第8章　布线（Routing）

8.1 布线概述、问题、算法

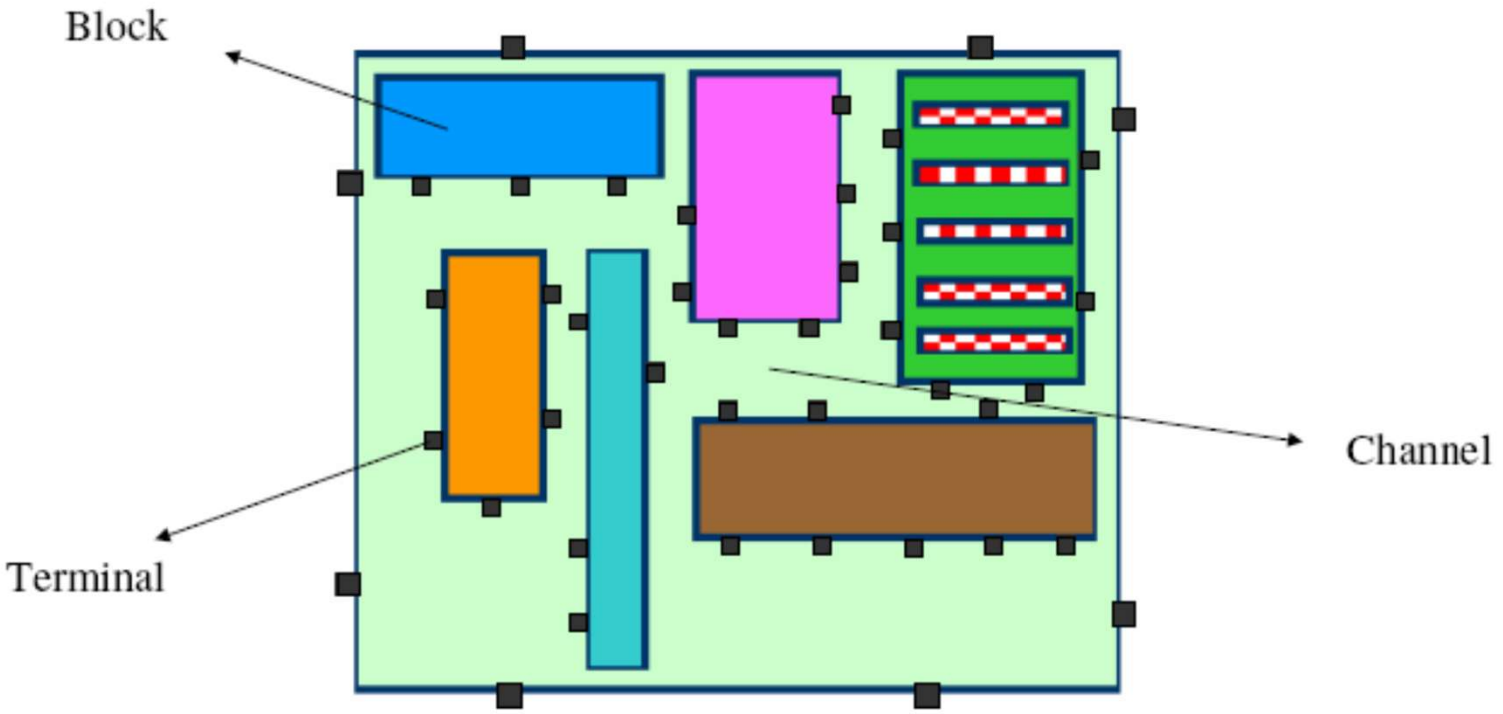
8.2 单条线网布线算法

8.3 多条线网布线算法

一、布线问题及算法

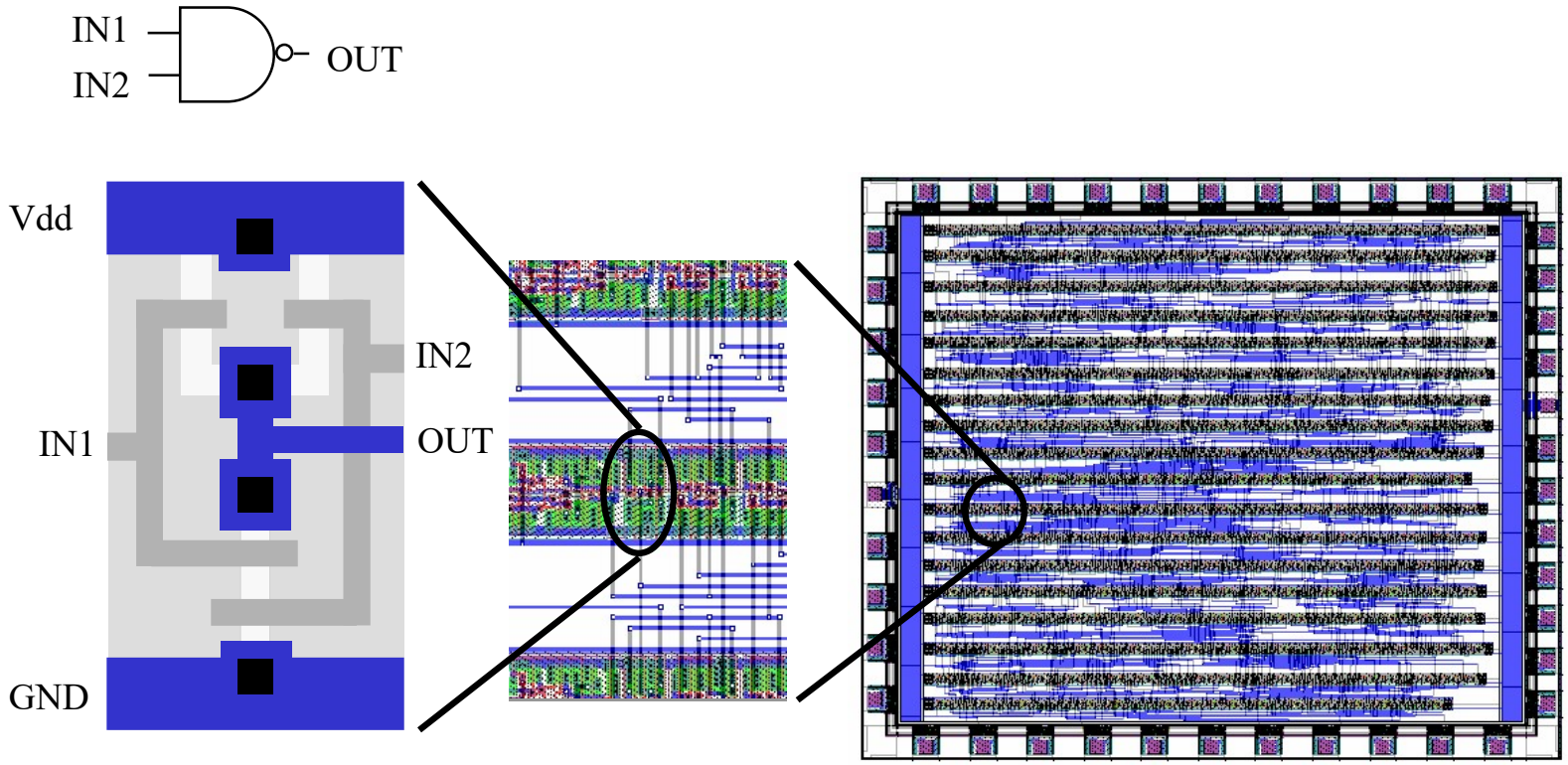
- 布线问题
- 迷宫算法(Maze Routing Algorithms)
- 线探索法(Line Probe Algorithms)
- Pattern-Based Routing
- 布线顺序的影响及其处理
- 线性规划/拆线重布策略
- 并行布线
- Negotiation-Based Routing

互连线需要实现电等价连接



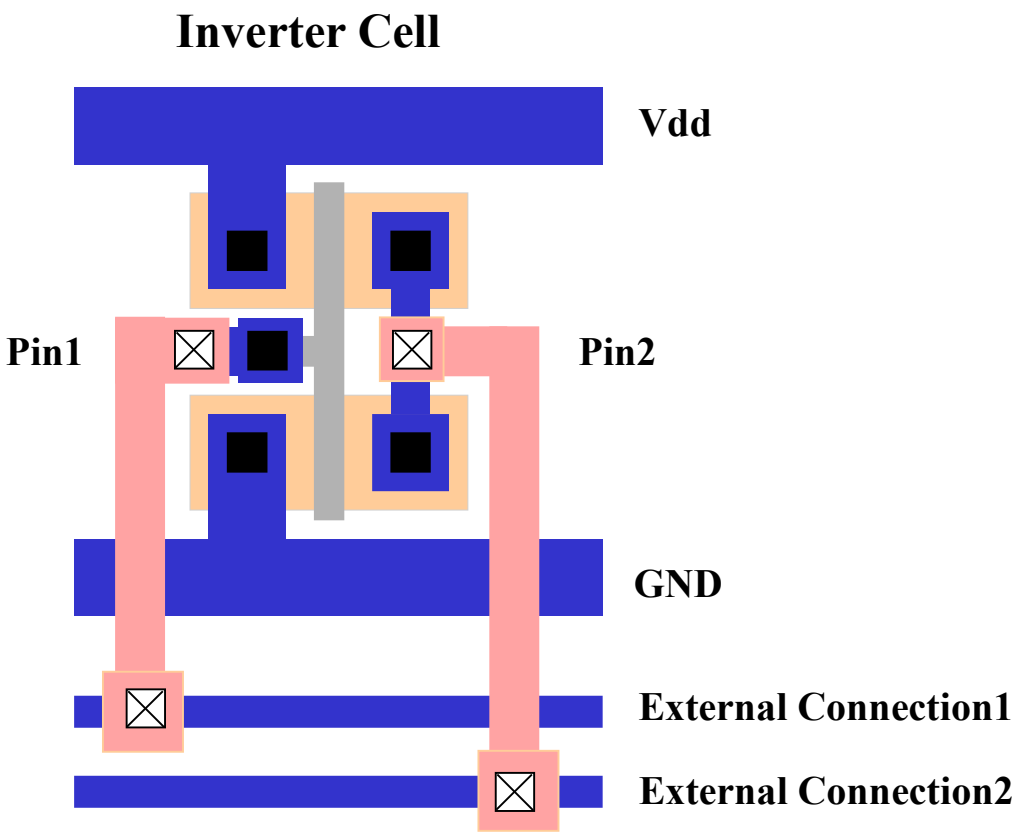
Layout of circuit blocks and pins after placement

Standard cell routing



标准单元连线

Layout of an inverter cell with external connections



要求完成下面的连接：

- Pin1---External Connection1
- Pin2---External Connection2

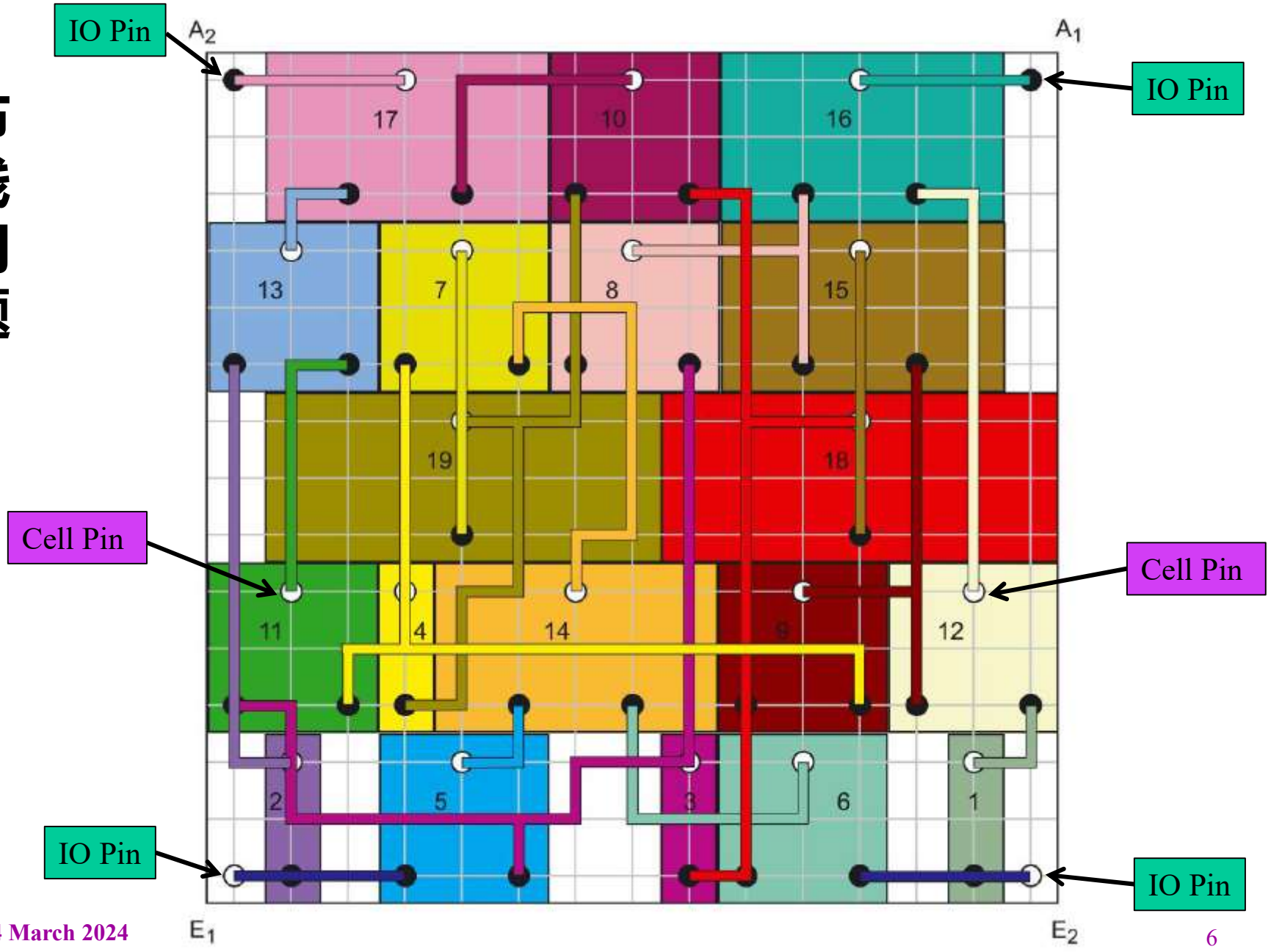
Metal2 ■ Contact

Metal1 ☒ Via

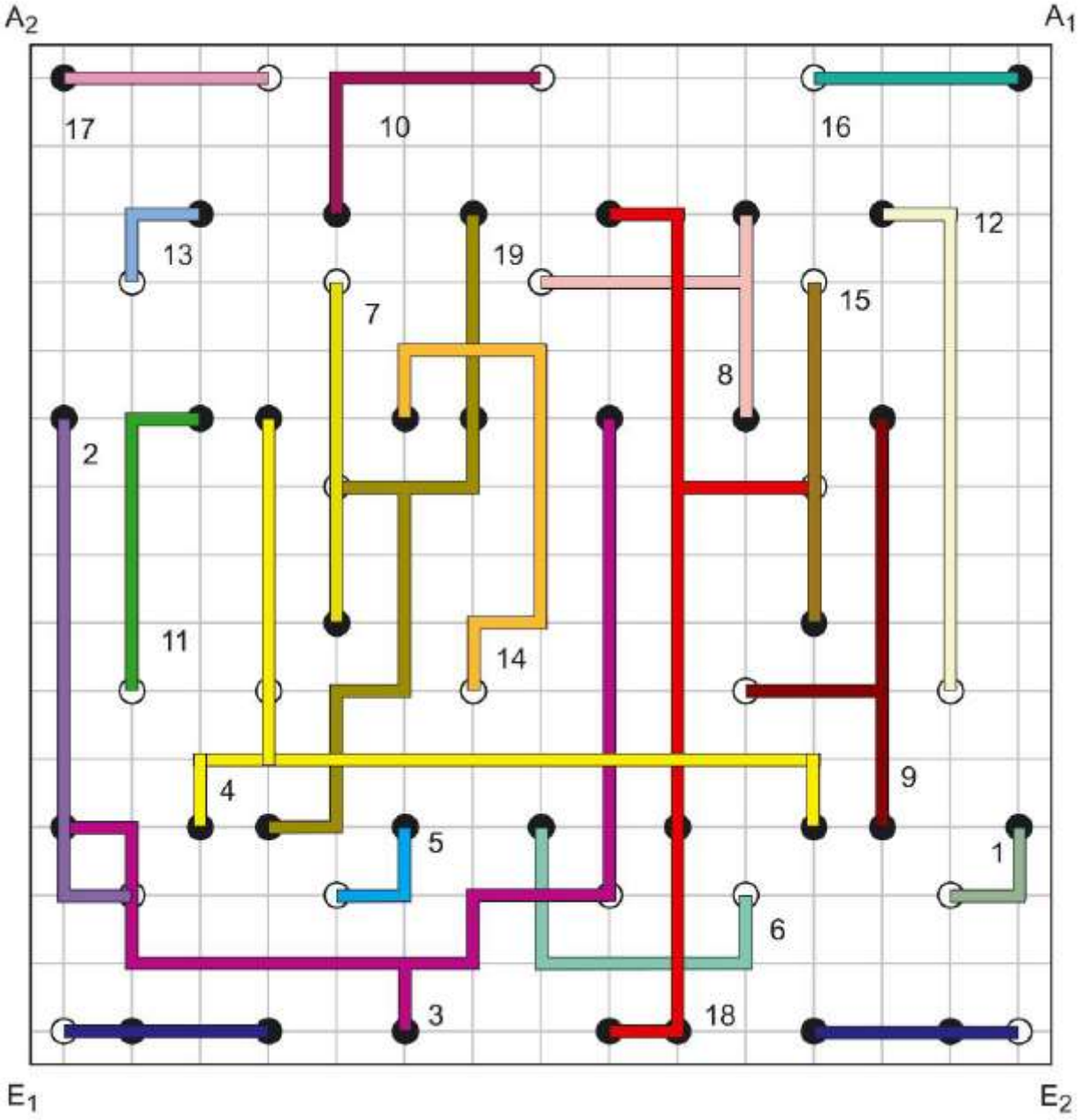
polysilicon

p/n diffusion

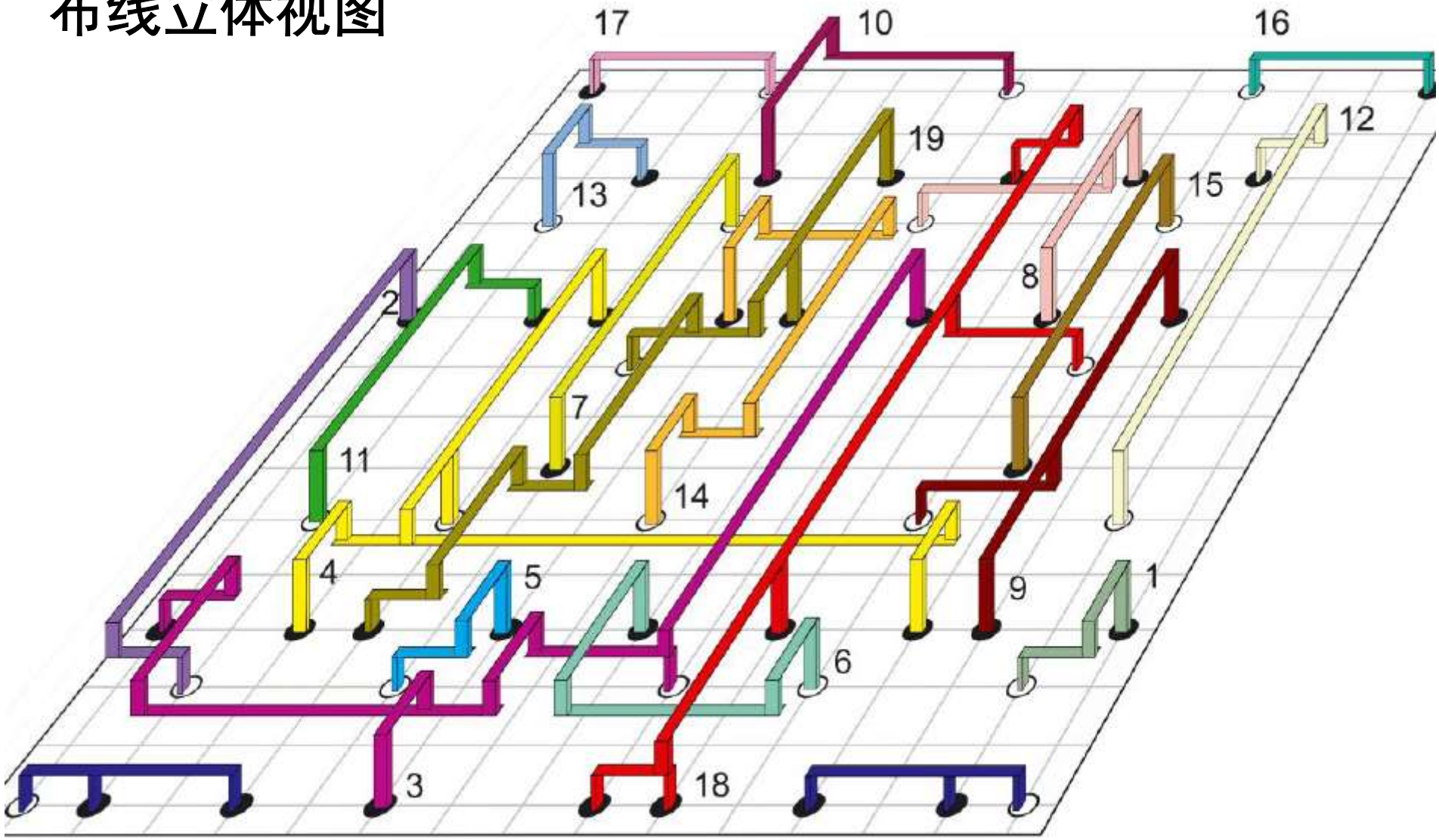
布线问题

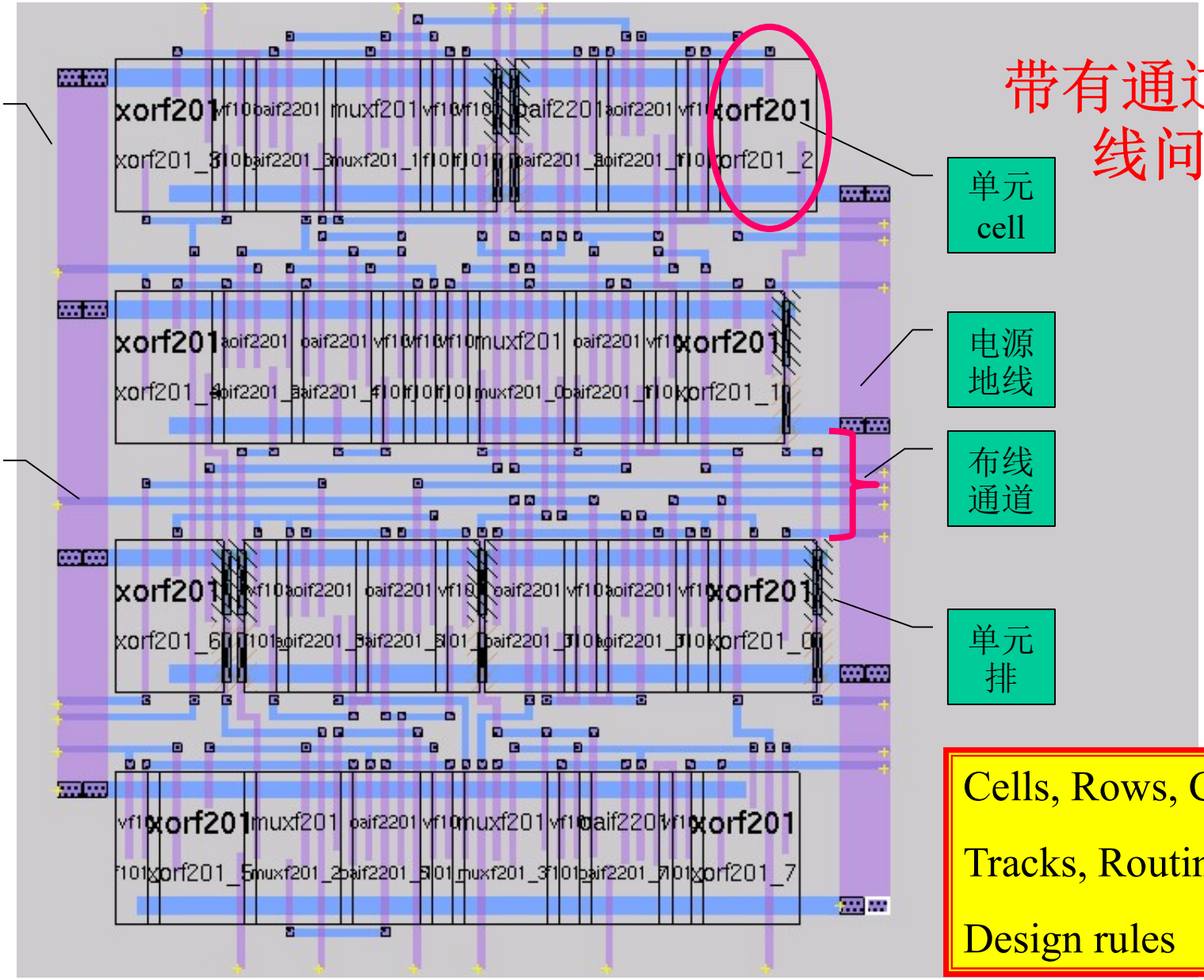

IO Pin
A2
A1
IO Pin
17
10
16
13
7
8
15
19
18
Cell Pin
Cell Pin
11
4
14
9
12
2
5
3
6
1
IO Pin
IO Pin
E1
E2

布线抽象



布线立体视图





带有通道的布线问题

Cells, Rows, Channels

Tracks, Routing layers

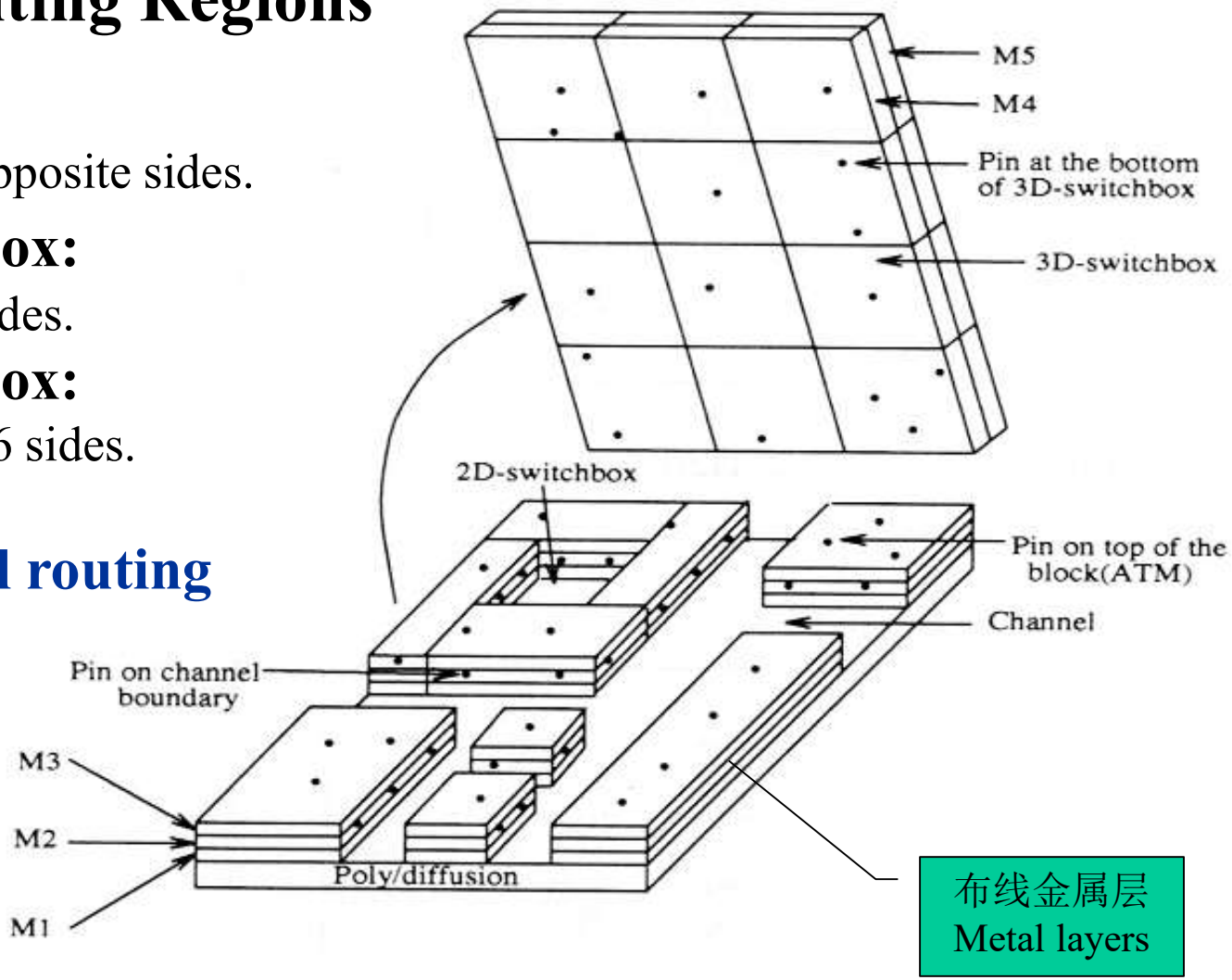
Design rules

布线问题

3 Types of Routing Regions

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

- **Over the cell routing**



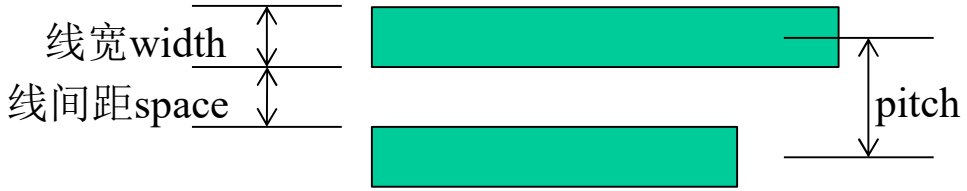
布线问题

■ 布线容量(Capacity)

每根互连线是有宽度的多边形金属线，布线区内能容下的最大走线数，由设计规则、布线区大小和线宽决定。

■ 布线设计规则约束：

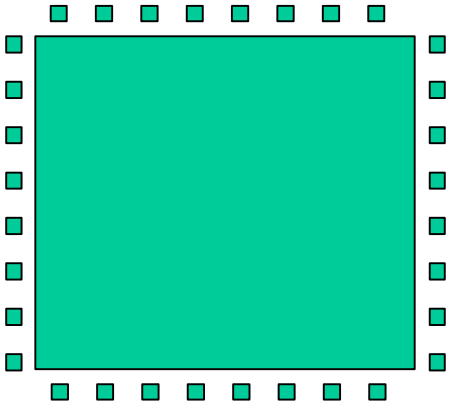
–最小线宽和最小线间距



■ 布线的目标

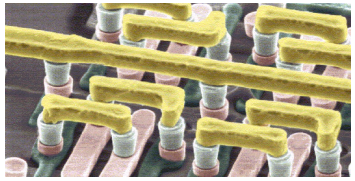
–连线总长最短或芯片面积最小。

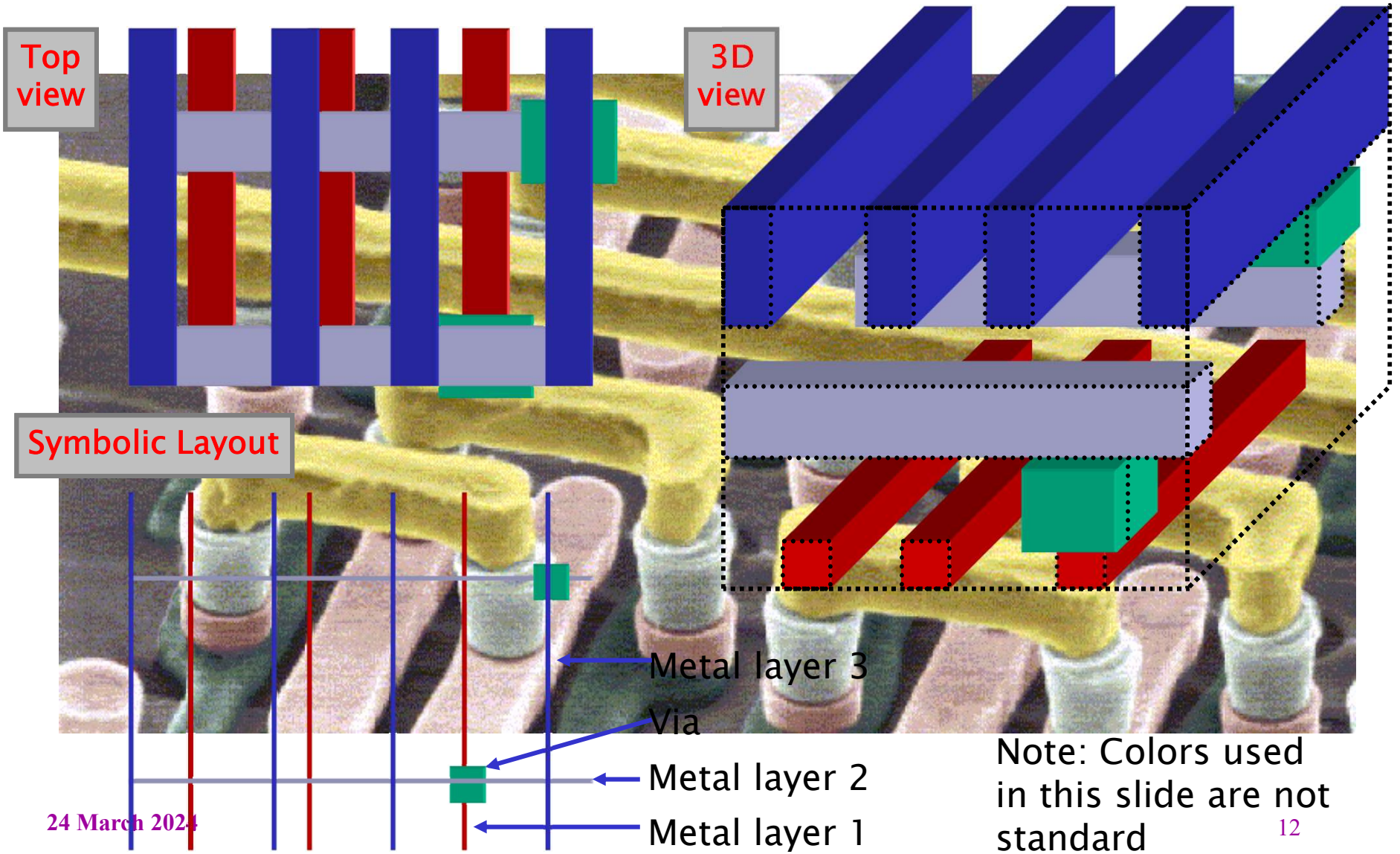
–最小化芯片的时延以达到优化整个芯片的性能。



在布线资源约束下，完成连线任务，实现布线目标。

Routing Anatomy





Note: Colors used in this slide are not standard

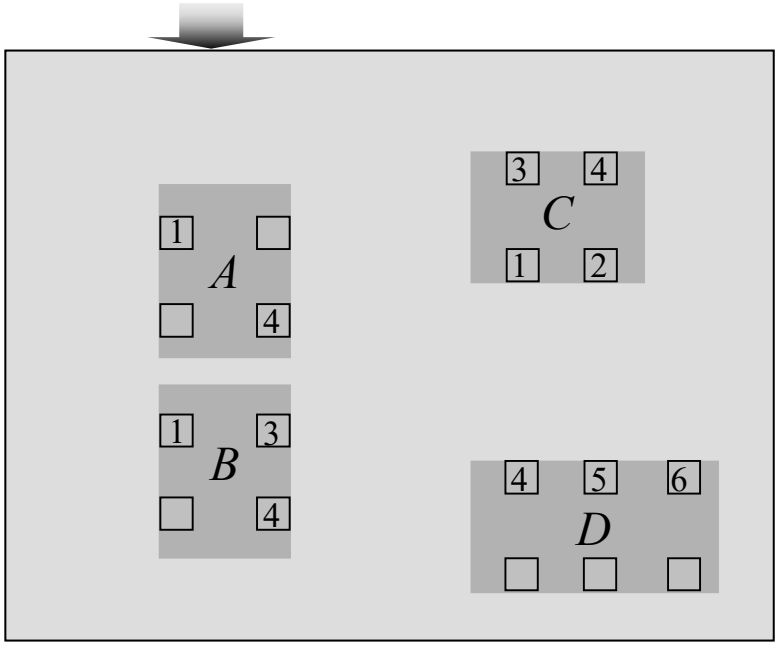
布线问题

Netlist:

$N_1 = \{C_4, D_6, B_3\}$
$N_2 = \{D_4, B_4, C_1, A_4\}$
$N_3 = \{C_2, D_5\}$
$N_4 = \{B_1, A_1, C_3\}$

Technology Information
(Design Rules)

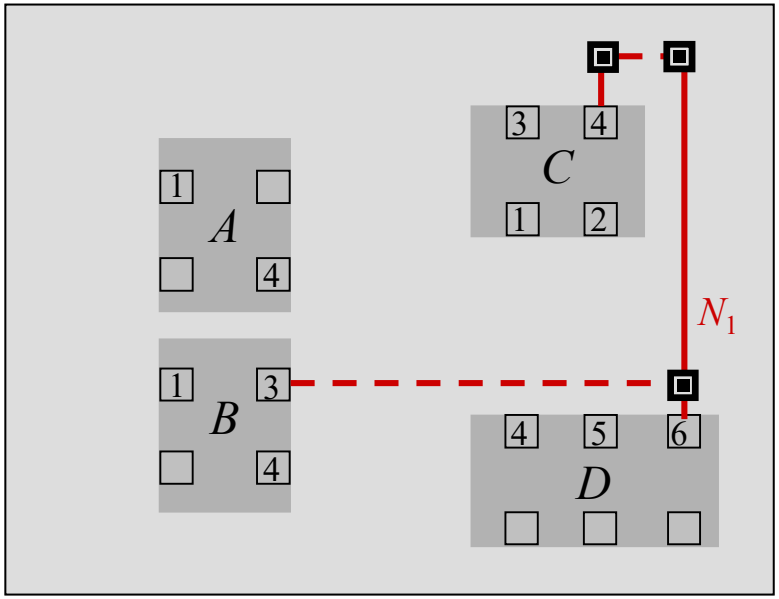
Placement result



Netlist:

$N_1 = \{C_4, D_6, B_3\}$
$N_2 = \{D_4, B_4, C_1, A_4\}$
$N_3 = \{C_2, D_5\}$
$N_4 = \{B_1, A_1, C_3\}$

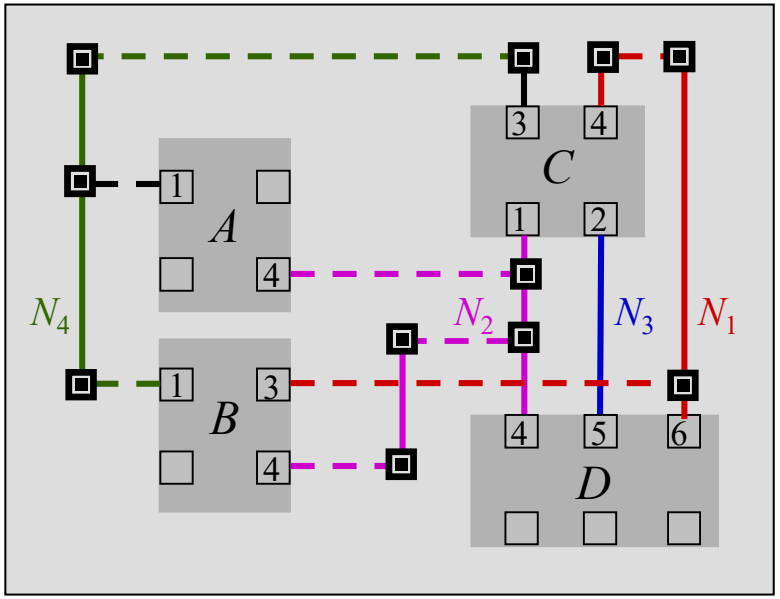
Technology Information
(Design Rules)



Netlist:

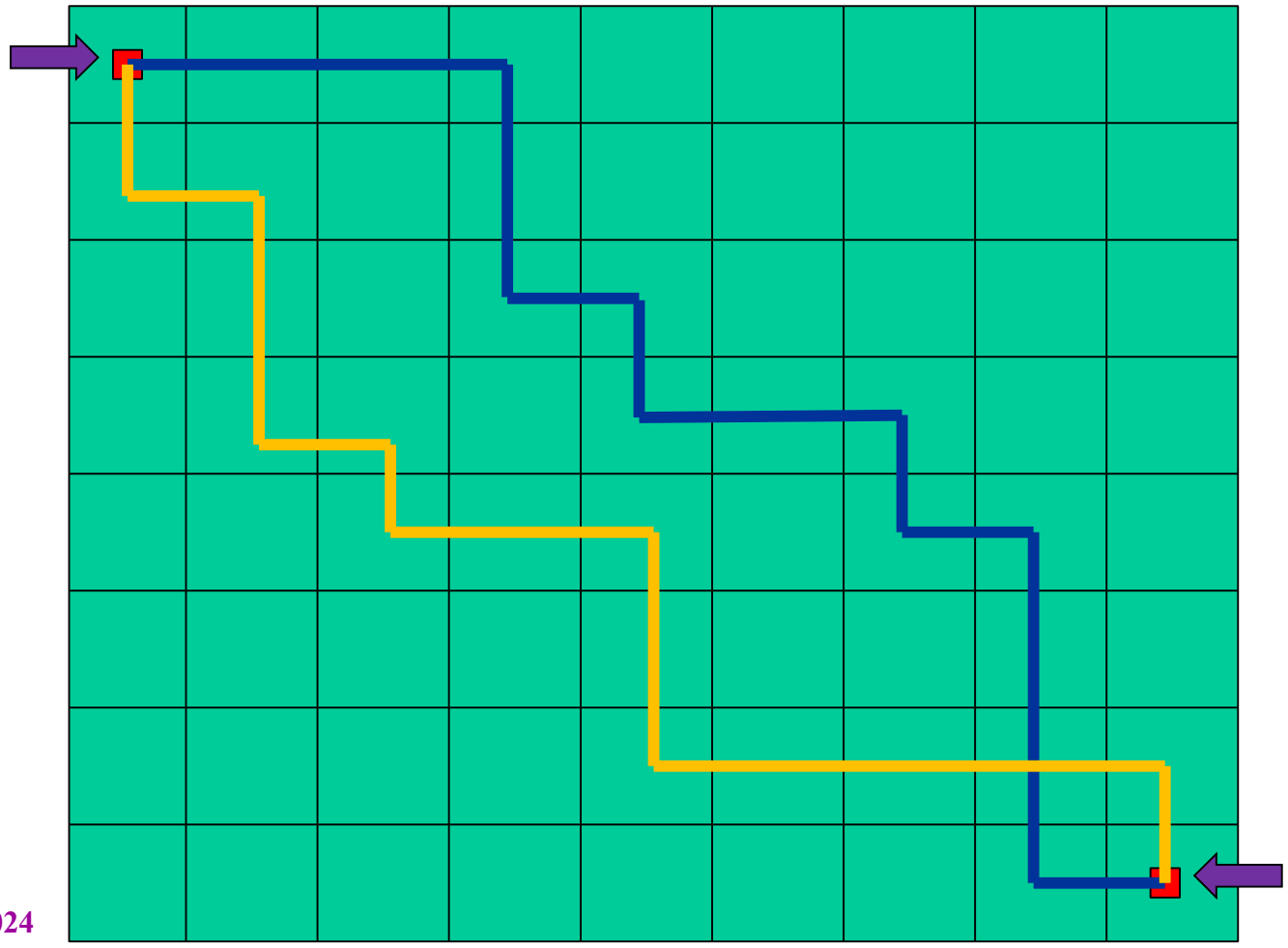
$N_1 = \{C_4, D_6, B_3\}$
$N_2 = \{D_4, B_4, C_1, A_4\}$
$N_3 = \{C_2, D_5\}$
$N_4 = \{B_1, A_1, C_3\}$

Technology Information
(Design Rules)



两点间最短路径布线

有很多种可能的路径

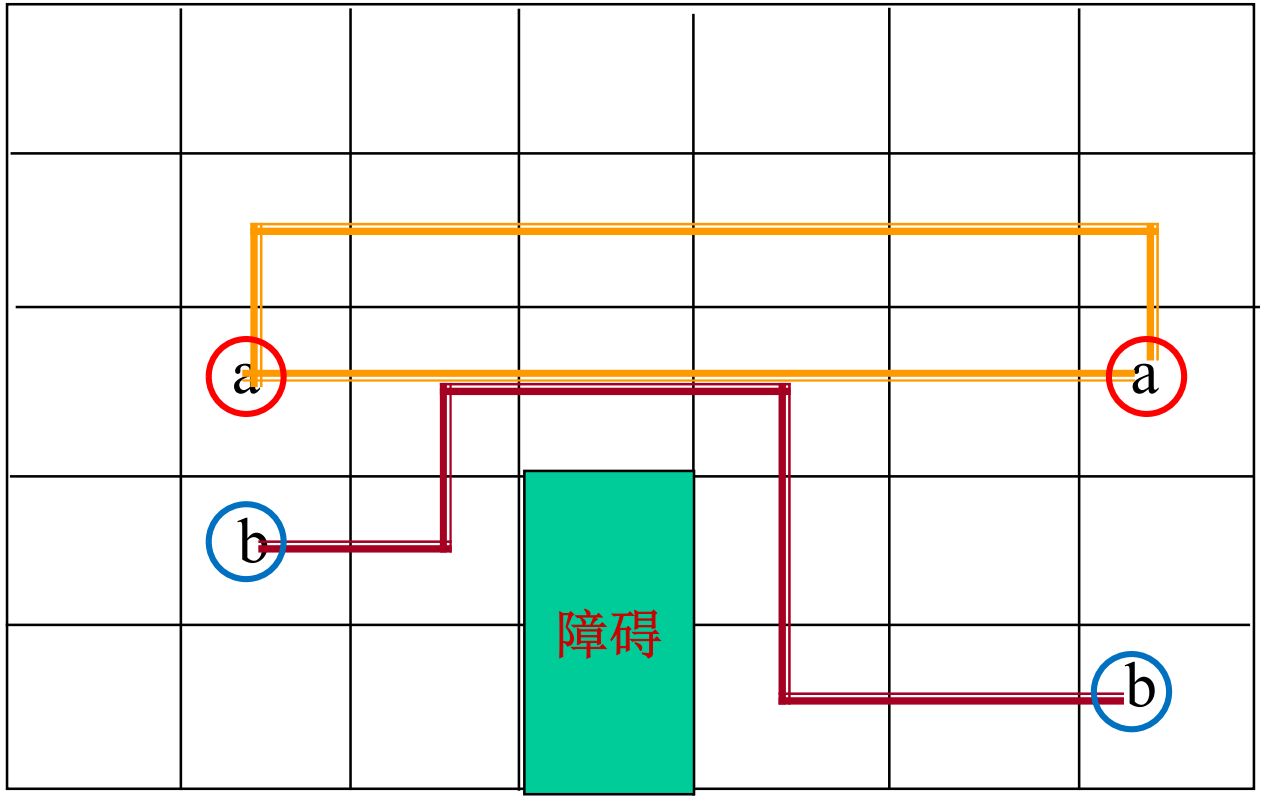


单条线网及多条线网布线问题

在资源约束情况下，多条线网如何完成布线任务？

- 布线资源是共同的，而且是有限的；
- 每条线网都要对布线资源进行占用。

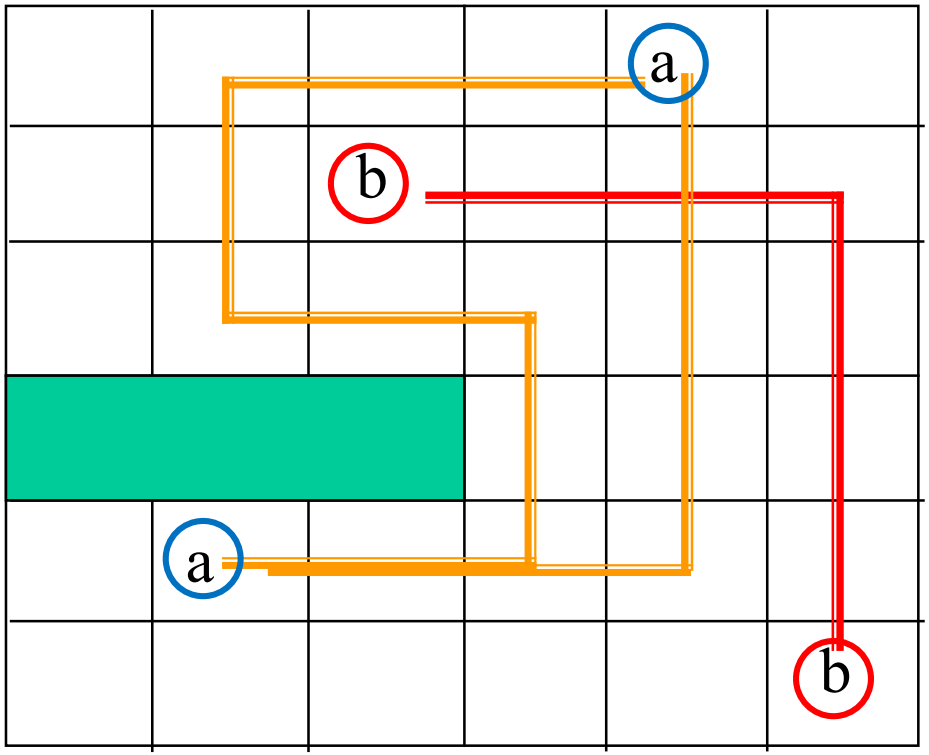
单层布线



a 线网优先

b 线网优先

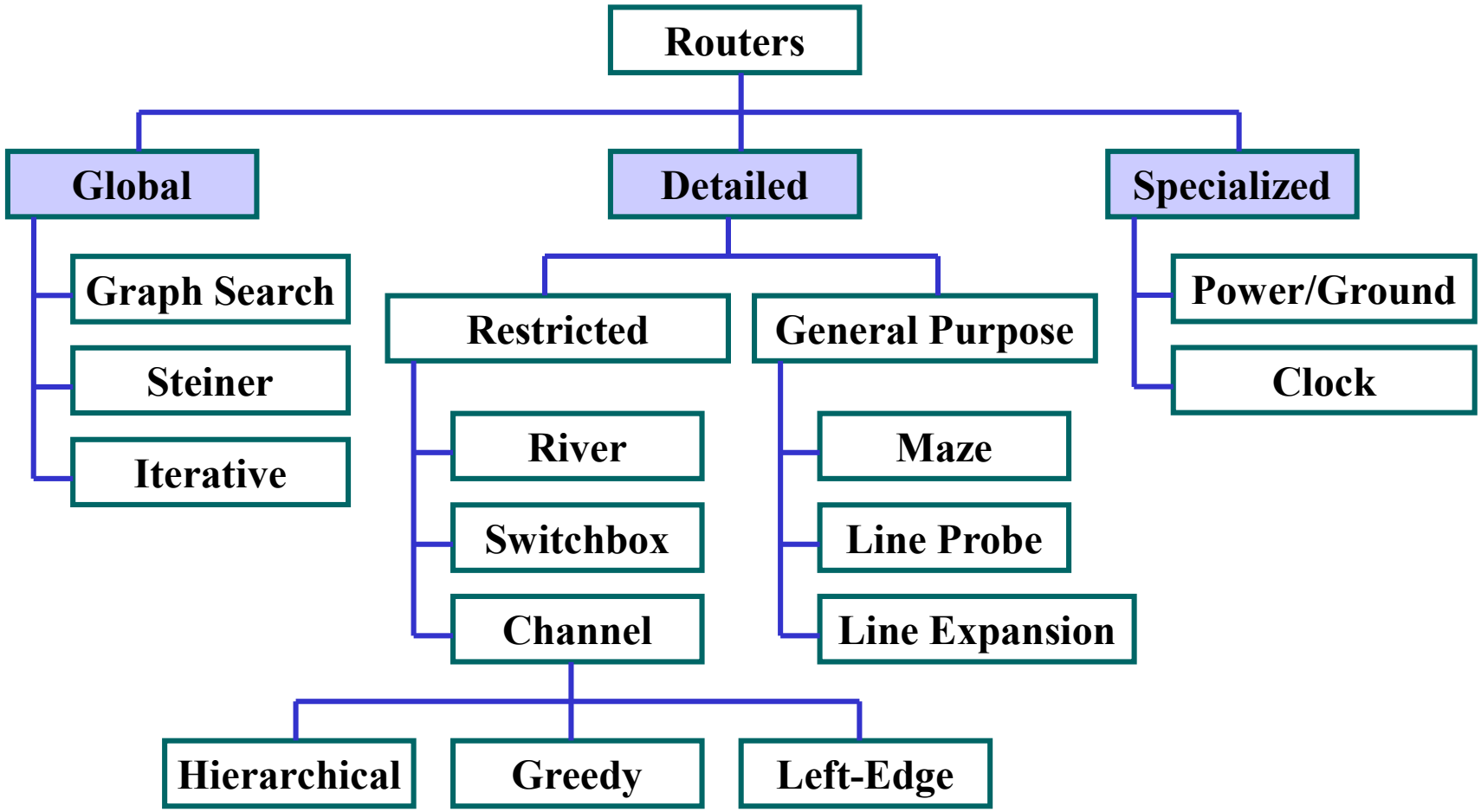
单层布线



a 线网
优先

b 线网
优先

布线问题分类

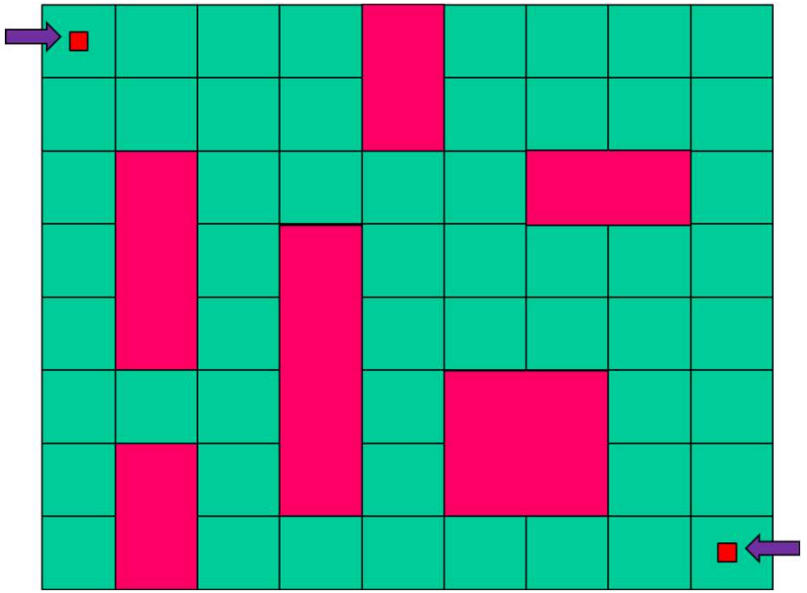

Routers
Global
Detailed
Specialized
Graph Search
Steiner
Iterative
Restricted
General Purpose
Power/Ground
Clock
River
Switchbox
Channel
Maze
Line Probe
Line Expansion
Hierarchical
Greedy
Left-Edge

二、单条线网布线算法

1、迷宫布线算法

2、线探索布线方法

3、模式布线方法



1. 迷宫算法 (Maze Routing Algorithms)

C. Y. Lee, An Algorithm for Connections and Its Applications,
IRE Transactions on Electronic Computers, 1961, pp:346-365

- Lee于1961年提出了一个两端线网的布线算法，称为李氏(Lee)算法。
- 有许多对该算法的改进，包括提高速度和减小空间。Lee算法及其改进形成了一类布线算法，称为迷宫算法(Maze Routing Algorithms)。
- 迷宫算法用来寻找两个待连线网端点的一条路径。
  - 把平面表示成一个网格图，布线区域表示成由实心和空心顶点构成的顶点图，实心点表示布线障碍，空心点表示可布区域。
  - 一个端点称为源，另一个称为目标。
  - 迷宫算法的目标就是在源点和目标点间寻找一条空心顶点表示的连通路径。
  - 广度优先搜索策略。

Lee算法

① 数据准备：

建立二维数组G(I,J)，对应芯片上的矩形网格。定义两个波前(WaveFront)：一个为当前波前WF_1，另一个为上一个波前WF_0。初始时，WF_0和WF_1都为空集，数组G(I,J)中未占用的元素值为0。

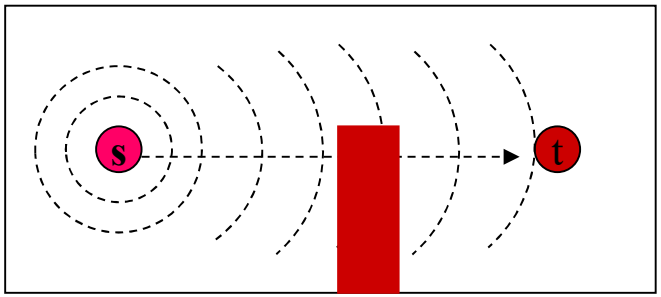
② 扩展过程：

选取源点，将其放入WF_0，在数组G(I,J)相应位置标志始点和目标点。WF_0非空时，顺次将其中元素取出，向邻接网格进行扩展。扩展后的网格值为WF_0元素网格值加1。记录该格值为V，当其邻格不是障碍或原格值大于V时，将V填入邻格，并将它送入WF_1。当WF_0中元素处理完毕后，将WF_1送入WF_0并将WF_1清零。

扩展时，若其中的一个邻格为目标点，扩展过程结束。若尚未达到目标点而WF_1为空，则说明该线网布线失败。

③ 回找过程：

可能存在多条连接A和B的路径。

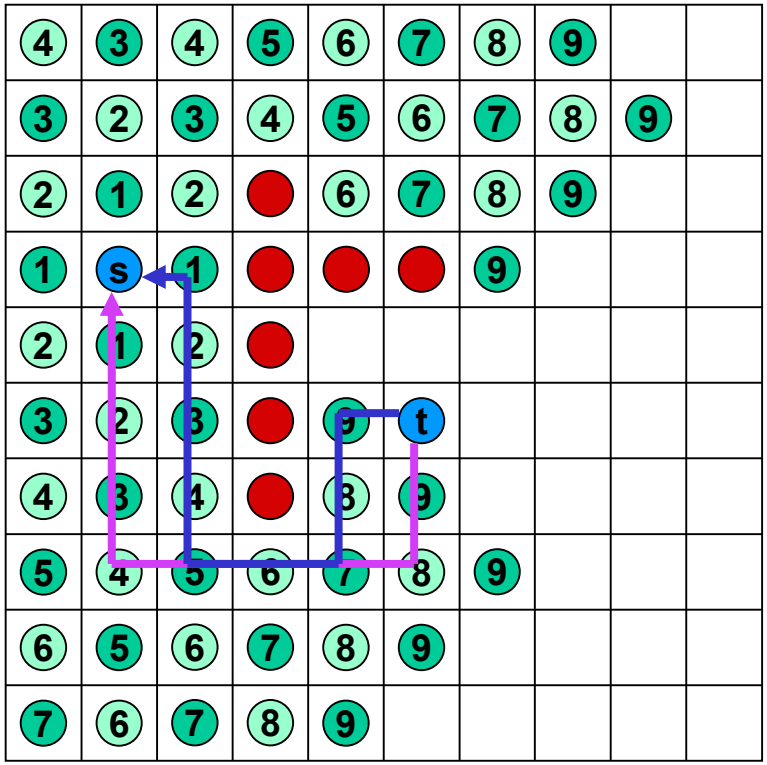


迷宫算法示例

① 数据准备：

② 扩展过程：

③ 回找过程：



- **Lee算法优点：**

  如果两点间存在路径，那么它一定能找到它。同时算法能保证找到最短的路径。Lee算法的另一个优点是有很好的适应性。

- **Lee算法缺点：**

  需要较大的存储空间和计算时间。

- Lee算法的时间和空间复杂度是O(h×w)，其中的h，w分别为网格高和宽方向的网格数。

普通最短路径问题

- *Dijkstra*算法
  - 给定带有非负边权的图(V, E)，寻找给定起始顶点s到终点t的最短路径
- *A**算法

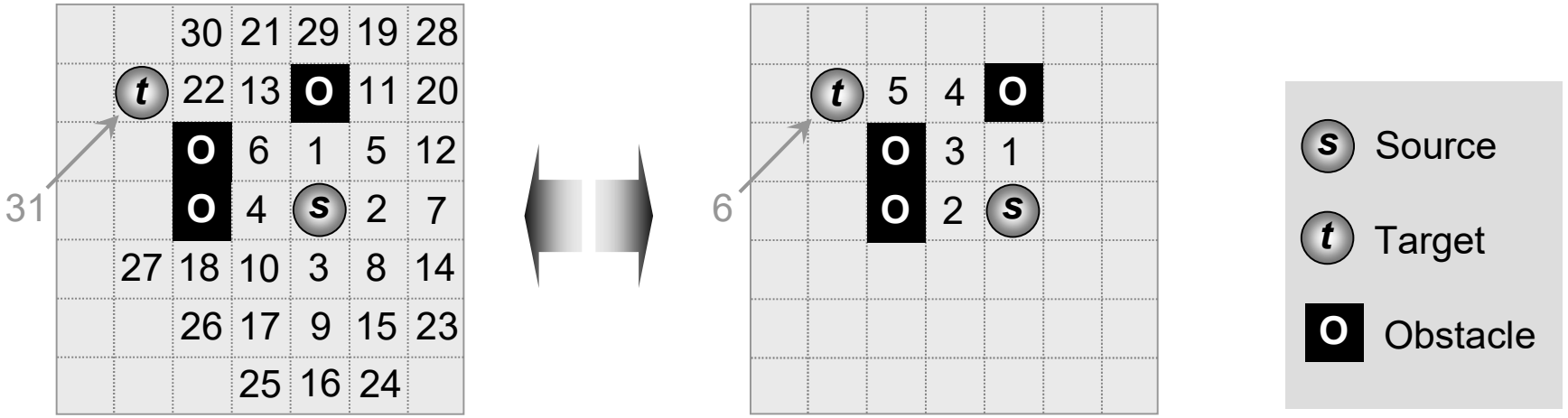
Finding Shortest Paths with Dijkstra's Algorithm

Find a shortest path between two specific nodes in the routing graph

- **Input**
  - graph $G(V, E)$ with non-negative *edge weights* W,
  - *source* (starting) node s, and
  - *target* (ending) node t
- **Maintains three groups of nodes**
  - **Group 1 – contains the nodes that have not yet been visited**
  - **Group 2 – contains the nodes that have been visited but for which the shortest-path cost from the starting node has not yet been found**
  - **Group 3 – contains the nodes that have been visited and for which the shortest path cost from the starting node has been found**
- **Once t is in Group 3, the algorithm finds the shortest path by backtracing**

Finding Shortest Paths with A* Search

- A* search operates similarly to Dijkstra's algorithm, but extends the cost function to **include an estimated distance from the current node to the target**
- **Expands only the most promising nodes**; its best-first search strategy eliminates a large portion of the solution space

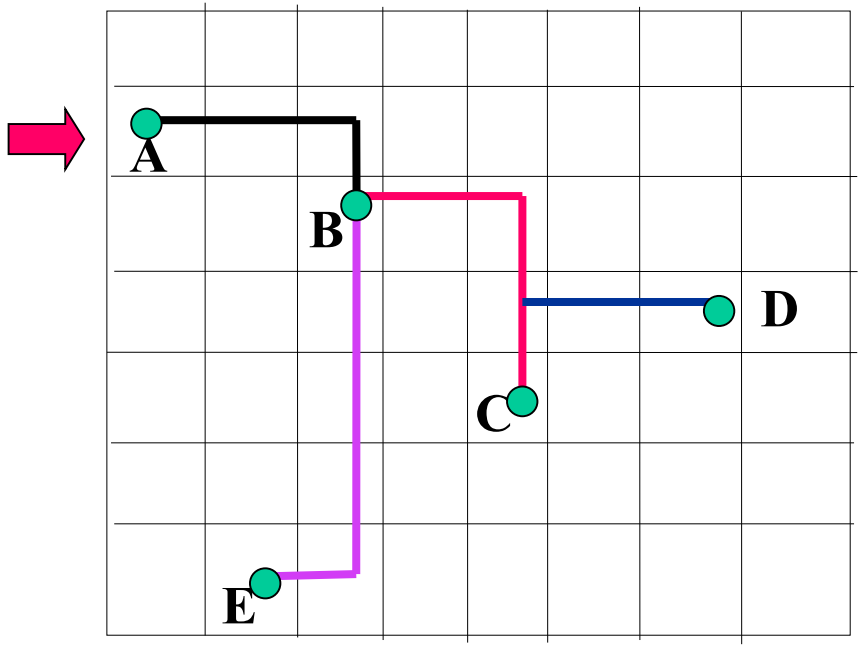


Dijkstra's algorithm
(exploring 31 nodes)

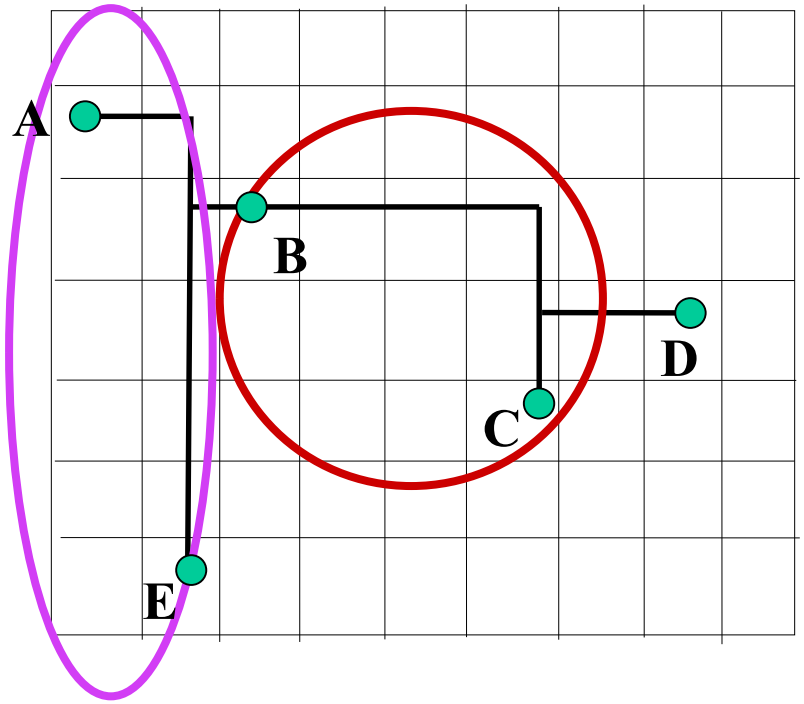
A* search
(exploring 6 nodes)

多端点线网布线

- 采用迷宫法，从一个顶点开始扩展，然后从建立的连线对外扩展，求解多端线网的Steiner树问题，但不能保证得到最优解。



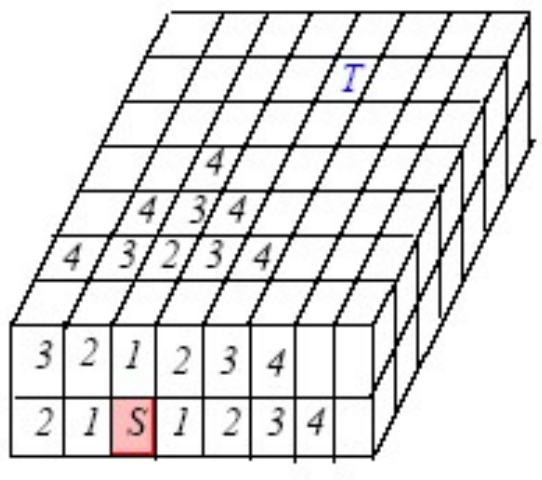
- 也可将它们首先分成两个子树，分别对它们用迷宫法布线，最后完成两棵子树的连接从而完成布线。



多层布线

- 迷宫算法可用于多层的布线。在多层布线时，布线网格为三维网格，仍采用波传播的方式向前扩展，每个网格为一个小立方体，填数在该布线区域的立方体上进行。

- 在多层迷宫算法中，如希望某一层走水平线，而另一层走垂直线，可以对网格填数进行加权处理。

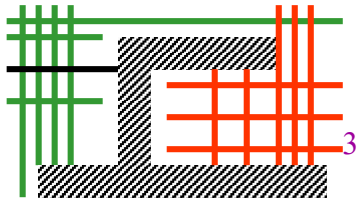


2. 线探索法(Line_Probe Algorithm)

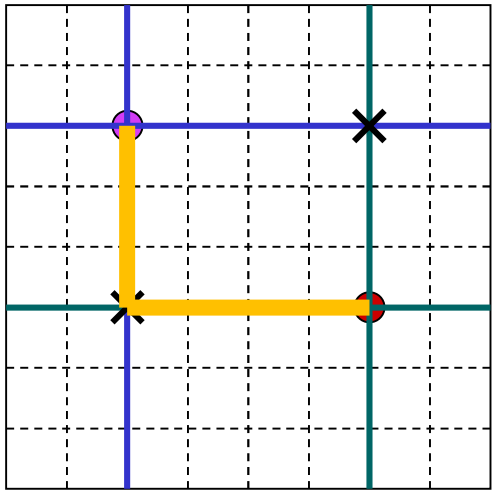
Mikami K. and K. Tabuchi, A Computer Program for optimal Routing of Printed Circuit Connectors, Proceedings of IFIP, H47: 1475-1478,1968

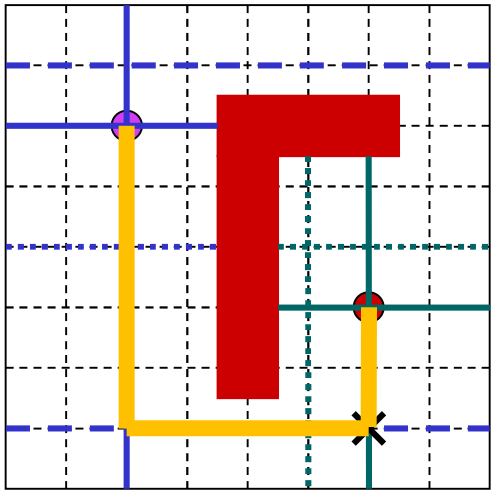
Hightower D. W., A Solution to Line-Routing Problem on the Continuous Plane, Proceedings of 6th Design Automation Workshop, pages 1-24, 1969

- Mikami 和Tebuchi在1968年，Hightower在1969年分别提出了线探索的思想和算法。他们算法的基本思想：
  - 深度优先策略
  - 用线探索的方法改变网格扩展
  - 减少存储空间和计算时间，计算时间与空间复杂度为O(L)，L为算法所产生的线数。



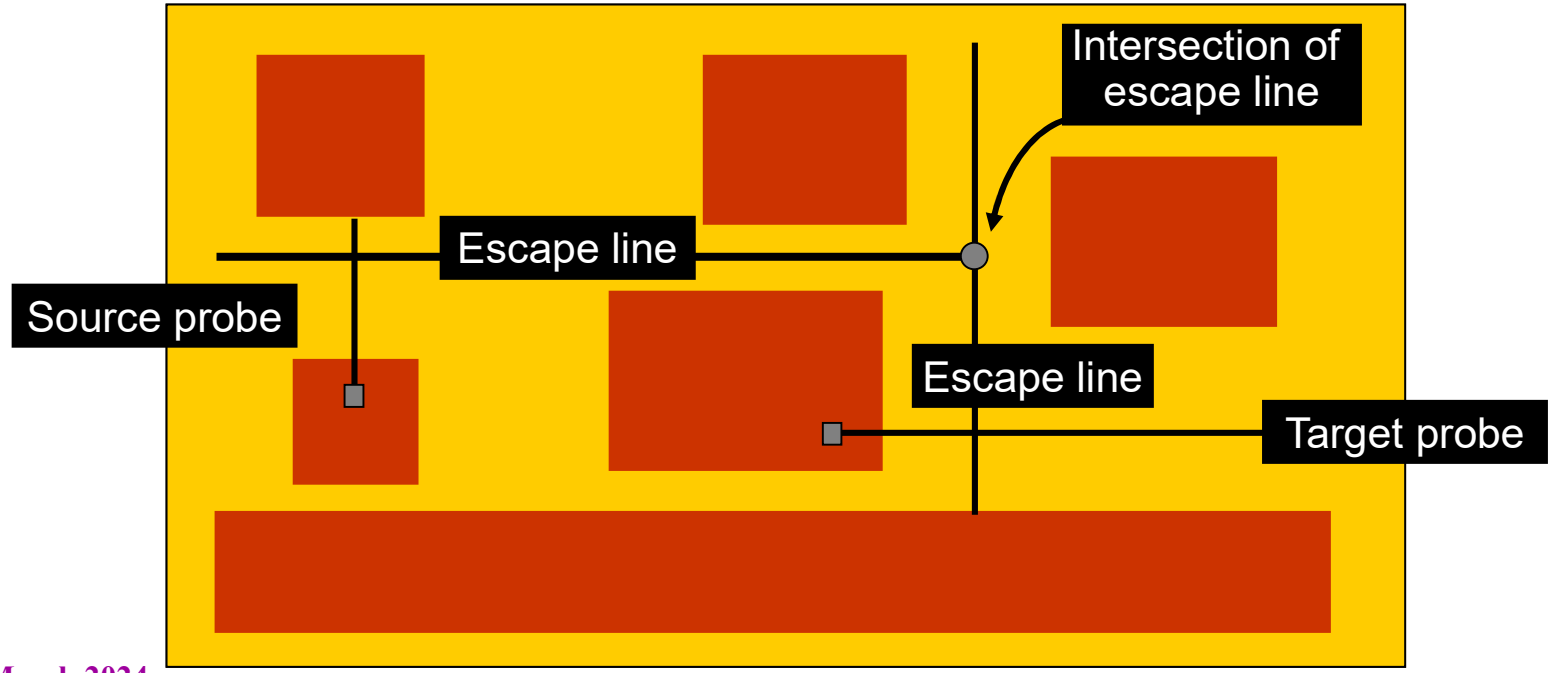
- 线探索法思想：
  - 从源点及目标点分别产生垂直放射直线；
  - 在没有障碍的情况下，如果这些直线相交，则找到两点之间的连接路径；
  - 当存在障碍物时，两边分别不断地产生放射直线，如果有直线相交则存在连接通路。
  - 后续放射线的产生方法不同，形成不同的算法。





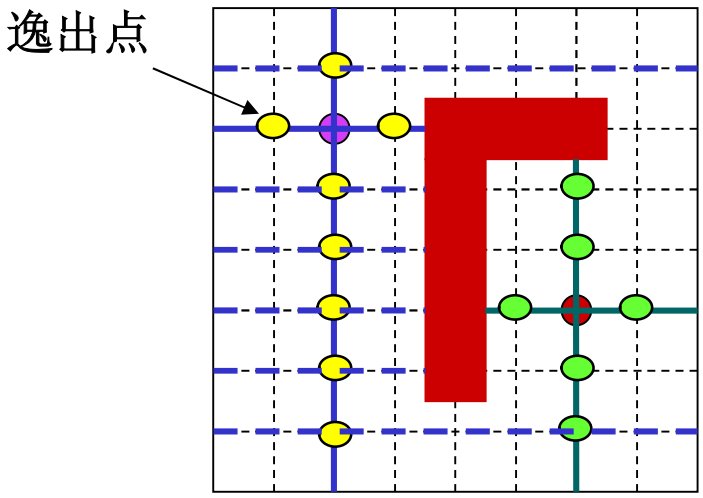
Line-Probe Summary

- **Fast, handles large nets / distances / designs**
- **Routing may be incomplete**

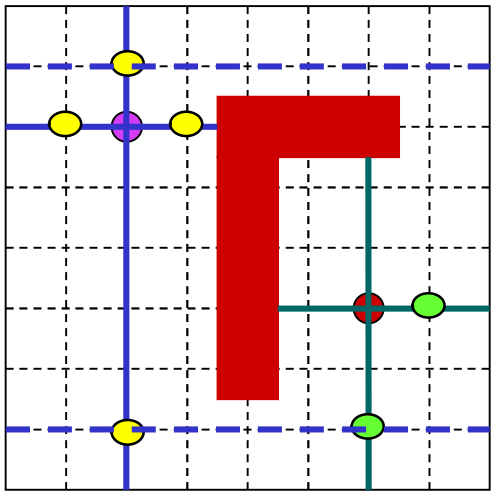


- Mikami和Hightower两个算法的主要区别在于逸出点的选择。
  - Mikami算法中，线上的每一个点都可为逸出点，相当于广度优先搜索，它保证能找到一条存在的通路，但不一定是最短的。
  - Hightower算法中，每条线只有一个逸出点。Hightower列举了三种在不同障碍面前寻找逸出点的方法。由于逸出点较少，该算法不一定能找到两点建立的一条通路。

Mikami算法



Hightower算法



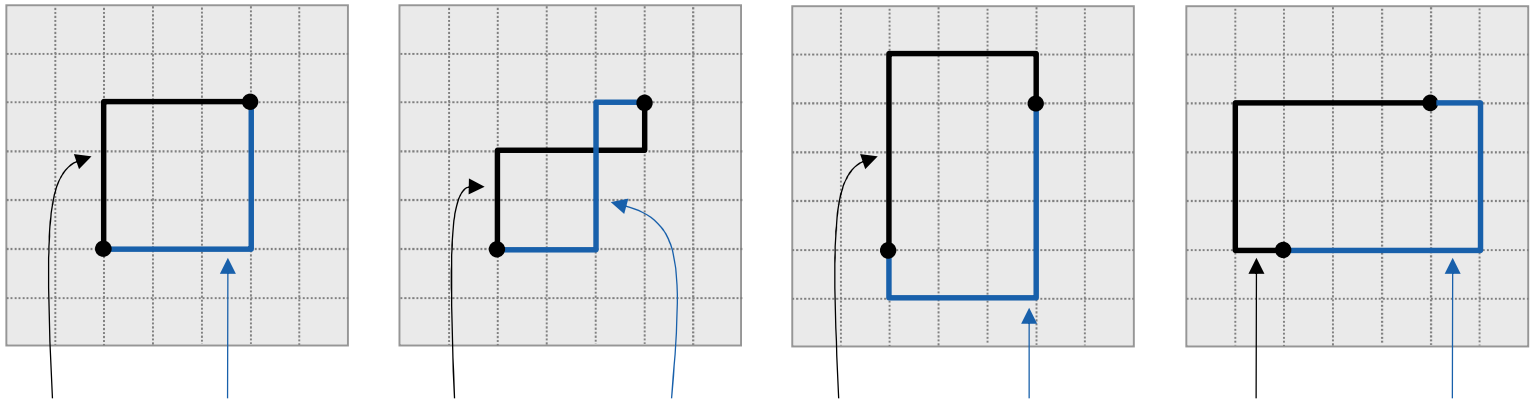
线探索法的描述：

```
Algorithm Line-Probe_Router(input:s, t; output:P)
    input: s, t;        output: P
  begin
    new_slist = line segments generated from s;
    new_tlist = line segments generated from t;
    while new_slist ≠ 0 and tlist ≠ 0 do
      slist = new_slist;     tlist = new_tlist;
      for each line segment l_i in slist do
        for each line segment l_j in tlist do
          if INTERSECT(l_I , l_j) = TRUE then
            path_exists = TRUE;    exit while;
        new_slist = 0;
        for each line segment l_i in slist do
          for each escape point e on l_i do
            GENERATE(l_k, e);    INSERT(l_k,  new_tlist);
      if path_exist = TURE then RETRACE;
      else a path can not be found;
  end
```

- GENERATE(l,e)：从逸出点e产生一条直线l；
- INSERT(l, list)：在list中加入一搜索直线l；
- INTERSECT(l_i, l_j)：当直线l_i同l_j相交时，返回TURE；否则，返回FALSE。

3. Pattern-Based Routing

- Searches through a small number of route patterns to improve runtime
- Topologies commonly used in pattern routing: *L*-shapes, *Z*-shapes, *U*-shapes



三、多条线网布线算法

1、布线顺序及处理方法

2、线性规划技术和拆线重布技术

3、并行布线

4、 Negotiation-Based Routing

1. 布线顺序及处理方法

- 短线序：定义一种线网长度S_i，按S_i的顺序，S_i小的先布。
- 长线序：S_i长的线先布(S_i大的难度大，不易布)。
- 点干扰度：线网的点干扰度定义为覆盖线网的最小矩形内其它线网的顶点数目。干扰度小的线网优先布线(即不影响其它布线)。
- 线网最小覆盖矩形大小
- 线网宽长比（net's aspect ratio）

依然是一个未能解决的问题！

2. 线性规划技术和拆线重布技术

■ 线性规划技术

- 对线网构造多种布线拓扑方案
- 在布线资源约束下，采用规划的方法，从线网的布线拓扑方案中选取合适的组合
- 实现完成布线条数最多、线长最短等布线目标

■ 拆线重布策略

各条线网独自完成布线，在有资源冲突的情况下，拆除一些线网，重新进行布线

- 整条线网和按线段；
- 冲突区域如何选择被拆线网或线段
- 重布顺序；
- 重布算法

3. 并行布线

每条连线不是一次，而是多次(如五次)完成布线，每次只完成线网的一部分布线，每条线网的后续部分的布线建立在前一轮布线的结果之上，从轮次上形成线网并行生长。

算法思想为：

① 用李氏算法以某种线序，逐条对每条线网进行布线

- 每条线网布线完成后仅承认其由接点引出的1/10总长的部分；
- 删去该线网的其它部分，进行下一条线网的布线。

② 仍用李氏算法按一定顺序对每条连线进行布线

- 在保留其它线网已完成的布线部分基础上，先将该线网原布线结果全部删去，然后重布。
- 布完后，承认其接点引出的2/10总长的部分，其余部分删去。

重复上述过程共五次，就完成了整个布线过程。

游戏：

有n个人，m件物品。各个人对每件物品的喜爱程度不同，还有可能不喜欢。每件物品有单独的价格，每人都是以最小付出贪婪地购买一件最喜爱的物品。给出一种分配方法，每人获得一件物品，总的满意程度最高（价格除以喜爱程度最小化）。

讨论：

- 如何形式化描述问题？
- 如何通过协商解决人获取的物品和顺序？
- 给出相应的程序流程？

实例（1）

问题：三个人(p_1, p_2, p_3)，三个苹果(a_1, a_2, a_3)，每个苹果的价格为1。每人购买一个苹果，三个人对三个苹果有不同的喜爱：

- p_1: （10, 3, 7）
- p_2: （5, 10, 1）
- p_3: （10, 4, 0）

在人对苹果最小付出下贪婪占有的情况下：

1. 选择顺序（P_1，P_2，P_3）的结果？
2. 最佳解的选择顺序？
3. 在给定选择顺序时，如何获得最佳分配方案？

协商——价格随需求翻倍

p_1: （10, 3, 7）
p_2: （ 5, 10, 1）
p_3: （ 10, 4, 0）

- 选择顺序（P_1，P_2，P_3）下的协商情况:
  - 第一轮:

    p_1: a_1 贪婪占有
    p_2: a_2 贪婪占有
    p_3: a_1 贪婪占有

    满意度:

a_1	a_2	a_3
1/10	1/3	1/7
1/5	1/10	1/1
1/10	1/4	1/0

  - 第二轮: a_1价格加倍

    p_1: a_3 成本增加改选
    p_2: a_2 贪婪占有
    p_3: a_1 贪婪占有

    满意度:

a_1	a_2	a_3
2/10	1/3	1/7
2/5	1/10	1/1
2/10	1/4	1/0

总结: 在第二轮的时候，p_1看到苹果a_1有两个人申请，成本增加，他采取避免冲突的策略，选择次优。

实例（2）

问题：三个人(p_1, p_2, p_3)，三个苹果(a_1, a_2, a_3)，每人分摊一个苹果。三个人对三个苹果有不同的喜爱：

- p_1：（6，10，0）
- p_2：（0，3，10）
- p_3：（0，0，10）

在人对苹果贪婪占有的情况下：

协商——价格随需求翻倍

p1：（6, 10, 0）
p2：（0, 3, 10）
p3：（0, 0, 10）

- 选择顺序（P_1，P_2，P_3）下的协商情况：
  - 第一轮：

    p_1：a_2 贪婪占有
    p_2：a_3 贪婪占有
    p_3：a_3 贪婪占有

    满意度：

a_1	a_2	a_3
1/6	1/10	1/0
1/0	1/3	1/10
1/0	1/0	1/10

  - 第二轮：a_3价格加倍

    p_1：a_2 贪婪占有
    p_2：a_3 贪婪占有
    p_3：a_3 贪婪占有

    满意度：

a_1	a_2	a_3
1/6	1/10	2/0
1/0	1/3	2/10
1/0	1/0	2/10

总结： 在第二轮的时候，虽然苹果a_3有两个人申请价格增加，但是由于p_2对苹果a_2的偏爱差，价格随需求加倍仍无法回避冲突。

协商——价格再随历史翻倍

p1：（6, 10, 0）
p2：（0, 3, 10）
p3：（0, 0, 10）

- 选择顺序（P_1，P_2，P_3）下的协商情况：
  - 第三轮：a_3价格再加倍　　满意度：

	a_1	a_2	a_3
p_1：a_2 贪婪占有	1/6	1/10	4/0
p_2：a_2 贪婪占有	1/0	1/3	4/10
p_3：a_3 贪婪占有	1/0	1/0	4/10

  - 第四轮：a_2价格加倍　　满意度：

	a_1	a_2	a_3
p_1：a_1 贪婪占有	1/6	2/10	2/0
p_2：a_2 贪婪占有	1/0	1/3	4/10
p_3：a_3 贪婪占有	1/0	1/0	2/10

总结：在第三轮的时候，由于苹果a_3有历史上曾有两次两个人申请，价格将再提高，由于价格的提高，p_2将转向苹果a_2，回避了冲突。

4. Negotiation-Based Routing

R. Nair. A simple yet effective technique for global wiring. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(6):165–172, March 1987.

L. Mcmurchie and C. Ebeling. PathFinder: A negotiation-based performance driven router for FPGAs. In *Proceedings of International Symposium on Field-Programmable Gate Arrays*, February 1995.

- **Global router: to route all nets in passes by rip-up and retry method**
  - **on a pass, nets are ripped up and rerouted one by one in same order**
  - **Dynamically adjusts the congestion penalty of each routing resource based on the demands nets place on that resource.**
  - **cost function gradually increases the penalty for sharing**

$$c_n = (b_n + h_n) * p_n + bend$$

  - b_n **is the base cost of using** n
  - h_n **is related to the history of congestion on** n **during previous passes**
  - p_n **is related to the number of other signals presently using** n
  - ***bend*** **is a penalty on bend number**

- **First pass: routing all nets in a order by no cost for sharing routing resource**
  - **individual routing resources may be used by more than one net**

- **Subsequent passes: the penalty is gradually increased so that nets in effect negotiate for resources.**
  - **Nets may use shared resources that are in high demand if all alternative routes utilize resources in even higher demand; other nets will tend to spread out and use resources in lower demand.**

Congestion-based routing algorithm

Negotiation Loop

Net Routing

For sinks

While shared resource exist **(Global Router)**
  Loop over all signals *i* **(Net Router)**
    Rip up routing tree RT_i
    $RT_i = s_i$
    Loop until all sinks t_{ij} have been found
      Initialize priority queue *PQ* to RT_i at cost 0
      Loop until new t_{ij} is found
        Delete lowest cost node *m* from *PQ*
        Loop over fanouts *n* of node *m*
          Add *n* to *PQ* at cost $c_n + p_{im}$
      End
    End
    Loop over nodes *n* in path t_{ij} to s_i (Backtrace)
      Update c_n
      Add *n* to RT_i
    End
  End
End

Cost函数分析 —— p_n的作用示例

Nets:

net_1:(S_1, D_1)
net_2:(S_2, D_2)
net_3:(S_3, D_3)

Routing order: (3,2,1)

Assuming: h_n=0, *bend*=0

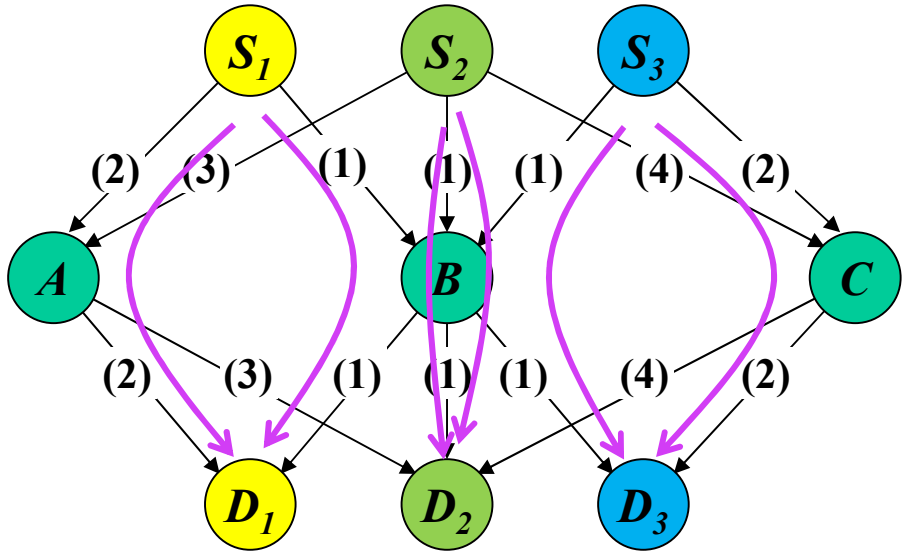
First iteration: p_n=1

net_1:(A-2, B-1)
net_2:(A-3, B-1, C-4)
net_3:(B-1, C-2)

Second iteration: p_B=3

net_1:(A-2, B-3)
net_2:(A-3, B-3, C-4)
net_3:(B-3, C-2)

$$c_n = (b_n \qquad)^* p_n + bend$$



Cost函数分析 —— h_n的作用示例

Nets:

net_1:(S_1, D_1)
net_2:(S_2, D_2)
net_3:(S_3, D_3)

Routing order: (1,2,3)

Assuming: h_n=0, *bend*=0

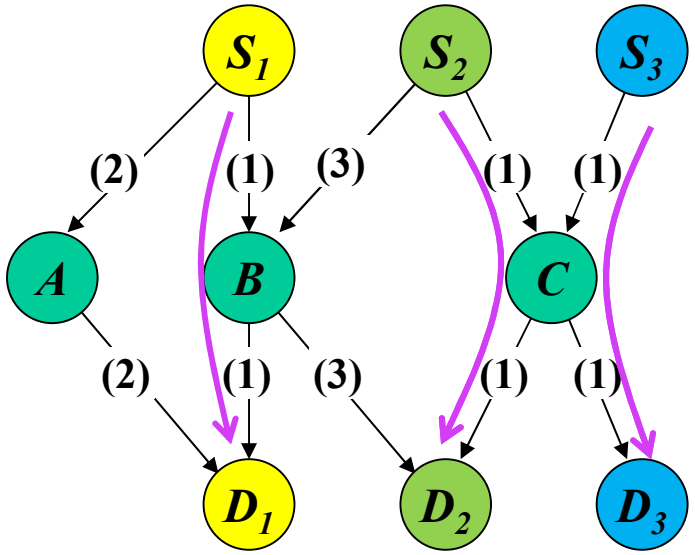
First iteration: p_n=1

net_1:(A-2, B-1)
net_2:(B-3, C-1)
net_3:(C-1)

Second iteration: p_C=2

net_1:(A-2, B-1)
net_2:(B-3, C-2)
net_3:(C-2)

$$c_n = (b_n \qquad) * p_n + bend$$



Third iteration: p_C=2

net_1:(A-2, B-1)
net_2:(B-3, C-2)
net_3:(C-2)

单靠p_n因素，
无法使net_2走B！

Cost函数分析 —— h_n的作用示例　　增加历史因素

Routing order: (1,2,3)

Assuming: *bend*=0

$$c_n = (b_n + h_n) * p_n + bend$$

First iteration: p_n=1, h_n=0

net_1:(A-2, B-1)

net_2:(B-3, C-1)

net_3:(C-1)

Second iteration: p_C=2, h_C=1

net_1:(A-2, B-1)

net_2:(B-3, C-4)

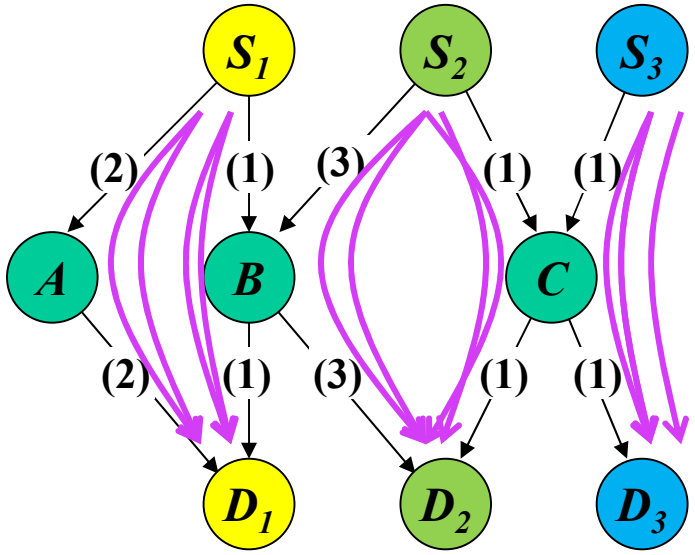
net_3:(C-4)

Third iteration: p_B=2, h_B=1, h_C=1

net_1:(A-2, B-4)

net_2:(B-8, C-2)

net_3:(C-2)



Fourth iteration: p_C=2, h_B=1, h_C=2

net_1:(A-2, B-2)

net_2:(B-4, C-6)

net_3:(C-6)

布线问题及算法（总结）

- 布线问题
- 迷宫算法(Maze Routing Algorithms)
- 线探索法(Line Probe Algorithms)
- Pattern-Based Routing
- 布线顺序的影响及其处理
- 线性规划/拆线重布策略
- 并行布线
- Negotiation-Based Routing