第8章 详细布线 (Detailed Routing)

8.5.0 详细布线问题

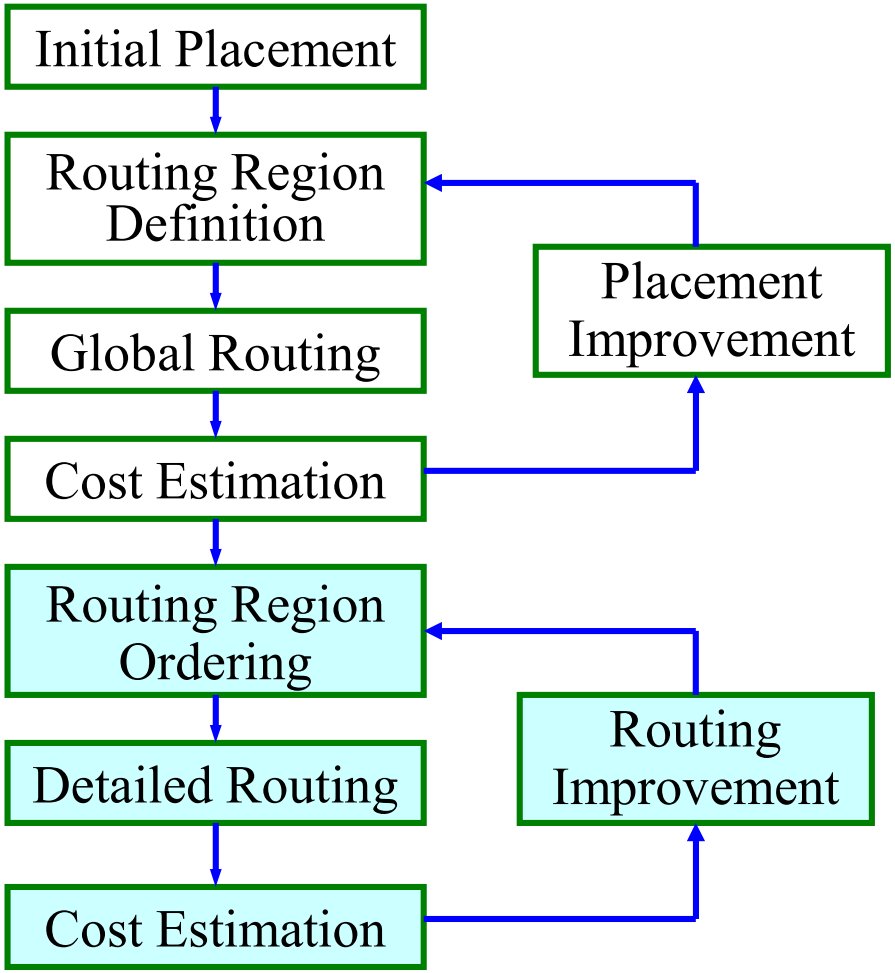
8.5.1 通道布线问题

8.5.2 通道布线的定义和约束方法

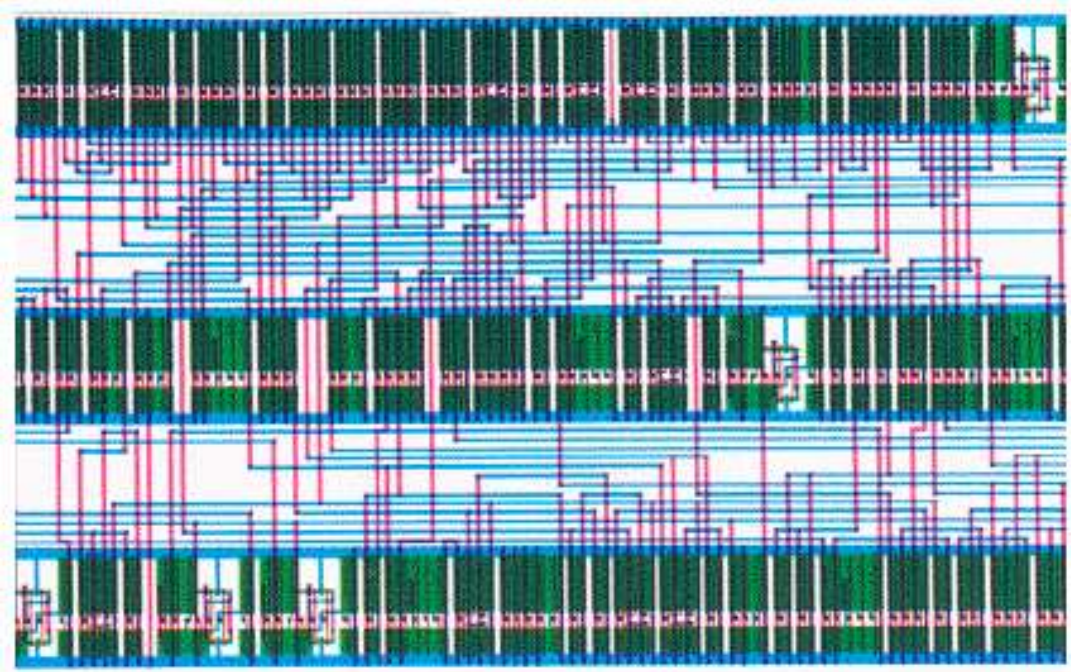
8.5.3 常用的几种通道布线算法

8.5.4 其他布线问题

Placement and Routing Flow Review



标准单元布图示例



规模挑战:

- Chip: 360000×360000 tracks
- Global Cell: 600×600 regions
- Routing Region: 600×600 tracks

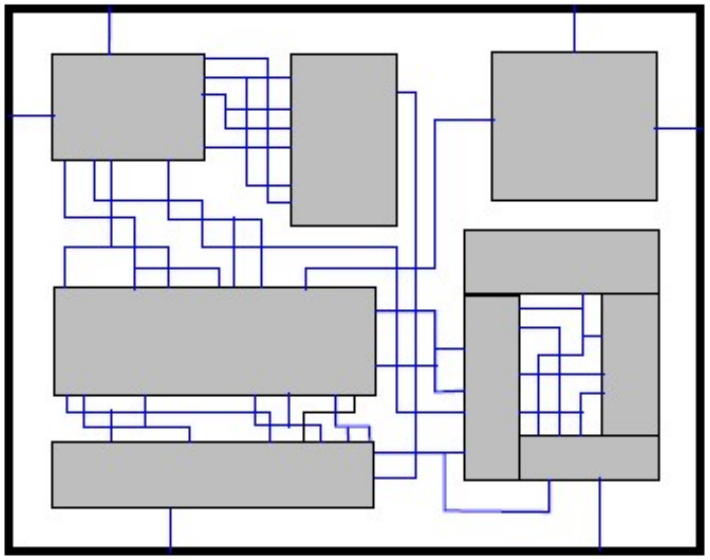
8.0 详细布线问题

- 布图问题的最后步骤;
- 主要任务是确定线网各个线段在布线区域中的具体位置，从而完成线网在布线区域的最后定位。

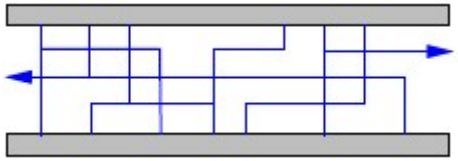
详细布线问题分类:

- **单层布线问题:**
  - River Routing
- **双层布线问题:**
  - 通道布线问题（Channel Routing）;
  - 开关盒布线问题（Switchbox Routing）。
- **多层布线问题**
  - 区域布线问题：管脚可能发生在布线区的任何地方。

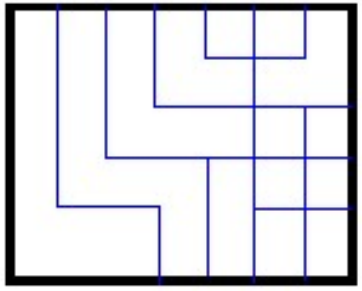
详细布线示例



Detialed Routing



Channel Routing

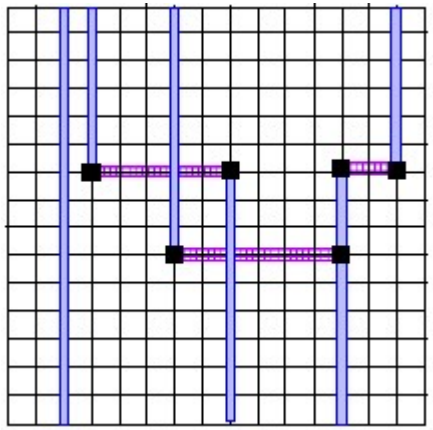


Switch Box Routing

有网格布线问题和无网格布线问题

有网格布线问题

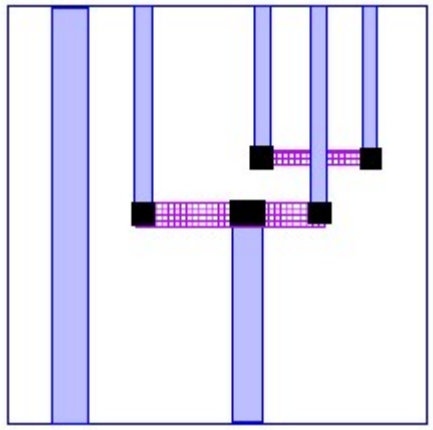
- 按照工艺要求，用统一的规则，将整个布图区域的各个布线层划分成供走线使用的轨道，线网在走线轨道上建立电等价连接;
- 有利于设计自动化;
- 降低了性能设计的灵活性。



Grid-based Routing

无网格布线问题

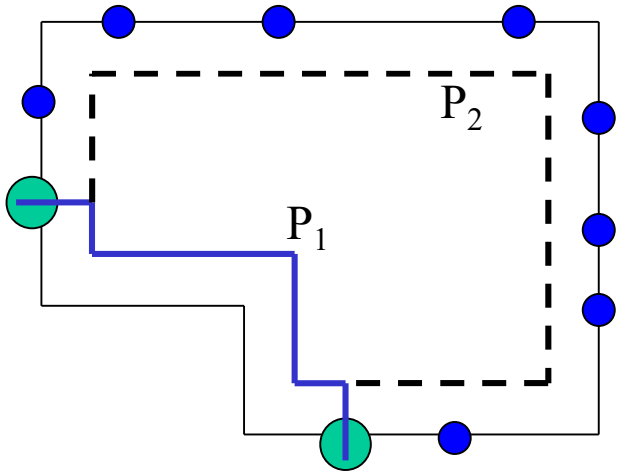
- 在不违反设计规则的情况下，连线的宽度可以任意变化;
- 设计自动化复杂度提高;
- 增加了性能设计的灵活性。



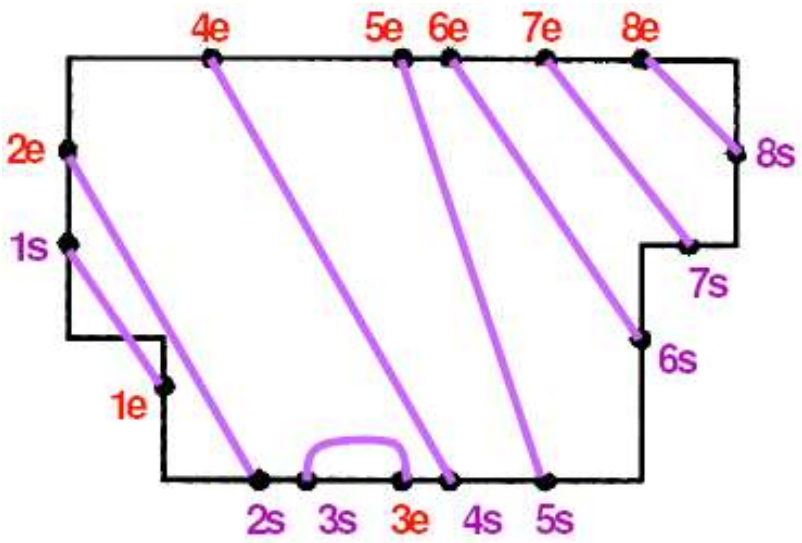
Gridless Routing

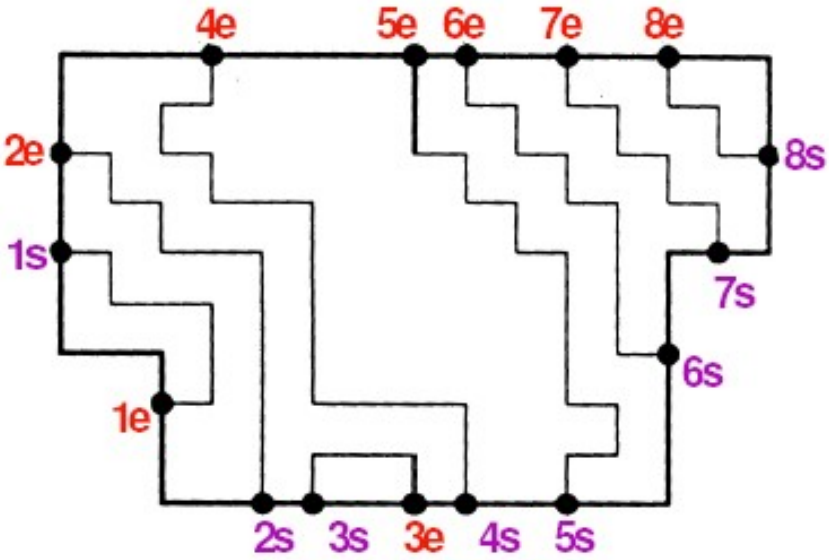
单层布线问题:

- For clock, power, ground routing still may need to solve single-layer routing;
- Two possible paths per net along boundary;
- Path is alternating sequence of horizontal and vertical segment connecting two terminals of a net.



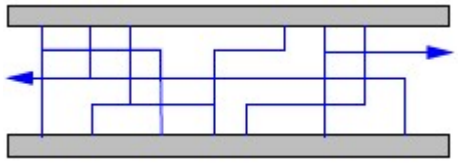
- 用圆弧连接布图边界上线网的两个端点;
- 用紧贴布图区域边界的横竖相间的线段实现圆弧的连接;
- 必要时进行调整改善线网走线形状。





- **Single-layer routability problem is NP-Complete!**

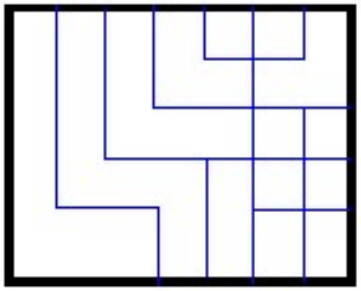
布线区域划分



双层布线工艺下，布线区域分成两种:

- **两边通道(Channel Routing Region)**
  - 两边通道布线问题中，线网引脚分布在通道的上下两边界;
  - 线网可以从通道的左右两侧进出，但位置不固定（可以将两侧进出位置映射到上下两边界）;
  - 布线问题称为两边通道布线(Channel Routing)。

- **开关盒(Switchbox Routing Region)**

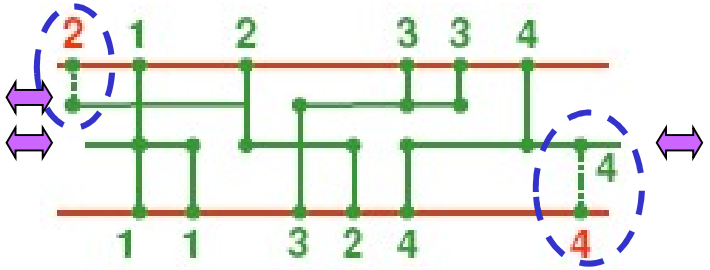


  - 开关盒也称为四边通道，在四边通道布线问题中，线网引脚分布在布线区域的四周;
  - 布线问题称为开关盒布线(Switchbox Routing)。

通道布线与开关盒布线示例

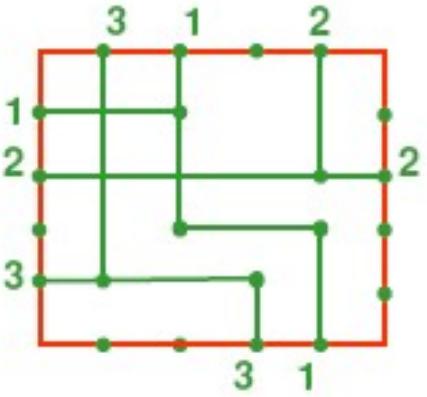
❖ 通道布线(Channel Routing)

- 线网1-4端点分布上下边界;
- 线网1-2有左连接、线网4有右连接;
- 线网2左连接映射到上边界;
- 线网4的右连接映射到下边界

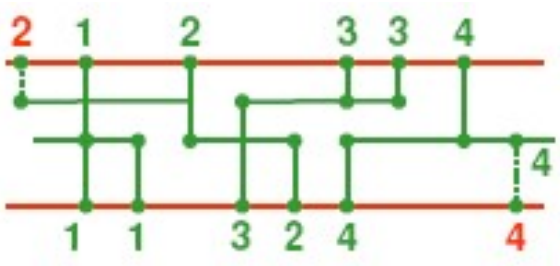


❖ 开关盒布线(Switchbox Routing)

- 线网1-3端点分布四周边界



8.1 通道布线问题



输 入:

- 布线通道上下两边的线网管脚位置。

输出:

- 布线通道中线网对布线轨道资源的占用。

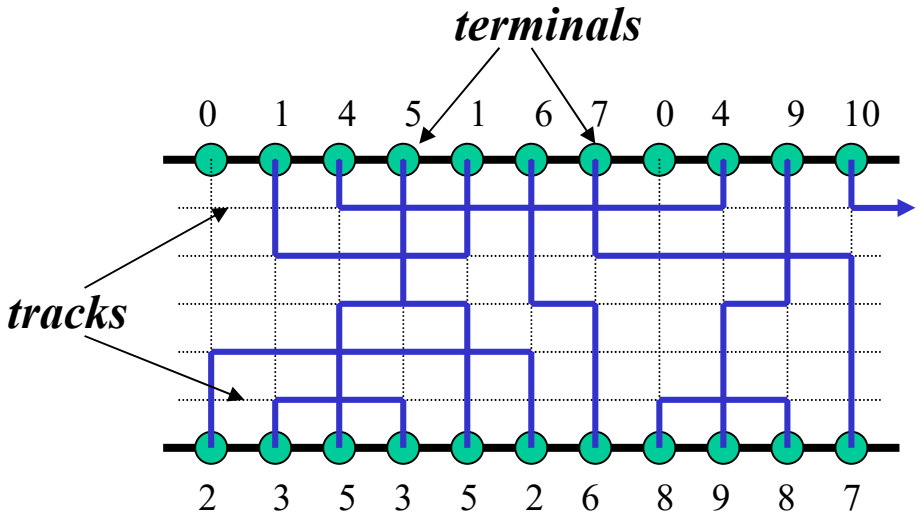
约束条件:

- 布线网格;
- 两层布线，严格分层。一层用于水平方向走线，一层用于垂直方向走线;
- 通过通孔连接水平走线层线段与垂直走线层线段。

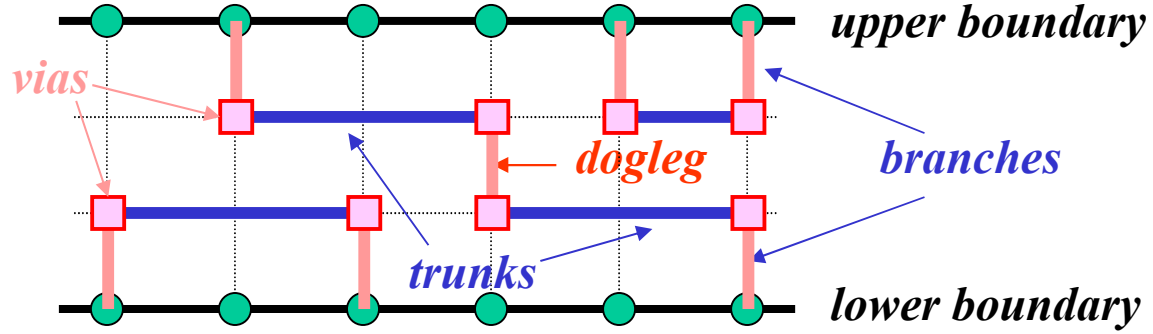
优化目标: 最小化

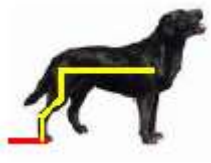
- 水平通道的高度，即需要的水平布线轨道数;
- 总的连线长度;
- 所需的通孔数。

♣ 通道布线中的一些概念



local density 1 3 5 5 4 3 3 3 4 3 2 *channel density: max local density*





通道布线问题说明

- **水平线段分配:**
  - 分配线网的水平走线段到某一走线道上。
- **垂直线段分配:**
  - 实现相同线网，在不同走线道上水平线段的连接;
  - 实现上下通道边界管脚与水平线段的连接。
- **水平和垂直约束:**
  - 同一轨道上，不同线网的水平走线段不能出现重叠;
  - 同一列上，不同线网的垂直走线段不能发生重叠。
- **优化目标:**
  - 通道的高度最小化，即面积最小化。

♣ **预留层模型(Reserved Layer Model):**

规定某些走线需求必须在某些指定的布线层上完成，例如:

➢ 两层布线:

- ✓ HV模型：严格规定第一层布水平走线，第二层布垂直走线;
- ✓ VH模型：严格规定第一层布垂直走线，第二层布水平走线。

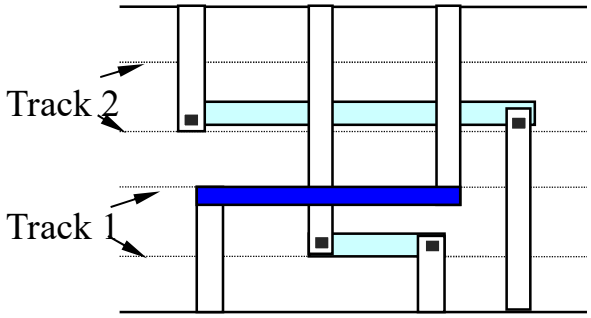
➢ 三层布线:

- ✓ VHV(Vertical - Horizontal - Vertical)模型;
- ✓ HVH(Horizontal - Vertical - Horizontal)模型;
- ✓ 通常采用HVH模型，通道需求比较少。

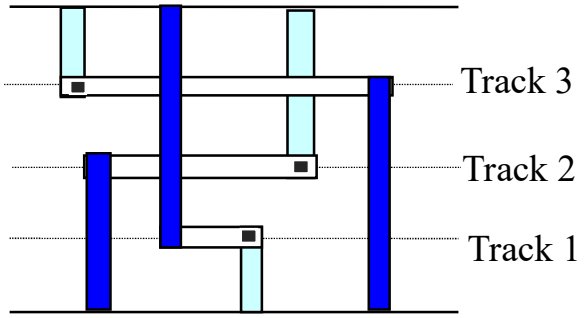
♣ **非预留层模型(Unreserved Layer Model):**

没有分层限制，也称非严格分层。

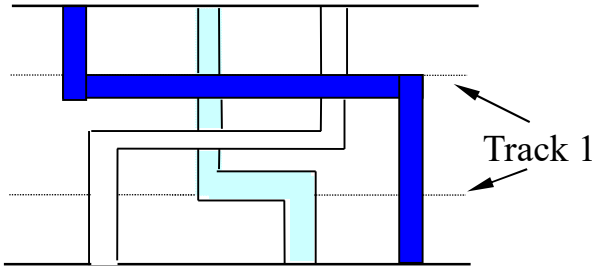
多层布线模型示例



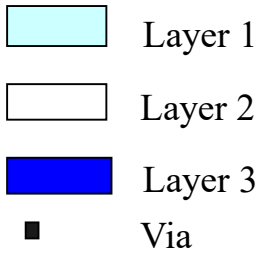
HVH模式



VHV模式



非预留层模式



Basic Left-edge Example

input: $U=\{I_1, I_2, \ldots\ldots, I_6\}$;

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**

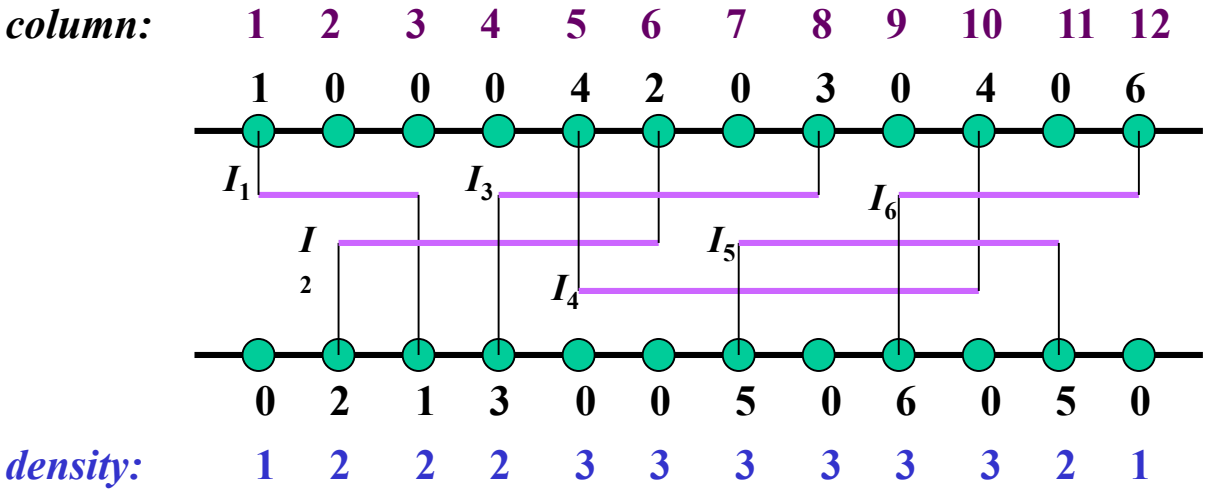
- Route I_1: watermark = 3;
- Route I_3: watermark = 8;
- Route I_6: watermark = 12;

***t*=2**

- Route I_2: watermark = 6;
- Route I_5: watermark = 11;

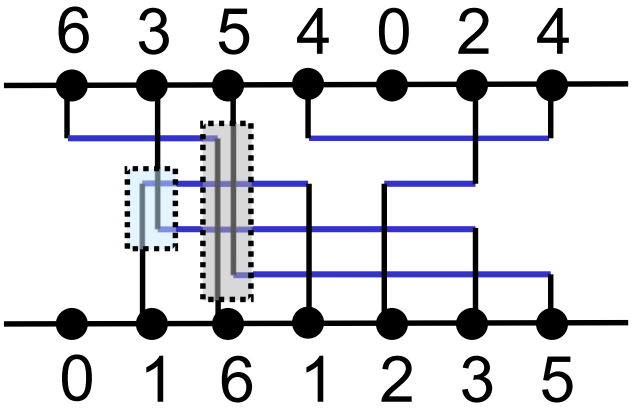
***t*=3**

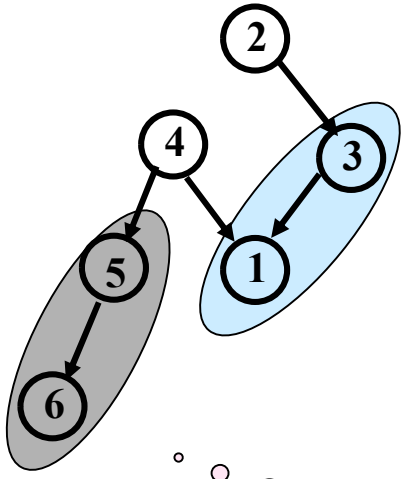
- Route I_4.



Vertical Constraint Consideration

- Left-edge algorithm ignores vertical constraints.
- Where there is only 1 vertical layer, Left-edge algorithm will produce overlapping of vertical wire segments.





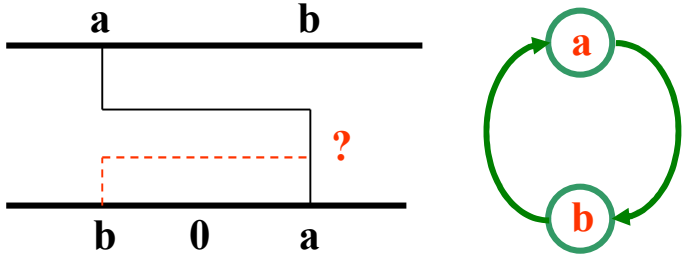
- **net_6 is constrained by net_5, can not route before/up net_5.**
- **net_1 is constrained by net_3, can not route before/up net_3.**

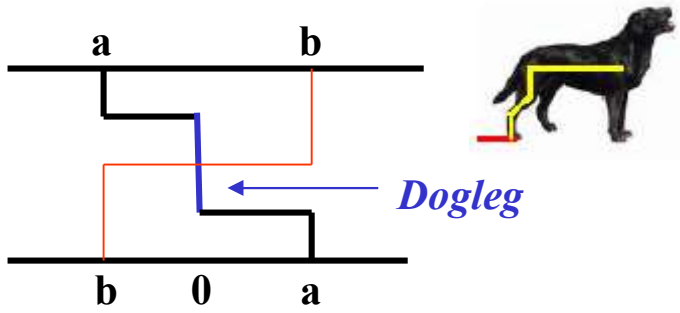
How to do?

❖ Drawback of Left-Edge (1):

– **Cannot handle the cases with vertical constraint cycles.**

➢ Doglegs are used to resolve constraint cycle.

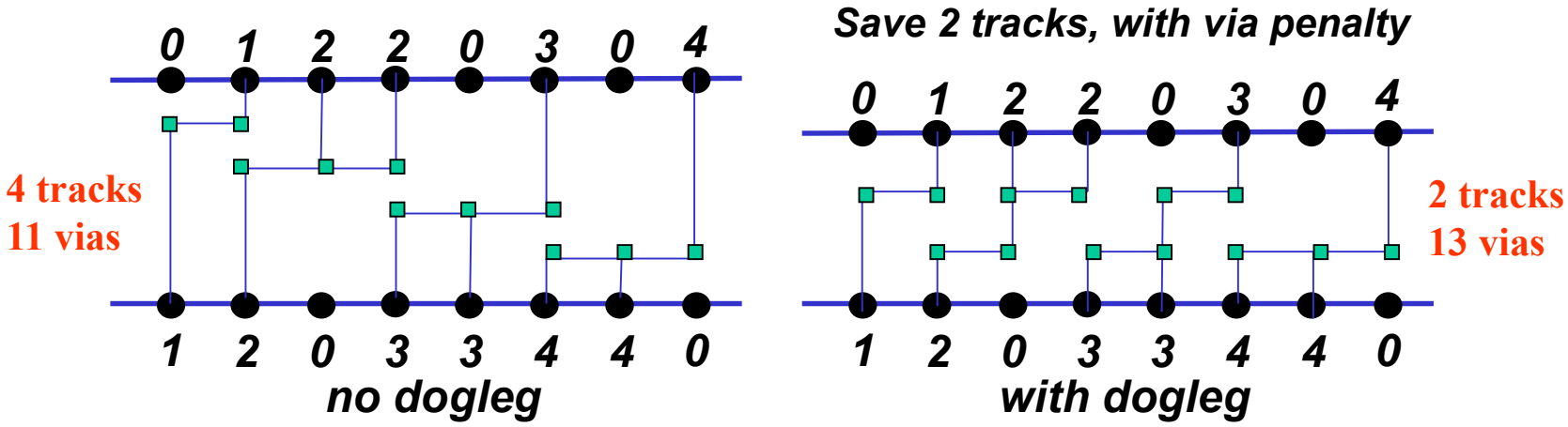




❖ Drawback of Left-Edge (2):

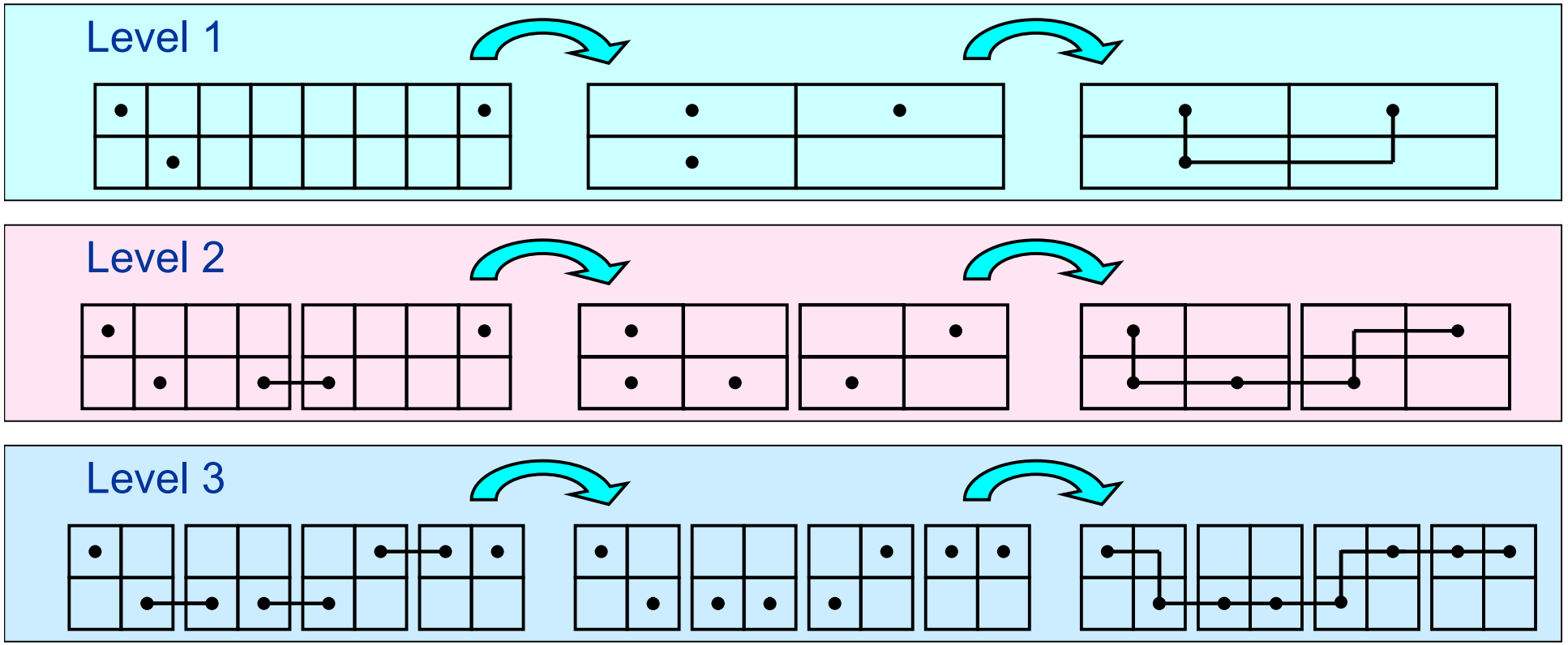
– **The entire net is on a single track.**

- Doglegs are used to place parts of a net on different tracks **to minimize channel height**.
- Might incur penalty for additional vias.

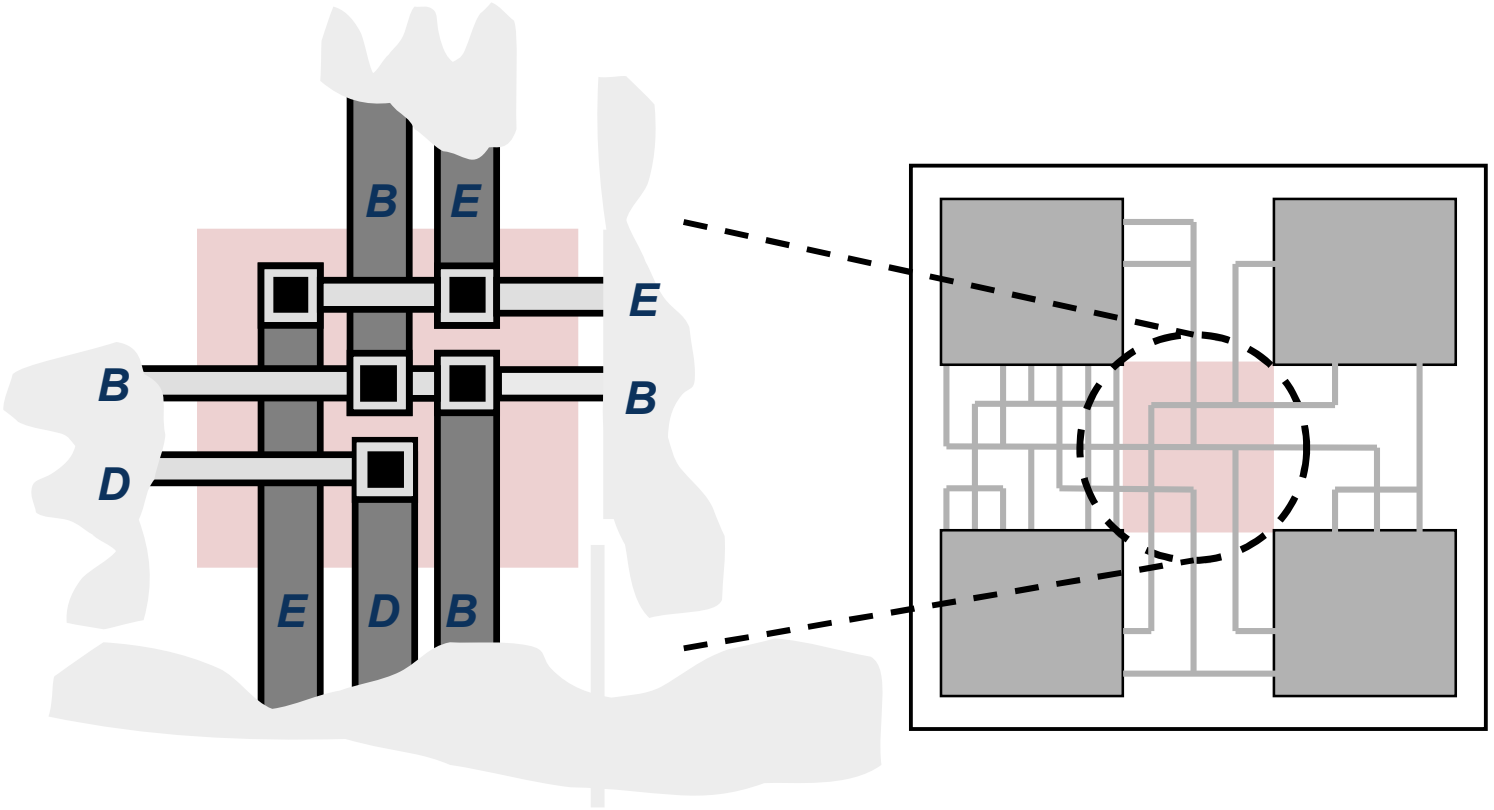


Hierachical Approach: Example

- Solving a 2xn routing problem hierachically.



Switchbox Routing

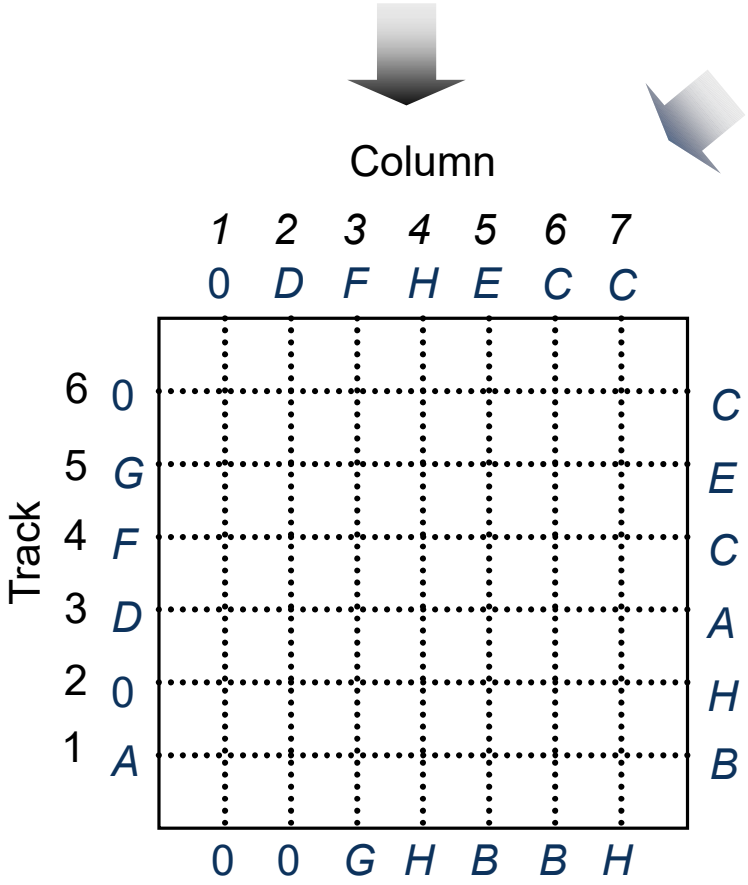


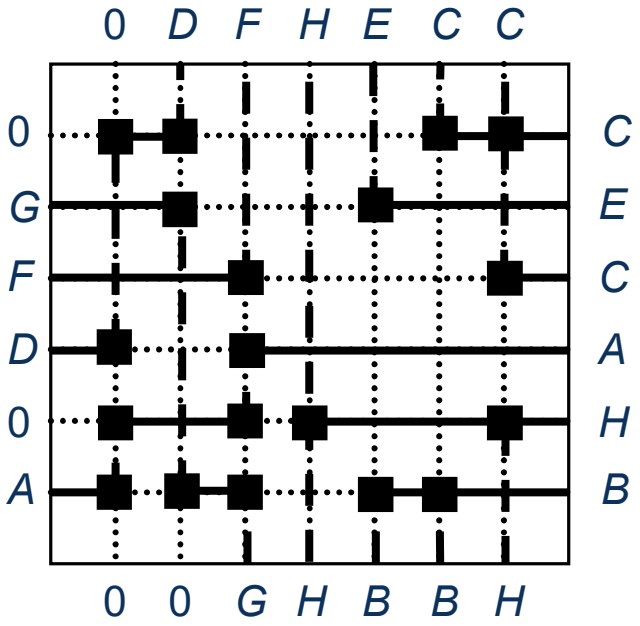
- Fixed dimensions and pin connections on all four sides
- Defined by four vectors *TOP*, *BOT*, *LEFT*, *RIGHT*
- Switchbox routing algorithms are usually derived from (greedy) channel routing algorithms

Switchbox Routing

$R = \{0, 1, 2, \ldots, 8\} \times \{0, 1, 2, \ldots, 7\}$

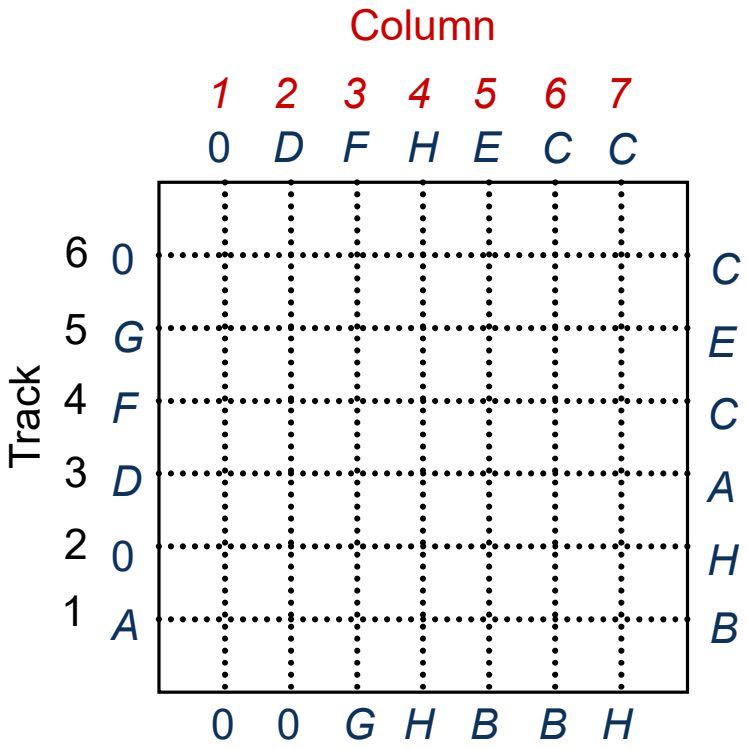
TOP = (1, 2, … , 7) = [0, *D*, *F*, *H*, *E*, *C*, *C*]
BOT = (1, 2, … , 7) = [0, 0, *G*, *H*, *B*, *B*, *H*]
LEFT = (1, 2, … , 6) = [*A*, 0, *D*, *F*, *G*, 0]
RIGHT = (1, 2, … , 6) = [*B*, *H*, *A*, *C*, *E*, *C*]





Switchbox Routing – Example

TOP = (1, 2, … , 7) = [0, *D*, *F*, *H*, *E*, *C*, *C*]
BOT = (1, 2, … , 7) = [0, 0, *G*, *H*, *B*, *B*, *H*]
LEFT = (1, 2, … , 6) = [*A*, 0, *D*, *F*, *G*, 0]
RIGHT = (1, 2, … , 6) = [*B*, *H*, *A*, *C*, *E*, *C*]



Switchbox Routing – Example

TOP = (1, 2, … , 7) = [0, *D*, *F*, *H*, *E*, *C*, *C*]
BOT = (1, 2, … , 7) = [0, 0, *G*, *H*, *B*, *B*, *H*]
LEFT = (1, 2, … , 6) = [*A*, 0, *D*, *F*, *G*, 0]
RIGHT = (1, 2, … , 6) = [*B*, *H*, *A*, *C*, *E*, *C*]