A UNIFIED APPROACH TO THE VIA MINIMIZATION PROBLEM

by

Xiao-Ming Xiong and Ernest S. Kuh

Memorandum No. UCB/ERL M87/80

20 November 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A UNIFIED APPROACH TO THE VIA MINIMIZATION PROBLEM

by

Xiao-Ming Xiong and Ernest S. Kuh

Memorandum No. UCB/ERL M87/80

20 November 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A UNIFIED APPROACH TO THE VIA MINIMIZATION PROBLEM

by

Xiao-Ming Xiong and Ernest S. Kuh

Memorandum No. UCB/ERL M87/80

20 November 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Unified Approach to the Via Minimization Problem

Xiao-Ming Xiong and Ernest S. Kuh

Electronics Research Laboratory
Department of EECS
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

A unified via minimization approach is presented for two layer routing of printed circuit boards and VLSI chips. We have analyzed and characterized different aspects of the problem and have derived an equivalent graph model for the problem from the linear programming formulation. Based on the analysis of our unified formulation, we posed a fast practical algorithm. The algorithm can handle both grid-based and gridless routing. Also, an arbitrary number of wires is allowed to intersect at a via and we allow both Manhattan and lock-knee routings.

November 17, 1987

A Unified Approach to the Via Minimization Problem

Xiao-Ming Xiong and Ernest S. Kuh

Electronics Research Laboratory
Department of EECS
University of California, Berkeley
Berkeley, California 94720

1. Introduction

Via minimization is an important step in both VLSI and PCB design, because vias not only increase the manufacturing cost but also reduce the reliability of the products. Depending on whether the via minimization step is done before or after the routing, we divide the via minimization problem into *unconstrained via minimization* (UVM) and *constrained via minimization* (CVM).

1. Constrained Via Minimization (CVM)
   Given a collection of paths representing the interconnection wires without specifying what layer they belong, find a layer assignment to all the path segments that will minimize the total number of vias used.
2. Unconstrained Via Minimization (UVM)
   Given a routing region and a set of terminals of signal nets, find the topological routing result and layer assignment that will minimize the number of vias used.

In practical VLSI or PCB design, the via minimization problem does not stand alone. We need to trade off between via minimization and wire length minimization, chip performance consideration and other constraints. For example, we may want to increase the total number of vias to get rid of some vias on the critical path or fix the layers for some nets to guarantee the chip performance.

We have developed a new topological routing method for the unconstrained via minimization problem in order to formulate both the CVM and UVM problems as a {0, 1} linear programming problem. However, due to the length limitation on the paper, we will discuss only the CVM problem. The UVM problem will be treated separately [24].

The via minimization problem is formulated as a {0, 1} linear programming problem. In our formulation, all arbitrary number of wires can intersect at a via and both Manhattan and knock-knee routings can be handled. In addition to via minimization, other constraints can be easily brought into consideration by modifying the objective function and constraint conditions. Thus we have unified the formulation for via minimization. To solve the problem, a practical algorithm is given which does not always guarantee the optimal solution but works very well in general.

2. Previous Work

The via minimization problem arose from printed circuit board (PCB) design, and was first formulated by Hashimoto and Stevens in 1971 [1]. The problem was believed to be an NP-complete problem for a long time. Various heuristic approaches were

This research is sponsored by the Semiconductor Research Cooperation under SRC 82-11-008.

applied to attack the problem ([1], [2], [3], and [4]). In 1980, Kajitani [5] presented a polynomial-time algorithm for finding a max-cut in a planar graph to minimize vias if the vias are only allowed at corners of paths (i. e. at places where a path changes its direction in a Manhattan routing design). In another paper, Ciesielski and Kinnen [6] gave an integer programming solution to the case in which vias are allowed wherever they fit, but the time complexity of their solution is exponential. Chan, Kajitani and Chan ([7], [8] and [9]) extended the approach in [5] to handle the case where three wires can intersect at a via and proposed an algorithm which can find the optimal solution in polynomial time. Independently, Pinter [10] found a polynomial algorithm to minimize the vias which can connect two wires, but the position of the vias can be anywhere they fit. The two optimal algorithms proposed in [9] and [10] were not implemented by the original authors. In 1986, Du and Chang [11] proposed a heuristic algorithm based on bipartiting the graph. And in 1987, Nakajima, Naclerio and Masuda [12] claimed that they successfully implemented the optimal algorithms in [9] with $O(n^3)$ time complexity.

In 1983, Hsu [13] presented a heuristic algorithm based on a net intersection graph to attack the UVM. In 1984, Marek-Sadowska [14] proved that the UVM problem in the simplest case, in which all signal nets are two-pin nets, is NP-complete and proposed a heuristic algorithm based on planarizing a net intersection graph.

In practice, minimizing the number of vias in an artificially defined small routing region, e. g. a channel or a switch box, is not enough, because the more complicated case is the interconnection of these small routing regions. What we really want is to minimize the number of vias in the entire chip. This puts very serious restrictions on via minimization algorithms. If n is the number of layout objects in the chip, even an $O(n^2)$ algorithm will cause serious space and run-time overflow problems. Therefore, a common way to do via minimization is to depend on pattern recognition and a few simple heuristics to reduce the number of existing vias in the entire chip. These kinds of techniques do not give bounds on the solution and time complexity, but have widely practical applications.

3. Problem Analysis

3.1. Terminology and Assumptions

First, let us assume that there are two interconnection layers for routing.

A *via candidate* is an intersection of the wires belonging to the same net or a maximal piece of wire which does not overlap any other wire and can accommodate at least one via.

The *degree* of a via candidate is the number of wires connected to the via candidate.

A *wire segment* is a piece of wire which has two end points attached to either terminals or via candidates.

A *cross point* is the intersection of wires belonging to different nets.

A *knock-knee point* is the overlapping point of two corners of two paths belonging to different nets.

EXAMPLE

In Fig. 1 (a), both cross points and knock-knee points are marked. The C_i's are via candidates and W_i's are wire segments. To simplify the discussion, let us assume that

terminals can be accessed on both layers. If a via candidate has degree one, we will merge the via candidate with the wire segment attached to it and delete the via candidate. For example, the via candidates C_4, C_5, C_6 and C_7 in Fig. 1 (a) are merged to wire segments W_9, W_1, W_4 and W_6 respectively in Fig. 1 (b).

From then on, we can assume that all via candidates have degree greater or equal to two.

3.2. Formulation of the problem

Lemma 3.1:

A layer assignment is feasible if and only if all the intersecting wire segments of different nets are assigned to different layers.

Proof:

Necessity: If any two wire segments belonging to different nets and intersecting each other are assigned to the same layer, the two signal nets will short-circuit at the intersecting point. Thus the layer assignment is illegal.

Sufficiency: If all the intersecting wire segments of different nets are assigned to different layers, then no short-circuit will occur. We can use vias to connect all the wire segments belonging to the same net but assigned to different layers (by definition of wire segment). So there is no open-circuit in the layout. Thus the layer assignment is feasible.

Corollary 3.1:

A layer assignment is feasible if and only if the wire segments attached to the same cross point and/or knock-knee point are assigned to different layers.

Now we can formulate the via minimization problem. For each wire segment, we assign a {0, 1} variable x_{nl} to it, where the first subscript n designates that the wire segment belongs to net n and the second subscript l is the label of the wire segment.

The value of x_{nl} determines the layer (either 0 or 1) on which the wire segment will be placed. By corollary 3.1, we have a constraint for every cross point and/or knock-knee point as follows:

$$x_{ni} + x_{mj} = 1 \qquad \text{--- (*)}$$

where $n \neq m$ and $i \neq j$.

We can directly obtain the following corollary from corollary 3.1.

Corollary 3.2:

A layer assignment is feasible if and only if the linear equations in (*) hold.

Notice that there are exactly two wire segment intersecting at a cross point and/or knock-knee point. Therefore, for each constraint, exactly two variables will be involved.

A via candidate may or may not be a via depending on the layers assigned to the wire segments attached to it. If all the wire segments attached to a via candidate are assigned on the same layer, the via candidate will not be a via. Otherwise, a via will be resulted if the wire segments attached to the via candidate are not assigned to the same layer. For any pair of wire segments attached to the same via candidate, a term

$$|x_{ni} - x_{nj}|$$

is used in our objective function. If both x_{ni} and x_{nj} are assigned to the same layer, this term will be zero. Otherwise, this term will be non-zero to reflect that a via is needed if x_{ni} and x_{nj} are assigned to different layers. Since we allow all arbitrary number of wire segments intersect at a via candidate, a cost function:

$$c_{nij} = w_n * d_{ij}$$

is defined, where w_n is the weight of the net n to be chosen and d_{ij} reflects the contribution of the pair of wire segments according to the following. For a via candidate of degree k, we have k(k-1)/2 items in the objective function. We define:

$$d_{ij} = 2 / [k(k-1)]$$

If a pair of wire segments are assigned to the same layer, a via candidate with degree two would not be a via. But a via candidate with degree greater than two still depends on the layer assignment of the other pairs of wire segments. The d_{ij} in our cost function exactly reflects the contribution of a pair of wire segments to the objective.

In order to minimize the number of vias, we need:

$$\min \{ \Sigma c_{nij} (|x_{ni} - x_{nj}|) \}$$

where x_{ni} and x_{nj} belong to the same net and are attached to the same via candidate.

We formulate the via minimization problem as the following {0, 1} integer programming problem:

$$f = \min \{ \Sigma c_{nij} * |x_{ni} - x_{nj}| \} \qquad \text{--- (1)}$$

where x_{ni} and x_{nj} belong to the same net and attach to the same via candidate.
Subject to:

$$x_{ni} + x_{mj} = 1 \qquad \text{--- (2)}$$

where $n \neq m$; $i \neq j$; x_{ni} and x_{mj} are associated with the same cross point or knock-knee point;

$x_{ni} \in \{0, 1\}$ for all n and i;

$c_{nij} > 0$ for all n, i, and j;

The above is a standard {0, 1} integer programming problem. However, it is not linear. In next section, we will transform the problem into a standard {0, 1} integer linear programming problem.

EXAMPLE

In Fig. 1 (c), we have used the x_{nl}'s to label the wire segments in the example given in Fig. 1 (a). The knock-knee point constraint is

$$x_{a1} + x_{c1} = 1$$

Other constraints for the cross points are

$$x_{a1} + x_{b1} = 1$$
$$x_{a1} + x_{d2} = 1$$
$$x_{c2} + x_{d1} = 1$$
$$x_{b2} + x_{d1} = 1$$
$$x_{d3} + x_{e1} = 1$$

Let us set $w_n = 1$ for all nets, we have

$$1/3 * |x_{d2} - x_{d3}| + 1/3 * |x_{d3} - x_{d1}| + 1/3 * |x_{d1} - x_{d2}|$$

in the objective function for the via candidate C_3. Also we have

$$1 * |x_{b1} - x_{b2}| \text{ and } 1 * |x_{c1} - x_{c2}|$$

in objective function for via candidates C_1 and C_2.

For minimizing the number of vias in the example given in Fig. 1, we need to solve the following problem:

$$f = \min \{ 1/3 * |x_{d2} - x_{d3}| + 1/3 * |x_{d3} - x_{d1}| + 1/3 * |x_{d1} - x_{d2}| + 1 * |x_{b1} - x_{b2}| + 1 * |x_{c1} - x_{c2}| \} \quad \text{--- (3)}$$

subject to

$$x_{a1} + x_{c1} = 1$$
$$x_{a1} + x_{b1} = 1$$
$$x_{a1} + x_{d2} = 1$$
$$x_{c2} + x_{d1} = 1$$
$$x_{b2} + x_{d1} = 1$$
$$x_{d3} + x_{e1} = 1$$
$$x_{a1}, x_{b1}, x_{b2}, x_{c1}, x_{c2}, x_{d1}, x_{d2}, x_{d3} \text{ and } x_{e1} \in \{0, 1\} \quad \text{--- (4)}$$

4. Formulation Analysis

As we see in formula (1), the objective function is not a linear function. Under formulas (1) and (2), the optimization problem represents a {0, 1} integer quadratic programming problem. Fortunately we can transform the problem into one with an objective function which is much easier to solve and having linear constraints.

4.1. Existence of a Feasible Solution and Constraints

First, let us take a close look at the constraints in (2). For each constraint, exactly two variables are involved. Furthermore, the values of the two variables in a constraint are complementary. By a *cluster*, we mean a sub-set of constraints in which every constraint shares at least one variable with at least another constraints in the sub-set. In other words, constraints in a cluster are linear dependent. A *maximal cluster* is a cluster which shares no variables with the constraints outside the cluster. That is, the maximal clusters are linearly independent of each other.

A cluster is feasible if and only if all constraints in the cluster can be satisfied. That is, the values of all variables involved in the cluster can be determined.

Theorem 4.1:

The via minimization problem, formulated by (1) and (2), has a feasible (not necessarily optimal) solution if and only if all clusters in (2) are feasible.

Proof:

Necessity: If a cluster is not feasible, at least one constraint cannot be satisfied. So there are intersecting wire segments of different nets which cannot be assigned on different layers. By lemma 3.1, we do not have a feasible solution.

Sufficiency: If all clusters in (2) are feasible, we have a unique value assignment for the variables involved in (2). We can assign all remaining variables arbitrary {0, 1} values. The layer assignment is feasible by corollary 3.2.

Q.E.D.

Another way to look at the problem is that we can build a constraint graph where every wire segment corresponds to a node in the graph, an edge existing between two nodes if and only if the two corresponding wire segments cross over or overlap each other. Fig. 2 gives the constraint graph for the example given in Fig. 1. Then a maximal cluster corresponds to a connected component in the graph. A cluster is feasible if and only if the corresponding connected component in the constraint graph is two-colorable. Then we have the following equivalent corollaries.

Corollary 4.1:

The via minimization problem formulated by (1) and (2) has a feasible (not necessarily optimal) solution if and only if the corresponding constraint graph is two-colorable (or contains no odd cycles).

If only Manhattan routing is used, we can always have a feasible layer assignment by placing all horizontal wires on one layer and all vertical wires on the other. This is a very popular layer assignment used in most of today's detailed routers. However, if we allow both Manhattan and knock-knee routings, we cannot always have a feasible layer assignment. A counter example is given in Fig. 3. In 1982, Preparata and Lipski proved that three layers is sufficient ([15]) for mixed Manhattan and knock-knee routings. By solving the linear equations in (2) or checking the corresponding constraint graph, we can easily test the feasibility of the original input. From now on, we will assume that the given input always has a feasible solution. That is, all clusters in (2) are feasible.

4.2. Formulation Transformation

For convenience of the discussion, we also use the term *cluster* or *maximal cluster* to refer to the sub-set of variables involved in the sub-set of constraints.

Lemma 4.1:

If the value of any one variable in a feasible cluster is fixed, the values of all variables in the cluster are fixed.

Proof:

Let v be the variable whose value is fixed. Because the values of the two variables in any given constraint are complementary, the variable(s) in the same constraint(s) as v has a fixed value. By the definition of a cluster, every constraint in a cluster must share at least one variable with at least one other constraint in the cluster. Thus, the values of all variables in the cluster will be fixed.

In the constraint graph, a cluster corresponds to a connected sub-graph and to assign the layer to wire segments is to color the constraint graph with two colors. Once a node is colored, the color of all the other nodes in the connected sub-graph is fixed.

Using the above lemma, we can reduce the number of variables in our original formulation. For a maximal cluster, i, we need only one variable, x_i, which takes value 1 if the corresponding wire is assigned on layer one and 0 otherwise.

Let us define the new {0, 1} variable y_{ij} as follows:

$y_{ij} = 0$ if x_i has the same value as x_j

$y_{ij} = 1$ otherwise

Then, we have the following objective function:

$$\min \{ \Sigma (C_{ij} * y_{ij} + D_{ij}) \}$$

Both D_{ij} and C_{ij} are obtained from (1). D_{ij} is the value when y_{ij} is zero and C_{ij} is the difference of the value, when y_{ij} takes one, and the value when y_{ij} takes zero.

To calculate the value of D_{ij}, we first assign an arbitrary value to x_i. Next we let x_j equals x_i, i. e. let y_{ij} equals zero. Notice that all variables of the ith and the jth maximal cluster in (1) are determined (by Lemma 4.1). Then we delete all terms which contain undetermined variable in (1). The value of f in (1) is the value of D_{ij}. If we assign the complementary value of x_i to x_j, i. e. let y_{ij} equals one, and delete all terms containing undetermined variable in (1), the value of f in (1) is denoted as M_{ij}. Then we have:

$C_{ij} = M_{ij} - D_{ij}$.

Notice that C_{ij} can be either negative or positive.

By introducing the dummy boolean variables z_{ij} for each y_{ij} and eliminating the constant D_{ij} from the objective function, we can transform the formulations (1) and (2) to:

$$f = \min \{ \Sigma C_{ij} * y_{ij} \} \quad \text{--- (5)}$$

s. t.

$$x_i + x_j - y_{ij} = 2 * z_{ij} \quad \text{--- (6)}$$

Thus we transform the via minimization problem to a {0, 1} integer linear programming problem.

EXAMPLE

Let us redraw the example given in Fig. 1, in Fig. 4. We only need three variables, $x_1 = x_{a1}$, $x_2 = x_{b2}$ and $x_3 = x_{d3}$ to present the three maximal clusters, as shown in Fig. 4.

Then we can calculate D_{ij}'s and C_{ij}'s. To obtain D_{12}, we assign $x_1 = 1$ and the let $x_2 = x_1 = 1$. Notice that the values of the variables in the same cluster as x_1 are determined by (4). We have: $x_{b1} = x_{d2} = x_{c1} = 0$. Also the values of the variables in the same cluster as x_2 are determined. We have $x_{d1} = 0$ and $x_{c2} = 1$. Dropping the terms containing undetermined variables in (3), in our example terms containing x_{d3} and x_{e1}, we have:

$$D_{12} = 1/3 * |x_{d1} - x_{d2}| + 1 * |x_{b1} - x_{b2}| + 1 * |x_{c1} - x_{c2}|$$
$$= 1/3 * |0 - 0| + 1 * |0 - 1| + 1 * |0 - 1| = 2.$$

Similarly, we have the following:

$D_{13} = 1/3$

$D_{23} = 1/3$

To obtain C_{12}, we need to get M_{12} first. We assign $x_1 = 1$ and we have: $x_{b1} = x_{d2} = x_{c1} = 0$. Now let $x_2 = 0$. The values of the variables in the same cluster as x_2 are determined by (4). We have $x_{d1} = 1$ and $x_{c2} = 0$. Dropping the terms containing undetermined variables in (3), in our example terms containing x_{d3} and x_{e1}, we have:

$$M_{12} = 1/3 * |x_{d1} - x_{d2}| + 1 * |x_{b1} - x_{b2}| + 1 * |x_{c1} - x_{c2}|$$
$$= 1/3 * |1 - 0| + 1 * |1 - 1| + 1 * |0 - 0| = 1/3.$$

Thus we obtain:

$$C_{12} = M_{12} - D_{12} = 1/3 - 2 = -5/3$$

Similarly, we get:

$$C_{13} = 0 - D_{13} = -1/3$$

$$C_{23} = 0 - D_{23} = -1/3$$

Therefore, we have:

$$f = \min \{ -5/3 * y_{12} - 1/3 * y_{13} - 1/3 * y_{23} \}$$

subject to:

$$x_1 + x_2 - y_{12} = 2 * z_{12}$$

$$x_1 + x_3 - y_{13} = 2 * z_{13}$$

$$x_2 + x_3 - y_{23} = 2 * z_{23}$$

4.3. Problem Extensions

In IC layout, the layers may be not equal in performance. Some wire segments may have pre-assigned layers. For each of these wire segments, we need to add another constraint:

$$x_p = K,$$

where x_p is the corresponding variable and K is a constant, 0 or 1.

To optimize the chip performance, we may want to limit the number of contacts on a critical net: say the number of contacts on net n must be less than a constant N. Then we can pose the following constraints to take care of the case:

$$\Sigma |x_{ni} - x_{nj}| < N,$$

where x_{ni} and x_{nj} belong to net n and share a via candidate.

With a more complicated cost function, we can also reflect the trade-offs between the number of contacts and the wire length on a preferred layer or we can maximize the wire length on the preferred layer as a secondary goal.

5. Problem Solution

5.1. Underlining Ideas

Because of the huge number of variables in the formulation, any kind of known constraint relaxation technique is not practical due to the space and/or run-time problems. We propose a graph theoretical approach to solve our problem. First, let us build a weighted cluster graph G(V, E) as follows:

A node, v_i, in V has a one-to-one correspondence to a maximal cluster in the formulation. An edge, $e = \{v_i, v_j\}$, in E has a one-to-one correspondence to the boolean variable y_{ij} in the problem formulation. A weight, $w_{ij} = -C_{ij}$, is assigned to the corresponding edge. The weighted cluster graph for the example given in Fig. 4 is shown in Fig. 5.

The problem now becomes to find a maximal cut in the weighted cluster graph which will divide the vertex set into two parts. The vertices in one part will be colored with one color while the vertices in another part will be colored the other color. If the weighted cluster graph is planar, we can find the optimal solution in polynomial time ([16] and [17]). Unfortunately, the weighted cluster graph is not planar in general and we

know that to find a maximal cut in a general graph is NP-complete ([18] and [19]). So we will try the heuristics to solve the problem.

5.2. Special Case of the Problem

Let us conduct some special cases of the problem from the graph theory point of view.

Lemma 5.1:

If the weighted cluster graph is a planar graph, the optimal solution can be found in $O(|V|^{1/2} * |E|)$ time ([16], [17], [20] and [21]).

A graph is called *biconnected* if and only there exist at least two different paths connecting each pair of nodes. The *biconnected components* of a graph are maximal *biconnected* subgraphs of the graph. Notice that two biconnected components of a graph are either disconnected or connected by a node, an edge or a simple path. Then we have the following:

Lemma 5.2:

The maximum cut in a biconnected component of the weighted cluster graph is a part of the maximum cut for the graph.

Theorem 5.1:

The biconnected components of the weighted cluster graph can be handled independently.

The above theorem gives a way to partition the original problem into smaller ones if it is possible.

5.3. The Algorithm and Complexity Analysis

step 1:

Get input data; identify all cross points, knock-knee points and via candidates; and build the weighted cluster graph $G = (V, E)$;

step 2:

Sort the edges of G by their weights in decreasing order in E and use the "edge-pick-up" procedure to get an initial solution.

step 3:

Improve the initial result by swapping the nodes.

edge-pick-up procedure

Input:

weighted cluster graph $G = (V, E, W)$

Output:

an edge cut, C, partitioning the node set V into two disjoint sets, V_s and V_t

Objective:

maximize $W = \Sigma w_e$

```
{
    While ( ((e = {v_i, v_j}) ∈ E) = NULL )
    {
        if ( both v_i and v_j NOT marked )
        {
            mark v_i ∈ V_s and v_j ∈ V_t;
```

```
                put e ∈ C;
            }
            else if ( both v_i and v_j MARKED )
                put e ∈ C;
            else
            {
                mark the unmarked node ∈ the different set to the marked
                node;
                put e ∈ C;
            }
            go to next edge in E;
        }
}
```

To implement step one, we can use the plane-sweep technique to find all cross points, knock-knee points and via candidates; the time complexity will be O(nlogn + k) where n is the number of layout objects in the input chip and k is the total number of cross points, knock-knee points and via candidates [22]. In step two, we need O($|E|\log|E|$) time to sort the edges and O($|E|$) to find the cut. So the time complexity for step two is O($|E|\log|E|$) where $|E|$ is the number of edges in the weighted cluster graph.

In order to improve the initial result obtained by the above procedure, we propose the following iterative improvement approach. After the initial solution is obtained, the node set V of the weighted cluster graph is divided into two sub-node-sets, V_s and V_t, by the cut C. For each node v, w_{vs} is the sum of the weights of the edges incident to v and the nodes which belong to the same sub-node-set as v; w_{vd} is the sum of the weights of the edges incident to v and the nodes which belong to the complementary sub-node-set. If w_{vs} is larger than w_{vd} for a node v, then we will move the node, v, from its original sub-node-set to the complementary sub-node-set.

To further improve the result, we can group the nodes in their sub-node-set into pseudo nodes and repeat the above process.

Each iteration of such improvement takes O($|V|$) time where $|V|$ is the number of the nodes in the weighted cluster graph.

Usually, the number of layout objects is much larger than the number of edges, $|E|$, and the number of nodes, $|V|$, in the weighted cluster graph. So the overall time complexity of our algorithm is O(nlogn + k).

6. Results

Following the above algorithm, we find (x_1, x_2) and (x_1, x_3) as the maximal cut for the example given in Fig. 4. The layer assignment is shown in Fig. 6. The algorithm is implemented in C on a u-VAXII workstation and was used as a post-process step for a gridless channel router -- Glitter [23]. Fig. 7 and Fig. 8 show the layer assignment of two other examples. In all the examples given here, the optimal solution is obtained. In other words, the number of vias in these examples is the minimum possible. Notice that the number of vias is the minimum possible under the given topology. As an example, let us use our topological routing method to re-route the channel given in Fig. 7. Fig. 9 shows the result with only one via.

Fig. 10 shows the channel routing result of D. N. Deutsch's difficult example with 406 vias. The number of vias is minimized by our approach in about 10 minutes. The

final number of vias is 335, reduced by 17.5 percents, and the result is shown in Fig. 11.

7. Concluding Remarks

We used a unified {0, 1} linear programming formulation for the via minimization problem. Based on the analysis of the formulation and graph theory, we developed an efficient algorithm to solve the general practical via minimization problem.

To obtain the global optimal solution for the problem, the simulated annealing technique can be applied to find the maximum cut in the weighted cluster graph we proposed.

The rapid advance in VLSI fabrication technology has made it possible that more than two layers are available for routing. So how to minimize the number of vias in more than two layer routing is important. We are going to extend our approach to handle the problem involving more than two routing layers.

References

[1]. Hashimoto, A., and Stevens, J., "Wire Routing by Optimizing Channel Assignment within Large Apparatus", Proc. 8th Design Automation Workshop, 1977, pp 155-169.

[2]. Sakamoto, A., et al., "OSACA: A System for Automated Routing on Two-Layer Printed Wiring Boards", USA-Japan Design Automation Symposium, 1975, pp 100-107.

[3]. Servit, M., "Minimizing the Number of Feed Throughs in Two Layer Printed Boards", Digital Processes, Vol 3, 1977, pp 177-183.

[4]. Stevens, K.R. and Van Cleemput, W. M., "Global Via Elimination in Generalized Routing Environment", Proc. 1979 ISCAS, 1979, pp 689-692.

[5]. Kajitani, Y., "On Via Hole Minimization of Routing on a 2-layer Boards", Proc. 1980 ICCC, 1980, pp 295-298.

[6]. Ciesielski, M. J. and Kinnen, E., "An Optimum Layer Assignment for Routing in ICs and PCBs", Proc. 18th Design Automation Conference, 1981, pp 733-737.

[7]. Chan, R. W., Kajitani, Y., and Chan, S. P., "On the Via Minimization Problem for a Two-Layer Printed Circuit Board", Conference Records of the 15th Asilomar Conference on Circuits, systems and Computers, 1981, pp 22-26.

[8]. Chan, R. W., Kajitani, Y., and Chan, S. P., "Topological considerations of the Via Minimization Problem for two Layer PC. Boards", Proc. ISCAS, 1982, pp 968-971.

[9]. Chan, R. W., Kajitani, Y., and Chan, S. P., "A Graph-Theoretic Via Minimization Algorithm for Two-Layer Printed Circuit Boards", IEEE Transactions on Circuits and Systems, Vol. CAS-30, No. 5, May 1983, pp 284-299.

[10]. Pinter, R. Y., "Optimal Layer Assignment Interconnect", Proc. ICCC, 1982, pp 398-401.

[11]. Chang, K. C. and Du, D. H-C., "Efficient algorithms for layer assignment problem", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. CAD-6, no. 1, pp. 67-78, Jan. 1987.

[12]. Naclerio, N. J., Masuda, S. and Nakajima, K., "Via Minimization For Gridless Layout", Proc. 24th Design Automation Conference, 1987, pp 159-165.

[13]. Hsu, C.-P., "Minimum-Via Topological Routing", IEEE Transactions on CAD of Integrated Circuits and Systems, Oct. 1983.

[14]. Marek-Sadowska, M., "An Unconstrained Topological Via Minimization Problem for Two Layer Routing", IEEE Transactions on CAD of Integrated Circuits and Systems, July 1984.

[15]. Preparata, F. P. and Lipski, W., "Three Layers are enough", IEEE 23rd Symp. on Found. of Compt. Sci., 1982, pp. 350-357.

[16]. Hadlock, F., "Finding a Max. Cut of a Planar Graph in Polynomial Time", SIAM J. Comp., vol. 6, 1, 1977, pp 86.

[17]. Aoshima, K. and Iri, M., "Comments on F. Hadlock's paper: 'Finding a Maximal Cut of a Planar Graph in Poly. time' ", SIAM J. Comp., vol. 6, 1, Mar. 1977, pp 86-87.

[18]. Garey, M. R. and Johnson, D. S., "Computers and Intractability -- Guide to the Theory of NP-Completeness", W. H. Freeman and Co., San Francisco, 1979.

[19]. Orlova, G. I. and Dorfman, Y. G., "Finding the Maximal cut in a graph", Eng. Cybernetics, vol. 10, 1972, pp 502-506.

[20]. Edmonds, J., "Maximum matching and a polyhedron with 0,1 -- vertices", J. Res. Nat. Bur. Stand, B69, June 1965, pp 125-130.

[21]. Micali, S. and Vazirani, V. V., "An $O(|V|^{1/2} * |E|)$ Algorithm for Finding Maximum Matching in General Graphs", Proceedings of 21st Annual Symposium on Foundations of Computer Science, Oct. 1981, pp 17-27.

[22]. Nievergelt, J. and Preparata, F. P., "Plane-Sweep Algorithms for Intersecting Geometric Figures", Communications of the ACM, Volume 25, Number 10, October 1982, pp. 739-747.

[23]. Chen, H. and Kuh, E. S., "A Variable-Width Gridless Channel Router", Proc. of ICCAD-85, November 1985, pp. 304-306.

[24]. Xiong, X.-M. and Kuh, E. S., "A New Algorithm for Topological Via Minimization", to be published.

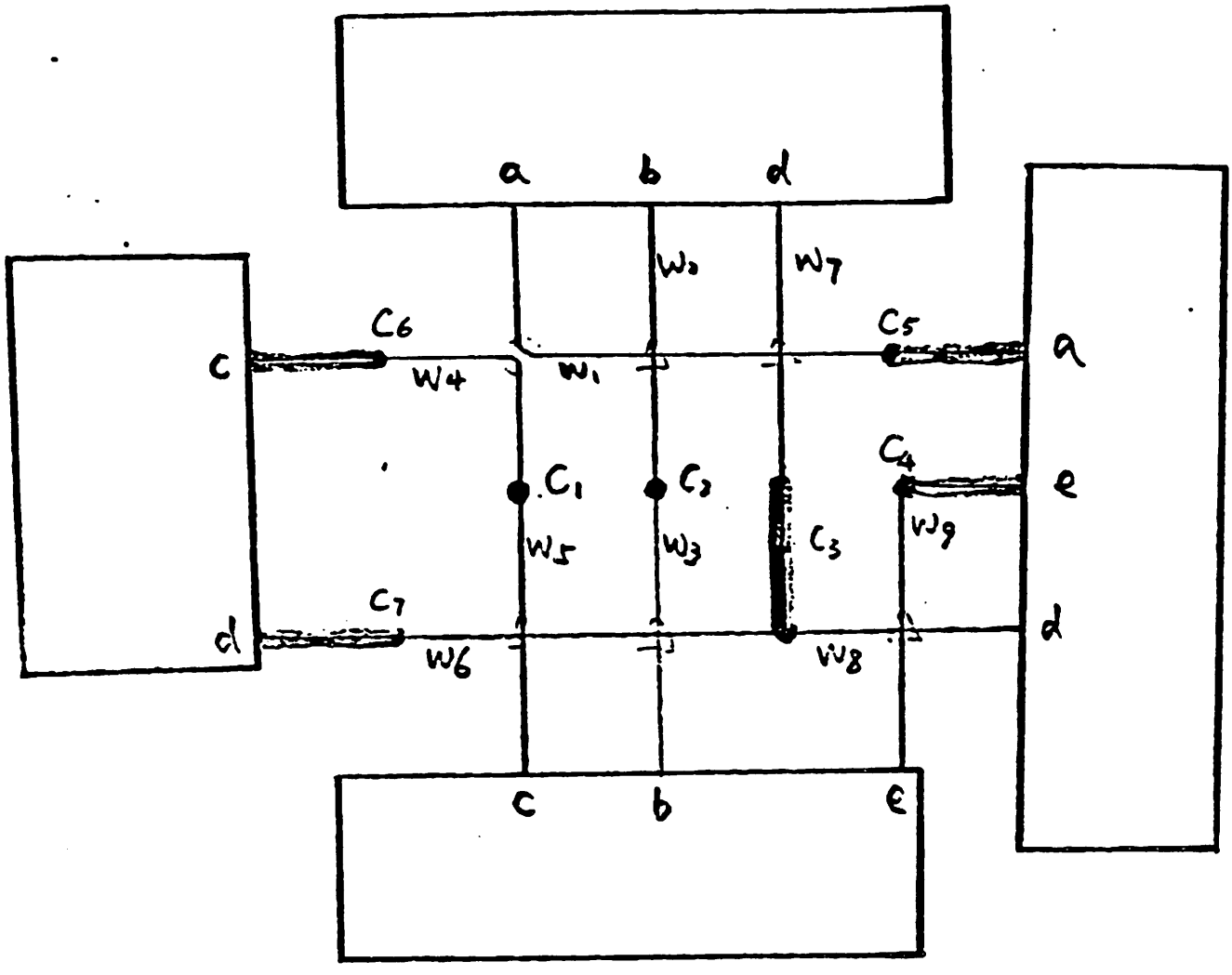


Fig. 1 (a)

△ cross point

○ knock-knee point

● ▬ via candidate

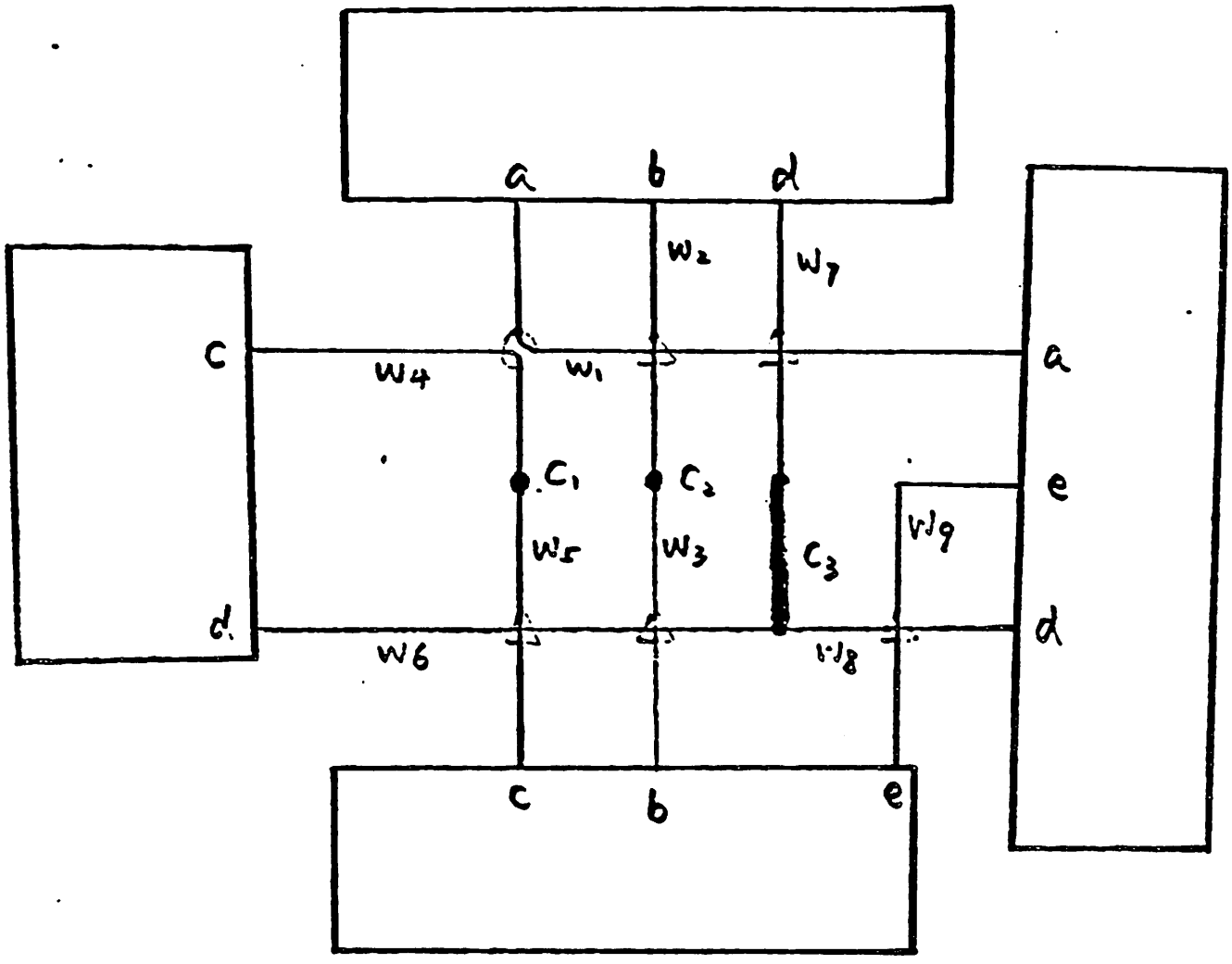


Fig 1 (b)

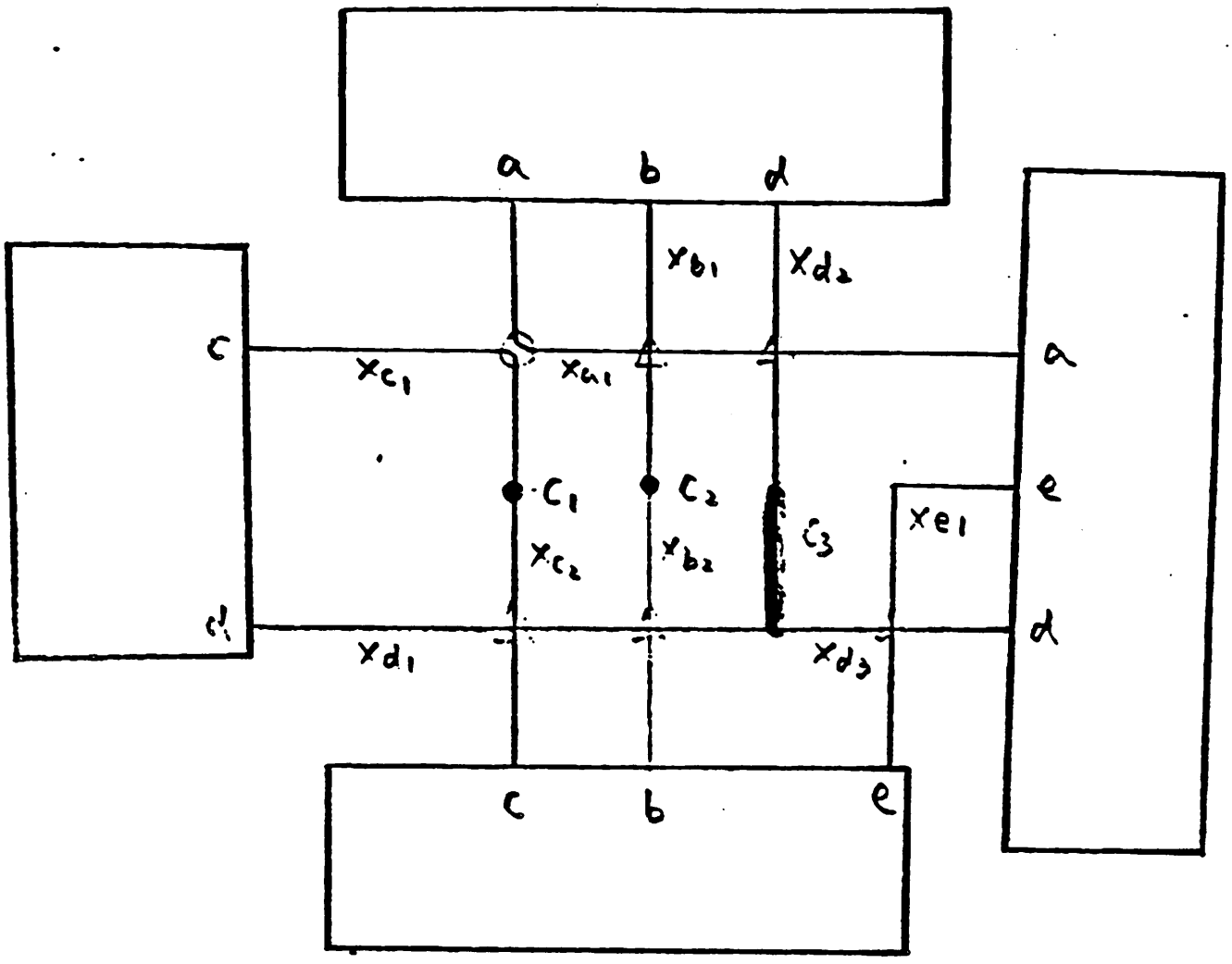


Fig 1 (c)

$w_1 \leftrightarrow x_{a1}$

$w_2 \leftarrow x_{b1}$

$w_3 \leftrightarrow x_{b2}$

$w_4 \leftarrow x_{c1}$

$w_5 \leftrightarrow x_{c2}$

$w_6 \leftrightarrow x_{d1}$

$w_7 \leftrightarrow x_{d2}$

$w_8 \rightarrow x_{d3}$

$w_9 \leftarrow x_{e1}$

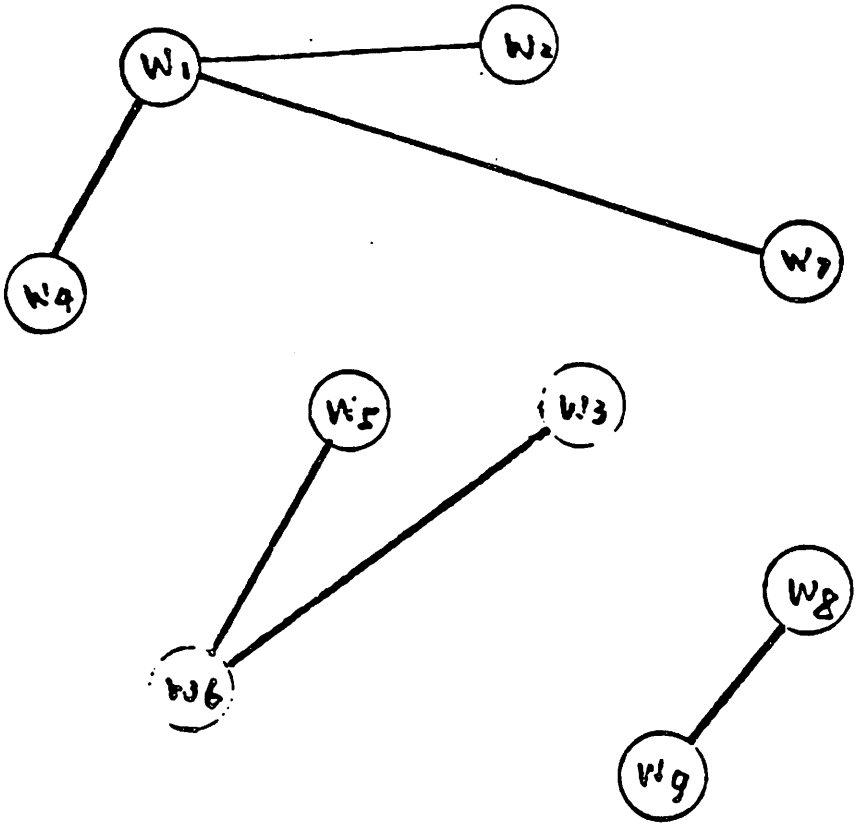


Fig. 2

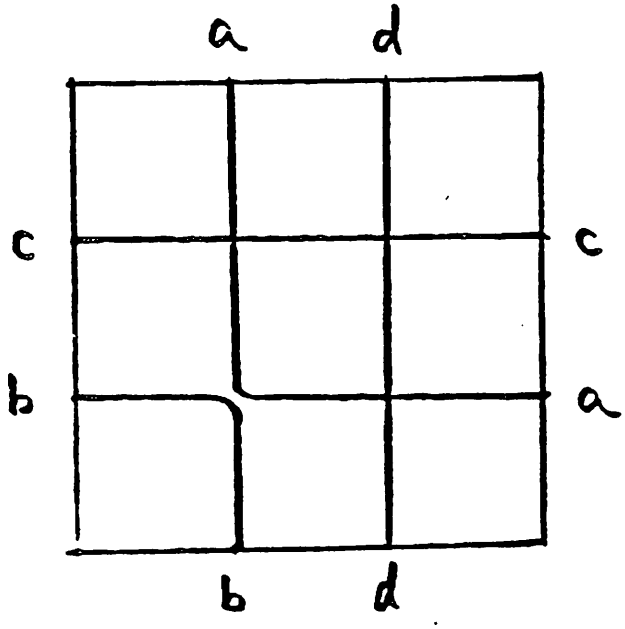


Fig. 3

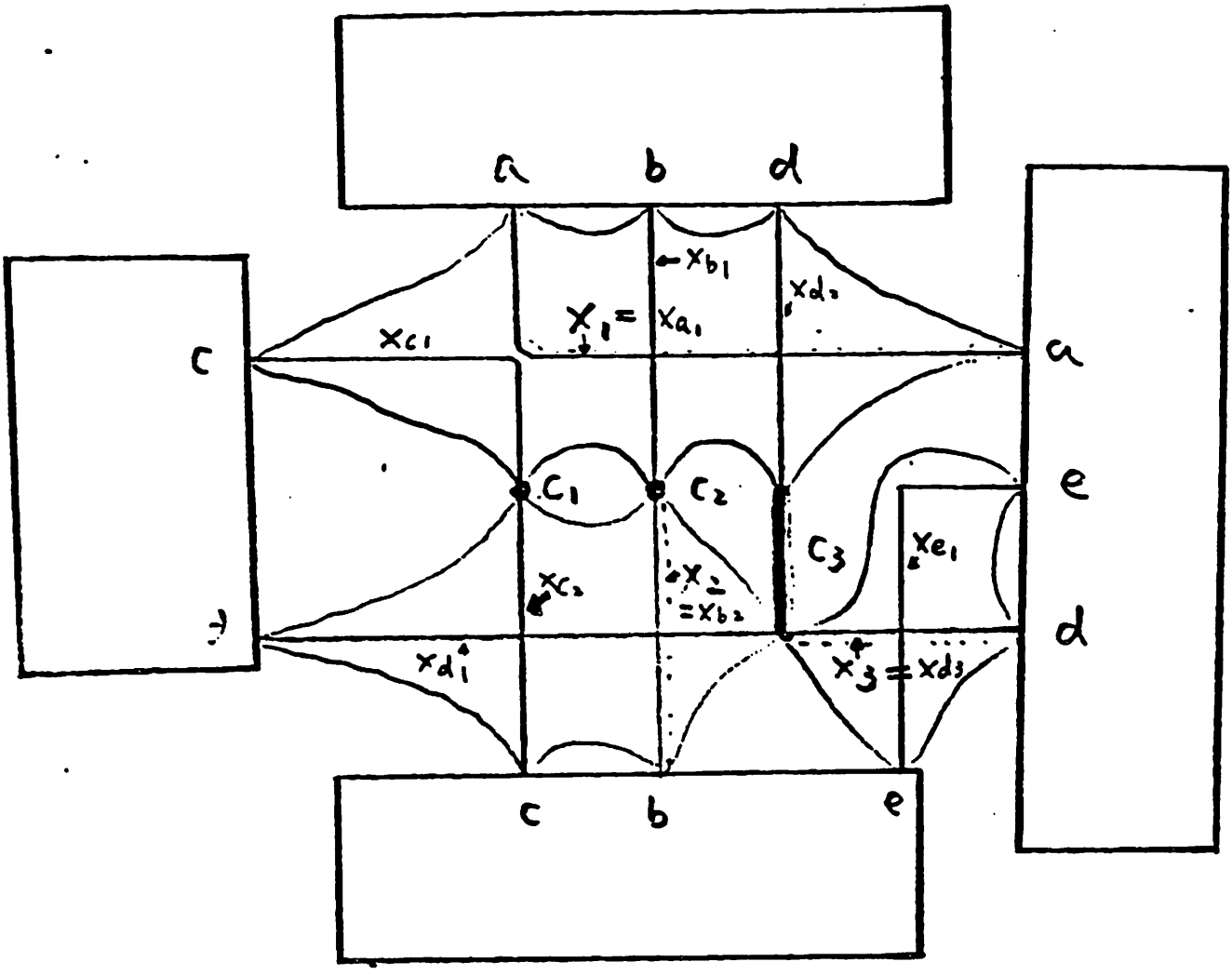


Fig 4

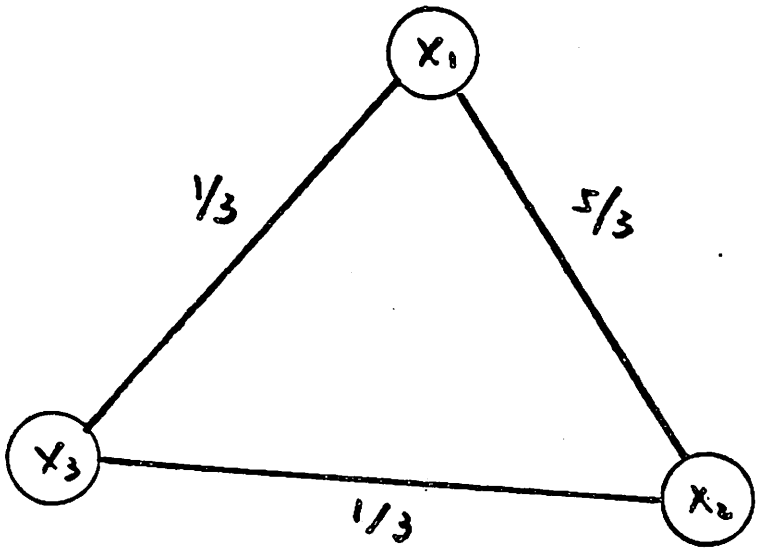


Fig. 5

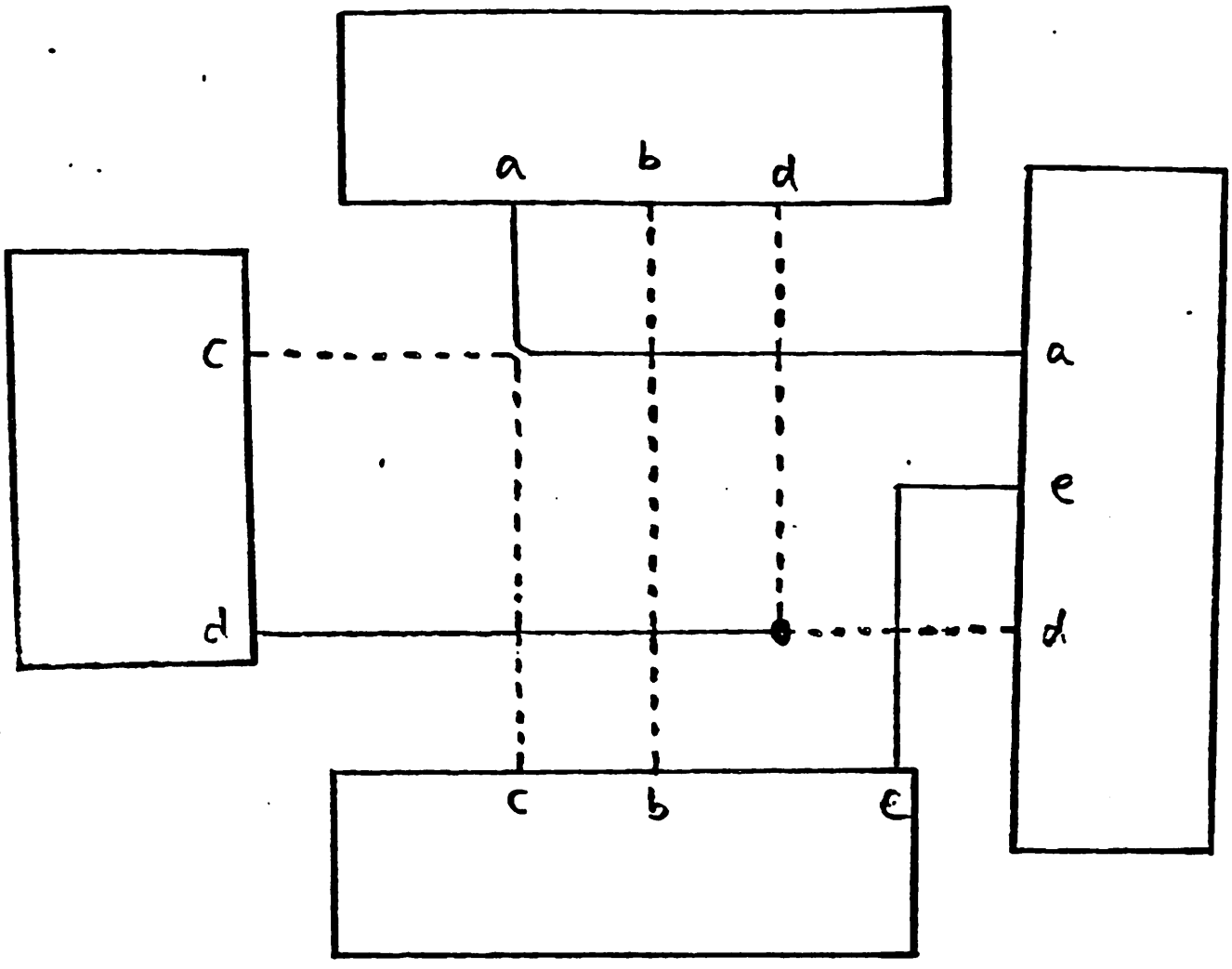


Fig 6

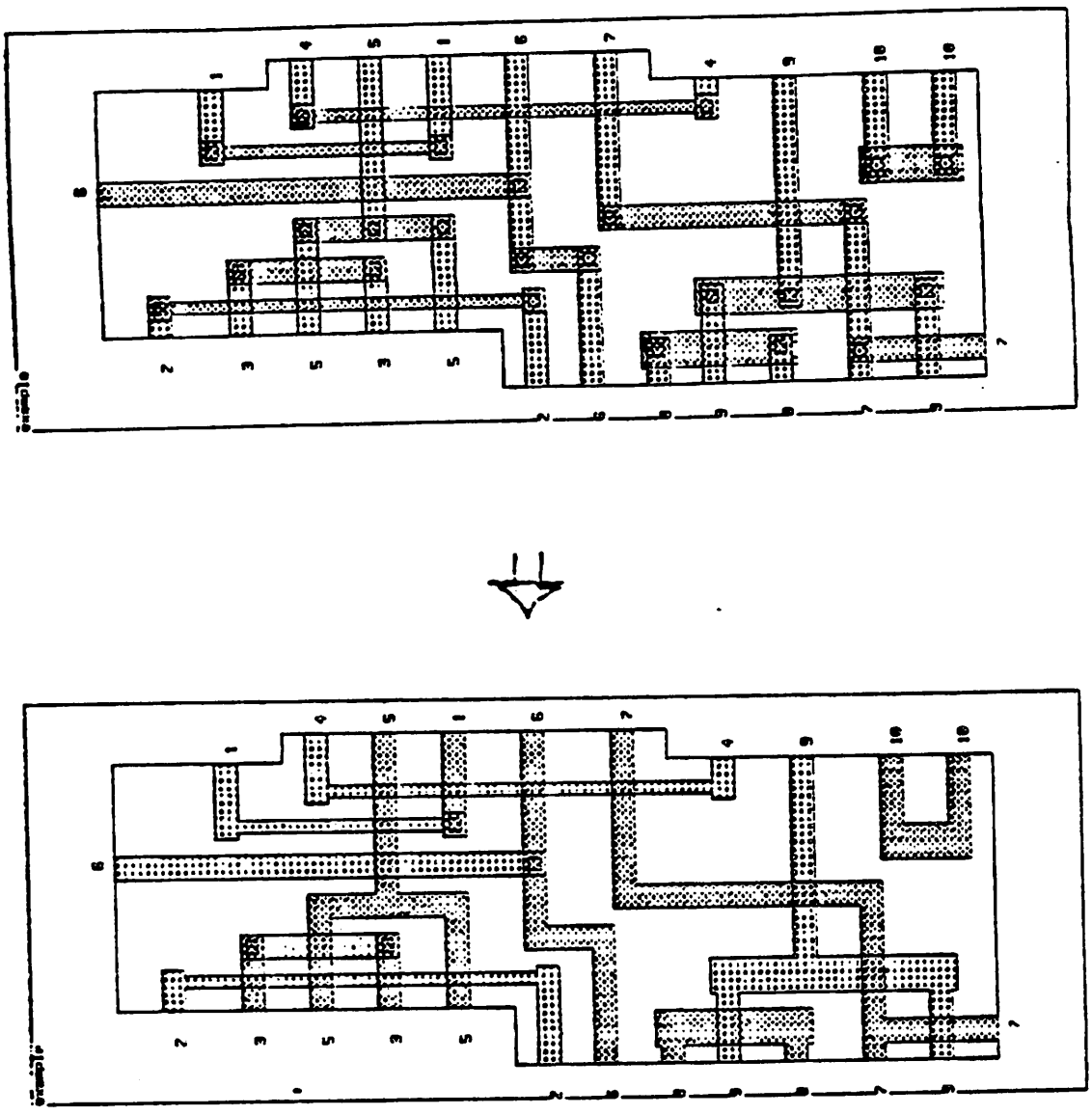
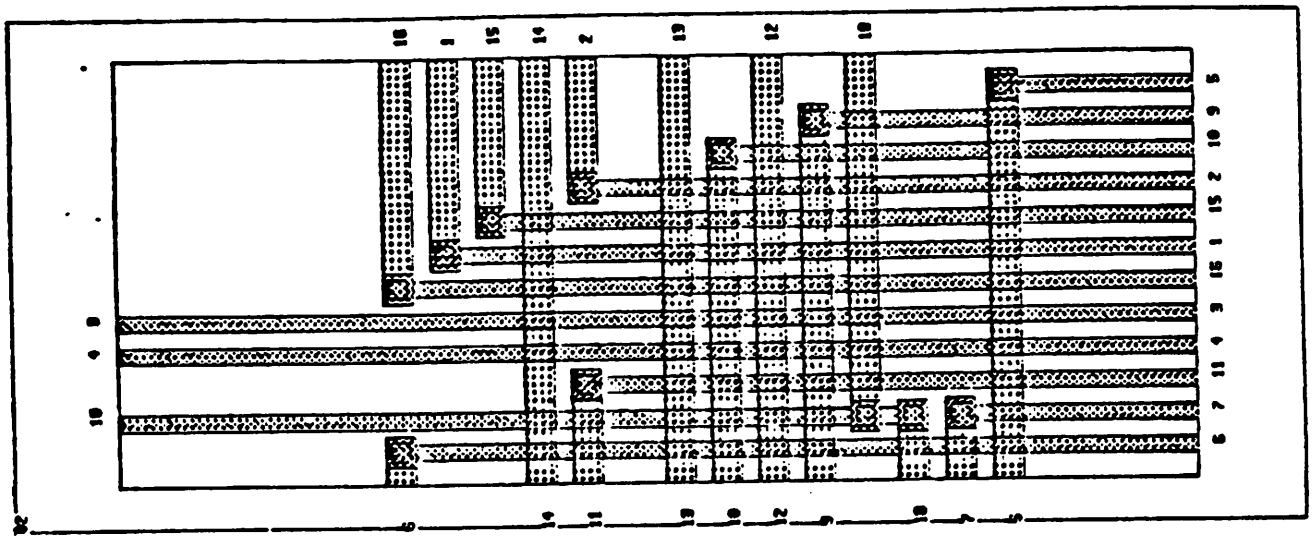


Fig. 7



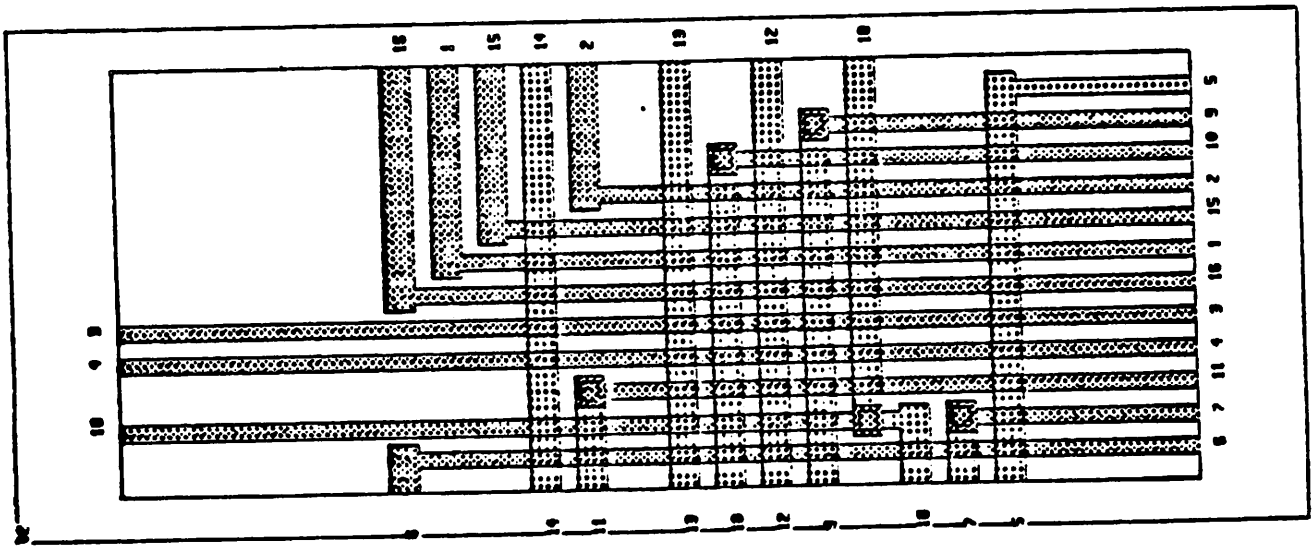


Fig. 8

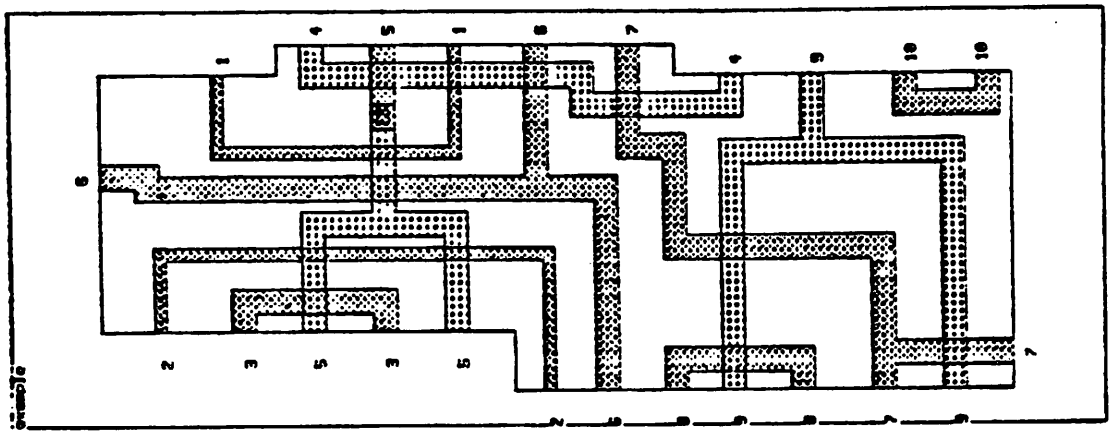


Fig. 9


cifplot* Window: -54.45 4.95 -53.7 590.7 @ u=200 --- Scale: 1 micron is 0.015 inches (381x)
diff

Fig. 10

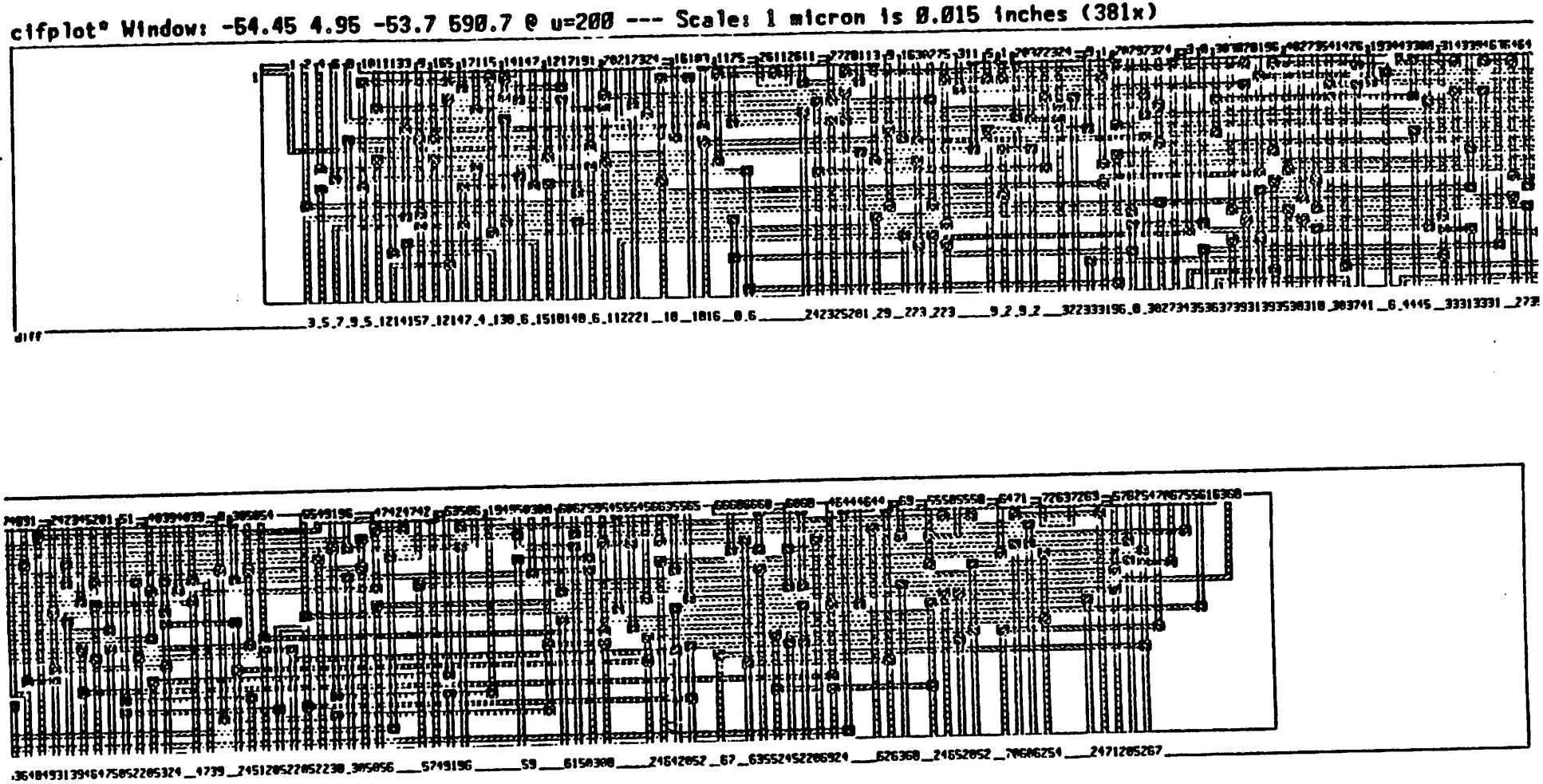

cifplot* Window: -54.45 4.95 -53.7 590.7 @ u=200 --- Scale: 1 micron is 0.015 inches (381x)

Fig. 11