ZAMM · Z. angew. Math. Mech. 69 (1989) 11, 393–399

GRÖTSCHEL, M.; JÜNGER, M.; REINELT, G.

Via Minimization with Pin Preassignments and Layer Preference

Das Problem der Via-Minimierung mit Anschlußvorbelegungen und Lagenbevorzugung tritt beim VLSI- bzw. Leiterplatten-Entwurf auf. Es kann als Problem des maximalen Schnittes in einem Graphen formuliert werden. In diesem Aufsatz berichten wir über die Anwendung approximativer Algorithmen für dieses Problem zur Minimierung der Anzahl der Vias industriell gefertigten elektronischen Schaltkreisen.

The problem of minimizing the number of vias subject to pin preassignments and layer preferences comes up in VLSI respectively printed circuit board design. It can be formulated as a max-cut problem. In this paper, we discuss the application of approximative algorithms for the max-cut problem to minimize the number of vias of (real-world) electronic circuits.

Задача минимизации числа пересоприкосновений относительно закрепления преназначений и слоевых предпочтений вырастет в СБИС и в конструкции плат с печатной схемой, соответственно. Можно формулировать эту задачу в виде задачи максимального сечения. В этой статье обсуждается применение приблизительных алгоритмов на задачу максимального разреза, чтобы минимизировать число пересоприкосновений в (реалистических) электронических цепях.

1. Introduction

The physical layout of integrated circuits is usually split up into phases like placement, routing, layer assignment, and compaction. Assuming that placement and routing is completed we focus here on the layer assignment phase. The objective of this phase is to assign wire segments to layers such that crossing (or touching) segments belonging to different nets are assigned to different layers. The places where wires change layers are called *vias*. Vias need additional space, they are obstacles in the compaction phase, and the yield in production decreases with the number of vias. Thus it is desirable to find a layer assignment such that the number of vias is as small as possible.

For the case of two layers (and assuming that the transient routing contains no k-way junctions for $k \geqq 4$), PINTER [5] and CHEN, KAJITANI and CHAN [3] have shown independently that this problem is solvable in polynomial time by giving reductions to a max-cut problem in a planar graph. However, these reductions do not cover certain side constraints required in practice. In many cases one of the two layers is preferred and pins are preassigned to some layer. For instance, in standard cell design in 3-μ-CMOS technology there are a polysilicon (poly) and an aluminum (alu) layer. Preferably, wires should be in the alu-layer whereas pins are usually in the poly-layer. In ECL technology we have two metal layers (alu 1 and alu 2), and pins are usually preassigned to one of the layers.

BARAHONA, GRÖTSCHEL, JÜNGER and REINELT [2] pointed out that PINTER's reduction can be generalized to the via minimization problem subject to layer preference and pin preassignments. However, the max-cut problem that results from this reduction is NP-hard. We outline this transformation in Section 2, discuss several heuristic approaches to solve this problem approximatively in Section 3, and present computational results on real-world layout problems in Section 4.

2. Transformation

The max-cut problem is a standard model in combinatorial optimization. An instance of it can be described as follows. Given a (undirected) graph $G = (V, E)$ with edge weights w_{ij} for all $ij \in E$, find a cut $C \subseteq E$ such that $\sum_{ij \in C} w_{ij}$ is as large as possible. (Here, a *cut* is an edge set $C \subseteq E$ with $C = \{ij \in E \mid i \in W, j \in V \setminus W\}$ for some node set $W \subseteq V$. The transformation of the via minimization problem (given two layers) with layer preference and pin preassignments to a max-cut problem consists of a sequence of reductions which we describe now.

We suppose that all cells are placed on a chip and all nets have been routed but that the assignment of wire segments to layers has not been performed yet (i.e., a so-called *transient routing* is given). A net may connect two or more pins. In the latter case, the net may contain 3-way junctions, but we assume that there are no 4-way junctions. (Fortunately they seldom occur in practice.) Figure 1 shows a transient routing of eight nets, one of which (net 2) is a 4-pin net, one of which (net 1) is a 3-pin net, and the rest are 2-pin nets. There are three 3-way junctions.

The first step of the transformation is the following. We partition each wire into *free* and *critical segments*, such that vias are allowed on free but forbidden on critical segments. Obviously, at crossings of different nets no vias are allowed. There may be further design rules which restrict the placement of vias such as: two wires which run parallel at minimum feasible distance within a certain interval may not contain a via within this interval (placing a via on one of the wires here may create a contact with the other wire). In Figure 2 there are 24 critical segments (drawn solid and numbered 1, ..., 24), the free segments are drawn dotted and are not numbered. Note that, by construction, free segments never cross.

We represent the topology of the free and critical segments in a *layout graph* $G = (V, E)$ and assign certain weights w_{ij} to some of the edges ij of E. Each node in V corresponds to a critical segment, and there are two kinds of edges in E. Nodes i and j are joined by a *conflict edge* ij whenever the associated critical segments must be on different layers. Nodes i and j are joined by a *continuation edge* ij whenever the associated critical segments are

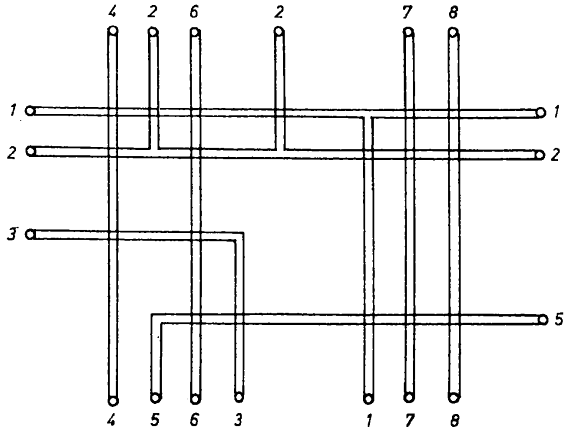

Fig. 1

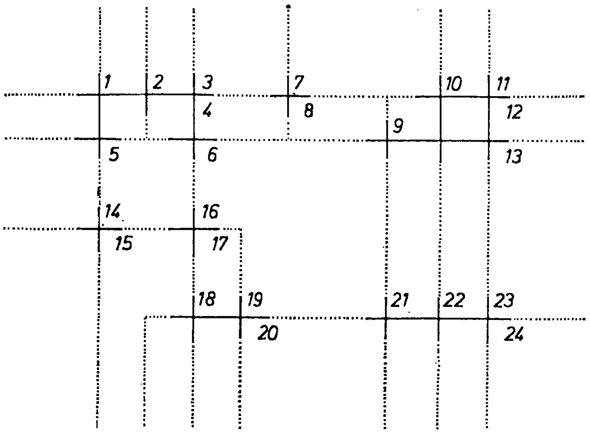

Fig. 2

connected by a free segment. So $E = A \cup B$, where A are the conflict edges and B the continuation edges. If the critical segments i, j, k are joined by a 3-way junction, the three resulting continuation edges receive the weight $w_{ij} = w_{ik} = w_{kj} = 1/2$, whereas all other continuation edges ij obtain weight $w_{ij} = 1$. Conflict edges do not get weights. In our example the layout graph looks as shown in Figure 3. Here the conflict edges are represented by solid lines, the continuation edges with weight 1 by dotted lines, and the continuation edges with weight 1/2 by broken lines.

If the *conflict subgraph* $H = (V, A)$ of G is not bipartite it is easy to see that there is no feasible assignment of wire segments to the two layers (in this case the transient routing has to be corrected). Otherwise H partitions into connected bipartite components $(V_1, A_1), \dots, (V_k, A_k)$. (In our example, there are seven such components.) Clearly, the assignment of one node (respectively, its corresponding wire segment) of a component (V_i, A_i) to a layer implies the assignment of all other nodes of V_i to some layer. The via minimization problem can now be phrased in terms of the layout graph G as follows. Find a cut C of G that contains all conflict edges A such that the sum of the weights of the continuation edges in C is as small as possible. This is a max-cut problem with an additional side constraint. To formulate this as a max-cut problem without a further constraint we proceed as follows.

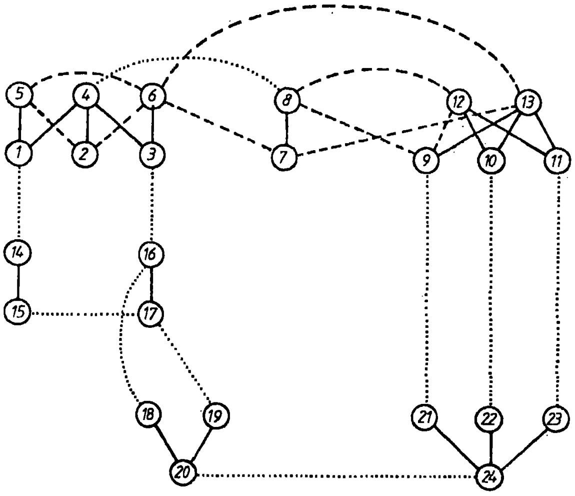

Fig. 3

We construct a *reduced layout graph* $R = (W, F)$ in the following way. For each component (V_i, A_i) of H we arbitrarily select a "representative" node v_i and set $W = \{v_1, \dots, v_k\}$. F contains an edge linking v_i and v_j, if and only if G contains a (continuation) edge linking some node in V_i to some node in V_j. Note that two nodes of a component (V_i, A_i) may be linked by a continuation edge, so R may contain loops. Observe also that each edge of R represents some free wire segments. Since these segments do not cross, R is a planar graph. (In graph theoretic terms, R is obtained from G by contracting the node sets $V_1, \dots, V_k$ to single nodes and replacing multiple edges by single edges.)

For each edge $v_i v_j$, $v_i \neq v_j$, in F we define two weights α_{ij} and β_{ij} in the following way.

α_{ij} := sum of the weights of all those continuation edges in G linking nodes in V_i to nodes in V_j which are assigned to different layers if the representing nodes v_i and v_j are assigned to the same layer.

β_{ij} := sum of the weights of all those continuation edges in G linking nodes in V_i to nodes in V_j which are assigned to different layers if the representing nodes v_i and v_j are assigned to different layers.

For the loops v_iv_i in F we define

α_{ii} := sum of the weights of all those continuation edges in G connecting nodes in V_i which have to be assigned to different layers.

$\beta_{ii} := 0$.

The reduced layout graph of our example is shown in Figure 4. For each component (V_i, A_i) of H an arbitrary representative node is chosen. The edge weights in this figure are $(\alpha_{ij}, \beta_{ij})$.

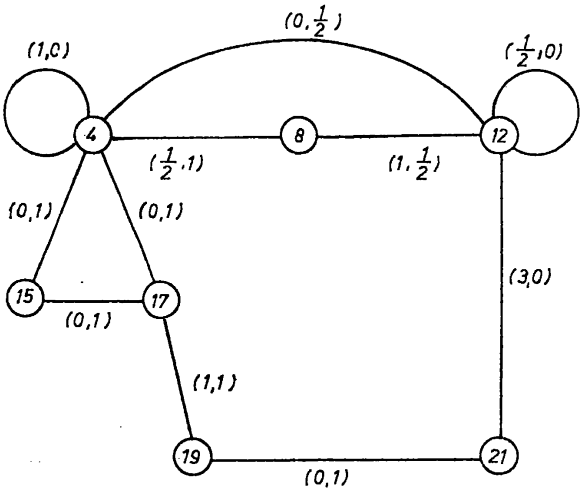

Fig. 4

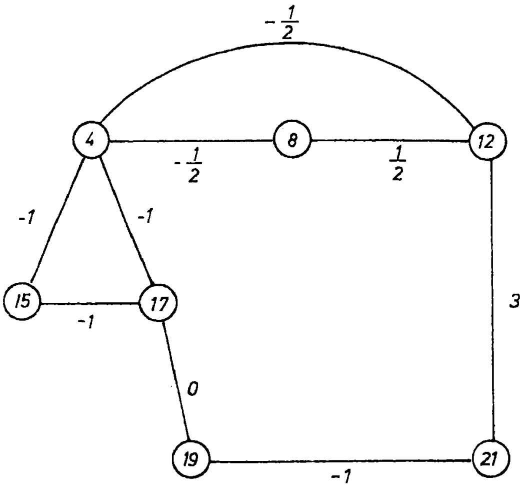

Fig. 5

A layer assignment corresponds — by construction — to a partition W^+, W^- of W. Given such a partition the number of vias is

$$\text{VIA}(W^+, W^-) := \sum_{\substack{v_iv_j \in F\\ v_i, v_j \in W^+}} \alpha_{ij} + \sum_{\substack{v_iv_j \in F\\ v_i, v_j \in W^-}} \alpha_{ij} + \sum_{\substack{v_iv_j \in F\\ v_i \in W^+\\ v_j \in W^-}} \beta_{ij} .$$

Defining $C := \sum_{v_iv_j \in F} \alpha_{ij}$ we get

$$\text{VIA}(W^+, W^-) - C = \sum_{\substack{v_iv_j \in F\\ v_i \in W^+\\ v_j \in W^-}} (\beta_{ij} - \alpha_{ij})$$

and so for $c_{ij} := \alpha_{ij} - \beta_{ij}$ the problem of minimizing the number of vias is equivalent to the max-cut problem in $R = (W, F)$ with the weights c_{ij} on the edges $v_iv_j \in F$. If M is the maximum weight of a cut in R, then $C - M$ is the smallest number of vias needed to make a feasible layer assignment. Since max-cut problems in planar graphs can be solved in polynomial time (using matching techniques) the minimum number of vias can be computed in polynomial time. In our example, we have $C = 7$, and the edge weights c_{ij} are shown in Figure 5. The loops do not play a role anymore and are left out.

A maximum weight cut in this graph is induced by partitioning W into $\{12\}$ and $\{4, 8, 15, 17, 19, 21\}$ and therefore at least $C - 3 = 4$ vias are needed. The corresponding (optimal) layer assignment is shown in Figure 6.

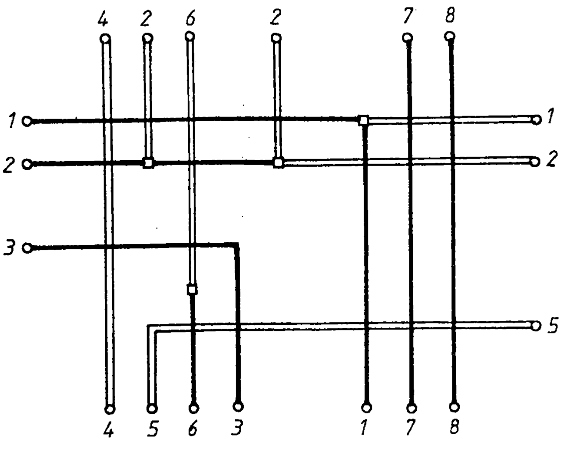

Fig. 6

We will now show how preferal of one layer to the other is taken into account. We start from a given transient routing and, after having defined free and critical wire segments as before, we subdivide each free segment (further) into a free segment followed by a critical segment followed by a free segment. From this new set of critical and free wire segments the layout graph $G = (V, E)$, $V = \{1, \dots, n\}$, and the weights w_{ij} of the continuation edges are constructed as described before. An additional node, called 0, is associated to the preferred layer. We join each node i, $1 \leqq i \leqq n$, to 0 by a continuation edge. Let us call this graph the *extended layout graph* $\hat{G} = (\hat{V}, \hat{E})$. With each continuation edge $0i$, $i \in V$, we have to associate a weight w_{0i}. For each particular problem instance, this has to be done by the layouter individually. The weights w_{0i} should reflect the degree to which segment i is attracted to the preferred layer (negative values mean attraction to the preferred layer, positive values mean attraction to the other layer). The individual assignment of values w_{0i} to the wire segment can also be used to reflect various technological requirements.

Let us remark here that via minimization and layer preference are conflicting goals. By increasing the number of vias more wire segments can be placed on the preferred layer. Large absolute values for the w_{i0}'s give strong emphasis to layer preference while setting $w_{i0} = 0$ for all $i \in V$ results in via minimization without side constraints. This suggests to introduce a parameter, to be set by the user, with which the relative emphasis given to the two goals can be controlled. We will suggest a way to handle this in Section 4.

Starting from $\hat{G} = (\hat{V}, \hat{E})$ we can construct a reduced layout graph $\hat{R} = (\hat{W}, \hat{F})$ with weights $c_{ij} = \alpha_{ij} - \beta_{ij}$ for all edges $ij \in \hat{F}$ as described before. We will denote the node of $\hat{R}$ representing the new node 0 of $\hat{G}$ by v_0. Every maximum weight cut in $\hat{R}$ corresponds to an optimum solution of the via minimization problem subject to the given layer preferences.

Pin preassignments are handled as follows. For each preassigned pin we introduce a critical segment starting at that pin. In the extended layout graph this pin is represented by some node. Let u be such a node. One way to realize the preassignment of the corresponding pin would be to join node u to node 0 by a continuation edge. We set the weight $w_{0u} = -M$ if the pin has to be on the preferred layer or $w_{0u} = M$ if the pin has to be on the other layer, where M is a very large positive number. But this would produce many edges in the reduced layout graph. A better way to achieve pin preassignments works as follows. If the pin is preassigned to the preferred layer then we identify node u with node 0, otherwise we connect u to 0 by a conflict edge. All preassignments are now represented by node v_0 in the reduced extended layout graph.

The reduced layout graph coming up in the above construction is in general not planar, but it is *almost planar* in the sense that the removal of one node (node v_0 in this case) results in a planar graph. The max-cut problem for almost planar graphs is NP-hard (BARAHONA [1]). So one cannot expect to find a fast algorithm for this type of problems. However, if there is no layer preference but possibly pin preassignments then the problem is polynomially solvable provided that all pins are on the outer border of the transient routing because in this case the reduced extended layout graph is still planar. Note that this is frequently the case in channel routing problems.

3. Heuristics

The max-cut model for the via minimization problem under side constraints introduced above provides a clean model for attacking this problem with techniques from combinatorial optimization. In this note we concentrate on the heuristic approach, i.e. the design and implementation of fast approximative algorithms. We will now briefly outline the heuristic procedures (and their combinations) we have used in our computational experiments. We apply our heuristics to the reduced (extended) layout graph $\hat{R}$. In the following we assume that we deal with the reduced extended layout graph containing node v_0.

Construction heuristics

The purpose of these heuristics is to find a suitable initial solution very fast.

C1) Trivial heuristics

(a) Trivial Cut (all nodes on one side of the cut)

(b) Random Cut

(c) All nodes v_i with negative $c_{v_0v_i}$ on the side of the cut containing node v_0, and the others on the other side

It may seem strange to consider such heuristics, but in previous work on other combinatorial optimization problems we frequently found that some improvement heuristics perform best if initialized with a "well-chosen bad" starting solution.

C2) Greedy-type heuristic

Starting with $S = \{v_0\}$ and $T = \emptyset$ we successively assign the remaining nodes to S or T. In each step we compute the increase of the objective function for all nodes in $W \setminus (S \cup T)$ when put into S, respectively T, and assign a node with maximum increase to S or T until $S \cup T = W$.

C3) Spanning tree heuristic

This heuristic is motivated by the goal to have edges with large positive weight in the cut and edges with large negative weight not in the cut. To achieve this we compute a maximum weight spanning tree in $\hat{R}$ with edge weights $|c_{v_iv_j}|$. The tree induces a bipartition of W where the nodes v_i, v_j with positive $c_{v_iv_j}$ are on different sides of the cut, and the nodes with negative $c_{v_iv_j}$ are on the same side.

Improvement heuristics

Starting with some initial solution these heuristics perform local modifications to improve the current solution.

I1) 1-Exchange heuristic

For each node it is checked whether the objective function grows when it is moved to the other shore of the cut, and if it grows, the exchange is made. This is done until no more improvements are possible.

I2) Local enumeration heuristic

For each node v_i (except for node v_0) we find the best possible assigment of v_i and all of its neighbours in $\hat{R}$ by complete enumeration. This is done until no more improvements are possible. (This is feasible for our application because the degrees of the nodes $v_1, \ldots, v_k$ are usually small.)

I3) Kernighan-Lin exchange heuristic

One often observes that local improvement heuristics which only allow the exchange of few nodes (as e.g. 1-exchange) get stuck in local minima. Kernighan and Lin [4] formulated an exchange procedure which is relatively fast and allows the exchange of many nodes in each step. This method is well known so we do not describe it here.

Obviously, the three exchange heuristics described above can be embedded into a Monte Carlo respectively simulated annealing scheme.

4. Results

We do not intend to give a complete and detailed analysis of the performance of each possible combination of a construction heuristic with an improvement heuristic. Our experiments showed that, on the average, running the spanning tree heuristic (C3) first and improving this solution by the local enumeration heuristic (I2) afterwards produced the best results. This observation is, however, based on a rather small sample of small to medium sized real-world problems; so further computational testing is necessary to derive a definitive conclusion about the relative merit of these heuristics. Anyway, the improvements we obtained compared with what is used in industrial applications is encouraging.

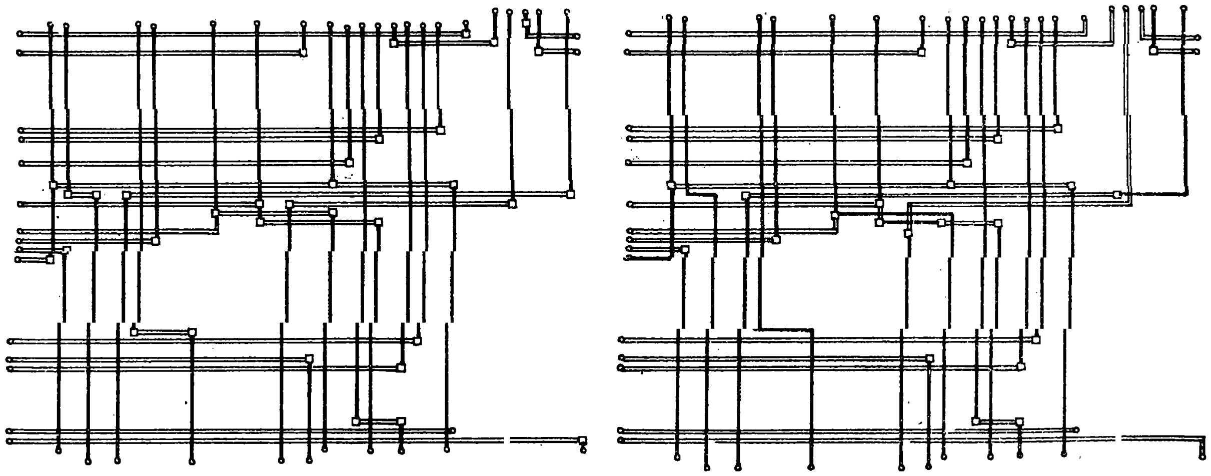

Fig. 7

To give an idea about the improvements that can be obtained with the heuristic (spanning tree and local enumeration) proposed here we will now describe some via minimization experiments with a real-world design of one channel in a standard cell circuit that we obtained from SIEMENS Germany. The channel consists of 216 nets connecting 573 pins, and there are 630 vias in the original layout. The whole channel is too large to be displayed here, so we use small extract of it (the final part of the channel) to explain our experiments. Figure 7a shows the original layout of this part which has 34 vias. Note that all horizontal wire segments are on the white layer, and almost all vertical wire segments are on the black layer. This is a usual situation in industrial design which is not due to technological requirements but to features of the commonly used channel routing algorithms.

Figure 7b shows the result of a pure via minimization with no pin preassignments or layer preference. It contains 24 vias, i.e., 10 vias less than the original design. If we preassign the pins as they are given in the original design we can save only those four (unnecessary) vias which are caused by the strict assignment of horizontal wire segments to the white layer.

In our experiments we handle layer preference as follows There are two parameters h and λ. For each free wire segment of length μ greater than or equal to λ, we introduce a critical segment as described in Section 2 and attract it with "force" $\mu \cdot h$ to the preferred layer. The parameter h is the control parameter for the user to decide on the relative emphasis given to the conflicting goals. The stipulation $h = 0$ results in a via minimization subject to preassignments of some pins and wire segments, while $|h|$ large enough favours one of the layers strongly. Clearly, some care has to be given to setting h.

Figure 8 shows the tradeoff between via minimization and layer preference. The parameter λ is set so small that practically every free segment is affected. As an example, Figure 9 shows the layer assignment if the white layer is preferred by setting $h = 6$.

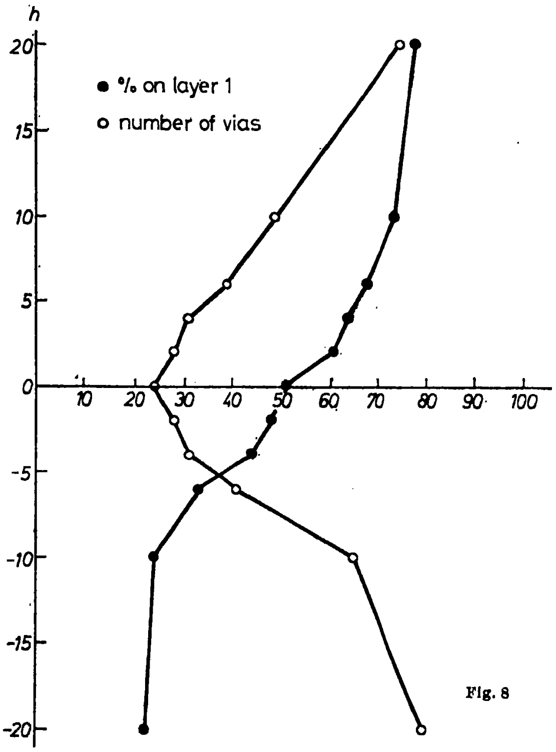

Fig. 8

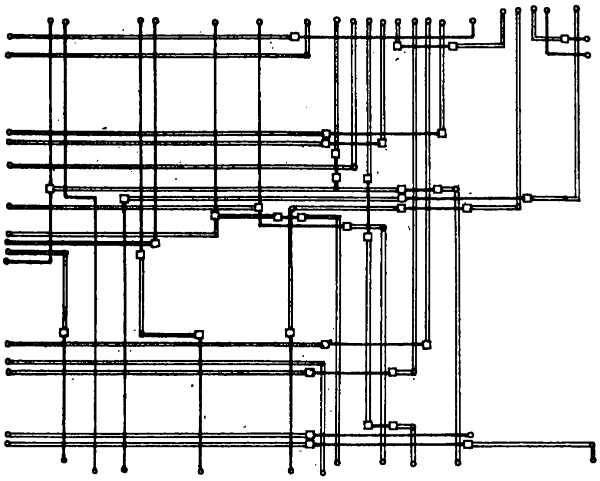

Fig. 9

The results for the complete channel are the following. For pure via minimization the reduced layout graph has 925 nodes and 1740 edges.

- Original design: 630 vias, 64% of the wiring area on the white layer.
- Pure via minimization with all pins preassigned: 585 vias, 68% of the wiring area on the white layer.
- Pure via minimization without pin preassignment: 525 vias, 31% of the wiring area on the white layer.

We also ran a layer preference example as described above. We set $h = -6$ to prefer the black layer. The resulting reduced layout graph had 4056 nodes and 6521 edges.

- Layer preference ($h = -6$) without pin preassignment: 666 vias, 28% of the wiring area on the white layer.

Finally we show the pure via minimization results on five more channels (all supplied by SIEMENS) in Table 1.

Table 1

	C1	C2	C3	C4	C5
nodes in reduced layout graph	828	980	1327	1202	1366
edges in reduced layout graph	1445	1775	2480	2234	2606
vias in original design	421	434	683	650	782
via minimization with preassignments	302	376	563	504	645
via minimization without preassignments	272	347	513	475	610

All these results were obtained with Pascal programs running in a few minutes on the Motorola 68000 based Unix microcomputer CADMUS 9230.

We have also experimented with simulated annealing. Of course, by running this procedure long enough, one will always find some improvement (or the optimum solution). We spent considerable time on fine-tuning the parameters for the simulated annealing scheme for this special case. In all our experiments simulated annealing was allowed to run ten times longer than the slowest of all our other heuristics. Despite these efforts simulated annealing never produced a better solution than the one found by the combination of C3 and I2, actually it rarely found a solution of similar quality. Thus, in our opinion, simulated annealing does not appear to be an attractive alternative.

5. Conclusion

We feel that our success in decreasing the number of vias in industrially designed chips indicates that our approach is a promising one even if only heuristics are used. The large scale max-cut problems arising in determining ground states of Ising spin glasses (BARAHONA, GRÖTSCHEL, JÜNGER and REINELT [2]) which we solved optimally with a cutting plane approach are very similar in structure to the problems in the context of via minimization. This makes us confident that even optimal solutions to the via minimization problem (subject to side contraints like layer preference and pin preassignments) can be found within reasonable computation time. We are quite certain that good lower bounds on the necessary number of vias can be found efficiently (using methods from polyhedral combinatorics) so that optimality up to a few percent can be guaranteed.

A critique to the approach described above is necessary. The via minimization problem comes up only because the layout problem is hierarchically split into many subproblems. It would (probably) be a good idea to combine the routing and layer assignment phase and design routing algorithms which take the number of vias into account. However, at present such a methodology is not available and via minimization as described above is indispensable.

This research has been partially supported by Deutsche Forschungsgemeinschaft, Forschungsschwerpunkt „Anwendungsbezogene Optimierung und Steuerung“.

References

1 BARAHONA, F.: On the computational complexity of Ising spin glass models. J. Phys. A: Math. Gen. **15** (1982), 3241–3253.
2 BARAHONA, F.; GRÖTSCHEL, M.; JÜNGER, M.; REINELT, G.: An application of combinatorial optimization to statistical physics and circuit layout design. Preprint No. 95, Institut für Mathematik, Universität Augsburg 1986, Operations Research. **36** (1988), 493–513.
3 CHEN, R.-W.; KAJITANI, Y.; CHAN, S.-P.: A graph-theoretic via minimization algorithm for two-layer printed circuit boards. IEEE Trans. Circuits and Systems **30** (1983), 284–299.
4 KERNIGHAN, B. W.; LIN, S.: An efficient heuristic procedure for partitioning graphs. Bell Systems Techn. J. **49** (1979), 291–308.
5 PINTER, R. Y.: Optimal layer assignment for interconnect. J. of VLSI and Computer Systems **1** (1984), 123–137.

Received September 12, 1988

Address: Prof. Dr. MARTIN GRÖTSCHEL, Dr. M. JÜNGER, Dr. G. REINELT, Universität Augsburg, Institut für Mathematik, Memminger Straße 6, D-8900 Augsburg