Optimal Two Shore Manhattan mode Channel Routing

Abstract:

The channel routing problem is NP complete, some heuristic algorithms are developed to solve the Channel Routing problem. This implementation of channel routing assigns wires track by track. Starting at the topmost track, it processes track by track downwards. The channel routing problem is completely characterized by two graphs: Vertical constraint graph and horizontal constraint graph. A comparison is made by solving the channel routing problem selecting the pins from left - edge and proceeding towards right step by step and selecting pins one from left and one from right alternatively. This algorithm is designed to pursue 100% routing completion.

Problem Studied :

In this project, channel routing problem was studied.Connections between terminal pins are to be routed within routing region called channel. Channel is usually a rectangular grid with pin locations on top and bottom boundaries.Manhattan model routing is used. The relative position of nets within the channel is modeled using Horizontal and Vertical constraint graphs. Tracks can be added as per requirement for routing and routing of every pin is guaranteed.

VLSI/CAD software used/ Engineering application:

This program can be used to minimize number of tracks required to complete routing of pins. The program is implemented in Python programming language.

Theoretical basis for the solution:

Assumptions:
Available routing layers : 2
One layer used only for horizontal routes
Other layer used for vertical routes.

Terminology:

a) Horizontal Constraint:
Horizontal Constraint is said to exist if horizontal segments two of nets overlap.

Representation of Horizontal Constraints:
Zone representation is used to represent horizontal constraints between net segments. An example of zone representation is shown in Fig. - 1.
No two nets in the same zone can share a track.

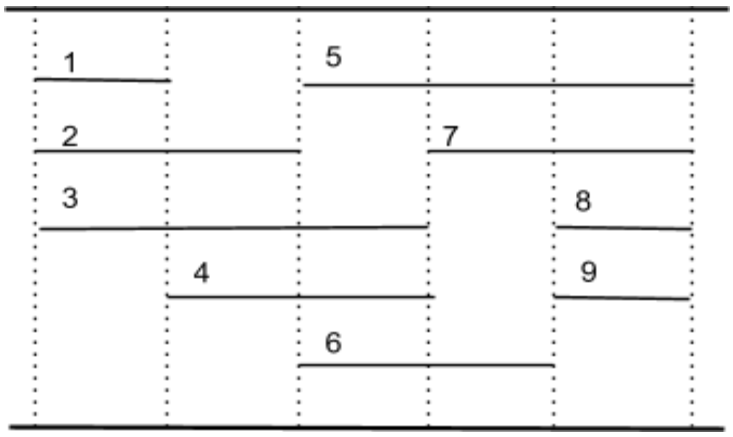

Fig - 1. Zone representation

A set $S(col)$ contains all nets that pass through column (col). In Fig - 1. Dotted lines represent columns. To avoid horizontal segments from overlapping, each net in $S(col)$ must be assigned to a different track.

In the zone representation shown in Fig. - 1. Net 1 cannot share track with nets 2 and 3 even if they do not have any vertical constraints.

Vertical Constraint:
Vertical Constraint is said to exist if two nets have pins in the same column.

Representation of Vertical Constraints:
Vertical constraints are represented by Vertical Constraint Graph $VCG(V,E)$.
Let node $v \in V$ represents a net.
A directed edge $e(i,j) \in E$ connects nodes i and j ,if net i must be located above net j.

An example of a VCG is shown in the Fig 2. below indicates that net 1 should be above net 2 and 3. Similarly net 3 should be above 4 and 5 and so on.

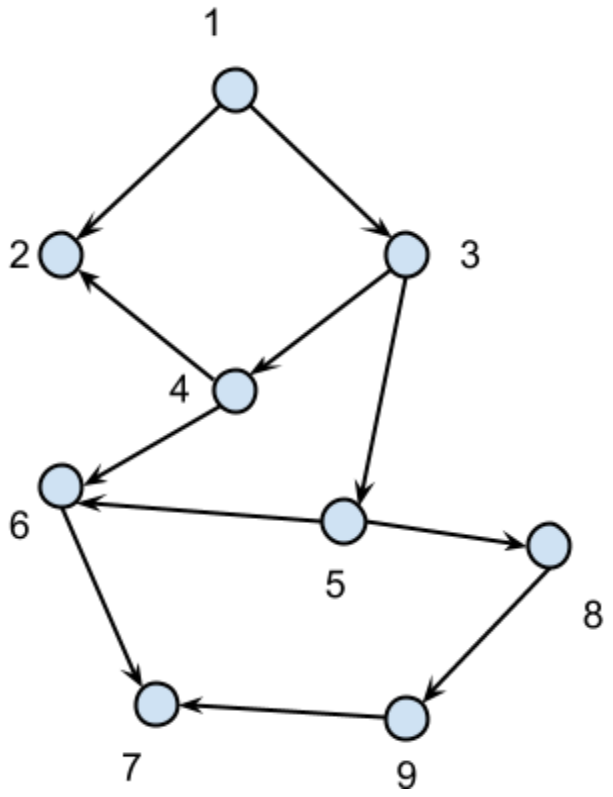

Fig - 2. Vertical Constraint Graph representing vertical constraints between pins.

Objective-
The objective of the channel routing problem is to connect the terminal pins by utilizing minimum number of tracks and respecting both horizontal and vertical constraints.

Implementation of Channel Routing-

Terminal pins are stored in two lists named top and bottom

Input file : Terminal pins on Top and Bottom of the channel

Inputs are stored in two lists : top [ ] and bottom [ ].

Constraint Generation.
Procedure:
The assignment of nets to tracks is modeled using Horizontal Constraints and Vertical Constraints.

- Generating Horizontal Constraints-

Step-1. Building $S(col)$:

Rightmost and leftmost position of every pin is stored in a data structure :
$pin[i].l$ stores leftmost position of $pin\ 'i'$.
$pin[i].r$ stores rightmost position of $pin\ 'i'$.
This information is used to find all nets in a particular column to build - $S(col)$.
$'Column'$ is a list of lists. i.e. $Column = [S(0), S(1), S(2)...]$

Step-2. Maximum Compatibles

$'Zone'$ is a list of lists that holds maximum compatibles. These are built by eliminating the subsets from the list $Column[\ ]$

Step-3. Zone representation :

$'Zone'$ consisting of maximal compatibles is used to find horizontal constraints.This constraint prevent nets from overlapping. No two nets in the same zone can be placed on the same track.
Lower bound on the number of tracks required to complete the routing can be found from zone representation.

- Generating Vertical Constraint Graph-

  Vertical constraints are stored in a data structure $cell[i].below$ which is a list of its parent pins. These constraints can be directly found by reading the top[ ] and bottom[ ] lists.

- Net modeling using the generated Horizontal constraints and Vertical Constraint Graph.

  1. Store left - to - right ordering (left-edge implementation) of all unassigned nets in a list named, $leftEdge[]$
  2. Store a list of nets unassigned in a list named $unassigned[\ ]$.
  3. Store a list of all nets without parents in a list named $parentless[\ ]$.
  4. For every pin that does not cause conflict on the VCG i.e for pin in list $parentless[\ ]$, scan through the zone representation list, for any horizontal conflict.
  5. If the net under consideration has no parent and does not cause a net overlap, assign net to current track.
  6. Clear the pin under consideration from Zone representation and Vertical Constraint graph.
  7. Clearing a pin from the VCG may create more parentless pins for the next iteration
  8. Clearing a net from zone representation removes the horizontal constraint enforced by the net which has already been assigned.
  9. Whenever conflict occurs move to next track.

RESULTS:

Following are the results after running this channel routing program on several examples to route terminal top and bottom pins:

Experiment 1 :

Running this program on an example taken from the publication - “Crossing-Aware Channel Routing for Integrated Optics” by Christopher Condrat, Priyank Kalla and Steve Blair.
For representing pin names, letters were replaced with integers from the above example.
INPUT:
Following the Left - Edge pin order:

top = [0,1,4,5,1,6,7,0,4,9,10,10]
bottom = [2,3,0,3,5,2,6,8,9,8,7,0]

OUTPUT:

zone representation is [[1, 2, 3, 4, 5], [2, 4, 6], [4, 6, 7], [4, 7, 8, 9], [7, 10]]
number of zones: 5
track 1 has nets[1, 10]
track 2 has nets[4]
track 3 has nets[5, 7]
track 4 has nets[3, 6, 9]
track 5 has nets[2, 8]
total tracks used 5

The routing as per the output provided by the program is shown in Fig-3.

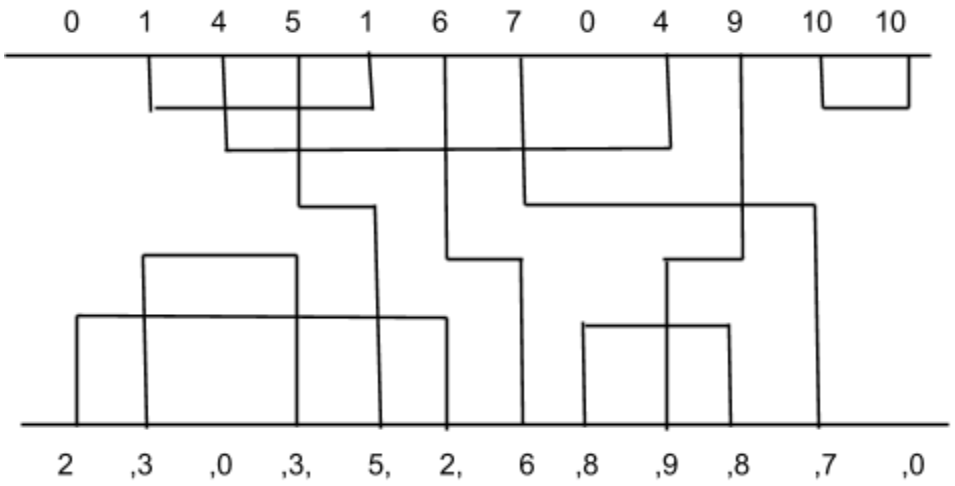

Fig-3. Routing completed in 5 tracks.

Experiment 2:

Repeating the above problem by Taking pin one from the left and one from the right:

INPUT:

top = [0,1,4,5,1,6,7,0,4,9,10,10]
bottom = [2,3,0,3,5,2,6,8,9,8,7,0]

OUTPUT:
zone representation is [[1, 2, 3, 4, 5], [2, 4, 6], [4, 6, 7], [4, 7, 8, 9], [7, 10]]
number of zones: 5
track 1 has nets[1, 10]
track 2 has nets[7, 5]
track 3 has nets[4]
track 4 has nets[9, 6, 3]
track 5 has nets[8, 2]
total tracks used 5

Experiment 3:

Following Left - Edge pin Order :
INPUT

top = [1,1,4,2,3,4,3,6,5,8,5,9]
bottom = [2,3,2,0,5,6,4,7,6,9,8,7]

OUTPUT

number of pins to route 9
zone is [[1, 2, 3], [2, 3, 4], [3, 4, 5, 6], [5, 6, 7], [5, 7, 8, 9]]
number of zones: 5
track 1 has nets[1]
track 2 has nets[3]
track 3 has nets[4]
track 4 has nets[2, 5]
track 5 has nets[6, 8]
track 6 has nets[9]
track 7 has nets[7]
total tracks used 7

Experiment 4:

Repeating the above problem but solving it taking one pin from the right and one pin from the left:

INPUT:

top = [1,1,4,2,3,4,3,6,5,8,5,9]
bottom = [2,3,2,0,5,6,4,7,6,9,8,7]

OUTPUT:

number of cells to route 9
zone is [[1, 2, 3], [2, 3, 4], [3, 4, 5, 6], [5, 6, 7], [5, 7, 8, 9]]
number of zones: 5
track 1 has nets[1]
track 2 has nets[3]
track 3 has nets[4]
track 4 has nets[2, 5]
track 5 has nets[8,6]
track 6 has nets[9]
track 7 has nets[7]
total tracks used 7

Observations:

For a given problem when the program was run with Left-Edge ordering of pins and then repeated by taking one from left and one from the right, no improvement in the quality of solution was observed for most examples.
Total number of tracks required to complete route remains unchanged.

Future Work:

Current program does not perform net splitting to tackle cyclic dependencies. If a cyclic dependency is detected it prints “Deadlock Detected” and terminates. I am currently working on designing my program to handle cyclic dependencies in the VCG and dogleg routing.