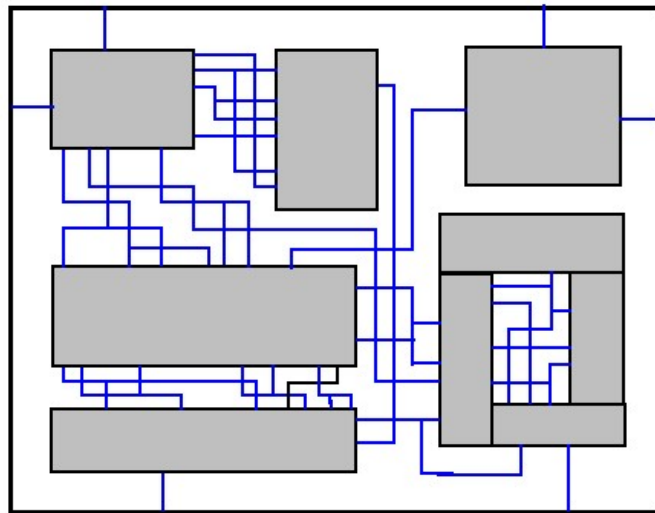


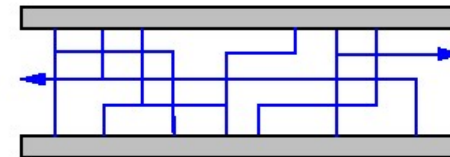
# Unit 7: Detailed Routing

---

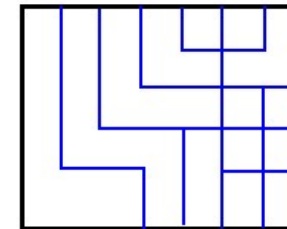
- Course contents
  - Channel routing
  - Clock routing
  - Power/ground routing



Detailed routing



channel routing



switchbox routing

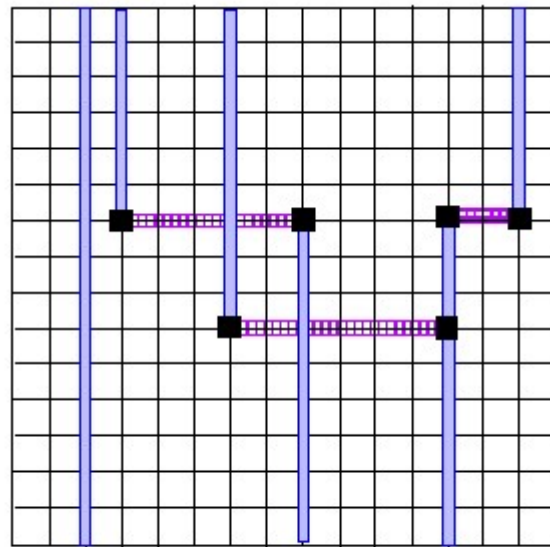
# Routing Considerations

---

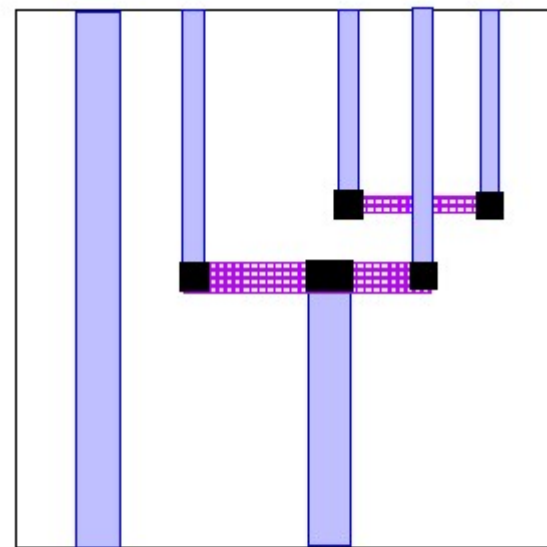
- Number of terminals (two-terminal vs. multi-terminal nets)
- Net widths (power and ground vs. signal nets)
- Via restrictions (stacked vs. conventional vias)
- Boundary types (regular vs. irregular)
- Number of layers (two vs. three, more layers?)
- Net types (critical vs. non-critical nets)

# 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.

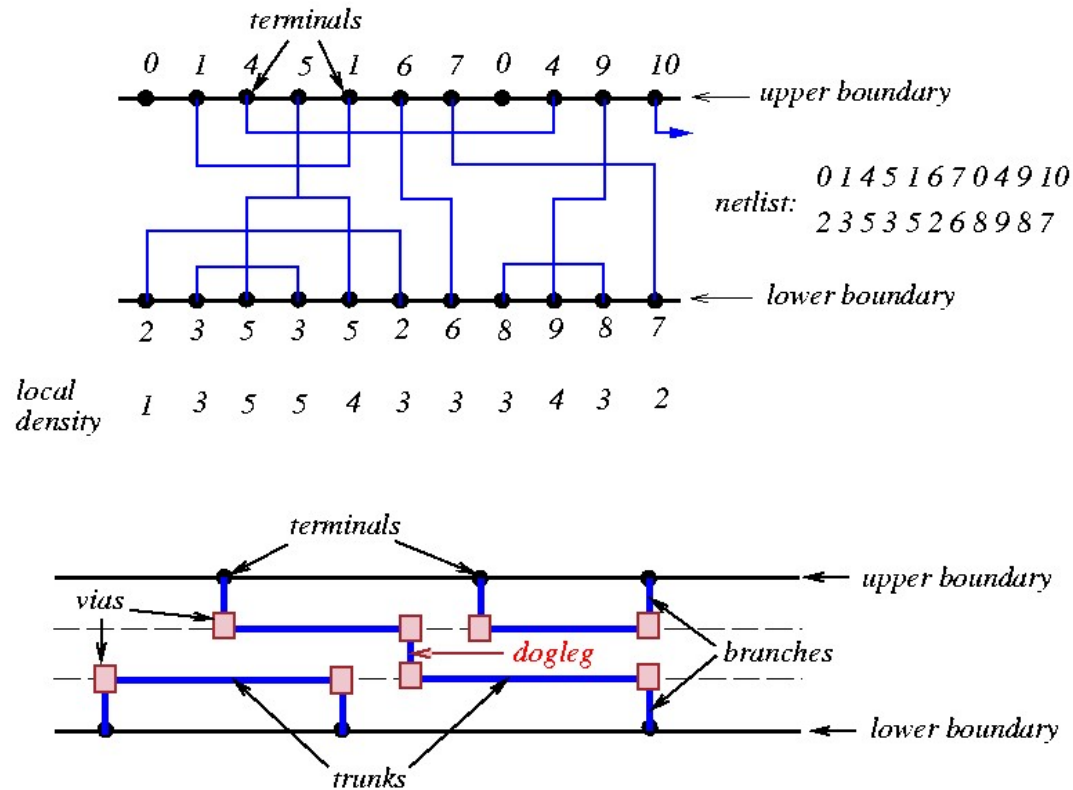


grid-based



gridless

# Terminology for Channel Routing



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

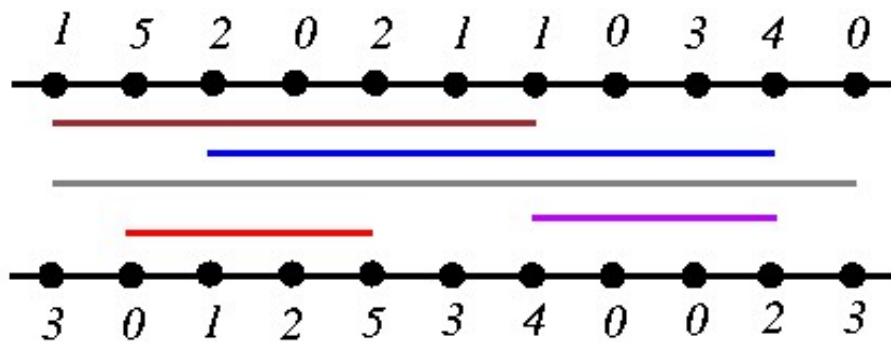
# Channel Routing Problem

---

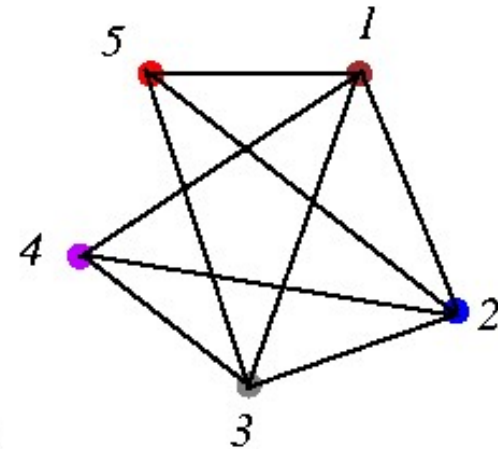
- **Assignments of horizontal segments of nets to tracks.**
- **Assignments of vertical segments to connect.**
  - horizontal segments of the same net in different tracks, and
  - the terminals of the net to horizontal segments of the net.
- **Horizontal and vertical constraints must not be violated.**
  - 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
  - $V = \{v_i \mid v_i \text{ represents a net } n_i\}$
  - $E = \{(v_i, v_j) \mid \text{a horizontal constraint exists between } n_i \text{ and } n_j\}$ .
- For graph  $G$ : vertices  $\Leftrightarrow$  nets; edge  $(i, j) \Leftrightarrow$  net  $i$  overlaps net  $j$ .

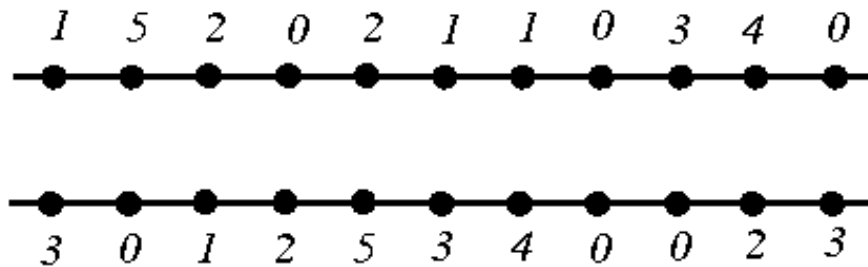


A routing problem and its HCG.

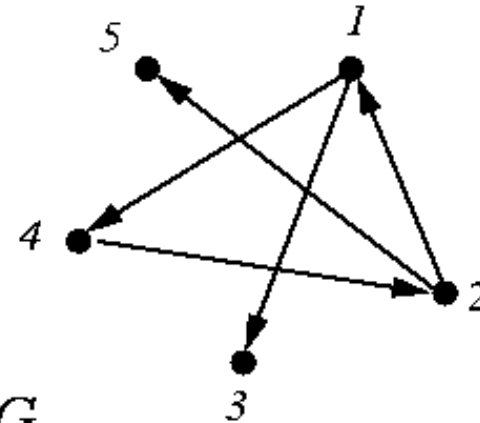


# Vertical Constraint Graph (VCG)

- VCG  $G = (V, E)$  is **directed** graph where
  - $V = \{v_i \mid v_i \text{ represents a net } n_i\}$
  - $E = \{(v_i, v_j) \mid \text{a vertical constraint exists between } n_i \text{ and } n_j\}$ .
- 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)  $I_1, \dots, I_n$ ;

$I_j = [s_j, e_j]$ : interval  $j$  with left-end x-coordinate  $s_j$  and right-end  $e_j$ ;

$track[j]$ : track to which net  $j$  is assigned.

1 **begin**

2  $U \leftarrow \{I_1, I_2, \dots, I_n\}$ ;

3  $t \leftarrow 0$ ;

4 **while** ( $U \neq \emptyset$ ) **do**

5      $t \leftarrow t + 1$ ;

6      $watermark \leftarrow 0$ ;

7     **while** (there is an  $I_j \in U$  s.t.  $s_j > watermark$ ) **do**

8         Pick the interval  $I_j \in U$  with  $s_j > watermark$ ,  
           nearest  $watermark$ ;

9          $track[j] \leftarrow t$ ;

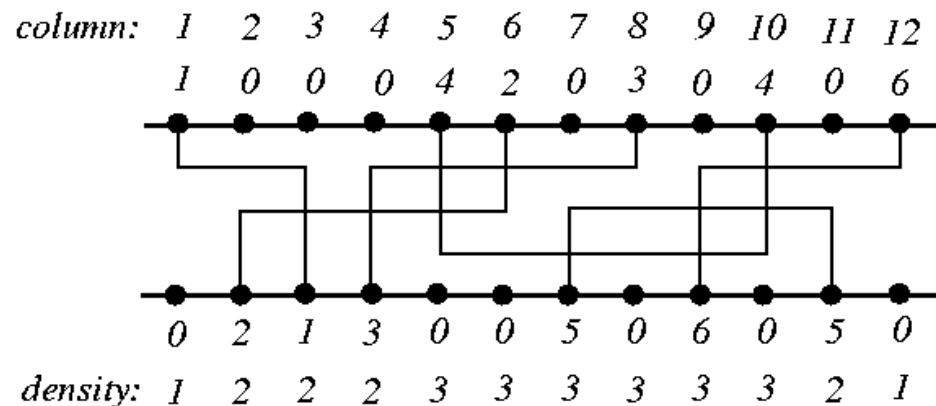
10         $watermark \leftarrow e_j$ ;

11         $U \leftarrow U - \{I_j\}$ ;

12 **end**

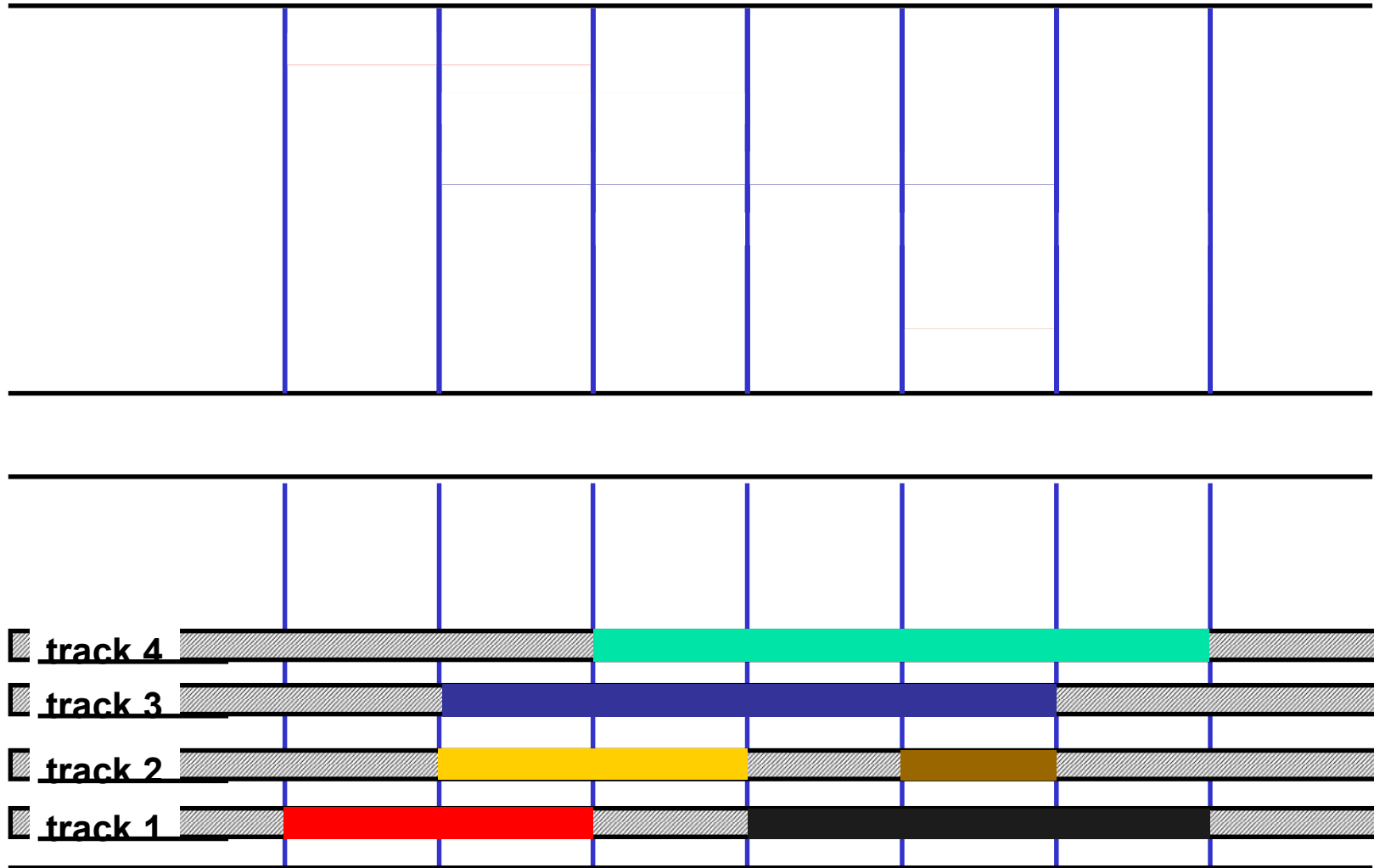
# Basic Left-Edge Example

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



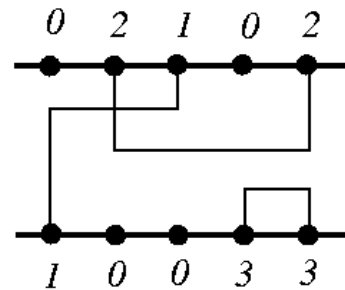
# Basic Left-Edge Example

---

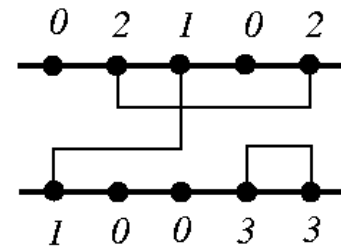


# Basic Left-Edge Algorithm

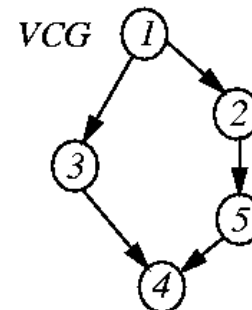
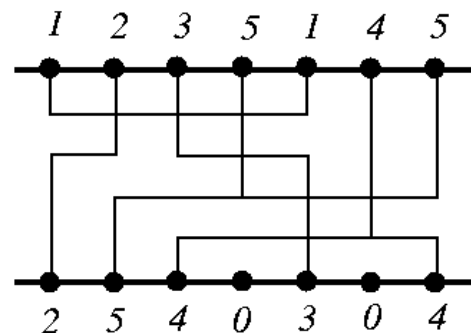
- If there is no vertical constraint, the basic left-edge algorithm is optimal.
- If there is any vertical constraint, the algorithm no longer guarantees optimal solution.



result from basic  
left-edge algorithm  
3 tracks



optimal routing: 2 tracks



# Constrained Left-Edge Algorithm

**Algorithm: Constrained\_Left-Edge( $U, track[j]$ )**

$U$ : set of unassigned intervals (nets)  $I_1, \dots, I_n$ ;

$I_j = [s_j, e_j]$ : interval  $j$  with left-end x-coordinate  $s_j$  and right-end  $e_j$ ;

$track[j]$ : track to which net  $j$  is assigned.

1 **begin**

2  $U \leftarrow \{I_1, I_2, \dots, I_n\}$ ;

3  $t \leftarrow 0$ ;

4 **while** ( $U \neq \emptyset$ ) **do**

5      $t \leftarrow t + 1$ ;

6      $watermark \leftarrow 0$ ;

7     **while** (there is an **unconstrained**  $I_j \in U$  s.t.  $s_j >$   
    $watermark$ ) **do**

8         Pick the interval  $I_j \in U$  that is unconstrained,  
       with  $s_j > watermark$ , nearest  $watermark$ ;

9          $track[j] \leftarrow t$ ;

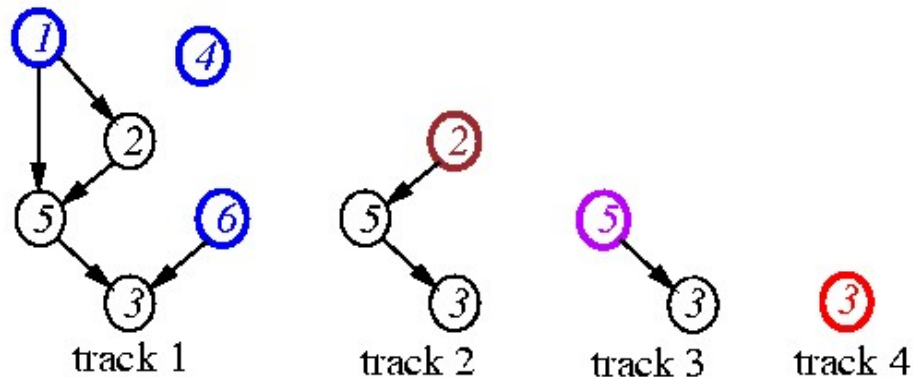
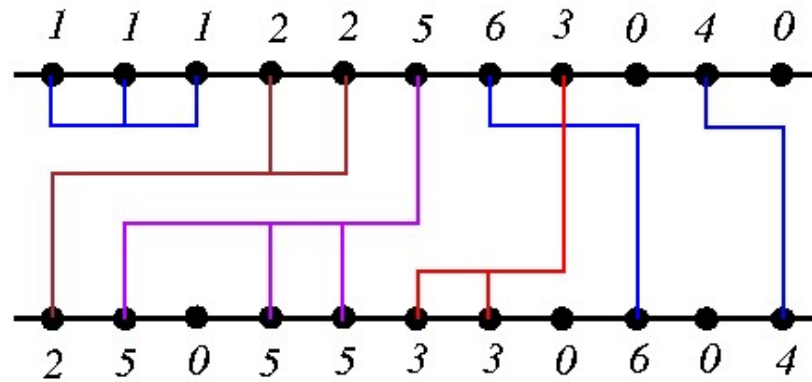
10         $watermark \leftarrow e_j$ ;

11         $U \leftarrow U - \{I_j\}$ ;

12 **end**

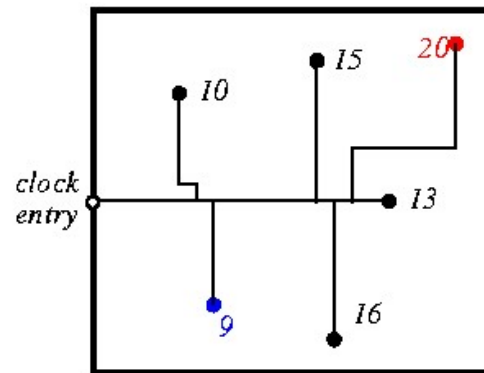
# Constrained Left-Edge Example

- $I_1 = [1, 3]$ ,  $I_2 = [1, 5]$ ,  $I_3 = [6, 8]$ ,  $I_4 = [10, 11]$ ,  $I_5 = [2, 6]$ ,  $I_6 = [7, 9]$ .
- 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  $I_3$ .

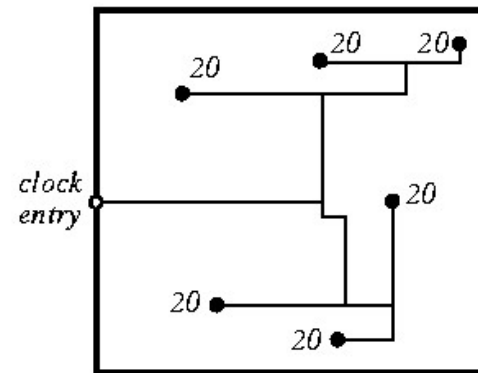


# The Clock Routing Problem (CRP)

- Digital systems
  - **Synchronous systems:** Highly precise clock achieves communication and timing.
  - **Asynchronous systems:** Handshake protocol achieves the timing requirements of the system.
- **Clock skew** is defined as the difference in the minimum and the maximum arrival time of the clock.



$$\text{clock skew} = 20 - 9 = 11$$



$$\text{clock skew} = 0$$

- **CRP:** Routing clock nets such that
  1. clock signals arrive simultaneously
  2. clock delay is minimized
    - Other issues: total wirelength, power consumption, etc.

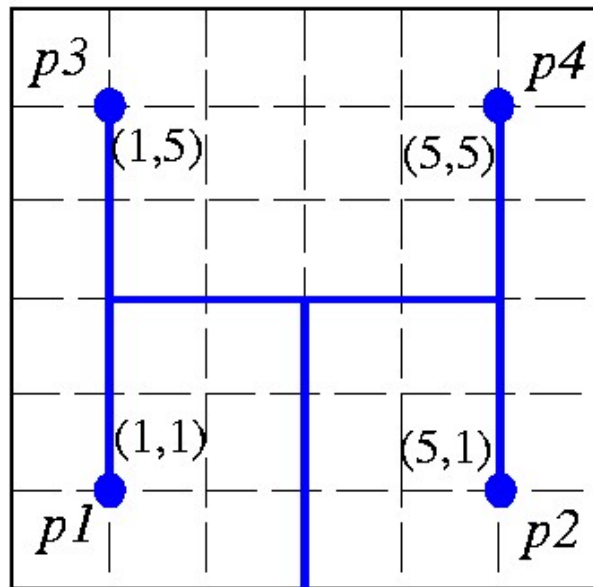
# Clock Routing Problem

---

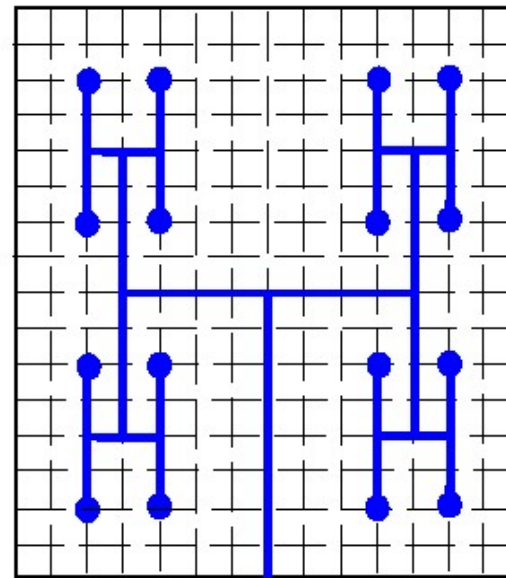
- Given the routing plane and a set of points  $P = \{p_1, p_2, \dots, p_n\}$  within the plane and clock entry point  $p_0$  on the boundary of the plane, the **Clock Routing Problem (CRP)** is to interconnect each  $p_i \in P$  such that **clock skew**,  $\max_{i, j \in P} |t(0, i) - t(0, j)|$ , and **clock latency**,  $\max_{i \in P} t(0, i)$ , are both minimized.
- Pathlength-based approaches
  1. *H-tree*: Dhar, Franklin, Wang, ICCD-84; Fisher & Kung, 1982.  
Geometric matching: Cong, Kahng, Robins, DAC-91.
- RC-delay based approaches:
  1. Exact zero skew: Tasy, ICCAD-91.
  2. Lagrangian relaxation: Chen, Chang, Wong, DAC-96.

# H-Tree Based Algorithm

- *H*-tree: Dhar, Franklin, Wang, “Reduction of clock delays in VLSI structure,” ICCD-84.



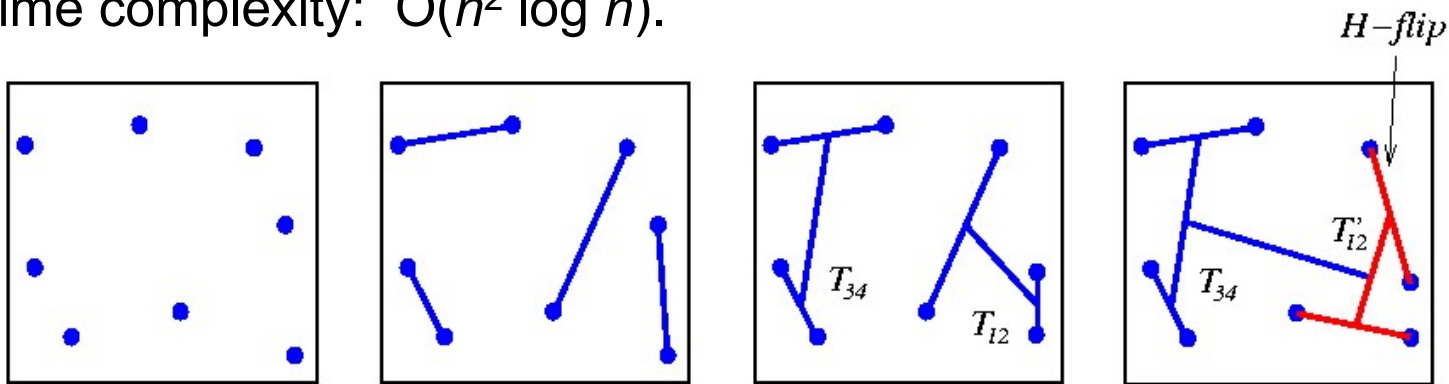
$p_0(3,0)$   
*H-tree over 4 points*



*H-tree over 16 points*

# The Geometric Matching Algorithm

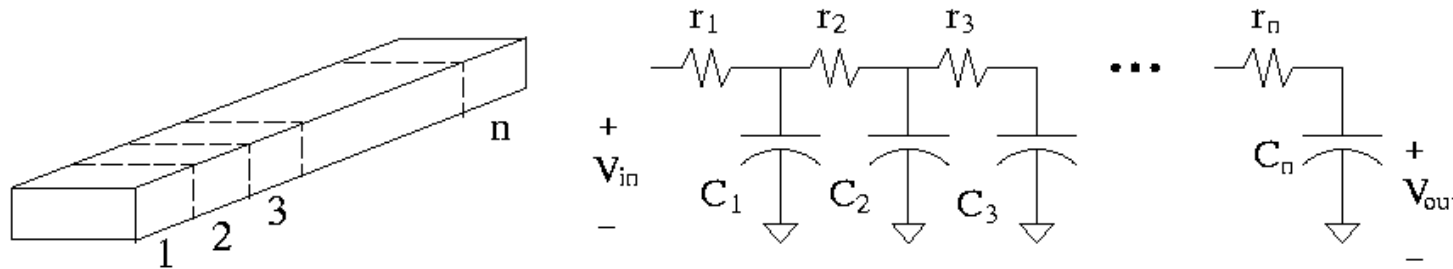
- Cong, Kahng, Robins, “Matching based models for high-performance clock routing,” IEEE TCAD, 1993.
- Clock pins are represented as  $n$  nodes in the clock tree ( $n = 2^k$ ).
- Each node is a tree itself with clock entry point being node itself.
- The minimum cost matching on  $n$  points yields  $n/2$  segments.
- The clock entry point in each subtree of two nodes is the point on the segment such that length of both sides is same.
- Above steps are repeated for each segment.
- Apply  $H$ -flipping to further reduce clock skew (and to handle edge intersection).
- Time complexity:  $O(n^2 \log n)$ .



# Elmore Delay: Nonlinear Delay Model

- Parasitic resistance and capacitance dominate delay in deep submicron wires.
- Resistor  $r_i$  must charge all downstream capacitors.
- **Elmore delay:** Delay can be approximated as sum of sections: resistance  $\times$  downstream capacitance.

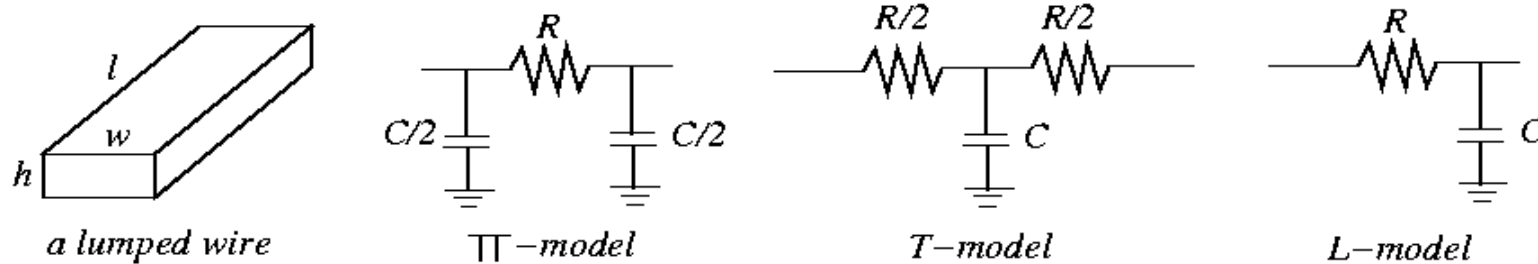
$$\delta = \sum_{i=1}^n \left( r_i \sum_{k=i}^n c_k \right) = \sum_{i=1}^n r_i (n - i + 1) c = \frac{n(n+1)}{2} r c.$$



- Delay grows as **square** of wire length.

# Wire Models

- Lumped circuit approximations for distributed RC lines:  **$\pi$ -model** (most popular), *T*-model, *L*-model.

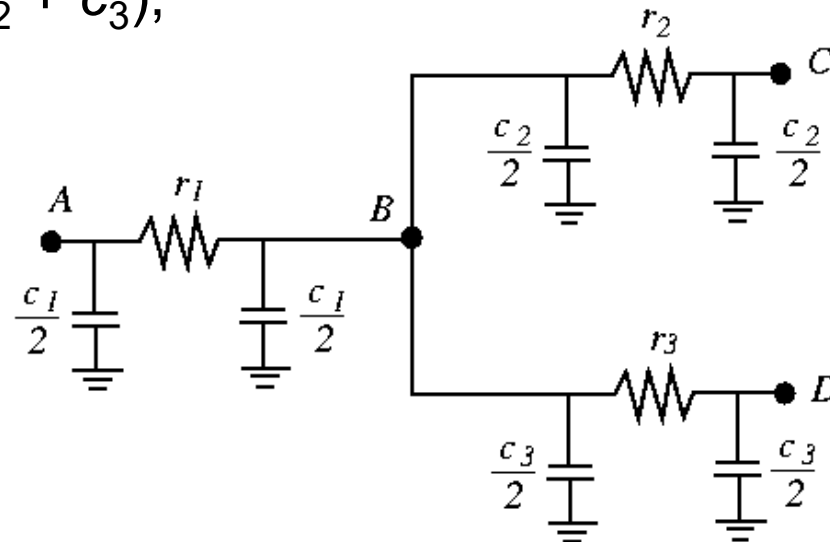


- $\pi$ -model: If no capacitive loads for  $C$  and  $D$ ,

$$A \text{ to } B: \delta_{AB} = r_1 (c_1/2 + c_2 + c_3);$$

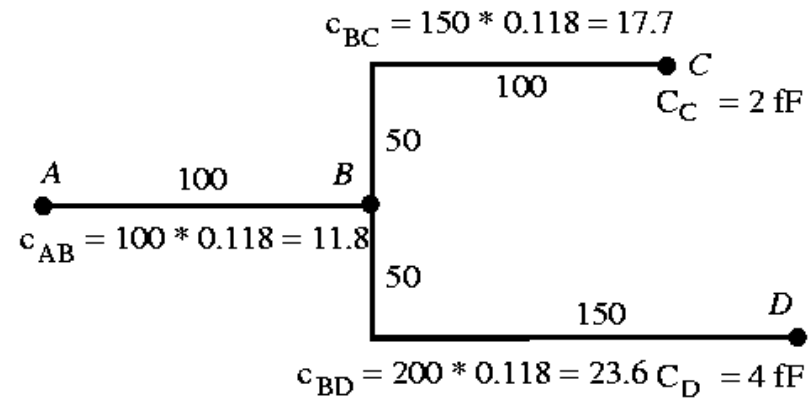
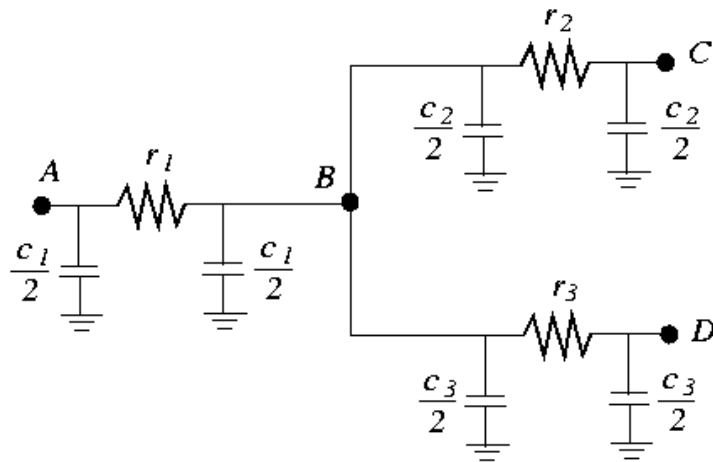
$$B \text{ to } C: \delta_{BC} = r_2 (c_2/2);$$

$$B \text{ to } D: \delta_{BD} = r_3 (c_3/2).$$



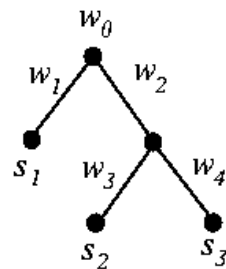
# Example: Elmore Delay Computation

- 0.18  $\mu m$  technology.: unit resistance  $\bar{r} = 0.075 \Omega / \mu m$ ; unit capacitance  $\bar{c} = 0.118 fF / \mu m$ .
  - Assume  $C_C = 2 fF$ ,  $C_D = 4 fF$ .
  - $\delta_{BC} = r_{BC} (c_{BC} / 2 + C_C) = 0.075 \times 150 (17.7 / 2 + 2) = 120 fs$
  - $\delta_{BD} = r_{BD} (c_{BD} / 2 + C_D) = 0.075 \times 200 (23.6 / 2 + 4) = 240 fs$
  - $\delta_{AB} = r_{AB} (c_{AB} / 2 + C_B) = 0.075 \times 100 (11.8 / 2 + 17.7 + 2 + 23.6 + 4) = 400 fs$
  - Critical path delay:  $\delta_{AB} + \delta_{BD} = 640 fs$ .

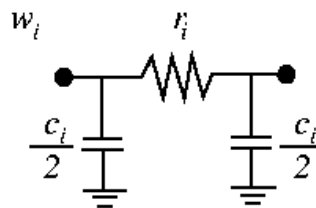


# Delay Calculation for a Clock Tree

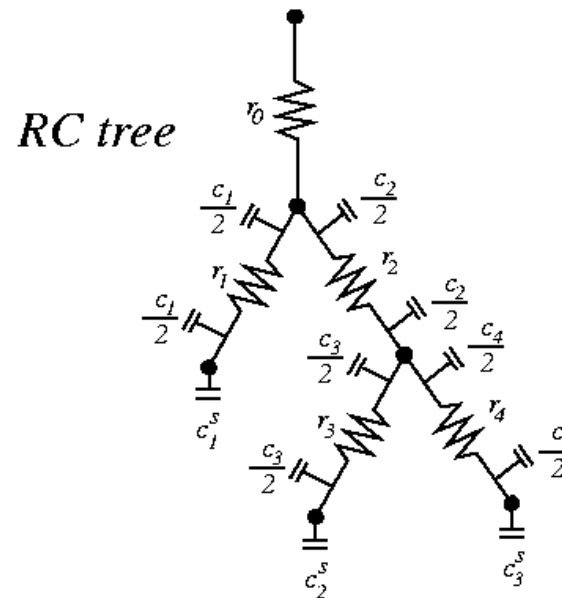
- Let  $T$  be an RC tree with points  $P = \{p_1, p_2, \dots, p_n\}$ ,  $c_i$  the capacitance of  $p_i$ ,  $r_i$  the resistance of the edge between  $p_i$  and its immediate predecessor.
- The subtree capacitance at node  $i$  is given as  $C_i = c_i + \sum_{j \in S_i} C_j$ , where  $S_i$  is the set of all the immediate successors of  $p_i$ .
- Let  $\delta(i, j)$  be the path between  $p_i$  and  $p_j$ , excluding  $p_i$  and including  $p_j$ .
- The delay between two nodes  $i$  and  $j$  is  $t_{ij} = \sum_{j \in \delta(i, j)} r_j C_j$ ,
- $t_{03} = r_0 (c_1 + c_2 + c_3 + c_4 + c_1^s + c_2^s + c_3^s) + r_2 (c_2/2 + c_3 + c_4 + c_2^s + c_3^s) + r_4 (c_4/2 + c_3^s)$ .



clock tree



delay model



RC tree

# Exact Zero Skew Algorithm

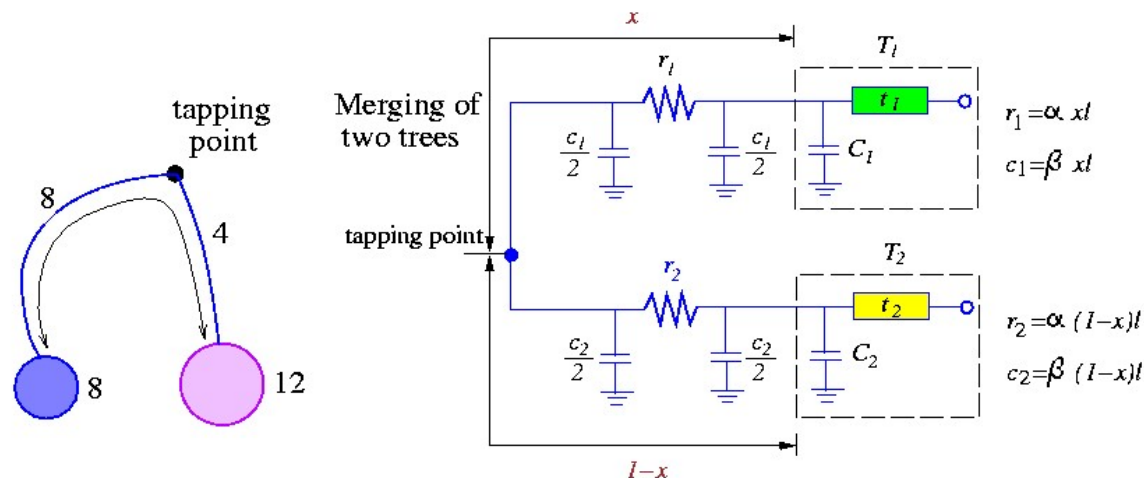
- Tasy, “Exact zero skew algorithm,” ICCAD-91.
- To ensure the delay from the **tapping point** to leaf nodes of subtrees  $T_1$  and  $T_2$  being equal, it requires that

$$r_1 (c_1/2 + C_1) + t_1 = r_2 (c_2/2 + C_2) + t_2.$$

- Solving the above equation, we have

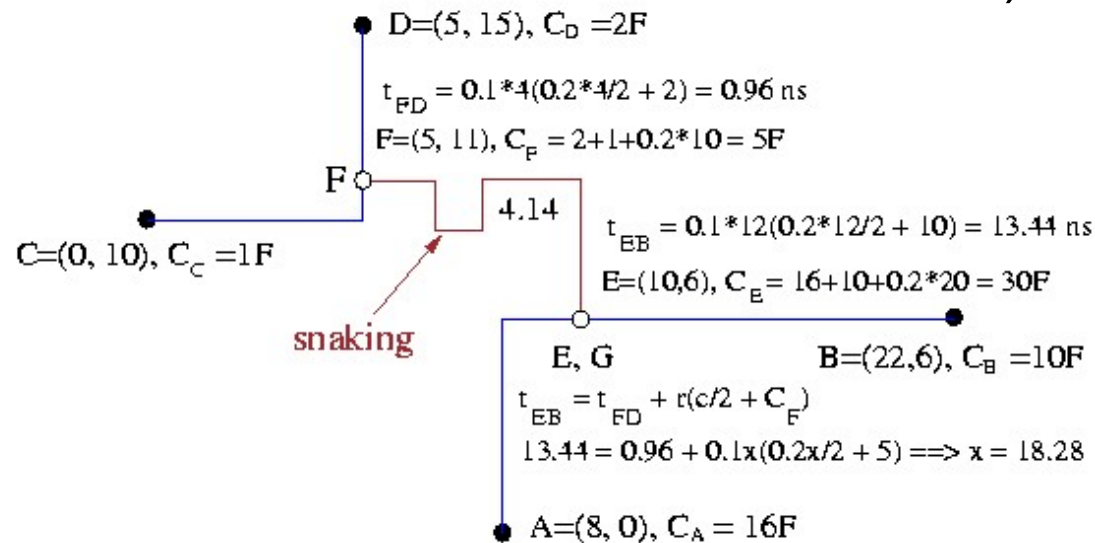
$$x = \frac{(t_2 - t_1) + \alpha l \left( C_2 + \frac{\beta l}{2} \right)}{\alpha l (\beta l + C_1 + C_2)},$$

where  $\alpha$  and  $\beta$  are the per unit values of resistance and capacitance,  $l$  the length of the interconnecting wire,  $r_1 = \alpha x l$ ,  $c_1 = \beta x l$ ,  $r_2 = \alpha (1 - x) l$ ,  $c_2 = \beta (1 - x) l$ .



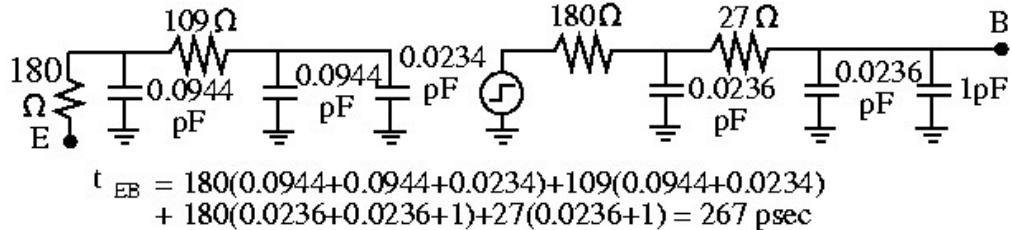
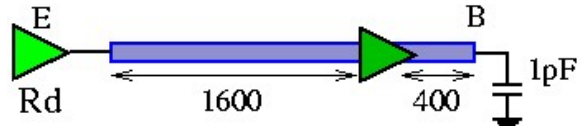
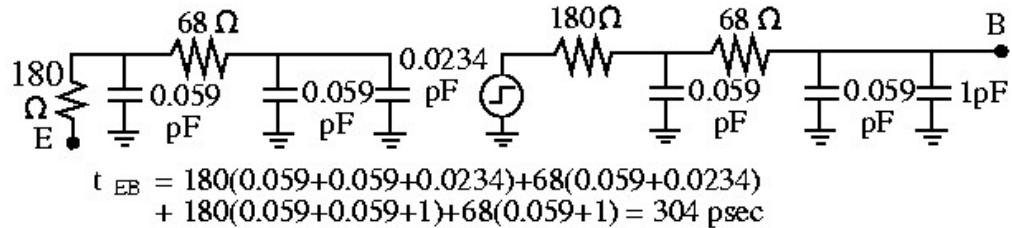
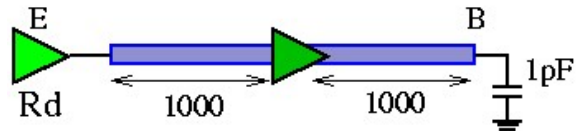
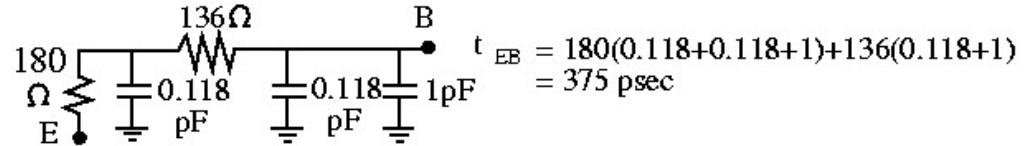
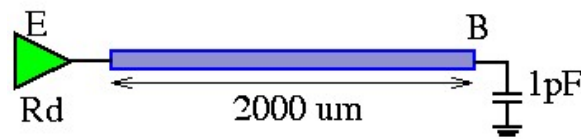
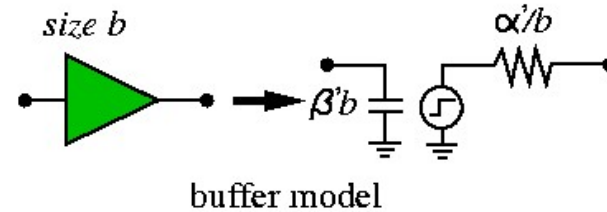
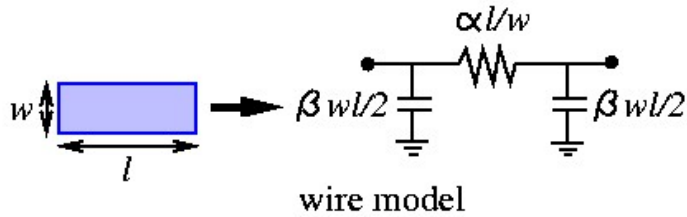
# Zero-Skew Computation

- **Balance delays:**  $r_1(c_1/2 + C_1) + t_1 = r_2(c_2/2 + C_2) + t_2$ .
- **Compute tapping points:**  $x = \frac{(t_2 - t_1) + \alpha \left( C_2 + \frac{\beta l}{2} \right)}{\alpha(\beta l + C_1 + C_2)}$ ,  $\alpha$  ( $\beta$ ): per unit values of resistance (capacitance);  $l$ : length of the wire;  
 $r_1 = \alpha x l$ ,  $c_1 = \beta x l$ ;  $r_2 = \alpha(1 - x) l$ ,  $c_2 = \beta(1 - x) l$ .
- If  $x \notin [0, 1]$ , we need **snaking** to find the tapping point.
- Exp:  $\alpha = 0.1 \Omega / \text{unit}$ ,  $\beta = 0.2 F / \text{unit}$ . (Find tapping points  $E$  for  $A$  and  $B$ ,  $F$  for  $C$  and  $D$ , and  $G$  for  $E$  and  $F$ .)

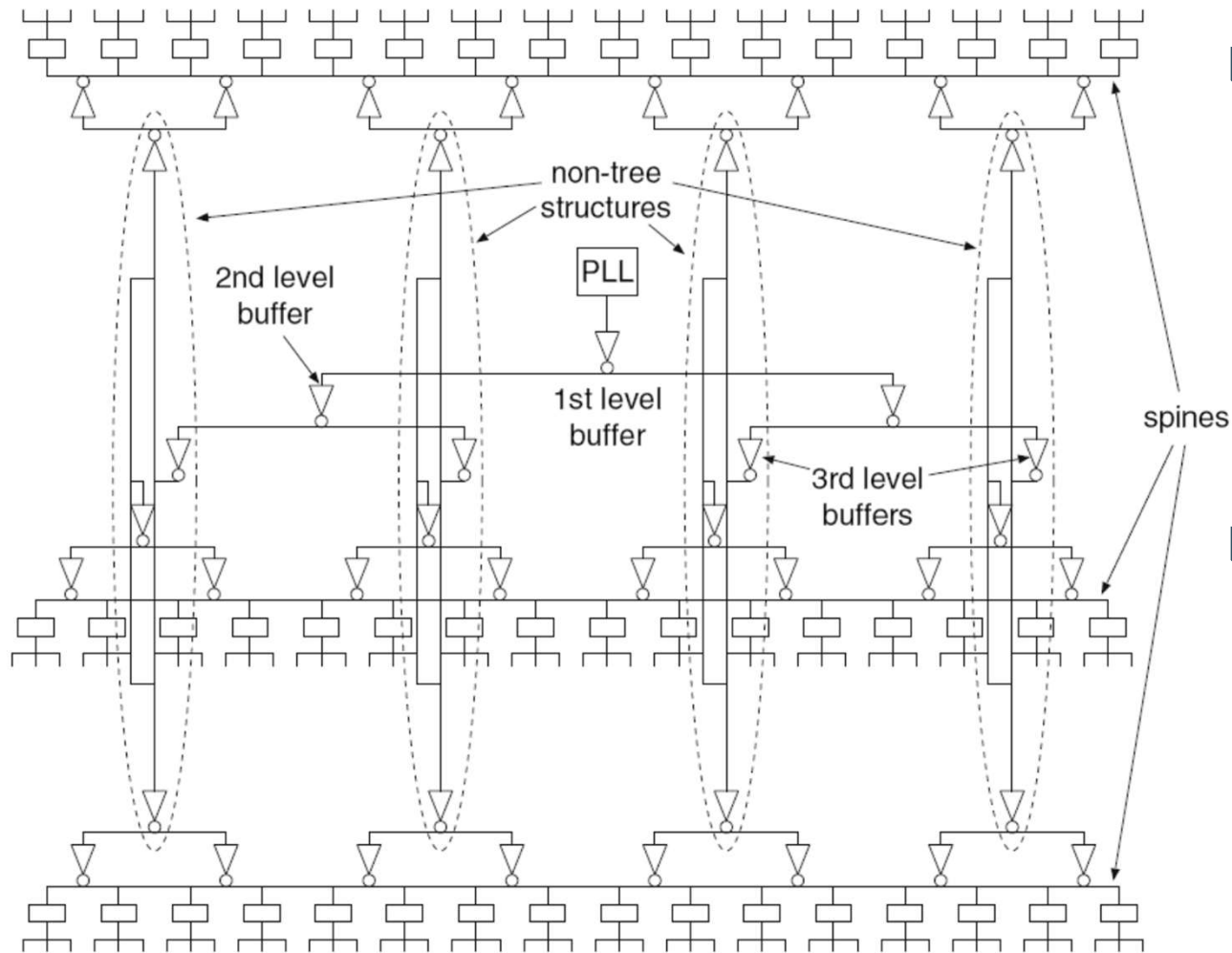


# Delay Computation for Buffered Wires

- Wire:  $\alpha = 0.068 \Omega / \mu\text{m}$ ,  $\beta = 0.118 \text{ fF} / \mu\text{m}^2$ ; buffer:  $\alpha' = 180 \Omega / \text{unit size}$ ,  $\beta = 23.4 \text{ fF} / \text{unit size}$ ; driver resistance  $R_d = 180 \Omega$ ; unit-sized wire, buffer.



# Modern Clock Design: Clock spines

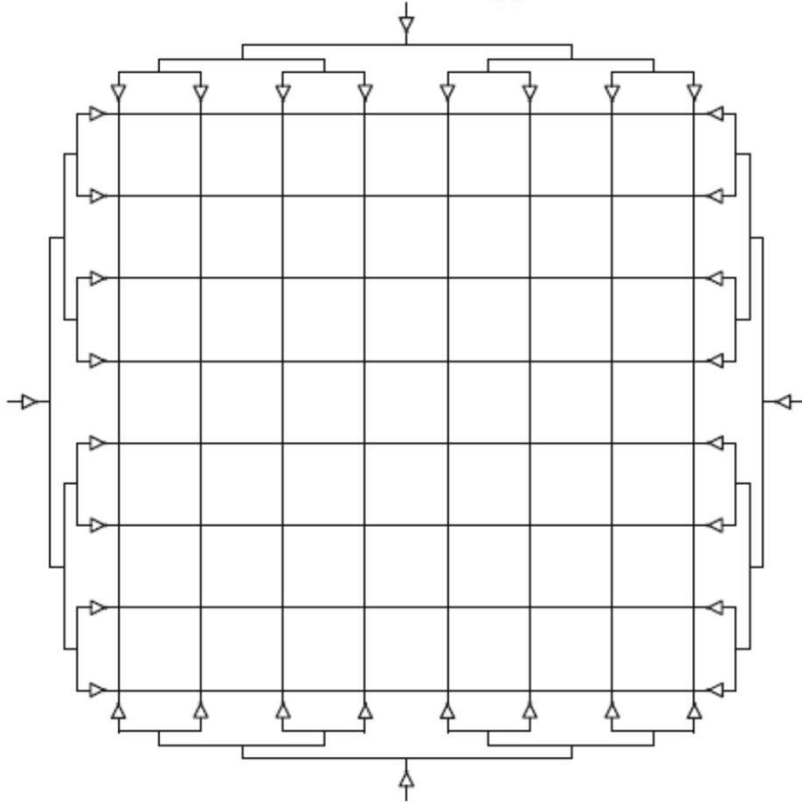


- More robust to variations because of alternative paths to clock sinks
- All points on a spine (trunk) considered to have the same delay

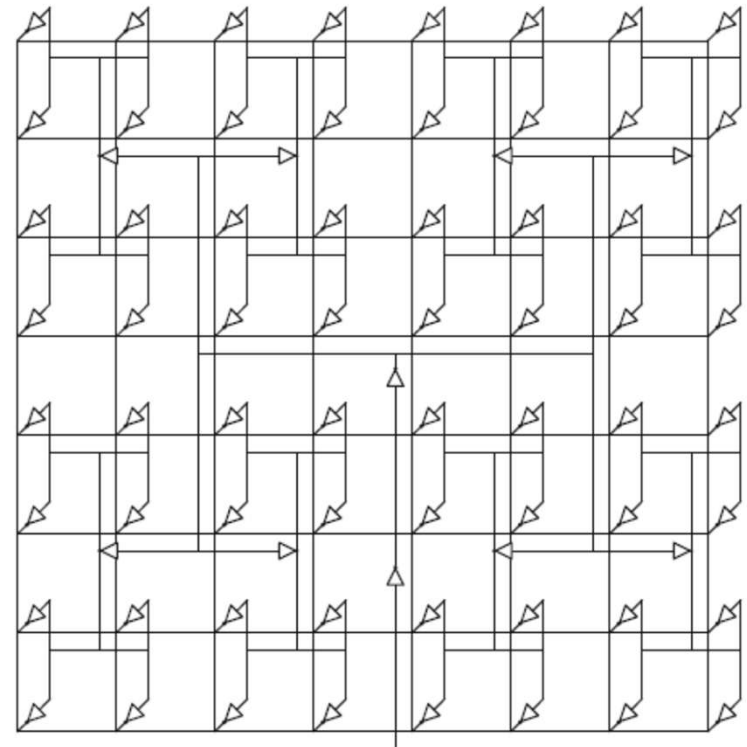
Intel Pentium 4 processor [Kurd *et al.* 2001]

# Clock meshes

- ❑ Even more alternative paths to clock sinks
- ❑ Drive mesh from the boundary or from grid points
- ❑ H-tree is a good candidate to drive mesh



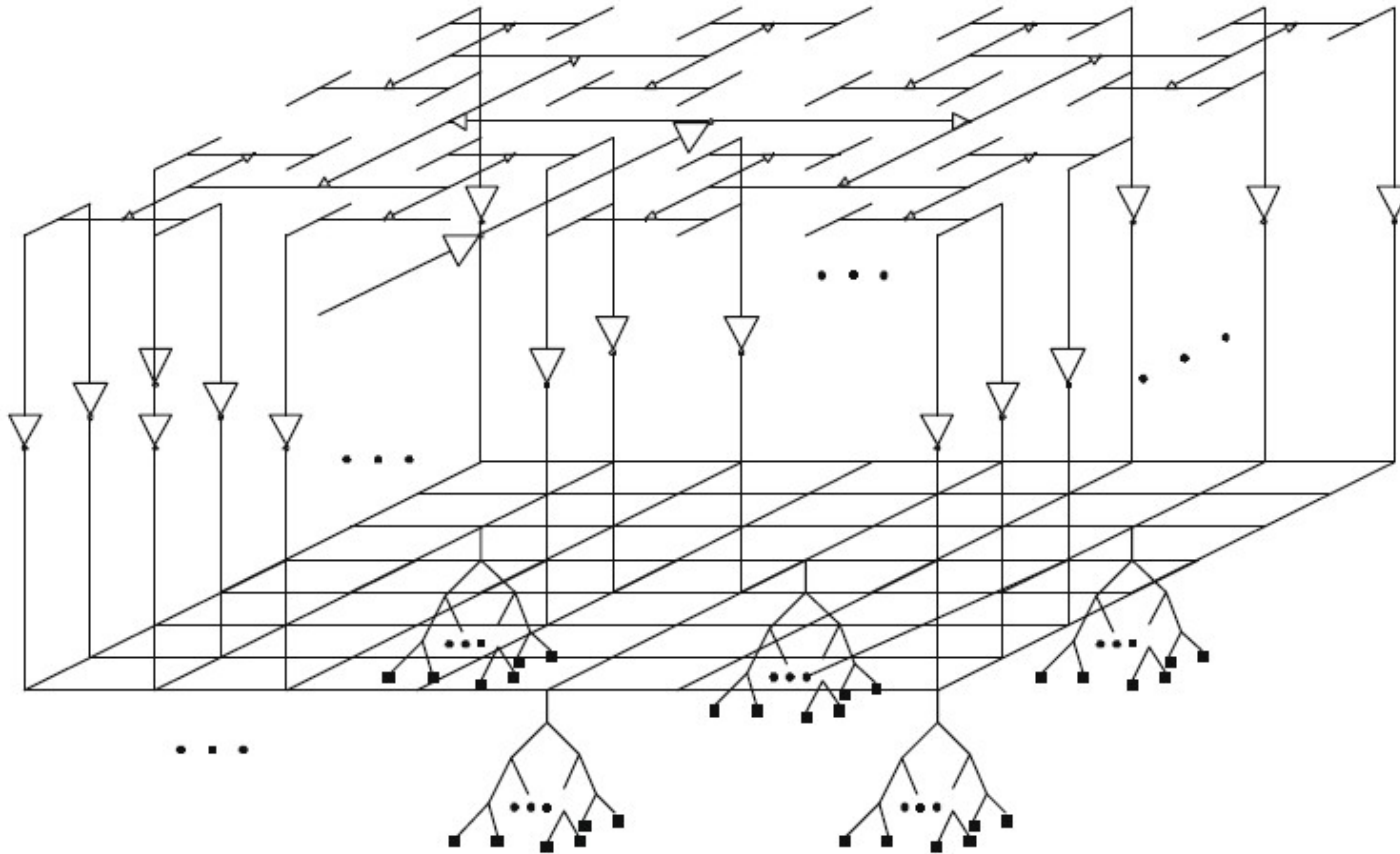
Alpha 21264 processor [Bailey *et al.* 1998]



IBM Power4 processor [Anderson *et al.* 2001]

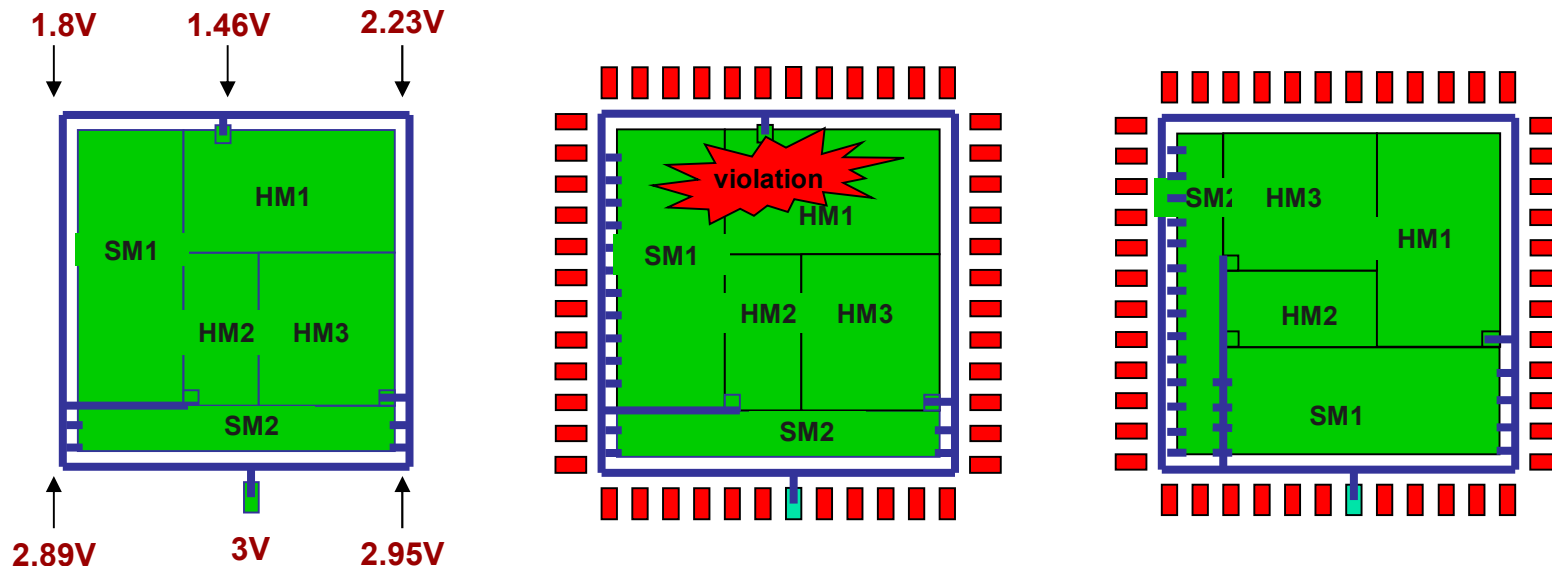
# *Hierarchical clock network*

- Global level and local level
- Use different topologies at different levels



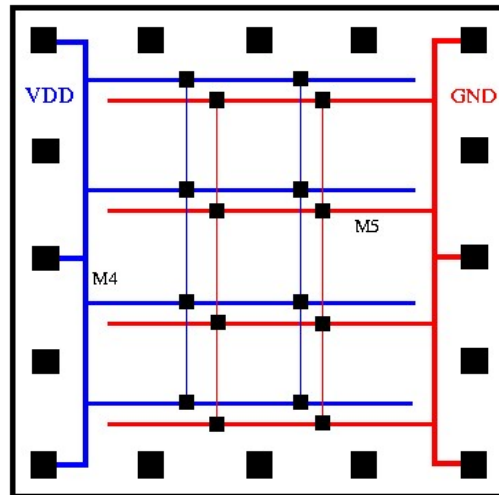
# IR (Voltage) Drop

- Power consumption and rail parasitics cause actual supply voltage to be lower than ideal
  - Metal width tends to decrease with length increasing in nanometer design
- Effects of IR drop
  - Reducing voltage supply reduces circuit speed (5% IR drop => 15% delay increase)
  - Reduced noise margin may cause functional failures

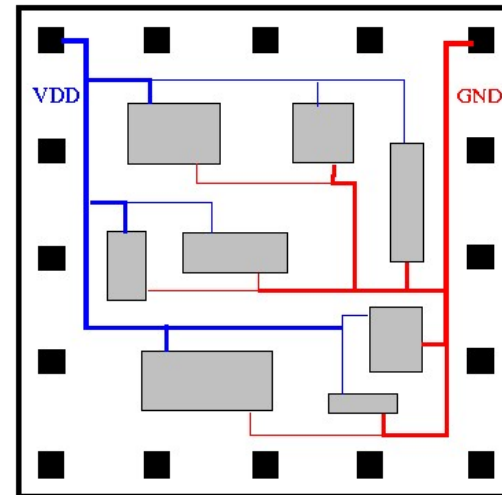


# Power/Ground (P/G) Routing

- Are usually laid out entirely on metal layers for smaller parasitics.
- Two steps:
  1. **Construction of interconnection topology:** non-crossing power, ground trees.
  2. **Determination of wire widths:** prevent metal migration, keep voltage (IR) drop small, widen wires for more power-consuming modules and higher density current (1.5 mA per  $\mu m$  width for Al). (So area metric?)



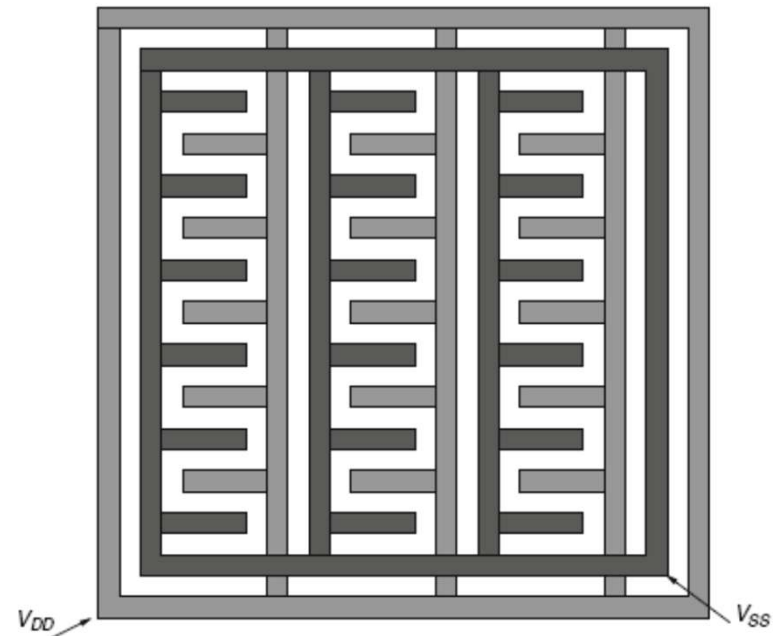
grids



interdigitated trees

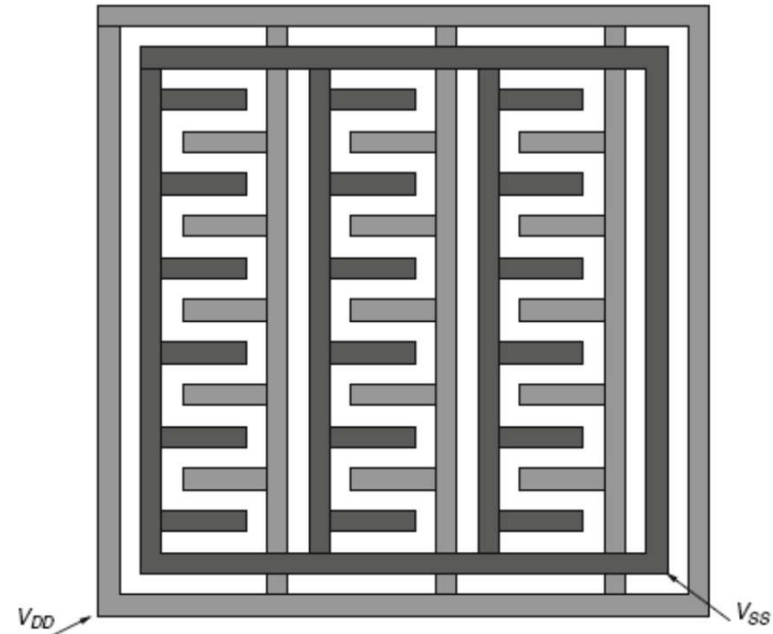
# Power/Ground Topologies

- ❑ The design of P/G distribution networks begins with the construction of an appropriate routing topology.
- ❑ A simple power distribution scheme is shown consisting of two large concentric rings (one power and one ground).



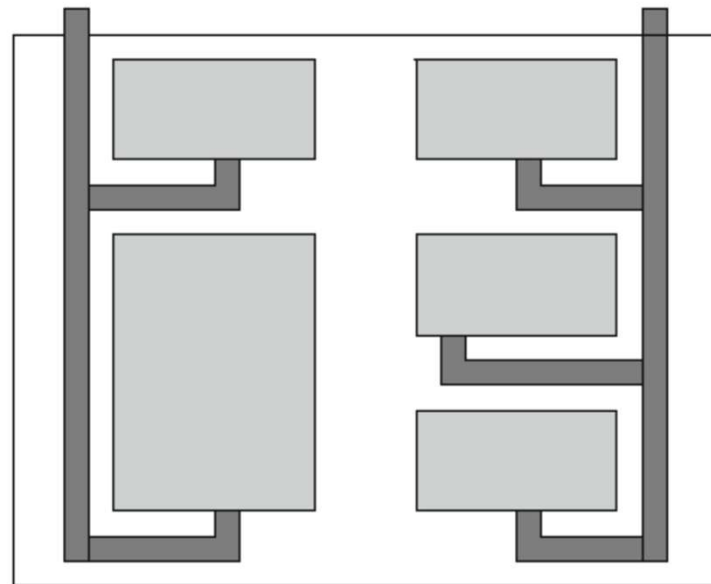
# Power/Ground Topologies

- ❑ Each attached comblike structure is commonly used for standard-cell designs.
- ❑ To counter power supply noise and electromigration, the concentric rings are typically of a larger width.
- ❑ The presence of larger modules such as memory blocks or bus lines would destroy the regularity shown.



# Power/Ground Topologies

- ❑ **Tree** and **mesh** structures are the most common topologies for power and ground routing
  - Power tree structures (shown) offer efficiency for resource consumption
  - Mesh structures typically perform better in minimizing voltage and current variations in the supply networks

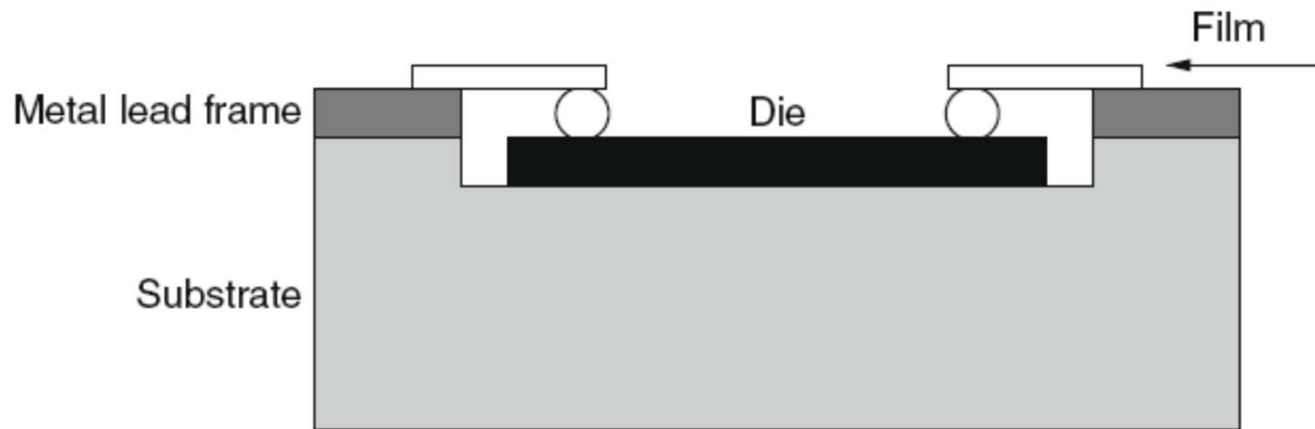


# *Power/Ground Topologies*

- ❑ As in the case of clock network design, it is common to see a combination of these various topologies in a single P/G network.
- ❑ Because of its robustness, a **mesh** structure typically sits at the **topmost level** in the hierarchy of a P/G network.
- ❑ Comblike structures and tree topologies, with wire width tapering, are usually used for the local distribution of power supply.

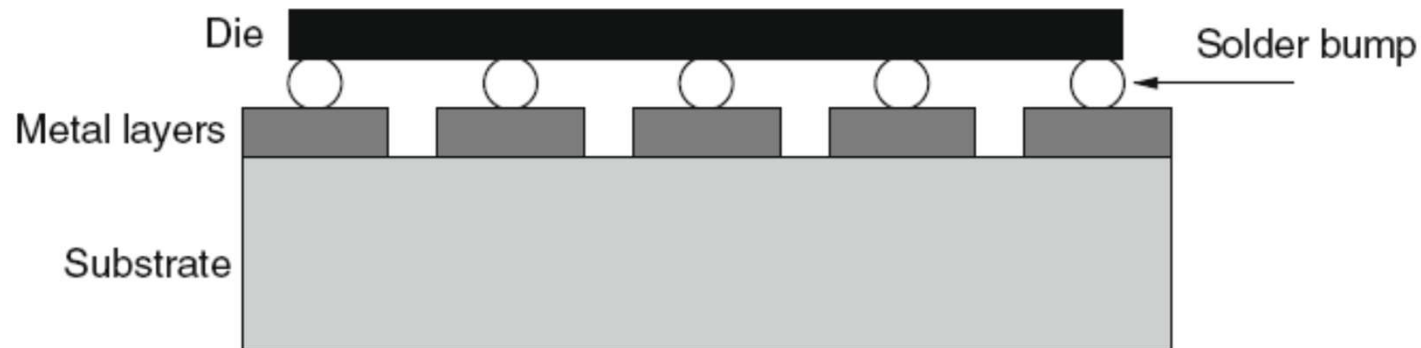
# Power/Ground Topologies

- ❑ Packaging technologies also play a significant role in enhancing the robustness of power supply.
- ❑ One of the most common interfaces for external power being supplied to an IC is along the periphery of the die.



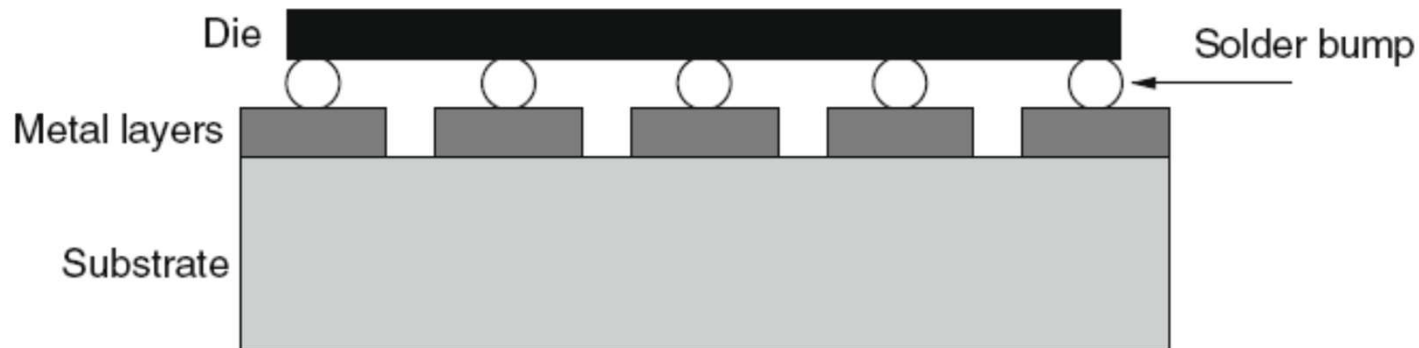
# Power/Ground Topologies

- ❑ Flip-chip packaging makes it possible to supply external power into the interior of the die area directly.
- ❑ For flip-chip mounting, VDD and GND pads are distributed across the topmost layer. The die is flipped upside down and connected to the substrate of the package with solder bumps.



# Power/Ground Topologies

- ❑ Flip-chip packaging provides two benefits: the power supply is available at any position on the chip and the parasitic inductances and capacitances of such packages are lower.
- ❑ Used in conjunction with a power mesh, the VDD (or GND) pads usually reside on the grid points of the mesh.



# *Power/Ground Network Analysis*

## □ DC Analysis

- Static IR drop
- Steady state values

## □ Transient Analysis

- Finding voltage fluctuations
- Dynamic IR drop
- $Ldi/dt$  noise

# *Power/Ground Network Synthesis*

## □ Problems

- Determine topology of P/G network
- Place power pads
- Insert decoupling capacitances

## □ Constraints

- Maximum current density for each wire
- Maximum voltage drop at each node

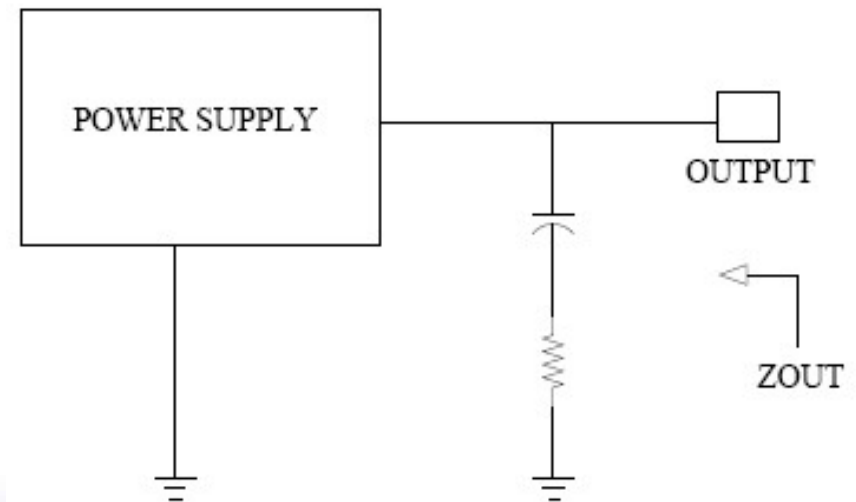
## □ Minimize wiring resource consumption

# Decoupling Capacitance

□ Decoupling capacitor placed next to load

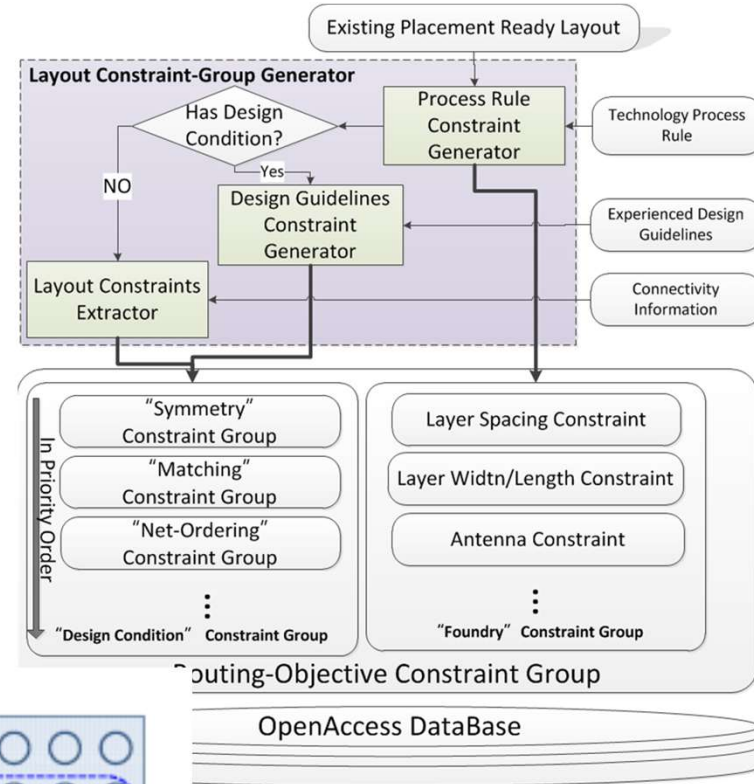
- Reduces size of current loop
- Reduces current induced noise
  - $IR$ ,  $Ldi/dt$

□ 10% of chip area needed for decoupling capacitors



# Miscellaneous Routing

- Analog routing



- Substrate/PCB routing

