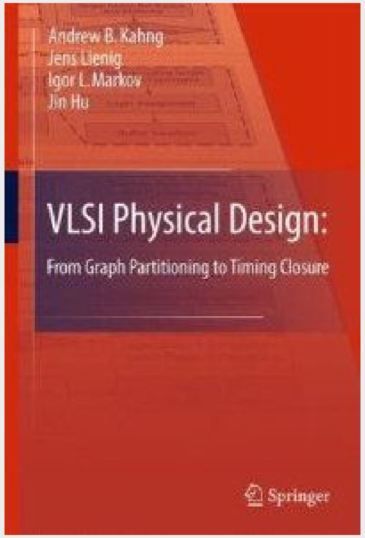
*VLSI Physical Design: From Graph Partitioning to Timing Closure*

# Chapter 6 – Detailed Routing

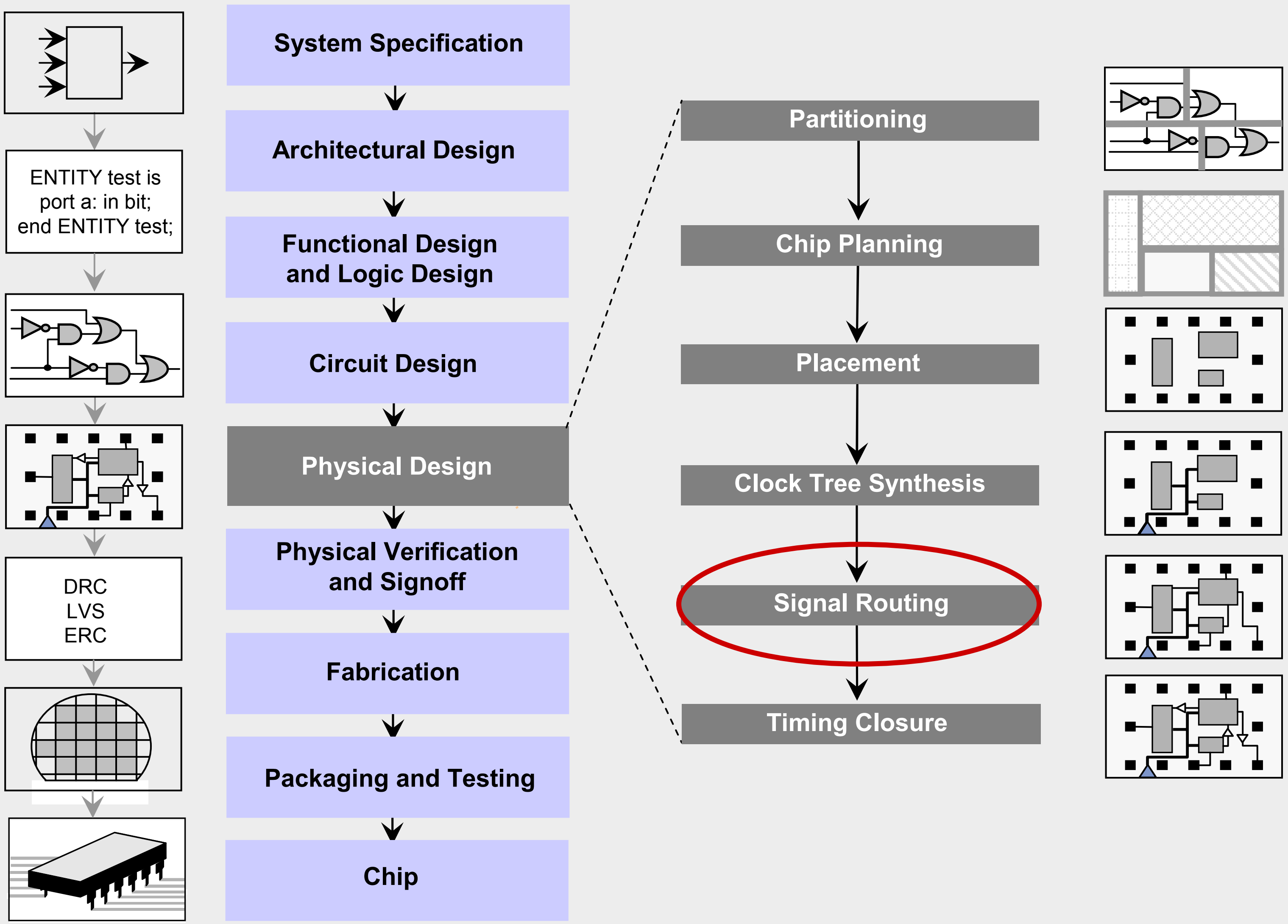



Original Authors:

Andrew B. Kahng, Jens Lienig, Igor L. Markov, Jin Hu

# Chapter 6 – Detailed Routing

# 6 Detailed Routing



- **Routing**
  - **Multi-Stage Routing of Signal Nets**
    - **Global Routing**: Coarse-grain assignment of routes to routing regions (Chap. 5)
    - **Detailed Routing**: Fine-grain assignment of routes to routing tracks (Chap. 6)
    - **Timing-Driven Routing**: Net topology optimization and resource allocation to critical nets (Chap. 8)
  - **Large Single-Net Routing**: Power (VDD) and Ground (GND) routing (Chap. 3)
  - **Geometric Techniques**: Non-Manhattan and clock routing (Chap. 7)

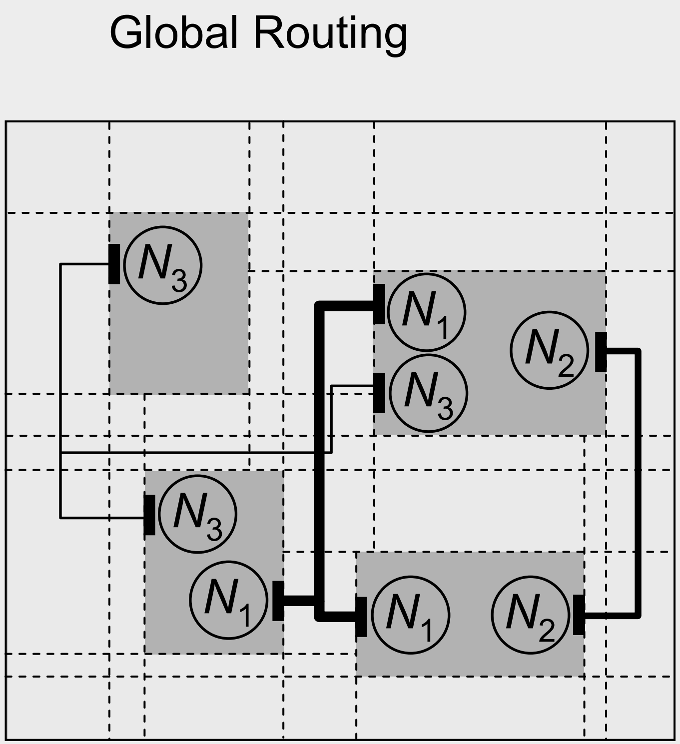
# 6 Detailed Routing

- The objective of detailed routing is to assign route segments of signal nets to specific routing tracks, vias, and metal layers in a manner consistent with given global routes of those nets

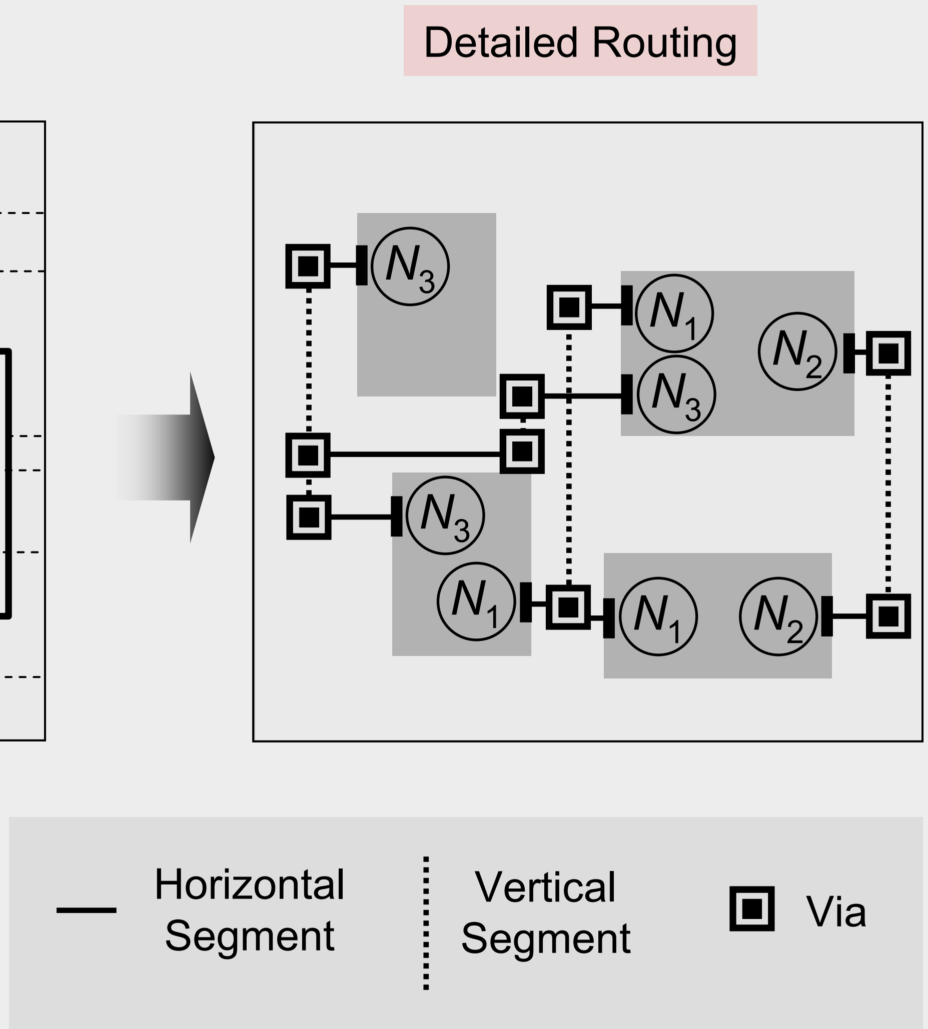
- Similar to global routing
  - Use physical wires to do connections
  - Estimating the wire resistance and capacitance, which determines whether the design meets timing requirements
- Detailed routing techniques are applied within routing regions, such as
  - channels (Sec. 6.3), switchboxes (Sec. 6.4) , and global routing cells (Sec. 6.5)
- Detailed routers must account for manufacturing rules and the impact of manufacturing faults (Sec. 6.6)

# Detailed Routing

- Detailed Routing Stages
  - Assign routing tracks
  - Perform entire routing – no open connection left
  - Search and repair – resolving all the physical design rules
  - Perform optimizations, e.g. add redundant vias (reduce resistivity, better yield)



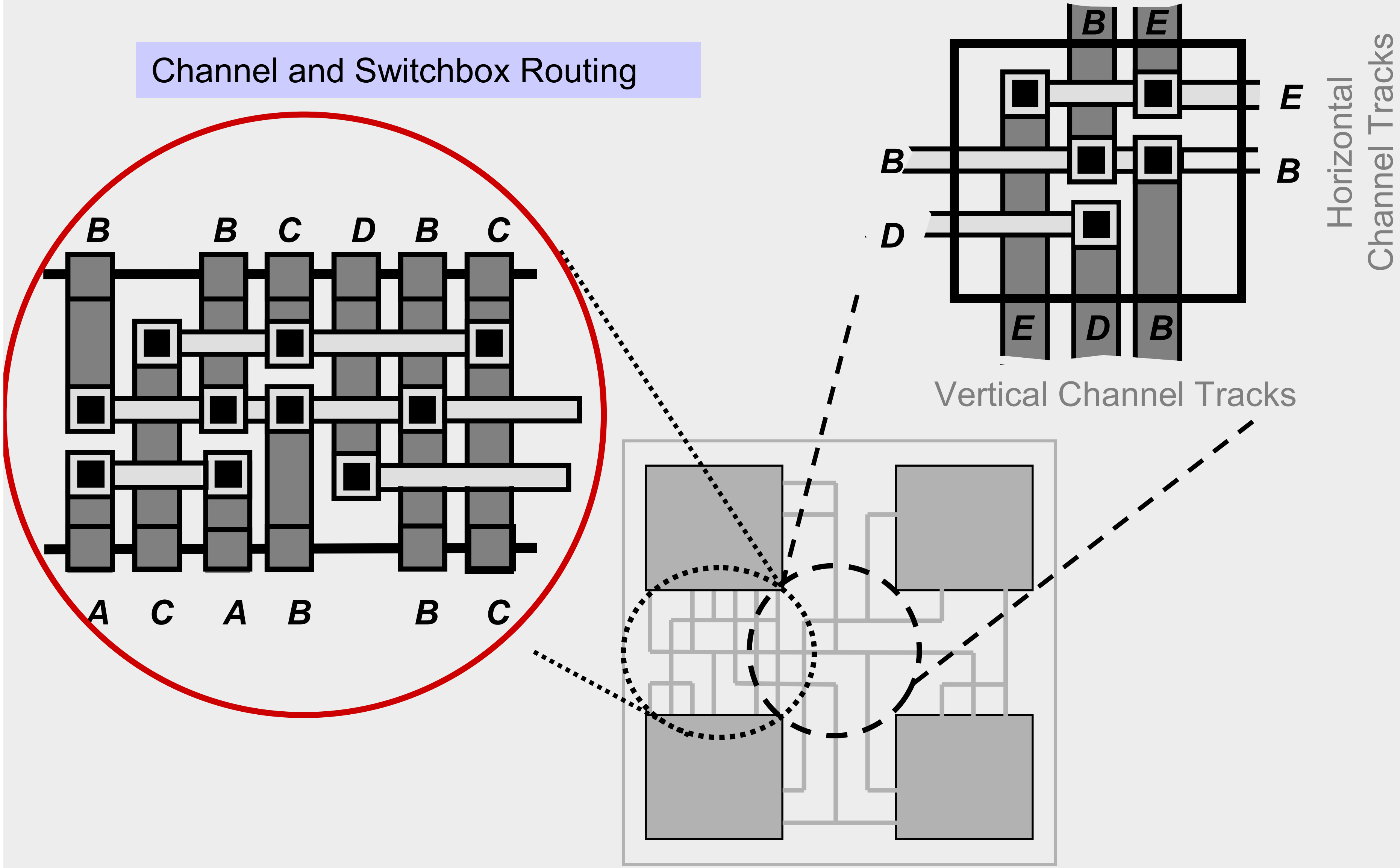
Global Routing
$N_3$
$N_1$
$N_2$
$N_3$
$N_3$
$N_1$
$N_1$
$N_2$

Detailed Routing
$N_3$
$N_1$
$N_2$
$N_3$
$N_3$
$N_1$
$N_1$
$N_2$
Horizontal Segment
Vertical Segment
Via

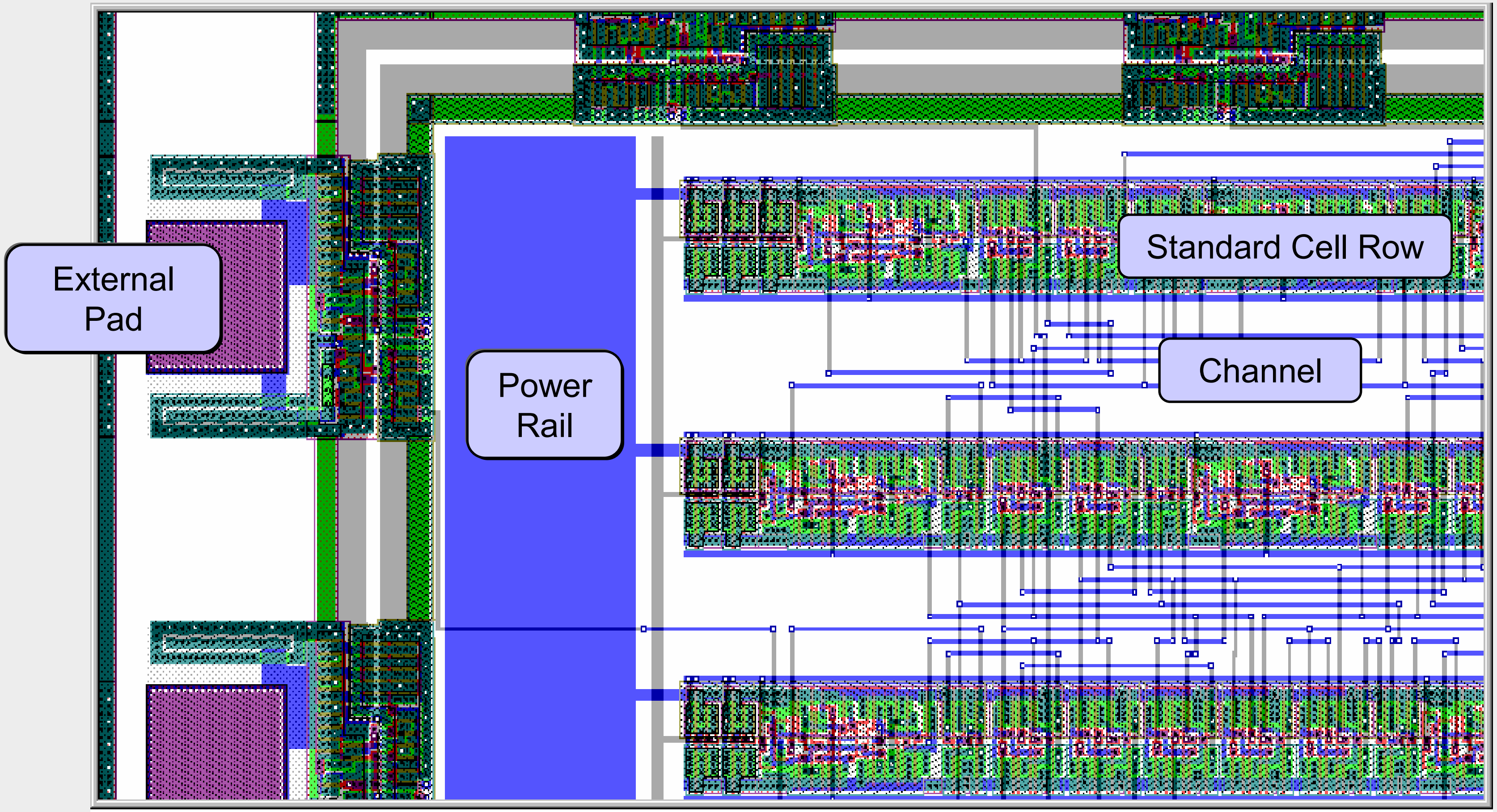
# 6.1 Terminology

Channel and Switchbox Routing

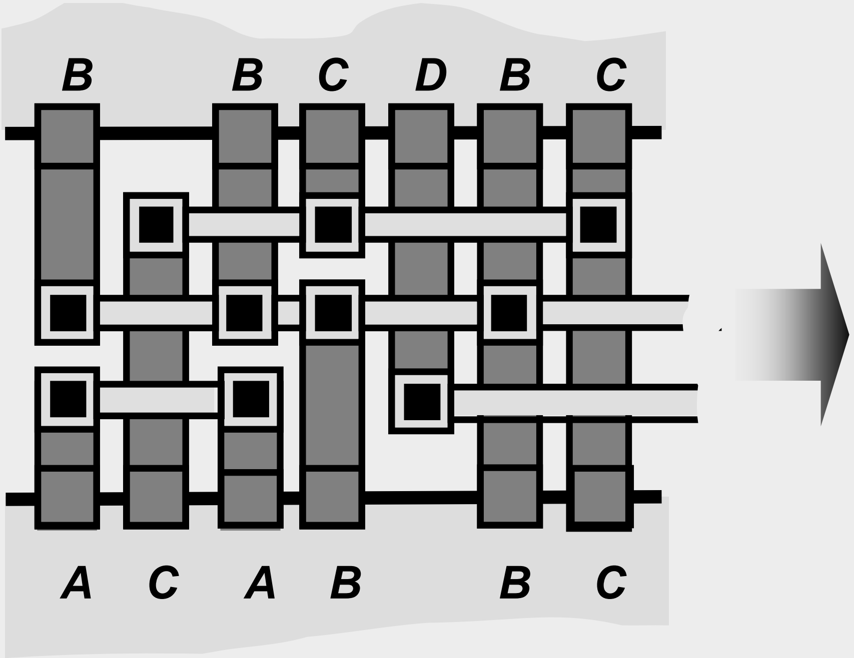


# 6.1 Terminology

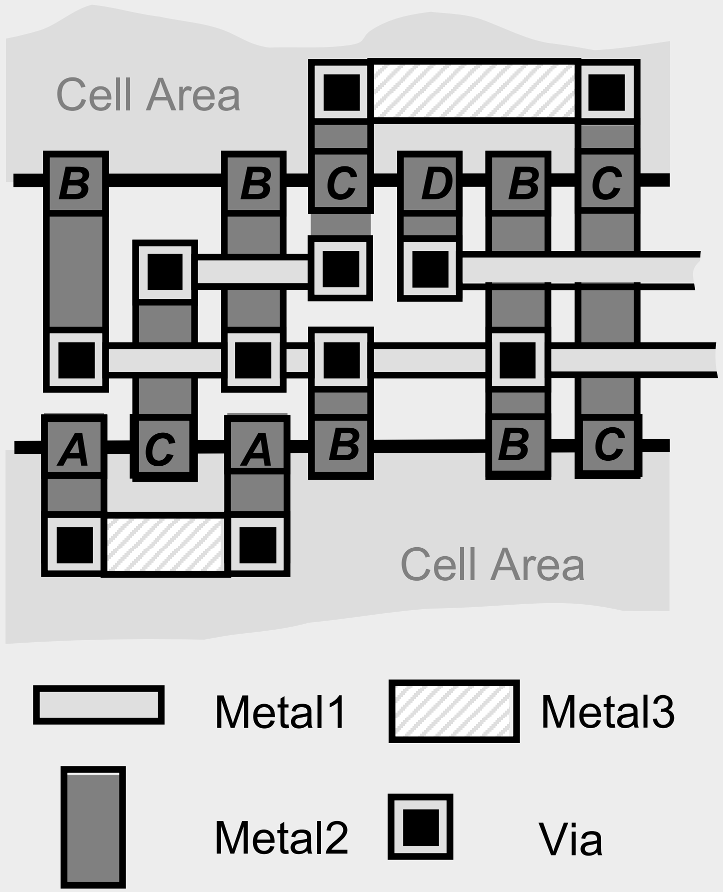
## Channel Routing

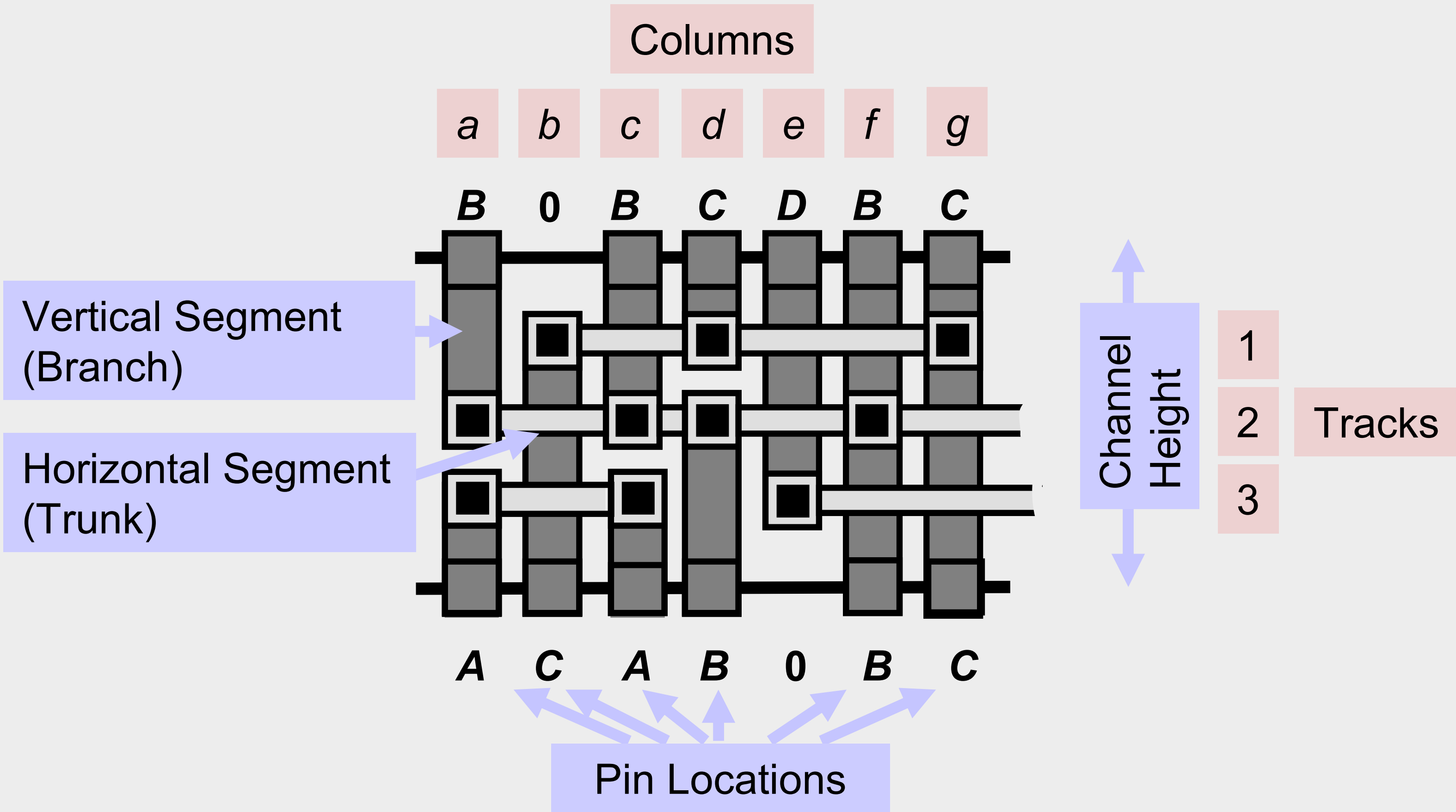


Two-Layer Channel Routing



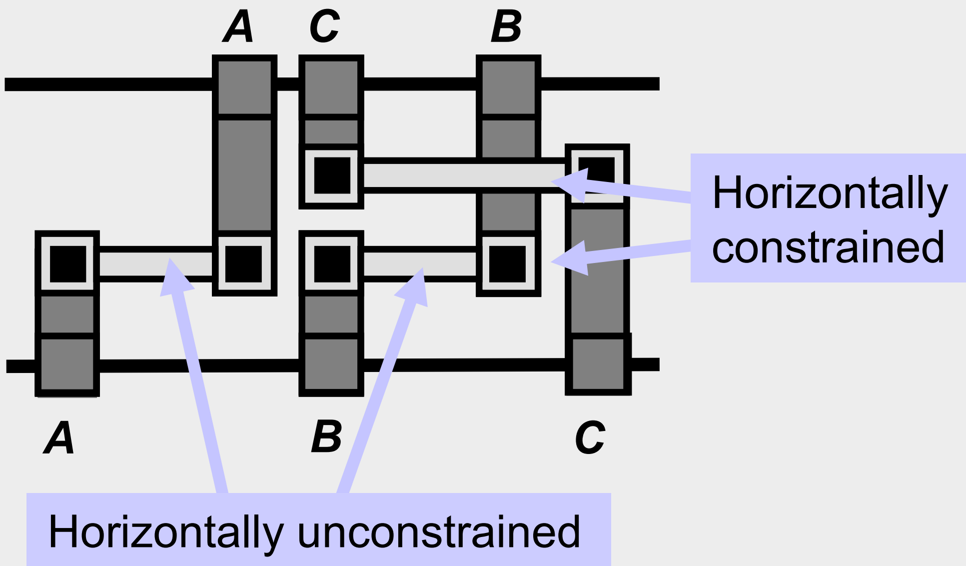

Three-Layer OTC Routing
OTC: Over the cell




Columns
a
b
c
d
e
f
g
B 0 B C D B C
Vertical Segment (Branch)
Horizontal Segment (Trunk)
Channel Height
1
2
3
Tracks
A C A B 0 B C
Pin Locations

## Horizontal Constraint

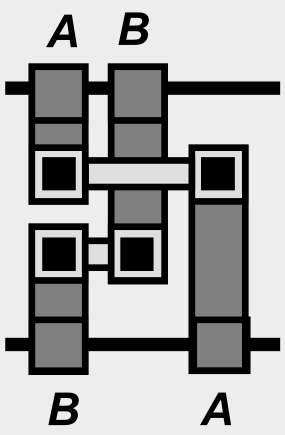
- Assumption: one layer for horizontal routing
- A horizontal constraint exists between two nets if their horizontal segments overlap



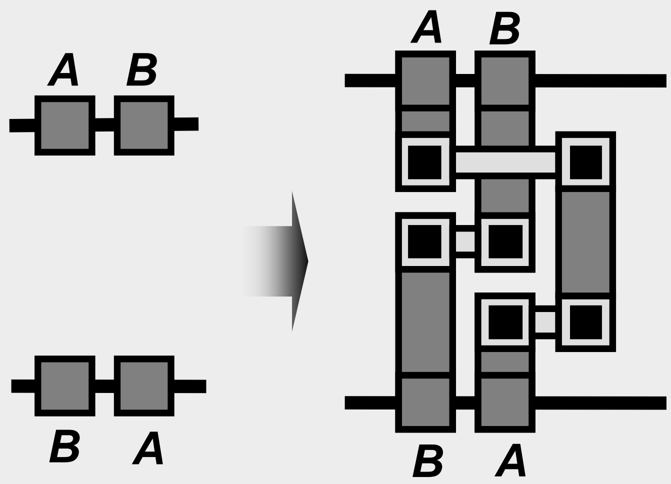
## Vertical Constraint

- A vertical constraint exists between two nets if they have pins in the same column

⇒ The vertical segment coming from the top must “stop” before overlapping with the vertical segment coming from the bottom in the same column




Vertically constrained without conflict




Vertically constrained with a vertical conflict

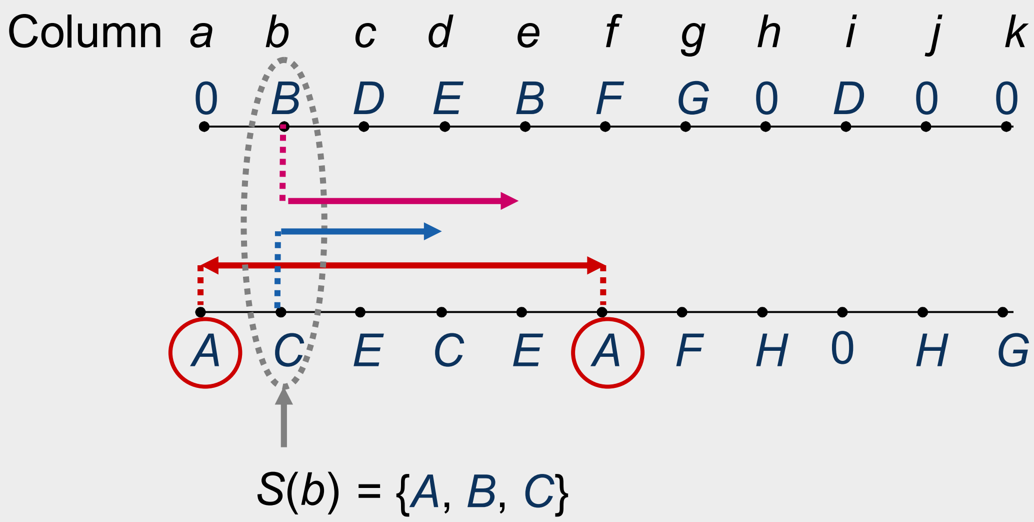
# 6.2 Horizontal and Vertical Constraint Graphs

## 6.2 Horizontal and Vertical Constraint Graphs

- The relative positions of nets in a channel routing instance can be modeled by horizontal and vertical constraint graphs

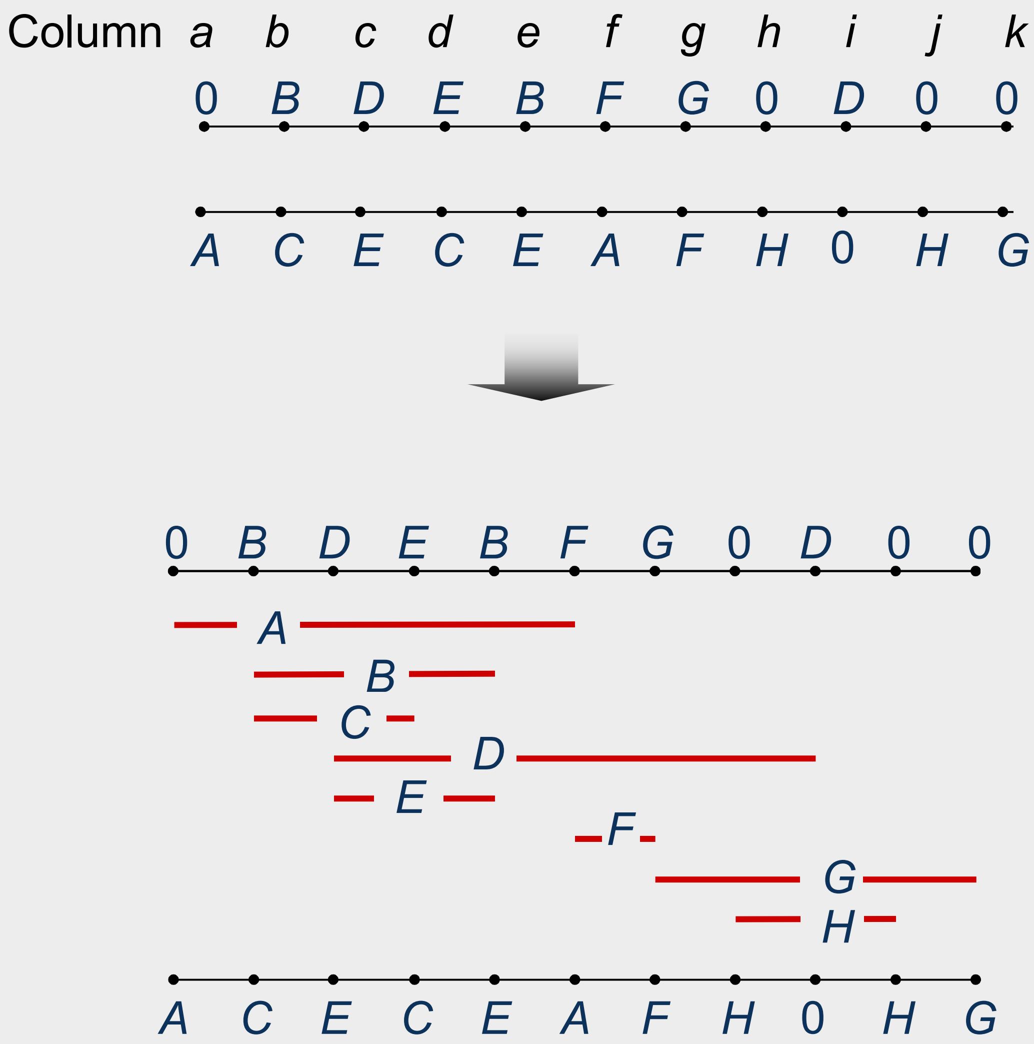
- These graphs are used to
  - initially predict the minimum number of tracks that are required
  - detect potential routing conflicts

# 6.3.1 Horizontal Constraint Graphs

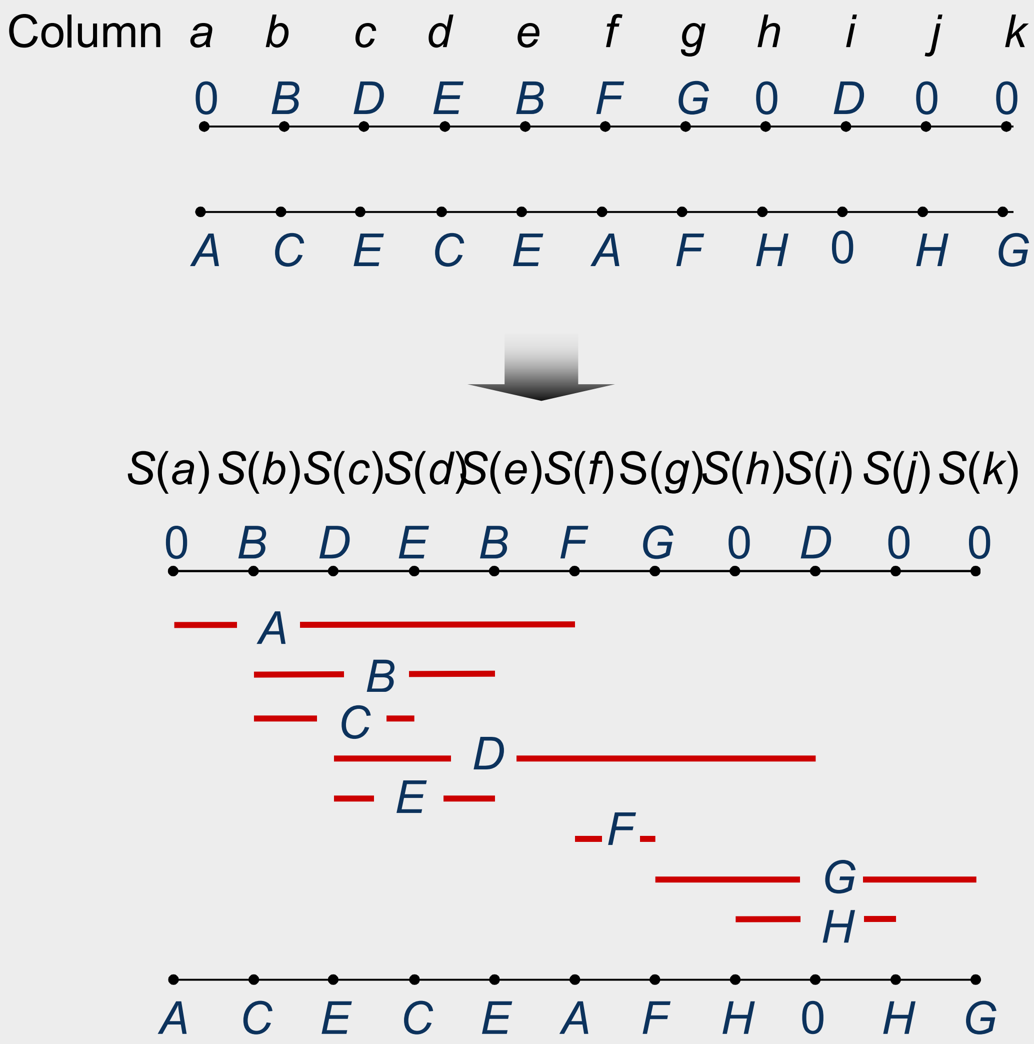



- Let $S(col)$ denote the set of nets that pass through column *col*
- $S(col)$ contains all nets that either (1) are connected to a pin in column *col* or (2) have pin connections to both the left and right of *col*
- Since horizontal segments cannot overlap, each net in $S(col)$ must be assigned to a different track in column *col*
- $S(col)$ represents the lower bound on the number of tracks in colum *col*; lower bound of the channel height is given by maximum cardinality of any $S(col)$

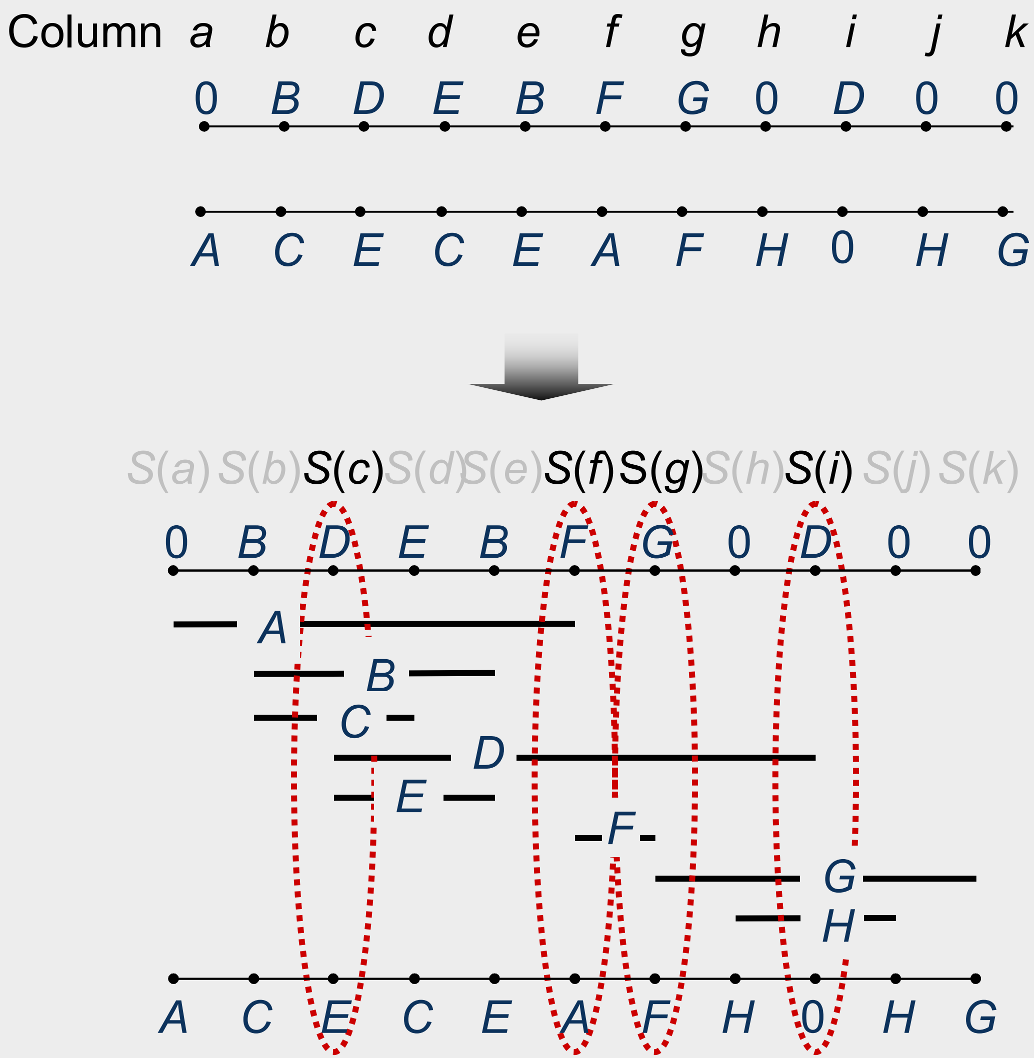
## 6.3.1 Horizontal Constraint Graphs



# 6.3.1 Horizontal Constraint Graphs




$S(a) = \{A\}$
$S(b) = \{A,B,C\}$
$S(c) = \{A,B,C,D,E\}$
$S(d) = \{A,B,C,D,E\}$
$S(e) = \{A,B,D,E\}$
$S(f) = \{A,D,F\}$
$S(g) = \{D,F,G\}$
$S(h) = \{D,G,H\}$
$S(i) = \{D,G,H\}$
$S(j) = \{G,H\}$
$S(k) = \{G\}$




$S(a) = \{A\}$

$S(b) = \{A,B,C\}$

$S(c) = \{A,B,C,D,E\}$

$S(d) = \{A,B,C,D,E\}$

$S(e) = \{A,B,D,E\}$

$S(f) = \{A,D,F\}$

$S(g) = \{D,F,G\}$

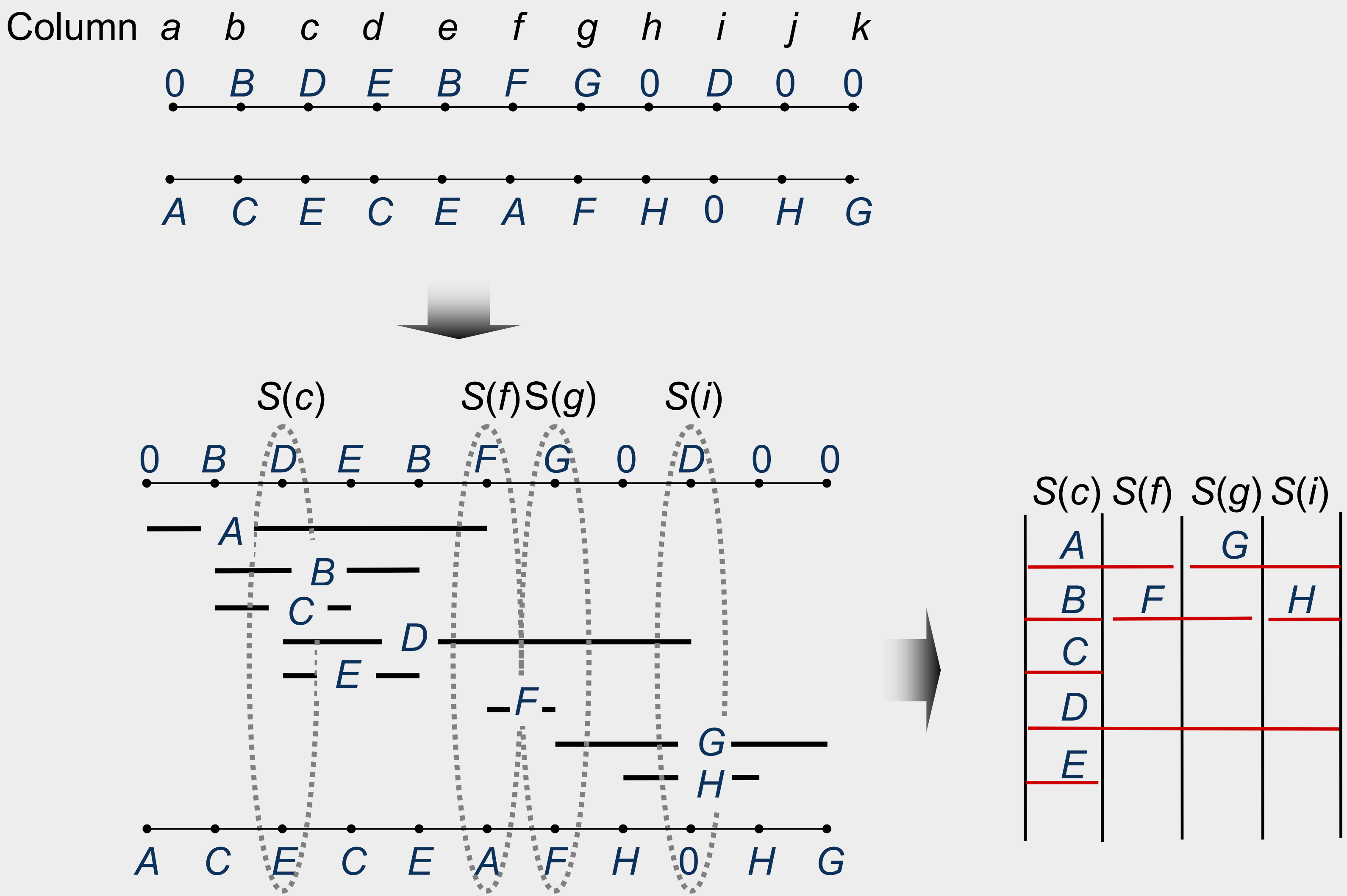
$S(h) = \{D,G,H\}$

$S(i) = \{D,G,H\}$

$S(j) = \{G,H\}$

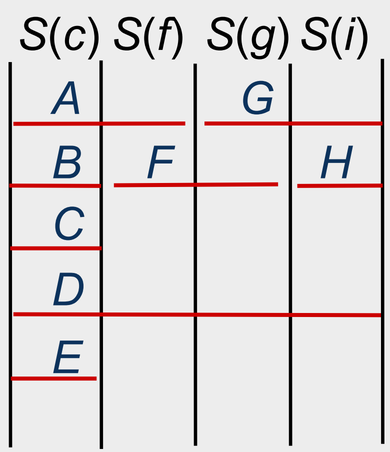
$S(k) = \{G\}$

## 6.3.1 Horizontal Constraint Graphs



| Column | *a* | *b* | *c* | *d* | *e* | *f* | *g* | *h* | *i* | *j* | *k* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | *B* | *D* | *E* | *B* | *F* | *G* | 0 | *D* | 0 | 0 |
| | *A* | *C* | *E* | *C* | *E* | *A* | *F* | *H* | 0 | *H* | *G* |

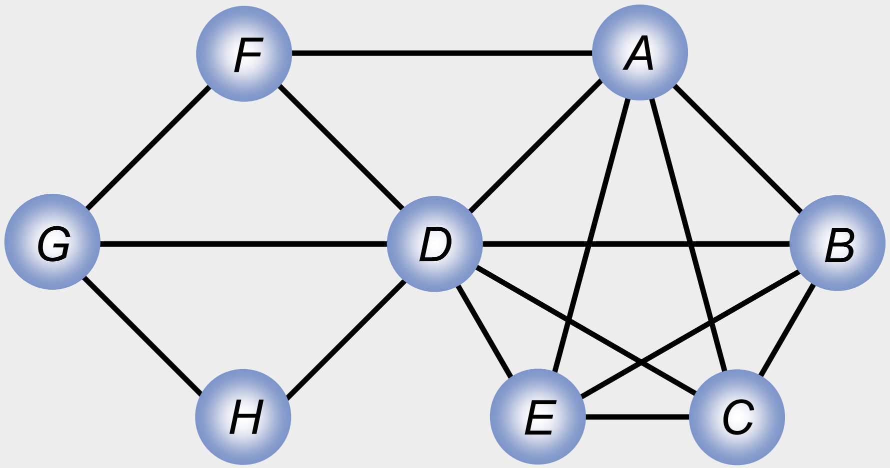
Lower bound on the number of tracks = 5

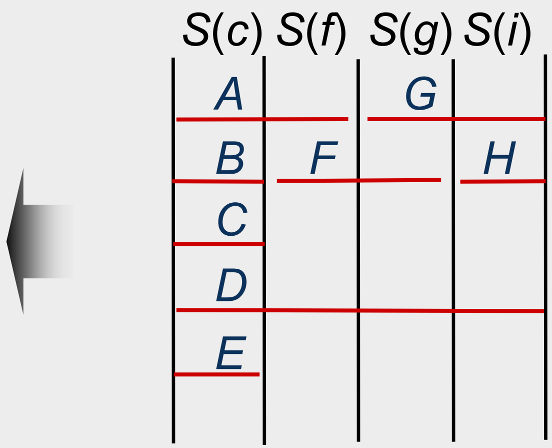


# 6.3.1 Horizontal Constraint Graphs

| Column | a | b | c | d | e | f | g | h | i | j | k |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | B | D | E | B | F | G | 0 | D | 0 | 0 |
| | A | C | E | C | E | A | F | H | 0 | H | G |

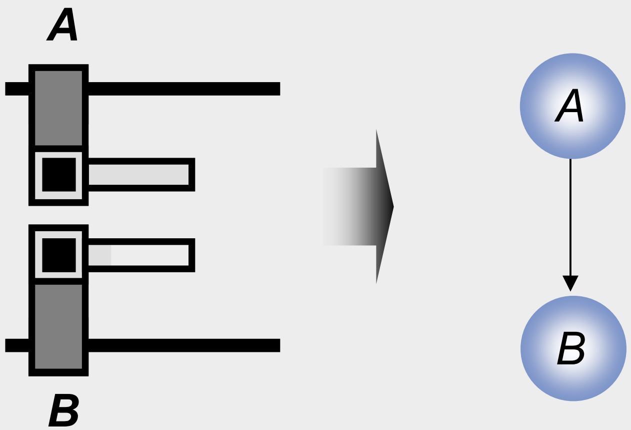
## Graphical Representation



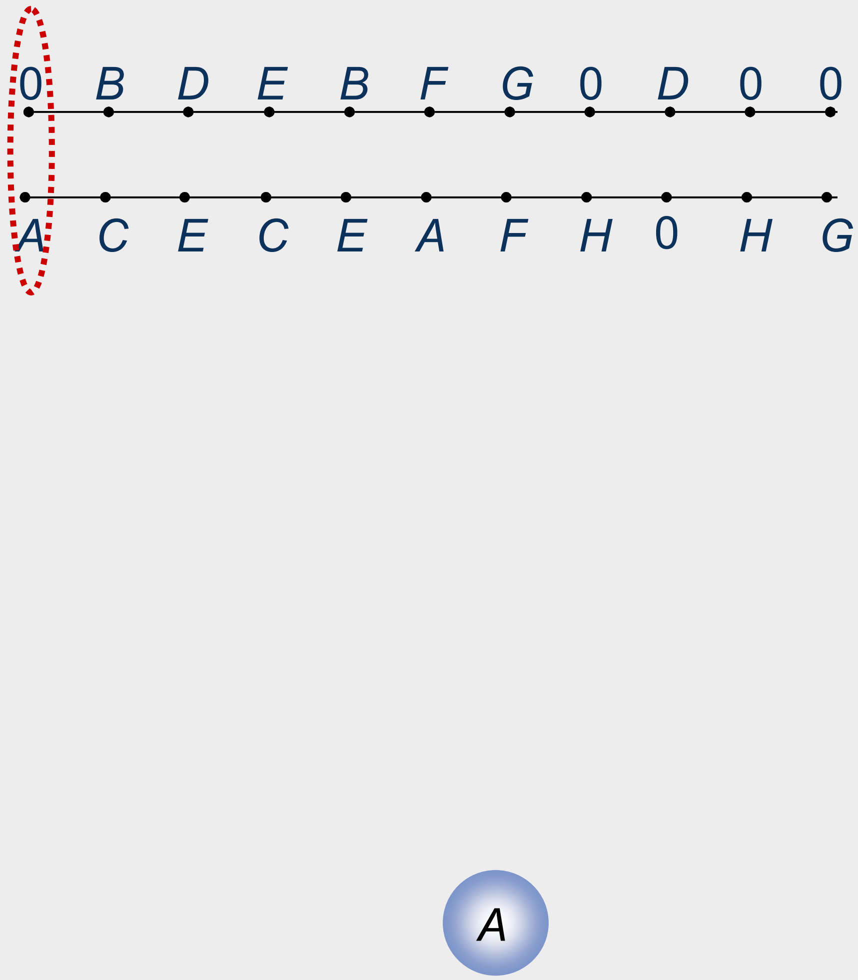



## 6.3.2 Vertical Constraint Graphs

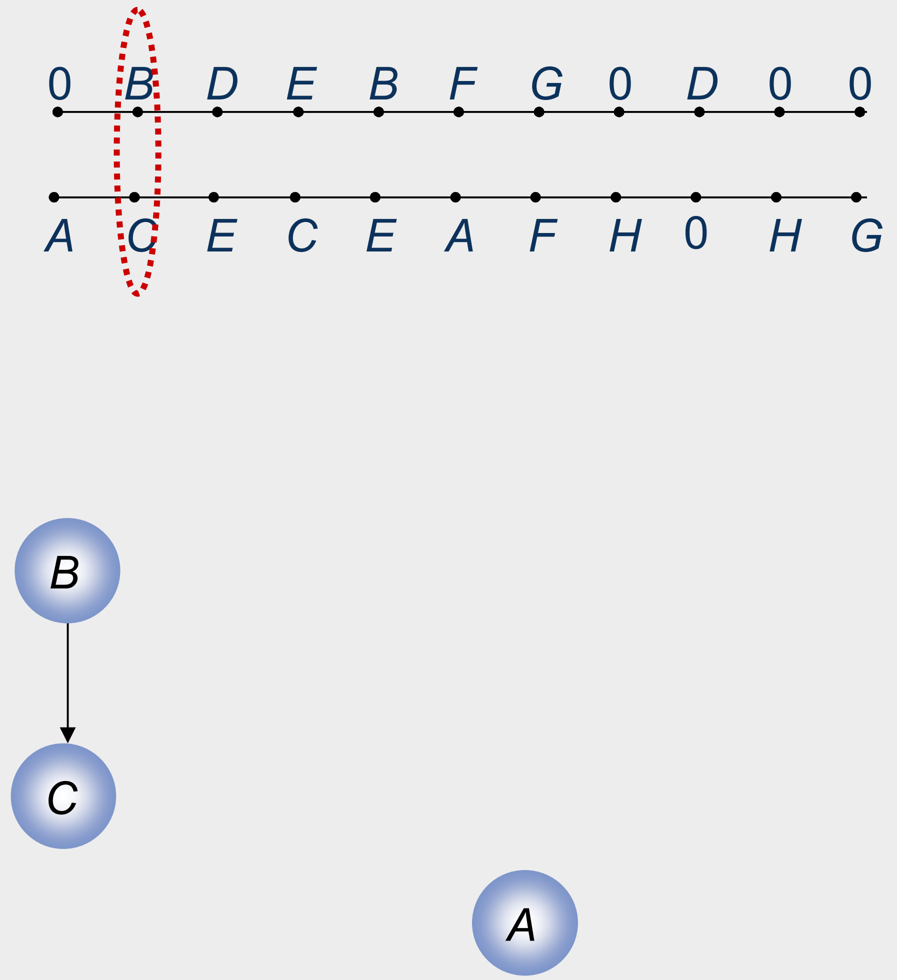
- A directed edge $e(i,j) \in E$ connects nodes $i$ and $j$ if the horizontal segment of net $i$ must be located above net $j$



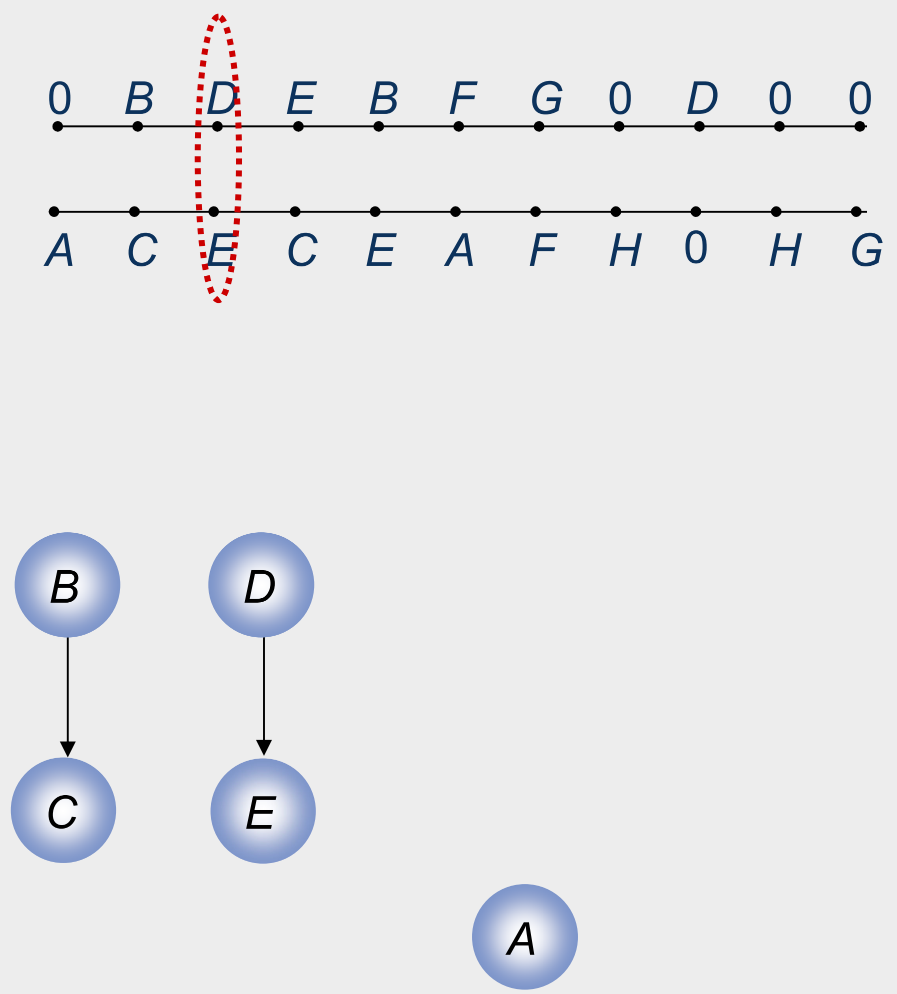
## 6.3.2 Vertical Constraint Graphs



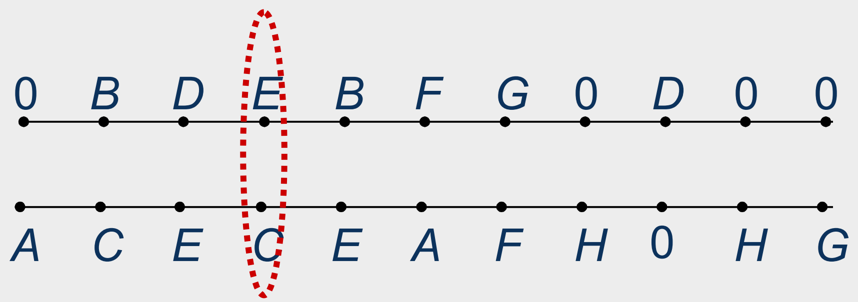
## 6.3.2 Vertical Constraint Graphs

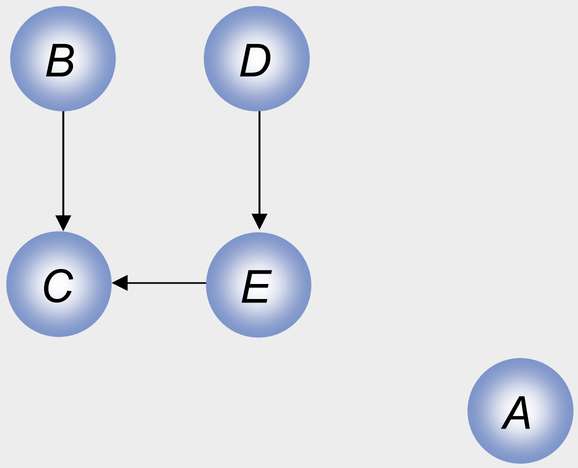


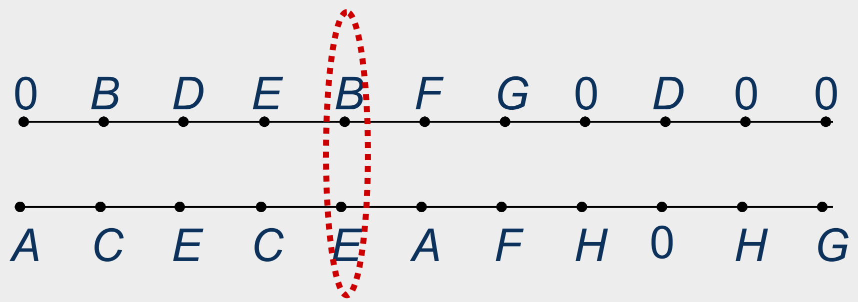
## 6.3.2 Vertical Constraint Graphs

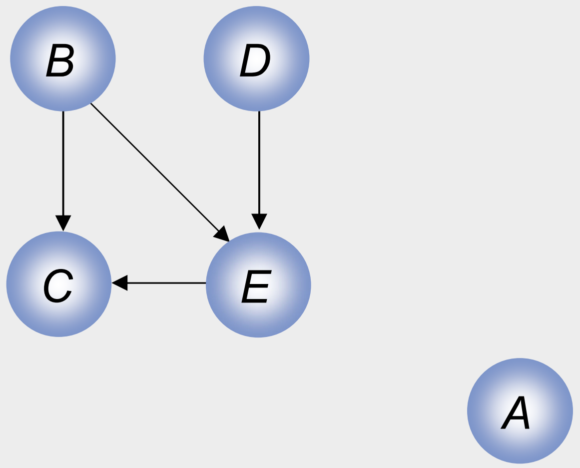


## 6.3.2 Vertical Constraint Graphs



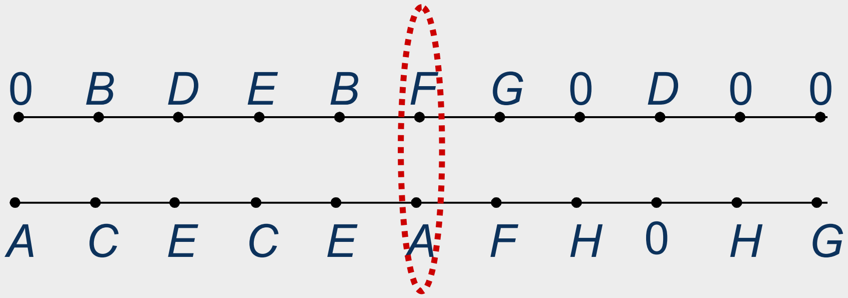


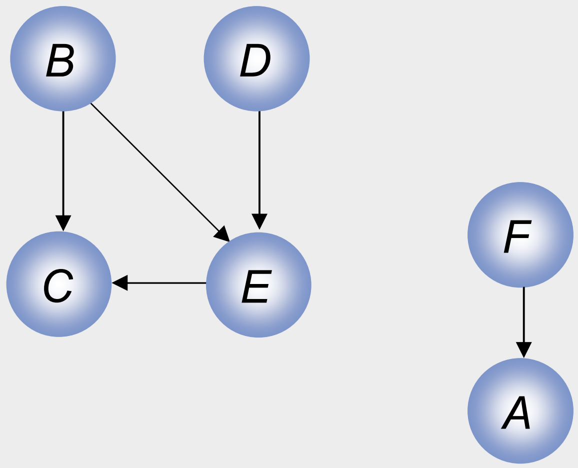




Vertical Constraint Graph (VCG)

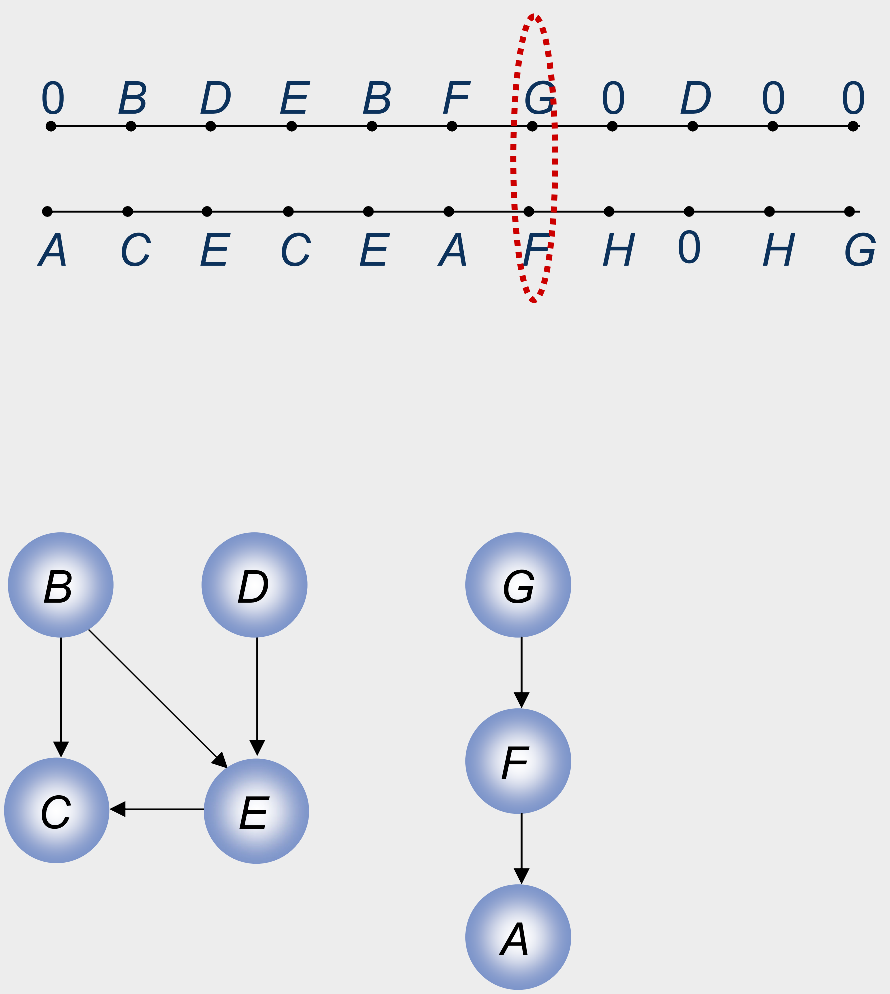
Note: an edge that can be derived by transitivity is not included, such as edge (*B*,*C*)

## 6.3.2 Vertical Constraint Graphs

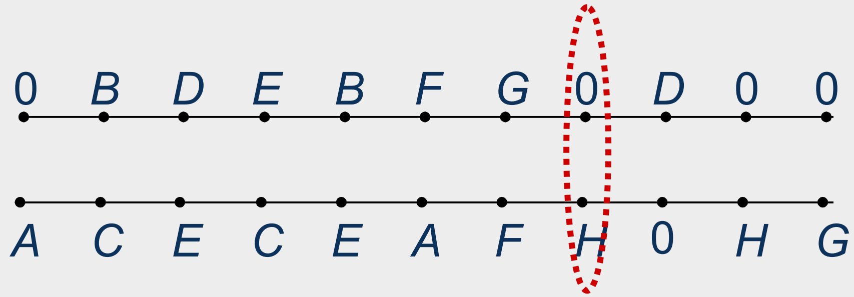


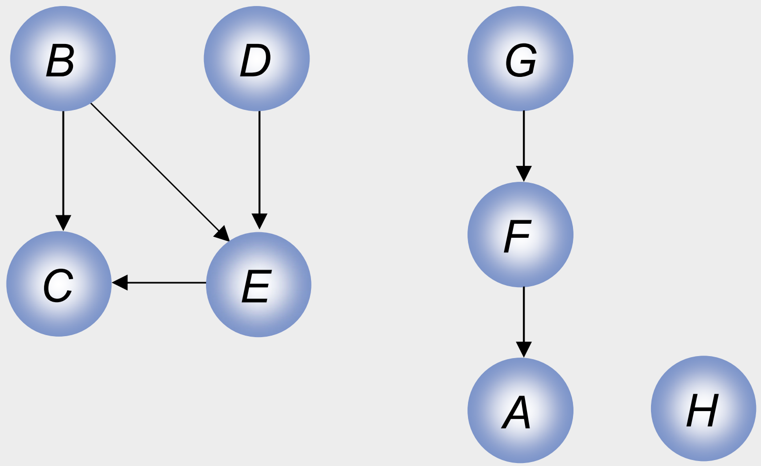



## 6.3.2 Vertical Constraint Graphs

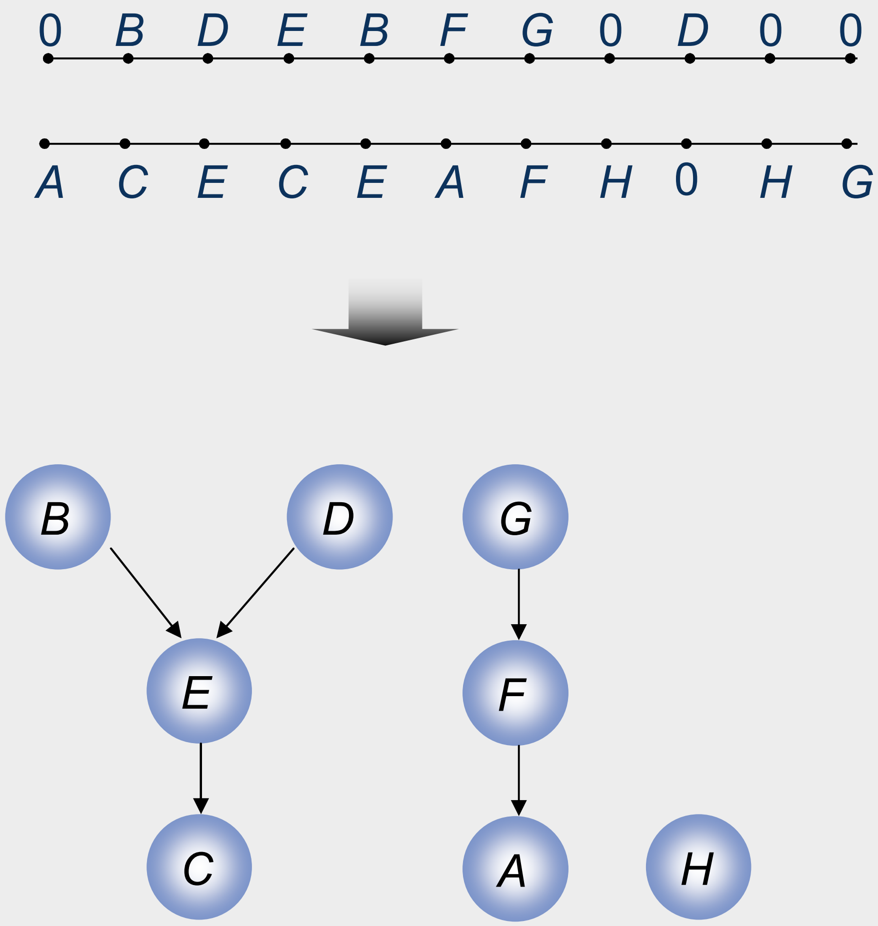


# 6.3.2 Vertical Constraint Graphs

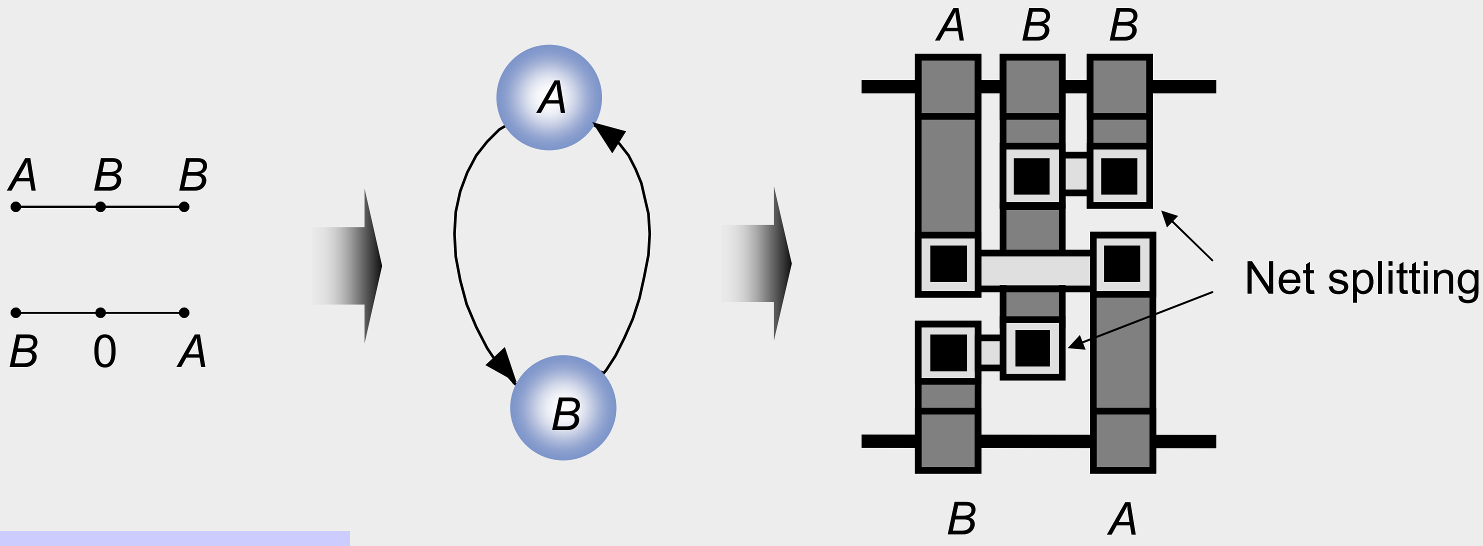


## 6.3.2 Vertical Constraint Graphs



# 6.3.2 Vertical Constraint Graphs




Cyclic conflict

# 6.3 Channel Routing Algorithms

## 6.3.1 Left-Edge Algorithm

- Based on the VCG and the zone representation, greedily maximizes the usage of each track
  - VCG: assignment order of nets to tracks
  - Zone representation: determines which nets may share the same track

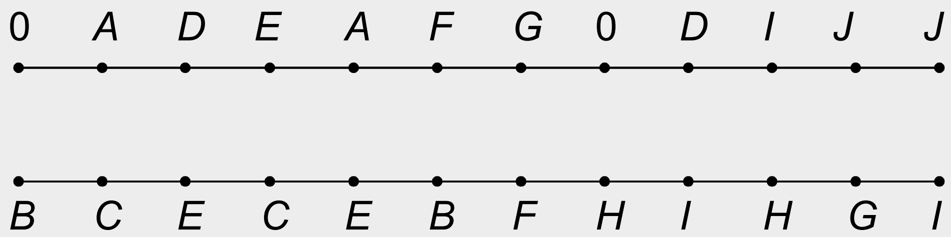
- Each net uses only one horizontal segment (trunk)

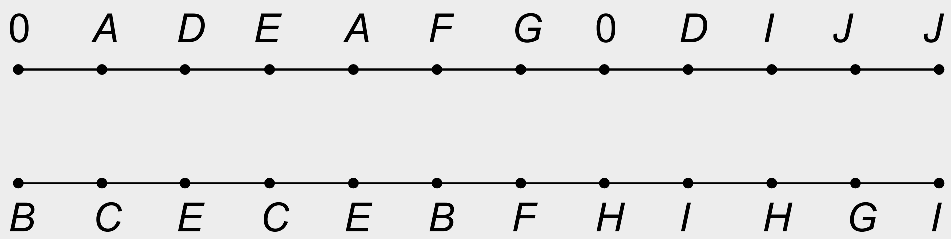
## 6.3.1 Left-Edge Algorithm

**Input:** channel routing instance *CR*
**Output:** track assignments for each net

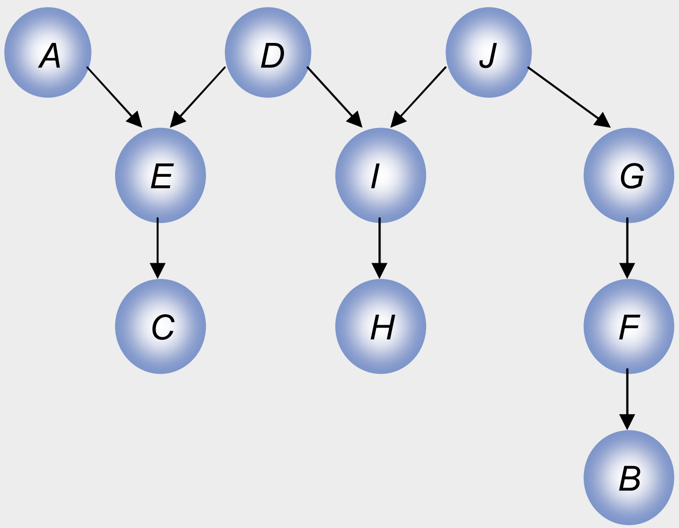
```
curr_track = 1                                  // start with topmost track
nets_unassigned = Netlist
while (nets_unassigned != Ø)                    // while nets still unassigned
   VCG = VCG(CR)                                // generate VCG and zone
   ZR = ZONE_REP(CR)                            //   representation
   SORT(nets_unassigned,start column)           // find left-to-right ordering
                                                //   of all unassigned nets
   for (i =1 to |nets_unassigned|)
      curr_net = nets_unassigned[i]
      if (PARENTS(curr_net) == Ø &&             // if curr_net has no parent
          (TRY_ASSIGN(curr_net,curr_track))     //   and does not cause
                                                //   conflicts on curr_track,
          ASSIGN(curr_net,curr_track)           //   assign curr-_net
          REMOVE(nets_unassigned,curr_net)
   curr_track = curr_track + 1                  // consider next track
```

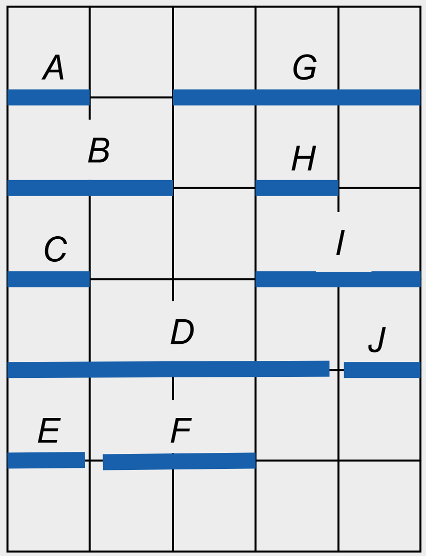
## 6.3.1 Left-Edge Algorithm – Example



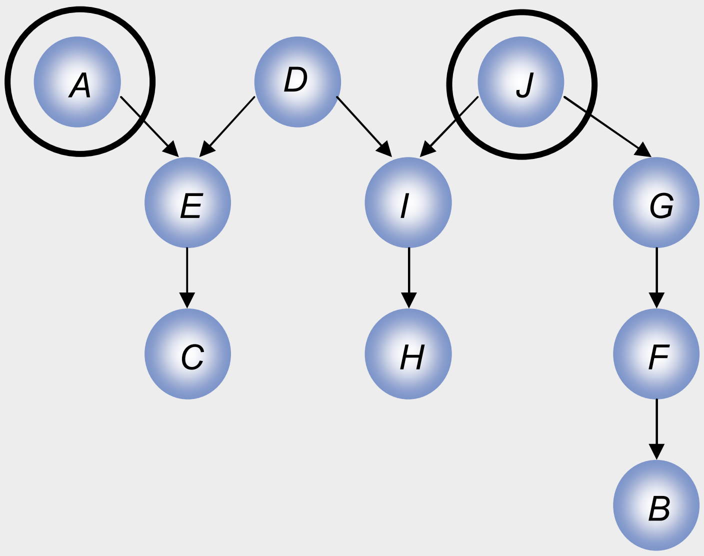


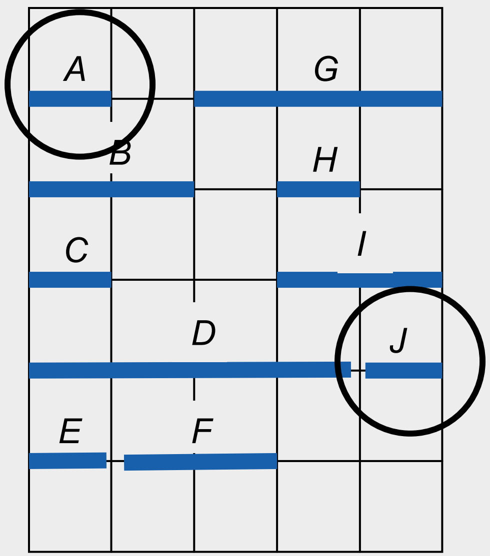

1. Generate VCG and zone representation



## 6.3.1 Left-Edge Algorithm – Example



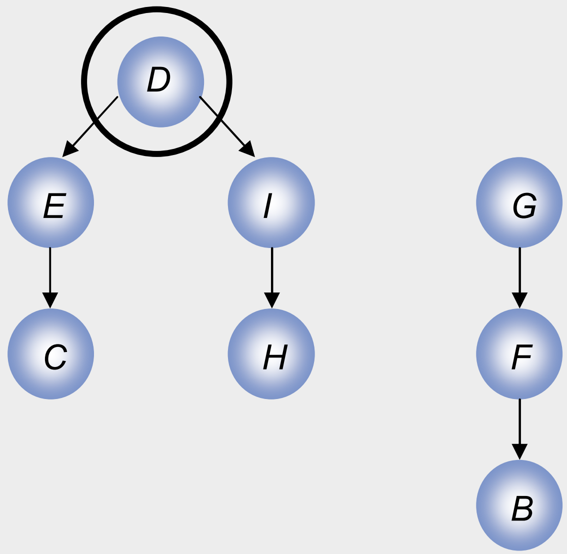

2. Consider next track
3. Find left-to-right ordering of all unassigned nets
   If *curr_net* has no parents and does not cause conflicts on *curr_track* assign *curr_net*

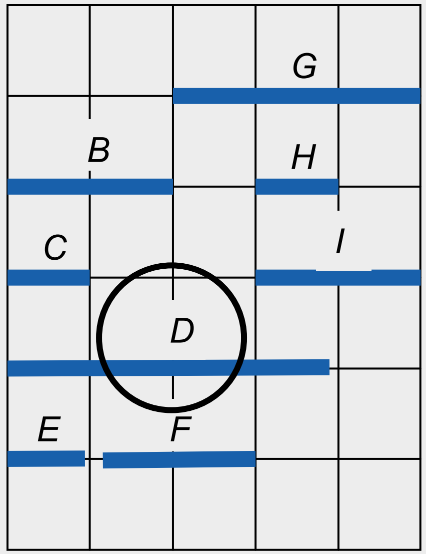
| *curr_track* = 1: | Net *A* | Net *J* |
|---|---|---|

4. Delete placed nets (*A*, *J* ) in VCG and zone represenation

# 6.3.1 Left-Edge Algorithm – Example

# 6.3.1 Left-Edge Algorithm – Example



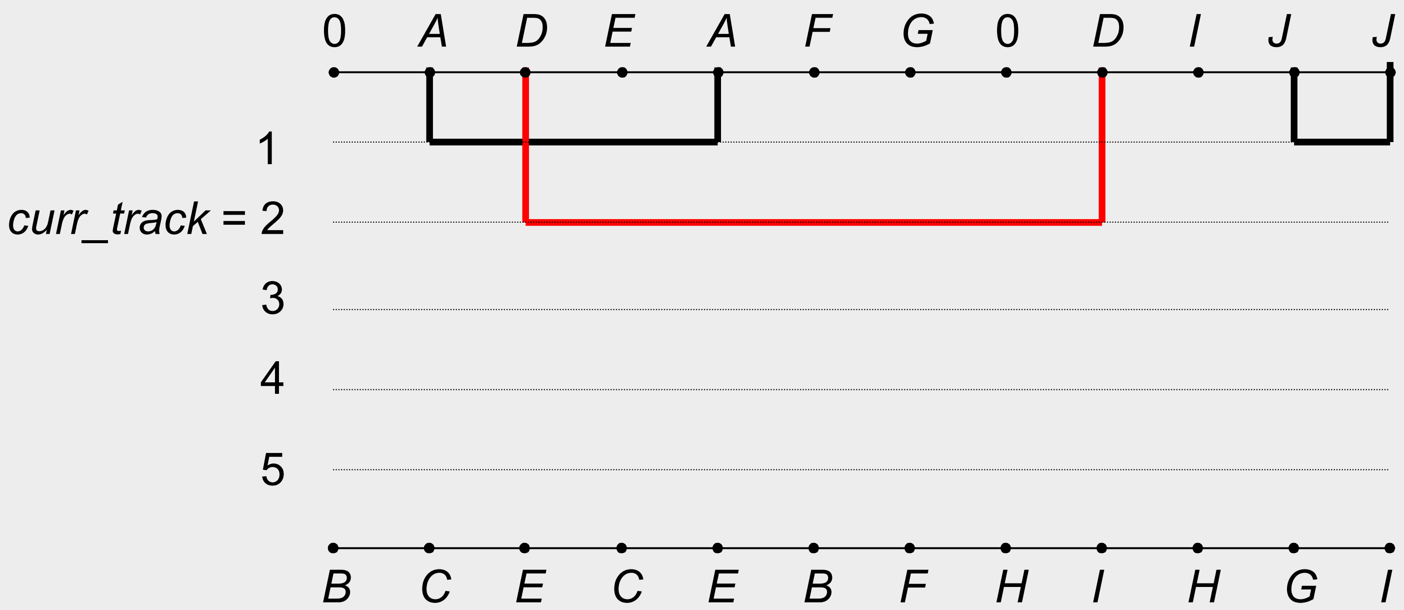




2. Consider next track
3. Find left-to-right ordering of all unassigned nets
   If *curr_net* has no parents and does not cause conflicts on *curr_track* assign *curr_net*

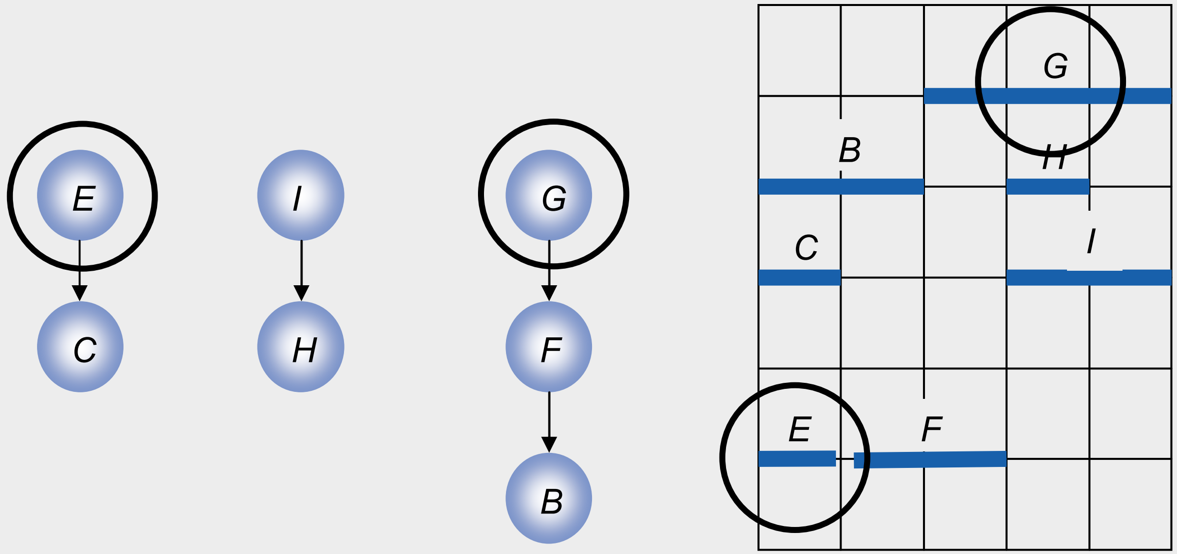
*curr_track* = 2: Net *D*

4. Delete placed nets (*D*) in VCG and zone representation

# 6.3.1 Left-Edge Algorithm – Example



# 6.3.1 Left-Edge Algorithm – Example

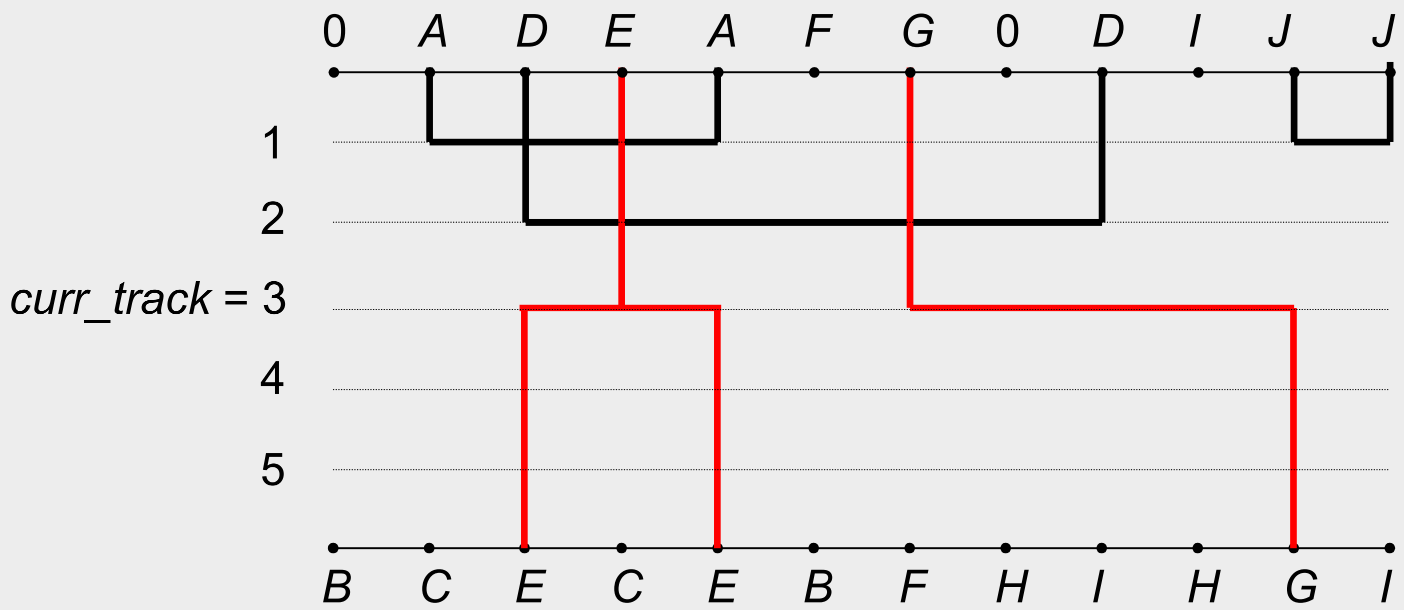



2. Consider next track
3. Find left-to-right ordering of all unassigned nets
   If *curr_net* has no parents and does not cause conflicts on *curr_track* assign *curr_net*

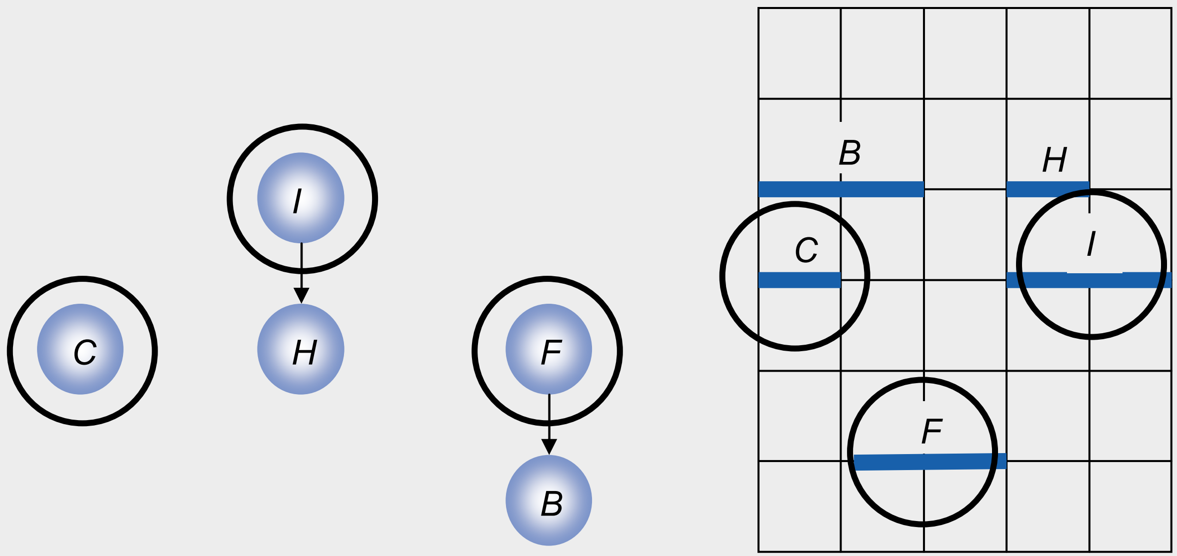
*curr_track* = 3: Net *E* Net *G*

4. Delete placed nets (*E*, *G* ) in VCG and zone representation

# 6.3.1 Left-Edge Algorithm – Example



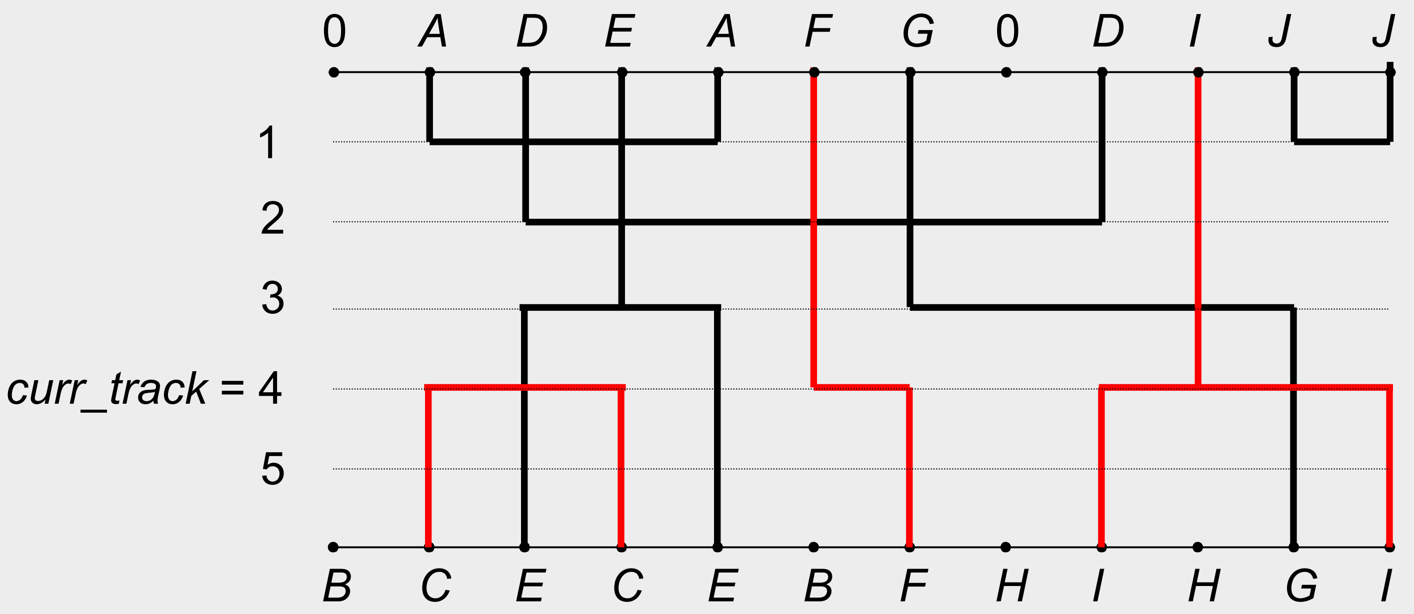
# 6.3.1 Left-Edge Algorithm – Example




2. Consider next track
3. Find left-to-right ordering of all unassigned nets
   If *curr_net* has no parents and does not cause conflicts on *curr_track* assign *curr_net*

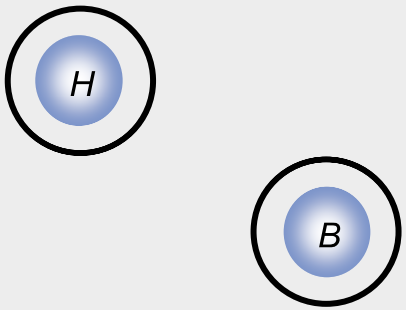
| *curr_track* = 4: | Net *C* | Net *F* | Net *I* |
|---|---|---|---|

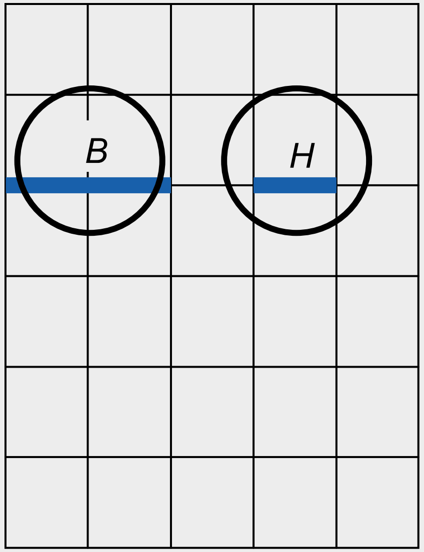
4. Delete placed nets (*C*, *F*, *I* ) in VCG and zone representation



0 A D E A F G 0 D I J J
1
2
3
curr_track = 4
5
B C E C E B F H I H G I

## 6.3.1 Left-Edge Algorithm – Example

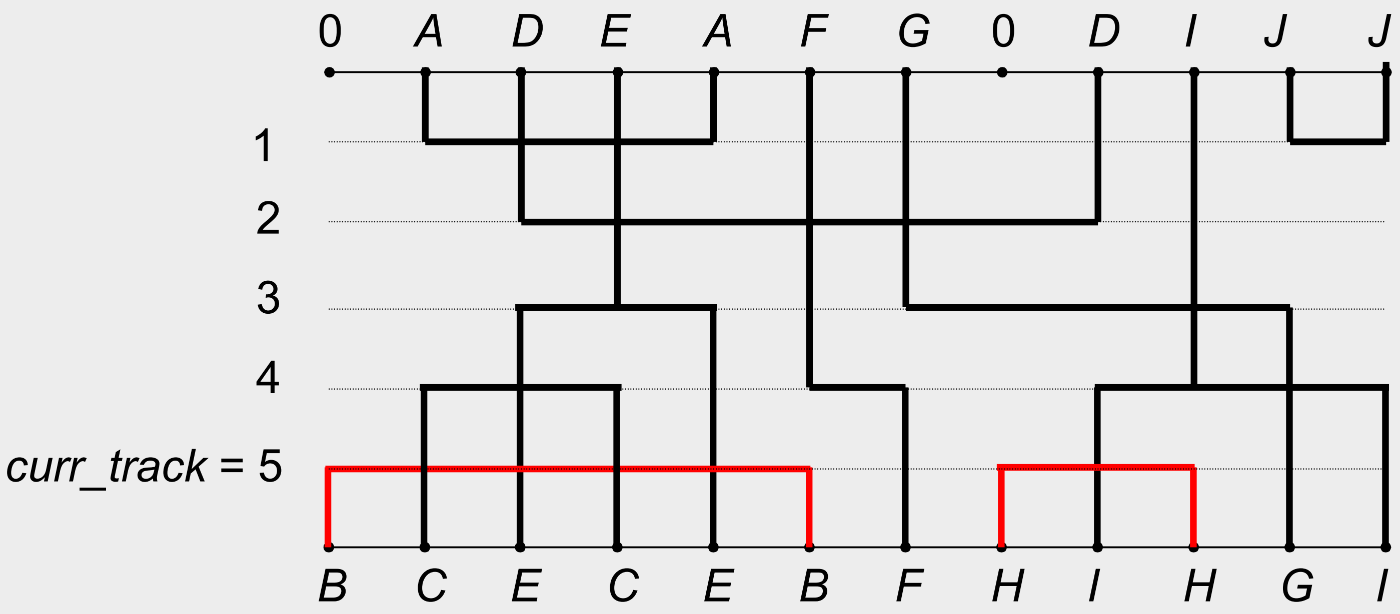



2. Consider next track
3. Find left-to-right ordering of all unassigned nets
   If *curr_net* has no parents and does not cause conflicts on *curr_track* assign *curr_net*

| *curr_track* = 5: | Net *B* | Net *H* |
| --- | --- | --- |

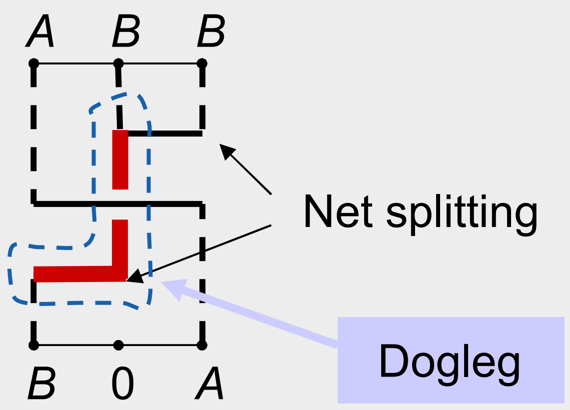
4. Delete placed nets (*B*, *H*) in VCG and zone representation



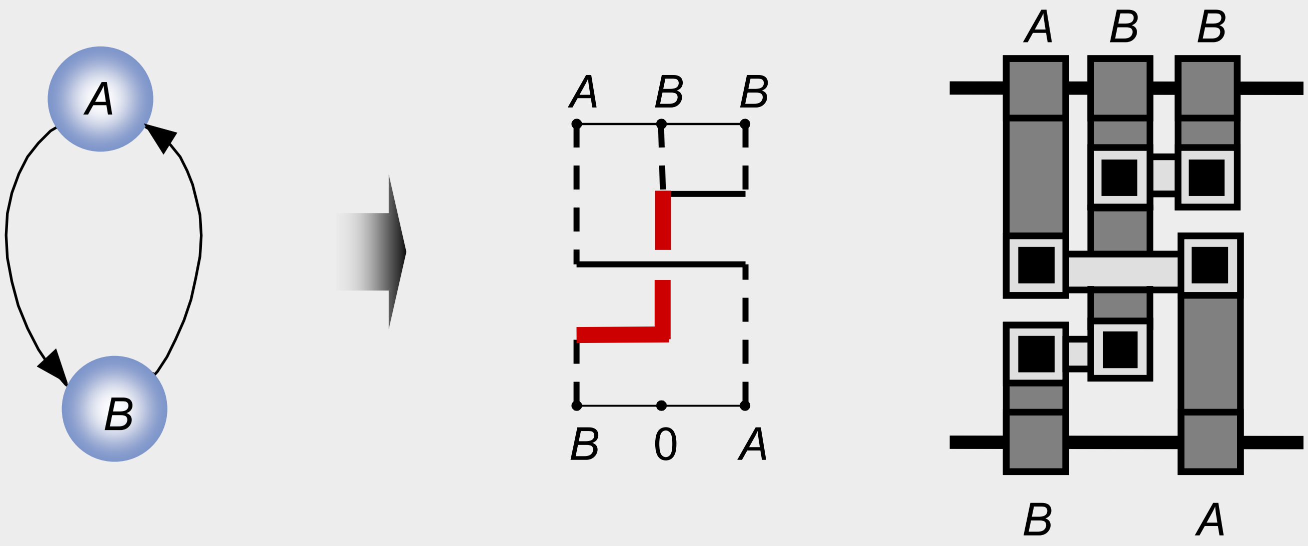

Routing result

# 6.3.2 Dogleg Routing

- Improving left-edge algorithm by net splitting
- Two advantages:
  - Alleviates conflicts in VCG
  - Number of tracks can often be reduced

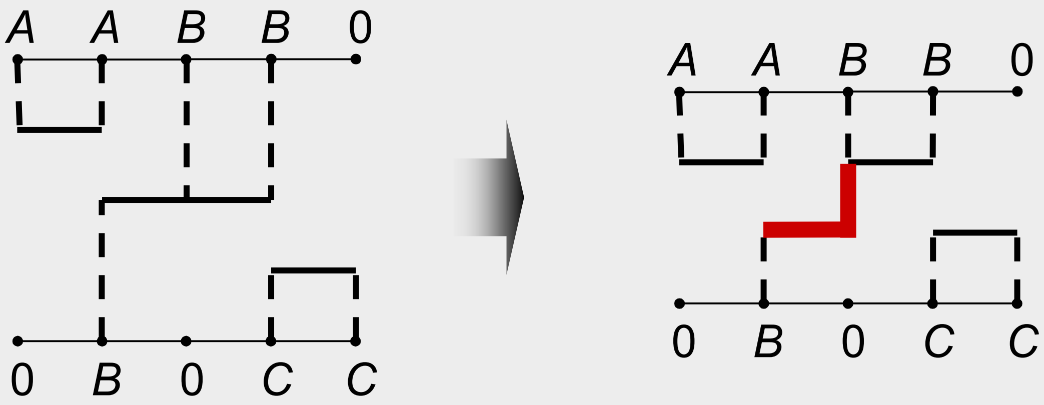


**Conflict alleviation using a dogleg**

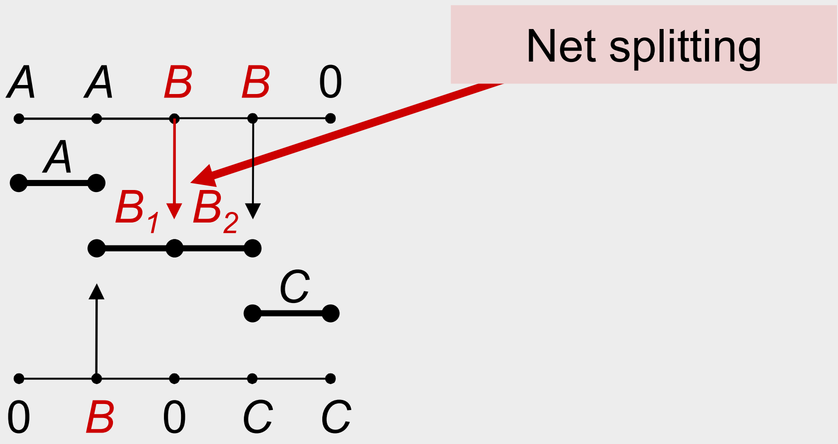


# 6.3.2 Dogleg Routing

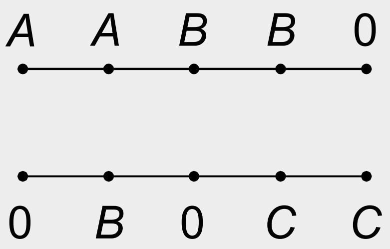
**Track reduction using a dogleg**



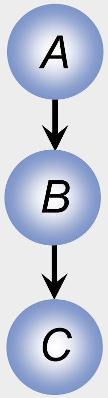
- Splitting $p$-pin nets ($p > 2$) into $p - 1$ horizontal segments
- Net splitting occurs only in columns that contain a pin of the given net
- After net splitting, the algorithm follows the left-edge algorithm



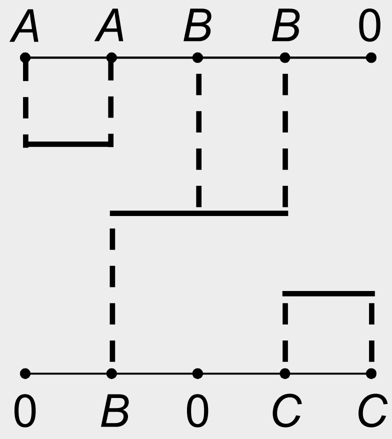
# 6.3.2 Dogleg Routing



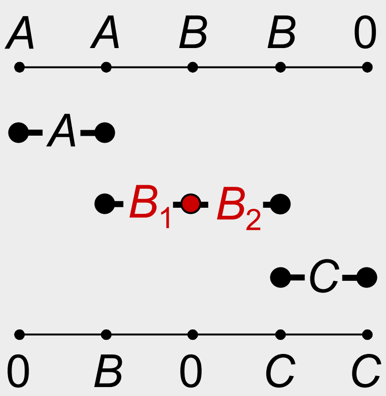

Channel routing problem



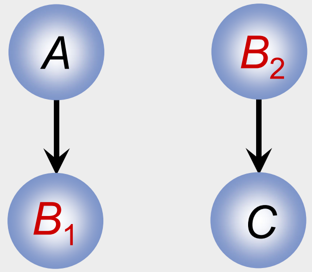

VCG without net splitting



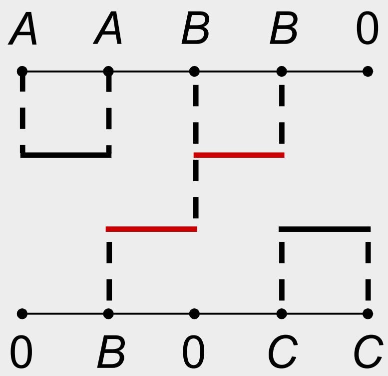

Channel routing solution




Net splitting

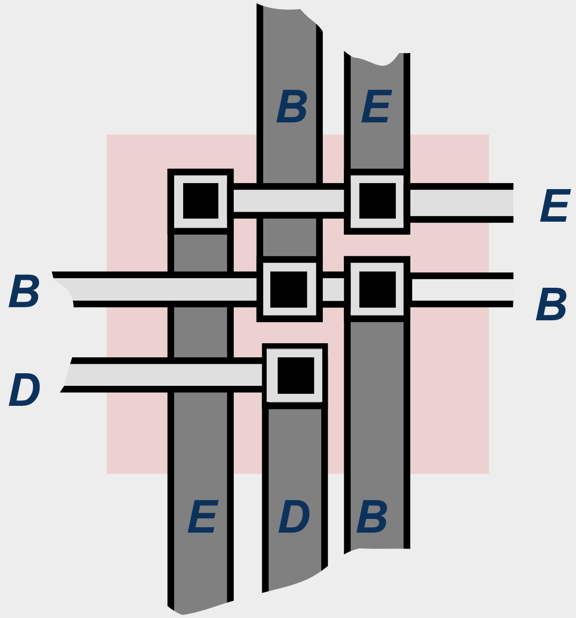


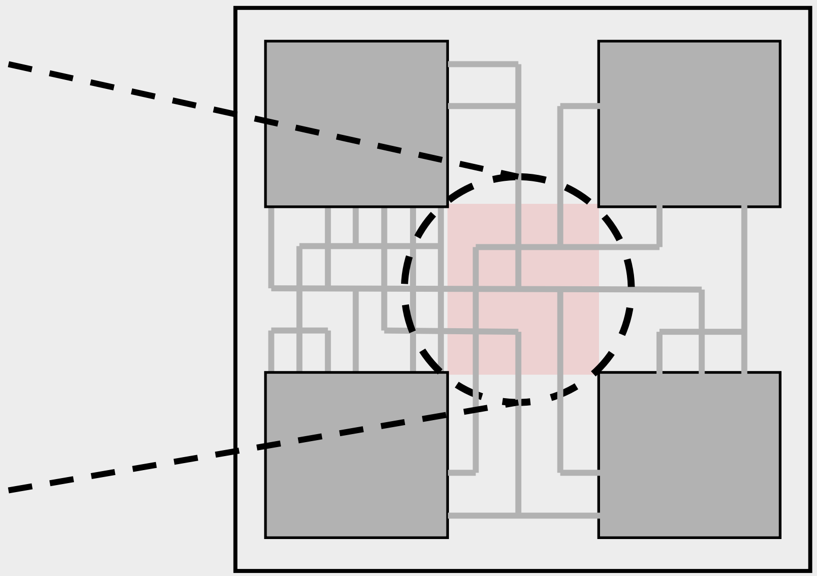

VCG with net splitting




Channel routing solution

# 6.4 Switchbox Routing



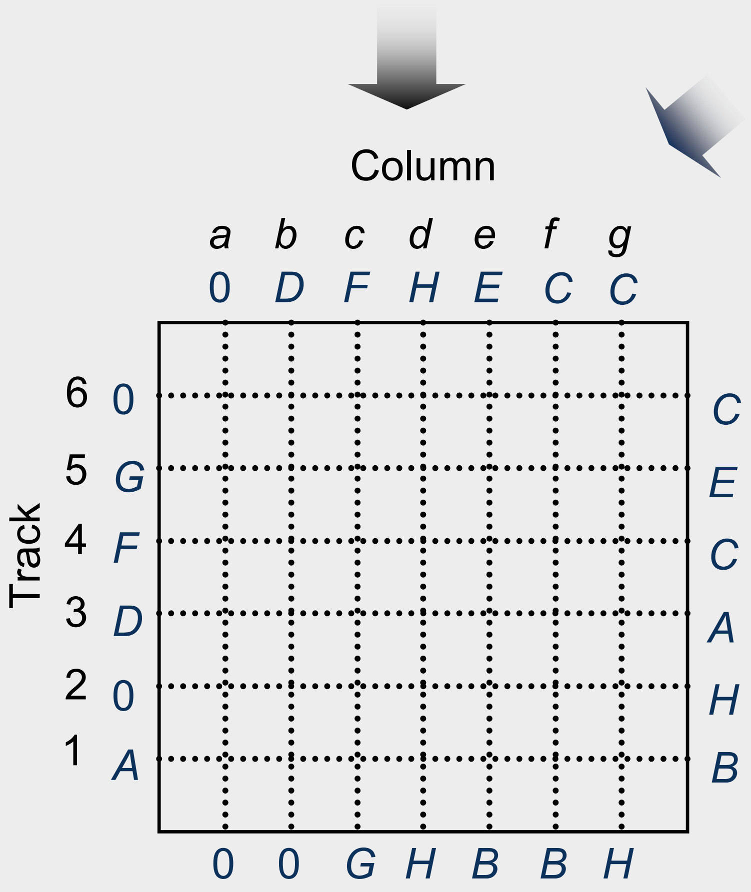



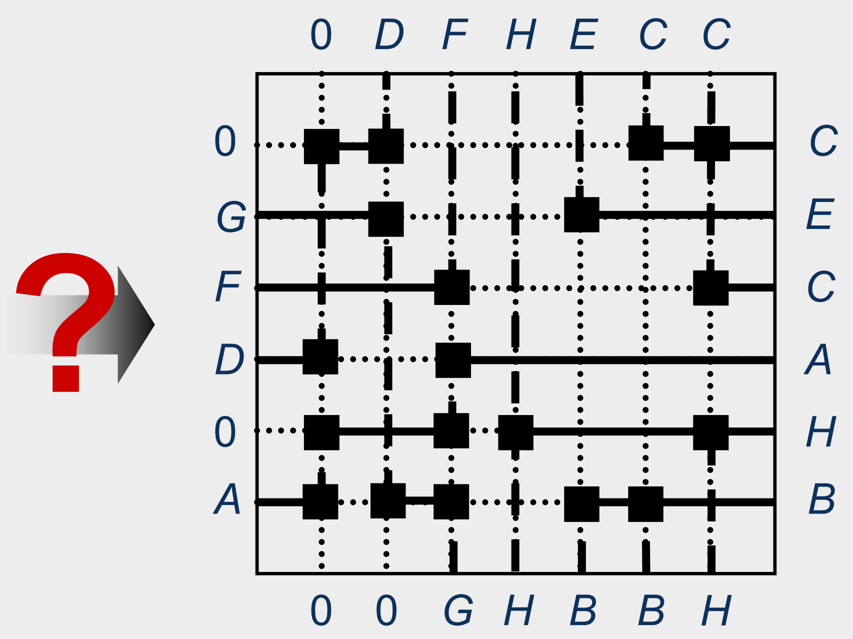
- Fixed dimensions and pin connections on all four sides
- Defined by four vectors *TOP*, *BOT*, *LEFT*, *RIGHT*
- Switchbox routing algorithms are usually derived from (greedy) channel routing algorithms

# 6.4 Switchbox Routing

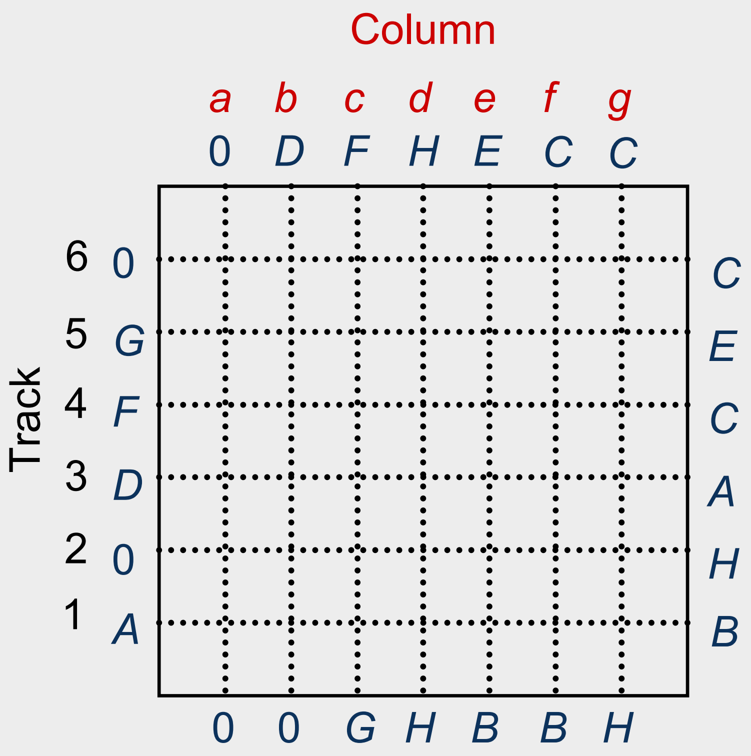
$R = \{0, 1, 2, \ldots, 8\} \times \{0, 1, 2, \ldots, 7\}$

| | |
|---|---|
| *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*] |

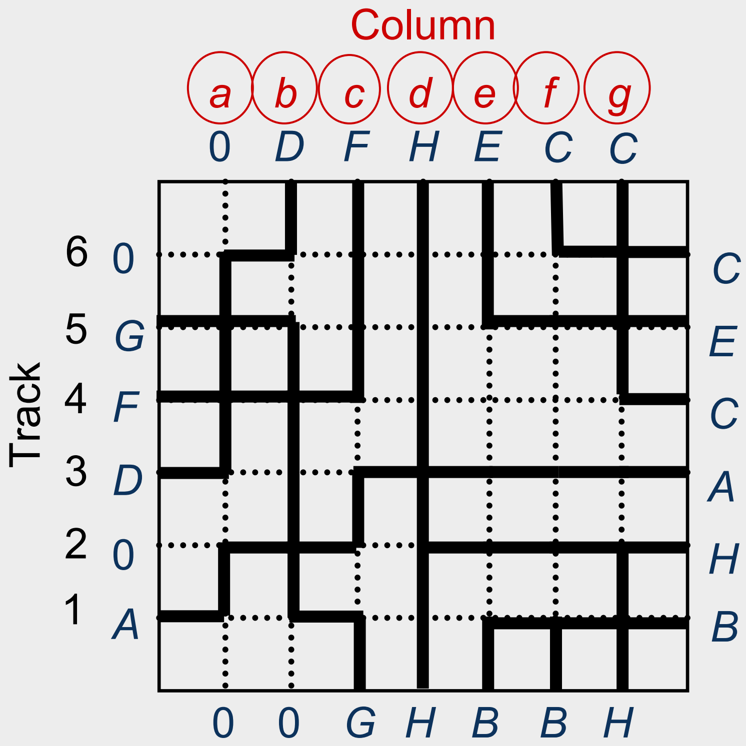


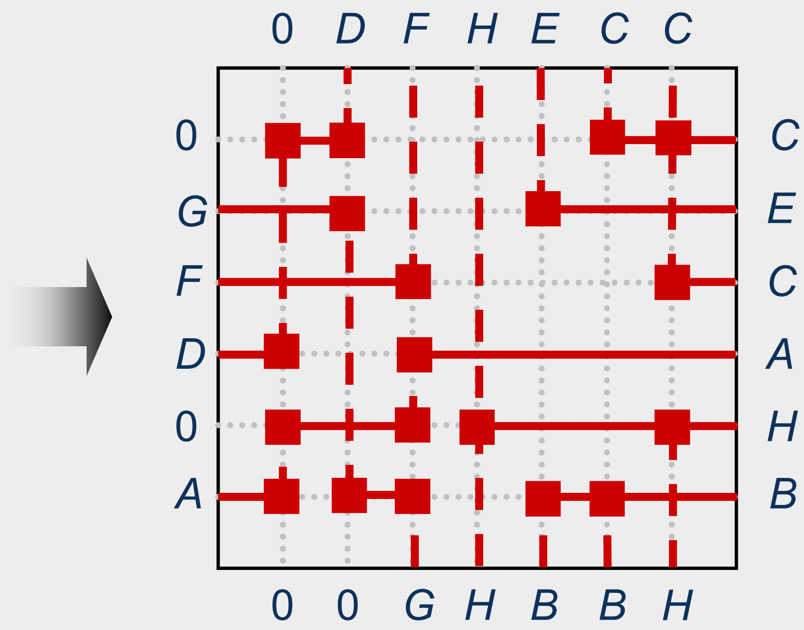



*TOP* = (1, 2, … , 7) = [0, *D*, *F*, *H*, *E*, *C*, *C*]
*BOT* = (1, 2, … , 7) = [0, 0, *G*, *H*, *B*, *B*, *H*]
*LEFT* = (1, 2, … , 6) = [*A*, 0, *D*, *F*, *G*, 0]
*RIGHT* = (1, 2, … , 6) = [*B*, *H*, *A*, *C*, *E*, *C*]



*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*]

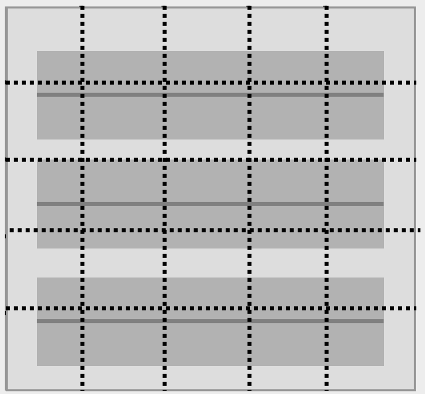


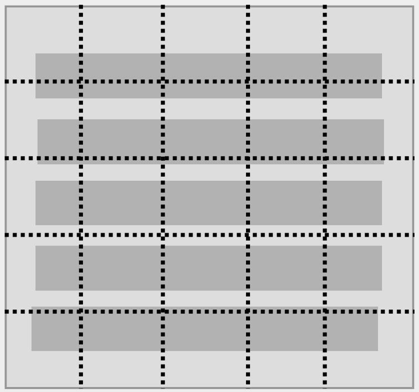
# 6.5 Over-the-Cell Routing Algorithms

## 6.5 Over-the-Cell Routing Algorithms

- Standard cells are placed back-to-back or without routing channels
- Metal layers are usually represented by a coarse routing grid made up of global routing cells (gcells)



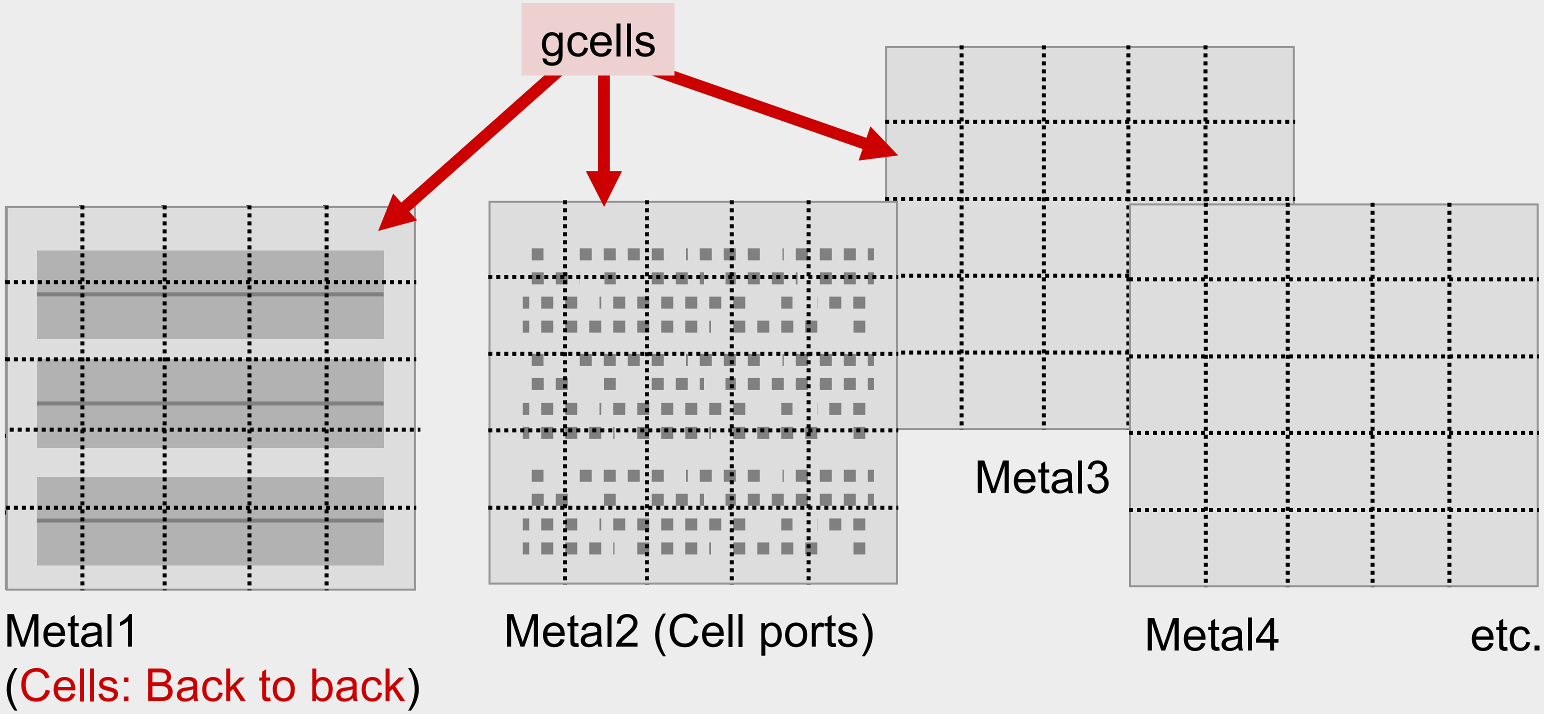
Back to back



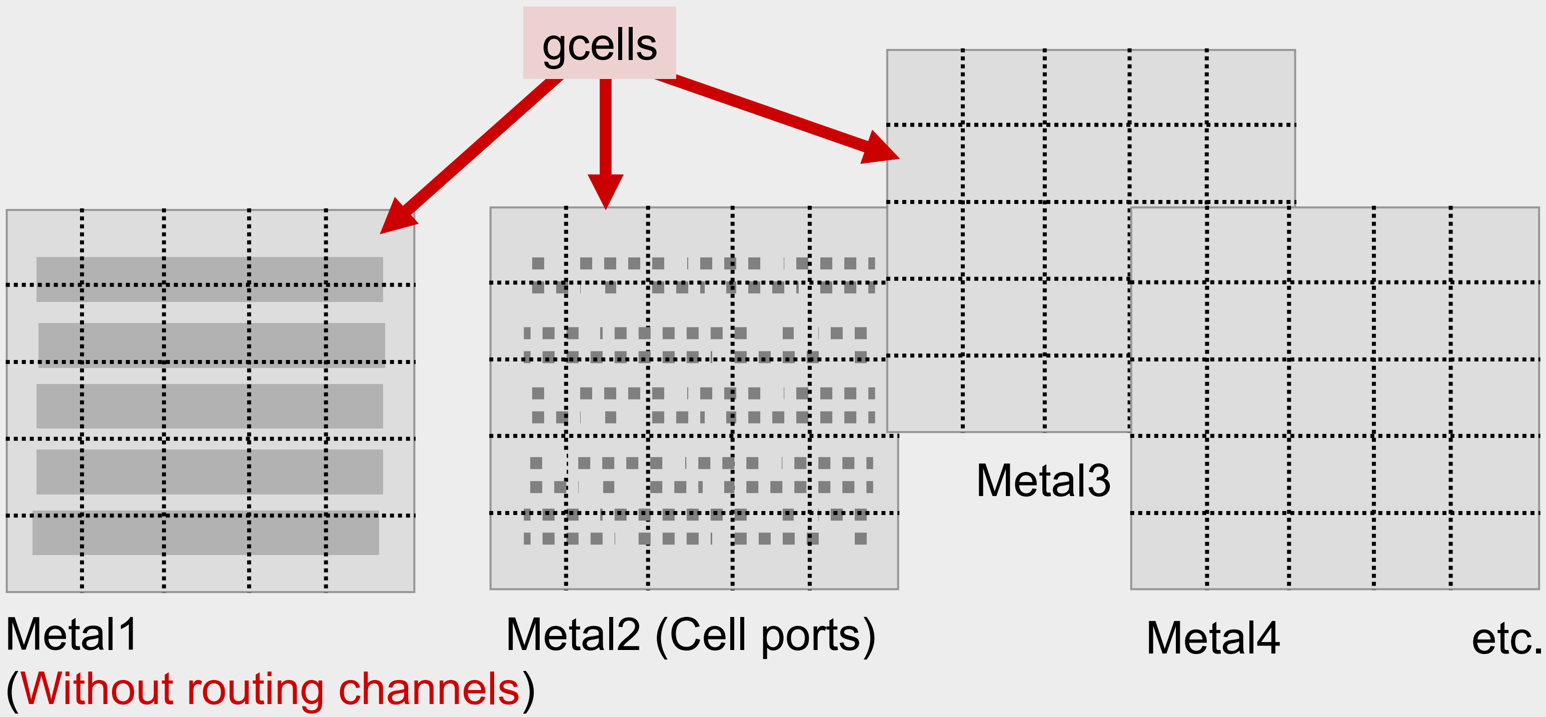
Without routing channels

# 6.5 Over-the-Cell Routing Algorithms

- Standard cells are placed back-to-back or without routing channels
- Metal layers are usually represented by a coarse routing grid made up of global routing cells (gcells)



- Standard cells are placed back-to-back or without routing channels
- Metal layers are usually represented by a coarse routing grid made up of global routing cells (gcells)

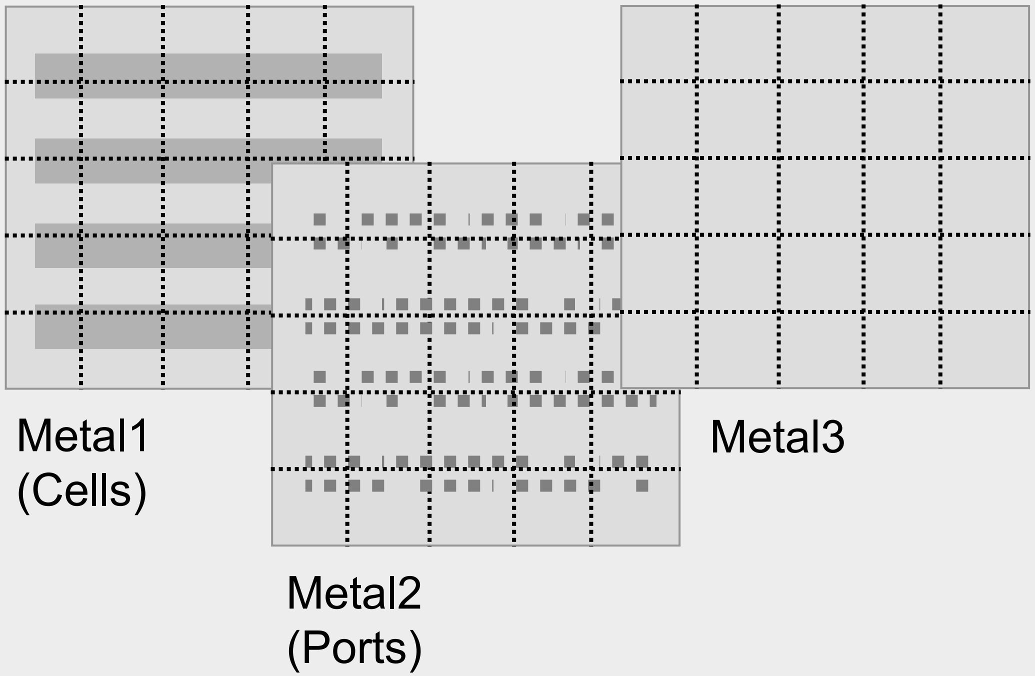


## 6.5 Over-the-Cell Routing Algorithms

- Standard cells are placed back-to-back or without routing channels
- Metal layers are usually represented by a coarse routing grid made up of global routing cells (gcells)
- Layers that are not obstructed by standard cells are typically used for over-the-cell (OTC) routing
- Nets are globally routed using gcells and then detail-routed

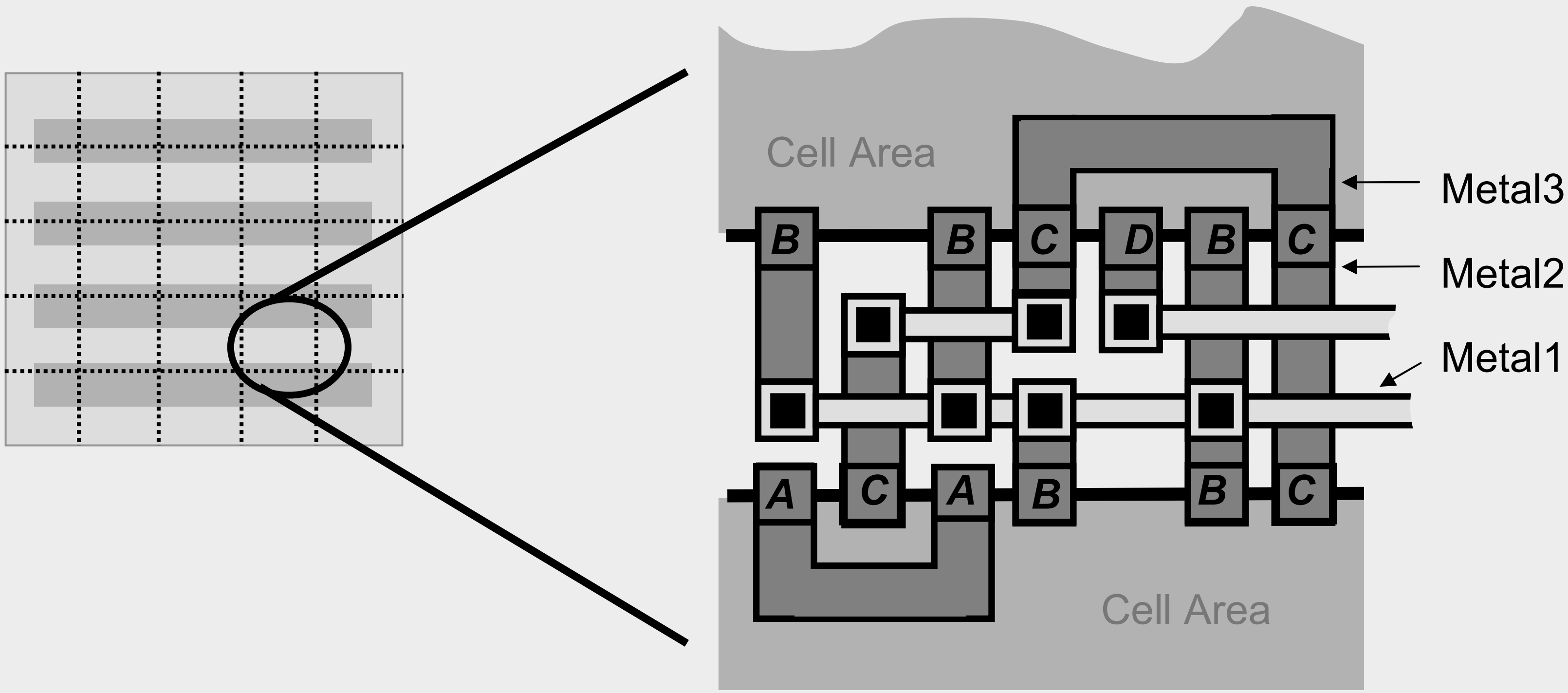
**Three-layer approach**

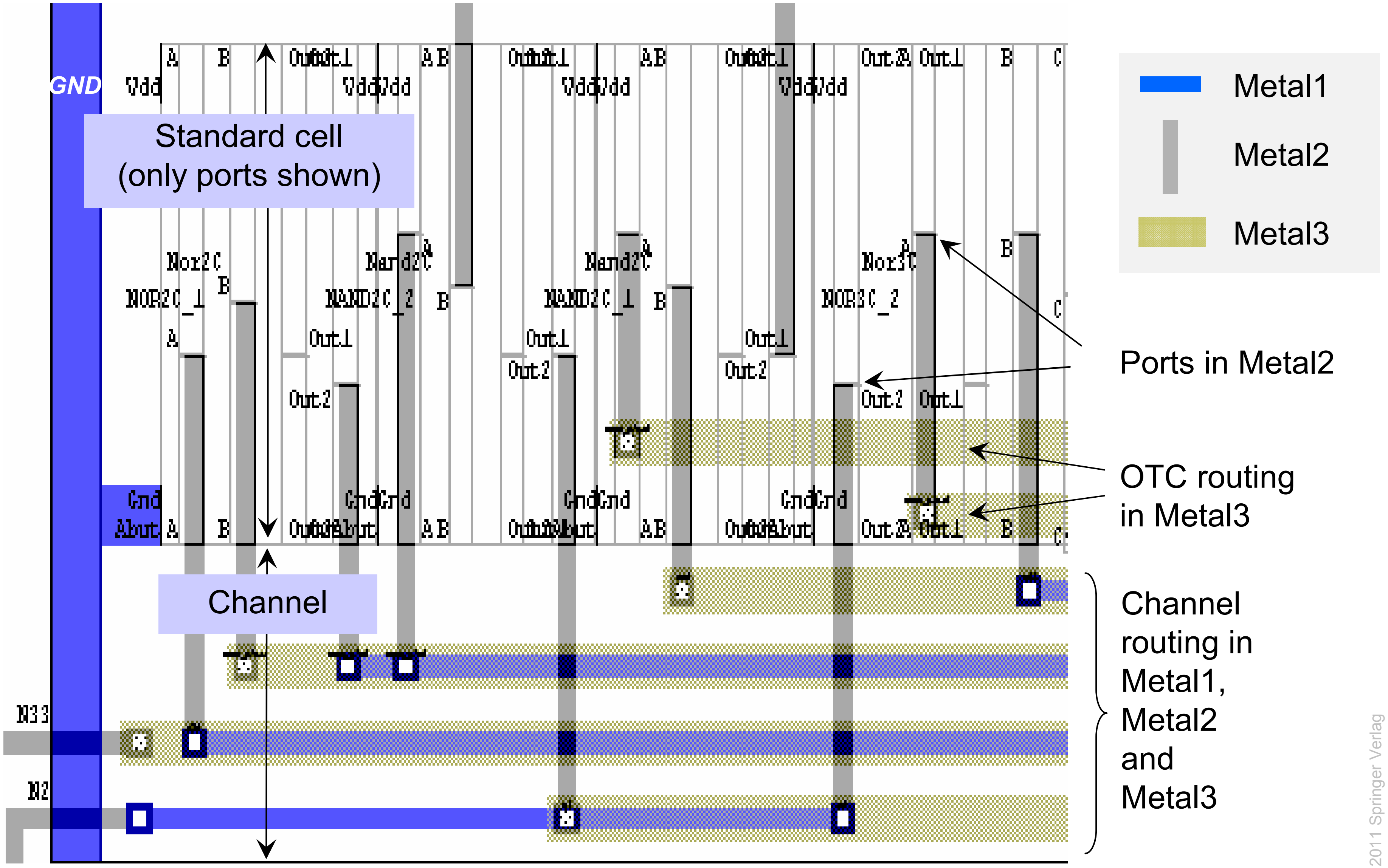
- Metal3 is used for over-the-cell (OTC) routing



**Three-layer approach**

- Metal3 is used for over-the-cell (OTC) routing





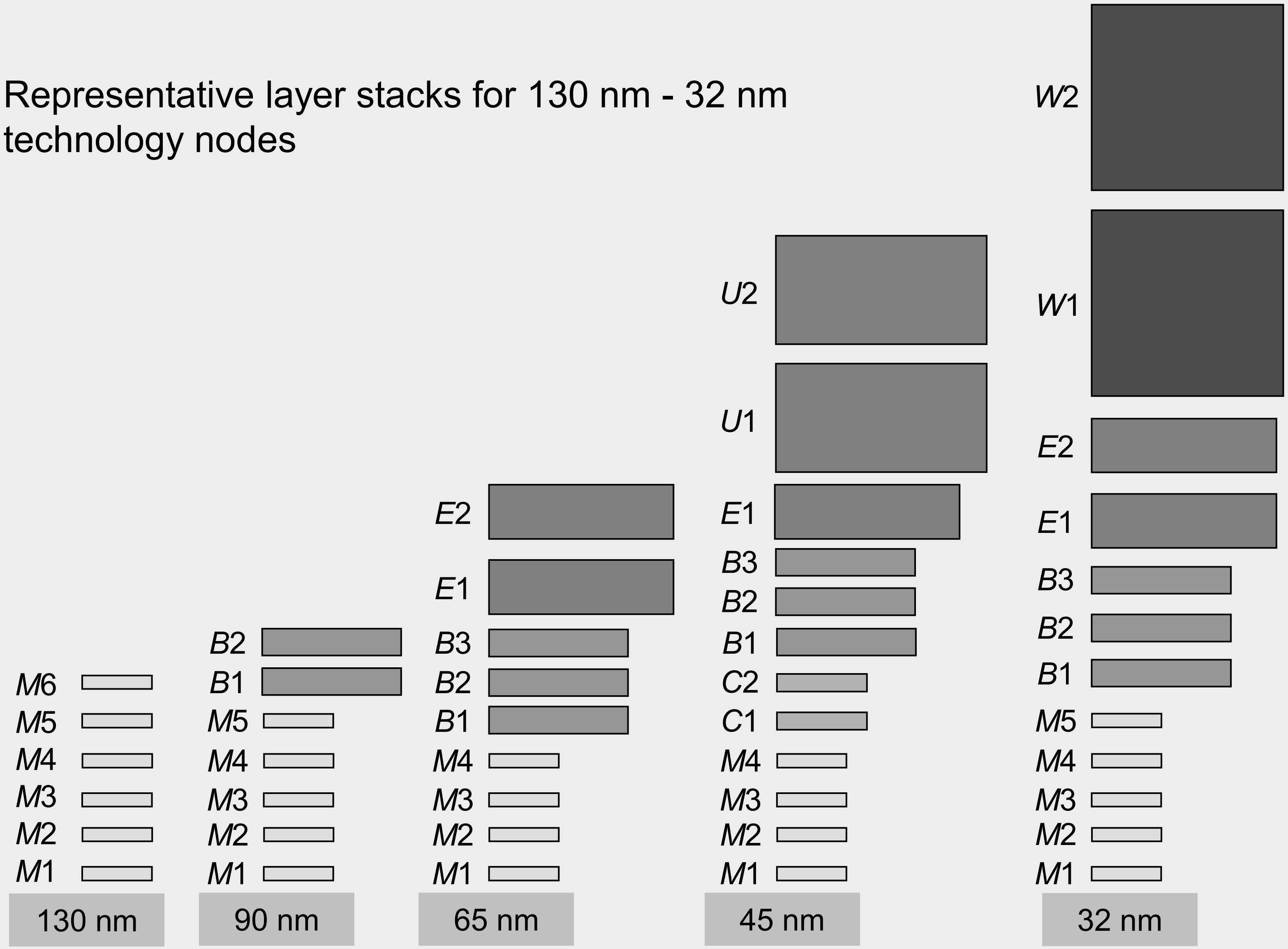
GND
Standard cell
(only ports shown)
Channel
Metal1
Metal2
Metal3
Ports in Metal2
OTC routing
in Metal3
Channel
routing in
Metal1,
Metal2
and
Metal3

# 6.6 Modern Challenges in Detailed Routing

## 6.6 Modern Challenges in Detailed Routing

- Manufacturers today use different configurations of metal layers and widths to accommodate high-performance designs
- Detailed routing is becoming more challenging, for example:
  - Vias connecting wires of different widths inevitably block additional routing resources on the layer with the smaller wire pitch
  - Advanced lithography techniques used in manufacturing require stricter enforcement of preferred routing direction on each layer

Representative layer stacks for 130 nm - 32 nm technology nodes

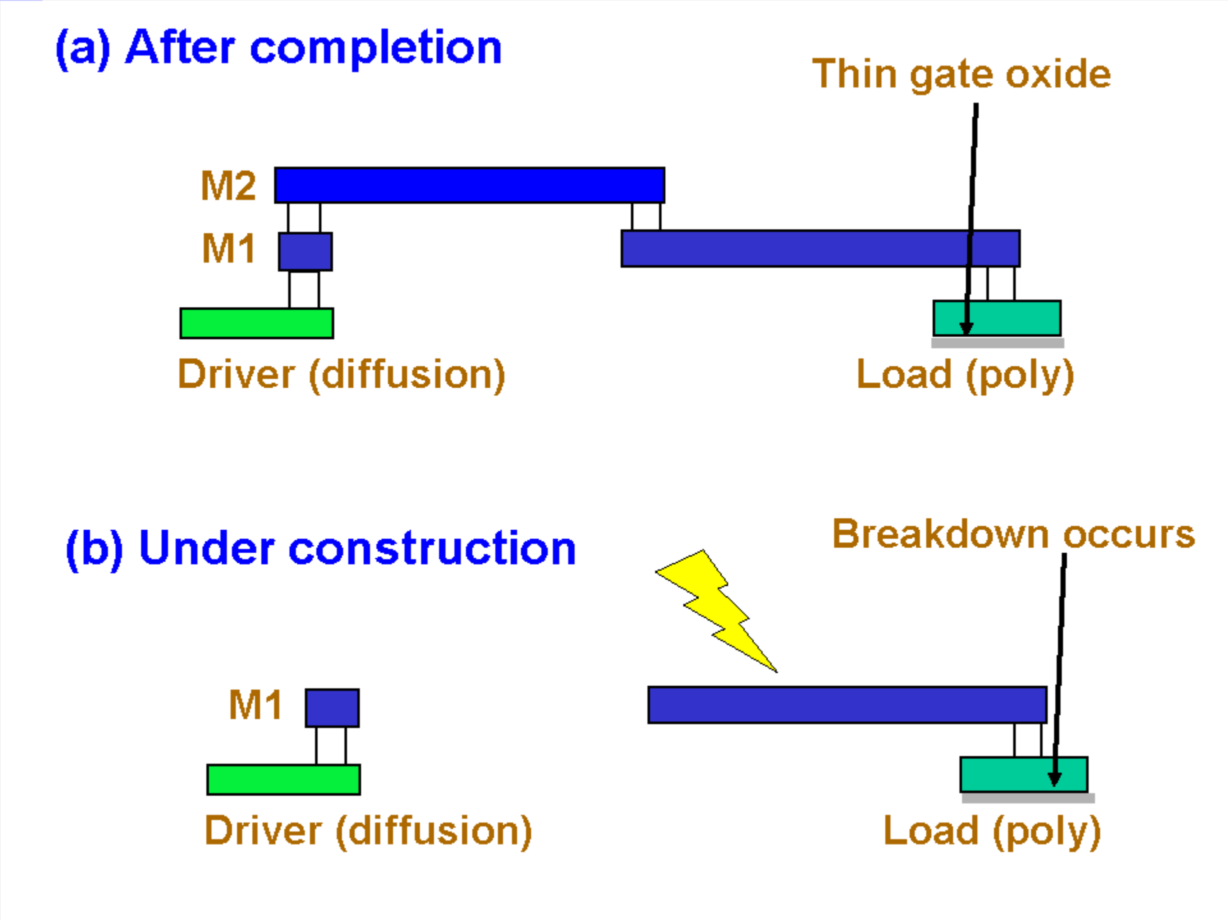


## 6.6 Modern Challenges in Detailed Routing

- Semiconductor manufacturing yield is a key concern in detailed routing
  - Redundant vias and wiring segments as backups (via doubling and non-tree routing)
  - Manufacturability constraints (design rules) become more restrictive
  - Forbidden pitch rules prohibit routing wires at certain distances apart, but allows smaller or greater spacings

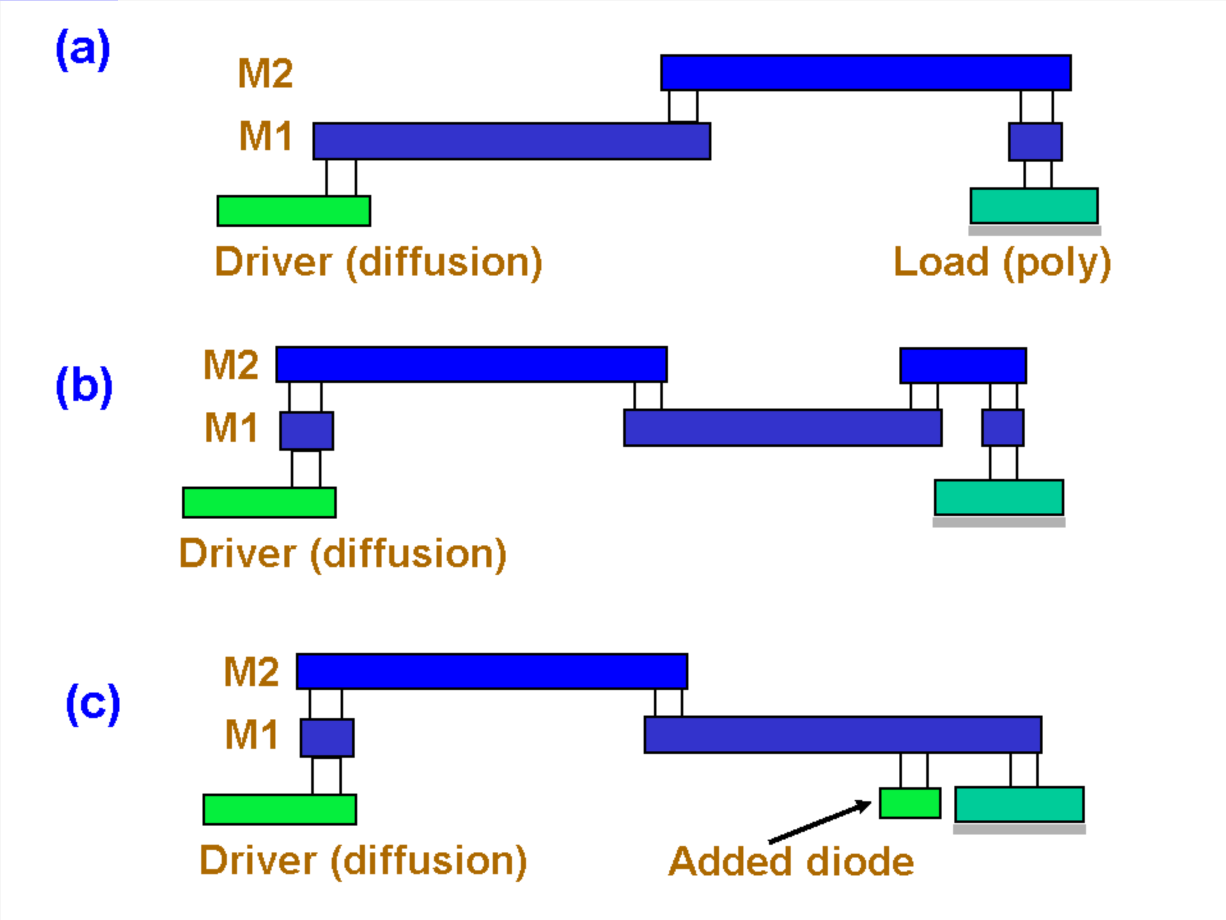
- Detailed routers must account for manufacturing rules and the impact of manufacturing faults
  - Via defects: via doubling during or after detailed routing
  - Interconnect defects: add redundant wires to already routed nets
  - Antenna-induced defects: detailed routers limit the ratio of metal to gate area on each metal layer

## Antenna Effect




Source: `http://en.wikipedia.org/wiki/Antenna_effect`

## Antenna Effect Fix




Source: `http://en.wikipedia.org/wiki/Antenna_effect`

## Summary of Chapter 6 – Context

- Detailed routing is invoked after global routing
- Usually takes about as much time as global routing
  - For heavily congested designs can take much longer
- Generates specific track assignments for each connection
  - Tries to follow "suggestions" made by global routing, but may alter them if necessary
  - A small number of failed global routed (disconnected, overcapacity) can be tolerated
- More affected by technology & manufacturing constraints than global routing
  - Must satisfy design rules

## Summary of Chapter 6 – Routing Regions

- Breaks down the layout area into regions
  - Channels have net terminals (pins) on two sides
  - Switch-boxes have terminals on four sides
  - Channels are joined at switchboxes
- When the number of metal layers is >3, use over-the-cell (OTC) routing
  - Divide the layout region into a grid of global routing cells (gcells)
  - OTC routing makes the locations of cells, obstacles and pins less important
  - Channel and switchbox routing can be used during OTC routing when upper metal layers are blocked (by wide buses, other wires, etc.)
- The capacity of a region is limited by the number of tracks it contains
  - Channels, switchboxes, gcells

## Summary of Chapter 6 – Algorithms and Data Structures

- Horizontal and vertical constraint graphs capture constraints that must be satisfied by valid routes
- Simplest algorithms for detailed routing are greedy
  - Every step satisfies immediate constraints with minimal routing cost
  - Use as few bends as possible (doglegs are used when additional bends are needed)
  - Very fast, do a surprisingly good job in many cases
  - Insufficient for congested designs
- Switchbox routing algorithms are usually derived from channel routing algorithms
- Strategy 1: Do not create congested designs and rely on greedy algorithms
- Strategy 2: Accommodate congested designs and develop stronger algorithms

## Summary of Chapter 6 – Modern Challenges

- Variable-pitch wire stacks
  - Not addressed in the literature until 2008
- Satisfying more complex design rules
  - Min spacing between wires and devices
  - Forbidden pitch rules
  - Antenna rules
- Soft rules
  - Do not need to be satisfied
  - Can improve yield by decreasing the probability of defects
- Redundant vias
  - In case some vias are poorly manufactured
- Redundant wires
  - In case some wires get disconnected