# Grid Routing

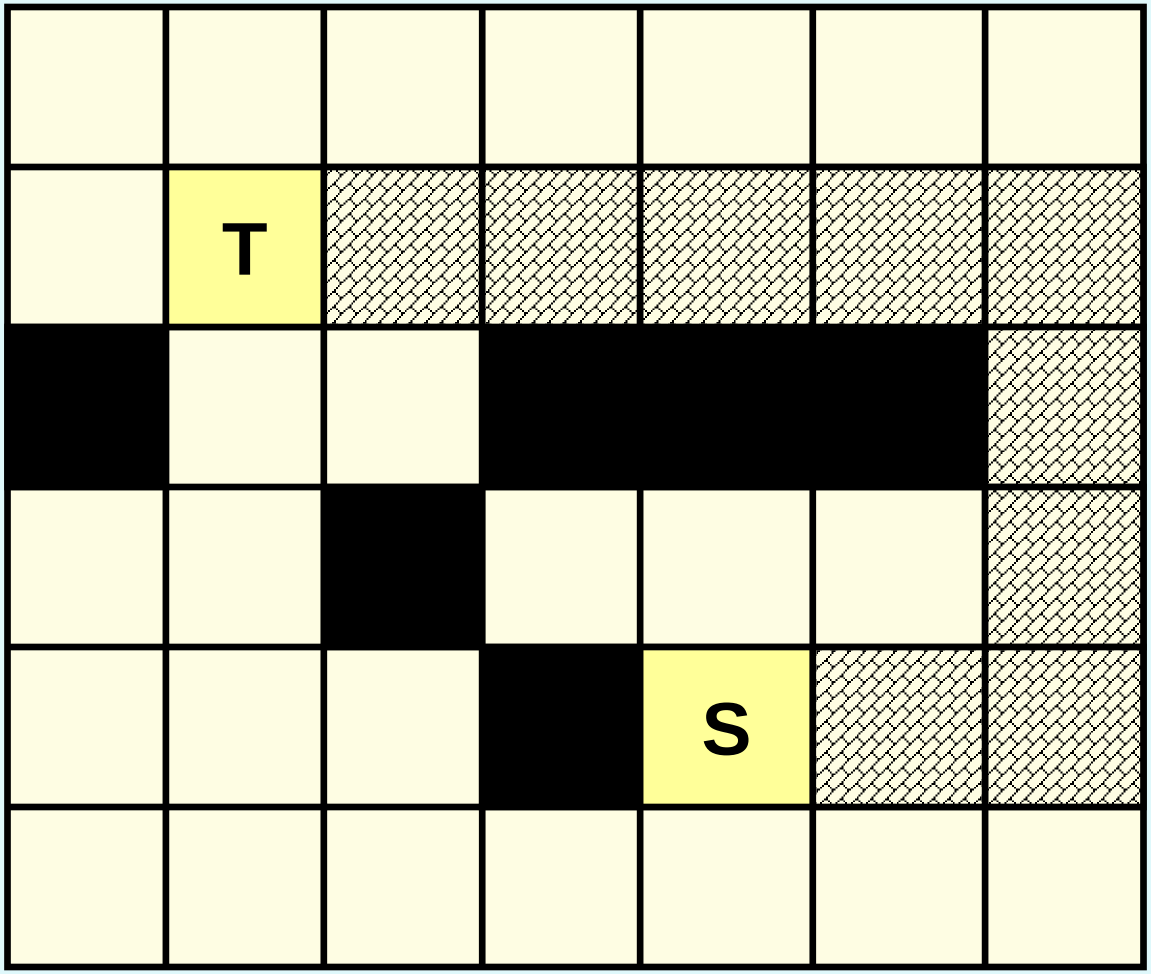
Prof. Dipanwita Roy Chowdhury
Department of Computer Science & Enggineering
Indian Institute of Technology Kharagpur
India

# Introduction

- In the VLSI design cycle, *routing* follows *cell placement*.
- During routing, precise paths are defined on the layout surface, on which conductors carrying electrical signals are run.
- Routing takes up almost 30% of the design time, and a large percentage of layout area.
- We first take up the problem of grid routing.

# What is Grid Routing?

- The layout surface is assumed to be made up of a rectangular array of grid cells.
- Some of the grid cells act as obstacles.
  - Blocks that are placed on the surface.
  - Some nets that are already laid out.
- Objective is to find out a path (sequence of grid cells) for connecting two points belonging to the same net.
- Two broad class of algorithms:
  - Maze routing algorithms.
  - Line search algorithms.


T
S

# Problem Definition

- The general routing problem is defined as follows.
- Given:
  - A set of blocks with pins on the boundaries.
  - A set of signal nets.
  - Locations of blocks on the layout floor.
- Objective:
  - Find suitable paths on the available layout space, on which wires are run to connect the desired set of pins.
  - Minimize some given objective function, subject to given constraints.

# Contd.

- **Types of constraints:**
  - **Minimum width of routing wires.**
  - **Minimum separation between adjacent wires.**
  - **Number of routing layers available.**
  - **Timing constraints.**

# Grid Routing Algorithms

1. Maze running algorithm
   - Lee's algorithm
   - Hadlock's algorithm
2. Line search algorithm
   - Mikami-Tabuchi's algorithm
   - Hightower's algorithm
3. Steiner tree algorithm

# Maze Running Algorithms

- The entire routing surface is represented by a 2-D array of grid cells.
  - All pins, wires and edges of bounding boxes that enclose the blocks are aligned with respect to the grid lines.
  - The segments on which wires run are also aligned.
  - The size of grid cells is appropriately defined.
    - Wires belonging to different nets can be routed through adjacent cells without violating the width and spacing rules.
- Maze routers connect a single pair of points at a time.
  - By finding a sequence of adjacent cells from one point to the other.

# Lee's Algorithm

- **The most common maze routing algorithm.**
- **Characteristics:**
  - **If a path exists between a pair of points S and T, it is definitely found.**
  - **It always finds the shortest path.**
  - **Uses breadth-first search.**
- **Time and space complexities are $O(N^2)$ for a grid of dimension $N \times N$.**

# Phase 1 of Lee's Algorithm

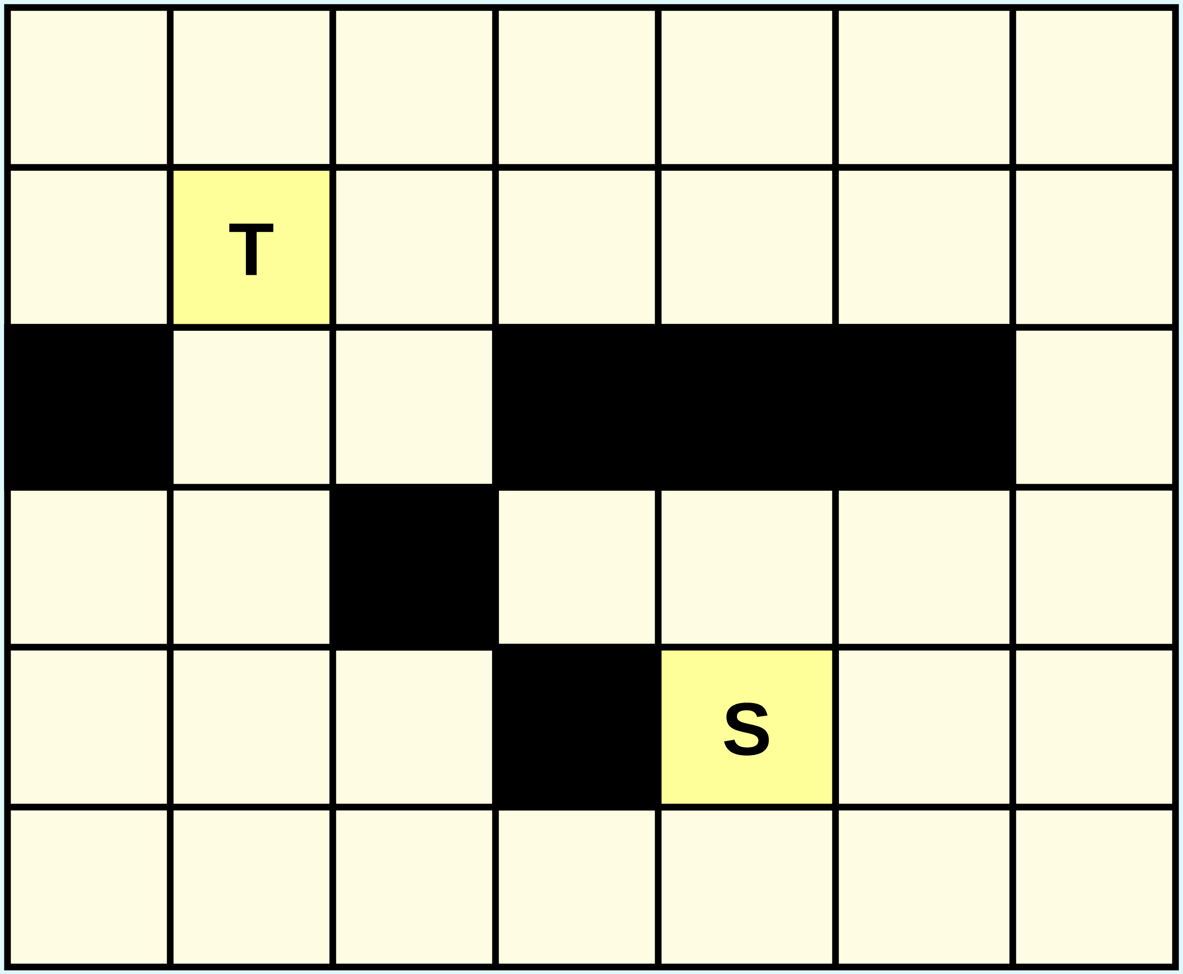
- **Wave propagation phase**
  - Iterative process.
  - During step i, non-blocking grid cells at Manhattan distance of i from grid cell S are all labeled with i.
  - Labeling continues until the target grid cell T is marked in step L.
    - L is the length of the shortest path.
  - The process fails if:
    - T is not reached and no new grid cells can be labeled during step i.
    - T is not reached and i equals M, some upper bound on the path length.

# Phase 2 of Lee's Algorithm

- **Retrace phase**
  - Systematically backtrack from the target cell T back towards the source cell S.
  - If T was reached during step i, then at least one grid cell adjacent to it will be labeled i-1, and so on.
  - By tracing the numbered cells in descending order, we can reach S following the shortest path.
    - There is a choice of cells that can be made in general.
    - In practice, the rule of thumb is not to change the direction of retrace unless one has to do so.
    - Minimizes number of bends.

# Phase 3 of Lee's Algorithm

- **Label clearance**
  - All labeled cells except those corresponding to the path just found are cleared.
  - Search complexity is as involved as the wave propagation step itself.


T
S

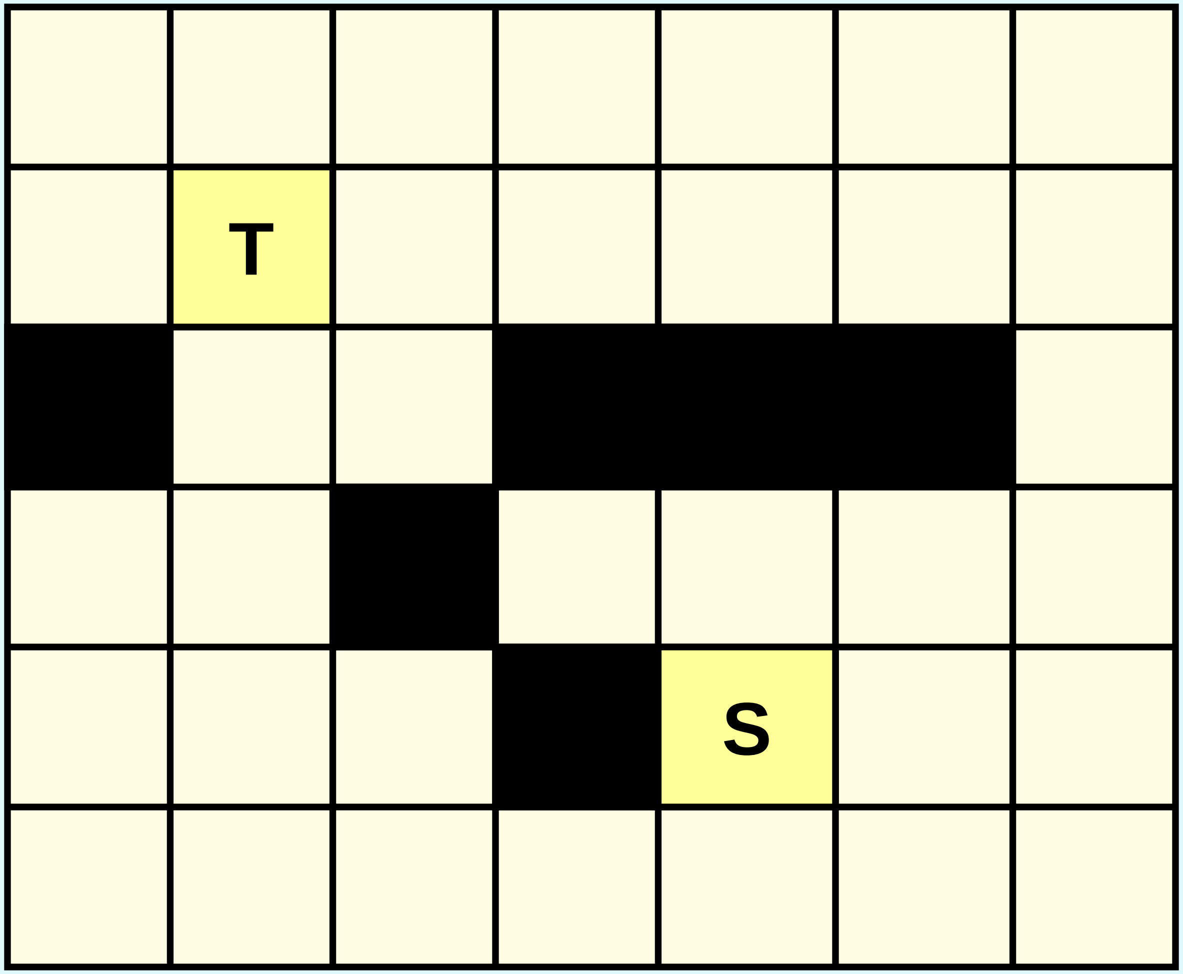
- **Memory Requirement**
  - **Each cell needs to store a number between 1 and L, where L is some bound on the maximum path length.**
  - **One bit combination to denote empty cell.**
  - **One bit combination to denote obstacles.**

**$\log_2(L+2)$ bits per cell**

- **Improvements:**
  - Instead of using the sequence 1,2,3,4,5,….. for numbering the cells, the sequence 1,2,3,1,2,3,… is used.
    - For a cell, labels of predecessors and successors are different. So tracing back is easy.

      $\log_2(3+2) = 3$ bits per cell.
  - Use the sequence 0,0,1,1,0,0,1,1,…..
    - Predecessors and successors are again different.

      $\log_2(2+2) = 2$ bits per cell.


T
S

# Reducing Running Time

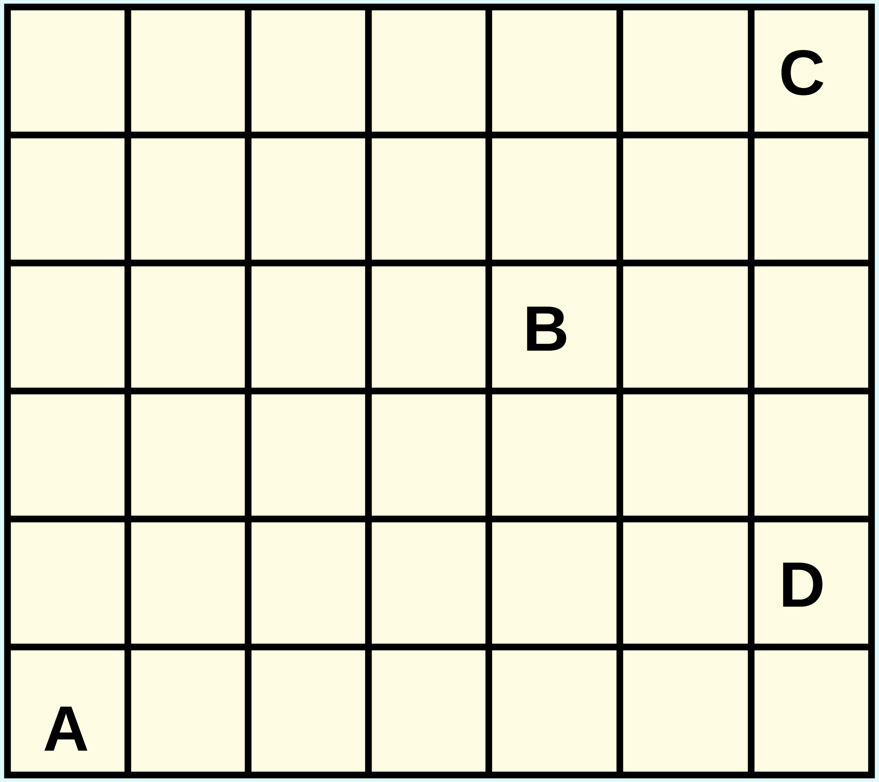
- **Starting point selection**
  - Choose the starting point as the one that is farthest from the center of the grid.
- **Double fan-out**
  - Propagate waves from both the source and the target cells.
  - Labeling continues until the wavefronts touch.
- **Framing**
  - An artificial boundary is considered outside the terminal pairs to be connected.
  - 10-20% larger than the smallest bounding box.

# Illustration

# Connecting Multi-point Nets

- **A multi-pin net consists of three or more terminal points to be connected.**
- **Extension of Lee's algorithm:**
  - **One of the terminals of the net is treated as source, and the rest as targets.**
  - **A wave is propagated from the source until one of the targets is reached.**
  - **All the cells in the determined path are next labeled as source cells, and the remaining unconnected terminals as targets.**
  - **Process continues.**

# Illustration



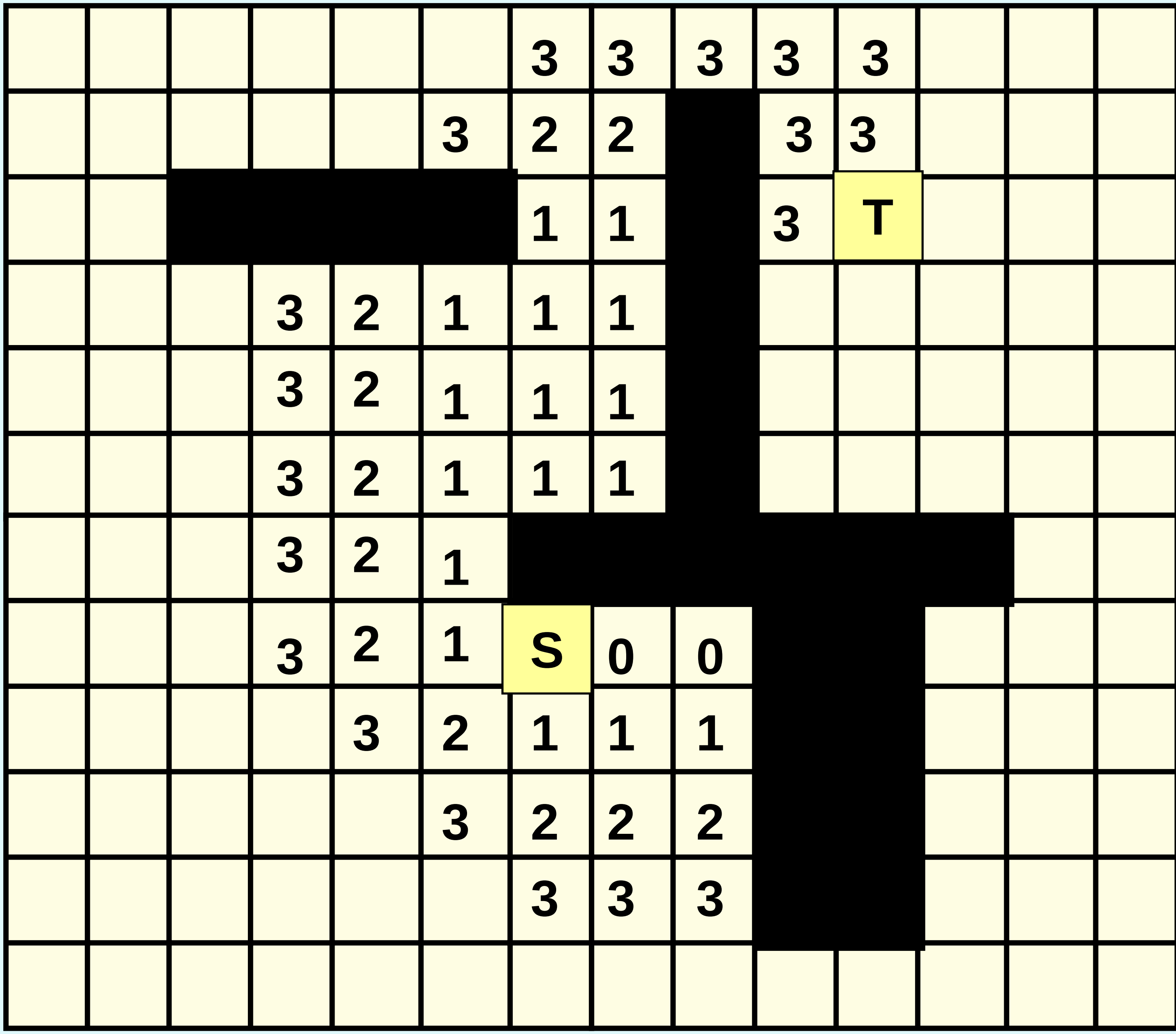
# Hadlock's Algorithm

- **Uses a new method for cell labeling called <u>detour numbers</u>.**
  - **A goal directed search method.**
  - **The detour number d(P) of a path P connecting two cells S and T is defined as the number of grid cells directed away from its target T.**
  - **The length of the path P is given by**

$$\text{len}(P) = MD(S,T) + 2\, d(P)$$

**where MD (S,T) is the Manhattan distance between S and T.**

- **The cell filling phase of Lee's algorithm can be modified as follows:**
  - **Fill a cell with the detour number with respect to a specified target T (not by its distance from source).**
  - **Cells with smaller detour numbers are expanded with high priority.**
- **Path retracing is of course more complex, and requires some degree of searching.**


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

- **Advantages:**
  - Number of grid cells filled up is considerably less as compared to Lee's algorithm.
  - Running time for an NxN grid ranges from $O(N)$ to $O(N^2)$.
    - Depends on the obstructions.
    - Also locations of S and T.

# Line Search Algorithm

- In maze running algorithms, the time and space complexities are too high.
- An alternative approach is called line searching, which overcomes this drawback.
- Basic idea:
  - Assume no obstacles for the time being.
  - A vertical line drawn through S and a horizontal line passing though T will intersect.
    - Manhattan path between S and T.
  - In the presence of obstacles, several such lines need to be drawn.

# Contd.

- **Line search algorithms do not guarantee finding the optimal path.**
  - **May need several backtrackings.**
  - **Running time and memory requirements are significantly less.**
  - **Routing area and paths are represented by a set of line segments.**
    - **Not as a matrix as in Lee's or Hadlock's algorithm.**

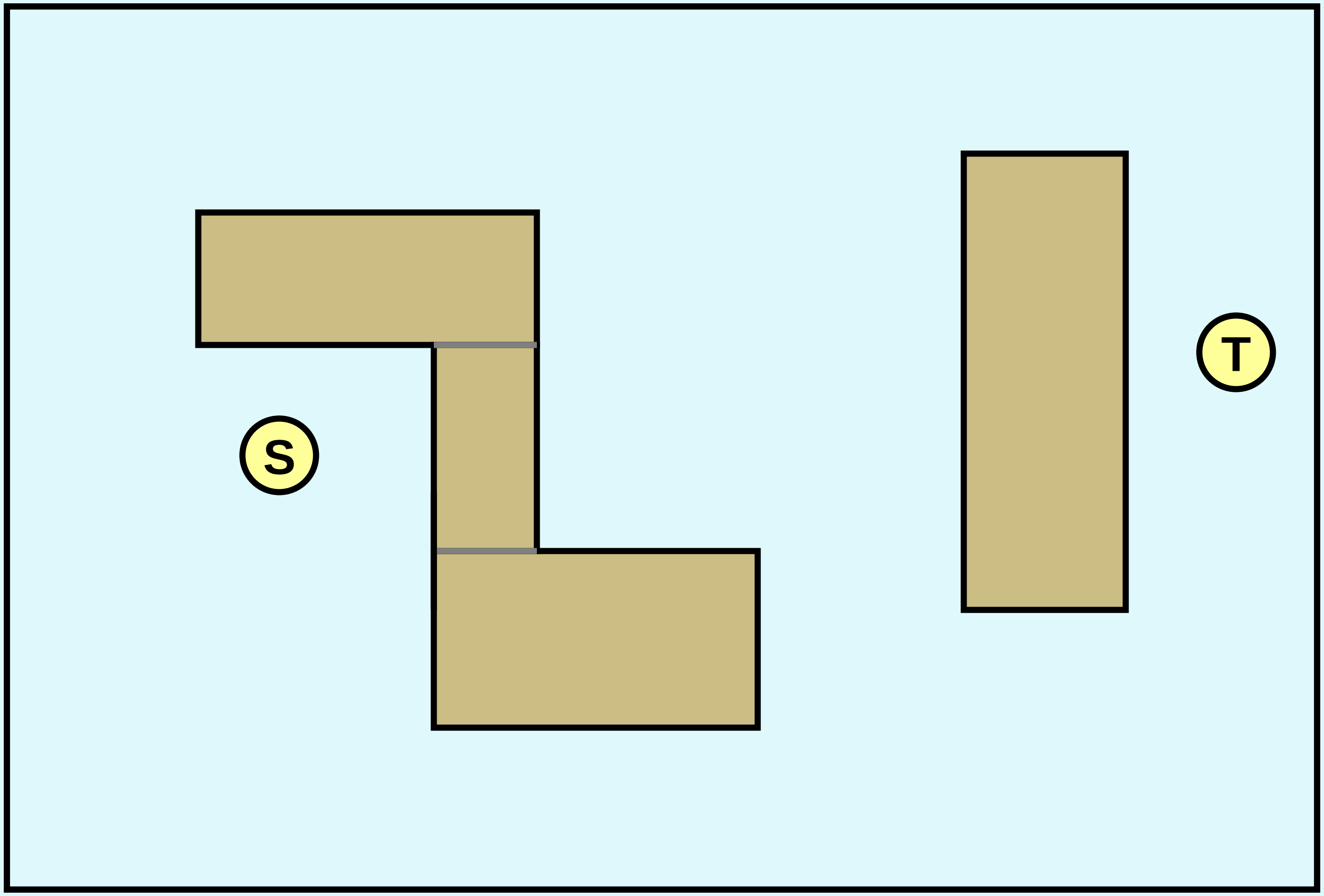
# Mikami-Tabuchi's Algorithm

- Let S and T denote a pair of terminals to be connected.
- <u>Step 1:</u>
  - Generate four lines (two horizontal and two vertical) passing through S and T.
  - Extend these lines till they hit obstructions or the boundary of the layout.
  - If a line generated from S intersects a line generated from T, then a connecting path is found.
  - If they do not intersect, they are identified as trial lines of level zero.
    - Stored in temporary storage for further processing.

# Contd.

- **Step i of Iteration:**
  - Pick up trial lines of level i, one at a time.
    - Along the trial line, all its grid points are traced.
    - Starting from these grid points, new trial lines (of level i+1)are generated perpendicular to the trial line of level i.
  - If a trial line of level i+1 intersects a trial line (of any level) from the other terminal point, the connecting path can be found.
    - By backtracing from the intersection point to S and T.
    - Otherwise, all trial lines of level (i+1) are added to temporary storage, and the procedure repeated.
- The algorithm guarantees to find a path if it exists.

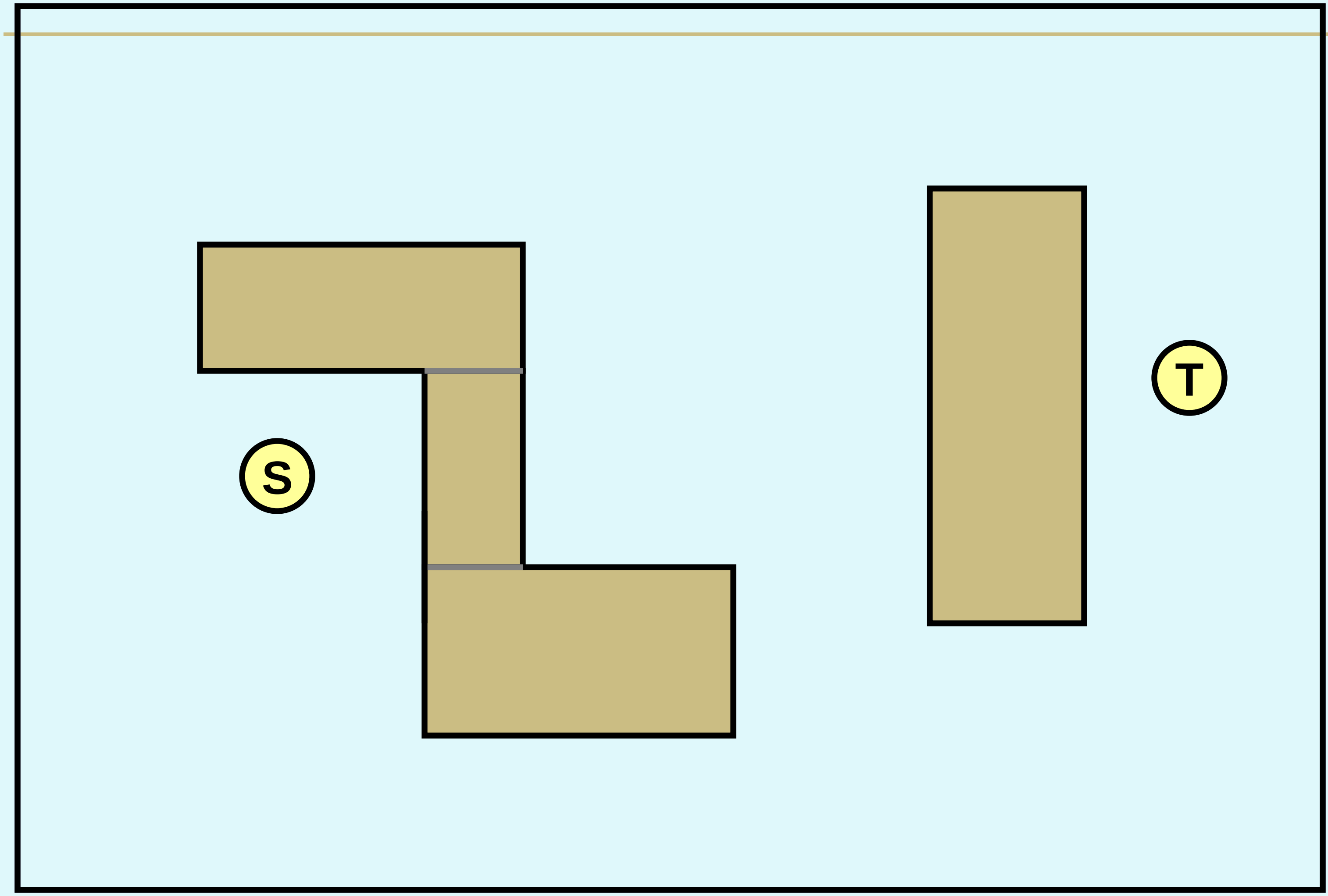
# Illustration



# Hightower's Algorithm

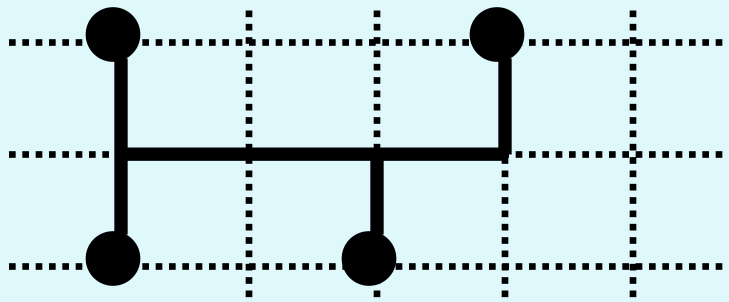
- Similar to Mikami-Tabuchi's algorithm.
  - Instead of generating all line segments perpendicular to a trial line, consider only those lines that can be extended beyond the obstacle which blocked the preceding trial line.
- <u>Steps of the algorithm:</u>
  - Pass a horizontal and a vertical line through source and target points (called first-level probes).
  - If the source and the target lines meet, a path is found.
  - Otherwise, pass a perpendicular line to the previous probe whenever it intersects an obstacle.
    - Concept of escape point and escape line.

# Illustration



# Steiner Trees

- A tree interconnecting a set $P=\{P_1,\ldots,P_n\}$ of specified points in the rectilinear plane and some arbitrary points is called a (rectilinear) Steiner tree of P.



- A Steiner tree with minimum total cost is called a Steiner minimal tree (SMT).
  - The general SMT problem is NP-hard.

# Steiner Tree Based Algorithms

- **Minimum length Steiner trees:**
  - Goal is to minimize the sum of the length of the edges of the tree.
  - Both exact and approximate versions exist.
- **Weigted Steiner trees:**
  - Given a plane partitioned into a collection of weighted regions, an edge with length L in a region with weight W has cost LW.
- **Steiner trees with arbitrary orientations:**
  - Allows lines in non-rectilinear directions like $+45^o$ and $-45^o$.

# Global Routing

# Basic Idea

- The routing problem is typically solved using a two-step approach:
  - Global Routing
    - Define the routing regions.
    - Generate a tentative route for each net.
    - Each net is assigned to a set of routing regions.
    - Does not specify the actual layout of wires.
  - Detailed Routing
    - For each routing region, each net passing through that region is assigned particular routing tracks.
    - Actual layout of wires gets fixed.
    - Associated subproblems: channel routing and switchbox routing.

# Routing Regions

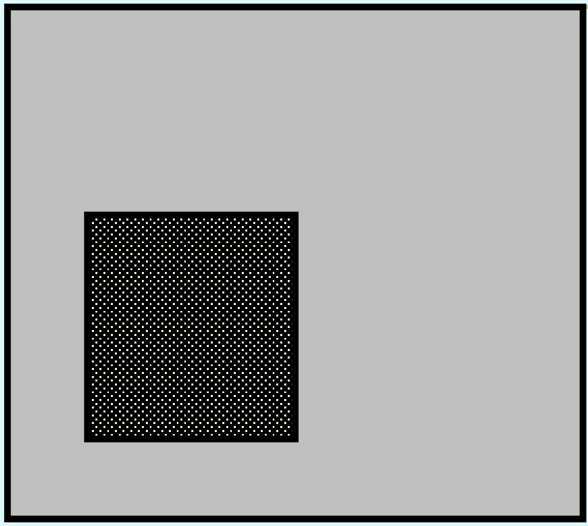
- Regions through which interconnecting wires are laid out.
- How to define these regions?
  - Partition the routing area into a set of non-intersecting rectangular regions.
  - Types of routing regions:
    - Horizontal channel: parallel to the x-axis with pins at their top and bottom boundaries.
    - Vertical channel: parallel to the y-axis with pins at their left and right boundaries.
    - Switchbox: rectangular regions with pins on all four sides.

- **Points to note:**
  - Identification of routing regions is a crucial first step to global routing.
  - Routing regions often do not have pre-fixed capacities.
  - The order in which the routing regions are considered during detailed routing plays a vital part in determining overall routing quality.

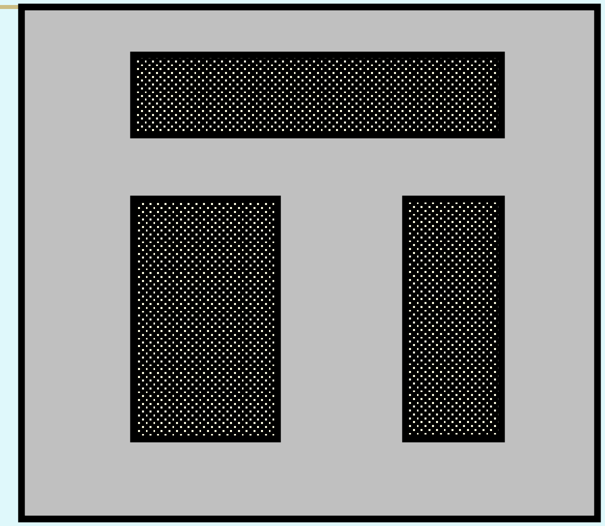
# Types of Channel Junctions

- Three types of channel junctions may occur:
  - **L-type**:
    - Occurs at the corners of the layout surface.
    - Ordering is not important during detailed routing.
    - Can be routed using channel routers.
  - **T-type**:
    - The leg of the "T" must be routed before the shoulder.
    - Can be routed using channel routers.
  - **+-type**:
    - More complex and requires switchbox routers.
    - Advantageous to convert +-junctions to T-junctions.

# Illustrations

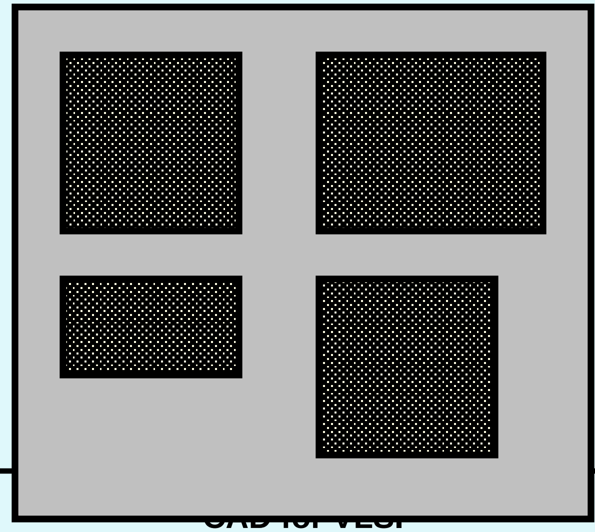


L Type



T Type

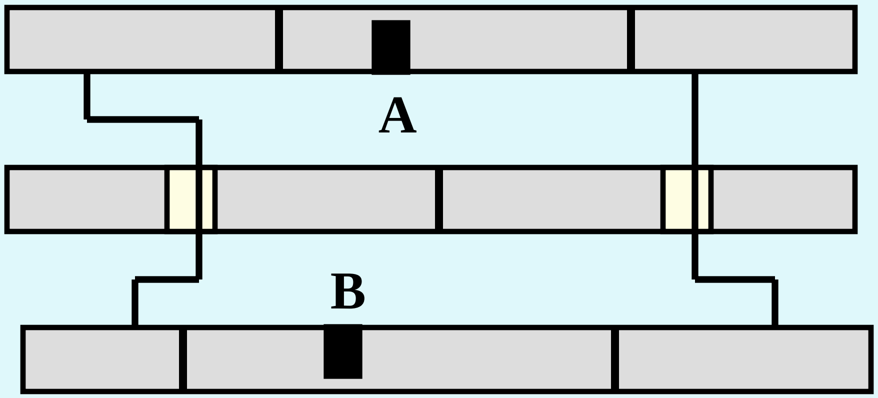
+ Type



# Design Style Specific Issues

- **Full Custom**
  - The problem formulation is similar to the general formulation as discussed.
    - All the types of routing regions and channels junctions can occur.
  - Since channels can be expanded, some violation of capacity constraints are allowed.
  - Major violation in constraints are, however, not allowed.
    - May need significant changes in placement.

- **Standard Cell**
  - At the end of the placement phase
    - Location of each cell in a row is fixed.
    - Capacity and location of each feed-through is fixed.
    - Feed-throughs have predetermined capacity.
  - Only horizontal channels exist.
    - Channel heights are not fixed.
  - Insufficient feed-throughs may lead to failure.
  - Over-the-cell routing can reduce channel height, and change the global routing problem.




**A cannot be connected to B**

- **Gate Array**
  - The size and location of cells are fixed.
  - Routing channels & their capacities are also fixed.
  - Primary objective of global routing is to guarantee routability.
  - Secondary objective may be to minimize critical path delay.