BUILDING BLOCK LAYOUT: ROUTING REGION DEFINITION AND ORDERING SCHEME

by

W.-M. Dai, T. Asano, and E. S. Kuh

Memorandum No. UCB/ERL M85/14

26 February 1985

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE:

Building Block Layout: Routing Region Definition and Ordering Scheme

AUTHORS:

Wei-ming Dai

Tetsuo Asano*

Ernest S. Kuh

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory,
University of California, Berkeley

Mailing Address:
Wei-Ming Dai
c/o Professor E.S. Kuh
Department of EECS,
University of California, Berkeley,
CA 94720, U.S.A.

*on leave from Osaka Electro-Communication University, Neyagawa, Osaka, Japan.

Building Block Layout: Routing Region Definition and Ordering Scheme

by

Wei-ming Dai
Tetsuo Asano*
Ernest S. Kuh

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory,
University of California, Berkeley

ABSTRACT

We present a new routing region definition and ordering (RRDO) scheme for building block layout. Given an arbitrary placement of rectangular blocks (including the case with cycles in the channel precedence constraints), without modifying the placement, our scheme defines and orders channels so that when a new channel is being routed, its width can be expanded or contracted without destroying the previously routed channels. The cycles in the channel precedence constraints are broken by introducing a new kind of channels --- L-shaped channels. Unlike switchboxes, L-shaped channels can be expanded or contracted to permit the completion of routing without rerouting other existing channels. An efficient greedy RRDO algorithm has been implemented, which tries to generate as few L-shaped channels as possible since L-shaped channels are harder to route than straight channels. Our algorithm represents routing regions by a floor plan graph, for which we provide a precise definition and a construction algorithm. The experimental results of the RRDO algorithm are promising.

1. Introduction

The building block layout problem can be described as follows: place a set of circuit blocks with rectilinear boundaries (in this paper we assume rectangular boundaries) and arbitrary sizes, and route a set of interconnections among the blocks. Since it is an extremely difficult problem [26], we divide it into two subproblems to reduce its complexity: placement and routing. The placement phase is responsible for positioning the blocks on a layout surface while the routing phase is responsible for finding the specific interconnections of nets. To reduce the complexity further, we divide the routing phase into two stages --- global routing (loose routing), which finds global path of each net, and local routing (detailed routing, fine routing), which assigns the specific tracks to paths inside a routing region. Since the division of the problem is artificial, the subproblems are dependent on each other. From the experience of the Berkeley Building-block Layout system ([3]), we found that the organization of the subproblems to produce the globally optimal solution is more important than the optimality of the solution to each subproblem. Our routing region definition and ordering scheme considers the interaction between the placement and routing phases of a layout during the global and local routing stages.

The major contribution of this paper is to present a new scheme for the routing region definition and ordering which deals with arbitrary placements. Section 2 surveys the existing routing schemes. Section 3 provides the precise definition and construction algorithm for the floor plan graph, an underlying data structure for the new scheme. Section 4 presents the new scheme for routing definition and ordering. For the new scheme, an efficient greedy algorithm and exact definition of channel routing region are described in sections 5 and 6 and some special cases are discussed in section 7. Section 8 concludes paper by giving some remarks on the new scheme and the greedy algorithm.

2. Existing Routing Schemes

Search routers (grid routers[17] and line routers [8]) consider the routing area as one region, and route nets one at a time. Using these schemes, we know that a net routed later may be blocked from completion by nets routed earlier. Furthermore, the space complexity of such algorithms is high because all routing must be represented at once in memory.

The channel routing schemes ([6,5,1,30,25,2]) have less net order dependency and space complexity because they partition the routing area into a set of regions, called _channels_, and route one channel at a time.

Almost all the interesting channel routing problems have been proven to be NP-complete problems [28]. Furthermore, the upper bound (or worst-case performance) of the channel width guaranteed by a channel router is far from the theoretical lower bound width given by the channel's density. (Even though they perform well in some difficult examples [5]) For instance, Rivest [23] proposed a router with upper bound for channel width of 2*density-1. Therefore, we cannot accurately predict how wide a channel must be before the channel can be routed.

One alternative is to deliberately overestimate the channel size and try to reclaim the empty chip space after the local routing; this is very time-consuming. So channels must be adjusted during local routing. However, adjusting one channel will alter the neighboring channels. We call this well known phenomenon <u>channel interference</u>. Therefore, we need a <u>feasible routing order</u> for the channels. In a feasible routing order, when a new channel is being routed, all the pins along its two edges are fixed and its width can be expanded or contracted without destroying the previously routed channels.

Finding a feasible routing order is not a trivial problem because some placements result in cyclic channel precedence constraints. Even though cyclic channel precedence constraints had been discovered ten years ago [15], most proposed solutions attempt to prevent or remove cycles rather than face the issue directly. Cyclic constraints can be removed by perturbing the placement [4] or by shrinking the block shapes [16]. They can be prevented by restricting the acceptable placement [20]. These methods restrict the resulting placement to a special class of all possible placements, so-called slicing structures [20]. However, in many cases the placement possible with non-slicing structure is better than that possible with slicing structure, so we need to deal with cyclic channel precedence constraints. One way to handle the cyclic constraints is to estimate the width of one channel on the cycle and route it last [22]. Iterations are needed when the estimation is too small. Another way is to generate so called switchbox ([26,10]), routing regions with terminals on four sides. Thus all the adjacent channels of a switchbox need to be routed before the switchbox itself. If the switchbox routing fails because of insufficient routing area, forcing us to expand the region, some of the adjacent channels must be rerouted. Therefore, these schemes have no feasible routing order. The only known scheme which has a feasible routing order is the insertion of a special channel to break the cycle [9], which may lead to unnecessarily long wiring of some nets.

We propose a new routing scheme which not only generates efficient channels but also provides a feasible routing order. Such scheme is based on a floor plan graph, the data structure which represents the topological location of blocks and thus the routing regions.

3. Floor Plan Graph

We represent the topology of a placement with a floor plan graph. Similar graphs have been used in many papers ([27,14,29,7,16,12]), but to our best knowledge, neither precise definitions nor construction algorithms for such graphs have been explicitly given.

In this section, we define the tile planes and the floor plan graph. The floor plan graph is used as the underlying data structure for the routing region definition and ordering, while the tile planes are used to perform fast block movement operations for channel adjustments and propagate the corresponding topological changes to the floor plan graph.

The whole layout area is divided into rectangles referred to as tiles. There are two varieties of tiles: block tiles, which represent blocks, and space tiles, which represent empty space to be routed. The tiles may be implemented using corner stitching [OUSTER 84]. We define two tile planes: the horizontal plane, where all tiles are maximal horizontal strips, and the vertical tile plane, where all tiles are maximal vertical strips. The horizontal tile plane can be obtained by extending each horizontal edge of a block until some vertical edge of other block or the bounding box of the layout is encountered, as shown in Fig.1. (By the bounding box, we mean the smallest convex rectilinear box which encloses the blocks in a layout. In our scheme, we treat the surrounding routing regions separately). This process can be done by sorting horizontal edges of blocks in order and performing a so-called plane sweep method ([18,11,19]). (There is an analogous process for the vertical tile plane). Thus, the tile planes can be obtained in O (nlogn) time, where n is the number of blocks. The <u>interval</u> of a horizontal tile is the range of X-coordinates covered by the tile (for vertical tiles, Y-coordinates). A space tile is called <u>dominant</u> if the interval of the tile includes the intervals of all its adjacent space tiles. Non-dominant tiles are called <u>subordinate</u> (Fig.1).

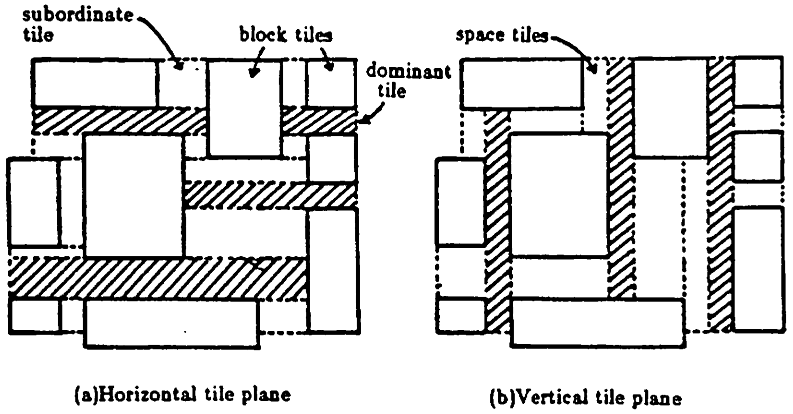


Fig. 1 Tile planes.

A floor plan graph can be efficiently derived from these tile planes (Fig.2). Corresponding to each horizontal or vertical dominant tile, we draw a <u>wall</u> for the floor plan graph. At each intersection of a horizontal and vertical dominant tile, we draw a <u>wall junction</u> connecting the tiles' walls. Because we assume all blocks, and thus all block tiles, are rectangular, every wall junction is formed by a 'T' shaped or '+' shaped intersection of the walls. Among wall junctions, those incident to only one wall segment are called <u>external</u>; others are called <u>internal</u>. We call a portion of a wall between two adjacent junctions a <u>wall segment</u>, and a region bounded by walls but containing no walls a <u>room</u>. Most rooms correspond to a block tile. Those rooms that do not are called <u>empty rooms</u>. (Fig. 2)

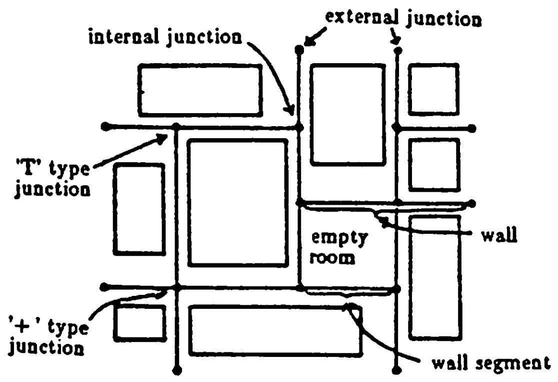


Fig. 2 Floor plan graph.

Looking at the above figures we observe correspondences between regions of routing area in the layout and the walls in the floor plan graph. Let regions of empty space bounded by two adjacent blocks in a layout be called <u>element channels</u>. The following theorem is obvious.

<u>Theorem 1</u>: Ignoring empty rooms, there is a one-to-one correspondence between element channels and wall segments. There is also a one-to-one correspondence between the intersecting areas of adjacent element channels and wall junctions. Thus all the empty space available for routing is represented by the floor plan graph.

4. A New Routing Scheme Using L-Shaped Channels

Channel interference is caused by the basic requirements of channel routing algorithms --- the channel rigidity requirement and the pin definition requirement. By channel rigidity requirement, we mean that after a channel has been routed, the relative positions of two edges of the channel can not be altered in the channel direction (parallel to the channel's edges) if we want to preserve the existing routing in the channel (Fig.3). By pin definition requirement, we mean that the positions of all pins along the two edges of a channel must be fixed (Fig.4).

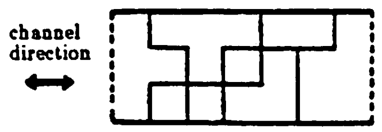


Fig. 3 Rigidity requirement: A channel already routed cannot be altered in its channel direction.

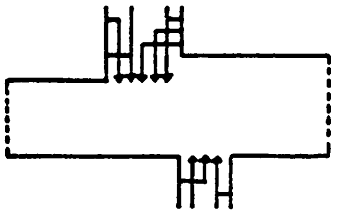


Fig. 4 Pin definition requirement: The positions of all pins along the two edges of a channel must be fixed.

Channel interference occurs at channel intersections. Consider two channels that intersect in a 'T' shape: the channel on the base of the 'T' intersection must be routed before the channel on the crosspiece of the 'T' intersection. (We will show that the '+' type intersections can be transformed into 'T' type intersections later.)

Since channels correspond to walls in the floor plan graph, we represent channel interference by a precedence relation on a set of walls in the floor plan graph.

<u>Definition</u>: (<u>wall precedence relation</u>) Let W be a set of walls. A relation (->) on W is a wall precedence relation if for any wi, wj in W, wi->wj implies that the channel corresponding to wi must be routed before the channel corresponding to wj.

For any wi and wj in W, such that wi and wj form a 'T' type junction with wi as the base wall and wj as the cross wall, then wi must precede wj as mentioned earlier (Fig.5). In any other case no direct precedence relation exists.

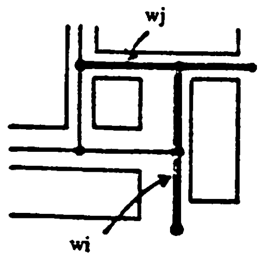


Fig. 5 Wall precedence relation (wi -> wj): The channel corresponding to wi must be routed before the channel corresponding to wj.

If the precedence relation contains no cycles, clearly it is a partial ordering relation. Any partial ordering relation can always be embedded into a total ordering relation, which gives the feasible routing order. So Theorem 2 follows.

Theorem 2: Given a floor plan graph, G, the whole routing region can be defined by a set of straight channels with a feasible routing order if and only if the wall precedence relations of walls of G contains no cycle.

We introduce a new concept -- the L-shaped channel -- to break cycles in the wall precedence relation. Assume wi -> wj belongs to some cycles in the relation, such that wi and wj form a 'T' junction. Also assume that each of wi and wj has one external junction, implying each of them has only one successor in the relation. Instead of defining two channels corresponding to wi and wj, we divide wj into two walls, wj' and wj", joining one portion of wj with wi as a corner (denoted as wj"+wi) to define an L-shaped channel. The remaining portion of wj defines a straight channel (Fig.6). Because of the channel rigidity and pin definition requirements, the precedence relation between these two newly defined walls becomes wj' -> wj"+wi. Since wi has only one successor before this division, the cycles involving wi in the relation have been broken.

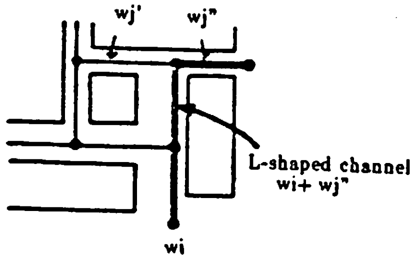


Fig. 6 L-shaped channel approach to breaking cyclic channel constraints.

Note that the feasible routing order exists after introducing L-shaped channels: after the straight channel corresponding to wj' has been routed, we can adjust the width of the L-shaped channel corresponding to wj"+wi in two dimensions (Fig.7). So we have a feasible routing order for L-shaped channels and straight channels while we have no such order for switchboxes and straight channels.

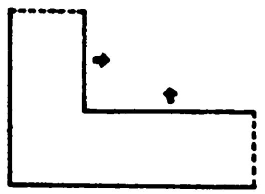


Fig. 7 An L-shaped channel may be expanded in these two directions without destroying previously routed channels.

5. A Greedy Algorithm for RRDO

5.1 Underlying Idea

In this section, we propose a greedy algorithm for the routing region definition and ordering. The algorithm takes a floor plan graph as the input and produces as output a set of channel definitions in terms of walls, with a feasible routing order.

This algorithm is a variation of the classic selection sort. We repeatedly determine and define the channel which must be routed after all other undefined channels, pushing this channel onto a channel definition stack. Popping the channels from the stack reverses their order, defining a feasible routing order.

The basic operations in the RRDO algorithm are wall slicing and corner cutting (Fig.8). Initially all the exterior junctions in the floor plan graph are marked as external. Whenever there exists a wall with two external junctions, we bisect the graph along the wall into two parts by marking all the junctions on the wall as external. Such operation is called wall slicing. When the wall slicing operation is not applicable, indicating cyclic constraints in the wall precedence relation, we bisect the graph along an L-shaped wall with two external junctions by marking all the junctions on the pair of walls which form the L-shaped wall as external. Such an operation is called corner cutting and a wall junction at which a corner cutting operation is applied is called a corner. A wall slicing operation defines a straight channel while a corner cutting operation defines an L-shaped channel (Fig.8).

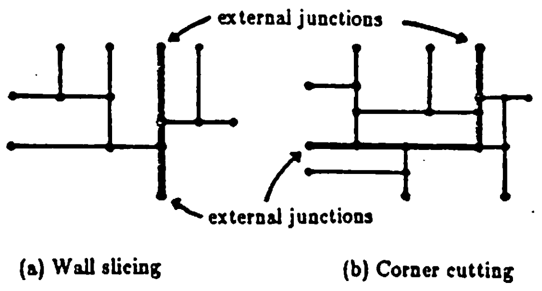


Fig. 8 Two basic operations on the floor plan graph.

Since straight channels are easier to route than L-shaped channels, our algorithm tries to minimize the number of L-shaped channels by selecting a particular class of corner, the pertinent corner, to cut. If there is more than one corner that can be cut, our choice could affect number of L-shaped channels.

Let c1 and c2 be two corners, we say c1 is horizontally (vertically, resp.) dependent on c2 if the corner junction (the middle junction of the corner) of c1 is on the horizontal (vertical, resp.) part of the wall of c2. For example in Fig. 9, c1 is vertically dependent on c2 and c2 is horizontally dependent on c3. A corner which is not dependent on any other corner (either horizontally or vertically) is called a totally independent corner. As is easily seen, if c1 is horizontally (or vertically) dependent on c2, then cutting c2 will break all cycles which would be broken if we cut c1. Such corner dependencies are transitive (Fig.9). But all corner dependencies may form a cycle, and hence no totally independent corner exists. If totally independent corners exist, we define the pertinent corners to be these independent corners. Even if no totally independent corner exists, a horizontally or vertically independent corner does exist. In this case we arbitrarily pick such a corner and travel from it to the corner to the corner upon which it depends and from that corner to the corner upon which it depends, proceeding until we encounter a previously visited corner. This corner is obviously part of a cycle of dependent corners, so cutting it will have the effect of cutting all corners on its cycle and on the path we followed to reach the cycle. Even though we may visit a horizontally and vertically dependent corner, which depends on two corners, we may choose to travel to either corner and be assured of arriving at a cycle. If we did not arrive at a cycle, this would imply the existence of a totally independent corner, a contradiction.

For the efficient implementation of the RRDO algorithm, we maintain three lists for totally, horizontally, and vertically independent corners, respectively. These lists are updated whenever any new corner is found or any new corner dependency is detected. This guarantees the corner we cut is pertinent. The following theorem states the heuristic nature of the algorithm.

Theorem 3: The greedy RRDO algorithm produces an optimum solution if there is at most one totally independent corner whenever the corner cutting operation becomes necessary.

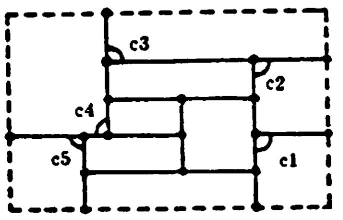


Fig. 9 Corner dependencies: c1 vertically depends on c2, c2 horizontally depends on c3, c3 vertically on c4, and c5 horizontally on c4. Only c4 is totally independent.

5.2 RRDO Algorithm

```
Routing_Region_Definition_And_Ordering

while there is an "internal" junction

        (Slice walls) As long as there exists a wall
        bounded by two external junctions, slice the wall,
        making junctions on it "external". Push the
        corresponding straight channel definition onto the
        channel definition stack.

        if there is still an "internal" wall junction,
            indicating cyclic wall precedence relations

            (Cut a pertinent corner)
            Update_Pertinent_Corner_List,
            and cut one of the pertinent corners, making
            junctions on it "external".
            Push the corresponding L-shaped channel definition
            onto the channel definition stack.
      end if

end while

Pop the channel definitions from the channel definition
stack, yielding the feasible routing order.


Update_Pertinent_Corners

for each external junction with an "unexplored" wall
```

```
    if the junction on the other end of the wall is a corner
       junction

       Remove pertinent corners on the base wall and the cross
       wall of the corner junction if any.

       Add the corner into the pertinent corner list.

       On the two walls which form the corner, starting from
       the corner junction, stopping at the first corner
       junction encountered or the end junction, apply
       Mark_Walls to every "unexplored" side walls.

    end if
end for


Mark_Wall (the_wall)

Mark the_wall "explored".

Remove pertinent corners on the_wall if any.

if one of the end junction of the_wall is external
   Apply Mark_Wall to every side walls on the_wall.
end if
```

5.3 Correctness and Complexity of the RRDO Algorithm

Theorem 4 (Correctness) The RRDO algorithm defines channels covering the whole routing space and provides a feasible channel routing order.

Proof. When the RRDO algorithm terminates, all wall junctions are marked "external", which indicates all the wall segments have been processed. By Theorem 1, the whole routing space is covered by the channels. Every time we perform a wall slicing or corner cutting operation to define a channel, the two end junctions are external. This implies that the first such operation divides the whole space into two parts, and the second operation divides one of the resulting parts into two parts, and so on. A channel routing order is given as the reverse of the order we push channels onto the stack. Thus, the first channel to be routed is between two blocks. If we merge them into one super block by routing the channel, the second channel is also between two blocks. In this way, according to the channel routing order obtained, we can merge two blocks associated with a channel into one super block by routing the channel. It is easily seen that once a channel is routed the result is kept, i.e., no rerouting process is needed. This guarantees the feasibility of the channel routing order.

Theorem 5 (Complexity) Both time and space complexity of the RRDO algorithm are linear in the number of blocks in the layout.

Proof. Since each wall segment is either processed by wall slicing or corner cutting, the total number of slicing and cutting operations performed is linear to the wall segments. Also we can identify a sliceable wall or a corner in constant time by keeping an external junction indicator for each wall. Other operations, Push, Pop, and Mark, are performed only once for each wall segment. Even though at each stage, the number of wall segments scanned to update the pertinent corner list is not constant, each wall segment is scanned at most once in the whole process. The space linearity is easy to verify because the floor plan graph can be represented by linked list data structures with a number of elements linear to the number of wall junctions. Also, the length of the pertinent corner list is bounded by the number of junctions in the graph. Since the floor plan graph is planar, the number of junctions or segments is linear to the number of rooms or block tiles.

The following figures (Fig.10) illustrate the greedy algorithm.

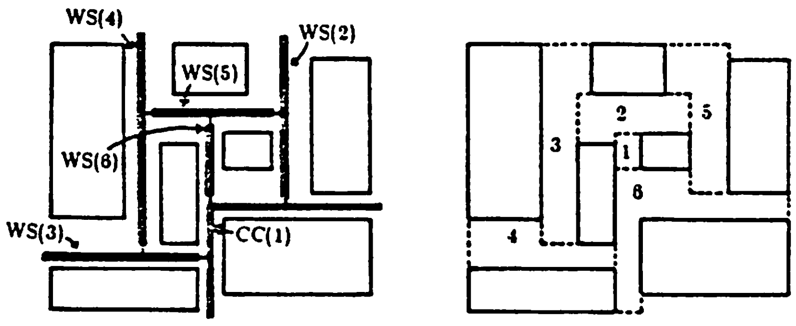


Fig. 10 An example of RRDO.
CC: corner cutting. WS: wall slicing

6. Exact Definition of Channel Routing Region

In this section we will give an exact definition of a channel routing region for an ordinary straight channel and an L-shaped channel.

Definition: (primitive straight channel) (1) A horizontal (vertical, resp.) primitive straight channel is a rectangular region with four sides. The left and right (upper and lower, resp.) sides -- referred to as open sides -- are fixed in location while the lengths are not fixed. (2) Terminals are placed on the upper and lower (left and right) sides. A set of nets going out of each open side is specified.

Examples are shown in Fig.11. We can generalize the above definition in the following way.

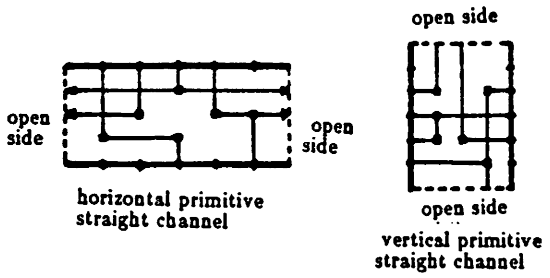

open side
open side
horizontal primitive straight channel
open side
open side
vertical primitive straight channel

Fig. 11 Primitive straight channels.

Definition: (horizontal straight channel)

(1) A horizontal straight channel is a rectilinear simple polygon with the boundary consisting of four distinctive

portions: left and right open sides; upper and lower boundaries.

(2) The open sides are vertical line segments. They are fixed in location while their lengths may be changed to complete the routing in the channel. The upper and lower boudaries are horizontally monotone, i.e., when we traverse them from left to right, the x-coordinates are non-decreasing and terminals are placed only on the horizontal line segments.

(3) A set of nets going out of the each open side is specified.

(4) There exists some horizontal line which intersects both of the open sides but does not intersect the upper and lower boundaries.

A vertical straight channel is defined in a similar manner. Fig.12 illustrates horizontal and vertical straight channels. For L-shaped channels, we have four different types shown in Fig.13.

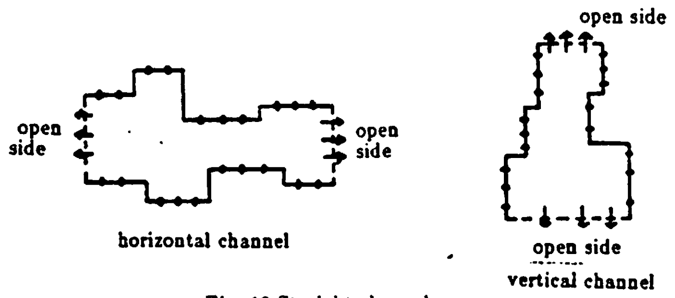


Fig. 12 Straight channels.

Definition: (L-shaped channel channel of type I)

(1) An L-shaped channel of type I is a rectilinear simple polygon with the boundary consisting of six distinctive portions: two open sides; the internal and external boundaries each of which consists of a horizontally monotone portion and a vertically monotone portion. (In Fig. 13, VI is a vertical monotone portion of the internal boundary and HE is a horizontally monotone portion of the external boundary.)

(2) We can move the internal boundary as long as it does not cross the external boundary. The lengths and locations of the open sides depend on the location of the internal boundary.

(3) Terminals are placed on the boundary.

(4) A set of nets going out of each open side is specified.

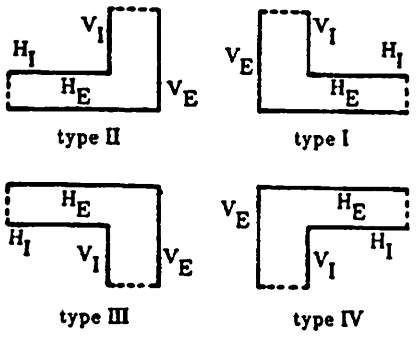


Fig. 13 Four types L-shaped channels.

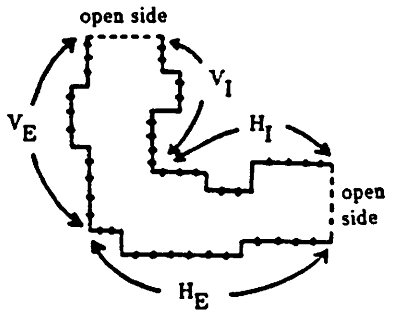
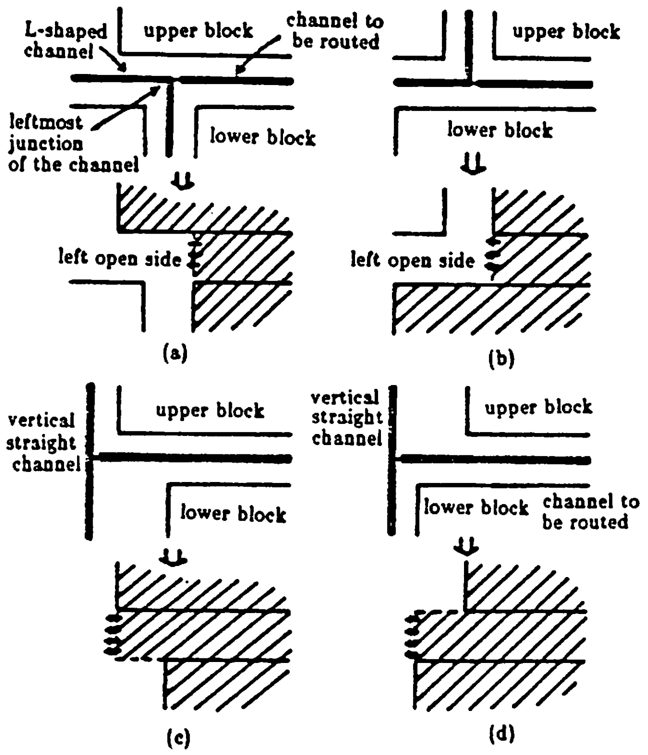


Fig. 14 An example of an L-shaped channel of type I.

Fig. 14 shows an example of an L-shaped channel of type I. L-shaped channels of other types are defined in a similar way. We now show how to define a channel routing region by means of wall segments. As mentioned before, in our RRDO algorithm each channel, which is represented by a set of wall segments, is picked up in the channel routing order given by the RRDO algorithm. After the channel is routed, the two blocks bounding the channel are merged into one super block to preserve the routing in the channel. The remaining decision in defining the region is the positioning of the open sides of the channel. Fig.15 illustrates four different situations at the left open side and the possible positions of the open side's boundary. In cases (a) and (b) of Fig. 15 where an L-shaped channel is adjacent to the channel to be routed, the left open side's boundary is an extension of the left edge of the inner block bounding the channel. In cases (c) and (d) where a straight channel is adjacent to the channel, the left open side's boundary is formed by the extensions of the vertical and horizontal edges of the blocks bounding the channel (see Fig. 15). Note that in the cases (c) and (d) terminals for those nets going out of the channel must be placed on the vertical part of the left open side's boundary (arrows indicate such terminals in Fig.15) to preserve the monotonicity requirement for channel definition.



(a) (b) The leftmost junction of the channel to be routed is a corner of an L-shaped channel.

(c) (d) The two wall segments incident to the leftmost junction of the channel are included in a vertical straight channel.

Fig. 15 Definition of the left open side of the channel to be routed.

7. Complications in the Floor Plan Graph -- '+' Type Junctions and Empty Rooms

7.1 + Type Junctions

In a floor plan graph, there may be '+' type junctions as well as 'T' type junctions. Since channels are disjoint routing regions, we can not define two channels corresponding to two walls which form a '+' type junction. One of walls has to be divided into two walls, with each of these new walls forming a 'T' type junction with the undivided wall. We call this process normalization of '+' type junction. For any wi and wj in W, such that wi and wj form a '+' type junction, we have the option to define channels in two ways (Fig.16):

(1) if we divide wi into two walls, wi' and wi",
then wi' -> wj and wi" -> wj.

(2) if we divide wj into two walls, wj' and wj",
then wj' -> wi and wj" -> wi.

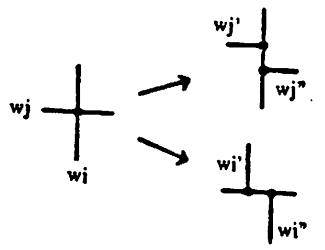


Fig. 16 Two choices in the normalization of a '+' type junction.

Note that the choices in the normalization of precedence relation and may result in producing a different number of L-shaped channels (Fig.17).

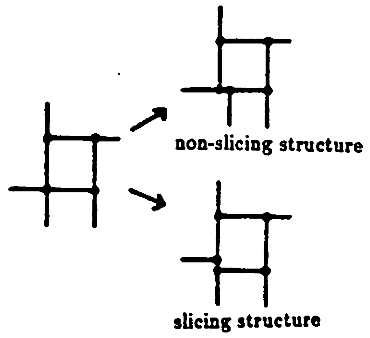


Fig. 17 Difficulty of normalization: an improper normalization of a '+' type junction leads to a non-optimal solution.

Kimura [16] proposed a heuristic algorithm to normalize a '+' type junction with as few cyclic precedence constraints as possible.

Observe that in our RRDO algorithm, a '+' type junction never provides additional cyclic precedence constraints. Initially we have no precedence relations between two walls which form a '+' type junction. In the RRDO algorithm, after one of the walls has been sliced and the corresponding channels pushed onto the channel definition stack, the '+' type junction is marked as external and the other wall is divided into two walls. The channels corresponding to these two walls will be defined and pushed onto the stack later. So the precedence relations are automatically satisfied. Note that we never specify any precedence relations between the two walls formed by normalizing the '+' junction. Hence the RRDO algorithm normalizes a '+' type junction optimally.

7.2 Empty Rooms

The existence of empty rooms violates the one-to-one correspondence between walls and routing regions. So we should ignore one of the wall segments bounding the empty room when we define channels corresponding to the walls of the room.

With minor modifications of the RRDO algorithm, the empty room problem can be handled as follows. Before RRDO, we identify the empty rooms in the floor plan graph. On each wall slicing or corner cutting operation, we ignore a wall segment if it is adjacent to an empty room whose remaining wall segments have been processed. Note that each empty room consists of exactly four wall segments. The following example illustrates this idea (Fig.18).

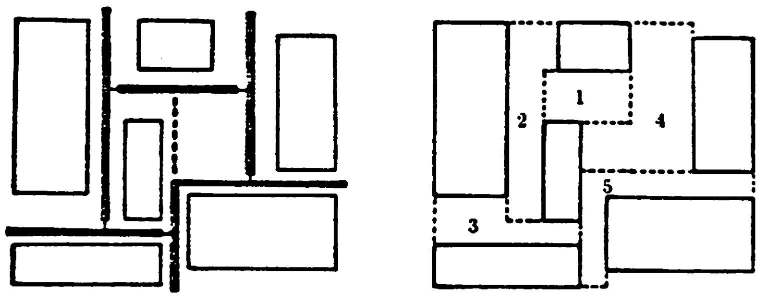


Fig. 18 Extension of RRDO algorithm for the empty room case: one of the four wall segments adjacent to an empty room is ignored.

8. Concluding Remarks

We conclude with some observations on the new scheme.

If we assume '+' type intersections in addition to 'T' type intersections in the floor plan graph, the 4-cycle Theorem proved by K. J. Supowit and E. A. Slutz [27] is no longer valid. The number of cycles, which need to be broken, in the channel precedence constraints could be exponential to the number of blocks. Therefore, we conjecture the problem of minimizing the number of L-shaped channels as an NP problem.

The greedy RRDO algorithm has been implemented in C on a Vax 11/780, and the results are promising.

We could modify our greedy RRDO algorithm by using some probabilistic methods: when there is more than one pertinent corner, we could randomize the choice of the pertinent corner to be cut.

The new RRDO scheme creates a new routing problem --- L-shaped channel routing. Since adjusting one portion of an L-shaped channel will alter the relative positions of two edges of the other portion of the channel, we will have no feasible routing order if we divide the L-shaped channel into two straight channels. So we should route an L-shaped channel as a whole. L-shaped channels are a special kind of two dimensional routing region because they have a special shape and two open ends (with no fixed pins).

Acknowledgement

The authors wish to thank Dr. Mallgorzata Marek-Sadowska for helpful discussions, and to Andrew Purshottam for helpful discussions and his assistance in constructing programs. The first author wishes to thank Bell Labs for research support.

References

[1] T. Asano, T. Kitahashi, K. Tanaka, H. Horino, and N. Amano, "A Wire-Routing Scheme Based on Trunk-division Methods", IEEE Trans. Comput., C-26, 8, pp.764-772, 1977.

[2] M. Burstein and R. Pelavin, "Hierarchical Channel Router", Integration, the VLSI Journal, April 1983.

[3] N. P. Chen, C. P. Hsu, and E. S. Kuh, "The Berkeley Building Block Layout System for VLSI Design", ERL memo. UCB/ERL M83/10, University of California at Berkeley, Feb. 1983, also VLSI 83 (Eds. F. Anceau and E.J. Aas), North-Holland, pp.37-84, August 1983.

[4] T. Chiba, N. Okuda, T. Kambe, I. Nishioka, and S. Kimura, "SHARPS: A Hierarchical Layout System for VLSI", Proc. 18th Design Automation Conference, pp.820-827, June 1981.

[5] D. N. Deutsch, "A Dogleg Channel Router", Proc. 13th Design Automation Workshop, pp.425-433, 1976.

[6] A. Hashimoto and J. Stevens, "Wire Routing by Optimizing Channel Assignment Large Apertures", Proc. 8th Design Automation Workshop, pp.155-169, 1971.

[7] J. E. Hassett, "Automated Layout in ASHLAR: An Approach to the Problems of 'General Cell' Layout for VLSI", Proc. 19th Design Automation Conference, pp.777-784, 1982.

[8] D. Hightower, "A Solution to the Line Routing Problem on the Continuous Plane", Proc. 6th Design Automation Workshop, pp.1-24, 1969.

[9] C. Horng and M. Lie, "An Automatic/Interactive Layout Planning System for Arbitrarily-sized Rectangular Building Blocks", Proc. 18th Design Automation Conference, pp.293-300, June 1981.

[10] C. P. Hsu, "A New Two-Dimensional Routing Algorithm", Proc. 19th Design Automation Conference, pp.46-50, 1982.

[11] H. Imai and T. Asano, "Dynamic Segment Intersection Search with Applications", Proc. 25th IEEE Symposium on Foundations of Computer Science, pp.393-402, October 1984.

[12] M. Ishii, N. Harada, S. Ido, M. Koyama, and T. Inoue, "A Channel Router Having Layer Assignment Capability for Arbitrary-sized Rectangular Building Blocks", Proc. International Conference on Computers and Communications, pp. 260-264, 1982

[13] Y. Kajitani, "Order of Channels for Safe Routing and Optimal Compaction of Routing Area", IEEE Trans. on Computer Aided Design, vol.CAD-2, no. 4, pp.293-300, 1983.

[14] K. Kani, H. Kawanishi, and A. Kishimoto, "ROBIN: A Building Block LSI Routing Program", Proc. Int. Symposium on Circuits and Systems, pp.658-661, 1976.

[15] H. Kawanishi, S. Goto, T. Oyamada, and K. Kani, "A Routing Method of Building Block LSI", Rec. 7th Asilomar Conference on Circuits, Systems and Computers, pp.119-123, 1973.

[16] S. Kimura, N. Kubo, T. Chiba, and I. Nishioka, "An Automatic Routing Scheme for General Cell LSI", IEEE Trans. on Computer Aided Design, vol.CAD-2, no.4, pp.285-292, 1983.

[17] C. Y. Lee, "An Algorithm for Path Connections and Its Applications", IRE Trans. on Electronic Computers, vol.EC-10, pp.346-365, Sept 1961.

[18] D. T. Lee and F. P. Preparata, "Computational Geometry: A Survey", IEEE Trans. on Computers, vol.C-33, no.12, pp.1072-1101, December 1984.

[19] T. Ohtsuki, M. Sato, M. Tachibana, and S. Torii, "Minimum Partitioning of Rectilinear Regions", Trans. of Information Processing Society of Japan, vol.24, pp.647-653, September 1983.

[20] R. H. Otten, "Automatic Floorplan Design", Research Report RC9656, IBM Thomas J. Watson Research Center, 1982.

[21] J. K. Ousterhout, G. T. Hamachi, R. N. Maya, W. S. Scott, and G. S. Taylor, "Magic: A VLSI Layout System", Proc. 21st Design Automation Conference, pp.152-159, 1984.

[22] B. T. Preas and W. M. vanCleemput, "Routing Algorithm for Hierarchical IC Layout", Proc. Int. Symposium on Circuits and Systems, pp.482-485, 1979.

[23] R. L. Rivest, A. E. Baratz, and G. Miller, "Provably Good Channel Routing Algorithm", VLSI Systems and Computations, ed. H.T. Kung, B. Sproull, and G. Steele, pp.143-152, Computer Science Press, Rockville,

Maryland, 1981.

[24] R. L. Rivest, "The 'PI' (Placement and Interconnection) System", Proc. 21st Design Automation Conference, pp.475-481, 1982.

[25] R. L. Rivest and C. M. Fiduccia, "A Greedy Channel Router", Proc. 19th Design Automation Conference, pp.418-424, 1982.

[26] J. Soukup, "Circuit Layout", Proceedings of the IEEE, vol.69, no.10, pp.1281-1304, 1981.

[27] K. Supowit and E. A. Slutz, "Placement Algorithms for Custom VLSI", Proc. 20th Design Automation Conference, pp.164-170, 1983.

[28] T. Szymanski, "Dogleg Channel Routing is NP-Complete", IEEE Trans. on Computer Aided Design, vol.CAD-4, no.1, pp.31-40, 1985.

[29] H. Th. Verheyen, R. Nouta, and P. Dewilde, "FLOORPLANNING, A New Placement and Routing Strategy", Proc. Int. Symposium on Circuits and Systems, pp.453-456, 1984.

[30] T. Yoshimura and E. S. Kuh, "Efficient Algorithm for Channel Routing", IEEE Trans. on Computer Aided Design, vol.CAD-1, no.1, pp.25-35, 1982.

LIST OF ALL FIGURES

CC: corner cutting. WS: wall slicing

Fig. 11 Primitive straight channels.

Fig. 12 Straight channels.

Fig. 13 Four types L-shaped channels.

Fig. 14 An example of an L-shaped channel of type I.

Fig. 15 Definition of the left open side of the channel to be routed.

(a) (b) The leftmost junction of the channel to be routed is a corner of an L-shaped channel.

(c) (d) The two wall segments incident to the leftmost junction are included in a vertical straight channel.

Fig. 16 Two choices in the normalization of a '+' type junction.

Fig. 17 Difficulty of normalization: an improper normalization of a '+' type junction leads to a non-optimal solution.

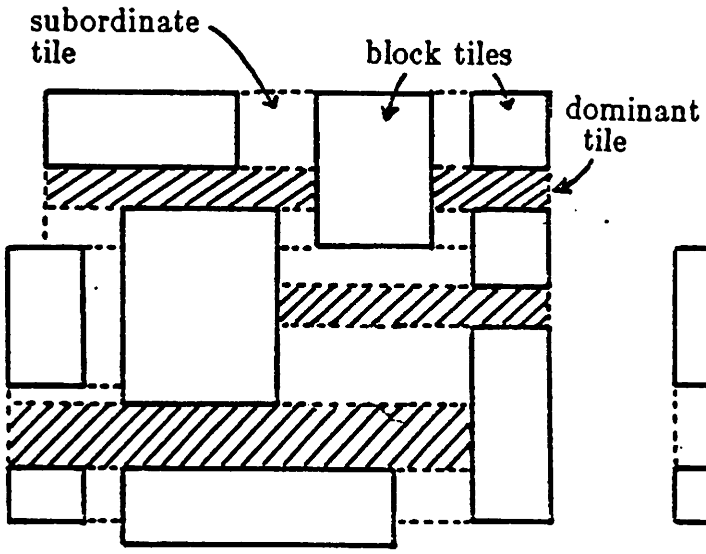
Fig. 18 Extension of RRDO algorithm for the empty room case: one of the four wall segments adjacent to an empty room is ignored.

List of all footnotes

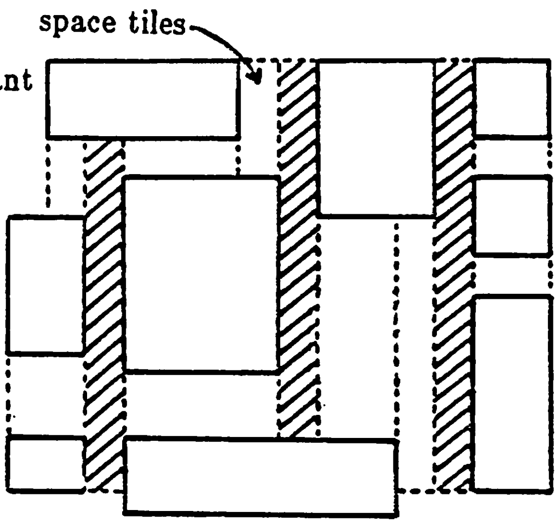
Manuscript received ______________________________.

*on leave from Osaka Electro-Communication University, Neyagawa, Osaka, Japan.

This work was supported by Semiconductor Research Corporation and NSF under Grants ECS-8201580.



(a)Horizontal tile plane



(b)Vertical tile plane

Fig. 1 Tile planes.

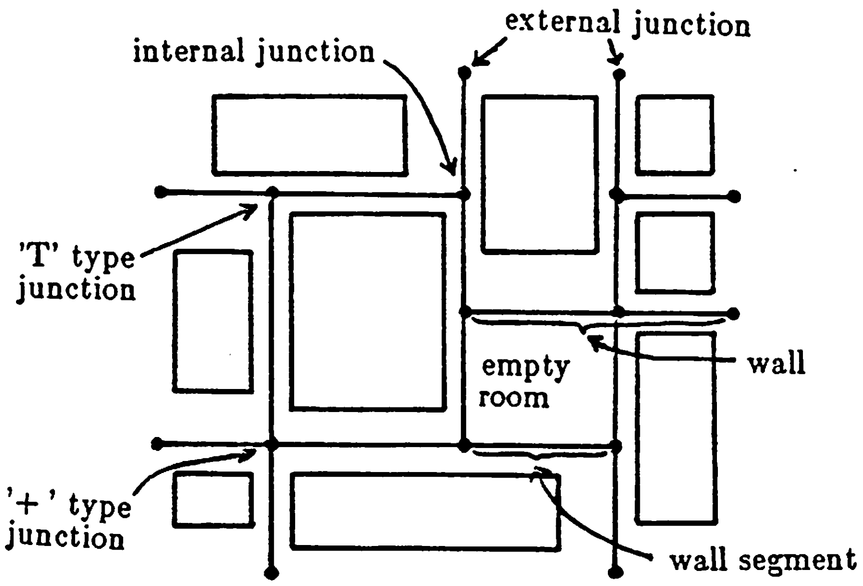


Fig. 2 Floor plan graph.

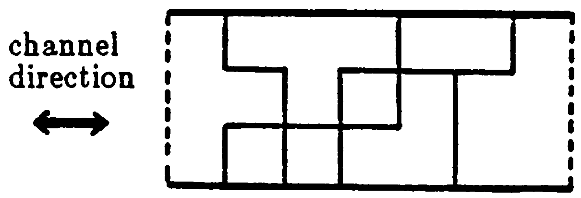


Fig. 3 Rigidity requirement: A channel already routed cannot be altered in its channel direction.

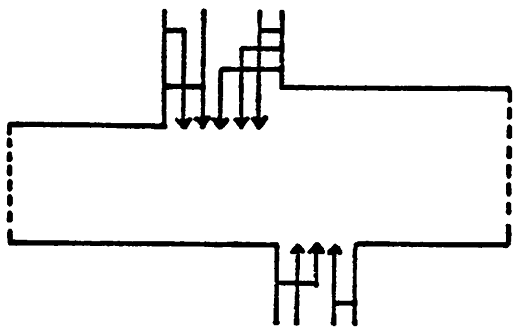


Fig. 4 Pin definition requirement: The positions of all pins along the two edges of a channel must be fixed.

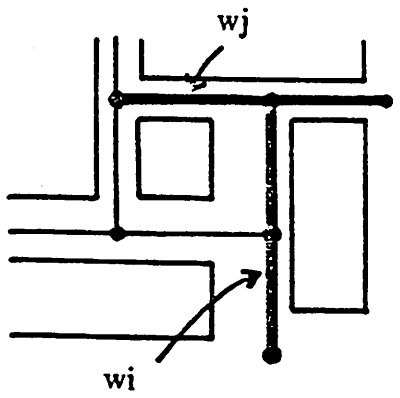


Fig. 5 Wall precedence relation (wi -> wj): The channel corresponding to wi must be routed before the channel corresponding to wj.

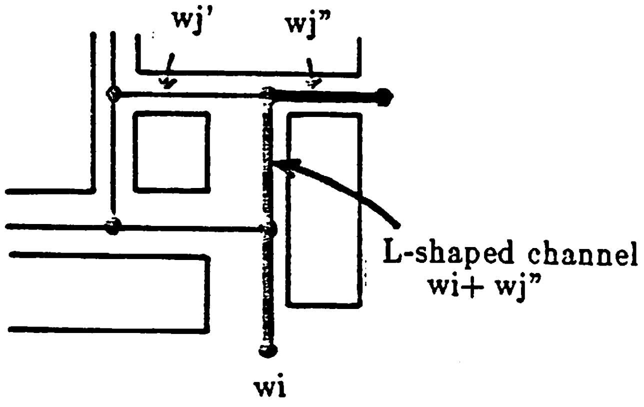


Fig. 6 L-shaped channel approach to breaking cyclic channel constraints.

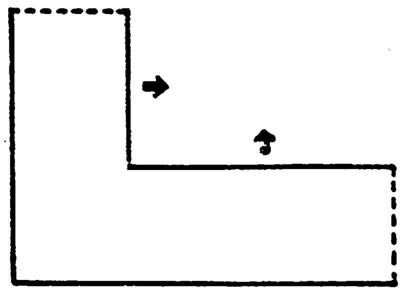


Fig. 7 An L-shaped channel may be expanded in these two directions without destroying previously routed channels.

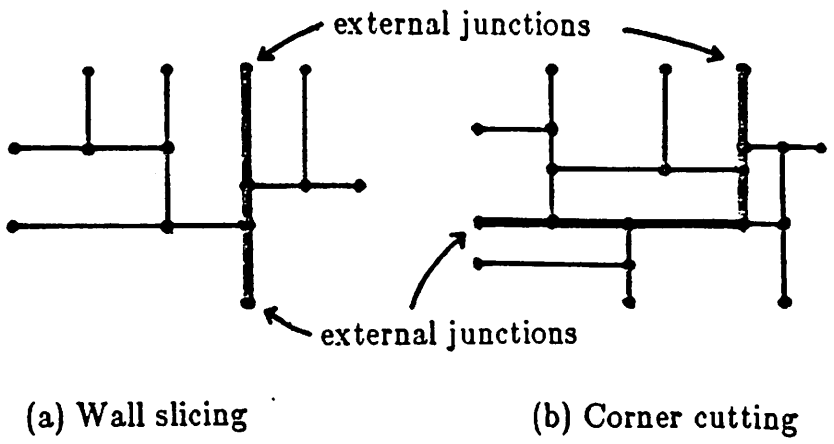


Fig. 8 Two basic operations on the floor plan graph.

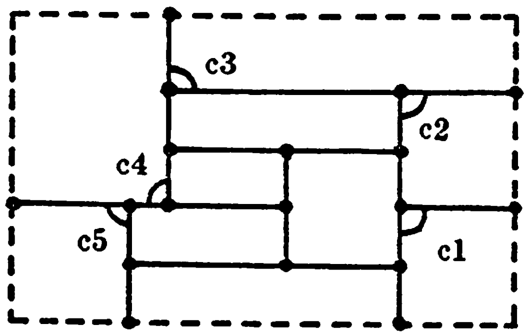


Fig. 9 Corner dependencies: c1 vertically depends on c2, c2 horizontally depends on c3, c3 vertically on c4, and c5 horizontally on c4. Only c4 is totally independent.

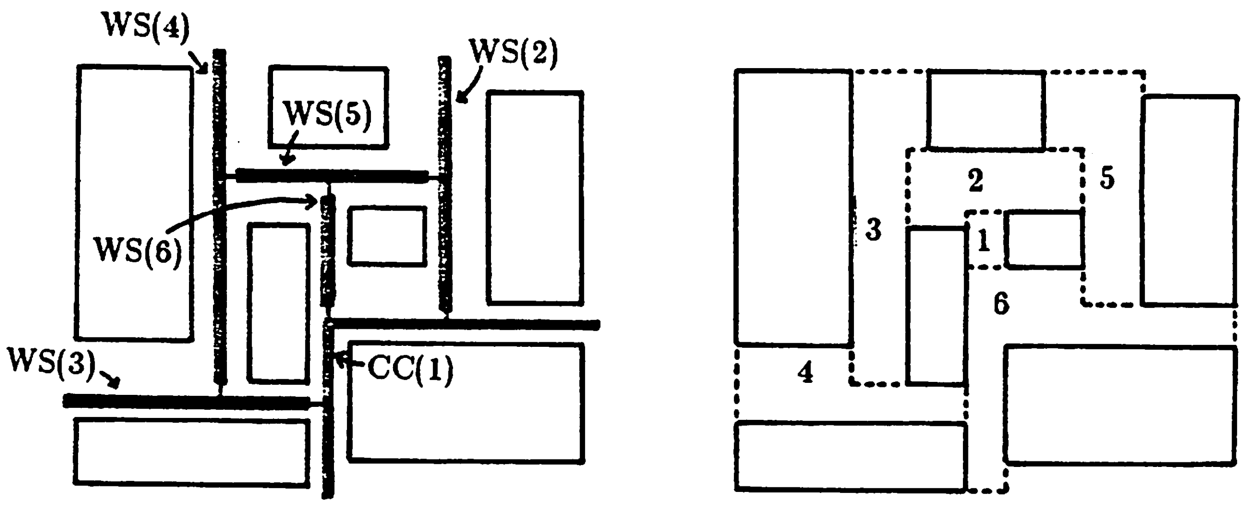
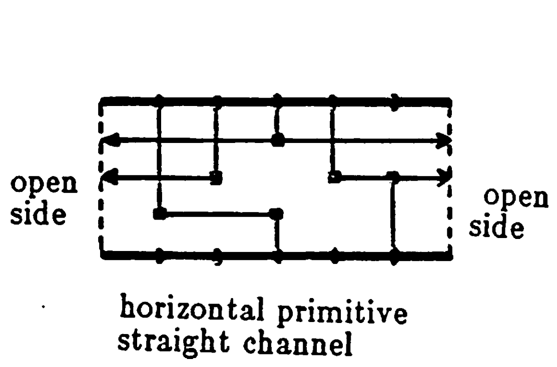


Fig. 10 An example of RRDO.
CC: corner cutting. WS: wall slicing



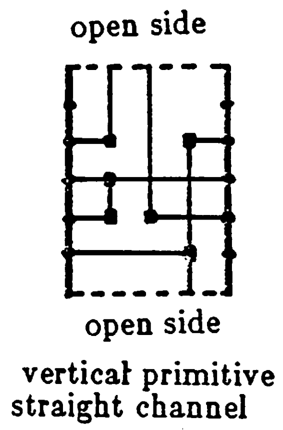
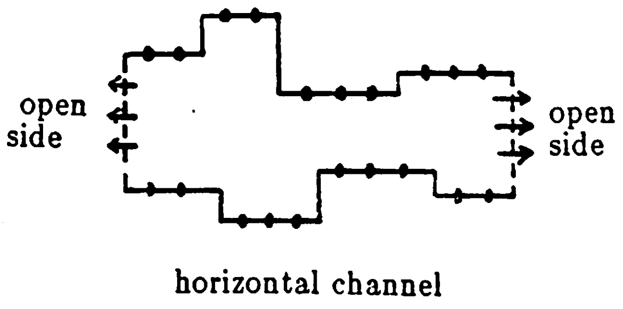


Fig. 11 Primitive straight channels.



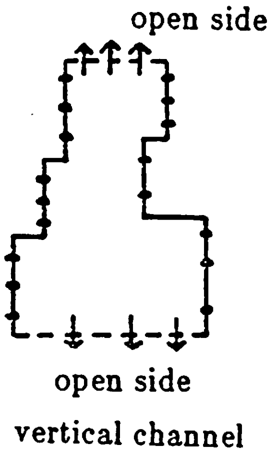


Fig. 12 Straight channels.

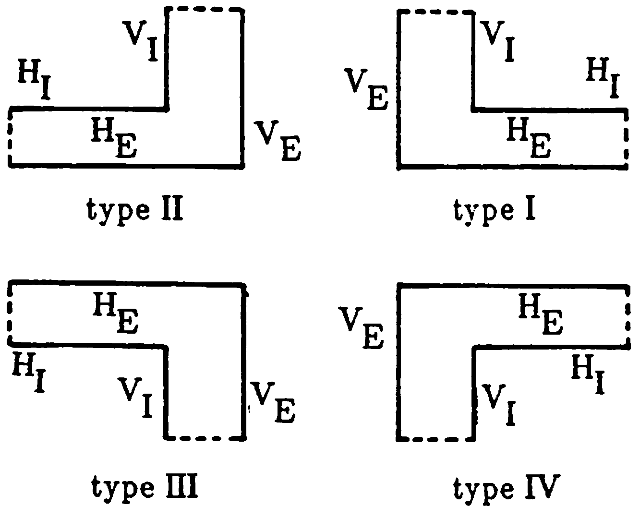


Fig. 13 Four types L-shaped channels.

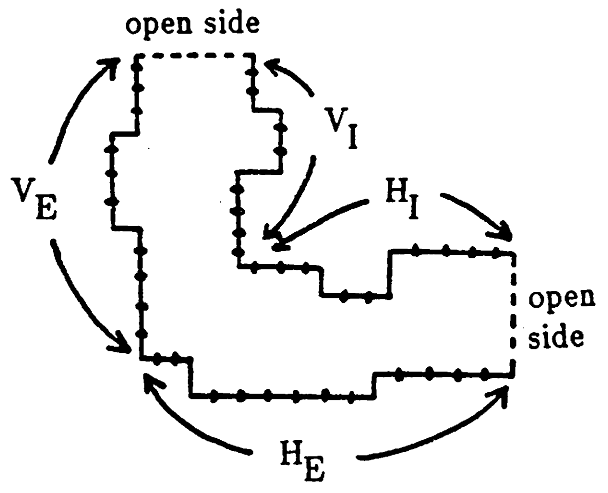
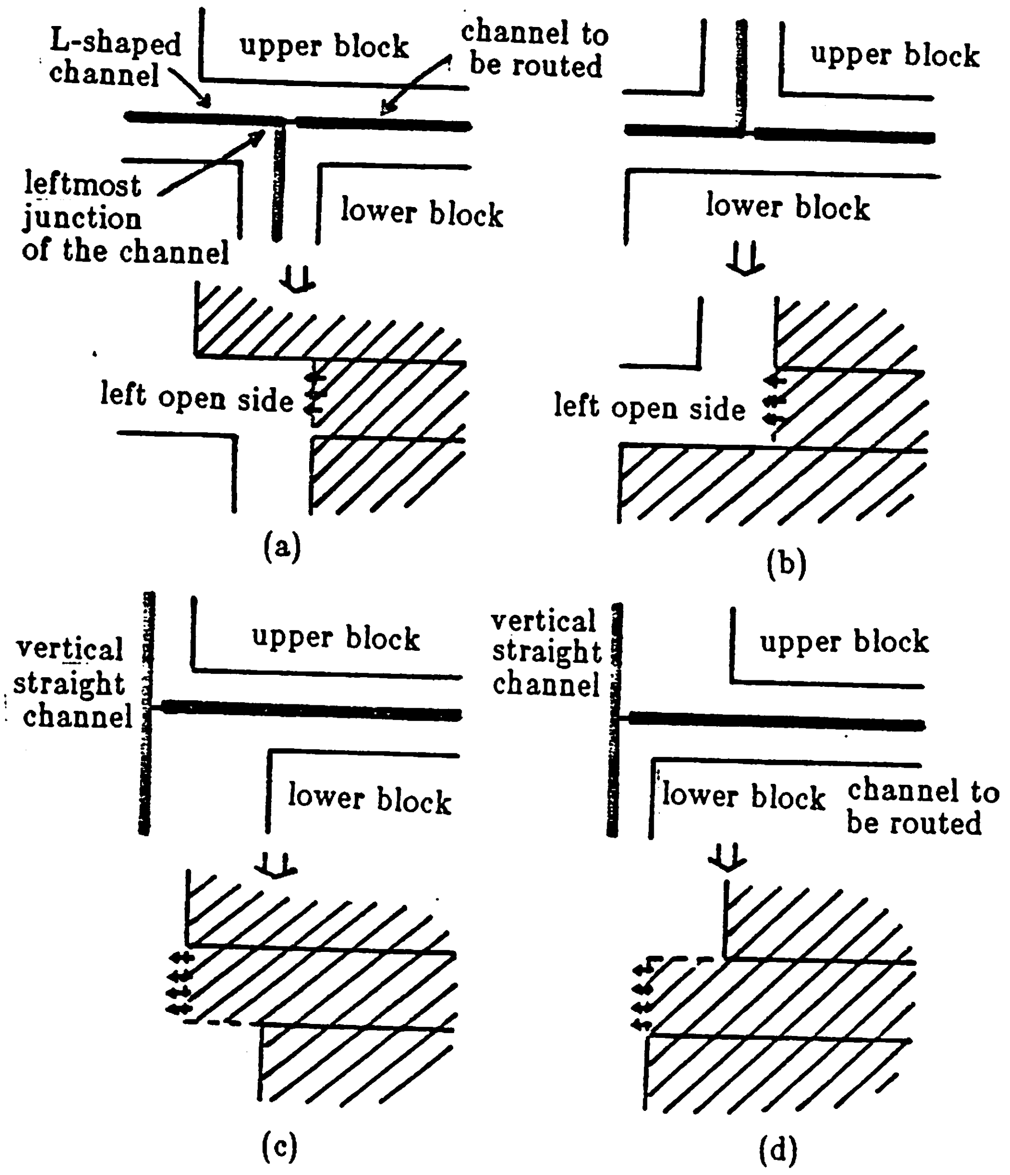


Fig. 14 An example of an L-shaped channel of type I.



(a) (b) The leftmost junction of the channel to be routed is a corner of an L-shaped channel.

(c) (d) The two wall segments incident to the leftmost junction of the channel are included in a vertical straight channel.

Fig. 15 Definition of the left open side of the channel to be routed.

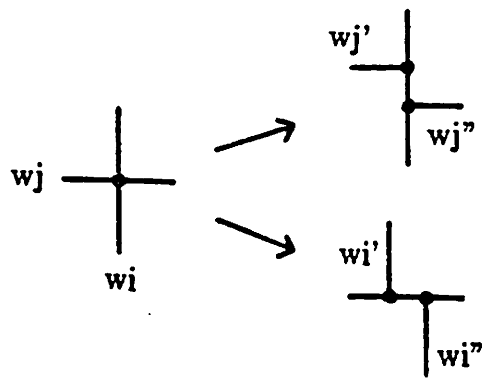


Fig. 16 Two choices in the normalization of a '+' type junction.

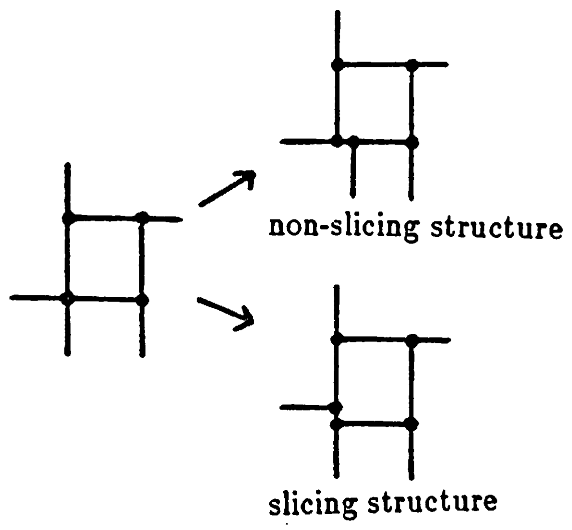


Fig. 17 Difficulty of normalization: an improper normalization of a '+' type junction leads to a non-optimal solution.

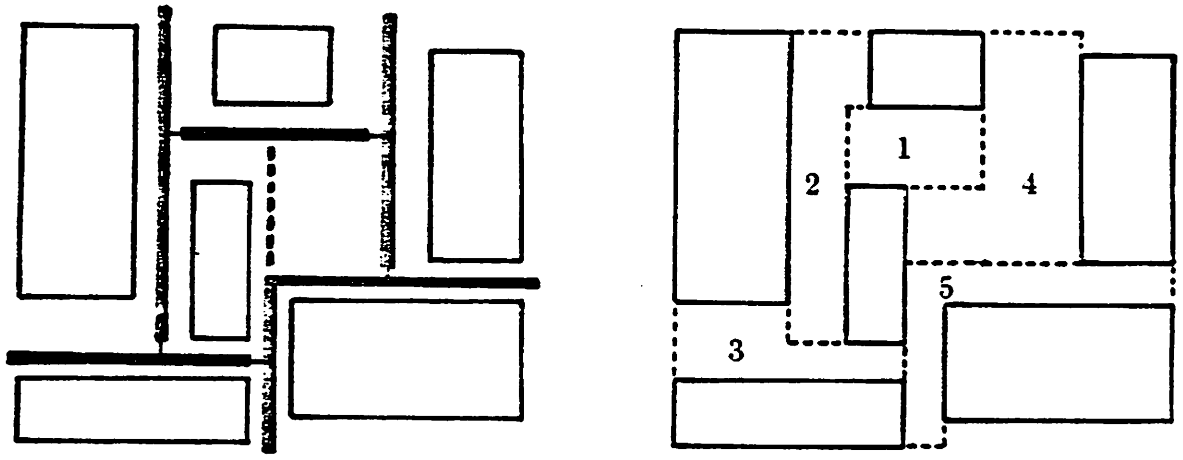


Fig. 18 Extension of RRDO algorithm for the empty room case: one of the four wall segments adjacent to an empty room is ignored.