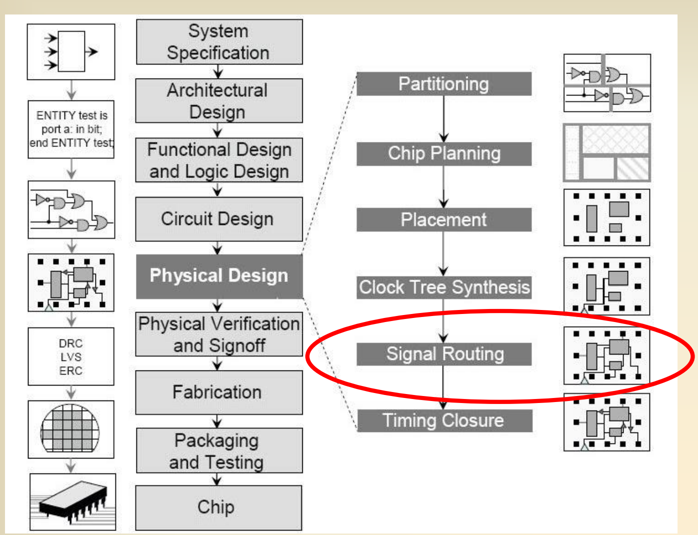
EE6094
CAD for VLSI Design

Chapter 9
Routing

Spring 2024
Andy, Yu-Guang Chen
Assistant Professor, Department of EE
National Central University
andyygchen@ee.ncu.edu.tw

Where are we now…



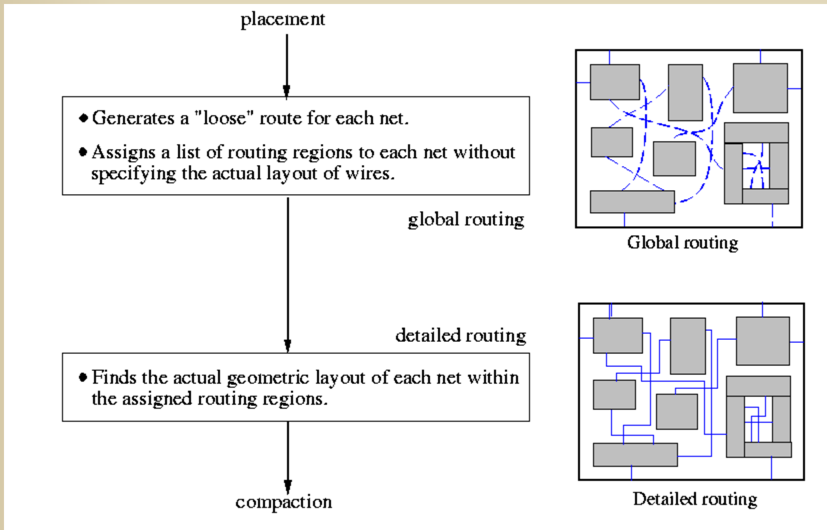
Routing Outline

- Routing an overview
- Global routing
  - Sequential Approach (Rip-up and Re-route)
    - Maze Routing
    - Line probing
    - Shortest Path Based Algorithms
    - Steiner Tree Based Algorithms
  - Concurrent Approach
    - Integer Programming
- Detailed routing
  - Channel routing

Routing Outline

- Routing an overview
- Global routing
  - Sequential Approach (Rip-up and Re-route)
    - Maze Routing
    - Line probing
    - Shortest Path Based Algorithms
    - Steiner Tree Based Algorithms
  - Concurrent Approach
    - Integer Programming
- Detailed routing
  - Channel routing

Routing

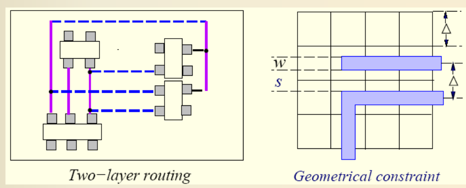


The Routing Problem

- Apply it after floorplanning/placement
- Input:
  - Netlist
  - Timing budget for, typically, critical nets
  - Locations of blocks and pins
- Output:
  - Geometric layouts of all nets
- Objective:
  - Minimize the total wirelength, the number of vias, or just completing all connections without increasing the chip area.
  - Each net meets its timing budget

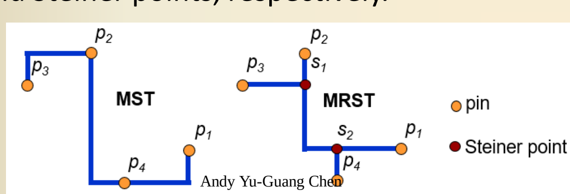
The Routing Constraints

- 100% routing completion + area minimization, under a set of constraints:
  - Placement constraint: usually based on fixed placement
  - Number of routing layers
  - Geometrical constraints: must satisfy design rules
  - Timing constraints (performance-driven routing): must satisfy delay constraints
  - Physical/Electrical/Manufacturing constraints:
    - Crosstalk
    - Process variations, yield, or lithography issues?



The Routing-Tree Problem

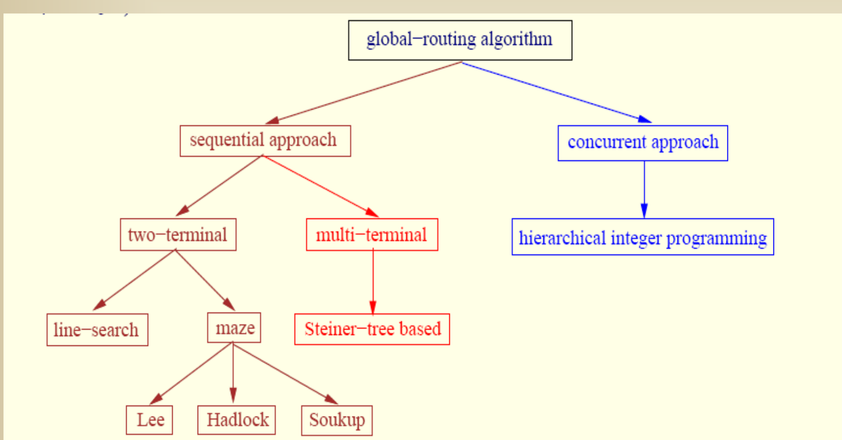
- **Problem:** Given a set of pins of a net, interconnect the pins by a "routing tree."
- **Minimum Spanning Tree (MST):** a minimum-length tree of edges connecting all the pins
- **Minimum Rectilinear Steiner Tree (MRST) Problem:** Given *n* points in the plane, find a minimum-length tree of rectilinear edges which connects the points. (Very useful in routing of VLSI circuits, but NP-hard)
- $MRST(P) = MST(P \cup S)$, where P and S are the sets of original points and Steiner points, respectively.



Routing Problem is Very Hard

- Minimum Spanning Tree Problem:
  - Use Prim's or Kruskal Algorithm
  - This is a P problem. Easy!
- Minimum Steiner Tree Problem:
  - Given a net, find the Steiner tree with the minimum length
  - This problem is NP-Complete!
- May need to route tens of thousands of nets simultaneously without overlapping
- Obstacles may exist in the routing region

Classification of Routing Algorithm



Approaches for Routing

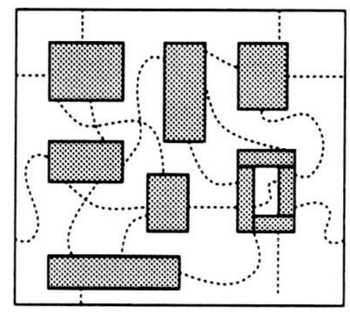
- **Sequential Approach:**
  - Route the nets one at a time
  - Order depends on factors like criticality, estimated wire length, and number of terminals
  - When further routing of nets is not possible because some nets are blocked by nets routed earlier, apply 'Rip-up and Reroute' technique (or 'Shove-aside' technique)
- **Concurrent Approach:**
  - Consider all nets simultaneously, i.e., no ordering
  - Can be formulated as integer programming

Extraction and Timing Analysis

- After global routing and detailed routing, information of the nets can be extracted and delays can be analyzed
- If some nets fail to meet their timing budget, detailed routing and/or global routing needs to be repeated

Global Routing

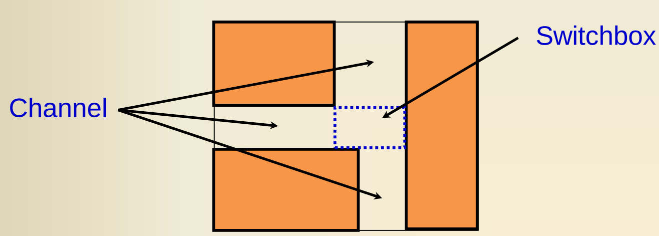
- Global routing is divided into 3 phases:
  - Region definition
  - Region assignment
  - Pin assignment to routing regions



Global Routing

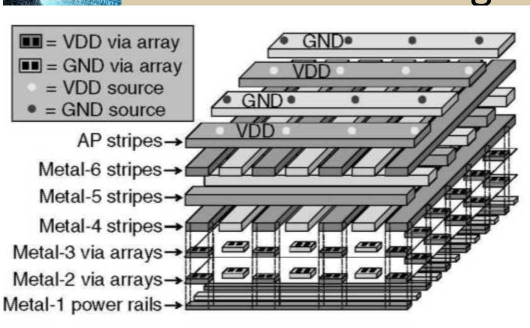
Region Definition

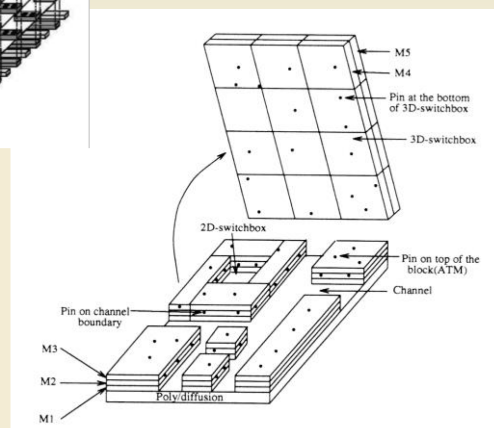
- Divide the routing area into routing regions of simple shape (rectangular):



- Channel: Pins on 2 opposite sides
- 2-D Switchbox: Pins on 4 sides
- 3-D Switchbox: Pins on all 6 sides

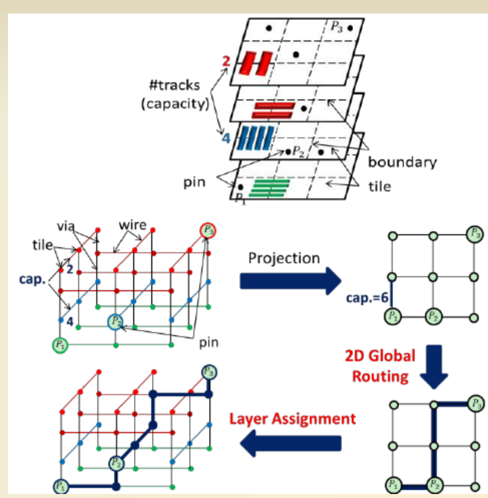
Routing Regions



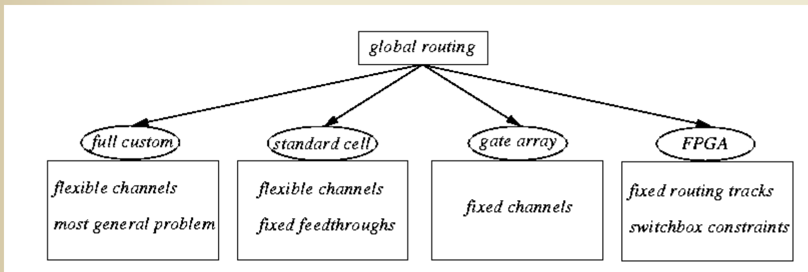


2D vs. 3D Global Routing

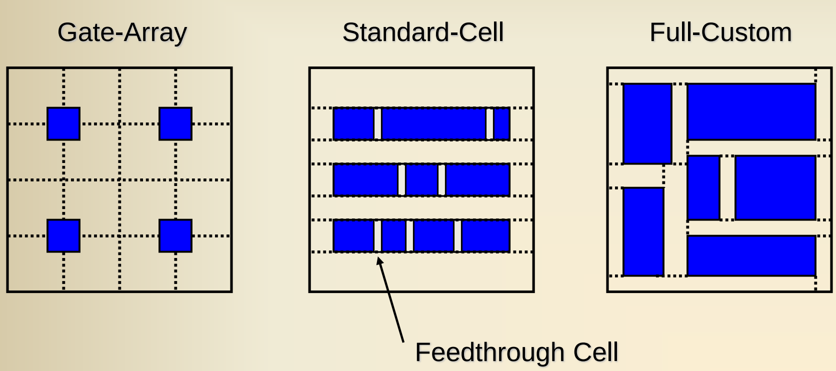
- 2-D global routing determines the global path for each net
- 3-D global routing additionally determines the used layer of every routing path



Global Routing in different Design Styles

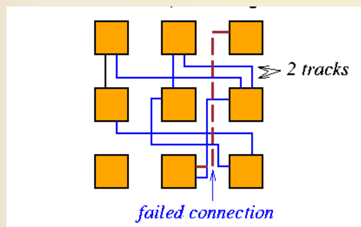


Routing Regions in Different Design Styles



Global Routing in Gate Array

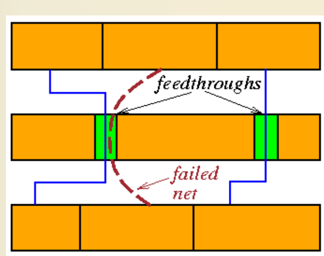
- Objective
  - **Guarantee 100% routability.**
- For high performance,
  - Minimize the maximum wire length.
  - Minimize the maximum path length.



Each channel has a capacity of 2 tracks.

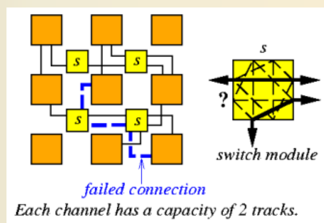
Global Routing in Standard Cell

- Objective
  - Minimize total channel height.
  - Assignment of **feedthrough**: Placement? Global routing?
- For high performance,
  - Minimize the maximum wire length.
  - Minimize the maximum path length.



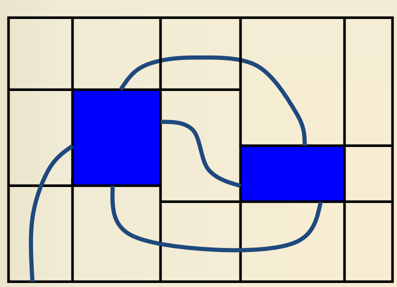
Global Routing in FPGA

- Objective
  - Guarantee 100% routability.
  - Consider **switch-module architectural constraints.**
- For performance-driven routing,
  - **Minimize # of switches used.**
  - Minimize the maximum wire length.
  - Minimize the maximum path length.



Region Assignment

- Assign routing regions to each net.
- Need to consider timing budget of nets and routing congestion of the regions

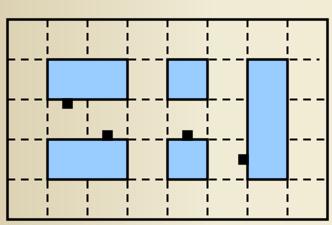


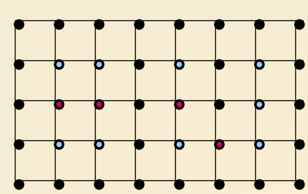
Graph Modeling of Routing Regions

- Grid Graph Modeling
- Checker Board Graph Modeling
- Channel Intersection Graph Modeling

Grid Graph

- Each cell is represented by a vertex.
- Two vertices are joined by an edge if the corresponding cells are adjacent to each other.
- The occupied cells are represented as filled circles, whereas the others are as clear circles.
- Given a 2-terminal net, the routing problem is to find a path between the corresponding vertices in the grid graph.





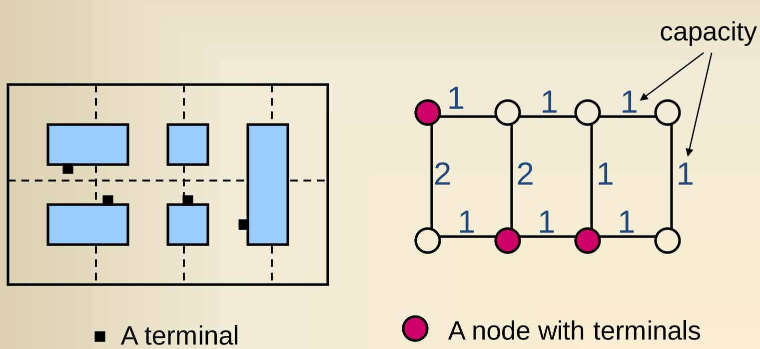
■ A terminal

• A node with terminals

Checker Board Graph

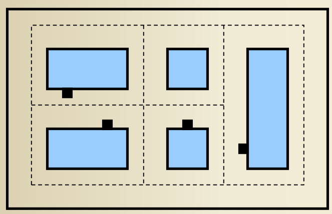
- More general than the grid graph model.
- Approximates the layout as a coarse grid.
- Checker board graph is generated in a manner similar to the grid graph.
- The edge capacities are computed based on the actual area available for routing on the cell boundary.
  - The partially blocked edges have a capacity of 1.
  - The unblocked edges have a capacity of 2.
- Given the cell numbers of all terminals of a net, the global routing problem is to find a path in the coarse grid graph.

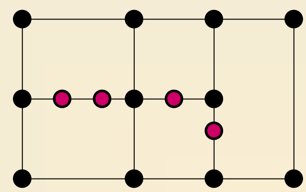
Checker Board Graph



Channel Intersection Graph

- Most general and accurate model for global routing.
- Channels are represented as edges.
- Channel intersections are represented as vertices.
- Edge weight represents channel capacity.
- Extended channel intersection graph: terminals are also represented as vertices.





■ A terminal

● A node with terminals

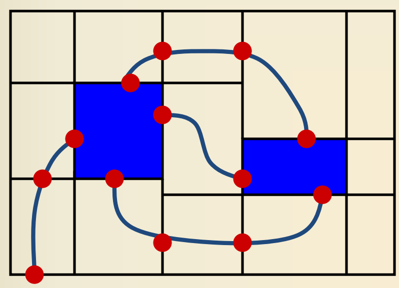
Routings along the channels

Channel Intersection Graph

- The global routing problem is simply to find a path in the channel intersection graph.
  - The capacities of the edges must not be violated.
  - For 2-terminal nets, we can consider the nets sequentially.
  - For multi-terminal nets, we can have an approximation to minimum Steiner tree.

Pin Assignment

- Assign pins on routing region boundaries for each net (Prepare for the detailed routing stage for each region)



Routing Outline

- Routing an overview
- Global routing
  - Sequential Approach (Rip-up and Re-route)
    - Maze Routing
    - Line probing
    - Shortest Path Based Algorithms
    - Steiner Tree Based Algorithms
  - Concurrent Approach
    - Integer Programming
- Detailed routing
  - Channel routing

Global Routing Approaches

- ◆ Sequential Approach (Rip-up and Re-route)
  - ➢ Maze Routing
  - ➢ Line probing
  - ➢ Shortest Path Based Algorithms
  - ➢ Steiner Tree Based Algorithms
- ◆ Concurrent Approach
  - ➢ Integer Programming

Global Routing Approaches

- ◆ Sequential Approach (Rip-up and Re-route)
  - ➢ Maze Routing
  - ➢ Line probing
  - ➢ Shortest Path Based Algorithms
  - ➢ Steiner Tree Based Algorithms
- ◆ Concurrent Approach
  - ➢ Integer Programming

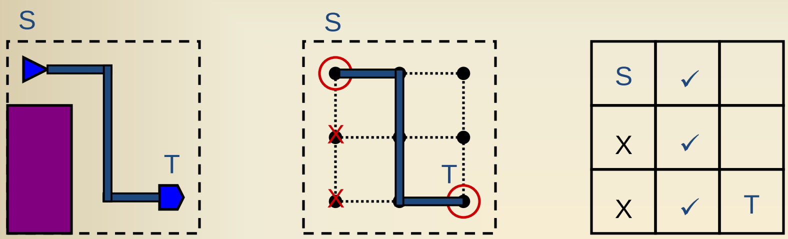
Maze Router: Lee Algorithm

- Lee, "An algorithm for path connection and its application," *IRE Trans. Electronic Computer*, EC-10, 1961.
- Discussion mainly on single-layer routing
- **Strengths**
  - Guarantee to find connection between 2 terminals if it exists.
  - Guarantee minimum path.
- **Weaknesses**
  - Requires large memory for dense layout.
  - Slow.
- Applications: global routing, detailed routing

Maze Routing Problem

- Given:
  - A planar rectangular grid graph
  - Two points S and T on the graph
  - Obstacles modeled as <u>blocked vertices</u>
- Objective:
  - Find the shortest path connecting S and T
- This technique can be used in global or detailed routing (switchbox) problems

Grid Graph

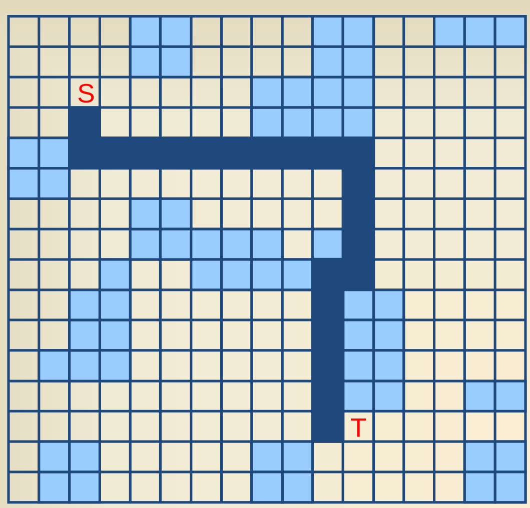


Area Routing

Grid Graph (Maze)

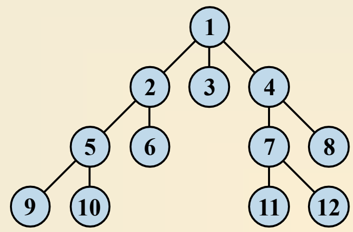
Simplified Representation

Maze Routing



Basic Idea

- A Breadth-First Search (BFS) of the grid graph
- Always find the shortest path possible
- Consists of two phases:
  - Wave Propagation
  - Retrace



An Illustration

S 0	1	2	3
1	2	3	
	3	4	5
5	4	5	T 6

Wave Propagation

- At step k, all vertices at Manhattan-distance k from S are labeled with k
- A Propagation List (FIFO) is used to keep track of the vertices to be considered next

S 0			
			T

After Step 0

S 0	1	2	3
1	2	3	
	3		
			T

After Step 3

S 0	1	2	3
1	2	3	
	3	4	5
5	4	5	T 6

After Step 6

Retrace

- Trace back the actual route
- Starting from *T*
- At *vertex with k, go to any vertex with label k*-1

S 0	1	2	3
1	2	3	
	3	4	5
5	4	5	T 6

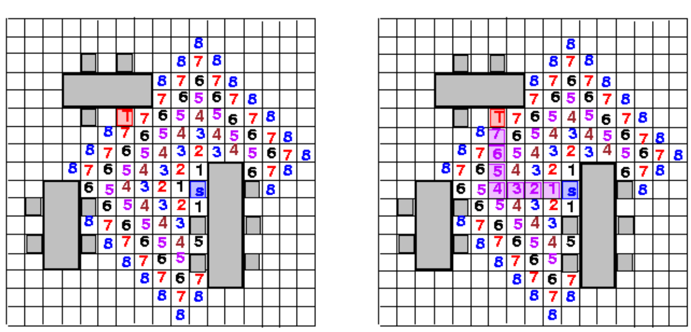
Final labeling

How many grids visited using Lee's algorithm?

13	12	11	10			7	6	7	7			9	10			
12	11	10	9			6	5	6	7			8	9	10	11	12
11	10	9	8	7	6	5	4					7	8	9	10	11
10	9	8	7	6	5	4	3					6	7	8	9	10
		7	6	5	4	3	2	1	2	3	4	5	6	7	8	9
		6	5	4	3	2	1	S	1	2	3	4	5	6	7	8
9	8	7	6			3	2	1	2	3	4	5	6	7	8	9
10	9	8	7						3		5	6	7	8	9	10
11	10	9	8	9	10					7	6	7	8	9	10	11
12	11			10	11	12	11	10	9	8			9	10	11	12
13	12			11	12	13	12	11	10	9			10	11	12	13
				12	13		13	12	11	10			11	12	13	
				13				13	12	11			12	13		
									13	12	T		13			
										13						

Lee's Algorithm

- Find a path from *S* to *T* by "wave propagation".



Filling Retrace

Time and Space Complexity

- For a grid structure of size $w \times h$:
  - Time per net = $O(wh)$
  - Space = $O(wh \log wh)$ ($O(\log wh)$ bits are needed to store each label.)
- For a 4000×4000 grid structure:
  - 24 bits per label
  - Total 48 Mbytes of memory!

Improvement to Lee's Algorithm

- Improvement on memory:
  - Aker's Coding Scheme
- Improvement on run time:
  - Starting point selection
  - Double fan-out
  - Framing
  - Hadlock's Algorithm
  - Soukup's Algorithm

Improvement to Lee's Algorithm

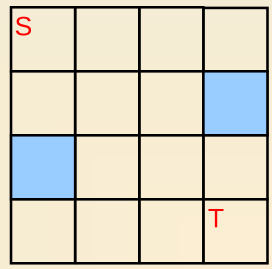
- Improvement on memory:
  - Aker's Coding Scheme
- Improvement on run time:
  - Starting point selection
  - Double fan-out
  - Framing
  - Hadlock's Algorithm
  - Soukup's Algorithm

Aker's Coding Scheme

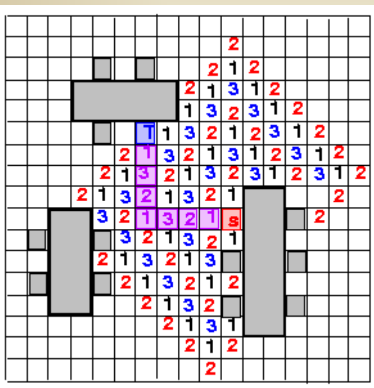
- For the Lee's algorithm, labels are needed during the retrace phase
- But there are only two possible labels for neighbors of each vertex labeled *i*, which are, *i*-1 and *i*+1
- So, is there any method to reduce the memory usage?

Aker's Coding Scheme

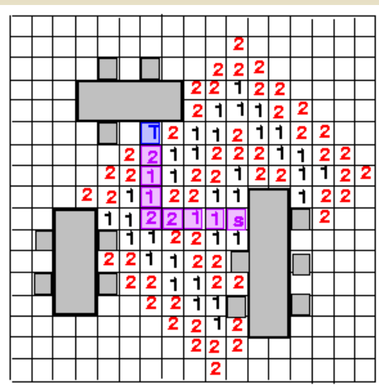
- One bit (independent of grid size) is enough to distinguish between the two labels
- Sequence: … (what sequence can be applied?)
  - The labels before and after each label must be different in order to tell the forward or the backward directions
- How to represent empty / blocked ?
  - Way 1: coding sequence 1, 2, 3, 1, 2, 3, …; states: 1, 2, 3, empty, blocked (3 bits required)
  - Way 2: coding sequence 1, 1, 2, 2, 1, 1, 2, 2, …; states: 1, 2, empty, blocked (need only 2 bits)



Aker's Coding Example



Sequence: 1, 2, 3, 1, 2, 3, …



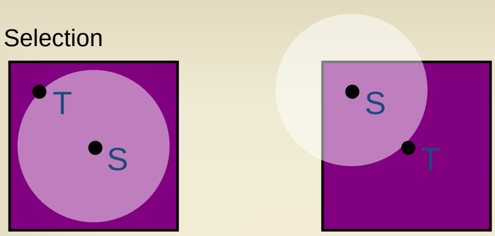
Sequence: 1, 1, 2, 2, 1, 1, 2, 2, …

Schemes to Reduce Run Time

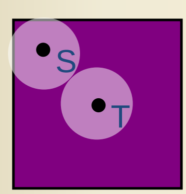
- Starting point selection: Choose the point farthest from the center of the grid as the starting point.
- Double fan-out: Propagate waves from both the source and the target cells.
- Framing: Search inside a rectangle area 10--20% larger than the bounding box containing the source and target.
  - Need to enlarge the rectangle and redo if the search fails.

Schemes to Reduce Run Time

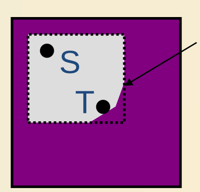
Starting Point Selection



Double Fan-Out



Framing



The frame can be 10-20% larger than the bounding box containing *S* and *T*. Redo if the search fails.

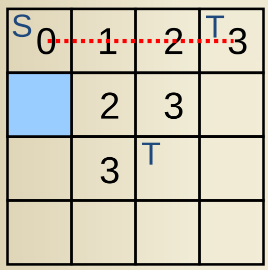
Multi-Terminal Nets

- For a *k*-terminal net, connect the *k* terminals using a rectilinear Steiner tree with the shortest wire length on the maze
- This problem is NP-Complete
- Just want to find some good heuristics

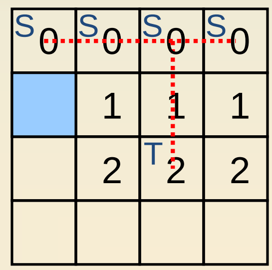
Multi-Terminal Nets

- This problem can be solved by extending the Lee's algorithm:
  - Connect one terminal at a time, or
  - Search for several targets simultaneously, or
  - Propagate wave fronts from several different sources simultaneously

Extension to Multi-Terminal Nets



1st Iteration



2nd Iteration

Using Maze Routing for Global Routing

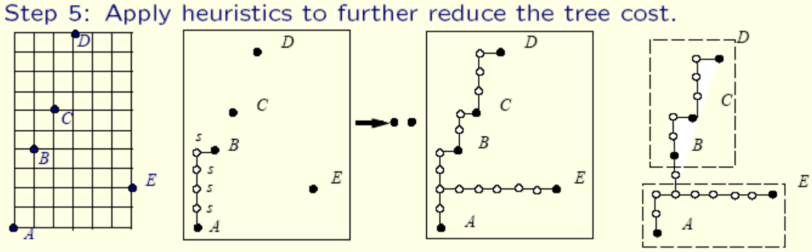
- Sequential Approach
- Algorithm:

We can use different methods to do this

1. Graph modeling of the routing regions
2. For each net k:
   2.1 Find a route r for net k on the graph.
   2.2 For each edge e in r:
      2.2.1 capacity(e) = capacity(e) - 1
      2.2.2 if capacity(e) < 0 then cost(e) = $\alpha \times$ cost(e)

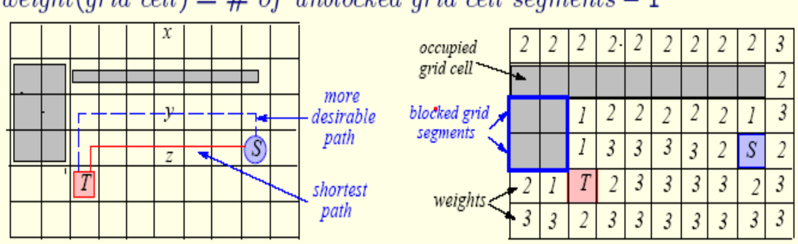
Maze Routing for Multi-Terminal Nets

- Step 1: Propagate wave from the source s to the closet target.
- Step 2: Mark ALL cells on the path as s.
- Step 3: Propagate wave from ALL s cells to the other cells.
- Step 4: Continue until all cells are reached.
- Step 5: Apply heuristics to further reduce the tree cost.

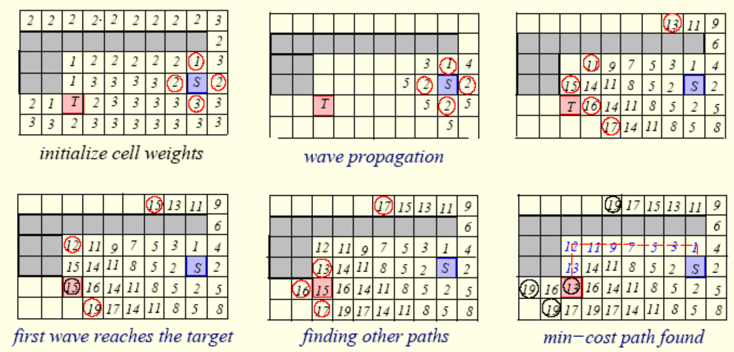


Maze Routing on Weighted Graph

- Motivation: finding more desirable paths
- $weight(grid\ cell) = \#\ of\ unblocked\ grid\ cell\ segments - 1$



Example of Weighted Graph



Branch-and-bound

We can stop propagating a wavefront n with label >= shortest path length discovered so far. This is because the label of n's neighbors will always increase IF they are propagated from n. So we only need to propagate the node with label 12. We stop when there is no more wavefront to propagate.

Improvement to Lee's Algorithm

- Improvement on memory:
  - Aker's Coding Scheme
- Improvement on run time:
  - Starting point selection
  - Double fan-out
  - Framing
  - Hadlock's Algorithm
  - Soukup's Algorithm

Hadlock's Algorithm

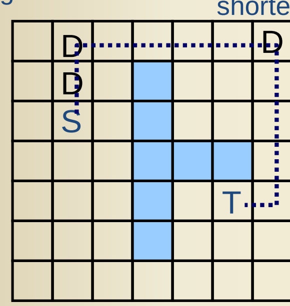
- Hadlock, "A shortest path algorithm for grid graphs," *Networks*, 1977.
- Uses detour number (instead of labeling wavefront in Lee's router)
  - Detour number, $d(P)$: # of grid cells directed **away from** its target on path P.
  - $MD(S, T)$: the Manhattan distance between S and T.
  - Path length of P, $l(P)$: $l(P) = MD(S, T) + 2\, d(P)$.
  - $MD(S, T)$ fixed! $\Rightarrow$ Minimize $d(P)$ to find the shortest path.
  - For any cell labeled i, label its adjacent unblocked cells **away from** T i+1; label i otherwise.
- Time and space complexities: $O(MN)$, but substantially reduces the # of searched cells.
- Finds the shortest path between S and T.

Detour Number

- For a path P from S to T, let detour number d(P) = # of grids directed away from T, then
  - L(P) = MD(S,T) + 2d(P)

length

shortest Manhattan distance



D: Detour

d(P) = 3
MD(S,T) = 6
L(P) = 6+2x3 = 12

- So minimizing L(P) and d(P) are the same

Hadlock's Algorithm

- Label vertices with detour numbers
- Vertices with smaller detour number are expanded first
- Therefore, favor paths without detour

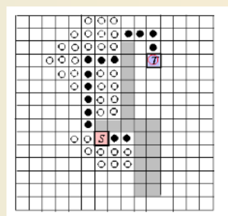
3	2	2	2	2	2	3
2	1	1		2	2	
1	S	0		2		
1	0	0				
1	0	0		2	T	
2	1	1		2	2	
3	2	2	2	2	2	

Improvement to Lee's Algorithm

- Improvement on memory:
  - Aker's Coding Scheme
- Improvement on run time:
  - Starting point selection
  - Double fan-out
  - Framing
  - Hadlock's Algorithm
  - Soukup's Algorithm

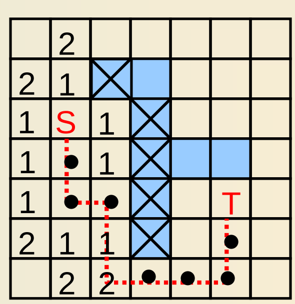
Soukup's Algorithm

- Soukup, “Fast maze router,” DAC-78.
- Combined breadth-first and depth-first search.
  - Depth-first (**line**) search is first directed toward target *T* until an obstacle or *T* is reached.
  - Breadth-first (Lee-type) search is used to “bubble” around an obstacle if an obstacle is reached.
- Time and space complexities: $O(MN)$, but 10--50 times faster than Lee's algorithm.
- Find **a** path between *S* and *T*, but may not be the shortest!



Soukup's Algorithm

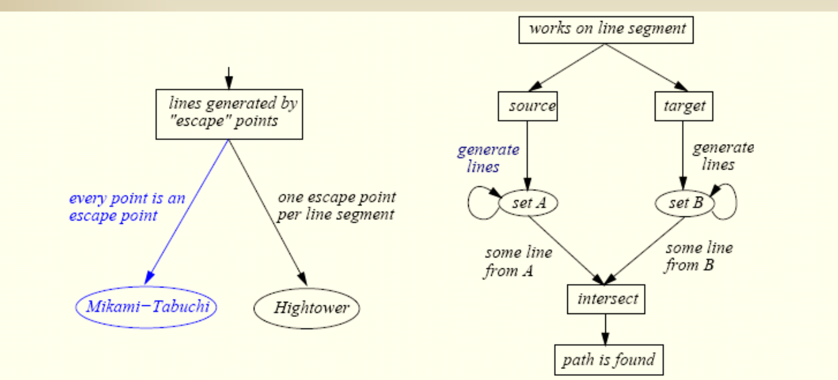
- Soukup’s Algorithm: DFS+BFS
  - Explore in the direction towards the target without changing direction. (DFS)
  - If obstacle is hit, search around the obstacle. (BFS)
- May get Sub-Optimal solution



Global Routing Approaches

- Sequential Approach (Rip-up and Re-route)
  - Maze Routing
  - Line probing
  - Shortest Path Based Algorithms
  - Steiner Tree Based Algorithms
- Concurrent Approach
  - Integer Programming

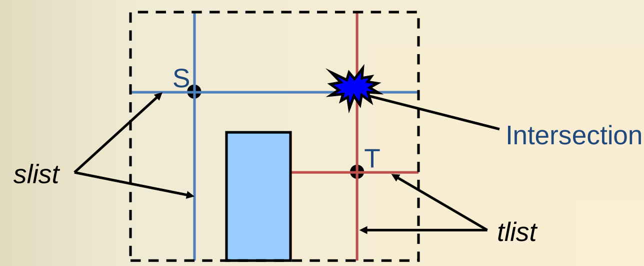
Futures of Line-Search Algorithms



- Time and space complexities: $O(L)$, where L is the $\#$ of line segments generated.

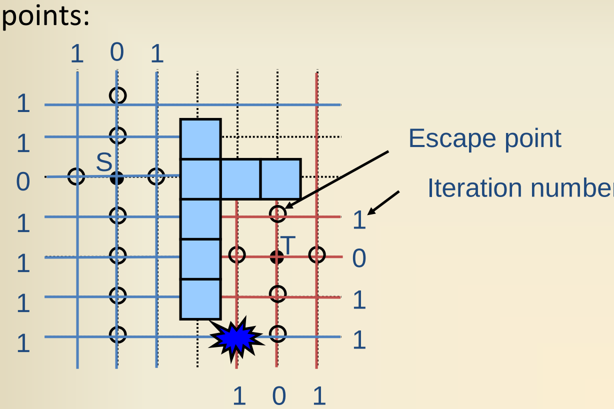
Line Probing

- Keep two lists of line segments, *slist* and *tlist*, for the source and the target respectively
- If a line segment from *slist* intersects with one from *tlist*, a route is found; else, new line segments are generated from the escape points



Mikami & Tabuchi's Algorithm

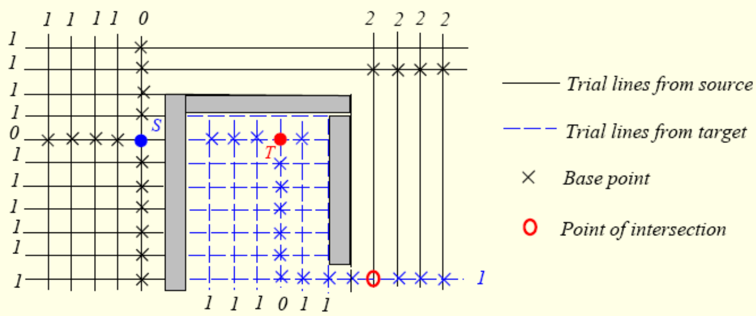
- We can use all the grid vertices on the line segments as escape points:



- Always find a path but may not be optimal

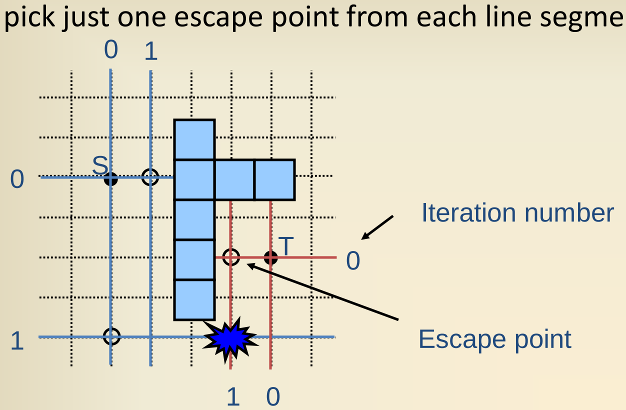
Mikami & Tabuchi's Algorithm

- Mikami & Tabuchi, "A computer program for optimal routing of printed circuit connectors," IFIP, H47, 1968.
- Every grid point is an escape point.



Hightower's Algorithm

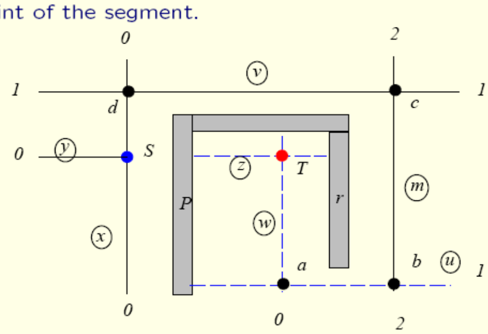
- ◆ We can pick just one escape point from each line segment



- ◆ May fail to find a path even if one exists

Hightower's Algorithm

- Hightower, "A solution to line-routing problem on the continuous plane," DAC-69.
- A single escape point on each line segment.
- If a line parallels to the blocked cells, the escape point is placed just past the endpoint of the segment.



Comparison of Algorithms

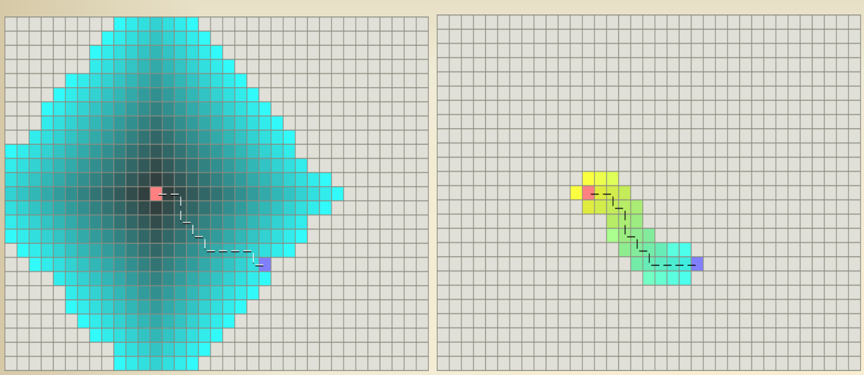
	Maze routing			Line search	
	Lee	Soukup	Hadlock	Mikami	Hightower
Time	$O(MN)$	$O(MN)$	$O(MN)$	$O(L)$	$O(L)$
Space	$O(MN)$	$O(MN)$	$O(MN)$	$O(L)$	$O(L)$
Finds path if one exists?	yes	yes	yes	yes	no
Is the path shortest?	yes	no	yes	no	no
Works on grids or lines?	grid	grid	grid	line	line

- Soukup, Mikami, and Hightower all adopt some sort of line-search operations $\Rightarrow$ cannot guarantee shortest paths.

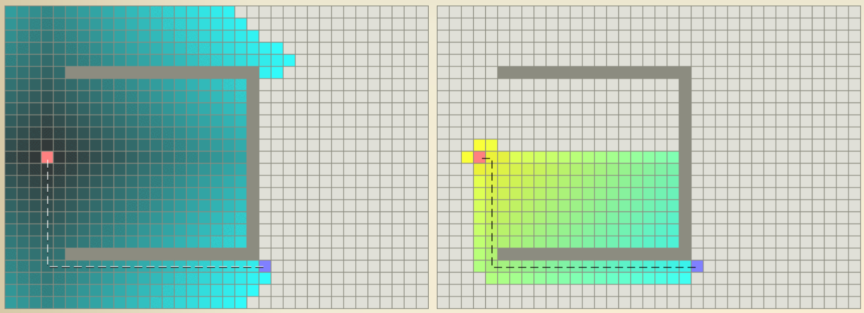
BFS based Maze Routing (A*)

- Need to search whole space?
  - Guide the search to the goal explicitly
- A* search is faster if you need "good path", not "perfect path"
  - Use priority queue
  - C(n) = F(n)+H(n)
    - F(n) is a computed cost from source to current location
    - H(n) is a predicted cost from current location to target
    - If H(n)=0, it becomes maze routing!
  - Optimal (shortest path) when H(n) <= H'(n) (no overestimation)
    - H'(n) is the exact cost
    - H(n)=0 never overestimates!

Maze vs A* routing (I)



Maze vs A* routing (II)



Global Routing Approaches

- Sequential Approach (Rip-up and Re-route)
  - Maze Routing
  - Line probing
  - Shortest Path Based Algorithms
  - Steiner Tree Based Algorithms
- Concurrent Approach
  - Integer Programming

Shortest Path Based Algorithms

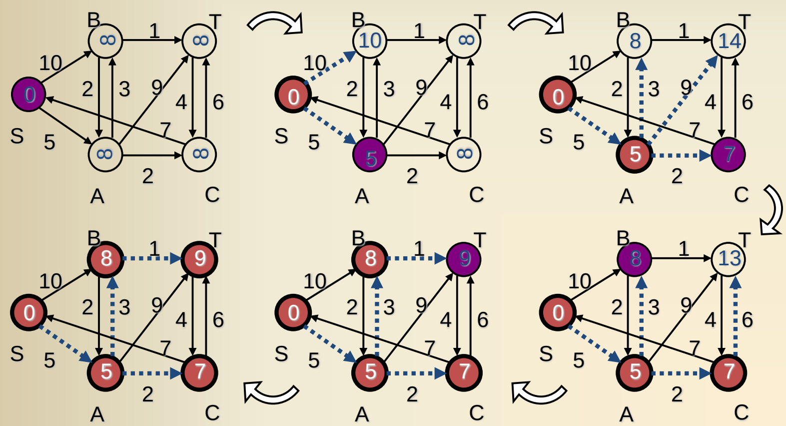
- For 2-terminal nets only
- Use Dijkstra's algorithm to find the shortest path between the source *s* and the sink *t* of a net
- Different from Maze Routing:
  - The graph need not be a rectangular grid
  - The edges need not be of unit length

Dijkstra’s Shortest Path Algorithm

- Label of vertices = Shortest distance from S
- Let P be the set of permanently labeled vertices
- Initially,
  - P = Empty Set.
  - Label of S = 0, Label of all other vertices = infinity
- While (T is not in P) do
  - Pick the vertex v with the min. label among all vertices not in P
  - Add v to P
  - Update the label for all neighbors of v

Dijkstra's Algorithm: Example

P (Permanently Labeled) | Min. Label Vertex

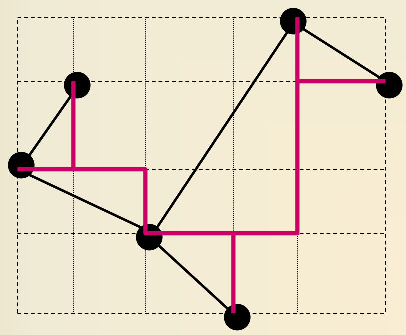


Global Routing Approaches

- Sequential Approach (Rip-up and Re-route)
  - Maze Routing
  - Line probing
  - Shortest Path Based Algorithms
  - Steiner Tree Based Algorithms
- Concurrent Approach
  - Integer Programming

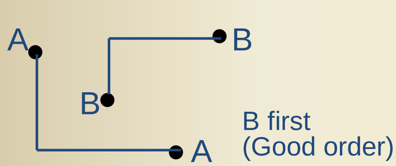
Steiner Tree Based Algorithms

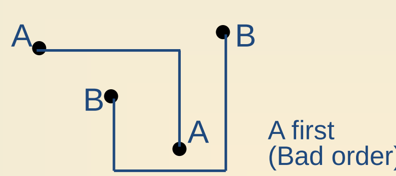
- For multi-terminal nets
- Find Steiner tree instead of shortest path
- Construct a Steiner tree from the minimum spanning trees (MST)



Net Ordering

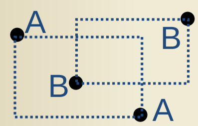
- In sequential approach, we need some <u>net ordering</u>
- A bad net ordering will increase the total wire length, and may even prevent completion of routing for some circuits which are indeed routable





Criteria for Net Ordering

- Criticality of net – critical nets first
- Estimated wire length – short nets first since they are less flexible
- Consider bounding rectangles (BR)

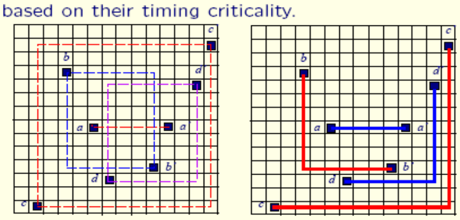


B is in A's BR

Which one should be routed first and why? (Note that this rule of thumb is not always applicable.)

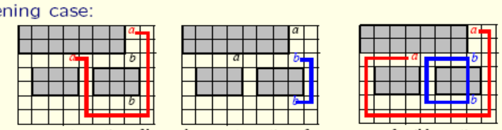
Net Ordering (cont'd)

- Order the nets in the ascending order of the # of pins within their bounding boxes.
- Order the nets in the ascending (or descending??) order of their lengths.
- Order the nets based on their timing criticality.



routing ordering: a (0) −> b (1) −> d (2) −> c (6)

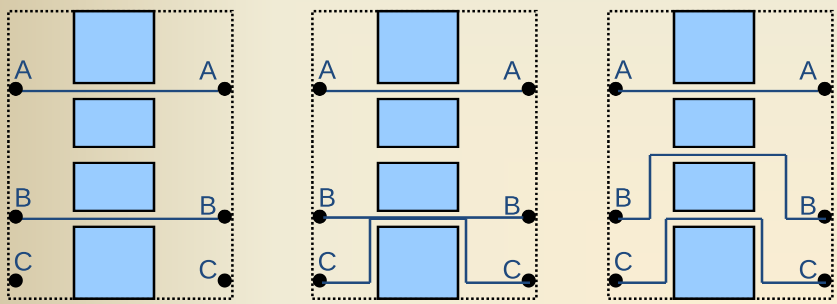
- A mutually intervening case:



a prevents routing of b *b prevents routing of a* *a feasible routing*

Rip-up and Re-route

- It is impossible to get the optimal net ordering
- If some nets are failed to be routed, the rip-up and re-route technique can be applied:



Cannot route C

So rip-up B
and route C first

Finally route B

Global Routing Approaches

- Sequential Approach (Rip-up and Re-route)
  - Maze Routing
  - Line probing
  - Shortest Path Based Algorithms
  - Steiner Tree Based Algorithms
- Concurrent Approach
  - Integer Programming

Concurrent Approach

- Consider all the nets simultaneously
- Formulate as an integer programming
- Given:

Nets	*Set of possible routing trees*
net 1	T_{11}, T_{12}, ... , T_{1k_1}
⋮	⋮
net *n*	T_{n1}, T_{n2}, ... , T_{nk_n}

L_{ij} = Total wire length of T_{ij}
C_e = Capacity of edge e

- Determine variable x_{ij} s.t. x_{ij} = 1 if T_{ij} is used
  x_{ij} = 0 otherwise

Integer Program Formulation

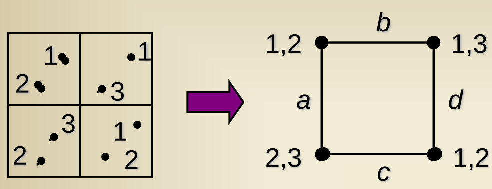
$$\text{Min.} \sum_{i=1}^{n} \sum_{j=1}^{k_i} L_{ij} \times x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{k_i} x_{ij} = 1 \quad \text{for all } i = 1, \ldots, n$$

$$\sum_{i,j \text{ s.t. } e \in T_{ij}} x_{ij} \le C_e \quad \text{for all edge } e$$

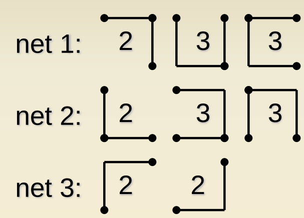
$$x_{ij} = 0 \text{ or } 1 \quad \forall i, j$$

Concurrent Approach: Example



$C_a = C_b = C_c = C_d = 2$

Possible trees:



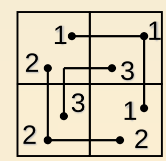
Min. $2x_{11} + 3x_{12} + 3x_{13} + 2x_{21} + 3x_{22} + 3x_{23} + 2x_{31} + 2x_{32}$

$$\text{s.t.} \begin{cases} x_{11} + x_{12} + x_{13} = 1; \\ x_{21} + x_{22} + x_{23} = 1; \\ x_{31} + x_{32} = 1; \\ x_{ij} = 0 \text{ or } 1 \quad \forall i, j; \end{cases}$$

What are the constraints for edge capacity?

$$\boxed{x_{12} + x_{13} + x_{21} + x_{23} + x_{31} < C_a}$$

Solution



2024/5/16 Andy Yu-Guang Chen 89

Integer Programming Approach

- Standard techniques to solve IP
- No net ordering → Give global optimum
- Can be extremely slow, especially for large problems
- To make it faster, a fewer choices of routing trees for each net can be used. May make the problem infeasible or give a bad solution
- Determining a good set of choices of routing trees is a hard problem by itself

2024/5/16 Andy Yu-Guang Chen 90

Integer Programming Approach

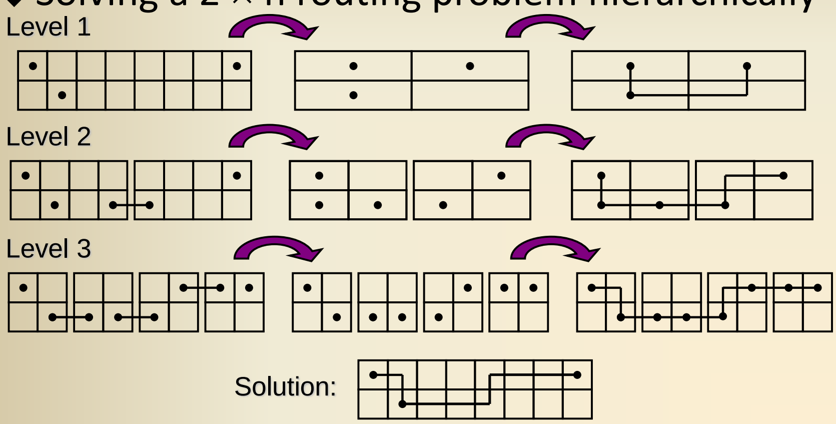
- Hierarchical Approach to Speed Up Integer Programming Formulation For Global Routing
- M. Burstein and R. Pelavin, “Hierarchical Wire Routing”, IEEE TCAD, vol. CAD-2, pages 223-234, Oct. 1983.

Hierarchical Approach

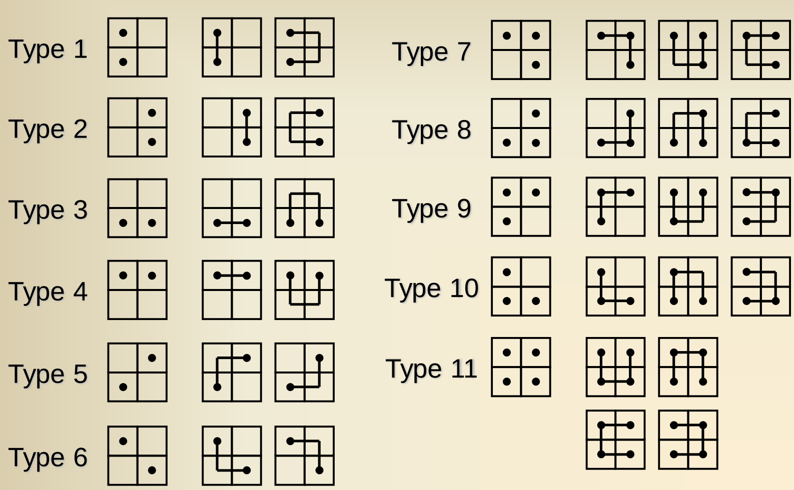
- Large Integer Programs are difficult to solve
- Hierarchical Approach reduces global routing to routing problems on a 2×2 grid
- Decompose recursively in a top-down fashion
- Those 2×2 routing problems can be solved optimally by integer programming formulation

Hierachical Approach: Example

◆Solving a 2 × n routing problem hierarchically



Types of 2×2 Routing Problems



Objective Function of 2×2 Routing

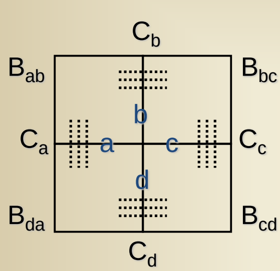
- Possible Routing Trees:

  $T_{11}, T_{12}, T_{21}, T_{22}, \ldots, T_{11,1}, \ldots, T_{11,4}$
- # of nets of each type: $n_1, \ldots, n_{11}$
- Determine
  - x_{ij}: # of type-i nets using T_{ij} for routing.
  - y_i: # of type-i nets that fails to route

$$y_i + \sum_j x_{ij} = n_i \quad i = 1, \ldots, 11$$

$$\text{Minimize} \quad \sum_{i=1}^{11} y_i$$

Constraints of 2×2 Routing



Constraints on Edge Capacity:

$$\sum_{i,j \text{ s.t. } a \in T_{ij}} x_{ij} \le C_a$$

$$\sum_{i,j \text{ s.t. } b \in T_{ij}} x_{ij} \le C_b$$

$$\sum_{i,j \text{ s.t. } c \in T_{ij}} x_{ij} \le C_c$$

$$\sum_{i,j \text{ s.t. } d \in T_{ij}} x_{ij} \le C_d$$

Constraints on # of Bends in a Region:

$$\sum_{i,j \text{ s.t. } T_{ij} \text{ has a bend in region } ab} x_{ij} \le B_{ab}$$

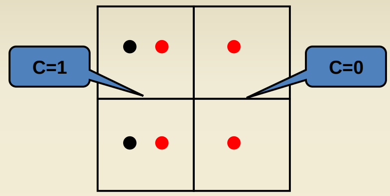
$$\sum_{i,j \text{ s.t. } T_{ij} \text{ has a bend in region } bc} x_{ij} \le B_{bc}$$

$$\sum_{i,j \text{ s.t. } T_{ij} \text{ has a bend in region } cd} x_{ij} \le B_{cd}$$

$$\sum_{i,j \text{ s.t. } T_{ij} \text{ has a bend in region } da} x_{ij} \le B_{da}$$

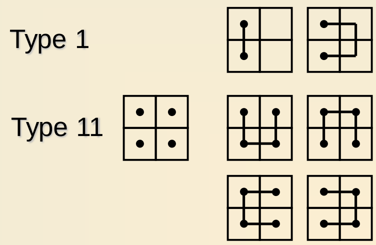
Pop Quiz

- If there are two nets, one with 2 pins, the other with 4 pins with a zero capacity edge
  - What is going to be the result?



$$y_i + \sum_j x_{ij} = n_i \quad i = 1, \ldots, 11$$

$$\text{Minimize} \quad \sum_{i=1}^{11} y_i$$



ILP Formulation of 2×2 Routing

$$\begin{aligned}
\text{Min.} \quad & \sum_{i=1}^{11} y_i \\
\text{s.t.} \quad & y_i + \sum_j x_{ij} = n_i \quad i = 1, \ldots, 11 \\
& x_{ij} \geq 0,\ y_i \geq 0 \quad \forall i, j
\end{aligned}$$

$$\sum_{i,j\ \text{s.t.}\ a \in T_{ij}} x_{ij} \leq C_a \qquad \sum_{i,j\ \text{s.t.}\ T_{ij}\ \text{has a bend in region}\ ab} x_{ij} \leq B_{ab}$$

$$\sum_{i,j\ \text{s.t.}\ b \in T_{ij}} x_{ij} \leq C_b \qquad \sum_{i,j\ \text{s.t.}\ T_{ij}\ \text{has a bend in region}\ bc} x_{ij} \leq B_{bc}$$

$$\sum_{i,j\ \text{s.t.}\ c \in T_{ij}} x_{ij} \leq C_c \qquad \sum_{i,j\ \text{s.t.}\ T_{ij}\ \text{has a bend in region}\ cd} x_{ij} \leq B_{cd}$$

$$\sum_{i,j\ \text{s.t.}\ d \in T_{ij}} x_{ij} \leq C_d \qquad \sum_{i,j\ \text{s.t.}\ T_{ij}\ \text{has a bend in region}\ da} x_{ij} \leq B_{da}$$

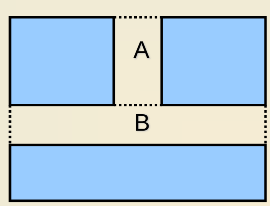
- Only 39 variables (28 x_{ij} and 11 y_i) and 19 constraints (plus 38 non-negative constrains)
- Problems of this size are usually not too difficult to solve

Routing Outline

- Routing an overview
- Global routing
  - Sequential Approach (Rip-up and Re-route)
    - Maze Routing
    - Line probing
    - Shortest Path Based Algorithms
    - Steiner Tree Based Algorithms
  - Concurrent Approach
    - Integer Programming
- Detailed routing
  - Channel routing

After Global Routing: Detailed Routing

- The routing regions are divided into channels and switchboxes

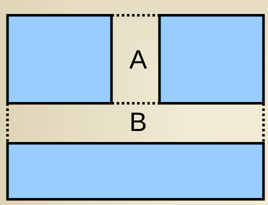


- So only need to consider the channel routing problem and the switchbox routing problem

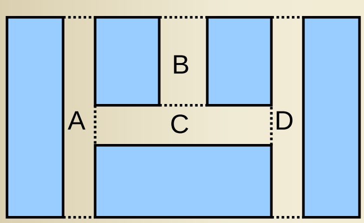
Channel Routing for Different Styles

- For gate-array design, channel widths are fixed. The goal is to finish routing of all the nets
- For standard-cell and full-custom design, channels are expandable. The goal is to route all nets using the minimum channel width
- We will consider the case when the channels are expandable

Channel Ordering



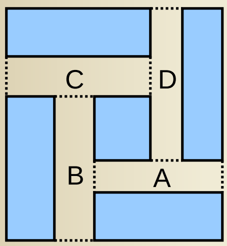
The width of A is not known until A is routed, we must route A first.



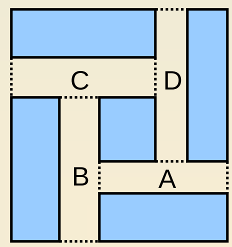
What should be the routing order for this example?

Channel Ordering

No feasible
channel order!



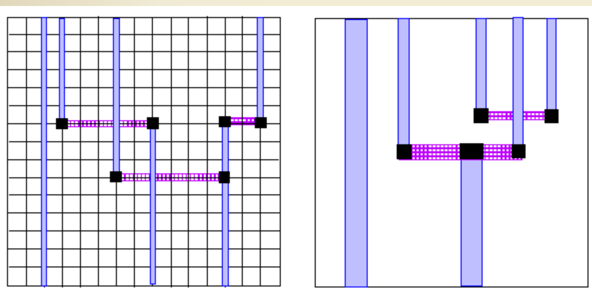
Need to use switchbox



1. Fix the terminals between A & B
2. Route B, C, then D (channel)
3. Route A (switchbox)

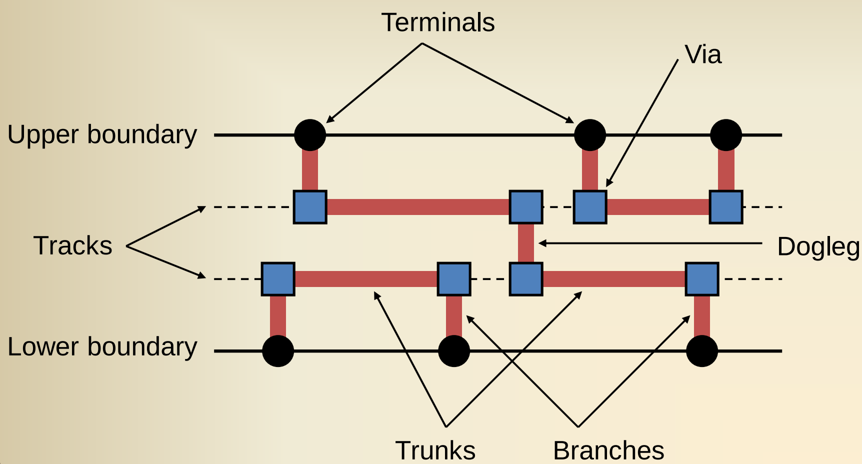
Routing Models

- **Grid-based model:**
  - A grid is super-imposed on the routing region.
  - Wires follow paths along the grid lines.
  - Pitch: distance between two grid lines.
- **Gridless model:**
  - Any model that does not follow this “gridded” approach.

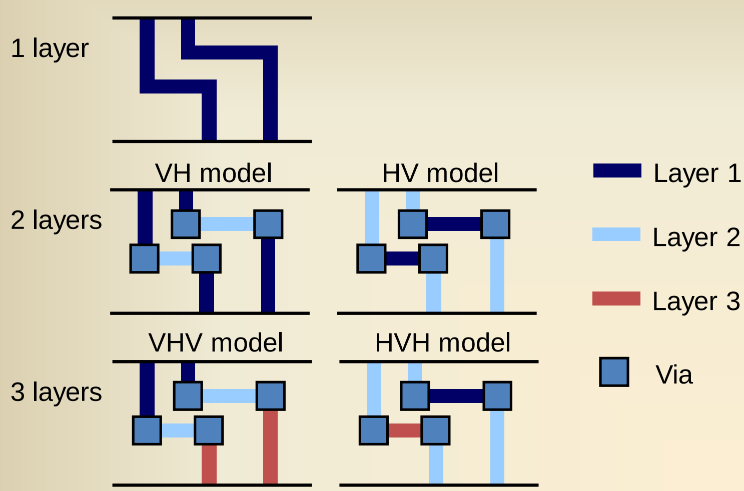


grid–based gridless

Channel Routing Terminology



Routing Layer Models



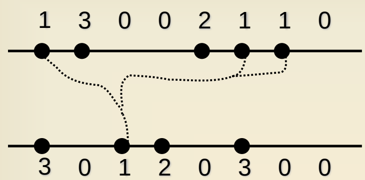
Channel Routing Problem

- Input:
  - Two vectors of the same length to represent the pins on two sides of the channel
  - Number of layers and layer model used
- Output:
  - Connect pins of the same net together
  - Minimize the channel width
  - Minimize the number of vias

Routing Considerations

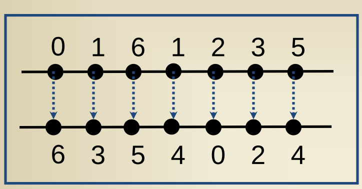
- Number of terminals (two-terminal vs. multi-terminal nets)
- Net widths (power and ground vs. signal nets)
- Via restrictions (stacked vs. conventional vias)
- Boundary types (regular vs. irregular)
- Number of layers (two vs. three, more layers?)
- Net types (critical vs. non-critical nets)

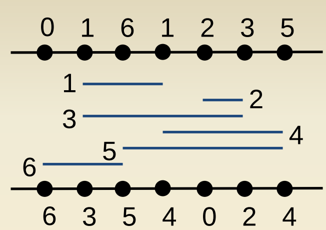
Channel Routing Problem

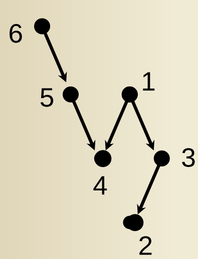


Example: (13002110)
(30120300)
where 0 = no terminal

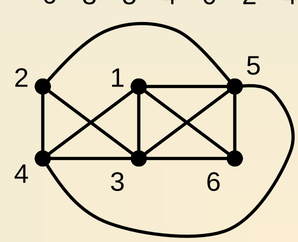
Constraint Graphs





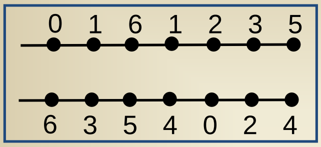


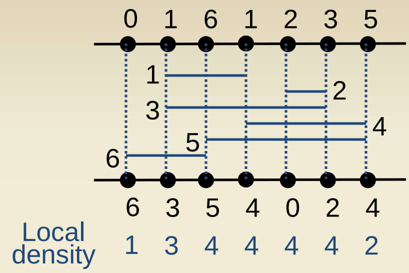
Vertical constraint graph



Horizontal constraint graph

Lower Bound on Channel Width





Channel density = Maximum local density

Channel density = 4

- Local density at column *i, d(i)*: total # of nets that crosses column *i*.
- **Channel density:** maximum local density
  - # of horizontal tracks required ≥ channel density.

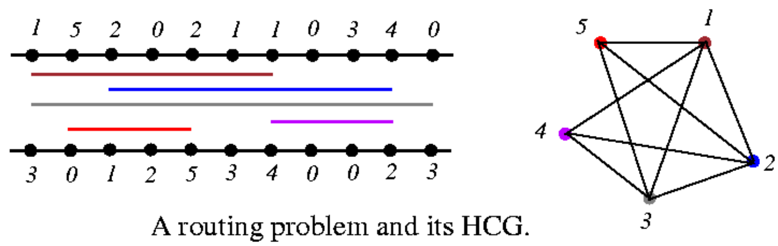
Lower bound on channel width = Channel density

Channel Routing Problem

- **Assignments of horizontal segments of nets to tracks**
- **Assignments of vertical segments to connect the following:**
  - horizontal segments of the same net in different tracks, and
  - the terminals of the net to horizontal segments of the net.
- **Horizontal and vertical constraints must not be violated.**
  - Horizontal constraints between two nets: the horizontal span of two nets overlaps each other
  - Vertical constraints between two nets: there exists a column such that the terminal on top of the column belongs to one net and the terminal on bottom of the column belongs to another net
- **Objective: Channel height is minimized** (i.e., channel area is minimized).

Horizontal Constraint Graph (HCG)

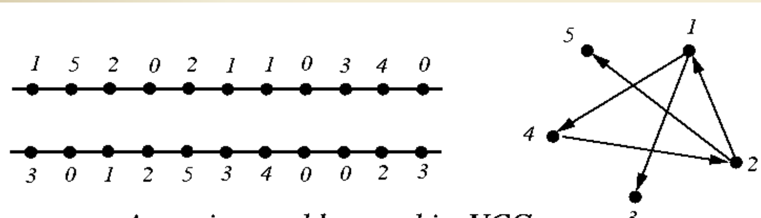
- HCG $G = (V, E)$ is **undirected** graph where
  - $V = \{ v_i \mid v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j) \mid$ a horizontal constraint exists between n_i and $n_j\}$.
- For graph G: vertices $\Leftrightarrow$ nets; edge $(i, j) \Leftrightarrow$ net i overlaps net j.



A routing problem and its HCG.

Vertical Constraint Graph (VCG)

- VCG $G = (V, E)$ is **directed** graph where
  - $V = \{ v_i \mid v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j) \mid$ a vertical constraint exists between n_i and $n_j\}$.
- For graph G: vertices $\Leftrightarrow$ nets; edge $i \rightarrow j \Leftrightarrow$ net i must be above net j.



A routing problem and its VCG.

2-L Channel Routing: Basic Left-Edge Algorithm

- Hashimoto & Stevens, "Wire routing by optimizing channel assignment within large apertures," DAC-71.
- **No vertical constraint.**
- HV-layer model is used.
- **Doglegs are not allowed.**
- Treat each net as an interval.
- Intervals are sorted according to their left-end *x*-coordinates.
- Intervals (nets) are routed one-by-one according to the order.
- For a net, tracks are scanned from top to bottom, and the first track that can accommodate the net is assigned to the net.
- Optimality: produces a routing solution with the minimum # of tracks (if no vertical constraint).

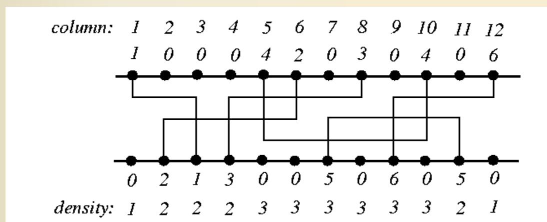
Basic Left-Edge Algorithm

Algorithm: Basic_Left-Edge(U, $track[j]$)
U: set of unassigned intervals (nets) $I_1, \ldots, I_n$;
$I_j=[s_j, e_j]$: interval j with left-end x-coordinate s_j and right-end e_j;
$track[j]$: track to which net j is assigned.

begin
$U \leftarrow \{I_1, I_2, \ldots, I_n\}$;
$t \leftarrow 0$;
while ($U \neq \emptyset$) **do**
  $t \leftarrow t + 1$;
  $watermark \leftarrow 0$;
  while (there is an $I_j \in U$ s.t. $s_j > watermark$) **do**
    Pick the interval $I_j \in U$ with $s_j > watermark$, nearest $watermark$;
    $track[j] \leftarrow t$;
    $watermark \leftarrow e_j$;
    $U \leftarrow U - \{I_j\}$;
end

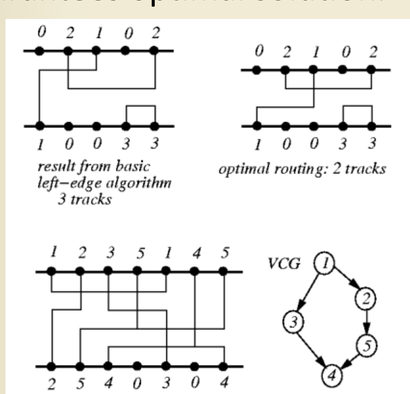
Basic Left-Edge Example

- $U = \{I_1, I_2, \ldots, I_6\}$; I_1 = [1, 3], I_2 = [2, 6], I_3 = [4, 8], I_4 = [5, 10], I_5 = [7, 11], I_6 = [9, 12].
- t =1:
  - Route I_1: *watermark* = 3;
  - Route I_3 : *watermark* = 8;
  - Route I_6: *watermark* = 12;
- t = 2:
  - Route I_2 : *watermark* = 6;
  - Route I_5 : *watermark* = 11;
- t = 3: Route I_4



Basic Left-Edge Algorithm

- If there is no vertical constraint, the basic left-edge algorithm is optimal.
- If there is any vertical constraint, the algorithm no longer guarantees optimal solution.



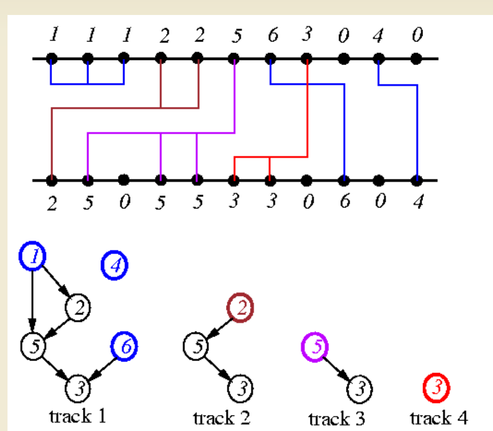
Constrained Left-Edge Algorithm

Algorithm: Constrained_Left-Edge(U, *track*[j])
U: set of unassigned intervals (nets) $I_1, \ldots, I_n$;
$I_j=[s_j, e_j]$: interval j with left-end x-coordinate s_j and right-end e_j;
track[j]: track to which net j is assigned.

```
begin
U ← { I1, I2, …, In };
t ← 0;
while (U ≠ ∅) do
    t ← t + 1;
    watermark ← 0;
    while (there is an unconstrained Ij ∈ U s.t. sj > watermark) do
        Pick the interval Ij ∈ U that is unconstrained,
          with sj > watermark, nearest watermark;
        track[j] ← t;
        watermark ← ej;
        U ← U - {Ij};
end
```

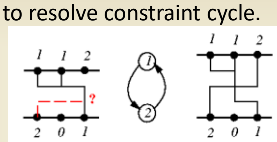

Constrained Left-Edge Example

- $I_1 = [1, 3]$, $I_2 = [1, 5]$, $I_3 = [6, 8]$, $I_4 = [10, 11]$, $I_5= [2, 6]$, $I_6 = [7, 9]$.
- Track 1: Route I_1 (cannot route I_3); Route I_6; Route I_4.
- Track 2: Route I_2; cannot route I_3.
- Track 3: Route I_5.
- Track 4: Route I_3.

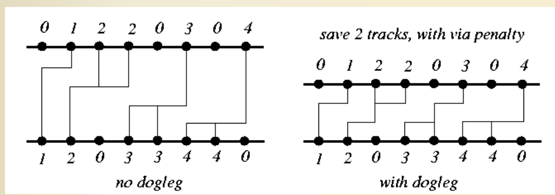


Dogleg Channel Router

- Deutch, “A dogleg channel router,” 13rd DAC, 1976.
- **Drawback of Left-Edge: cannot handle the cases with constraint cycles.**
  - **Doglegs** are used to resolve constraint cycle.

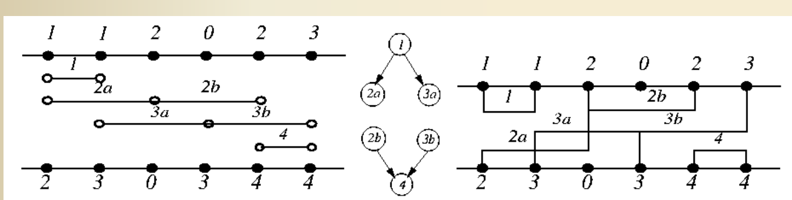


- **Drawback of Left-Edge: the entire net is on a single track.**
  - **Doglegs** are used to place parts of a net on different tracks to minimize channel height.
  - Might incur penalty for additional vias.



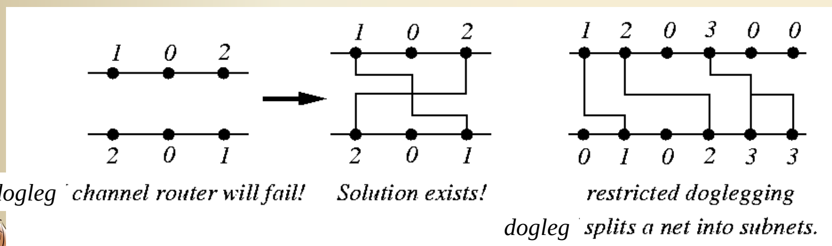
Dogleg Channel Router

- Each multi-terminal net is broken into a set of 2-terminal nets.
- Two parameters are used to control routing:
  - Range: Determine the # of consecutive 2-terminal subnets of the same net that can be placed on the same track.
  - Routing sequence: Specifies the starting position and the direction of routing along the channel.
- Modified Left-Edge Algorithm is applied to each subnet.



Restricted vs. Unrestricted Doglegging

- **Unrestricted doglegging:** Allow a dogleg even at a position where there is no pin.
- **Restricted doglegging:** Allow a dogleg only at a position where there is a pin belonging to that net.
- The dogleg channel router does not allow unrestricted doglegging.



2024/5/16 Andy Yu-Guang Chen 123

Thank You All for The Great Semester

Andy, Yu-Guang Chen
Assistant Professor, Department of EE, NCU
Email: andyygchen@ee.ncu.edu.tw
FB: Yu-Guang Chen
IG: ncu.eda.andy
Google hangout: andyron751986

124