Bull. Nat'l Taiwan Nor. Univ.
Vol. 37. pp. 243-265 (1992)

師大學報
第37期 第243-265頁(民國81年)

A 'HYBRID SENSE' ALGORITHM FOR LAYER ASSIGNMENT IN THREE-LAYER VLSI ROUTING

Kuo-En Chang

Department of Information and Computer Education

Abstract

The layer assignment, also called constrained via minimization, is to determine which layers can be used for routing the wire segments such that the number of vias can be minimized. Vias should be eliminated as many as possible in the layout design because vias will reduce the performance of the circuits and increase the manufacturing cost. In this paper, we present a heuristic algorithm to eliminate the vias in the three-layer routing instances using the hybrid sense method. Some associated constraints under practical considerations, such as restricted terminals and adjacent limitation, will be addressed and solved extensively. By our experiments, the algorithm is fast and efficient to generate very good solutions.

Index terms-Via minimization, layer assignment, channel routing, NP-complete.

I. INTRODUCTION

THE ROUTING PROBLEM is an important field in VLSI/LSI or PCB layout. In the current technologies, three layers are available for routing the interconnections of nets. There correspondingly are some efficient three-layer routing algorithms

[8-11] based on VHV or HVH mode. The assumption of both modes can efficiently avoid the possibilities of unacceptable short-crossing, that is, the crossings of different nets on the same layer. It would, however, produce a large number of vias in the layout. From a technological point of view, the number of vias should be kept as small as possible. In IC processing, the increase of vias will cause the decrease of yield significantly and the circuit performance will be reduced. Additionally, the increase of vias has the penalty of manufacturing cost increased. Hence, the via minimization has to be considered in the routing problem.

The *Constrained Via Minimization* (CVM), which is also called the layer assignment, assumes that the topologies of all nets are known. That is to say, the topology of layout is fixed while the wire segments are to be reassigned to the new layers in order to eliminate the unnecessary vias. The example of three-layer CVM problem is sketched in Fig. 1. Some of vias in the layout drawn in Fig. 1(a) are reduced by the layer reassignment of all wire segments, and the final result with one via only is shown in Fig. 1(b). This final result has the fixed topology of interconnections as shown in Fig. 1(a). In the opposite way, if the function of via minimization is embeded into the routing algorithms simulaneously, it is called as *Unconstrained Via Minimization* (UVM), The UVM problem was first introduced by Hsu in 1983 [13]. The aim of UVM is to minimize the number of vias based on the net list information without considering either the geometry of the region or design rules. But note that, in this paper, only the CVM problem is under consideration.

The algorithms proposed for CVM problem in the two-layer routing were presented by [1]-[7]. Via reduction of 20 percent or more can often be achieved. The two-layer CVM problem is not NP-complete when the junction degree (the number of segments connected to one via) is less than or equal to 3 and it can be solved in polynomial time. With the advent of VLSI technology, three-layer routing becomes feasible [8]-[11]. Therefore, CVM porblem for three-layer routing is needed to be solved. Chang and Du [12] first formulated the three-layer CVM (3CVM) problem and showed it is an *NP-complete* problem.

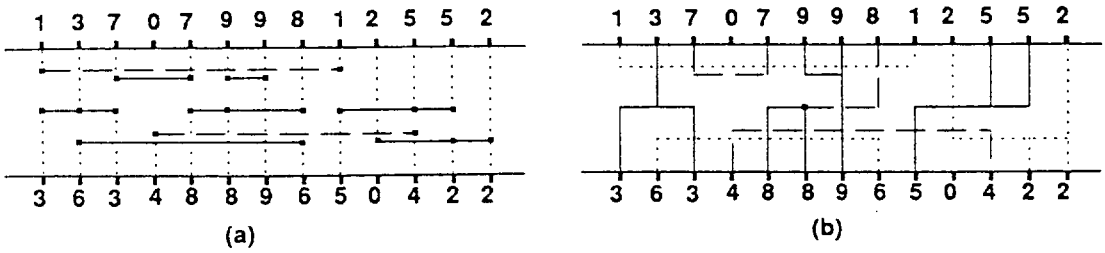

Fig. 1(a) The original layout. (b) Final layout.

Both *local sense and global sense* are considered in the algorithms to solve the three-layer CVM problem. An algorithm with local sense checks vias one by one to see if the via can be eliminated by reassigning new layers to the wire segments connected by this via. For global sense, the given topology is assumed to be *"layerless"* routing, and the minimization of vias is done by assigning globally all wire segments in the routing to new layers so that vias can be produced as small as possible. As stated above, the local sense algorithm would be seriously influenced by the given layer assignment of each wire segment because it considers only the condition in the neighborhood of a via. Due to this property, the algorithms with local sense are suitable to the problems with small size and these will become inefficient when the problem size is grown. In general, global sense algorithm is better than the local sense one.

Few papers were proposed for solving 3CVM problem up to the present. The heuristic algorithm presented by Chang and Du [12] was to minimize the vias with local sense. Hence, the algorithm is efficient to the routing examples with small problem size. Our first version [15] eliminated vias by global sense method and presented better results than those published in the paper [12].

In this paper, we present a hybrid sense algorithm for solving three-layer constrained via minimization problem. The basic idea is that, for any problem instance, we use the global sense method to minimize the vias first and then the local sense methods as the *initial* and *post-improving phases* of our hybrid sense algorithm, respectively. Although the result generated from the initial phase with global sense method is reasonably good, there still are some vias which can be further eliminated from the objected instance, however. The post-improving phase with local sense method is used to eliminate vias further after the initial phase. Fig. 2 sketches the

basic flow of our hybrid sense algorithm. The details will be discussed in the sections III and IV.

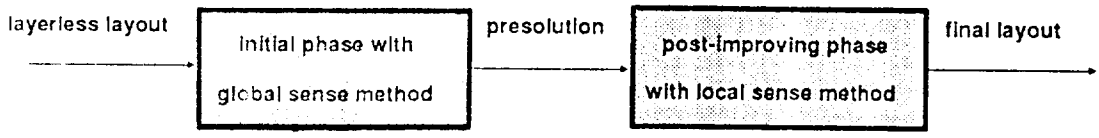

Fig. 2 The basic flow of the hybrid sense algorithm.

In practical applications, several special constraints must be taken into account, To solve the CVM problem, generally, the terminals are allowed to be available for any layer. But in practical, there may be only some layer(s) allowed. This is what we call *restricted terminals*. In addition, vias, are only allowed to connect wire segments in the adjacent layers, will get more important in VLSI design. Hence, the conception of *adjacent limitation* is obtained. For example, via A connects wire segments in first-layer and second-layer, but via B connects wire segments in first-layer and third-layer. It is clear that via A is more feasible than via B in consideration of performance and processing cost. Our previous paper [16] first took these associated constraints into account for practical applications. But, it was a global sense one.

Under these special constraints, we define four styles for the constrained via minimization problem as follows:

(1) GO_THROUGH style: There is no constraint on both terminals and vias.

(2) ADJACENT style: Vias are only allowed to connect wire segments in the successive layers, such as layer i and layer $i+1$.

(3) 1r_TERMINAL style: The terminals are restricted to only one layer (or only one layer is available for terminals) and there is no constraint on vias.

(4) 2r_TERMINAL style: The terminals are restricted to any of two specified layers and there is no constraint on vias.

It is obvious that the constraints of both adjacent limitation and restricted terminals can be mixed to form two new styles. But they are just trivial and therefore omitted in this paper.

In the next section, we introduce some foundations and the graph model for CVM problem. Based on this graph model, we do via minimization by coloring the associated graph in some special ways. Our hybrid sense algorithm for

GO_THROUGH style is presented in Section III and the initial phase is discussed right here. Section IV addresses the details about the post-improving phase. The associated constraints of restricted terminals and adjacent limitation are solved in Section V by some modifications of the algorithm presented in Section III. Several big examples are used to evaluate the algorithms. The results are satisfactory and shown in the last section.

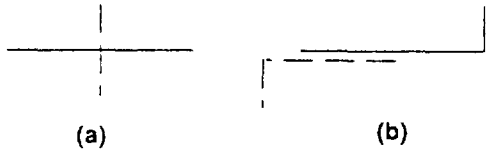

Fig. 3 (a) Crossover of two wire segments.
(b) Overlap of two wire segments.

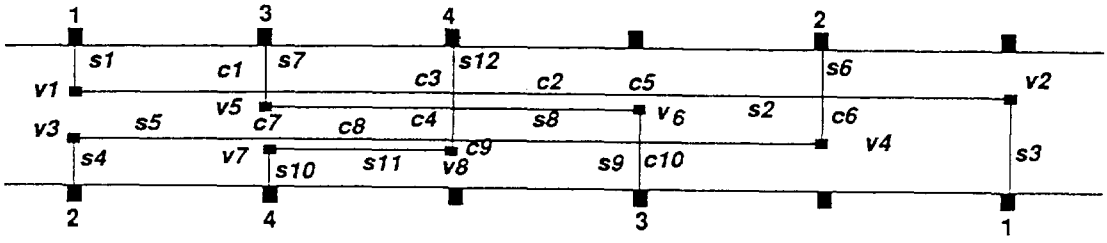

Fig. 4 The transient routing.

II. FOUNDATIONS AND GRAPH MODEL OF CONSTRAINED VIA MINIMIZATION

For understanding of the CVM problem, some terminologies should be introduced first.

A *net* is a collection of wire segments that electrically connects a set of terminals or pins. A *via* is a feedthrough hole or contact where the wire segments of a net on different layers are connected. A *crossing* is an intersection where two wire segments of different nets are intersected each other. There are two types of crossing as shown in Fig. 3. In the original layouts, the layers are divided into two classes of H-layer and V-layer which are used for routing the horizontal and verti-

cal wire segments, respectively. For example, in HVH routing mode, there are two H-layers and one V-layer. A *transient routing* is an incomplete or *"layerless"* physical routing in which the topology of nets in routing has existed, whereas the wire segments in the routing have not yet been assigned to layers. Fig. 4. is an example of transient routing.

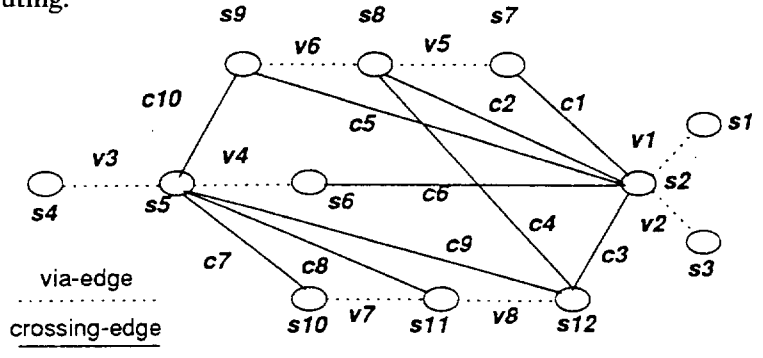

Fig. 5 Segment-crossing graph.

The transient routing can be represented as a graph called *Segment-Crossing Graph* (SCG) denoted as $G_s=(V_s, E_v \cup E_c)$. The graph is constructed as follows. Each vertex in V_s represents a wire segment in the transient routing. The edges in G_s are divided into two sets including the set of via-edges (denoted as E_v) and the set of crossing-edges (denoted as E_c). If there is a via connecting two wire segments, then an edge in E_v will connect the two corresponding vertices in SCG. An edge in E_c, which is incident on two vertices in SCG, means two corresponding wire segments are crossover each other. We use dot lines and solid lines for two different kinds of via-edges and crossing-edges, respectively. For the transient routing of Fig. 4, its associated SCG is illustrated in Fig. 5.

For each vertex $u \in V_s$, we define a set of vertices, which are adjacent to u by the edges in E_v, to be an adjacent list of u, and the other set of vertices which are connected to u by the edges in E_c is defined as a cluster of u. The adjacent list and the cluster of vertex u are denoted as $A(u)$ and $C(u)$, respectively.

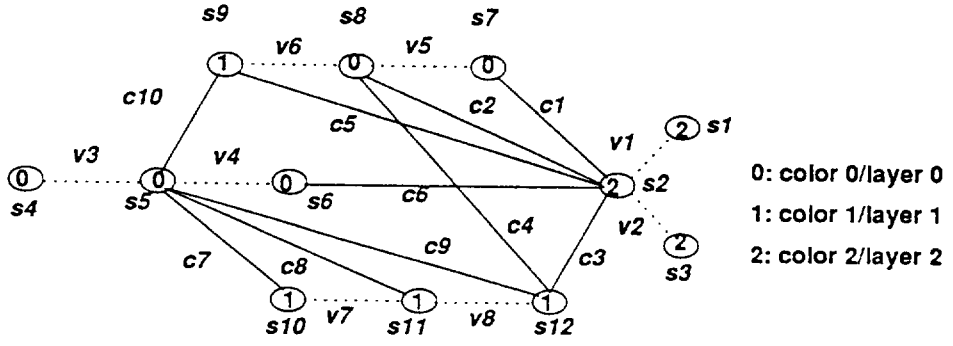

Fig. 6 Color assignment on the segment-crossing graph.

Let each layer in three-layer routing be represented by one color. The layer assignment of wire segments in the transient routing is to assign each vertex in the *'colorless'* graph G_s one color such that any two vertices, which are incident with an edge in E_c, are painted by different colors. On the other hand, any two vertices joined by an edge in E_v are painted as the same color as possible. In Fig. 5, it is obvious that when any two vertices incident with the via-edge are assigned to the same color, the corresponding via will minimized in the original routing. Based on this observation, we try to assign vertices incident with via-edges the same colors and vertices incident with crossing-edges the different colors (because of no violation to electrical requirement). The more vertices incident with the via-edges are assigned the same colors, the more vias are eliminated, Fig. 6 is a colored SCG of the graph shown in Fig. 5 and its corresponding layout is given in Fig. 7 with one via only.

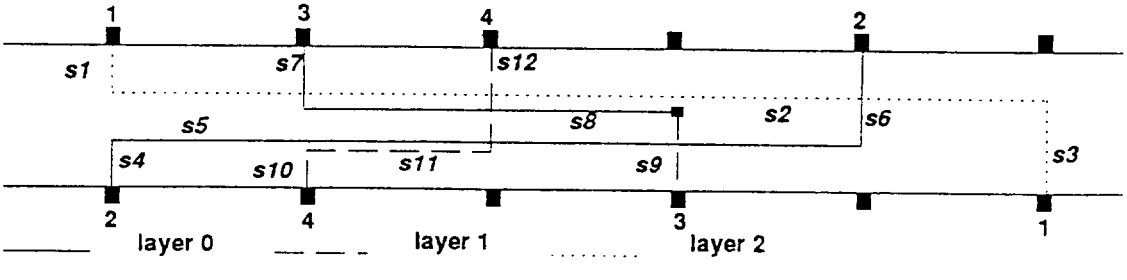

Fig. 7 The physical routing of Fig. 6.

III. HYBRID SENSE ALGORITHM FOR GO_THROUGH STYLE

The 3CVM was shown to be NP-complete [12]. Therefore, a heuristic algorithm with hybrid sense method is proposed in this section. Let us first consider the segment-crossing graph $G_s=(V_s, E_v \cup E_c)$ and two sets of vertices associated with each vertex $u \in V_s$. One is the adjacent list of u, and the other is the cluster of u. The colors assigned to the vertices in $A(u)$ are the *recommended colors* of vertex u. If the vertex u is colored with one of the recommended colors, some vias contacting to the wire segment corresponding to u may be eliminated. The colors assigned to the vertices in $C(u)$ are the *constrained colors* of vertex u. Any constrained color of vertex u indicates a color to which the vertex u cannot be assigned for avoidance of violating the electrical requirement. Therefore, the *expected colors* of vertex u are the colors with belong to the recommended colors of vertex u but not the constrained colors of vertex u. If vertex u can be assigned to one of the expected colors then there will be some vias eliminated.

A color that has been assigned to a vertex u is called *active color* of u denoted as $a(u)$. The vertex u that has assigned a color is called *active vertex*. Oppositely, it is called *inactive vertex* if it has not been assigned a color. Let $r(u)$, $c(u)$, and $e(u)$ be the collections of recommended colors, constrained colors, and expected colors of vertex u, respectively.

In summary, $r(u)$, $c(u)$, and $e(u)$ can be expressed in the following formulas.

$$r(u)=\{a(i) \mid i \in A(u)\}. \quad (1)$$

$$c(u)=\{a(j) \mid j \in C(u)\}. \quad (2)$$

$$e(u)=r(u)-(r(u) \cap c(u)). \quad (3)$$

In Fig. 8, it illustrates these definitions for demonstration.

A vertex u is called *p-constrained vertex* (or called *pC* vertex for simplicity) if $|c(u)|$ =p, where $|c(u)|$ is the number of distinct colors in $c(u)$. For the example of Fig. 8, vertex u is a 1C vertex.

A. THE HEURISTIC ALGORITHM

Using the definitions above, our hybrid sense algorithm can be presented step by step as below and the details will be addressed later.

ALGORITHM VIA-MINIMIZATION

/*The initial phase*/

1. input a layout and construct a segment-crossing graph G_s, and let all vertices in G_s be inactive vertices;
2. **REPEAT**
3. select an inactive vertex u with highest priority in G_s;
4. **IF** vertex u is 3C vertex **THEN** BACKTRACK(u);
5. **ELSE** assign vertex u to a color, $a(u)$;
6. calculate $r(i)$ by $r(i)=r(i) \cup a(u)$, for all $i \in A(u)$.
7. calculate $c(i)$ by $c(i)=c(u), \cup a(u)$, for all $i \in C(u)$.
8. **UNTIL** all vertices in G_s are active vertices.

/*The post-improving phase*/

9. POST-IMPROVE;

END ALGORITHM VIA-MINIMIZATION

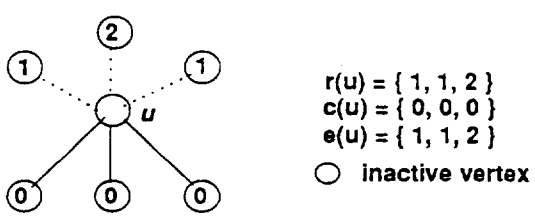

Fig. 8 The example of definitions of vertex u.

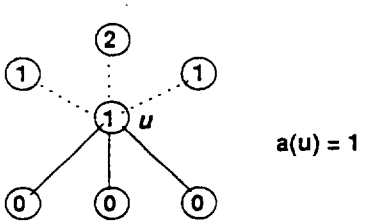

Fig. 9 The example of color assignment.

In the REPEAT-UNTIL loop, it makes use of global sense method to minimize the vias. We call this portion as the *initial phase* in our hybrid sense algorithm. In the last step, the POST-IMPROVE procedure with local sense method is used to enhance the outcomes generated from the initial phase. Accordingly, the *post-improving phase* is defined simultaneously. The details of the post-improving phase will be addressed in Section IV.

Initially, all vertices in G_s are inactive vertices. One of them will be selected to be assigned a color. Because the result of the algorithm depends on the order of selecting the inactive vertices, we use a priority function to determine the sequence of the selected vertices. The priority function is estimated at each vertex in G_s.

Step 3 selects an inactive vertex based on the nonincreasing order of priority of the vertices. Each inactive vertex will be classified into one of four kinds based on the $|c(u)|$, that is, it is going to be one of 0C, 1C, 2C, and 3C vertices when there are three colors needed for the three-layer routing. The highest priority vertex is 3C vertex and lowest one is 0C vertex. If some inactive vertices are the same kind, one with maximal degree of constraints, i.e., the maximal number of $|C(u)|$, will be selected first from them. Initially, all vertices in G_s are 0C vertices. In formula, the priority function of the inactive vertex u is defined as

$$P(u) = |c(u)| * deg\_max + deg(u); \quad (4)$$

where deg_max is the maximum degree of all vertices in $G_s^c = (V_s, E_c)$ and $deg(u)$ is the degree of vertex u in G_s^c.

This priority function first selects the vertex with maximal $|c(u)|$ and assigns it an appropriate color later. By doing such action, it will help in decreasing the complexity of constraints and the possibility of backtrackings when the program proceeds. If there is a tie among many inactive vertices with equal $|c(u)|$, the vertex with maximal degree in G_s^c is selected for tie-breaking. A vertex with more degree in G_s^c will affect more other vertices during color assignment of G_s. For the same meaning, the factor $deg(u)$ in formula (4) also helps in decreasing the complexity of constraints and the possibility of backtrackings when the program proceeds. In short words, for good results the vertex with highest priority had better be selected and assigned to an appropriate color first. Moreover, if there is still a tie occurring in formula (4), arbitrary one among these inactive vertices with equal priority will be selected.

In the case of 3C vertex, $a(u) = \phi$, that is, no available color can be assigned to vertex u. In order to hold the 3-layer requirement, the color reassignments of some active vertices in $C(u)$ are needed during the color assignment of 3C vertex. The BACKTRACK procedure is to reassign some active vertices in $C(u)$ to new colors such that the 3C vertex u can be assigned to a legal color. The apporach of backtracking will be discussed in the subsection B.

After a vertex is selected by the priority function, an active color has to be assigned to this selected vertex. Let $L = \{0,1,2\}$ be a set of colors corresponding to the three layers used in the routing. The active color of vertex, $a(u)$, must be one in the set of L-$c(u)$. The best color is determined as follows. Let $|r_\tau(u)|$ be the

number of color τ in $r(u)$ and $\tau \in L$. To eliminate the maximum number of vias, the color assigned to an inactive vertex u will be $a(u)$ satisfying the feasible condition that is defined below.

$$a(u)=\tau,\ \tau \in e(u) \text{ and } |r_\tau(u)| \text{ is maximum.} \quad (5)$$

For the example illustrated in Fig. 9, $|r_1(u)|$ and $|r_2(u)|$ are equal to two and one, respectively. According to formula (5), the active color of vertex u, i.e., $a(u)$, is set to color 1, therefore.

B. THE BACKTRACKING PROCEDURE

The backtracking will be treated to reassign some of active vertices to new colors so that any 3C inactive vertex (or 3C vertex for simplicity) can be assigned to a legal color. Since all colors in L have been assigned to the vertices in the cluster of the 3C vertex u, a color, which is assigned to some of vertices in $C(u)$, is selected to be the active color of the 3C vertex u. Each vertex in $C(u)$ may be one of 0C, 1C or 2C vertex. If a color assigned to the 2C vertex in $C(u)$ is selected for the color assignment of the 3C vertex u, this 2C vertex will change to be a 3C inactive vertex such that the work of backtracking will be continued. In the case of 0C or 1C vertex in $C(u)$, the backtracking are terminated because the vertex is not going to be a 3C vertex when its color is assigned to the 3C vertex u. The requirement of selecting a color is to minimize the number of active vertices which will become 3C vertices after this color is assigned to the current 3C vertex. For the example of Fig. 10, all wire segments except s_u are active and $C(s_u)=\{s_i, s_j, s_k\}$. The wire segments s_i, s_j, and s_k are corresponding to the 0C, 1C, and 2C vertices, respectively, and the wire segment s_u corresponds to 3C inactive vertex because of $|c(s_u)|=3$. The color assigned to the 0C vertex s_i will be selected as a color for the vertex s_u, since the vertex s_i will change to be 1C inactive vertex that has more flexibility to reassign. But it is not reasonable to select the color of the vertex s_k, because the vertex s_k will be 3C inactive vertex after its color is assigned to s_u. The algorithm of backtracking is presented in the following.

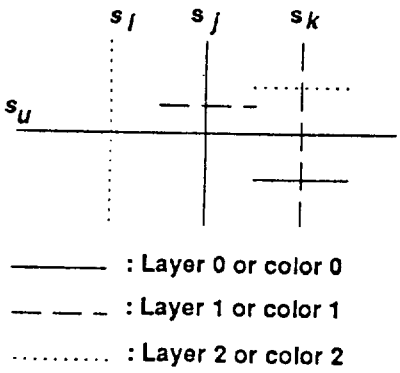

Fig. 10 An example for backtracking.

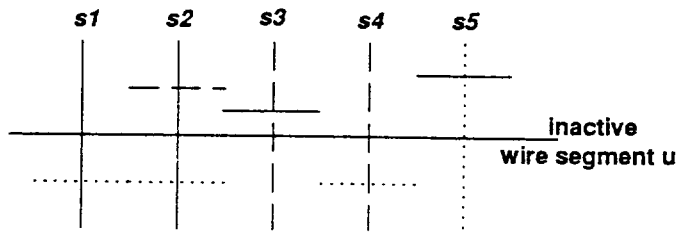

Fig. 11 An example for termination in backtracking.

PROCEDURE BACKTRACK(u)/*the u is a 3C vertex*/

1. calculate the $k_\tau(u)$ for τ =0, 1, and 2;
2. calculate the $|c_\tau(u)|$ for τ =0, 1, and 2;
3. $a(u)=\tau$, $k_\tau=(u)$ and $|c_\tau=(u)|$ are minimized such that there are fewest marked 2C vertices with active color τ in $C(u)$;
4. **FOR** all i in $C(u)$; **DO**
5. $c(i)=c(i) \cup a(u)$;
6. **IF** $(a(i)=a(u))$ **THEN** INACTIVE(i);
7. **END**
8. calculate $r(i)$ by $r(i)=r(i) \cup a(u)$ for all i in $A(u)$;
9. vertex u is marked;

END PROCEDURE BACKTRACK

PROCEDURE INACTIVE(u)

1. claculate $r(i)$ by $r(i)=r(i)$-$a(u)$ for all i in $A(u)$;
2. claculate $c(i)$ by $c(i)=c(i)$-$a(u)$ for all i in $C(u)$;
3. set vertex u to be inactive;

END PROCEDURE INACTIVE

Two parameters are needed in the BACKTRACK procedure. Note that only

one of three colors in the layer assignent will be selected for an active color of the 3C inactive vertex. Both parameters of $k_\tau(u)$ and $|c_\tau(u)|$ are used to determine the active color assigned to the 3C vertex. The parameter $k_\tau(u)$ is defined as follows.

$$k_\tau(u) = max(|c(i)|, \text{ for all vertices } i \in C(u) \text{ and } \tau = a(i)), \quad (6)$$

The $k_\tau(u) = p$ means some of vertices in $C(u)$, which have been assigned a color τ, are going to be $(p+1)$ constrained vertices after the τ color is selected as an active color of the 3C vertex u. For the example in Fig. 11. The inactive wire segment u corresponds to the 3C vertex and $k_0(u)=2$, $k_1(u)=1$, $k_2(u)=1$. If the color 0 is assigned to the 3C vertex u, the wire segment *s2* will become a 3C vertex, meanwhile, the backtracking will be continued. Therefore, the assigned color for 3C vertex u may be 1 or 2.

The parameter $|c_\tau(u)|$ is defined to be the number of active vertices which will become the inactive vertices after color τ is assigned to the 3C vertex u. $|c(u)|$ is calculated below.

$$|c_\tau(u)| = \text{The count of vertices } i \text{ for } a(i) = \tau \text{ and } i \in C(u). \quad (7)$$

If we choose τ to be the active color of 3C vertex u, there will be $|c_\tau(u)|$ active vertices in $C(u)$ which will turn to be inactive vertices. Of course. the number of these vertices should be kept as small as possible.

From the example in Fig. 11, one of colors 1 and 2 can be used as an active color of 3C vertex u. However, color 2 is more preferable as an active color of 3C vertex since $|c_1(u)| = 2$ and $|c_2(u)| = 1$. The active color $\tau = min(|c_\tau(u)|$, for $\tau = 0$, 1, and 2) means the resulting iterations in the backtracking process is minimal after color τ is assigned to the 3C vertex. Therefore, the active color is determined in the following formula:

$$a(u) = \tau,\ k_\tau(u) \text{ and } |c_\tau(u)| \text{ are minimized such that there are fewest marked 2C vertices with active color } \tau \text{ in } C(u). \quad (8)$$

After assigning a color to a 3C vertex u, we mark this vertex. In formula (8), the selected τ also minimizes the number of 2C vertices (if exist), which have been marked by this procedure, in the cluster of u. The marks of vertices are used for avoiding infinite cycle occurring in the procedure. For details, please refer to our paper [15].

C. THE TIME COMPLEXITY OF THE INITIAL PHASE

Initially, to construct the associated Segment-Crossing Graph G_s, i.e., to calculate the adjacent lists and clusters of all vertices in G_s, it is done by checking the connectivity and crossings of each segment with the other segments in layerless layout. Hence, it is clear the time complexity for constructing G_s is $O(\zeta^2)$; where $\zeta = |V_s|$ is the number of segments in the objective layout.

In the initial phase it is not sure that how many vertices will be backtracked, i.e., the number of 3C vertices occurred cannot be predicted. As a consequence, we isolate the BACKTRACK procedure from the initial phase for the analysis of time complexity.

To achieve the most efficient selection of an inactive vertex with highest priority, a *sorted bucket list* is utilized. The data structure and the associated operations of the sorted bucket list can be referred in [17]. After an inactive vertex u is selected, the number of updatings of recommended and constrained colors of the related vertices (steps 6-7 in VIA-MINIMIZATION algorithm) are $|A(u)|$ and $|C(u)|$, respectively. Now it can be addressed that the time complexity of coloring $G_s=(V_s, E_v \cup E_c)$ is $O(|E_v|+|E_c|) \leq O(\zeta^2)$ if there is no backtracking occurred.

In BACKTRACK procedure, the calculations of $k_r(u)$ and $|c_r(u)|$ are in $O(deg\_max)$. But the updatings of recommended and constrained colors of the related vertices (steps 4-8 in BACKTRACK procedure) are in $O(deg\_max^2)$. Therefore, the time complexity of the BACKTRACK procedure is $O(deg\_max^2)$.

From our emperical experience, the number β of backtrackings is small and it is reasonable to expect that time complexity $O(\beta * deg\_max^2)$ is lower than time complexity $O(\zeta^2)$ because deg_max is much smaller than ζ and β is assumed to be a small constant.

Based on the above discussions, therefore, the time complexity of the initial phase is $O(\zeta^2)$ under one reasonable assumption.

IV. POST-IMPROVEMENT

The solution achieved by the initial phase of VIA-MINIMIZATION algorithm is reasonably good. However, there still may be some vias which can be removed from this solution further. We call this solution as the *presolution.*

In a presolution, if there exists any vertex u and $e(u)$ is not a null set, in addition, $a(u)$ is not in $e(u)$, then the via associated with the corresponding wire segment u may be eliminated. In Fig. 12, for example, $e(u)=\{0, 0\}$ and $a(u)=2$. Therefore, if we reassign $a(u)$ to color 0 then one via will be eliminated. But assuming $a(v)=2$, it may be of no help to reassign $a(u)$ to color 0. The local sense algorithm for post-improvement is shown as below.

PROCEDURE POST-IMPROVE

```
1. REPEAT
2.    δ =0;
3.    FOR each vertex u in G_s DO
4.       IF (e(u) ≠ φ and a(u) ∉ e(u) THEN
5.       {rendomly select k ∈ e(u);
6.        INACTIVE(u);
7.        a(u)= k;
8.        UPDATE(u);
9.        δ = δ +1; }
10.   END /*END of FOR*/
11.   put δ into buffer B;
12. UNTIL (( δ =0) or ( δ is stable))
```

END PROCEDURE POST-IMPROVE

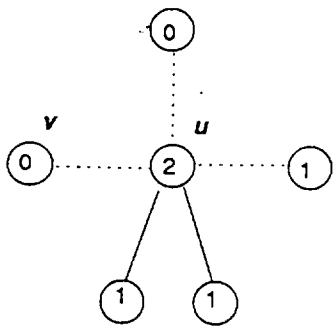

Fig. 12 A subgraph of G_s according to a presolution.

PROCEDURE UPDATE(u)

/*refresh associated recommended and constrained colors*/

1. calculate $r(i)$ by $r(i)=r(i) \cup a(u)$ for all $i \in A(u)$;
2. calculate $c(i)$ by $c(i)=c(i) \cup a(u)$ for all $i \in C(u)$;

END PROCEDURE UPDATE

In POST-IMPROVE procedure, δ is a counter to count the number of active vertices which have been reassigned to new colors. In each iteration of the REPEAT-UNTIL loop, we store δ into a buffer.

Ther terminating postulate is that $\delta = 0$ or δ is stable after several iterations. $\delta = 0$ means there is no active vertex to be reassigned to new color. On the other hand, we can check whether δ is stable or not after a few of iterations by checking the values stored in buffer B. We think δ is stable when the recent five δ's are equal or differed within the tolerance of a specified constant. These two conditions both mean that there will be no via which can be eliminated any more. By experimental experience, the POST-IMPROVE procedure converges within ten iterations in the most cases.

Anyway, it is impossible to predict exactly how many vertices will be reassigned to new colors in each iteration and also how many iterations will be processed in the POST-IMPROVE procedure. Thus, the time complexity of the POST-IMPROVE procedure is derived under some assumptions. As mentioned above, δ is much smaller than ζ, the number of segments in the objective layout, in each iteration. Since the number of iterations required is limited within a small constant α, $\alpha^* \delta$ may be quantified as ζ. Hence, the time complexity of the POST-IMPROVE procedure is $O(\zeta)$ under some assumptions.

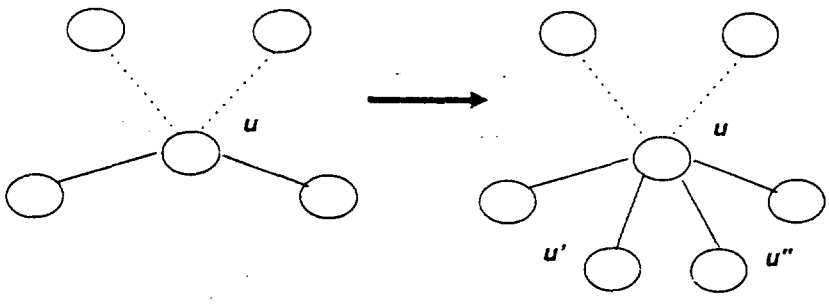

Fig. 13 Appending two dummy vertices u', u'' to vertex u.

V. RESTRICTED TERMINALS AND ADJACENT LIMITATION

Under practical considerations, the constraints of adjacent limitation and restricted terminals are very of importance and suitable for real applications. Adjacent limitation does not allow vias to connect wire segments which are not in successive layers. This constraint can enhance the circuit performance and reduce the processing cost of the design with comparison to the unconstrained one. Furthermore, in the most VLSI designs, the terminals are not neccessarily free to all layers. Thus, terminals restricted to one or two layers are usually used. In the following, we will address the appropriate modifications for meeting these constraints.

A. MODIFICATIONS OF THE ALGORITHM FOR 1r_AND 2r_TERMINAL STYLES

In 1r-TERMINAL style, terminals are available for one particular layer, each associated vertex of wire segment connected to the terminal can be appended by two *dummy* vertices. Oppositely those segments, which are not connected to terminals, may not be appended by any dummy vertex. In Fig. 13, two dummy vertices of a certain vertex u, which corresponding segment is connected to at least one terminal, are added to the cluster $C(u)$. After these, the dummy vertices are set to be active and painted by the suitable colors. For example, the terminals are restricted to the first layer (color 0), two dummy vertices of u are assigned to colors 1 and 2. They will force the associated vertex u to be only painted by the color 0 because there are at least two colors 1 and 2 in the constrained colors $c(u)$. It means that the wire segments connected to the terminal are all located at first layer. Of course, these dummy vertices should be fixed in color assignment, i.e., no reassigning color to any dummy vertex. For the case of terminals restricted to two specified layers, namely, in 2r_TERMINAL style, we only use a dummy vertex to force the corresponding wire segment, which is connected to at least one terminal, to be assigned to one of these two specified layers.

This dummy vertex model is very flexible and easy to be implemented (or less modifications) for any variation of restricted terminals constraints. It also has the potential for extension in multi-layer (n>3) CVM problem.

B. MODIFICATIONS OF THE ALGORITHM FOR ADJACENT STYLE

However, if we want to obtain the ADJACENT style solution, then the wire segments will be classified into two categories consisting of horizontal and vertical wire segments, Hence, the corresponding vertices are divided into two types of H-vertices and V-vertices, respectively. The priorities of H-vertices are higher than the V-vertices. Namely, the priority function in formula (4) is modified as

$$P'(u)=K*H(u)+P(u); \tag{9}$$

where $P(u)$ is defined in formula (4), $K=4*deg\_max+1$,

and if u is a H-vertex then $H(u)=1$, else $H(u)=0$.

From formular (4), it is clear that the possible highest priority associated with V-vertices is $P(u)=3*deg\_max + deg(u)$ which is less than or equal to $4*deg\_max$. If we expect that the priorities of H-vertices are always higher than V-vertices, constant K should be greater than any possible priority associated with all V-vertices. As a consequence, K is defined as $4*deg\_max+1$.

In the initial phase of VIA-MINIMIZATION algorithm, if we have to meet the constraint of color decision in consideration of the ADJACENT style, the layer assignments of horizontal wire segments are limited to be H-layers. But the layer assignment of each vertical wire segment can be assigned to any layer under adjacent limitation. In other words, for example, assume that the corresponding colors for coloring the SCG are 0, 1 and 2 in HVH routing mode. Hence, the H-vertices are limited to colors 0 and 2. But the V-vertices can be assigned to any color so long as this color is helpful to eliminate vias and legal to adjacent limitation in the initial phase of the hybrid sense algorithm. Anyway, the horizontal wire segments are not limited to H-layers in our solutions. After the execution of POST-IMPROVE procedure, i.e., the post-improving phase, the horizontal wire segments can be assigned to V-layers so as to eliminate vias further.

For the requirement of adjacent limitation, we must check the legality of the color assignment of any selected vertex in all the main program and other procedures. For instance, a vertex u and $r(u)=\{0\}$, it is illegal to assign this vertex u to color 2 even though color 2 is the best one for some criteria. In consideration of the adjacent limitation, only colors 0 and 1 are legal for coloring vertex u when $r(u)=\{0\}$.

VI. RESULTS AND CONCLUSIONS

The algorithm for three-layer via minimization was coded in *C* language and implemented on *SUN 4* workstation. The algorithms are efficient and have been evaluated on some examples with big problem size including examples *1, 3a, 3b, 3c, 4b,* and *5* presented in [8] and example *diff* is the Deutsch's difficult example [14]. The original layouts of these examples are all in three-layer HVH mode. In this paper, the results of example with small problem size in GO_THROUGH style can be referenced in [15]. They are better than the ones published in [12]. In the paper, [12], only the examples with small problem size were solved in GO_THROUGH style.

There are four styles to be tested in this paper. Table I shows the results of our algorithms for all styles. With comparison to GO_THROUGH style in paper [15], shown in Table II, this new hybrid sense algorithm can eleminate more vias for the original test examples on the average. It is clear that the improvement is achieved by the post-improving phase of the hybrid sense algorithm. Note again, the algorithm proposed in [15] was a global sense one and equivalent to the initial phase of the hybrid sense algorithm for GO_THROUGH style.

Table I also shows the results for ADJACENT, 1r_TERMINAL and 2r_TERMINAL styles. It has better outcomes than those published in [16] that has first addressed the problems of terminals restrictions. Table III shows the comparisons with [16]. As the data shown, it seems very reasonable to have these outcomes. That is, the more flexible the constraint is, the better the solution is obtained.

In this paper, we present a heuristic algorithm with hybrid sense for solving the problem of three-layer constrained via minimization. Some associated constraints are addressed and solved extensively. Besides the original CVM problem, the additional constraints of restricted terminals and adjacent limitation result into four styles of solutons.

Table I: The effections of via minimization on four styles solution

Examples (See Ref[8])	GO_THROUGH	ADJACENT		
		No terminal constraint	1r_TERMINAL	2r_TERMINAL
	No. of vias eliminated	No. of vias eliminated	No. of vias eliminated	No. of vias eliminated
1	26	21	2	11
3a	31	27	9	21
3b	79	70	41	65
3c	63	50	23	40
4b	69	60	8	27
5	81	54	4	20
diff	87	64	13	50

Table II

Ex.	Ours	[15]
	No. of Vias eliminated	No. of Vias eliminated
1	26	21
3a	31	24
3b	79	26
3c	63	62
4b	69	45
5	81	76
diff	87	86

Table III

Ex.	1r_TERMINAL		2r_TERMINAL	
	Ours	[16]	Ours	[16]
1	2	2	11	13
3a	9	1	21	12
3b	41	12	65	23
3c	23	5	40	19
4b	8	16	27	50
5	4	6	20	20
diff	13	17	50	48

REFERENCES

1. A Hashimoto and J. Steven, "Wire routing by optimizing channel assignment within large apertures," in *Proc. 8th Design Automation Workshop,* Santa Barbara, California, pp. 155-169. June 1971.
2. Y. Kajitani, "On via hole minimization of routing in a 2-layer board", in *Proc. IEEE 1980 Int. Conf. Circuit Computers,* Orlando, Florida, pp. 295-298, June 1980.
3. M. J. Ciesielske and E. Kinnen, "An optimum layer assignment for routing in IC's and PCB's," in *Proc, 18th Design Automation Conf.,* Nashville, Tennessee, pp. 733-737, June 1981.
4. R. Y. Pinter, "Optimal layer assignment for interconnect," *J. VLSI and Compuer Systems,* vol. 1, no. 2, 1984, pp. 123-137.
5. K. C. Chang and H. C. Du, "Efficient algorithms for layer assignment problem," *IEEE Trans. Computer-Aided Design,* vol. CAD-6, no. 1, pp. 67-78, JAN. 1987.
6. X. M. Xiong and E. S. Kuh, "The constrained via minimization problem for PCB and VLSI design," in *Proc. 25th Design Automation Conf.,* Anaheim, California. pp. 573-578, 1988.
7. Y. S. Kuo, T. C. Chern and W. K. Shih, "Fast algorithm for optimal layer assignment," in *Proc, 25th Design Automation Conf.,* Anaheim, California, pp. 554-559, 1988.
8. Y. K. Chen and M. L. Liu, "Three-layer channel routing," *IEEE Trans. Computer-Aided Design,* vol. CAD-3, no. 2 pp. 156-163, Apr. 1984.
9. V. Pitchumani and Q. Zhang, "A mixed HVH-VHV algorithm for three-layer channel routing," *IEEE Trans. Computer-Aided Design,* vol. CAD-6, no. 4, pp. 497-502, July 1987.
10. P. Bruell and P. Sun, "A 'Greedy' three layer channel router," in *Proc. ICCAD-85,* pp. 298-300, Nov. 18-21, 1985.
11. F. P. Preparata and W. Lipski, "Optimal three-layer channel routing," *IEEE Trans. Comput.,* vol. C-33, no. 5, pp. 427-437, May 1984.
12. K. C. Chang and H. C. Du, "Layer assignment problem for three-layer routing,"

IEEE Trans. Comput., vol. C-37, no. 5, pp. 625-632, May 1988.

13. C. P. Hsu, "Minimum-via topological routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, no. 4, pp. 235-245, Oct. 1983.
14. D. N. Deutsch, "A dogleg channel router," in *Proc. 20th Design Automation Conf.*, pp. 591-597, June 1983.
15. K. E. Chang, H. F. Jyu, and W. S. Feng, "Constrained via minimization for three-layer routing," *COMPUTER-AIDED DESIGN*, vol. 21, no. 6, pp. 346-354, July/August 1989.
16. K. E. Chang and W. S. Feng, "A 'delayed-layering' three layer channel routing", *IEE proc.*, vol. 137, part E, no. 4, pp. 229-238, July 1990.
17. C. M. Fiduccia and R.M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. 19th Design Automation Conference*, pp. 175-181, 1982.

有關三層 VLSI 佈局層指定的混合式演算法

張國恩

資訊教育研究所

摘　要

佈局層指定（又稱可限性穿孔減少）是決定佈局中各線段的佈局層位置使得佈局所產生的穿孔數能儘量少。由於穿孔數的增加會降低電路之執行效益和增加電路製造成本，因此減少佈局中的穿孔是重要的。本文提出一個有效的演算法以減少三層佈局中的穿孔數。文中採用一種混合式的方式，並考慮實際設計上的限制問題，如端點限制與鄰接限制。這些問題皆有助於VLSI電路之製造。經過實驗證明，本混合式演算法是快速而有效的，並得到很好的結果。