Minimizing the Number of Switchboxes for Region Definition and Ordering Assignment

Jin-Tai Yan and Pei-Yung Hsiao

Abstract—**For a building block placement, the routing space will be fully defined into channels and switchboxes because the definition of switchboxes releases all the cycles in a channel precedence graph and further yields a safe routing ordering. However, switchbox routing is more difficult than channel routing. Due to the requirements in the routing phase, it is important for us to minimize the number of switchboxes in the definition of channels and switchboxes. In this paper, a region definition and ordering assignment (RDAOA) algorithm on minimizing the number of switchboxes is proposed. First, the routing space is modeled as a floorplan graph. According to the routing precedence in one "*T*" type junction, the graph is further transformed into a channel precedence graph. Hence, the problem of minimizing the number of switchboxes in region definition will correspond to the minimum feedback vertex set (MFVS) problem in the channel precedence graph. Second, based on the cyclic properties in a channel precedence graph, the MFVS problem is solved by the minimal-cycle phase and the long-cycle phase. All the minimal cycles and most of the long cycles are broken in the minimal-cycle phase, and the remaining long cycles are further broken in the long-cycle phase. As a result, fewer switchboxes are defined, and these channels and switchboxes are assigned to guarantee a safe routing ordering. The time complexity of the algorithm is proved to be in $O(n)$ time, where n is the number of line segments in a given floorplan graph. Finally, several examples have been tested on the proposed algorithm and other published algorithms, and the experimental results show that our algorithm defines fewer switchboxes than other algorithms.**

I. Introduction

IN VLSI layout design, most of the layout problems [1]–[4] have been proved to be NP-complete, and heuristic approaches are usually applied to reduce the complexity of these problems. Thus, many efficient algorithms have been proposed to obtain near-optimal solutions for these layout problems in a reasonable time complexity. In general, these problems are classified into the following main subproblems: placement, global routing, region definition and ordering, detailed routing, and compaction. The overview of these subproblems is shown in Fig. 1.

Generally speaking, the routing space of a building block placement is modeled as a floorplan graph. If the graph is fully decomposed into channels and there exists a safe routing ordering, it will be clear that the width of any channel can be expanded or contracted without destroying previous routed channels. Furthermore, according to the assigned routing ordering, these defined channels can be successfully routed by a channel router in the routing phase. Such a building block placement will be called a slicing placement. However, it is not certain that each building block placement is a slicing placement.

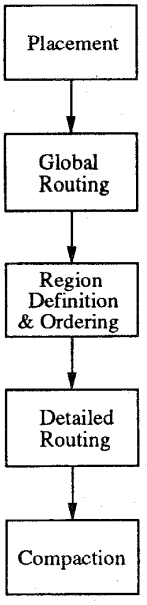


Fig. 1. Overview of the layout problem.

Consider a given building block placement shown in Fig. 2 and suppose that the regions A, B, C, and D are all to be defined as channels. As the terminals in region B are not fixed, region A must be routed before region B. For the same reason, region B must be routed before region C, region C must be routed before region D, and region D must be routed before region A. The iterative phenomenon of the precedence relations of regions A, B, C, and D will construct a channel precedence cycle. The cyclic constraint is named a cyclic channel precedence constraint (CCPC) [5]. Clearly, if a building block placement has some CCPC's, the routing space will be not defined into only channels and the placement will be a nonslicing placement. For a nonslicing placement, the definition of switchboxes or L-shaped channels is generally introduced to release these cyclic channel constraints and guarantees a safe routing ordering in the routing phase.

For region definition and ordering assignment, the problem of breaking CCPC's in a building block placement has been

Manuscript received October 11, 1993; revised April 26, 1994. This work was supported in part by the National Science Council of the Republic of China under Contract NSC 84-2213-E009-017. This paper was recommended by Associate Editor M. Sarrafzadeh.

The authors are with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

Publisher Item Identifier S 0278-0070(96)01345-0.

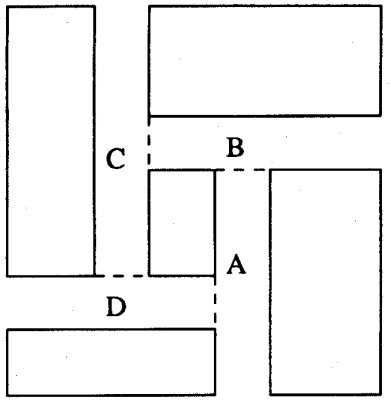


Fig. 2. Cyclic channel precedence constraint.

extensively studied for many years. Various approaches have been proposed and discussed in previous published papers [6]–[13]. First, Otten [6] restricts the acceptable placements of slicing structures in order to avoid these CCPC's. Hence, it is certain that no CCPC is in a building block placement. On the other hand, a building block placement may be modified to avoid these CCPC's by converting a nonslicing structure into a slicing structure. For example, Chiba [7] perturbs the placement to convert a nonslicing structure into a slicing structure, and Kimura [8] shrinks the placed modules to perform the same conversion. However, instead of breaking the CCPC's in region definition and ordering assignment, these restricted and modified techniques only change the nonslicing structure into the slicing structure for a building block placement. In recent years, the problem of breaking CCPC's have been really solved by introducing the definition of switchboxes [1], [9]–[11] or by combining straight channels into L-shaped channels [5], [12]–[13]. In addition, some other approaches [14]–[16] have been proposed and are well documented in the literature. In general, for a building block placement, the number of switchboxes is fewer than the number of L-shaped channels. Therefore, the definition of switchboxes is always applied to release all the CCPC's in a building block placement.

Since the definition of switchboxes releases all the cycles in a channel precedence graph and further yields a safe routing ordering, the routing space will be fully defined into channels and switchboxes. Consider a CCPC, we can refer again to the placement shown in Fig. 2. Firstly, the width of region A can be estimated by the position of pins and local connection in region A, and the positions of the floating pins between region A and B are fixed by the crossing distribution of one "T" type junction [17]. Then regions B, C, and D can now be sequentially routed as channels in that order. Finally, region A is routed as a switchbox with a fixed width and fixed terminals on three sides. Hence, the CCPC will be released by introducing the definition of a switchbox. For the same reason, all the CCPC's in a building block placement will be released by introducing the definition of some switchboxes. However, from the viewpoint of detailed routing, the width of a switchbox in region definition is fixed and switchbox routing is not guaranteed. In addition, it is well known that switchbox routing [18] is more difficult than channel routing. Due to the consideration of routing success in the routing phase, it is important for us to minimize the number of switchboxes in the definition of channels and switchboxes.

Although it is well known that the definition of switchboxes is applied to release all of the CCPC's, only few algorithms have been proposed to minimize the number of switchboxes in region definition and ordering assignment. In general, the routing space is modeled as a floorplan graph. According to the routing precedence in a T-type junction, the graph is further transformed into a channel precedence graph. Hence, the problem of minimizing the number of switchboxes in region definition will correspond to a minimum feedback vertex set (MFVS) problem in the channel precedence graph. However, the MFVS problem is NP-complete for a channel precedence graph. To our knowledge, only three algorithms are proposed to minimize the number of switchboxes in region definition and ordering assignment. First, a greedy approach [10] generates the definition of a switchbox whenever a CCPC is detected, and always generates the definition of unnecessary switchboxes for a complicated building block placement. Furthermore, Sur-Kolay and Bhattacharya [19] propose a heuristic algorithm based on a clique cover-table approach and a greedy approach to discuss the cycle structure of channel graphs in nonslicible floorplans. In this approach, the definition of switchboxes is divided into the minimal-cycle phase and the long-cycle phase. In the minimal-cycle phase, an $O(n^2)$ algorithm is proposed to break all the minimal cycle. Then, all the long cycles are broken by a greedy algorithm in the long-cycle phase. However, several unnecessary switchboxes are defined for a complicated building block placement. Finally, Cai and Wong [20] propose a graph-based algorithm to find a feedback vertex set in a channel precedence graph by the transformation between a reduced digraph and an intersection graph, and obtain significant improvement on the number of switchboxes. However, the time complexity of the algorithm is in $O(n^3)$ time.

In this paper, a region definition and ordering assignment (RDAOA) algorithm on minimizing the number of switchboxes is proposed. Based on the cyclic properties in a channel precedence graph, the MFVS problem is solved by the minimal-cycle phase and the long-cycle phase. All the minimal cycles and most of the long cycles are broken in the minimal-cycle phase, and the remaining long cycles are further broken in the long-cycle phase. As a result, fewer switchboxes are defined, and these channels and switchboxes are assigned to guarantee a safe routing ordering. The time complexity of the algorithm is proved to be in $O(n)$ time, where n is the number of line segments in a given floorplan graph. Finally, several examples have been tested on the proposed algorithm and other published algorithms, and the experimental results show that our algorithm defines fewer switchboxes than other algorithms.

The rest parts of this paper are organized as follows. Section II contains the preliminaries and the available definitions for minimizing the number of switchboxes in the RDAOA problem. In Section III, the graph transformations from a floorplan graph to a k-cycle intersection graph are discussed

for the minimal-cycle phase. Section IV provides the details of the proposed RDAOA algorithm. In Section V, experimental results in the tested examples are shown and compared. Finally, the conclusion for the proposed algorithm is presented in Section VI.

II. Preliminaries and Definitions

In building block design style, all the building blocks are located on fixed positions by a macro cell placement algorithm. The space between any pair of adjacent blocks is always applied to route all of the nets. In order to successfully route these nets in detailed routing, the routing space must be partitioned into rectilinear regions after global routing, and these regions to be routed must be further assigned in a safe ordering. In this paper, we define the routing space into channels and switchboxes to assign a safe ordering. Before we describe the proposed approach in detail for minimizing the number of switchboxes, some important assumptions and definitions will be modeled as follows:

First, assume that all the building blocks are rectangular modules. Therefore, the routing space between any pair of adjacent blocks will be represented by one horizontal or vertical line segment, and the intersection points of the line segments will represent the junction routing areas. From the geometrical relation between these line segments and intersection points, these line segments and intersection points will construct a floorplan graph for a given building block placement by a transformation algorithm [7]. Furthermore, the space between any block and any boundary can be ignored in the RDAOA problem. Thus, a modified floorplan graph without the boundary line segments is generated for the RDAOA problem. The modified floorplan graph is regarded as the floorplan graph throughout the rest of this paper unless explicitly stated otherwise. In Fig. 3, a building block placement, its related floorplan graph and its modified floorplan graph are shown.

Second, as a floorplan graph has been generated from a building block placement, it is necessary for the RDAOA problem to solve two conditions of empty rooms and "+" type junctions [5] in a floorplan graph. For empty rooms, whenever there exist empty rooms in a floorplan graph, they will be removed by recursively removing one wall segment from each empty room until none are present in the floorplan graph [21]. Thus, it may be assumed that all of the empty rooms in the floorplan graph have been removed in the RDAOA problem. On the other hand, for the "+" type junctions, several algorithms [5], [7], [22], [23] are proposed to split one "+" type junction into two "T" type junctions. Hence, all the "+" type junctions are successfully split by creating some "T" type junctions. It may be assumed that each "+" type junction in the floorplan graph has been separated into two "T" type junctions by a splitting algorithm in the RDAOA problem.

As mentioned above, there is no empty room and "+" type junction in a floorplan graph, and the floorplan graph only consists of horizontal and vertical line segments. In general, each horizontal or vertical line segment in a floorplan graph will be represented as one channel. As two rows of terminals

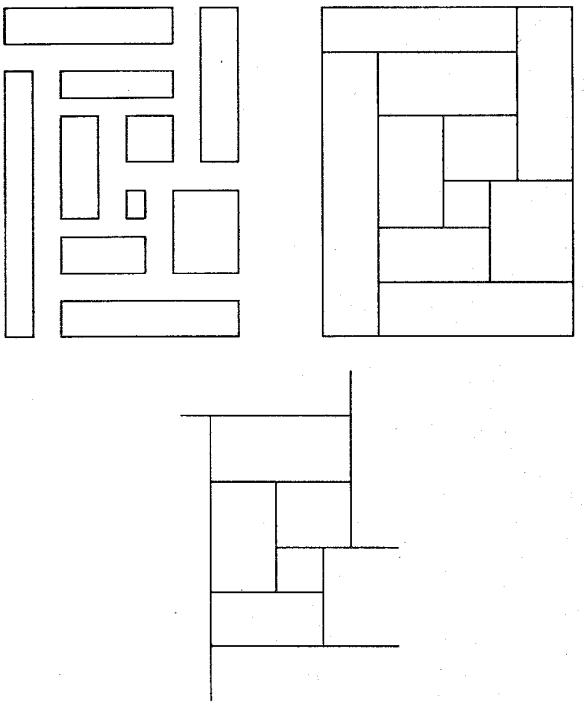


Fig. 3. A building block placement and its floorplan graph.

located on a channel are fixed, the channel will be routed successfully by a channel router. Hence, for one "T" type junction in a floorplan graph, the base channel of the "T" type junction must be routed before the top channel of that according to the specification of channel routing. Based on the precedence relation in one "T" type junction, we define a channel precedence graph from a floorplan graph as follows.

Definition 1: For a given floorplan graph, F, a channel precedence graph $G(F) = (V, A)$ is a directed graph defined as follows: Each line segment in the floorplan graph is represented as one vertex of V. Each directed edge (u, v) is in A if and only if the two line segments corresponding to u and v form one "T" type junction in the floorplan graph, and the one corresponding to u is the base of the "T" type junction.

In Fig. 4(a), the floorplan graph for a building block placement is shown, where v_i denotes the ith vertical line segment from the left, and the h_j denotes the jth horizontal line segment from the top. The channel precedence graph G of Fig. 4(a) is illustrated in Fig. 4(b), where V is $\{v_1, v_2, v_3, v_4, v_5, v_6, h_1, h_2, h_3, h_4, h_5\}$ and A is $\{(v_1, h_1), (v_2, h_2), (v_3, h_2), (v_3, h_4), (v_4, h_3), (v_5, h_3), (v_6, h_5), (h_1, v_2), (h_2, v_1), (h_2, v_5), (h_3, v_3), (h_3, v_6), (h_4, v_1), (h_4, v_4), (h_5, v_4)\}$.

Due to the geometrical properties in a floorplan graph, the generated channel precedence graph has several available graphical properties to help solve the RDAOA problem of minimizing the number of switchboxes.

Lemma 1: Let $G(V, A)$ be a channel precedence graph, G has the following graphical properties:

1) planar,
2) bipartite,
3) maximum out-degree is equal to 2,

4) if $|V| \geq 4$, then there are at least four vertices whose out-degree is not more than 1,
5) if $|V| \geq 4$, then the number of edges $|A|$ is at most $2n - 4$, where n is the number of vertices.

Proof: The proofs of statement (1)–(4) are the same as Lemma 1 in the Cai and Wong's paper [20] except for the correction to statement (4). Furthermore, by statement (3), since the out-degree of each vertex is not more than 2, the total number of edges $|A|$ in a channel precedence graph is not more than $2n$, where n is the number of vertices. By statement (4), it demonstrates that there are at least four vertices whose out-degree is not more than 1. Hence the number of edges in a channel precedence graph is at most $2n - 4$, where n is the number of vertices in a channel precedence graph. Q.E.D.

In general, each horizontal or vertical line segment in a floorplan graph will be represented as one channel. If the generate channel precedence graph is acyclic, the routing space will be fully decomposed into horizontal and vertical channels, and the safe ordering of these channels will be arranged by a topological sorting. Clearly, the RDAOA problem can be solved by a topological sorting, and no switchbox is defined in the building block placement. On the other hand, if the channel precedence graph is cyclic, the RDAOA problem will not be solved by a topological sorting and some cyclic channel constraints will be yielded in the building block placement. Therefore, the definition of switchboxes will be introduced to release all of the cyclic channel constraints in the building block placement. It is well known that switchbox routing is more difficult than channel routing. Due to the requirements in the routing phase, it is necessary for us to minimize the number of switchboxes in detailed routing.

From the viewpoint of a channel precedence graph, if a channel precedence graph is cyclic, a set of vertices will be removed from the graph to generate a new acyclic channel precedence graph. Clearly, the set of removed vertices in a channel precedence graph will correspond to the definition of switchboxes in a mapped building block placement. As the set of removed vertices is defined as switchboxes, all the CCPC's in the mapped building block placement will be fully released by the definition of switchboxes. Furthermore, the vertices in the remaining acyclic channel precedence graph will be only defined as channels. Finally, a safe routing ordering of these channels will be assigned by a topological sorting, and the ordering of the defined switchboxes will be further assigned randomly after all of the defined channels.

Therefore, the RDAOA problem of minimizing the number of switchboxes in a building block placement will correspond to the problem of minimizing the number of the set of removed vertices in a generated channel precedence graph. For a channel precedence graph, the problem of minimizing the number of the set of removed vertices is defined as the MFVS problem.

Definition 2: A feedback vertex set S of a directed graph $G = (V, A)$ is a subset of V. The vertex removal from G results in an acyclic directed graph. A minimum feedback vertex set $S_{\min}$ is a feedback vertex set with minimum cardinality.

It is well known that the minimum feedback vertex set (MFVS) problem [24] remains NP-hard for a channel precedence graph. Since the MFVS problem is to remove a minimal set of vertices to break all the cycles in a channel precedence graph, it will be important for the development of a heuristic algorithm to study the characterization of a channel precedence cycle in a channel precedence graph. According to the construction of a floorplan graph, some graphical properties of a channel precedence cycle are further described.

Lemma 2: For a channel precedence graph, three graphical properties of a channel precedence cycle are as follows:

1) the length is even,
2) the length of a minimal cycle [25] is 4,
3) all vertices are in at most four minimal cycles at the same time.

Proof: It is clear that a precedence relation in a channel precedence graph represents one "T" type junction in a floorplan graph. If there exists one cycle in a channel precedence graph, the number of vertices representing vertical line segments will be equal to that representing horizontal line segments in the cycle. Hence, the number of vertices in the cycle is even and the length of the cycle is even. Due to the geometrical position in a floorplan graph, a minimal cycle in a channel precedence graph is constructed by two vertices representing vertical line segments and two vertices representing horizontal line segments. The length of a minimal length is four. As mentioned above, the maximum out-degree of any vertex in a channel precedence graph is 2, in other words, each line segment in a floorplan graph is used as the base of at most two "T" type junctions. In addition, one "T" type junction is in at most two minimal cycles. Thus, each vertex in a channel precedence graph is in at most four minimal cycles at the same time. Q.E.D.

In general, the cycles in a channel precedence graph are divided into the minimal cycles whose length is four and the long cycles whose length is more than four. According to the physical structure of a floorplan graph, it may be assumed that most of the cycles in a channel precedence graph are minimal cycles. Therefore, it is important for the MFVS problem to break all the minimal cycles in a channel precedence graph. The MFVS problem in a channel precedence graph will be solved by the minimal-cycle phase and the long-cycle phase [19]. In the minimal-cycle phase, all of the minimal cycles and most of long cycles in the channel precedence graph are broken by removing a minimal of vertices to be defined as switchboxes. Finally, the remaining long cycles are broken by an iterative greedy approach in the long-cycle phase.

Since the minimal-cycle phase is important for the MFVS problem, some available graphs need to be further defined for the minimal-cycle phase. First, a weighted channel precedence graph is established to indicate cyclic information by assigning weight onto the vertices in a channel precedence graph.

Definition 3: A weighted channel precedence graph $G' = (V', A')$ is a channel precedence graph with weight assigned to V'. For each vertex u in V', a weight value $w(u)$ is assigned onto the vertex u, where $w(u)$ is the number of the minimal cycles in which u is contained.

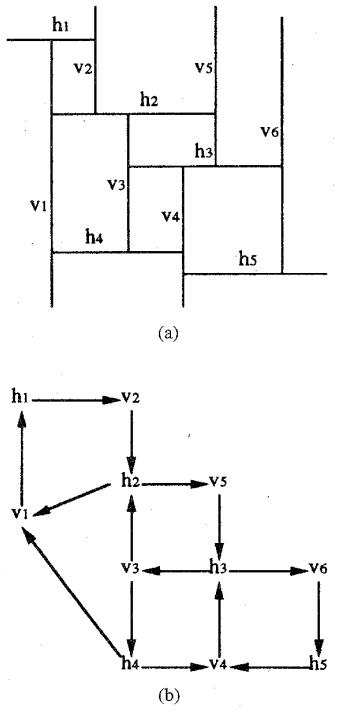


Fig. 4. (a) A given floorplan graph. (b) The channel precedence graph of (a).

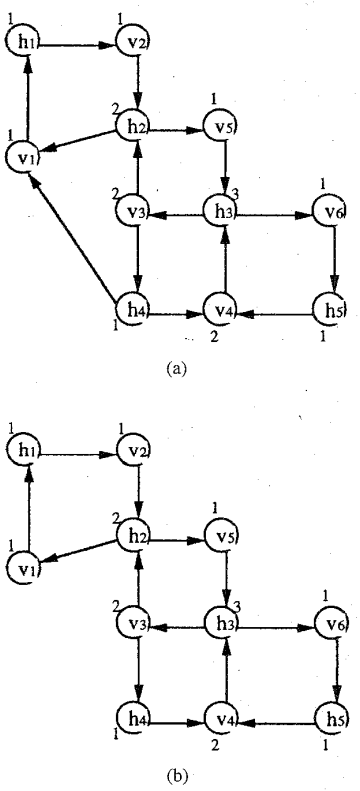


Fig. 5. (a) The weighted channel precedence graph of Fig. 4(a). (b) The cyclic channel precedence graph of Fig. 4(a).

In order to define a set of vertices in a channel precedence graph as switchboxes in the minimal-cycle phase, some vertices in a weighted channel precedence graph are of no use. Furthermore, a cyclic channel precedence graph is further established to simplify the size of the weight channel precedence graph by removing the vertices which are not in any minimal cycle.

Definition 4: A cyclic channel precedence graph $G^* = (V^*, A^*)$ is a subgraph of a weighted channel precedence graph by removing the vertices which are not in any minimal cycle, where V^* is a subset of V, and A^* is a subset of A.

Fig. 5(a) shows a weighted channel precedence graph of Fig. 4(b), and a cyclic channel precedence graph of Fig. 5(a) is shown in Fig. 5(b). In Fig. 5(b), four minimal cycles $\{h_1, v_2, h_2, v_1\}\{h_2, v_5, h_3, v_3\}, \{h_3, v_3, h_4, v_4\}$ and $\{h_3, v_6, h_5, v_4\}$ are in the cyclic channel precedence graph.

Furthermore, by Lemma 2, all vertices in a channel precedence graph are in at most four minimal cycles at the same time. For the same reason, all vertices in a cyclic channel precedence graph are in at least one minimal cycle and in at most four minimal cycles at the same time. Therefore, one vertex in a cyclic channel precedence graph will be contained in one minimal cycle, two minimal cycles, three minimal cycles, or four minimal cycles simultaneously. A k-cycle pattern in a cyclic channel precedence graph will be defined to optimally find a minimal set of vertices to be defined as switchboxes in the minimal-cycle phase.

Definition 5: A k-cycle pattern is a connected subgraph of a cyclic channel precedence graph and contains k connected minimal cycles. The weight of at least one common vertex in the k-cycle pattern is exactly k.

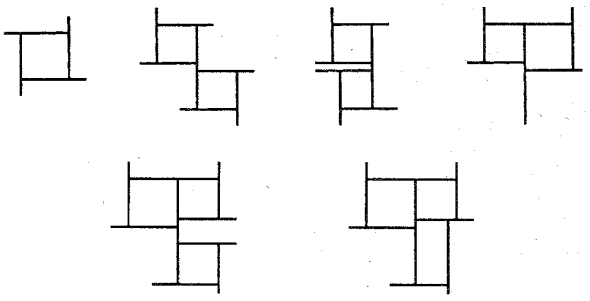


Fig. 6. All the possible k-cycle patterns.

Clearly, only 1-cycle, 2-cycle, 3-cycle, and 4-cycle patterns will be found in a cyclic channel precedence graph. All possible k-cycle patterns are shown in Fig. 6. The cyclic channel precedence graph in Fig. 5(b), it covers one 2-cycle pattern and one 3-cycle pattern. The 2-cycle pattern is formed by two minimal cycles $\{h_1, v_2, h_2, v_1\}$ and $\{h_2, v_5, h_3, v_3\}$. The other 3-cycle pattern is formed by three minimal cycles $\{h_2, v_5, h_3, v_3\}, \{h_3, v_3, h_4, v_4\}$ and $\{h_3, v_6, h_5, v_4\}$.

According to the definition of a k-cycle pattern, all of the minimal cycles in a k-cycle pattern will be broken by removing one common vertex. Hence, a k-cycle pattern will be viewed as a processed unit in the minimal-cycle phase. For any pair of k-cycle patterns in a cyclic channel precedence graph, the

intersection relation will be set by sharing at least one minimal cycle. Here, a k-cycle intersection graph is further established by modeling the intersection relation between any pair of adjacent k-cycle patterns.

Definition 6: For a cyclic channel precedence graph, a k-cycle intersection graph $G_{k\text{-cycle}} = (V_{k\text{-cycle}}, E_{k\text{-cycle}})$ is an undirected graph, where $V_{k\text{-cycle}}$ represents all of the k-cycle patterns in the cyclic channel precedence graph, and $E_{k\text{-cycle}}$ represents the intersection relations of all pairs of adjacent k-cycle patterns.

Lemma 3: For a floorplan graph, the number of defined switchboxes in the minimal-cycle phase is at most the number of vertices in its k-cycle intersection graph.

Proof: For a k-cycle intersection graph, each vertex represents a k-cycle pattern. As mentioned above, all of the minimal cycles in a k-cycle pattern are broken by removing a common vertex in the k-cycle pattern. If there exists intersection relations in all the k-cycle patterns, it is possible that all the minimal cycles in a k-cycle pattern are covered by the other k-cycle patterns. In fact, all the minimal cycles in a k-cycle pattern will be broken by removing at most one vertex. Since the removed vertex is defined as a switchbox, the number of defined switchboxes in the minimal-cycle phase is at most the number of vertices in its k-cycle intersection graph. Q.E.D.

For a k-cycle intersection graph, each vertex represents a k-cycle pattern. Hence, the vertex set is further divided into the independent set and the dependent set. For an independent vertex, the mapped k-cycle pattern will be independently broken in the minimal-cycle phase. On the other hand, for a dependent vertex, the mapped k-cycle pattern will not be independently broken in the minimal-cycle phase. Here, the definition of an independent vertex in a k-cycle intersection graph will be applied to define necessary switchboxes for its related building block placement. Furthermore, according to the construction of a k-cycle intersection graph, some graphical properties are further described.

Definition 7: For one vertex in a k-cycle intersection graph, the vertex represents a k-cycle pattern. If there exists any minimal cycle only in the k-cycle pattern, the vertex will be defined as an independent vertex. On the other hand, if all the minimal cycles in the k-cycle pattern are fully covered by the other k-cycle patterns, the vertex will be defined as a dependent vertex.

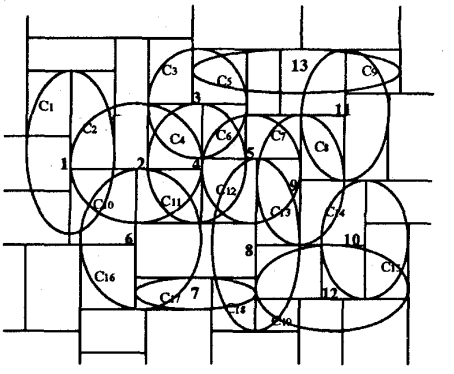
Lemma 4: For a k-cycle intersection graph, some graphical properties are as follows:

1) planar,
2) the intersection of two k-cycle patterns has at most two minimal cycles,
3) the degree of at least one vertex is less than 3,
4) one k-cycle pattern is not fully covered in the other k-cycle pattern,
5) any vertex with degree 1 is an independent vertex.

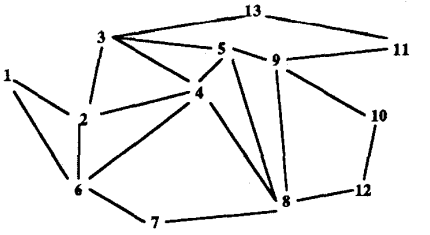
Proof: Because a cyclic channel precedence graph is planar and the intersection relation between two k-cycle patterns belongs to a neighbor relation, a k-cycle intersection graph will be planar. As mentioned above, at most two minimal cycles are located on the same "T" type junction and the intersection relation in a k-cycle intersection graph always appears on one "T" type junction. Thus, the intersection of two k-cycle patterns has at most two minimal cycles. By Lemma 2, one k-cycle pattern has at most four minimal cycles. If one 1-cycle or 2-cycle pattern is in a k-cycle intersection graph, the degree of the vertex representing the pattern will be less than 3. On the other hand, if not, only 3-cycle and 4-cycles patterns will be in a k-cycle intersection graph. According to the distribution of the minimal cycles in a floorplan graph and the construction of one 3-cycle or 4-cycle pattern, the degree of the vertex representing one corner pattern will be less than three. Hence, the degree of at least one vertex is less than three.

From the definition of a k-cycle pattern, there exists no covering relation between two k-cycle patterns, one k-cycle pattern will not be fully covered in the other k-cycle pattern. Finally, since one k-cycle pattern only intersects with one other k-cycle pattern and one k-cycle pattern is not fully covered in the other k-cycle pattern, any vertex with degree 1 in a k-cycle intersection graph is an independent vertex. Q.E.D.

In Fig. 7(a), the floorplan graph of a building block placement is illustrated and all the k-cycle patterns are labeled by circles. Furthermore, Fig. 7(b) shows and explains the related k-cycle intersection graph for the floorplan graph. For the k-cycle intersection graph, it is clear that vertices 1, 3, and 6 are independent and the other vertices are dependent.



(a)



Circle 1 => { C1, C2, C10 }
Circle 2 => { C2, C4, C10, C11 }
Circle 3 => { C3, C4, C5, C6 }
Circle 4 => { C4, C6, C11, C12 }
Circle 5 => { C6, C7, C12, C13 }
Circle 6 => { C10, C11, C16, C17 }
Circle 7 => { C17, C18 }
Circle 8 => { C12, C13, C18, C19 }
Circle 9 => { C7, C8, C13, C14 }
Circle 10 => { C14, C15 }
Circle 11 => { C8, C9 }
Circle 12 => { C15, C19 }
Circle 13 => { C5, C9 }

(b)

Fig. 7. (a) A given floorplan graph. (b) The k-cycle intersection graph of (a).

III. Graph Transformation

In order to minimize the number of switchboxes in the minimal-cycle phase, it is important to transform a floorplan graph into a k-cycle intersection graph. According to the construction of these graphs, the transformation can be divided into three partial transformations: the transformation between a floorplan graph and a channel precedence graph (FG-CPG transformation), the transformation between a channel precedence graph and a cyclic channel precedence graph (CPG-CCPG transformation) and the transformation between a cyclic channel precedence graph and a k-cycle intersection graph (CCPG-KIG transformation).

First, in the FG-CPG transformation, each line segment in a floorplan graph represents one vertex in a channel precedence graph and each "T" type junction in a floorplan graph represents one edge in a channel precedence graph. By Lemma 1, for a channel precedence graph $G(V, A), O(|V|) = O(|A|) = O(n)$, where n is the number of line segments in a floorplan graph. Hence the FG-CPG transformation will take $O(n)$ time by traversing all of the line segments in a floorplan graph by a transformation algorithm [7].

Then, in the CPG-CCPG transformation, it is necessary for the transformation to detect all of the minimal cycles in a channel precedence graph. For any minimal cycle, there are exactly four vertices to be connected by four precedence relations. Therefore, the detection of all of the minimal cycles will be solved by the Detect-All-Minimal-Cycle (DAMC) algorithm. The DAMC algorithm for a channel precedence graph is modified by a Depth-First-Search (DFS) algorithm [26]–[27]. As each vertex in the channel precedence graph is visited by the DAMC algorithm, all of the minimal cycles from the vertex will be detected by an exhaustive search. After visiting all of the vertices in a channel precedence graph, all the minimal cycles will be fully detected by the DAMC algorithm.

Theorem 1: The DAMC algorithm will detect all the minimal cycles in a channel precedence graph. If an adjacent list is applied to store the channel precedence graph, the time complexity of the DAMC algorithm is in $O(n)$ time, where n is the number of vertices in the channel precedence graph.

Proof: Basically, the DAMC algorithm is a DFS algorithm except that an exhaustive search is applied to detect all of the minimal cycles in each vertex visiting. As a result, the DAMC algorithm will detect all of the minimal cycles in the channel precedence graph. Because the length of a minimal cycle is four and the out-degree of each vertex is at most two, the detection of all the minimal cycles in one vertex visiting will take 16 (2^4) searches in an exhaustive search. Clearly, the time complexity of an exhaustive search in one vertex visiting is in $O(1)$ time. In addition, if an adjacent list is applied to store the channel precedence graph, the time complexity of the DFS algorithm will be in $O(|V| + |A|)$ time. Hence, the time complexity of the DAMC algorithm will be in $O(|V| + |A|)$ time. By Lemma 1, the number of edges $|A|$ is at most $2|V|-4$ in a channel precedence graph, that is, $O(|A|) = O(|V|)$. The time complexity of the DAMC algorithm is in $O(n)$ time, where n is the number of vertices in a channel precedence graph. Q.E.D.

Since all the minimal cycles have been detected by the DAMC algorithm, the CPG-CCPG transformation will be achieved by the CPG-CCPG algorithm to transform a channel precedence graph into a cyclic channel precedence graph. Basically, the CPG-CCPG algorithm is a DAMC algorithm with the exception of the weight assignment and the deletion of unnecessary vertices.

Theorem 2: The CPG-CCPG algorithm will transform a channel precedence graph into a cyclic channel precedence graph. The time complexity of the CPG-CCPG algorithm is in $O(n)$ time, where n is the number of vertices in a channel precedence graph.

Proof: By the definitions of CPG and CCPG, the CPG-CCPG algorithm is a DAMC algorithm except for the weight assignment and the deletion of unnecessary vertices. Before the CPG-CCPG algorithm, the weights of all vertices in a channel precedence graph are assigned as 0. For the CPG-CCPG algorithm, the detection of all of the minimal cycle is implemented by an exhaustive search in one vertex visiting. As a minimal cycle is detected, the weights of all vertices in the minimal cycle are increased by 1, and the vertices of the minimal cycle are marked. After traversing all the vertices in a channel precedence graph, all unmarked vertices will be removed from the channel precedence graph, and a cyclic channel precedence graph will be generated. Basically, the increment of the weights and the process of marking vertices do not effect the time complexity in the detection of all the minimal cycles in the algorithm. The deletion of unmarked vertices only takes $O(|V|) = O(n)$ time. Thus, the time complexity of the CPG-CCPG algorithm is in $O(n)$ time, where n is the number of vertices in a channel precedence graph. Q.E.D.

Finally, the CCPG-KIG transformation will be achieved by a CCPG-KIG algorithm to transform a cyclic channel precedence graph into a k-cycle intersection graph. The CCPG-KIG algorithm is implemented by visiting all of the minimal cycles in a cyclic channel precedence graph. In visiting a minimal cycle, the vertex with maximal weight in the minimal cycle is expressed as the representative of a k-cycle pattern containing the minimal cycle. After visiting all the minimal cycles, the representatives of the related k-cycle patterns will be transformed into the mapping vertices in a k-cycle intersection graph. Furthermore, if the representatives of any two k-cycle patterns are located on the same minimal cycle, an intersection relation between the two k-cycle patterns will be established. Besides that, if any minimal cycle on a k-cycle pattern has at least two vertices with a weight of 1, the k-cycle pattern will be defined as an independent vertex in a k-cycle intersection graph. Clearly, the classification of the vertices in a k-cycle intersection graph will be also achieved in the CCPG-KIG transformation.

Theorem 3: The CCPG-KIG algorithm will transform a cyclic channel precedence graph into a k-cycle intersection graph. The time complexity of the CCPG-KIG algorithm is in $O(n)$ time, where n is the number of vertices in a cyclic channel precedence graph.

Proof: Basically, the CCPG-KIG algorithm transforms a cyclic channel precedence graph into a k-cycle intersection

graph. Besides that, the classification of vertices in the k-cycle intersection graph is also done by the algorithm. By Lemma 2, any vertex is in at most four minimal cycles and the length of each minimal cycle is four. Therefore, the time complexity of visiting a k-cycle pattern is in $O(1)$ time, and the representative of each k-cycle pattern is found by visiting a minimal cycle in $O(1)$ time. Clearly, all the k-cycle patterns will be found by visiting all of the minimal cycles. It is easy to understand that the time complexity of visiting all of the minimal cycles is the same as that of the vertex traversal in a cyclic channel precedence graph by a DFS algorithm. Hence, the time complexity of finding all the k-cycle patterns is in $O(n)$ time. On the other hand, the intersection relation between any two k-cycle patterns is decided by visiting all the minimal cycles, and the classification of vertices in the k-cycle intersection graph is done by visiting all the minimal cycles. Since the time complexity of visiting all of the minimal cycles is in $O(n)$ time, the time complexity of the CCPG-KIG algorithm will be in $O(n)$ time, where n is the number of vertices in a cyclic channel precedence graph. Q.E.D.

IV. Region Definition and Ordering Assignment

As mentioned above, the RDAOA problem of minimizing the number of switchboxes in a building block placement will correspond to the MFVS problem for a channel precedence graph. Based on the solution of the MFVS problem for a channel precedence graph, an efficient RDAOA algorithm for minimizing the number of switchboxes is proposed. Basically, the solution of the MFVS problem for a channel precedence graph is obtained by the minimal-cycle phase and the long-cycle phase.

In the minimal-cycle phase, a channel precedence graph will be transformed into a k-cycle intersection graph in $O(n)$ time, where n is the number of vertices in a channel precedence graph. In the k-cycle intersection graph, each vertex represents one k-cycle pattern. As one vertex in the k-cycle intersection graph is selected, all the minimal cycles in the related k-cycle pattern will be broken by removing one common vertex to be defined as a switchbox. Hence, by breaking these selected the k-cycle patterns, all the minimal cycles in a channel precedence graph will be broken by removing a minimal set of vertices to be defined as switchboxes.

However, based on the physical structure of a floorplan graph, it is clear that some long cycles share vertices with minimal cycles. Furthermore, it is desirable that most of the long cycles can be broken at the same time in the minimal-cycle phase. Hence, in addition to the vertex weight, the second heuristic in-degree will be introduced to break one selected k-cycle pattern, that is, if the selected k-cycle pattern has at least two common vertices, one common vertex with larger in-degree will be removed to break the k-cycle pattern. As a result, all the minimal cycles and most of the long cycles in a channel precedence graph will be broken at the same time in the minimal-cycle phase, and the minimal-cycle phase will be solved by the following Minimal_Cycle algorithm.

In the algorithm, the procedure of removing a minimal set of vertices to break all the minimal cycles and most of long cycles is not stopped until the k-cycle intersection graph is empty. Basically, the operations in a loop statement are divided into four main steps: 1) vertex selection, 2) vertex removal, 3) vertex deletion, and 4) vertex search. In step 1), if there exists any independent vertex in the k-cycle intersection graph, the vertex will be selected. On the other hand, if not, any vertex with degree 2 will be selected. In step 2), all the minimal cycles in the related k-cycle pattern will be broken by removing one vertex with the largest (vertex-weight, in-degree) to be defined as a switchbox. In step 3), if unbroken minimal cycles in a connected k-cycle pattern are fully covered by one adjacent k-cycle pattern, the mapped vertex will be deleted from the k-cycle intersection graph. In step 4), the dependent vertices connecting the previous selected vertex or the deleted vertices will be checked whether these are new independent vertices. According to the description of the main steps, the Minimal_Cycle algorithm will be shown as follows:

Algorithm Minimal_Cycle
Input: a channel precedence graph;
Begin
Call the CPG-CCPG algorithm;
Call the CPG-KIG algorithm;
While (KIG is not empty)
Begin
{ Vertex Selection }
If (there exists any independent vertex u in KIG)
Retrieve the vertex u;
Else
Retrieve any vertex u with degree 2;
Endif
{ Vertex Removal }
Select the vertex v with maximal (vertex-weight, in-degree) in the related k-cycle pattern $P(u)$;
Mark the broken minimal cycles in the cyclic channel precedence graph;
$S_{\min} = S_{\min} \cup \{ v \}$;
{ Vertex Deletion }
Modify the k-cycle intersection graph by deleting the vertices which connect the vertex u and whose unbroken minimal cycles are fully covered by one adjacent k-cycle pattern;
{ Vertex Search }
Find new independent vertices from adjacent dependent vertices;
End
Return the vertex set, $S_{\min}$;
End

Theorem 4: The time complexity of the Minimal_Cycle algorithm is in $O(n)$ time, where n is the number of vertices in a channel precedence graph.

Proof: By Theorems 2 and 3, it is clear that the CPG-CCPG algorithm and the CCPG-KIG algorithm will take $O(|V|)$ time and $O(|V^*|)$ time, respectively. Because $O(|V|) = O(|V^*|) = O(n)$, the graph transformation from

a channel precedence graph to a k-cycle intersection graph will take $O(n)$ time, where n is the number of vertices in a channel precedence graph.

In a k-cycle intersection graph, each independent vertex represents a k-cycle pattern with at least one minimal cycle not covered by the other k-cycle patterns. In order to break these independent minimal cycles, all the independent vertices will be selected to break the minimal cycles in the related k-cycle patterns. Therefore, as there exists any independent vertex in a k-cycle intersection graph, the independent vertex will be selected to break all the minimal cycles in the related k-cycle pattern. On the other hand, if there exists no independent vertex in a k-cycle intersection graph, all the vertices in the graph are fully dependent vertices, that is, all the k-cycle patterns are covered by the other k-cycle patterns. By Lemma 4, the degree of at least one vertex in a k-cycle intersection graph is less than three. One dependent vertex with degree 2 will be selected to break all the minimal cycles in the related k-cycle pattern.

By Theorem 3, the independent vertices in a k-cycle intersection graph are found in the CCPG-KIG algorithm, and the dependent vertices with degree 2 in a k-cycle intersection graph are always adjacent the processed independent vertices. Therefore, the time complexity of the vertex selection will take $O(1)$ time. Again, because a k-cycle pattern has at most 11 vertices, the time complexity of the vertex removal will take $O(1)$ time by an exhaustive search. Since the intersection of two k-cycle patterns has at most two cycles, an independent vertex will have at most six adjacent dependent vertices. Therefore, the time complexity of the vertex deletion will take $O(1)$ time by an exhaustive detection. Finally, due to the construction of a k-cycle pattern, a dependent vertex will have at most eight adjacent dependent vertices. The time complexity of the vertex search will take $O(1)$ time by an exhaustive detection. Hence, the time complexity of the operations in the loop statement is in $O(1)$ time. Clearly, the time complexity of the Minimal_Cycle algorithm is in $O(|V_{k\text{-cycle}}|)$ time. Furthermore, because $O(|V_{k\text{-cycle}}|) = O(n)$, the time complexity of the Minimal_Cycle algorithm is in $O(n)$ time, where n is the number of vertices in a channel precedence graph. Q.E.D.

On the other hand, the remaining long cycles in the channel precedence graph will be broken by an iterative channel/switchbox definition and ordering assignment in the long-cycle phase. If there exists one vertex with in-degree 0, the vertex will be removed from the channel precedence graph, defined as a channel and assigned a routing order. If there exists one vertex with out-degree 0, the vertex will be removed from the channel precedence graph, defined as a channel and pushed into a stack. Based on the physical structure of a floorplan graph, the remaining long cycles may share vertices each other. In order to minimize the number of switchboxes in the long-cycle phase, the first heuristic out-degree and the second heuristic in-degree are applied to break the remaining long cycles. Until there exists no vertex with in-degree 0 or out-degree 0, one vertex with the largest (out-degree, in-degree) will be removed from the channel precedence graph and be defined as a switchbox. As the channel precedence graph is empty, the channels in the stack will be popped and assigned a routing order. Finally, all the vertices defined as switchboxes will be randomly assigned routing orders. According to the description of the algorithm, the RDAOA algorithm is shown in detail as follows:

```
Algorithm RDAOA
Input: a floorplan graph;
  Begin
    ORDER = 1; S_min = ∅; Stack = ∅;
    Call the FG-CPG algorithm;
    { The minimal-cycle phase }
    Call the Minimal_Cycle algorithm and
         Return a minimal set of vertex for
         the minimal cycles, S_min;
    Define all of the vertex in S_min as switchboxes;
    Assign the crossing pins for the switchboxes;
    Construct a new channel precedence graph
         by deleting S_min from the original
         channel precedence graph;
    { The long-cycle phase }
    While ( the channel precedence graph is not
         empty )
      Begin
        While (there exists any vertex w
             with in-degree 0 in the channel
             precedence graph)
          Begin
            Define the vertex w as a channel;
            Assign the order of the channel as
                 ORDER;
            ORDER = ORDER +1;
            Remove the vertex w from the
                 graph, and change the
                 in-degree value of the
                 related vertices;
          End
        While (there exists any vertex w with
             out-degree 0 in the channel
             precedence graph)
          Begin
            Define the vertex w as a channel;
            Push the vertex w into Stack;
            Remove the vertex w from the
                 graph, and change the
                 out-degree value of the
                 related vertices;
          End
        Retrieve the vertex w with
             the largest (out-degree,
             in-degree);
        Define the vertex w as a switchbox;
        S_min = S_min ∪ { w };
        Assign the crossing pins for
             the switchbox;
        Remove the vertex w from the
             graph, and change the
```

```
            in-degree value of the
            related vertices;
    End
While ( S_min is not empty )
    Begin
        Pop one vertex w in Stack;
        Assign the order of the
            channel as ORDER;
        ORDER = ORDER + 1;
    End
While ( S_min is not empty )
    Begin
        Retrieve any vertex w in S_min;
        Assign the order of the
            switchbox as ORDER;
        ORDER = ORDER + 1;
    End
End
```

Theorem 5: The time complexity of the RDAOA algorithm is in $O(n)$ time, where n is the number of line segments in a floorplan graph.

Proof: First, a floorplan graph is transformed into a channel precedence graph by the FG-CPG algorithm. The time complexity of the algorithm is in $O(n)$ time, where n is the number of line segments in a floorplan graph.

Basically, the RDAOA problem of minimizing the number of switchboxes is divided into the minimal-cycle phase and the long-cycle phase. As mentioned above, the minimal-cycle phase will be solved by the Minimal_Cycle algorithm and the time complexity of the algorithm is in $O(|V|)$ time. Again, the deletion of $S_{\min}$ from the channel precedence graph will require $O(|V|)$ time. Hence, the time complexity of the minimal-cycle phase is in $O(|V|)$ time.

On the other hand, the remaining long cycles will be broken by an iterative channel/switchbox definition and ordering assignment in the long-cycle phase. Basically, if there is not a long cycle in a channel precedence graph or all the long cycles have been broken in the minimal-cycle phase, the long-cycle phase will be implemented by a topological sorting. In contrast, if there exist some long cycles in the remaining channel precedence graph, the long-cycle phase will be implemented by a modified topological sorting except that one vertex with the largest (out-degree, in-degree) is defined as a switchbox when there is no vertex with in-degree 0 or out-degree 0. If the data structure of a channel precedence graph is maintained in an adjacent list, the topological sorting for the channel precedence graph will take $O(|V|)$ time. Finally, the ordering assignment for the channels in a stack and switchboxes will also take $O(|V|)$ time in the worst case. Hence, the time complexity of the long-cycle phase is in $O(|V|)$ time.

As mentioned above, the time complexity of the RDAOA algorithm for the RDAOA problem of minimizing the number of switchboxes will take $O(|V|)$ time. Since $O(|V|) = O(n)$, the time complexity of the RDAOA algorithm is in $O(n)$ time, where n is the number of line segments in a floorplan graph. Q.E.D.

V. Experimental Results

Our proposed algorithm has been implemented using standard C language and run on a SUN workstation under the Berkeley 4.2 UNIX operating system. Thirty examples including example Ex6 in the Cai and Wong's paper [20] are applied to measure the number of switchboxes in the RDAOA problem. On the other hand, an exhaustive algorithm based on the detection of all the cycles in a channel precedence graph and the exhaustive search of the clique cover problem is implemented to obtain the optimal solution of the number of switchboxes for these tested examples. Based on the optimal results of the exhaustive algorithm, a greedy algorithm[10] and the three versions, namely first, last, and arbitrary, of the Cai and Wong's algorithm [20] are also implemented to compare the number of switchboxes in the RDAOA problem. In the experimental results, the results of the greedy algorithm and the best result on these three versions of the Cai and Wong's algorithm are applied to compare the number of switchboxes with the results of our proposed algorithm.

In Table I, the experimental results of twelve representative examples with different sizes will be shown and compared on the exhaustive algorithm, the greedy algorithm, the Cai and Wong's algorithm, and our proposed algorithm. From the viewpoint of the numerical results, the contribution of our proposed algorithm is not only in the theoretical reduction of time complexity, but also in the number of switchboxes for the tested examples.

First, as mentioned above, the time complexity of our proposed algorithm in the RDAOA problem of minimizing the number of switchboxes is reduced from $O(n^3)$ described in the Cai and Wong's paper [20] to $O(n)$ time. Second, an optimal ratio γ_{optimal} is further applied to measure optimal degree of a proposed algorithm for any tested example such as

$$\gamma_{\text{optimal}}(\text{Alg}, \text{Ex}) = \frac{N_{\text{opt}}(\text{Ex})}{N(\text{Alg}, \text{Ex})} \times 100\%$$

where $N_{\text{opt}}(\text{Ex})$ is the number of switchboxes by an exhaustive algorithm for the example Ex, and $N(\text{Alg}, \text{Ex})$ is the number of switchboxes by a proposed algorithm Alg for the example Ex. The definition of the optimal ratio is applied to the greedy algorithm the Cai and Wong's algorithm and our proposed algorithm for all the tested examples. It is clear that our proposed RDAOA algorithm attains 100% optimal ratio for all the tested examples. The fact proves that the proposed algorithm is efficient for minimizing the number of switchboxes in the RDAOA problem.

For example 39 × 43, it contains 39 vertical line segments and 43 horizontal line segments in a floorplan graph, and these line segments will construct 37 minimal cycles and 9 long cycles in the floorplan graph. As a result, 37 minimal cycles and 9 long cycles will be broken in the minimal-cycle phase and 0 long cycle will be broken in the long-cycle phase. By running our algorithm, 14 switchboxes are defined to break all of the minimal cycles for the RDAOA problem and the experimental result is shown in Fig. 8. Furthermore, for example Ex6 in the Cai and Wong's paper [20], the experimental result is obtained by our algorithm and shown

TABLE I
THE EXPERIMENTAL RESULTS

Example (#V x #H)	# minimal cycle	# long cycle	Exhaustive Search		Greedy Algorithm			Cai and Wong's Algorithm			Our RDAOA Algorithm		
			#S	Time	#S	Time	%optimal	#S	Time	%optimal	#S	Time	%optimal
6 x 6	3	0	2	0.3 s	5	0.1 s	40.0%	2	0.1 s	100%	2	0.1 s	100%
13 x 12	9	1	4	1.5m	8	0.2 s	50.0%	4	0.2 s	100%	4	0.2 s	100%
16 x 18	10	0	4	6.1m	11	0.3 s	36.4%	4	0.5 s	100%	4	0.4 s	100%
23 x 21	11	1	6	14.2m	15	0.6 s	40.0%	6	0.8 s	100%	6	0.7 s	100%
20 x 24	16	3	9	31.3m	20	0.9 s	45.0%	9	1.2 s	100%	9	1.0 s	100%
28 x 35	21	4	10	45.6m	24	1.2 s	41.7%	10	1.6 s	100%	10	1.4 s	100%
32 x 42	31	6	15	1.2 h	33	1.4 s	45.5%	16	1.8 s	93.8%	15	1.7 s	100%
37 x 40	38	7	14	2.1 h	32	1.7 s	43.8%	15	2.3 s	93.3%	14	2.0 s	100%
39 x 43	37	9	14	1.7 h	34	1.8 s	41.2%	14	2.7 s	100%	14	2.2 s	100%
45 x 50	46	10	21	3.7 h	45	2.4 s	46.7%	22	3.1 s	95.5%	21	2.6 s	100%
51 x 49	54	8	16	5.2 h	41	2.6 s	39.0%	17	3.5 s	94.1%	16	2.8 s	100%
71 x 65	35	12	24	8.5 h	49	3.0 s	49.0%	25	4.1 s	96.0%	24	3.2 s	100%

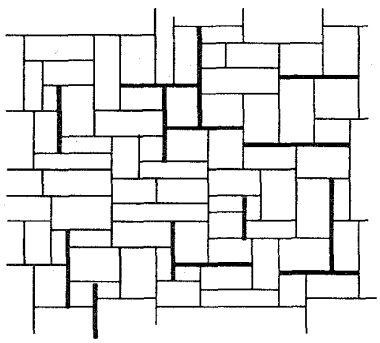


Fig. 8. An example 39 × 43.

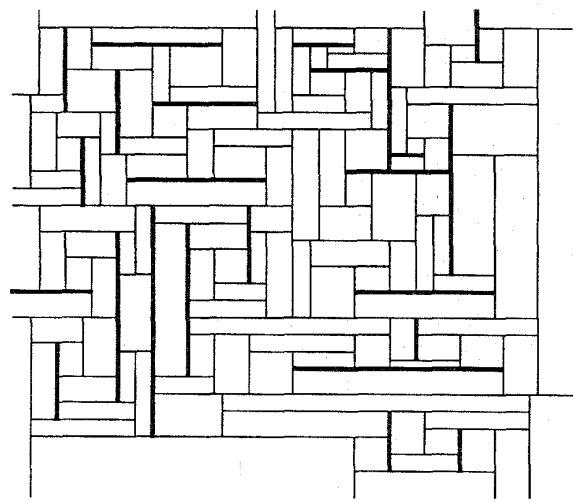


Fig. 9. An example 71 × 65 (Ex6).

TABLE II
THE COMPARISON OF 71 × 65 (EXAMPLE EX6)

Exhaustive Search	Greedy Method		The Cai-Wong Algorithm						Our RDAOA Algorithm	
			First		Last		Arbitrary			
	#S	%optimal	#S	%optimal	#S	%optimal	#S	%optimal	#S	%optimal
24	49	49.0%	25	96%	26	92.3%	25	96%	24	100%

in Fig. 9. The floorplan graph of example Ex6 contains 71 vertical line segments and 65 horizontal line segments. By the detection of an exhaustive search, the floorplan graph contains 35 minimal cycles and 12 long cycles. As a result, 24 switchboxes indicated by thick lines and 112 channels are defined to break all the minimal cycles and the long cycles.

Finally, the six results of example Ex6 has been obtained by six different algorithms: the exhaustive algorithm, the greedy algorithm, the first, last, and arbitrary versions of the Cai and Wong's algorithm, and our proposed algorithm. The number of switchboxes in the RDAOA problem are listed and compared in Table II. Clearly, the number of switchboxes defined by our algorithm is fewer than that defined by the other algorithms.

VI. CONCLUSION

For a building block placement, the routing space will be fully defined into channels and switchboxes because the definition of switchboxes releases all the cycles in a channel precedence graph and further yields a safe routing ordering. However, switchbox routing is more difficult than channel routing. Due to the requirements in the routing phase, it is important for us to minimize the number of switchboxes in the definition of channels and switchboxes. A region definition and ordering assignment (RDAOA) algorithm on minimizing the number of switchboxes is proposed in this paper. The time complexity of the algorithm is proved to be in $O(n)$ time, where n is the number of line segments in a given floorplan graph. Finally, several examples have been tested on the proposed algorithm and other published algorithms, and the experimental results show that our algorithm defines fewer switchboxes than other algorithms.

REFERENCES

[1] J. Soukup, "Circuit layout," *Proc. IEEE*, vol. 69, pp. 1281–1304, 1981.

[2] T. C. Hu and E. S. Kuh, Eds., *VLSI Circuit Layout: Theory and Design*. New York: IEEE, 1985.

[3] "Advance in CAD for VLSI," in *Layout Design Verification*, T. Ohtsuki, Ed., vol. 4. Amsterdam, The Netherlands: North Holland, 1986.

[4] B. Preas and M. Lorenzetti, Eds., *Physical Design Automat. VLSI Syst.* Menlo Park, CA: Benjamin/Cummings, 1988.

[5] W. M. Dai, T. Asano, and E. S. Kuh, "Routing region definition and ordering scheme for building-block layout," *IEEE Trans. Computer-Aided Design*, vol. 4, pp. 189–197, 1985.

[6] R. H. J. M. Otten, "Automatic floorplan design," in *Proc. 19th Design Automat. Conf.*, 1982, pp. 261–267.

[7] T. Chiba, N. Okuda, T. Kambe, I. Nishioka, and S. Kimura, "SHARPS: A hierarchical layout system for VLSI," in *Proc. 18th Design Automat. Conf.*, 1981, pp. 820–827.

[8] S. Kimura, N. Kubo, T. Chiba, and I. Nishioka, "An Automatic Routing Scheme for General Cell LSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 285–292, 1983.

[9] Y. Kajitani, "Order of channels for safe routing and optimal compaction of routing area," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 293–300, 1983.

[10] B. P. Preas and W. M. vanCleemput, "Routing algorithm for hierarchical IC layout," in *Proc. Int. Symp. Circuits Syst.*, 1979, pp. 482–485.

[11] C. P. Hsu, "A new two-dimensional routing algorithm," in *Proc. 19th Design Automat. Conf.*, 1982, pp. 46–50.

[12] F. Curatelli, "Region definition and ordering for macrocells with unconstrained placement," *Integration*, pp. 169–184, 1991.

[13] Y. Cai and D. F. Wong, "On minimizing the number of L-shaped channels in building-block layout," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 757–769, 1993.

[14] P. F. Dubois, "The channel intersection problem in the building block style layout," in *Proc. Int. Conf. Computer Design*, 1989, pp. 528–531.

[15] D. F. Wong and M. Guruswamy, "Channel ordering for VLSI layout with rectilinear modules," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1425–1431, 1991.

[16] A. T. Sherman, *VLSI Placement and Routing: The PI Project*. New York: Springer-Verlag, 1989.

[17] J. T. Yan and P. Y. Hsiao, "Routability crossing distribution and floating terminal assignment for T-type junction region," in *Fourth Great Lakes Symp. VLSI*, 1994, pp. 162–165.

[18] P. F. Dubois, A. Puissochet, and A. M. Tagant, "A general and flexible switchbox router: CARIOCA," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1307–1317, 1990.

[19] S. Sur-Kolay and B. B. Bhattacharya, "The cycle structure of channel graphs in nonslicible floorplans and a unified algorithm for feasible routing order," in *Int. Conf. Computer Design*, 1991, pp. 524–527.

[20] Y. Cai and D. F. Wong, "Channel/switchbox definition for VLSI building-block layout," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 1485–1493, 1991.

[21] H. Cai, "On empty rooms in floorplan graphs: comments on a deficiency in two papers," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 795–797, 1989.

[22] ——, "Connectivity biased channel construction and ordering for building-block layout," in *Proc. 25th Design Automat. Conf.*, 1988, pp. 560–565.

[23] H. Cai and H. J. M. Otten, "Conflict-free channel definition in building-block layout," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 981–988, 1989.

[24] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[25] K. J. Supowit and E. A. Slutz, "Placement algorithms for custom VLSI," in *Proc. 20th Design Automat. Conf.*, 1983, pp. 164–170.

[26] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.

[27] J. A. Mchugh, *Algorithmic Graph Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

Jin-Tai Yan received the B.S., M.S., and Ph.D. degrees in computer and information science from the National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1988, 1990, and 1995, respectively.

His main research interests are in the area of computer-aided design with an emphasis on routing and placement in VLSI layout, and logic synthesis of VLSI circuits.

Pei-Yung Hsiao was born on January 5, 1957 in Taiwan, R.O.C. He received the B.S. degree in chemical engineering from Tung Hai University in 1980, and the M.S. and Ph.D. degrees in electrical engineering from the National Taiwan University, R.O.C., in 1987 and 1990, respectively.

Since August 1990, he has been an Associate Professor with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan. In November 1991, he was a Technique Consultant with the Piping Engineering Department, CTCI Corporation, Taipei, Taiwan. He was a Visiting Senior Fellow with the National University of Singapore from July to September 1993. He was the President of Taiwan-IGUG for Intergraph Corporation from 1992 to 1994. His main research interests are VLSI-CAD, piping engineering design automation, DSP chip design, neural networks, and expert system applications. He has published more than 60 articles in conferences and authoritative journals.

Dr. Hsiao ranked second in the 1985 Electronics Engineering Award Examination conducted by the R.O.C. government for studying abroad, and was awarded the 1990 Acer Long Term Ph.D. Dissertation Award.