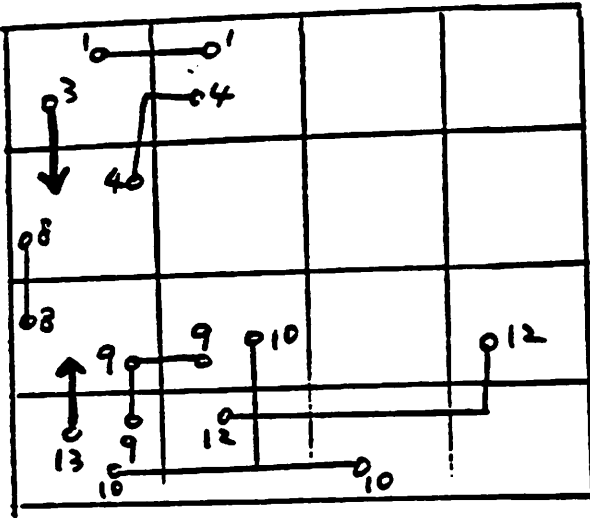
(g)

Fig. 17. (CONTZ)

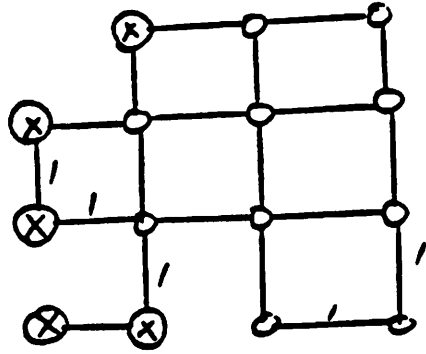


(j)

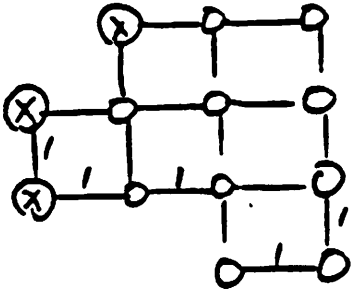
Fig 17 (CONTZ)



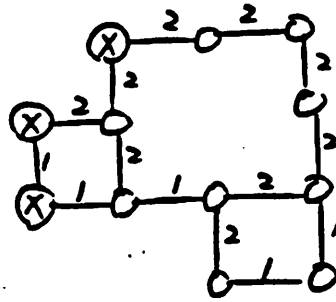
(k)



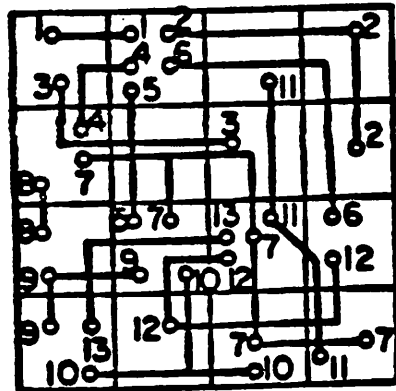
l) cell graph of Fig. 12(k)



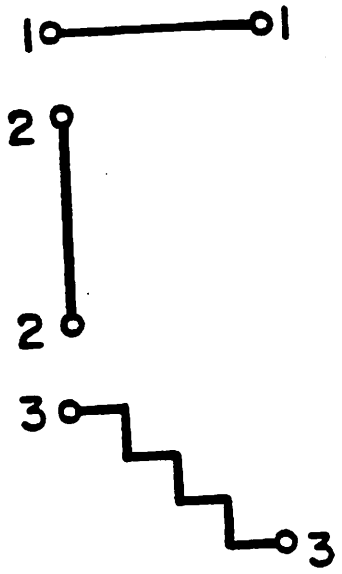
(m)



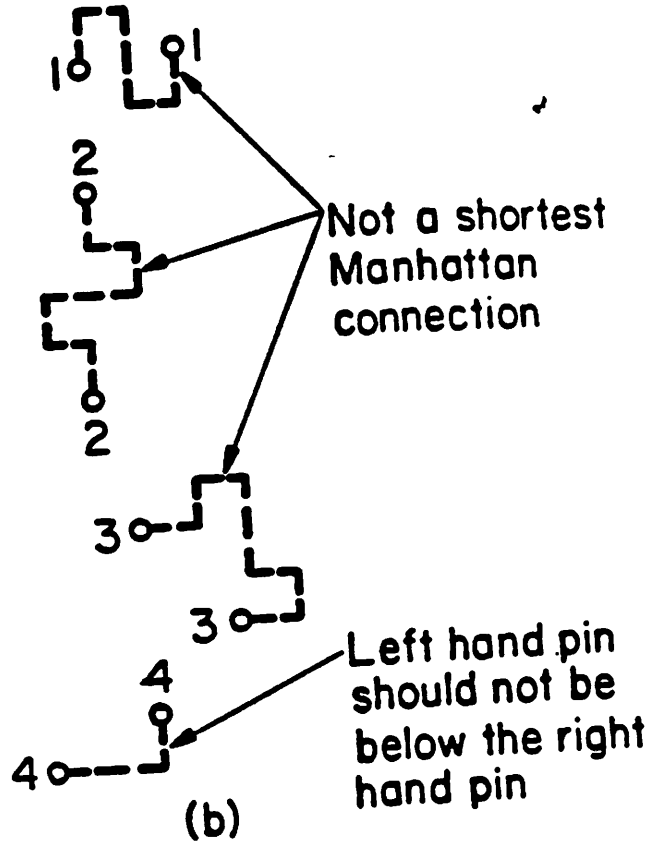
(n)



(o)



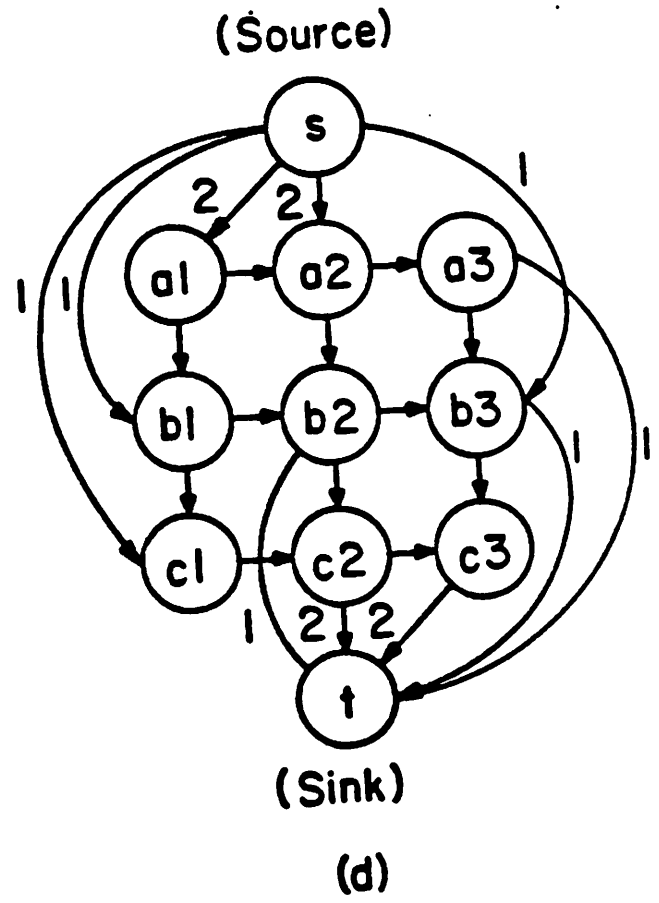
(a)



(b)

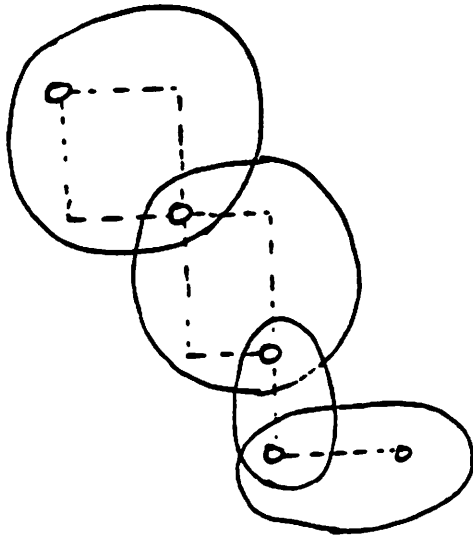
	1	2	3
a	○1 ○2	○5 ○6	○5
b	○3	○2	○6 ○7
c	○4	○3 ○4	○1 ○7

(c)



(d)

Fig. 18



NET 1 IS DECOMPOSED
INTO 4 SUBNETS
THEIR ENCLOSING RECTANGLES
INTERSECT AT MOST AT ONE
POINT.

Fig 19.

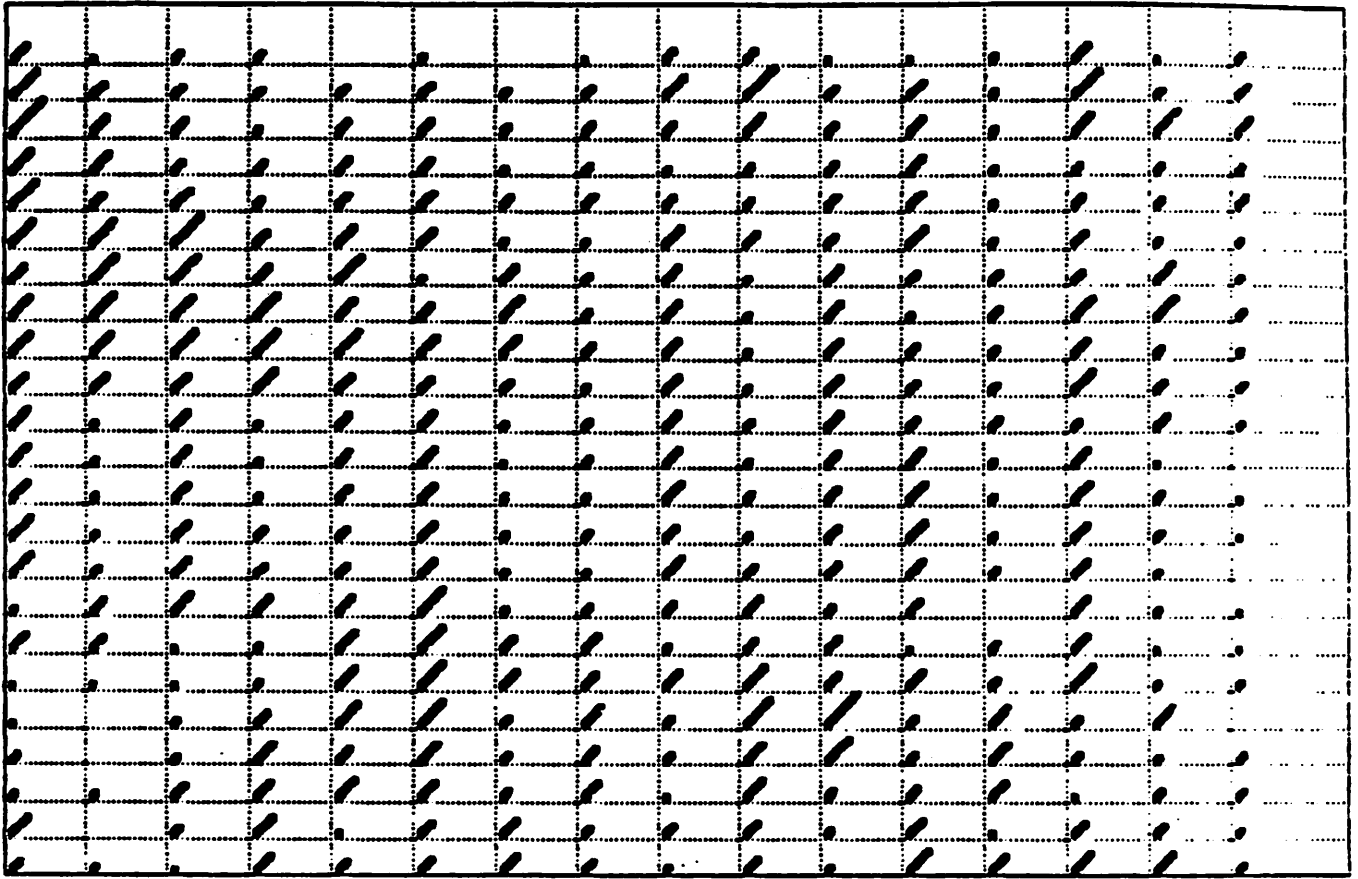
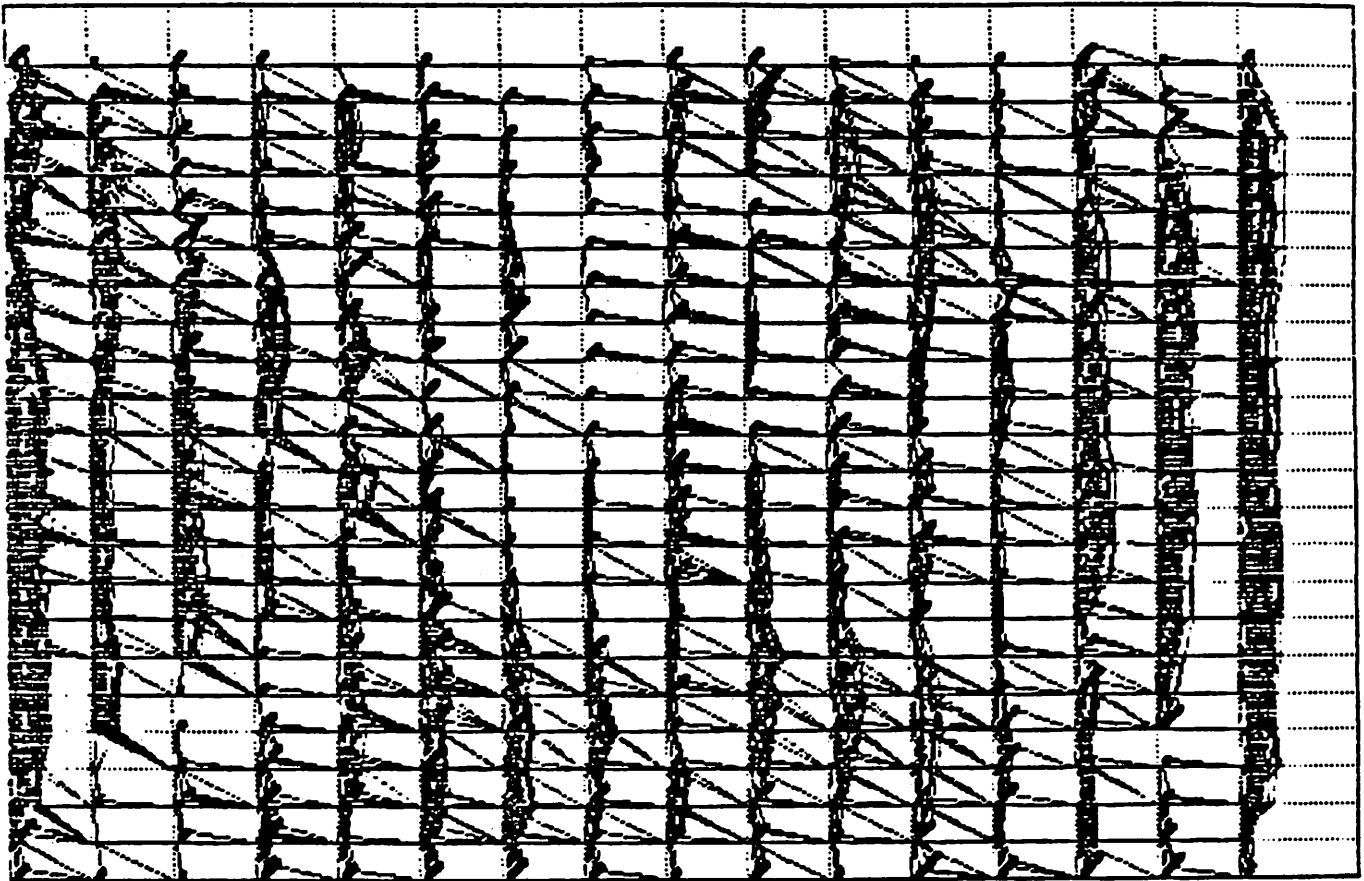


Fig. 20 (a)



No. finished nets = 464

Fig.20 (b)