An Efficient Approach for Via Minimization in Multi-Layer VLSI/PCB Routing*

Jong-Sheng Cherng, Sao-Jie Chen
Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan 106, R.O.C.

Chia-Chun Tsai
Dept. of Electronic Engineering
National Taipei Institute of Technology
Taipei, Taiwan, R.O.C.

Jan-Ming Ho
Institute of Information Science
Academia Sinica
Taipei, Taiwan 115, R.O.C.

Abstract

An efficient track permutation technique and a powerful Constrained Via Minimization (CVM) approach are proposed for multi-layer routing of VLSI chips and PCBs. Track permutation technique combined with existing channel routing algorithms can improve the routing results by permuting the tracks. For our multi-layer CVM approach, two procedures -- *Maximum-Difference Reduction* procedure and *Backtracking* procedure are developed. In particular, the *Maximum-Difference Reduction* problem can be successfully converted to that of *Minimum Clique Number Augmentation*, and a polynomial-time heuristic algorithm is thereafter derived. The algorithms are evaluated by some famous routing examples using three and five layers. The obtained results, 38 percent of vias eliminated on an average, are very encouraging.

1. Introduction

Whenever multi-layer routing became essential in the design of VLSI and PCBs, vias have been used to establish multi-layer connections. But for the sake of performance, minimizing the number of vias in a layout is necessary. Most channel routing algorithms [1]-[3] assumed that a wiring layer can carry wires in only one direction, and thus generated a lot of vias. If this constraint on each layer is released, more vias will be potentially reduced.

The via minimization problem is to decide the topology of routing nets and the layer assignment of wire segments such that the number of vias in the layout is minimized. If the topology of a given layout is fixed, the problem is referred to as a *Constrained Via Minimization* (CVM) problem. Algorithms for the two-layer and three-layer CVM problems have already been explored extensively [4]-[9]. With the advance of VLSI technology, multi-layer (more than three layers) routing turns out to be more and more feasible and multi-layer CVM problem becomes an issue. Few papers addressed this kind of multi-layer CVM problem. This paper is concerned with the CVM problem in a multi-layer routing environment.

The algorithm presented by Chang and Du [7] checked vias one by one to examine whether the via can be eliminated by reassigning new layers to the wire segments associated with this via. The algorithm proposed by Chang, et al. [9] used a graph to represent the layerless routing and reassigned all wire segments to new layers such that vias can be created as few as possible. Both of the above two papers concentrated directly on solving coloring problems on intersection graphs. But in our work, we first permute the horizontal tracks to derive a better configuration before solving CVM problem and then transform intersection graphs into interval graphs, finally, we solve coloring problems on derived interval graphs. Here, it is well known that solving coloring problem on intersection graphs is *NP-complete*, whereas it takes *O(nlogn)* for interval graphs. Hence, our CVM algorithm is superior to previous works potentially.

2. Track Permutation

The via candidates of the input layout could be classified into unremovable, absolutely removable, and possibly removable categories [8]. Fig. 1 shows these three categories. Obviously, having more absolutely removable vias will generate a better layout result.

To solve the routing problem for the HVHV... (or VHVH...) model, we use an existing good routing algorithms [1] [2] [3] [11] to obtain routing solutions and then permute the horizontal tracks to increase the number of absolutely removable via. Let S be a k-layer routing solution using tracks $t_1, t_2, ..., t_d$ (each t_i consists of i-th track of layer 1, 2,..., and layer k). Let π be a permutation on {1, 2,..., d}. Not every track permutation is valid. It is clear that π is a valid track permutation if and only if $\pi(S)$ has no vertical overlap of wires from different nets.

To characterize valid track permutation, we define the *track ordering graph* $G(S)$ for k-layer routing solution S. Each node in $G(S)$ corresponds to a track and a directed edge from track t_i to track t_j means that track t_i must be placed above track t_j. Now, our goal is to obtain a valid track permutation π such that the number of absolutely removable via in $\pi(S)$ is maximized for a given S. Our approach generating optimal track permutation is shown as follows:

Algorithm *Track Permutation*

Step 1: Input a k-layer routing solution S.

Step 2: Construct the *track ordering graph* $G(S)$.

Step 3: Repeat substeps a and b alternately until the vertices of $G(S)$ are completely removed.

a. Find the set of vertices in $G(S)$ that have no incoming degree and pick from the set a vertex v which contributes a maximum number of absolutely removable vias to place on a position t_i (i from 1 to d/2), Finally, remove v and its incident edges from $G(S)$.

b. The process is the same as substep a, except that it finds a no outgoing-degree vertex set and places a vertex picked from the set on a position t_j (j from d to d/2).

*This work was supported by the National Science Council, R.O.C., under Grant NSC84-2215-E002-027

3. Algorithm for Multi-Layer CVM Problem

Besides the track permutation, we will propose an algorithm for the multi-layer CVM problem. Given a k-layer CVM problem instance, a *crossing graph* $G = (V,E)$ can be constructed as follows: Each vertex $v \in V$ represents a set of wire segments that have no crossing between them. Two vertices v_i, $v_j \in V$ are adjacent only if there are crossings between them. By using our algorithm, first, we obtain the solution of initial layer assignment consisting of $|V|$ layers where $|V|$ is the number of nets and each layer only contains the wire segments of a net. Next step, we want to generate an m_i-layer routing solution by introducing a minimum via set w_{i-1} to m_{i-1} -layer routing environment (where $|V| = m_0 > m_1 >...> m_i > ...= k$). This step is repeated until the resultant m_i-layer is equal to k-layer and meanwhile, the sum of w_i previously introduced is equal to the optimal via set for our algorithm.

Note that, in the process of m value reduction, we achieve the goal of [7], where eliminating the number of vias as many as possible, by introducing the number of vias as few as possible. Here, from another point of view, we are interested in the maximum difference D (where $D = m_i - m_{i+1}$) for each reduction by introducing some specified vias. We call this problem a *Maximum-Difference Reduction* problem.

We uses a graph-theoretic approach to solve the *Maximum-Difference Reduction* problem. When the crossing relationship between wire segments is represented by a crossing graph $G = (V,E)$, a possible solution for the *Maximum-Difference Reduction* problem can be characterized by an interval graph $G = (V, E \cup F)$, where F is called an augmentation. We then succeed to convert the *Maximum-Difference Reduction* problem into a *Minimum Clique Number Augmentation* problem. The latter problem tries to find a supergraph by adding a set of edges on the given crossing graph, to make this supergraph an interval graph having the least clique number. Unfortunately, this problem turns out to be *NP-complete* [10].

Before going further we introduce a few definitions related to interval graphs.

Definition 1 [10]: A clique is called *dominant* if it is not a proper subset of another clique, that is, it is maximal.

Theorem 1 [10]: A graph is an *interval graph* if and only if there exists an ordering of dominant cliques such that the v.d.c. matrix (vertex versus dominant clique matrix) has the *consecutive ones property*.

Let $G = (V,E)$ be an interval graph with dominant cliques $C1$, $C2$, ..., Cp. Then a v.d.c. matrix $M = [mij]$ is a (0,1) matrix of size $|V|$ x p. If vertex $i \in Cj$ then $mij = 1$, otherwise $mij = 0$. The matrix is said to have the *consecutive ones property*, if the ones in each row occur in consecutive positions.

3.1 Maximum-Difference Reduction Procedure

From above discussions, it is available to convert the problem of *Maximum-Difference Reduction* into that of *Minimum Clique Number Augmentation* to generate an optimum interval graph with minimum chromatic number. An efficient polynomial-time heuristic algorithm for the *Minimum Clique Number Augmentation* problem has been derived and is presented below.

Fig. 2(a) (from Fig. 13(a) of [7]) gives a partial routing and its corresponding initial vertex versus clique matrix and crossing graph are illustrated in Fig. 2(b).

From Fig. 2(a), we construct the wire segment set W and initially, let $V = \{ V1, V2, ..., V13 \}$ be a partition of W, where set V_i contains the wire segments of one net. The column (clique) in the vertex-clique matrix can be derived by scanning the position in the partial routing layout which is a crossing point formed by wire segments of different nets. It is denoted as a circle in Fig. 2(a). Obviously, as shown in the configuration of Fig. 2(a), thirteen layers are initially needed. Our goal is to find a minimal clique number augmentation on the crossing graph, i.e., to permute the positions of columns (cliques) and let the final vertex-clique matrix has the consecutive ones property.

Before the minimal augmentation process, '*vertex merging* and *clique merging* operations can be used as preprocessing on the vertex-clique matrix to reduce the number of rows and columns. Fig. 2(c) shows the result of row and column reductions. Note that the matrix in Fig. 2(c) has no consecutive ones property and its clique set $C = \{ C3, C4, C9, C10, C18, C21, C23 \}$. Now, our objective is to execute the permutation on set C such that a minimal clique number augmentation can be achieved. Under the configuration of Fig. 2(c), if we fill in the gray areas with ones then it has consecutive ones property and the clique number for each column can be counted. Blindly permuting the columns may not result in the reduction of maximum clique number. On the other hand, it may lead to the generation of loop.· Intuitively, decreasing the maximum clique number is helpful for achieving the minimal clique number augmentation. Now, we define some terminologies for vertex-clique matrix. An entry one is called a *boundary one* if it locates on the left most (or right most) position of its corresponding row. An entry one is called an *internal one* if it is not a boundary one. An *available interval* (AI) is a consecutive zeros interval which its immediate left and right neighboring columns are ones. An available interval is called a *boundary available interval* (BAI) if it is adjacent to a boundary one.

The clique number of column $C9$ in Fig. 2(c) may be decreased by moving some columns located on the left side of column $C9$ to the right side of column $C9$, or moving some columns located on the right side of column $C9$ to the left side of column $C9$. But which columns should be moved for contributing to the reduction of clique number. The answer is as follows. First, find all BAIs in the column $C9$ and list all boundary ones adjacent to BAIs. The columns that contain the above specified boundary ones will be identified and be called the *available columns*. Since moving of available columns may be beneficial to the reduction of clique number of column $C9$, the available columns will be moved to the immediate right (left) of column $C9$, if they located on the left (right) of column $C9$. For column $C9$, the available columns are $C4$ and $C18$. We may move $C4$ to the immediate right of $C9$ and move $C18$ to the immediate left of $C9$. We check that moving $C4$ is really helpful for the reduction of maximum clique number. And it does not work after $C8$ being moved.

Fig. 2(d) shows the results after $C4$ being moved. The maximum clique number is really decreased from five to four. The same process continues from the matrix in Fig. 2(d), but we fail to further reduce the maximum clique number. Therefore, our *Maximum-Difference Reduction* procedure is completed with a maximum difference $D = 13 - 4 = 9$ (thirteen is the initial number of layers).

If it is needed for further reducing the number of layers, vias must be introduced. In general, any arbitrary via is the candidate. In our procedure, we consider the vias that can break all cliques which have the maximum clique number. In Fig. 2(d), *C*9, *C*4, *C*10, *C*18 have the maximum clique number and contain the vertex sets of { *V*1, *V*2, *V*4, *V*5 }, { *V*1, *V*2, *V*4, *V*7 }, { *V*1, *V*2, *V*4, *V*7 }, { *V*1, *V*2, *V*4, *V*7 }, respectively. We classify the four vertex sets into two vertex sets of { *V*1, *V*2, *V*4, *V*5 } and { *V*1, *V*2, *V*4, *V*7 }.

All we have to do is to choose available vias as few as possible. Here, a greedy strategy is an efficient methodology. First, we choose a via that can break most maximum cliques than other vias. Repeat this operation until all maximum cliques are broken. Fig. 2(e) shows the matrix after introducing minimum via set. Now, a new configuration has been generated and the *Maximum-Difference Reduction* procedure will be called again. The final vertex-clique matrix is shown in Fig. 2(f) with $k = 3$ (k is the specified number of layers for original routing).

3.2 Backtracking Procedure

It asserts that, in a valid layer assignment, if two layers *li* and *lj* are occupied by wire segments of one net through a via *h*, then no layer between *li* and *lj* can be occupied by wire segments of another net through the same via position.

After the *Maximum-Difference Reduction* step, we have only the information on which wire segments can be placed in the same layer and how many layers are needed for layout. We have no idea about the order of layers, i.e., we do not know which layer can be placed on the top layer, the middle, or the bottom layer without violating the validity of layer assignment. In this subsection, a *Backtracking* procedure will be used to check whether or not a valid layer assignment is obtained. Due to the limited space, we will not discuss the *Backtracking* procedure here.

If the layer assignment is valid then the whole algorithm is completed, otherwise, a greedy criterion, similar to the one described in previous subsection, for adding extra vias into the minimum via set will be adopted and *Backtracking* procedure will also be applied again until no violation is incurred.

4. Experimental Results

The overall algorithm was coded in C language and implemented under the Unix 4.2 BSD on a Sun4 SPARC workstation. The proposed algorithms have been evaluated on some examples including examples 1, 3a, 3b, 3c, 4b, 5, and Deutsch's difficult example. Table 1 shows the performance on these examples. Our results are better than that of [8] and [9]. Also, the five-layer original layouts of ex3b and Deutsch's difficult example presented in [3] and [11] were used as test inputs to evaluate our multi-layer CVM algorithm. The results for elimination of vias on five-layer are very encouraging. On average, the number of vias is reduced by around 38% in Table 1.

5. Conclusion

We have proposed an efficient track permutation technique and a new approach for the multi-layer CVM problem, where two major procedures, *Maximum-Difference Reduction* and *Backtracking*, are used to solve the CVM problem. To reduce the complexity, the *Maximum-Difference Reduction* problem was converted to that of *Minimum Clique Number Augmentation*. Experimental results have demonstrated the efficiency of our approach for via minimization. Recently, with advances in manufacturing technology, it has been made possible to use multichip module (MCM) for interconnections. Our next project is to extend the multi-layer CVM algorithm to MCM where the performance factors are the major concerns.

References

[1] T. Yoshimura and E.S. Kuh, "Efficient algorithm for channel routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, no. 1, pp. 25-35, Jan. 1982.
[2] 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.
[3] R.J. Enbody and H.C. Du, "Near-optimal n-layer channel routing," in *Proc. 23rd Design Automation Conf*, pp. 708-714, June 1986.
[4] 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.
[5] X.M. Xiong and E.S. Kuh, "The constrained via minimization problem for PCB and VLSI design," in *Proc. 25th Design Automation Conf.*, pp. 573-578, June 1988.
[6] M.J. Ciesielski and E. Kinnen, "An optimum layer assignment for routing in IC's and PCB's," in *Proc. 18th Design Automation Conf.*, pp. 733-737, June 1981.
[7] K.C. Chang and H.C. Du, "Layer assignment problem for three-layer routing," *IEEE Trans. Computers*, vol. 37, no. 5, pp. 625-632, May 1988.
[8] K. Ahn and S. Sahni, "Constrained via minimization," *IEEE Trans, Computer-Aided Design*, vol. 12, no. 2, pp. 273-282, Feb. 1993.
[9] 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.
[10] T. Ohtsuki et al., "One-dimensional logic gate assignment and internal graphs," *IEEE Trans. Circuits and Systems*, vol. CAS-26, pp. 675-684, Sept. 1979.
[11] T.T. Ho, "New models for four-and five-layer channel routing," in *Proc. 29th Design Automation Conf.*, pp. 589-593, June 1992.
[12] T.H. Lai, *The topological order determination for three-layer channel routing problem*, Master Thesis, National Taiwan University, Taipei, Taiwan (1987).

Table 1 : Experimental results for multi-layer CVM problem

Ex.	No. of layers (k)	No. of nets	No. of original vias	No. of vias required from [9]	No. of vias required from [8]	Our proposed algorithm			
						No. of vias required	Percentage of via elimination (%)	CPU time (sec)	Ref.
ex 1	3	21	57	36	40	34	40.1	2.83	[2] Fig. 9
ex 3a	3	30	68	50	48	40	41.2	3.07	[2] Fig.10
ex3b	3	47	107	83	78	71	33.6	4.53	[2] Fig.11
ex 3c	3	54	125	99	96	82	34.4	6.92	[2] Fig.12
ex 4b	3	57	179	108	104	93	48.0	9.46	[2] Fig.13
ex 5	3	63	150	105	94	90	40.0	8.78	[2] Fig.14
diff1	3	72	290	214	206	191	34.1	16.68	[12] Fig.6-7
ex 3b	5	47	62	-	-	50	19.4	6.22	[11] Fig. 1
diff2	5	72	288	-	-	148	48.6	37.19	[3] Fig. 6

Note: diff1 is Deutsch's different example with dogleg-free mode.
diff2 is Deutsch's different example with five wiring layers.
* Both [8] and [9] only deal with the three-layer CVM problem.
* Percentage of via elimination = (No. of original vias - No. of vias required by our algorithm) / (No. of original vias).

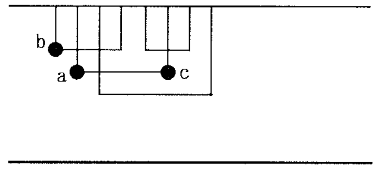
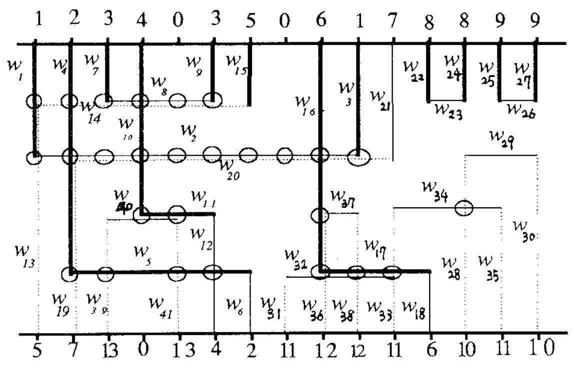


Fig. 1 Three categories of via candidates.
Point a: unremovable via
Point b: absolutely removable via
Point c: possibly removable via


1 2 3 4 0 3 5 0 6 1 7 8 8 9 9
5 7 13 0 13 4 2 11 12 12 11 6 10 11 10
w1 w4 w7 w8 w9 w15 w14 w10 w2 w16 w3 w21 w22 w24 w25 w27 w23 w26 w29 w20 w40 w11 w37 w34 w12 w30 w13 w5 w32 w17 w28 w35 w19 w39 w41 w6 w31 w36 w38 w33 w18

▬ : layer 1
— : layer 2
⋯ : layer 3
w_i : line segment
○ : cross point

Fig. 2(a) A partial routing problem instance.

V \ C	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}	c_{21}	c_{22}	c_{23}	c_{24}	c_{25}	c_{26}	c_{27}
V_1	1	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	1	1	1	0	0	1	0	0	0
V_2	0	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
V_3	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
V_4	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
V_5	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
V_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0
V_7	0	0	0	1	1	0	1	0	0	1	0	0	1	0	0	0	1	0	1	1	1	0	0	1	0	0	0
V_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V_9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
V_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
V_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0
V_{13}	0	0	0	0	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Note : V = vertex set ; C = clique set
$V1$ = { $w1, w2, w3$ } $V2$ = { $w4, w5, w6$ } $V3$ = { $w7, w8, w9$ }
$V4$ = { $w10, w11, w12$ } $V5$ = { $w13, w14, w15$ } $V6$ = { $w16, w17, w18$ }
$V7$ = { $w19, w20, w21$ } $V8$ = { $w22, w23, w24$ } $V9$ = { $w25, w26, w27$ }
$V10$ = { $w28, w29, w30$ } $V11$ = { $w31, w32, w33, w34, w35$ }
$V12$ = { $w36, w37, w38$ } $V13$ = { $w39, w40, w41$ }

Fig. 2(b) Initial vertex-clique matrix.

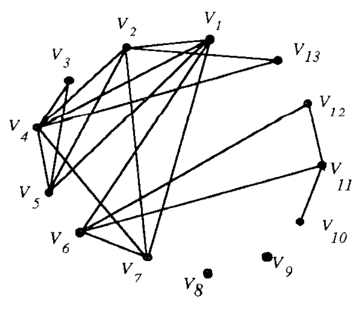


Fig. 2(b) Initial crossing graph. (continued)

V \ C	c_3	c_4	c_9	c_{10}	c_{18}	c_{21}	c_{23}
V_1	0	1	1	1	0	1	0
V_2	1	1	0	0	1	0	0
V_4	0	0	1	1	1	0	0
V_5	1	0	1	0	0	0	0
V_6	0	0	0	0	0	1	1
V_7	0	1	0	1	0	1	0
V_{11}	0	0	0	0	0	0	1
V_{12}	0	0	0	0	0	0	1
clique number	2	4	5	4	4	3	3

Note: ▨ means a BAI.

Fig. 2(c) Vertex-clique matrix with clique number and BAI illustration.

V \ C	c_3	c_9	c_4	c_{10}	c_{18}	c_{21}	c_{23}
V_1	0	1	1	1	0	1	0
V_2	1	0	1	0	1	0	0
V_4	0	1	0	1	1	0	0
V_5	1	1	0	0	0	0	0
V_6	0	0	0	0	0	1	1
V_7	0	0	1	1	0	1	0
V_{11}	0	0	0	0	0	0	1
V_{12}	0	0	0	0	0	0	1
clique number	2	4	4	4	4	3	3

Fig. 2(d) The resultant matrix after executing *Maximum-Difference Reduction* procedure. The maximum clique number is 4.

V \ C	c_3	c_{9-1}	c_4	c_{10-1}	c_{18-1}	c_{21}	c_{23}	c_{28}
V_1	0	1	1	1	0	1	0	0
V_2	1	0	1	0	1	0	0	0
V_{4a}	0	0	0	0	1	0	0	1
V_{4b}	0	1	0	1	0	0	0	1
V_5	1	1	0	0	0	0	0	0
V_6	0	0	0	0	0	1	1	0
V_7	0	0	1	1	0	1	0	0
V_{11}	0	0	0	0	0	0	1	0
V_{12}	0	0	0	0	0	0	1	0

Fig. 2(e) Vertex-clique matrix after introducing minimum via set.

V \ C	c_{9-1}	c_{21}	c_{23}
V_1	1	1	0
V_2	1	0	0
V_6	0	1	1
V_7	1	1	0
V_{11}	0	0	1
V_{12}	0	0	1
clique number	3	3	3

Note : $V1 = V1 \cup V4a$ $V2 = V2 \cup V4b$ $V7 = V5 \cup V7$

Fig. 2(f) Final vertex-clique matrix with k = 3.