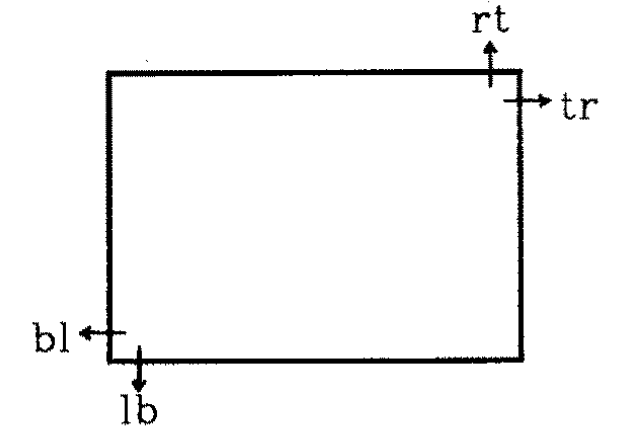
Linked List : rectangles (X

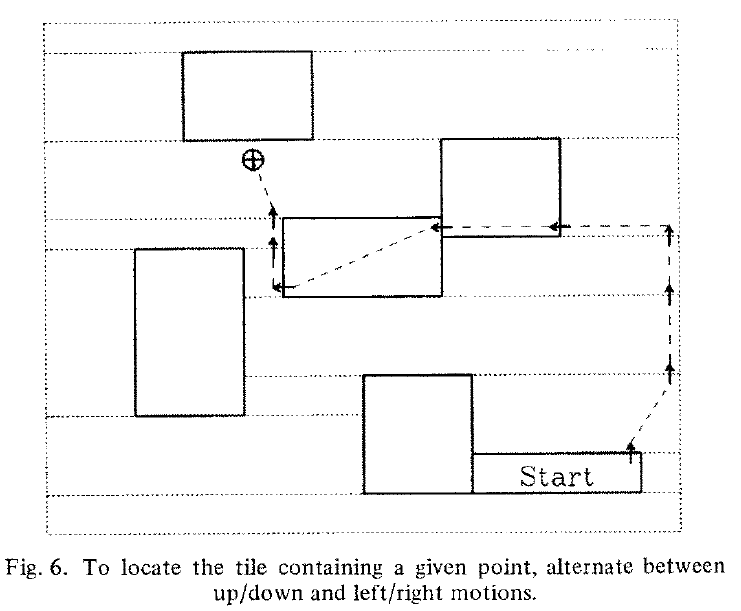
Bin : area of VLSI (X

Neighbor Pointers : indicate the horizontal and vertical adjacency (X

Corner stitching

Tile : 4 pointer of corner stitching (lower-left and upper-right)





Point Finding

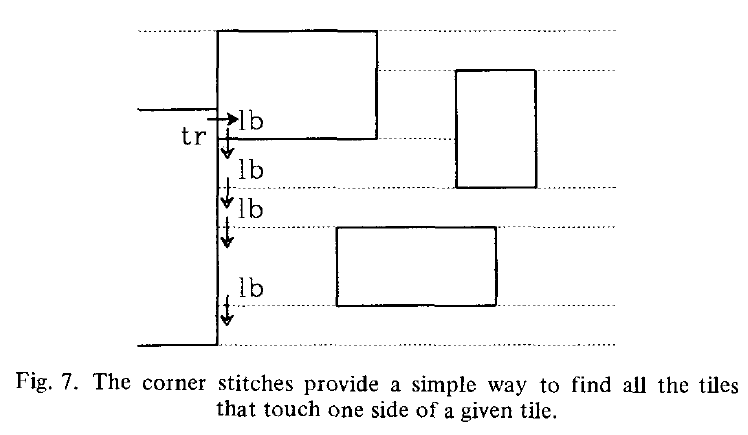
{

1.move up or down using **rt**(right top) and **lb**(left bottom) until a tile is found whose vertical range contains the desired point

2.move left or right using **tr** and **bl** until a tile is found whose horizontal range contains the desired point

3.iterate 1~2 to locate the tile containing the point

}



Neighbor Finding

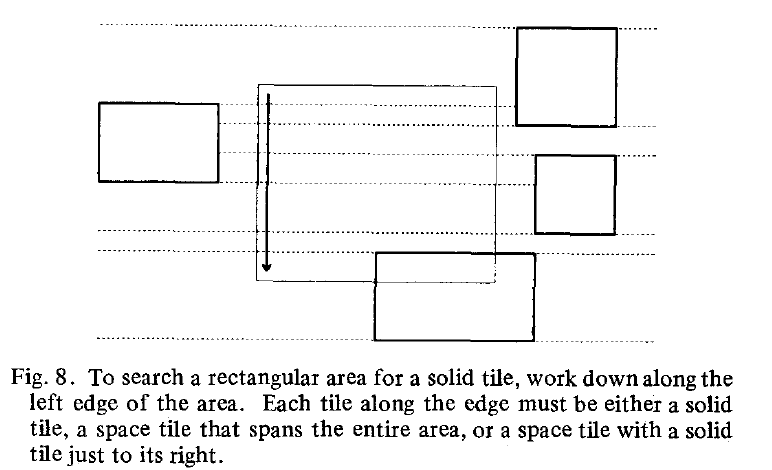
{

1. use **tr** of the stating tile to find its topmost right neighbor
2. trace down through **lb** until all the neighbors have been found

(the last neighbor is the first tile encountered whose lower y coordinate is less than or equal to the lower y coordinate of the starting tile)

}

**tilePtr->lb.LB.y <= tile.LB.y**



Area Search

{ // to find if there is any solid tile in the given area

1. use the **point-finding** algorithm to locate the tile containing the upper left corner of the area of interest.
2. See if the tile is solid

See if its right edge is within the area of interest

It is the edge of a solid tile

1. If the solid tile is found in step 2,then the search is complete.

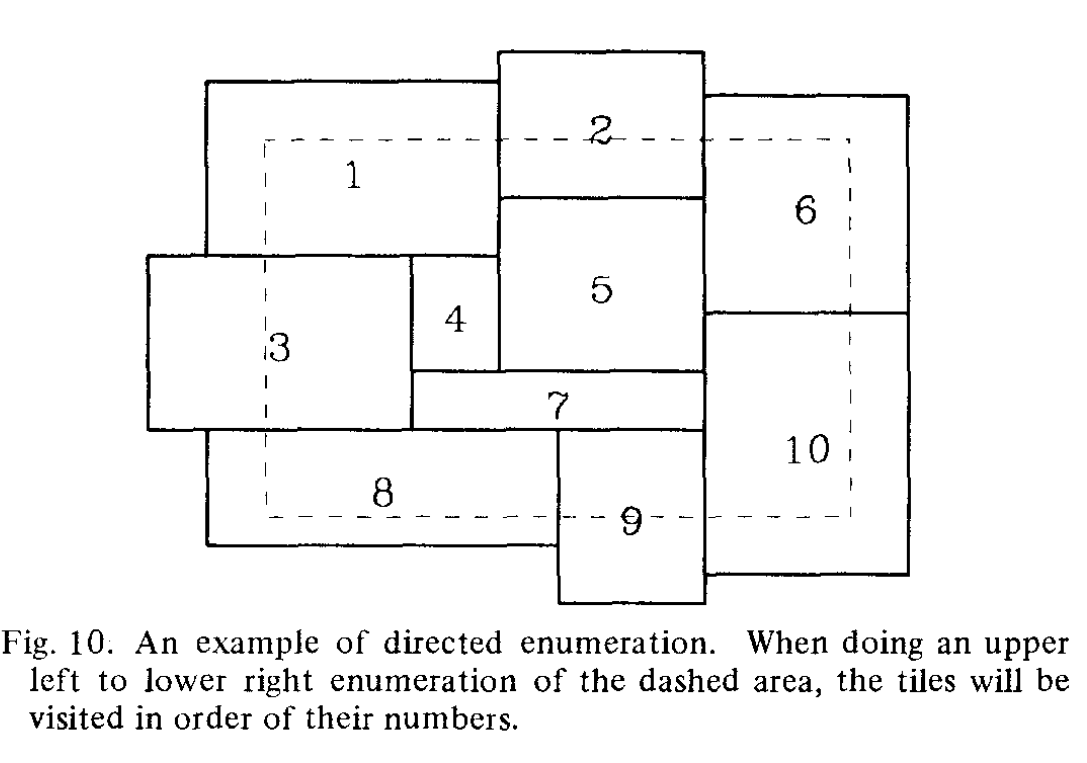
If not, move down to the next tile touching the right edge of the area of interest

(it can be down by point-finding or by traversing the **lb** stitch down and then traversing the **tr** stitch right

1. Repeat 2~3 until either a solid tile is found or the bottom of the area of interest is reached

}

O(height of the search area)

Directed Area Enumeration

{

1. As for the **area-search** algorithm, use the **point-finding** algorithm to locate the tile at the top left corner of the area of interest.

Then, step down through all the tiles along the left edge, using the same techniques as area-finding.

1. For each tile found in step 1, enumerate it recursively R procedure

R1) Enumerate the tile

R2) If the right edge of the tile is outside of the search area, then return from R

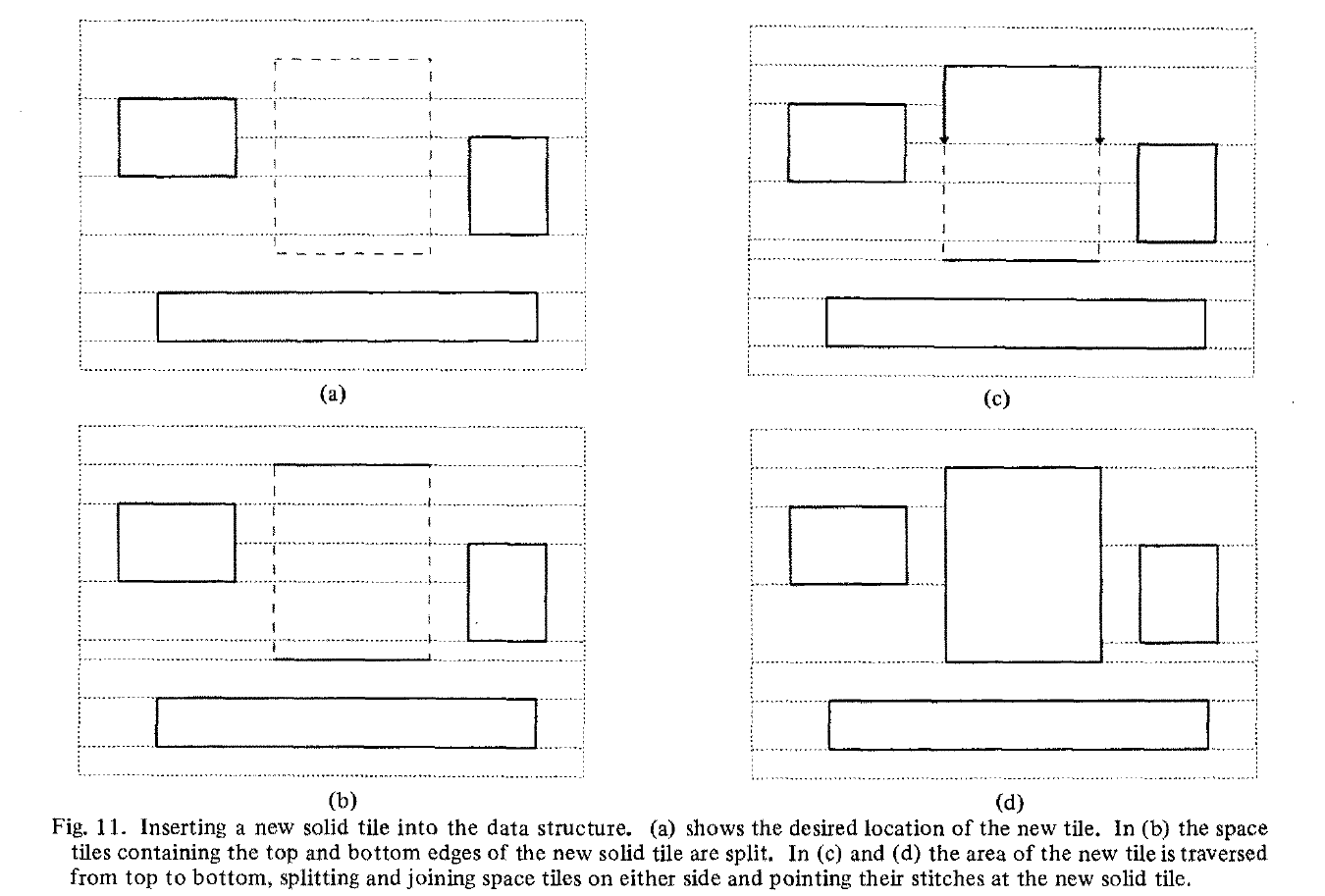
R3) Otherwise, using **neighbor-finding** algorithm to locate all the tiles that touch the right side of the current tile, and also intersect the search area.

R4) For each of these neighbors, if the bottom left corner of the neighbor touches the current tile then call R to enumerate the neighbor recursively (for example, this occurs in Fig. 10 when tile 1 is the current tile and tile 2 is the neighbor).

R5) Or, if the bottom edge of the search area cuts both the current tile and the neighbor, then call R to enumerate the neighbor recursively (in Fig. 10, this occurs when tile 8 is the current tile and tile 9 is the neighbor).

}

O(the numbers of tiles intersecting the search area)

Tile Creation

{

1. Check if there are no existing solid tiles in the desired area of the new tile.

Using **area-search** algorithm.

2. Insertion

Find the space tile containing the top edge of the area to be occupied by the new tile (because of the strip property, a single space tile must contain the entire edge).

Split the top space tile along a horizontal line into a piece entirely above the new tile and a piece overlapping the new tile. Update corner stitches in the tiles adjoining the new tile.

Find the space tile containing the bottom edge of the new solid tile, split it in the same fashion, and update stitches around it.

Work down along the left side of the area of the new tile, as for the **area-search** algorithm. Each tile along this edge must be a space tile that spans the entire width of the new solid tile.

Split the space tile into a piece entirely to the left of the new tile, a piece entirely to the right of the new tile, and a piece entirely within the new tile. This splitting may make it possible to merge the left and right remainders vertically with the tiles just above them: merge whenever possible.

Finally, merge the center space tile with the solid tile that is forming. Each split or merge requires stitches to be updated in adjoining tiles.

}

O(cost of splitting and merging the space tiles that cross the area)

Tile Deletion

{

1.Change the type of the dead tile to "space".

2.Use the **neighbor-finding** algorithm to search from top to bottom through all the tiles that adjoin the right edge of the dead tile.

3.For each space tile found in step 2), split either the neighbor or the dead tile, or both, so that the two tiles have the same vertical span, then merge the tiles together horizontally.

4.When the bottom edge of the original dead tile is reached, scan upwards along the left edge of the original dead tile to find all the space tiles that are left neighbors of the original dead tile.

5.For each space tile found in step 4), merge the space tile with the adjoining remains of the original dead tile. Do this by repeating steps 2)–3), treating the current space tile like the dead tile in steps 2)–3).

It is also necessary to do vertical merging in step 5). After each horizontal merge in step 5), check to see if the result tile can be merged just above and below it and merge if possible.

}