

Pin Assignment with Global Routing for General Cell Designs

Jingsheng (Jason) Cong, *Member, IEEE*

Abstract—In this paper, we present an algorithm which combines the pin assignment step and the global routing step in the physical design of VLSI circuits. Our algorithm is based on two key theorems: the channel pin assignment theorem and the block boundary decomposition theorem. These two theorems enable us to deal successfully with the high complexity resulting from combining the pin assignment and global routing steps. According to these two theorems, we only need to generate a coarse pin assignment and global routing solution. The exact pin locations and global routing topology can be determined optimally later by a linear time algorithm. We implemented a pin assignment and global routing package named *BeauticianGR* based on the proposed algorithm. *BeauticianGR* produces very satisfactory results on test circuits and is being integrated into the automatic layout design system at the National Semiconductor Corporation.

I. INTRODUCTION

GENERAL cell design style [10] is widely used for the design of complicated VLSI circuits. In general cell design style, a circuit is partitioned into a set of general cells, referred to as *blocks*. Usually, each block has a well-defined logic function and is to be implemented in the next level of design hierarchy (for example, using standard cells). The goal of a general cell layout design system is to determine the dimensions of the blocks, the locations of the blocks and the interconnections between the blocks so that chip area is utilized in the best possible way. Due to the high complexity of the physical design problem, most systems divide the physical design process into four steps: floorplanning, pin assignment, global routing, and detailed routing. In the floorplanning step, the dimensions and locations of the blocks are determined. In the pin assignment step, the locations of pins on the block boundary are determined. During global routing, we determine the connecting paths for pins that belong to the same net and are located in different blocks. During detailed routing, we use a channel router or a switchbox router to assign wires in each routing region to specific layers and tracks to implement the connecting paths determined in global routing. Clearly, such a partition of the physical design process is based on the well-

known methodology called *divide-and-conquer*. Much work has been done on floorplanning, global routing, and detailed routing. However, we have so far seen only limited progress in the pin assignment problem.

Previous approaches to the pin assignment problem can be briefly summarized as follows: the concentric circle mapping method [5] processes pins on a block by block basis. Two concentric circles are drawn for each block. The inner circle corresponds to the boundary of the block currently being considered. Pins in other blocks are mapped to points on the outer circle. The best mapping between points on the two circles is computed, which leads to an assignment of pin positions for the current block. In the nine-zone method [8], assignment of pins is also performed on a block by block basis. The plane is divided into nine possible zones to assist the assignment. The topological pin assignment method was proposed in [1], in which blocks are processed one at a time. A sweeping arm with one end fixed at the center of the current block sweeps the whole plane to determine the topological pin assignment. In [15], [16], a physical analogy was used for the pin assignment problem. Attractive forces are assumed between pins that belong to the same net. The equilibrium configuration of the system is determined which yields an assignment of pin positions for pins in all blocks.

A careful study shows that these approaches face two major difficulties.

1) It is very difficult to evaluate a pin assignment solution. The chip area and the total wirelength are the two most important objectives to be optimized in the physical design process. However, neither of them can be estimated accurately without carrying out the global routing step. Fig. 1(a) shows two possible assignments of the positions for pins of a two-pin net. However, it is difficult to determine which assignment is better.

2) The objective in the pin assignment step might conflict with the objective in the global routing step. Different global routers may produce different results on the same pin assignment solution. It is very difficult to design a good pin assignment algorithm without knowing the mechanism to be used in global routing. For example, if we assign the two pins of a net to locations as shown in Fig. 1(b) we could hope that they will be connected according to the dashed lines by a global router using a shortest path based algorithm. However, if we use a den-

Manuscript received January 26, 1990; revised June 8, 1990. This paper was recommended by Associate Editor A. E. Dunlop.

This is an expanded version of the paper presented at ICCAD'89, Santa Clara, CA.

The author is with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90024.

IEEE Log Number 9144590.

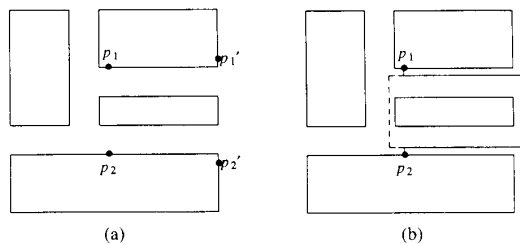


Fig. 1. Problems in previous approaches.

sity driven global router, it is quite possible that the global router detects that the middle vertical channel is too crowded, and thus chooses the connection path indicated by the solid lines. In this case, the choice made at the pin assignment step is not good at all.

Both difficulties are due to one reason: the pin assignment step is carried out independently of the global routing step. It is clear that the pin assignment step and the global routing step are very closely related. This suggests that considering pin assignment and global routing at the same time should lead to better designs if we can handle the higher complexity resulting from combining these two steps.

To overcome these difficulties, we apply in this paper a methodology called *marriage-before-conquest*. We design an algorithm which performs pin assignment and global routing in one step. Here the keyword is marriage. If we simply merge two problems, we shall face extremely high complexity. However, if we successfully marry two problems, we shall see very nice harmony. A successful marriage between the pin assignment problem and the global routing problem is based on the channel pin assignment theorem and the block boundary decomposition theorem to be shown in later sections. As a result of these two theorems, the combined problem is reduced to that of finding a coarse pin assignment and global routing solution, since the precise pin locations and global routing connections can then be determined later optimally in linear time. These two theorems also lead to a finite time deterministic algorithm for solving the pin assignment problem optimally. Prior to this work, there was no finite procedure which guarantees an optimal solution to the pin assignment problem.

Since most of the previous algorithms for pin assignment (except [15] and [16]) carry out the assignment step on a block by block basis, they also face another problem. Their pin assignment solutions are very sensitive to the order in which the blocks are processed. However, little is known about how to determine the best block ordering. The difficulty is also overcome in our approach, because our algorithm processes all the pins and blocks simultaneously.

The remainder of this paper describes our algorithm. In Section II, we formulate the new combined problem and state a few technical assumptions. In Section III, we state two important theorems, which form the foundation of our algorithm. In Section IV, we describe the algorithm in

detail. Experimental results are presented in Section V. Several extensions are presented in Section VI.

II. FORMULATION OF THE PROBLEM

In this paper, we combine the pin assignment problem and the global routing problem into one problem, which we call the *global connection arrangement problem*. Intuitively, a solution to the global connection arrangement problem consists of an assignment of the pins and a global routing solution associated with this assignment. In the rest of this section, we define the notion of validity and optimality of solutions to the global connection arrangement problem.

2.1. Valid Global Connection Arrangements

We are given a floorplan as input which specifies the location and the dimensions of each block. Fig. 2 shows a floorplan. For a block b , $x(b)$ and $y(b)$ denote the coordinates of the center of b , and $w(b)$ and $h(b)$ denote the width and the height of b , respectively. Channels are rectangular routing regions between blocks (or between a block and the chip boundary) as defined in [9]. (In Fig. 2, we label the channel formed between blocks, but we did not label the channels formed between the blocks and the chip boundary.) We use $d(C)$ to denote the density of channel C . Given a pin p , we use $blk(p)$ to denote the block that p belongs to. Given a net N , we use $blks(N)$ to denote the set of blocks which contain a pin in net N and we use $|N|$ to denote the size of net N (i.e., the number of pins in N). The *center* of a net N , $(cx(N), cy(N))$, is defined as follows:

$$cx(N) = \frac{\sum_{b \in blks(N)} x(b)}{|N|}, \quad cy(N) = \frac{\sum_{b \in blks(N)} y(b)}{|N|}.$$

The minimum distance between any two pins is specified by a constant λ , called *feature separation*, which is specified by the design rules. A *valid global connection arrangement* (valid GCA) consists of a mapping f and a Steiner forest F which satisfy the following two conditions.

C1: For each pin p , f maps p to a point $f(p)$ on the boundary of $blk(p)$. For two pins p and q on the same block, the distance between $f(p)$ and $f(q)$ is at least λ .

C2: For each net N , there is a Steiner tree $S(N)$ in F which connects all the $f(p)$'s for $p \in N$. The Steiner points in $S(N)$ are channel intersection points. Each edge in $S(N)$ represents a channel.

Note that a mapping f satisfying condition C1 is a valid solution to the pin assignment problem, and a Steiner forest F satisfying condition C2 is a valid solution to the global routing problem. In previous approaches, f is computed first before we start the computation of F . In our approach, we compute f and F at the same time. Fig. 3 shows the partial global connection arrangement for a net in Fig. 2.

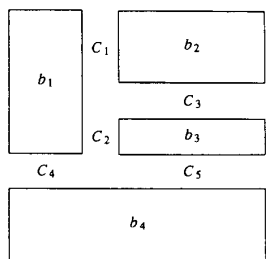


Fig. 2. A valid floorplan.

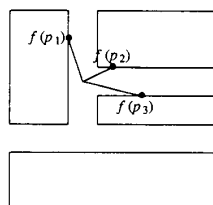


Fig. 3. The partial global connection arrangement for a net.

2.2. Optimal GCA

For a given valid GCA, we estimate the chip area as follows. For a horizontal line l_h , we define the *local width at l_h* , denoted $W(l_h)$, to be

$$W(l_h) = \sum_{b \cap l_h \neq \phi} w(b) + T \cdot \sum_{C \cap l_h \neq \phi} d(C)$$

where T is the track-to-track spacing, $b \cap l_h \neq \phi$ means that block b intersects line l_h , and $C \cap l_h \neq \phi$ means that (vertical) channel C intersects line l_h . (See Fig. 4.) We define the *estimated chip width* W to be $W = \max_{all l_h} W(l_h)$. Similarly, for a vertical line l_v , we define the *local height at l_v* , denoted $H(l_v)$, to be

$$H(l_v) = \sum_{b \cap l_v \neq \phi} h(b) + T \cdot \sum_{C \cap l_v \neq \phi} d(C).$$

We define the *estimated chip height* H to be $H = \max_{all l_v} H(l_v)$. The *estimated chip area* A is defined to be the product of the estimated chip width and the estimated chip height, i.e., $A = W \cdot H$.

Next, we estimate the total wirelength. We define the *span of a net N in a channel C* , denoted $s(C, N)$, to be the distance in the direction parallel to the channel between the leftmost pin of N in C and the rightmost pin of N in C . (See Fig. 5.) The *total span $TS(C)$ in a channel C* is defined to be

$$TS(C) = \sum_{N \cap C \neq \phi} s(C, N)$$

where $N \cap C \neq \phi$ means that there are pins that belong to net N in channel C . Clearly, $TS(C)$ is an estimation of total wirelength in channel C . We define the *estimate total wirelength WL* to be

$$WL = \sum_{all\ channel\ C} TS(C).$$

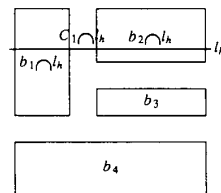


Fig. 4. The definition of local width at l_h .

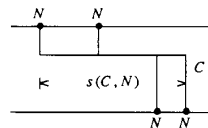


Fig. 5. Definition of the span of a net in a channel.

Note that in the estimation of chip width and chip height, we use channel density to approximate the number of tracks to be used in a channel. In the estimation of total wirelength in a channel, we leave out the short wires that are perpendicular to the channel. In general, these approximations are acceptable. Moreover, it is not difficult to see that the estimated chip area A and the estimated total wirelength WL thus defined are lower bounds on the final chip area and the final wirelength after detailed routing. It has been shown from experimental results that the final chip area and the final wirelength are very closely related to these lower bounds.

Given a valid GCA, it is straight forward to compute the estimated total wirelength since the total span in each channel can be obtained easily by summing up the spans of the nets. However, it is less trivial to compute the estimated chip area. Nevertheless, it can still be computed in linear time as follows: first, given a floorplan, we construct the *horizontal channel position graph*. Each node in the graph represents either a block or a channel. There is a direct edge from block b to channel C if b and C are adjacent in the floorplan and C is to the right of b . Also, there is a direct edge from channel C to block b if C and b are adjacent in the floorplan and b is to the right of C . For a block node b , the weight of the node is assigned to be $w(b)$, where $w(b)$ is the width of block b . For a channel node C , the weight of the node is assigned to be $T \cdot d(C)$, where $d(C)$ is the density of channel C . Fig. 6 shows the horizontal channel position graph for the floorplan shown in Fig. 2 (l_1 and l_2 represent the channels formed with the left side boundary of the chip, r_1 , r_2 , and r_3 represent the channels formed with the right side boundary of the chip.) It is not difficult to show that the horizontal channel position graph thus constructed is a direct acyclic graph. Moreover, the maximum weight of a direct path in the graph equals the estimated chip width. (The weight of a path is defined to be the sum of the weights of the nodes in the path.) Using depth-first search, we can compute a maximum weighted path in linear time [6]. Similarly, we can construct the vertical channel position graph and compute the estimated chip height in lin-

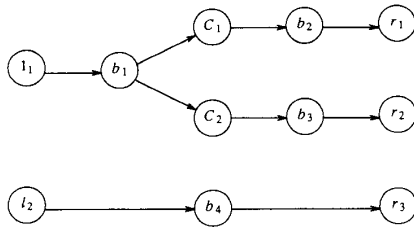


Fig. 6. Horizontal channel position graph for the floorplan in Fig. 2.

ear time. Therefore, the estimated chip area can be computed in linear time. Here we can see clearly one of the advantages of considering pin assignment and global routing together: we can estimate the chip area and the total wirelength easily and accurately. On the other hand, such estimations cannot be obtained efficiently if we are given only a pin assignment solution.

The *GCA problem* is to find a valid GCA such that the ordered pair (A, WL) is minimized with respect to the lexicographical order (i.e., $(A_1, WL_1) \leq (A_2, WL_2)$ iff $A_1 < A_2$ or $A_1 = A_2$ and $WL_1 \leq WL_2$). We call such a valid GCA an *optimal global connection arrangement* (optimal GCA). The lexicographic order we have chosen means that minimization of the chip area A has a higher priority than minimization of the total wirelength WL . Note that most physical design problems (such as the global routing problem by itself) have a finite (though, often exponential) number of valid solutions. Therefore, an optimal solution can always be found by exhaustive search in finite time. Unlike those problems, the pin assignment problem, as well as our global connection arrangement problem, have an infinite number of valid solutions, since we have an infinite number of ways to assign a pin to the boundary of its block as long as the feature separation is preserved. Therefore, a simple exhaustive search algorithm is not capable of producing an optimal solution in finite time. In fact, none of previous algorithms for the pin assignment problem is able to guarantee an optimal solution in finite time. We shall show that the two theorems in the next section will lead to a finite time algorithm to solve the global connection arrangement problem as well as the pin assignment problem optimally.

2.3. Some Technical Assumptions

For the simplicity of the presentation of our algorithm, we state a few technical assumptions here:

- A1: all blocks are rectangular in shape;
- A2: no pin has been pre-assigned;
- A3: each net has at most one pin on each block.

These assumptions are not essential to our algorithm. However, they make the presentation of our algorithm easier. We shall show in Section VI how our algorithm can be applied to situations where some of these assumptions do not hold. In particular, we show that our algorithm can be extended to the cases where the floorplan

contains L-shaped blocks, some blocks are predesigned (which means that the pin positions on these blocks are fixed), or some pins have constraints on their positions.

III. TWO FUNDAMENTAL THEOREMS

In this section, we state two theorems which form the foundation of our algorithm to be presented in the next section. These two theorems allow us to decompose the global connection arrangement problem into two subproblems such that the optimal solutions to the two subproblems can be combined to yield an optimal solution to the original problem.

3.1. Channel Pin Assignment Theorem

Let us look at a simplified pin assignment problem, i.e., the pin assignment problem for channels. Given a channel C , we are interested in the problem of arranging pins on the bottom edge and the top edge of the channel such that the ordered pair $(d(C), s(C))$ is minimum, where $d(C)$ is the density of C and $s(C)$ is the total span in C . We call this problem *the channel pin assignment problem*. (One basic step in our algorithm in the next section is to solve such a problem.) The general solution to this problem is not known. However, we shall show that this problem can be solved for a special class of channels, called basic channels, and this result on basic channels suffices the need of our algorithm.

Now we introduce the notion of a basic channel. Given a channel C , we say that a net N is a *basic net* in C if N has at most one pin on the top edge of C and one pin on the bottom edge of C , (N may also have up to two exits, one on the left side of C and the other on the right side of C .) We say that a channel C is a *basic channel* if every net in C is a basic net. The following theorem says that we can solve the channel pin assignment problem optimally for basic channels.

Channel Pin Assignment Theorem: Given a basic channel C , we can assign the pins on the top edge and the bottom edge of C in $O(n + e)$ time such that the ordered pair $(d(C), s(C))$ is minimum, where n is the number of pins in the channel and e is the number of exits in the channel.

We shall prove this theorem in Section IV-4.2, where we shall present a linear time algorithm to assign pins optimally in a basic channel. This theorem leads to the block boundary decomposition theorem in the following.

3.2. Block Boundary Decomposition Theorem

First, we introduce the concept of block boundary decomposition. For each block b in a given floorplan, we divide the boundary of b into a set of intervals such that each interval is either the top edge or the bottom edge of a channel adjacent to b . We call this operation *block boundary decomposition*. Such a decomposition can be easily obtained from the channel intersection graph [9]. Each edge in the channel intersection graph corresponds to an interval on the boundaries of the blocks adjacent to

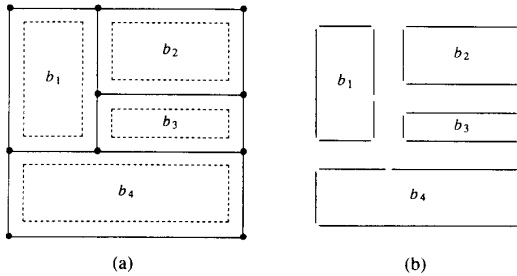


Fig. 7. Block boundary decomposition of the example in Fig. 2. (a) Channel intersection graph. (b) Block boundary decomposition.

the edge. Fig. 7(a) shows the channel intersection graph of the floorplan in Fig. 2, and Fig. 7(b) shows the block boundary decomposition for the given floorplan.

We now introduce the concept of coarse GCA. A *coarse global connection arrangement* (CGCA) is a pair (g, G) where g is a mapping which maps each pin p to an interval $g(p)$ on the boundary of $blk(p)$ and G is a Steiner forest satisfying condition C2 (in Section II-2.1). We use the term coarse arrangement because in a CGCA we only know the interval to which a pin belongs but do not know the exact location of the pin in its interval. For example, Fig. 8(a) shows a partial CGCA for a net. A CGCA (g, G) is *valid* if the number of pins assigned to each interval is no more than the maximum number of pins allowed in that interval. That is, for each interval I obtained by the block boundary decomposition, $\lambda \cdot pins(I) \leq length(I)$, where $pins(I)$ is the number of pins assigned to I by the mapping g and $length(I)$ is the length of the interval I . We say that a GCA (f, F) is a *refinement* of a CGCA (g, G) if $f(p)$ is in interval $g(p)$ for each pin p and F is isomorphic to G . For example, Fig. 8(b) shows the partial GCA which is a refinement of the CGCA in Fig. 8(a). Clearly, a valid CGCA always has a valid GCA as its refinement, since given a valid CGCA, we can simply assign pins in each channel in an arbitrary order to obtain a valid refinement provided that the minimum separation requirement is satisfied. A valid GCA (f, F) is an *optimal refinement* of a valid CGCA (g, G) if (f, F) is a refinement of (g, G) and its cost is minimum among all the valid CGA's which are refinements of (g, G) . Now we are ready to state the following important theorem.

Block Boundary Decomposition Theorem: For a given block boundary decomposition, there is only a finite number of CGCA's. Moreover, given a CGCA, we can find its optimal refinement in $O(n + k \cdot m)$ time, where k is the number of nets, m is the number of blocks, and n is the number of pins.

Proof: First, we show that there is a finite number of CGCA's under the block boundary decomposition. Let (g, G) be a CGCA. Clearly, there is finite number of nonisomorphic G 's since each G is a Steiner forest and each Steiner point has to be a channel intersection point. We shall show that there is finite number of distinct g 's. According to Assumption A1 that each block is a rectangle, it is not difficult to show by induction that the number

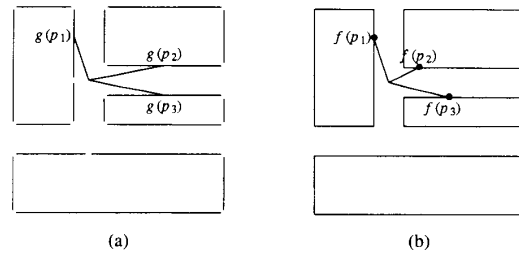


Fig. 8. CGCA and its refinement. (a) CGCA for a net. (b) Its refinement (CGCA for the net).

of intervals on the boundary of each block is at most $m + 3$. Therefore, there are at most $m + 3$ ways to map a pin to an interval on its block boundary. Thus there are at most $(m + 3)^n$ distinct g 's. Therefore, there is a finite number of CGCA's.

Next, we show that an optimal refinement of a CGCA can be computed efficiently. Suppose (g, G) is a given valid CGCA. Clearly, the assignment of pins in one channel can be carried out independently of the assignment of pins in other channels, since we do not need to shift a pin from one interval to another interval. Moreover, since the block dimensions are fixed, when $(d(C), s(C))$ is minimized for every channel C , (A, WL) is minimized, which results in an optimal refinement of (g, G) . For each channel C , according to the definition of the block boundary decomposition, the top edge of C belongs to the boundary of a single block, and the bottom edge of C also belongs to the boundary of a single block. According to Assumption A3, a net has at most one pin on each block. Therefore, each net in C is a basic net. Thus C is a basic channel. That is, every channel is a basic channel. According to the channel pin assignment theorem, for each channel C , we can assign the pins in C in time $O(n_c + e_c)$ such that $(d(C), s(C))$ is minimized, where n_c is the number of pins in C and e_c is the number of exits in C . It is not difficult to show by induction that the total number of channels is at most $8m$. Since each channel has at most $2k$ exits, the total number of exits $\sum_{all C} e_c \leq 2k \cdot 8m$. Therefore, the total time for processing all the channels is

$$\sum_{all C} (n_c + e_c) = \sum_{all C} n_c + \sum_{all C} e_c = O(n + k \cdot m).$$

□

The theorem has three important consequences. First, it shows that the global connection arrangement problem is in *NP*. Since there is a finite number of distinct CGCA's, we can design a polynomial time nondeterministic algorithm for the global connection arrangement problem which guesses a valid CGCA and then computes an optimal refinement in polynomial time. Second, it enables us to find an optimal GCA in finite time. We can easily design a branch-and-bound algorithm which enumerates all the valid CGCA's and computes an optimal refinement of each of them. An optimal GCA must be one of these optimal refinements. Third, the theorem suggests

that we can divide the global connection arrangement problem into two parts: CGCA and channel pin arrangement. Since we can solve the channel pin assignment problem optimally, we need only concentrate on the coarse global connection arrangement problem. Instead of assigning a pin to a point, we only need to assign a pin to a channel, which reduces the solution space significantly (when we say assigning a pin to a channel, we mean to assign the pin to the interval adjacent to the channel on the block boundary). Such a simplification allows us to design better and faster heuristic algorithms. We shall present an efficient heuristic algorithm for the coarse global connection arrangement problem and an optimal algorithm for the channel pin assignment problem in the next section.

IV. ALGORITHM FOR COMPUTING AN OPTIMAL GCA

In this section, we shall present an algorithm for computing an optimal GCA. It is not difficult to conclude that the problem of computing an optimal GCA is *NP*-hard since it includes the global routing problem. Although the block boundary decomposition theorem enables us to obtain an optimal GCA in finite time by exhaustive search, most likely the search time for an optimal solution will be exponential. The algorithm we shall present in this section is a polynomial time heuristic algorithm. Since the block boundary decomposition theorem enables us to reduce the solution space significantly, our algorithm performs well in general. Based on the block boundary decomposition theorem, our algorithm works in two stages:

Stage 1: compute an (approximation of) optimal valid CGCA (g, G) ;

Stage 2: compute an optimal refinement of (g, G) .

where an *optimal valid CGCA* is defined to be a valid CGCA whose optimal refinement is an optimal GCA. For Stage 2, the problem can be solved optimally in polynomial time. For Stage 1, we use a polynomial time heuristic algorithm.

4.1. Stage 1: Compute an Optimal Valid CGCA

As defined in the preceding section, a CGCA consists of a pair (g, G) where the mapping g specifies the channel to which each pin is assigned and the Steiner forest G specifies the channels used for connecting each net. In this section, we introduce a more compact representation of a CGCA, which we call a net connection forest. First, we extend the notation of a channel intersection graph [9]. Given a floorplan, the channel intersection graph is an undirected graph in which each vertex represents a channel intersection and each edge represents a channel. To construct the floor connection graph, we carry out the following operations: for each channel C , we introduce a *channel vertex* $v(C)$ which divides the edge corresponding to C in the channel intersection graph into two edges. Also, for each block b , we introduce a *block vertex* $v(b)$. We connect $v(C)$ and $v(b)$ whenever C and b are adjacent

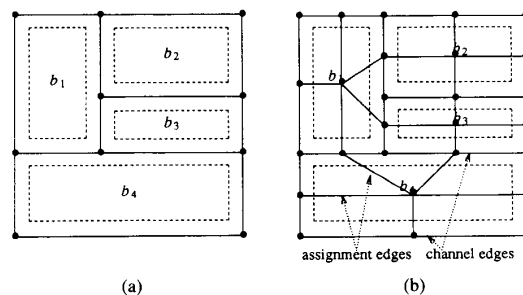


Fig. 9. Channel intersection graph and floor connection graph.

in the floorplan. We call the resulting graph the *floor connection graph*. Fig. 9(a) shows the channel intersection graph for the floorplan shown in Fig. 2, and Fig. 9(b) shows the corresponding floor connection graph. We classify the edges in a floor connection graph into two classes. If an edge connects a block vertex and a channel vertex, we call the edge an *assignment edge*. We call the rest of the edges *channel edges*. Clearly, each channel edge lies entirely in a channel and it is obtained by subdividing an edge in the corresponding channel intersection graph. In Fig. 9(b), we mark some assignment edges and channel edges.

For each net N , we shall represent the partial CGCA for the net by a tree $T(N)$, which we call the *net connection tree* of N . $T(N)$ is a subgraph of the floor connection graph in which the set of leaf vertices is the set of block vertices $\{v(b) | b \in \text{blks}(N)\}$ and each nonleaf vertex is a channel vertex or a channel intersection vertex. Fig. 10 shows a net connection tree for a three-pin net. It is important to note that $T(N)$ specifies both the assignment of pins in N to channels and the channels used to connect net N . For a pin p in N , assume that p is on block b . Then $v(b)$ appears in $T(N)$ as a leaf. Thus $v(b)$ is incident upon a unique assignment edge, the other end of which is a channel vertex $v(C)$ for some channel C , which specifies that we should assign p to channel C . The rest of the edges in $T(N)$ are channel edges, which specify the channels used to connect the pins in N . Let Σ be the disjoint union of the net connection trees of all the nets. We call Σ a *net connection forest*. Clearly, each net connection forest Σ corresponds to a CGCA (g, G) since the assignment edges in Σ define the mapping g and the channel edges in Σ define the Steiner forest G . Using such a compact representation, the objective of Stage 1 is to compute an optimal net connection forest. We say that a net connection forest Σ is *optimal* if there is an optimal GCA which is the refinement of the corresponding CGCA of Σ .

Our algorithm is basically an iterative elimination algorithm. It can be briefly outlined as follows. We say that a connected subgraph $H(N)$ of the floor connection graph is *admissible* for a net N if $H(N)$ contains at least one net connection tree of N as a subgraph. We say that an edge e in $H(N)$ is nonessential if $H(N)$ remains admissible after e is removed from $H(N)$. Initially, we compute an admissible graph $H(N)$ for each net N . Let Ω be the disjoint

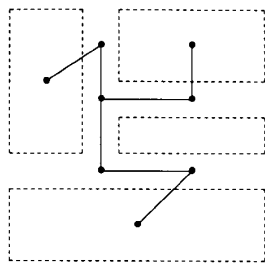


Fig. 10. Net connection tree.

union of the admissible graphs of all the nets. We repeatedly remove the “worst” nonessential edge from Ω until Ω is a net connection forest. Two important questions need to be answered: “how is the initial admissible graph for each net computed?” and “how are the edges to be removed in each iteration determined?”

We could use the whole floor connection graph as the admissible graph for each net initially since it definitely contains a net connection graph. Moreover, this set of initial graphs contains an optimal net connection forest since it contains all the possible net connection forests. However in this case there might be a lot of nonessential edges to be removed before a net connection tree is obtained. The existence of a large number of nonessential edges not only increases the computation time but also makes it more difficult to estimate congestion in channels. To overcome these difficulties, in our algorithm, for each net N , we compute a minimum rectilinear convex polygon $B(N)$ which encloses all the blocks in $blks(N)$. The initial graph $H(N)$ is chosen to be the subgraph of the floor connection graph in which each channel intersection and each channel is in $B(N)$. Moreover, we do not include a block vertex $v(b)$ in $H(N)$ if b is not in $blks(N)$. For example, assume that a net N_1 , $blks(N_1) = \{b_1, b_3\}$. Fig. 11(a) and (b) shows $B(N_1)$ and $H(N_1)$, respectively. Clearly, $H(N)$ is usually much smaller than the floor connection graph. We can show that this set of initial admissible graphs for all the net contains an optimal net connection forest. (The key step in the proof is that for any net N , we can always replace a path in $T(N)$ outside of $B(N)$ by a path inside of $B(N)$ without increasing any local width or height and without increasing total span in any channel. We leave out the details here.)

To determine the edges to be removed from Ω in the elimination process, we assign a weight $w(e)$ to each edge e in $H(N)$. A maximum weighted nonessential edge will be removed at each iteration. Weights of assignment edges and weights of channel edges are computed differently. For an assignment edge e in an admissible graph $H(N)$, suppose that e connects block vertex $v(b)$ and channel vertex $v(C)$. Let $I(b, C)$ denote the interval on the boundary of block b which is a top edge or a bottom edge of channel C . Let $n(b, C)$ denote the number of pins currently in $I(b, C)$. Moreover, let $m(b, C)$ denote the maximum number of pins allowed in $I(b, C)$ (which is determined by the length of the interval divided by the feature

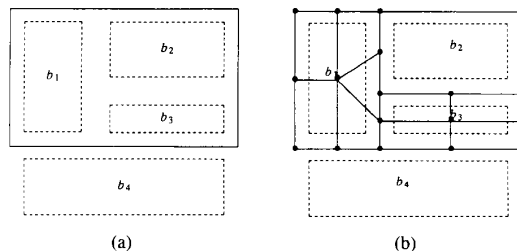


Fig. 11. Minimum rectilinear convex polygon and initial admissible graph (a) $B(N_1)$, (b) $H(N_1)$.

separation λ). Then, the weight of the assignment edge e is defined as follows:

$$w(e) = \alpha \cdot \frac{n(b, C)}{m(b, C)} + \beta \cdot t(C, N)$$

where α and β are two constant factors, and $t(C, N)$ is the rectilinear distance between the center of channel C and the center of net N . Note that $n(b, C)/m(b, C)$ measures how crowded the pins are in the interval $I(b, C)$. We choose α to be much larger than β to make sure that $n(b, C)$ never exceeds $m(b, C)$ (so that CGCA computed is valid). The appearance of the term $t(C, N)$ in the weight function reflects our preference to assign a pin to a channel closer to the center of the net which the pin belongs to.

For a channel edge e in an admissible graph $H(N)$, suppose that e corresponds to channel C . The weight of the channel edge e is defined to be

$$w(e) = a_c \cdot d(C) + b_c \cdot l(C) + c_c \cdot t(C, N)$$

where $d(C)$ is the density of C , $l(C)$ is the length of C , and $t(C, N)$ is the same as defined above. The parameters a_c , b_c , and c_c are constant factors satisfying the condition that $a_c > b_c > c_c$ (which weigh the contributions of $d(C)$, $l(C)$, and $t(C, N)$ to the weight of e). Moreover, we choose a_c (similarly for b_c and c_c) to be larger for a noncritical channel C . A channel is *critical* if an increase in the density of the channel will increase either the chip width or the chip height. Clearly, our algorithm tends to minimize congestion in critical channels since it assigns larger weights to the channel edges corresponding to the critical channels. Critical channels can be identified using the longest path algorithm for directed acyclic graphs [6]. We leave out the details here. The idea of identifying critical routing regions has also been used in other work, especially in [3].

We use a depth-first search algorithm similar to the bi-connectivity checking algorithm [12] to mark all the nonessential edges in $H(N)$ for each net N . This can be done in $O(m_N)$ time, where m_N is the number of edges in $H(N)$. In each iteration, we remove the maximum weighted nonessential edge in Ω from the corresponding admissible graph. Then we remark the nonessential edges in the resulting admissible graph and update the weights of affected edges due to the possible change of density of the

channel containing the deleted edge. Our algorithm stops when each $H(N)$ is a net connection tree (or equivalently, when Ω is a net connection forest).

To speed up the procedure of choosing a maximum weighted nonessential edge and the procedure of updating edge weights after the removal of an edge, we set up three *buckets* for each channel. For a channel C , the first bucket $bt_1(C)$ stores all the nonessential channel edges in Ω which lie in C , the second bucket $bt_2(C)$ stores all the nonessential assignment edges in Ω connected to the top edge of C , and the third bucket $bt_3(C)$ stores all the nonessential assignment edges in Ω connected to the bottom edge of C . Before we start the iterative elimination process, we sort the edges in each bucket according to their weights in nondecreasing order. Clearly, if we remove a channel edge in channel C , all the weights of the edges in $bt_1(C)$ decrease by the same amount (due to the reduction of channel density $d(C)$). Thus we do not have to resort the edges in $bt_1(C)$. Moreover, we can store the amount of change in a field called *offset* associated with each bucket, so that the update of all the weights of the edges in $bt_1(C)$ can be completed in constant time. Similarly, if we remove an assignment edge connected to the top (bottom) edge of C , all the weights of the edges in $bt_2(C)$ ($bt_3(C)$) decrease by the same of amount (due to the reduction of the number of pins assigned to that edge $n(b, C)$). Thus the update of all the weights of the edges in $bt_2(C)$ ($bt_3(C)$) can be completed in constant time by updating the *offset* field of $bt_2(C)$ ($bt_3(C)$). A maximum weighted nonessential edge can be found by comparing the maximum weighted nonessential edges in each bucket. (Remember that using the data structure suggested above, the weight of a nonessential edge e is computed as $w(e) - bucket(e).offset$, where $bucket(e)$ is the bucket which contains e). The algorithm for computing a CGCA is summarized in Fig. 12.

This algorithm has two advantages: first, the algorithm processes all the pins and blocks at the same time. Therefore, it avoids the difficult problem of determining the best processing order. Second, at any time, the algorithm maintains the global information on all the nets in all the channels. Therefore, it can eliminate connections in the most congested channels to avoid local congestion.

The complexity of our algorithm can be analyzed as follows. Let k be the number of nets, and m be the number of blocks. As we have pointed out in the proof of the block boundary decomposition theorem in Section III-3.2, the total number of channels is at most $8m$. Since each channel contributes three edges in the floor connection graph (two channel edges and one assignment edge), the number of edges in the floor connection graph is at most $24m$. Since each $H(N)$ is a subgraph of the floor connection graph, the number of edges in $H(N)$ is $O(m)$. Therefore, the total number of edges in Ω is $O(km)$. Now let us analyze the complexity of the algorithm shown in Fig. 12 line by line. Given a net N , the minimum rectangular convex polygon $B(N)$ can be computed using an algorithm similar to the algorithm for computing the convex hull of

Algorithm Computing a CGCA.

1. For each net N
 - compute $B(N)$ and $H(N)$.
 - Let Ω be the union of $H(N)$'s for all N .
2. For each net N
 - mark the non-essential edges in $H(N)$.
3. For each channel
 - create three buckets and sort edges in each bucket.
4. Repeat
 5. $e :=$ maximum weighted non-essential edge in Ω .
 6. remove e from the corresponding $H(N)$.
 7. re-mark the non-essential edges in $H(N)$.
 8. update the weights of the edges in the bucket containing e .
9. Until Ω is a net connection forest.

Fig. 12. Algorithm for computing a CGCA.

a point set in $O(m \log m)$ time [11]. Also, $H(N)$ can be obtained from $B(N)$ in $O(m)$ time. Thus Line 1 takes $O(km \log m)$ time. For each $H(N)$, we can mark the nonessential edges in it using an algorithm for computing the biconnected components in $O(m)$ time. Thus Line 2 takes $O(km)$ time. Let s be the total number of buckets. Then, $s = O(m)$, since the number of channels is $O(m)$. Let m_i denote the number of edges in the i th bucket. Since $\sum_{i=1}^s m_i = km$, Line 3 takes time

$$\sum_{i=1}^s m_i \log m_i \leq \left(\sum_{i=1}^s m_i \right) \cdot \log \left(\sum_{i=1}^s m_i \right) \leq km \log(km).$$

Therefore, Lines 1–3 (the initialization phase) take $O(km \log(km))$ time. Line 5 can be completed in $O(m)$ time by comparing the maximum weighted edge of each bucket. Line 6 takes constant time. Line 7 takes $O(m)$ time again by using the algorithm for computing the biconnected components. Line 8 can be completed in constant time by updating the *offset* field of the bucket containing e . Thus Lines 5–8 (each iteration) take time $O(m)$ time. Since there are $O(km)$ edges in Ω , we shall go through at most $O(km)$ edge deletions. Therefore, the entire elimination process takes $O(km^2)$ time. Hence, the complexity of our algorithm for computing a CGCA is

$$O(km \log(km)) + O(km^2) = O(km \cdot \max(\log k, m)).$$

4.2. Stage 2: Compute an Optimal Refinement of a Given CGCA

After we obtain a CGCA, we know the pins and exits in each channel. Moreover, there is no interaction between two different channels because the CGCA we computed in Stage 1 is always valid (which implies that the number of pins in each channel does not exceed the limit so that we do not have to shift pins from one channel to another). Therefore, we can process each channel independently. Furthermore, it has been shown in the proof of the block boundary decomposition theorem in Section III-3.2 that each channel is a basic channel. Thus we need only consider the pin assignment problem for basic channels. We say that a channel pin assignment is *optimal* if $(d(C), s(C))$ is minimized among all possible channel pin assignments. In the rest of this section, we present a linear time algorithm which always produces optimal pin assignments for basic channels. This result also completes the proof of the channel pin assignment theorem in Section III-3.1.

In a basic channel C , each net has at most two pins (one on the top edge and one on the bottom edge) and at most two exits (one at the left side and one at the right side). We can classify a net N in a basic channel into one of the following eight classes depending on whether N has pins on the top or bottom edge and has exits on the left or the right side:

Net Class	Pins	Exits
<i>pass by</i> (PB)	0	both
<i>left single</i> (LS)	1	left
<i>right single</i> (RS)	1	right
<i>balanced single</i> (BS)	1	both
<i>left pair</i> (LP)	2	left
<i>right pair</i> (RP)	2	right
<i>balanced pair</i> (BP)	2	both
<i>internal pair</i> (IP)	2	no

It is easy to see that such a classification includes all the possible basic nets. Fig. 13 illustrates each of the eight classes. Clearly, we can remove all the PB nets from a basic channel before we proceed to assign pins in the channel since there is no pin in the PB net to be assigned. Moreover, removal of the PB nets will not affect the optimal pin assignment for other nets since deleting a PB net will decrease channel density by one for any assignment solution. Therefore, in the rest of this section, we assume that there is no PB net in a basic channel. We call two LS nets a *pair of matched LS nets* if one net has the pin on the top edge of C and the other net has the pin on the bottom edge of C . (See Fig. 14(a).) Similarly, we call two RS nets a *pair of matched RS nets* if one net has the pin on the top edge of C and the other net has the pin on the bottom edge of C . (See Fig. 14(b).) In the description of our algorithm, we refer to a vertical line in the channel as a *column*. We do not require equal spacing between two adjacent columns, but we require the distance between any two adjacent columns to be at least λ . Our algorithm for channel pin assignment is as follows.

Algorithm: Optimal Channel Pin Assignment (OCPA)

Input: Given pins on the top edge and the bottom edge of the channel, and exits on the left side and right side of the channel. Each net is a basic net.

Step 1: Classify each net by one of the eight classes defined above. Remove all the PB nets from the channel.

Step 2: Starting from the left side of the channel, assign all matched LS nets pair-by-pair, one pair per column; Similarly, starting from the right side of the channel, assign all matched RS nets pair-by-pair, one pair per column.

Step 3: Starting from the left-most available position, assign all the LP nets one-by-one, one net per column; Similarly, starting from the rightmost available position, assign all the RP nets one-by-one, one net per column.

Step 4: Starting from the leftmost available position, assign unmatched LS nets one-by-one, one net per column.

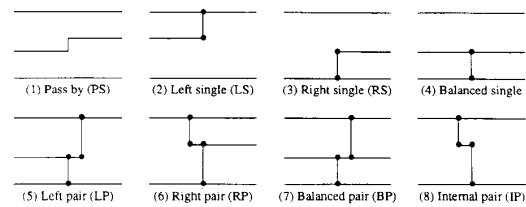


Fig. 13. Classification of nets in a basic channel.

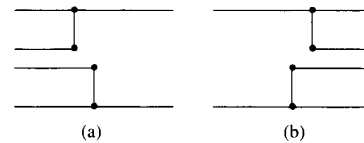


Fig. 14. A pair of matched (a) LS or (b) RS nets.

Step 5: Starting from the left-most position, assign all IP nets one-by-one, one net per column.

Step 6: Starting from the rightmost available position, assign all the unmatched RS nets one-by-one; some of them may share a column with unmatched LS nets assigned in Step 4.

Step 7: Assign BS nets and BP nets any available columns. They may share columns with unmatched LS nets or RS nets.

It is easy to see that if the number of pins on each edge of the channel does not exceed the maximum number of pins allowed on that edge (which is true since the CGCA we computed in Stage 1 is valid), our algorithm always gives a valid assignment. Optimality of the algorithm can be concluded from the following lemmas. The proofs of these lemmas are not difficult and will be omitted.

Lemma 1: There is an optimal solution in which each matched pair of LS nets or RS nets occupies a column, and all the matched LS pairs are to the left of the other nets and all the matched RS pairs are to the right of the other nets.

Lemma 2: Assume that there is no matched pair of LS nets or RS nets. There is an optimal solution in which each LP or RP net occupies a column, and all the LP nets are to the left of the other nets and all the RP nets are to the right of the other nets.

According to Lemmas 1 and 2, after the assignments in Step 1 and Step 2, we are guaranteed to have an optimal solution consistent with the current assignment. Now we need only consider channels with BS, BP, IP nets, and unmatched LS and RS nets. We call these channels *simplified basic channels*.

Lemma 3: In a simplified basic channel, there is an optimal solution in which each IP nets occupies a column. We call such a solution an *s-type optimal solution*.

Lemma 4: In a simplified basic channel, there is an *s-type optimal solution* in which all the unmatched LS nets are to the left of the other nets. We call such a solution an *st-type optimal solution*.

Lemma 5: The solutions produced by the OCPA algorithm are always *st*-type optimal channel pin assignment solutions for simplified basic channels.

Finally, combining these results, we have the following theorem.

Theorem: The OCPA algorithm produces an optimal channel pin assignment which minimizes $(d(C), s(C))$ in time $O(n_C + e_C)$, where n_C is the number of pins in the channel, and e_C is the number of exits in the channel.

V. EXPERIMENTAL RESULTS

Based on the algorithm described in Section IV, we implemented a pin assignment and global routing package named BeauticianGR. It is being integrated into the automatic physical design system at National Semiconductor Corporation. In the current version of BeauticianGR, some parts of the algorithm are simplified. For example, the floorplan input to BeauticianGR always has a slicing structure, since it is generated by a slicing floorplanner [14], [13]. We took advantage of the simplicity of slicing structures and reduced the problem of computing $B(N)$ to the problem of finding the least common ancestor in the slicing tree of the blocks in $blks(N)$ for a net N . Also, in the current implementation, the floorplan connection graph constructed by the program is simpler than the one defined in Section III-3.2. The package is written in the C language and run on a SUN 4/280 workstation. We tested BeauticianGR on two real VLSI circuits. One circuit, named TEST, is a small design with 61 nets. Another circuit, named SDDC, is a disk controller being designed at the National Semiconductor Corporation. Table I shows the complexity of the two circuits and the running time of our algorithm. We traced many nets in the outputs of our program and the quality of pin assignment and global routing is very good. We could not perform comparison with other approaches due to the lack of benchmark examples for the pin assignment problem.

VI. EXTENSIONS

The assumptions in Section II can be removed without affecting the structure of the algorithm. In the rest of this section, we extend our algorithm to the cases where the floorplan contains L-shaped blocks, or some blocks are predesigned (which means that pin positions on these blocks are fixed), or some pins have constraints on their positions (e.g., they must be placed in certain channels or on certain sides of the blocks).

6.1. Extension 1: The Floorplan Contains L-Shaped Blocks

When the floorplan contains L-shaped blocks, the routing region becomes more complicated. We still define a channel to be a rectangular routing region formed by two blocks. (Some people also use L-shaped channels [2] or monotone channels [4] when the floorplan contains L-shaped blocks.) The channel intersection graph can be

TABLE I
TEST CIRCUITS AND RUNNING TIME

Ex.	# Blocks	# IO Pads	# Nets	# Pins	Time (seconds)
TEST	6	15	61	145	10.0
SDDC	18	76	599	1619	4684.1

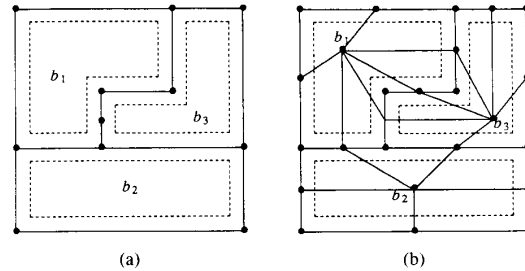


Fig. 15. The corresponding (a) channel intersection graph and (b) floor connection graph.

constructed efficiently using the plane sweeping technique. Fig. 15(a) shows the channel intersection graph of a floorplan with L-shaped blocks. We can still carry out the block boundary decomposition which divides the boundary of each block into a set of intervals such that each interval is adjacent to exactly one channel. It is not difficult to show that each channel is still a basic channel and the boundary of each block is divided into at most $2m + 4$ intervals (instead of $m + 3$ intervals when all the blocks are rectangular in shape as shown in Section III-3.1), where m is the number of blocks in the floorplan. Therefore, both the channel pin assignment theorem and the block boundary decomposition theorem still hold. The floor connection graph can be constructed from the channel intersection graph in the way as in Section IV-4.1 by introducing the block vertices and the channel vertices and then connecting each block vertex to its adjacent channel vertices. Fig. 15(b) shows the floor connection graph constructed from the channel intersection graph in Fig. 15(a). After we have constructed the floor connection graph, we can use the same algorithm shown in Fig. 12 to compute a good CGCA. Since each channel is still a basic channel, we can use the OCPA algorithm presented in Section IV-4.2 to compute the optimal refinement of the CGCA. Clearly, the same approach can be also applied to the floorplan with rectilinear blocks. (Rectangular blocks and L-shaped blocks are special cases of rectilinear blocks.) The only difference is in generating the floor connection graph.

6.2. Extension 2: Some Blocks Are Predesigned

If some of the blocks are predesigned, the positions of the pins in these blocks are fixed. In order to keep the fixed pins in their predetermined positions, we shall impose the following two constraints. i) In Stage 1, these pins remain in the channels they belong to. ii) In Stage 2,

these pins remain at the positions which are predetermined.

In order to meet constraint i), we exclude from the initial admissible graphs the assignment edges which assign the fixed pins to other channels which they do not belong to. For example, if net N_1 has two pins p_1 and p_2 , one is in block b_1 and the other is in b_3 , respectively. Moreover, block b_1 is pre-designed and pin p_1 is at the bottom side of b_1 as shown in Fig. 16(a). Then, the initial admissible graph $H(N_1)$ of net N_1 will be the one shown in Fig. 16(b). Note that we did not include in $H(N_1)$ the assignment edges from p_1 to the top channel, the left channel, or the two right channels (the reader may compare it with the initial admission graph shown in Fig. 11(b)). This forces us to assign p_1 to the bottom channel adjacent to b_1 .

In order to meet constraint ii), we need to extend the channel pin assignment algorithm in Section IV-4.2. It is easy to see that each channel is still a basic channel. However, some pins may have predetermined positions. Since the top or bottom edge of a channel is adjacent to exactly one block, there are the three following cases: a) no pins on the two channel edges have predetermined position. b) The pins on one edge have predetermined positions, while the pins on the other edge are free. c) The pins on both edges have predetermined positions. In case a), we can use the OCPA algorithm presented in Section IV-4.2. In case c), we do not have to assign any pins in the channel since all the pin positions are known. Now we present an algorithm for case b). Without loss of generality, we assume that the pins on the top edge of the channel have predetermined positions while the pins on the bottom edge are free.

Algorithm: Channel Pin Assignment with a Fixed Edge

Input: Pins on the top edge of the channel have predetermined positions, and the pins on the bottom edge are free. Each net is a basic net.

Step 1: Classify each net by one of the eight classes defined in Section IV-4.2. Remove all the PB nets from the channel.

Step 2: For each net of classes LP, RP, BP, or IP, assign the top pin at the predetermined position on the top edge and assign the bottom pin at the same column at the bottom edge.

Step 3: For each LS net, if its pin is on the top edge, assign it to the predetermined positions. If it is on the bottom edge, assign it to the leftmost available position on the bottom edge.

Step 4: For each RS net, if its pin is on the top edge, assign it to the predetermined positions. If it is on the bottom edge, assign it to the right-most available position on the bottom edge.

Step 5: For each BS net, if its pin is on the top edge, assign it to the predetermined positions. If it is on the bottom edge, assign it to any available position on the bottom edge.

It can be shown in a similar way as in Section IV-4.2 that this algorithm runs in linear time in terms of the num-

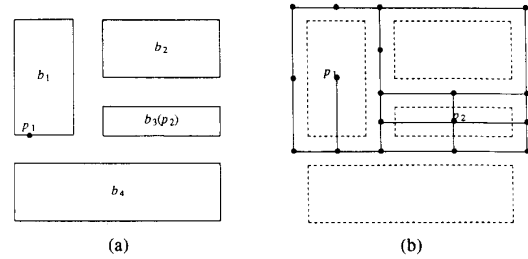


Fig. 16. Initial admissible graph for net with pre-designed pins (block b_1, p_1 is pre-designed). (a) Predetermined pins. (b) $H(h_1)$.

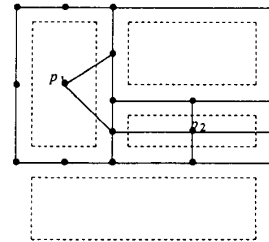


Fig. 17. Initial admissible graph for net with constrained pins. (p_1 has to be assigned to the right side of b_1 .)

ber of pins, exits in the channel, and minimizes the channel density and the total span in the channel (i.e., the ordered pair $(d(C), s(C))$).

6.3. Extension 3: Some Pins Have Constraints on Their Positions

It is possible that in practice some pins have constraints on their positions, although their positions are not completely fixed. For example, the designer may require that some pin be placed in certain channel or on certain side of the block. Usually, these constraints can be accommodated in our algorithm by choosing a subset of assignment edges for that pin in the initial admissible graph. Suppose that net N_1 has two pins p_1 and p_2 on blocks b_1 and b_3 , respectively. If p_1 has to be assigned to the bottom channel adjacent to block b_1 for some reason, we can use the initial admissible graph shown in Fig. 16(b) in the preceding subsection. Since in that graph p_1 is incident upon only one assignment edge which intersects the bottom channel, p_1 is guaranteed to be assigned to the bottom channel. If p_1 has to be assigned to the right side of block b_1 according to the designer, we can use the initial admissible graph shown in Fig. 17. In this graph, we let p_1 be incident upon only two assignment edges which intersect the right side of b_1 . Thus p_1 is forced to be assigned to one of the channels to the right of b_1 . From these examples, we can see that we can handle many constraints on pin positions by generating different initial admissible graphs at the beginning of our algorithm.

It is not difficult to see that in all three extensions, each channel is still a basic channel. Therefore, the channel pin assignment theorem and the block boundary decomposition theorem still hold. Thus the performance of our al-

gorithm is not affected. It is an open question whether we can solve the pin assignment problem optimally for non-basic channels. An optimal channel pin assignment algorithm was given in [7] for channels without exits.

ACKNOWLEDGMENT

The author thanks M. Esmail for introducing him to the problem and for his support throughout this project. He also thanks S. Sekar and P. Ramamswamy for their helpful discussions and assistance. Also, he is grateful to Prof. C. L. Liu for many valuable suggestions.

REFERENCES

- [1] H. N. Brady, "An approach to topological pin assignment," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 250-255, July 1984.
- [2] W. M. Dai, T. Assno, and E. S. Kuh, "Routing region definition and ordering scheme for building layout," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 189-197, July 1985.
- [3] W. M. Dai, M. Sato, and E. S. Kuh, "A dynamic and efficient representation of building-block layout," in *Proc. 24th Design Automation Conf.*, 1987, pp. 376-384.
- [4] M. Guruswamy and D. F. Wong, "Channel routing order for building-block layout and rectilinear modules," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1988, pp. 184-187.
- [5] N. L. Koren, "Pin assignment in automated printed circuit board design," in *9th Design Automation Workshop Proc.*, 1972, pp. 72-79.
- [6] E. L. Lawler, *Combinatorial Optimization*. New York: Holt, Rinehart and Winston, 1976.
- [7] H. W. Leong and C. L. Liu, "Permutation channel routing," in *Proc. Int. Conf. on Computer Design: VLSI in Computers*, 1985, pp. 579-584.
- [8] L. Mory-Rauch, "Pin assignment on a printed circuit board," in *15th Design Automation Conf. Proc.*, 1978, pp. 70-73.
- [9] B. Preas, "Placement and routing algorithms for hierarchical integrated circuit layout," Ph.D. dissertation, Stanford Univ., 1979.
- [10] B. Preas and M. Lorenzetti, eds., *Physical Design Automation of VLSI Systems*. Menlo Park, CA: Benjamin/Cummings, 1988.
- [11] F. P. Preparata and M. I. Shamos, *Computational Geometry*. New York: Springer-Verlag, 1985.
- [12] E. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1977.
- [13] S. Sekar, "CHIPMASON—A floorplanning program," National Semiconductor Corporation Internal Documents, 1988.
- [14] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, 1986, pp. 101-107.
- [15] X. Yao, M. Yamada, and C. L. Liu, "A new approach to the pin assignment problem," in *25th Design Automation Conf. Proc.*, 1988, pp. 566-572.
- [16] X. Yao and C. L. Liu, "Pin position assignment for movable pins in macro-cells," *Int. J. Computer-Aided VLSI Design*, vol. 2, pp. 239-250, 1990.



Jingsheng (Jason) Cong (S'88-M'90) received the B.S. degree in computer science from Peking University in 1985, and the M.S. and Ph.D. degrees in computer science from University of Illinois at Urbana-Champaign in 1987 and 1990, respectively.

Currently, he is an Assistant Professor in the Computer Science Department of University of California at Los Angeles. From 1986 to 1990, he was a Research Assistant in the Computer Science Department of the University of Illinois. He worked at Xerox Palo Alto Research Center in the summer of 1987. He worked at National Semiconductor Corporation in the summer of 1988. His research interests include computer-aided design of VLSI circuits, fault-tolerant design of VLSI systems, and design and analysis of efficient combinatorial and geometric algorithms.

Dr. Cong received the Best Graduate Award from Peking University in 1985. He received the Ross J. Martin Award for excellence in research from University of Illinois at Urbana-Champaign in 1989. He received the Student Research Paper Award from Sigma Xi, the scientific research society in University of Illinois at Urbana-Champaign in 1990. He is a member of ACM.