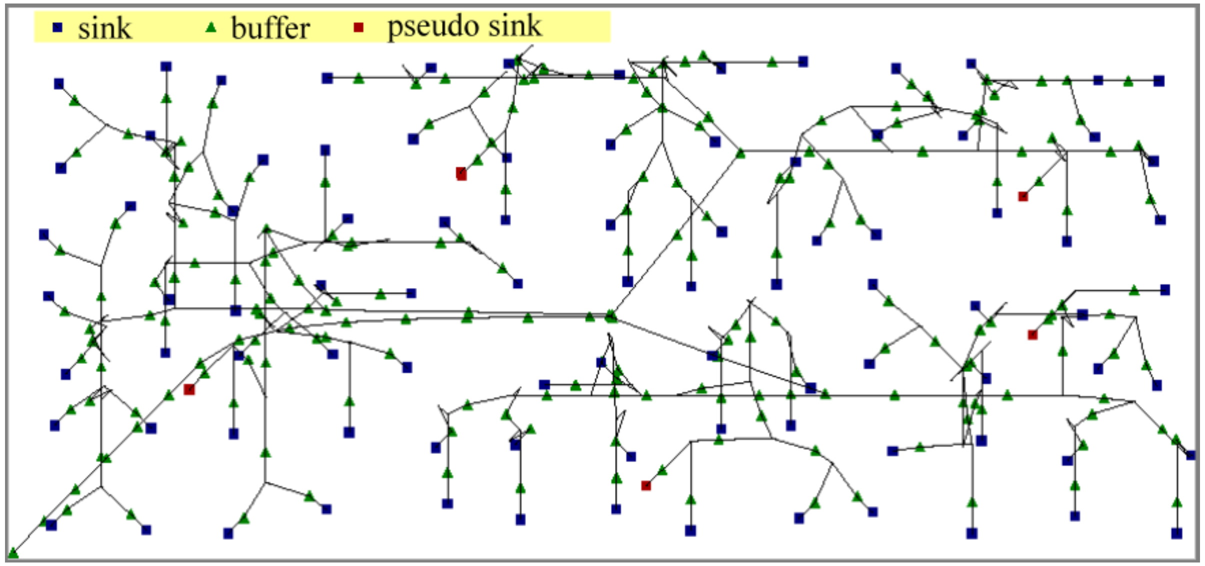
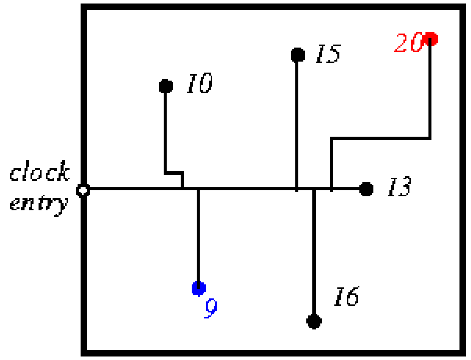
# Unit 7: Special Net Routing & Post-Layout Optimization

- Course contents:
  - Clock net routing
  - Power/ground routing
  - Performance optimization
- Readings
  - W&C&C: Chapter 13
  - S&Y: Chapter 7

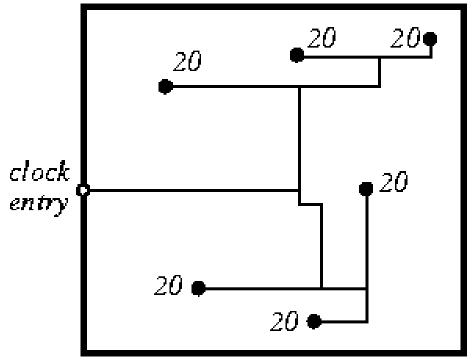


# The Clock Routing Problem

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



*clock skew* = 20 − 9 = 11



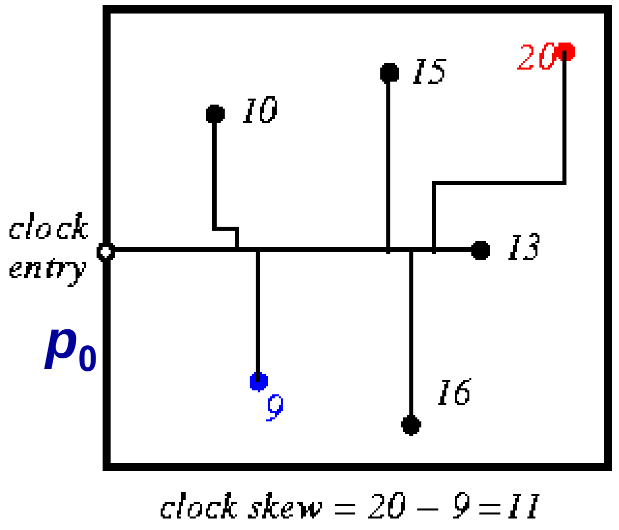
*clock skew* = 0

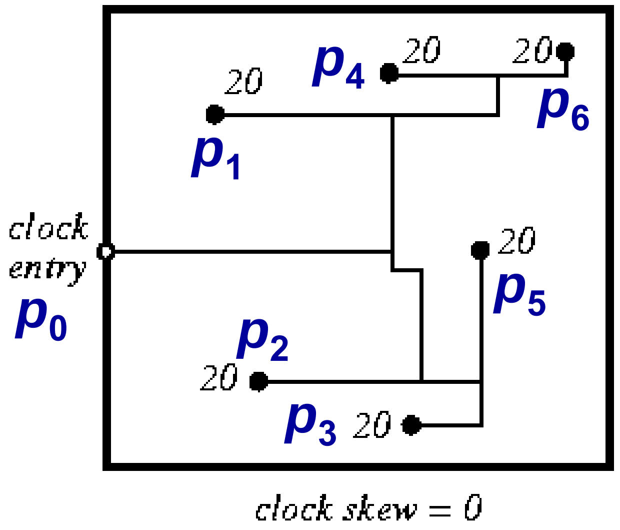
- **Clock routing:** Routing clock nets such that
  1. clock signals arrive simultaneously
  2. clock delay is minimized

  Other issues: total wirelength, power consumption

# Clock Routing

- Given the routing plane and a set of points $P = \{p_1, p_2, \ldots, p_n\}$ within the plane and clock entry point $p_0$ on the boundary of the plane, the **Clock Routing Problem** is to interconnect each $p_i \in P$ such that $\max_{i, j \in P}|t(0, i) - t(0, j)|$ and $\max_{i \in P} t(0, i)$ are both minimized.





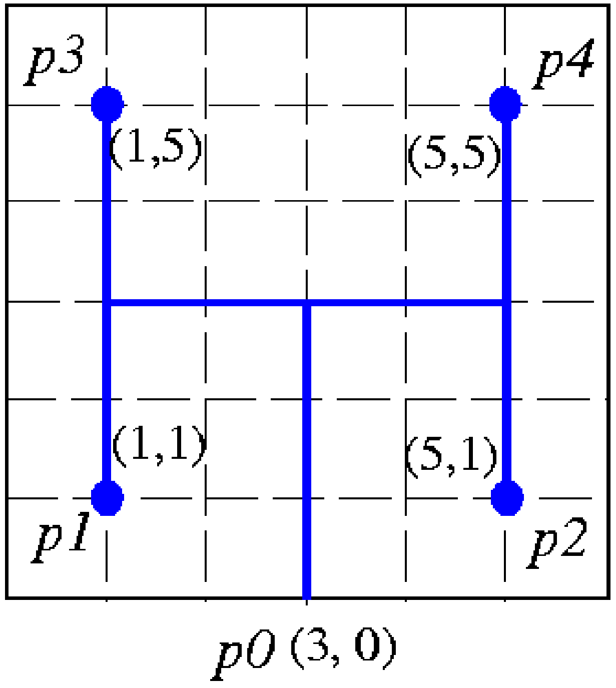
**Clock-tree synthesis (CTS): make the clock nets a tree**

# Clock Routing Algorithms

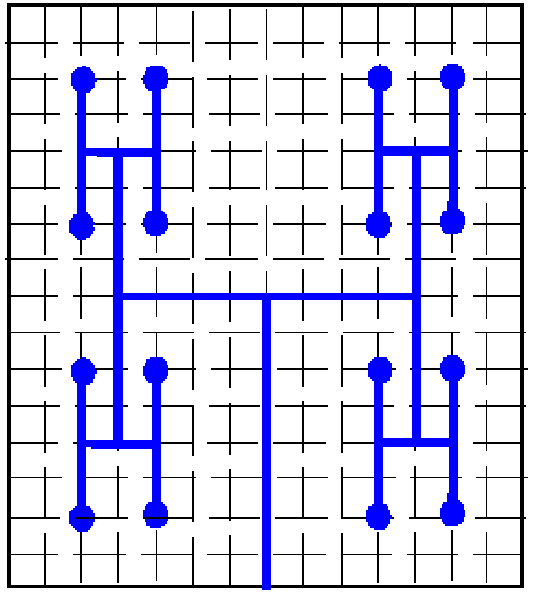
- Pathlength-based Clock-Tree Synthesis (CTS)
  1. *H*-tree: Dhar, Franklin, Wang, ICCD-84; Fisher & Kung, 1982.
  2. Methods of means & medians (MMM): Jackson, Srinivasan, Kuh, DAC-90.
  3. Geometric matching: Cong, Kahng, Robins, DAC-91.
- RC-delay based CTS
  1. Exact zero skew: Tsay, ICCAD-91.
  2. Deferred-merge embedding (DME) algorithm: Boese & Kahng, ASICON-92; Chao & Hsu & Ho, DAC-92; Edahiro, NEC R&D, 1991.
  3. Lagrangian relaxation: Chen, Chang, Wong, DAC-96.
- Simulation-based CTS
  - ISPD-09 CTS contest (ASP-DAC-10, DATE-10)
- Timing-model independent CTS
  - Shih & Chang, DAC-10; Shih et al., ICCAD-10.
- Mesh-based & tree-link-based clock routing

# H-Tree Based Algorithm

- *H*-tree: Dhar, Franklin, Wang, "Reduction of clock delays in VLSI structure," ICCD-1984.



*H–tree over 4 points*

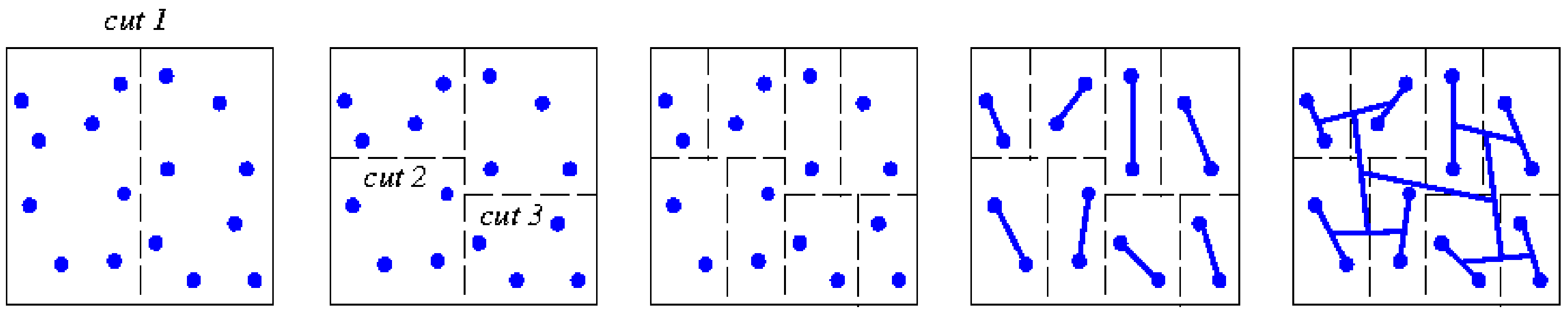


*H–tree over 16 points*

**Similar topology: X-tree**

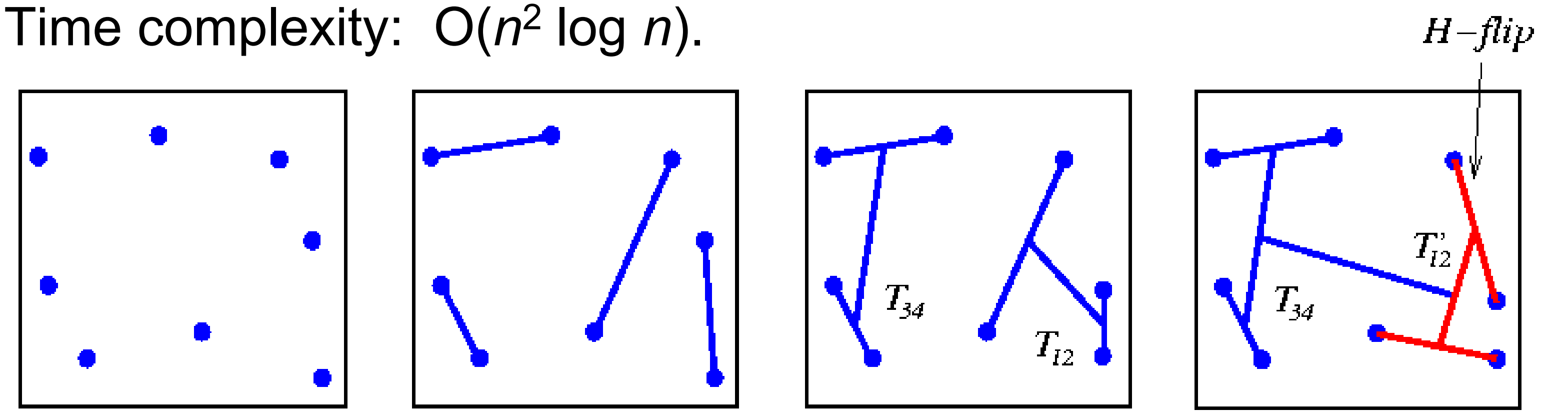
# The MMM Algorithm

- Jackson, Sirinivasan, Kuh, “Clock routing for high-performance ICs,” DAC-1990.
- Each block pin is represented as a point in the region, $S$.
- The region is partitioned into two subregions, $S_L$ and $S_R$.
- The center of mass is computed for each subregion.
- The center of mass of the region $S$ is connected to each of the centers of mass of subregion $S_L$ and $S_R$.
- The subregions $S_L$ and $S_R$ are then recursively split in $Y$-direction.
- Steps 2--5 are repeated with alternate splitting in $X$- and $Y$-direction.
- Time complexity: $O(n \log n)$.



# The Geometric Matching Algorithm

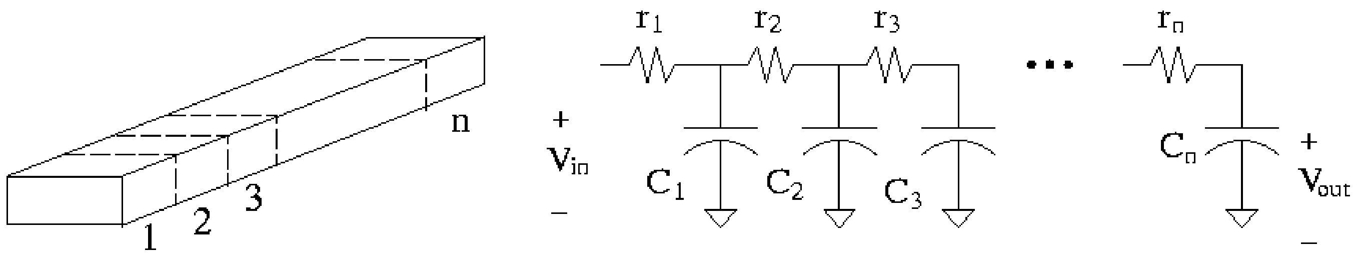
- Cong, Kahng, Robins, "Matching based models for high-performance clock routing," IEEE TCAD, 1993.
- Clock pins are represented as $n$ nodes in the clock tree ($n = 2^k$).
- Each node is a tree itself with clock entry point being node itself.
- The minimum cost matching on $n$ points yields $n/2$ segments.
- 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 edges 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.

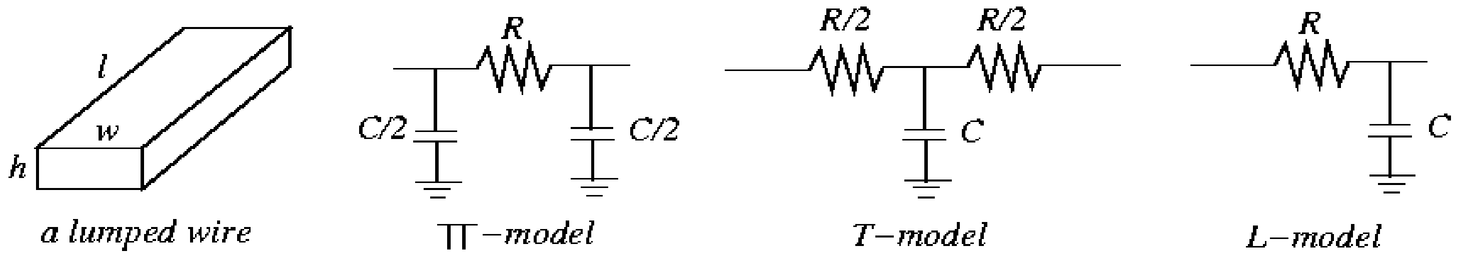
$$\delta = \sum_{i=1}^{n}\left(r_i \sum_{k=i}^{n} c_k\right) = \sum_{i=1}^{n} r(n-i+1)c = \frac{n(n+1)}{2}rc.$$



- Delay grows as **square** of wire length.
- Cannot apply to the delay with **inductance** consideration, which is important in high-performance design.

# Wire Models

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

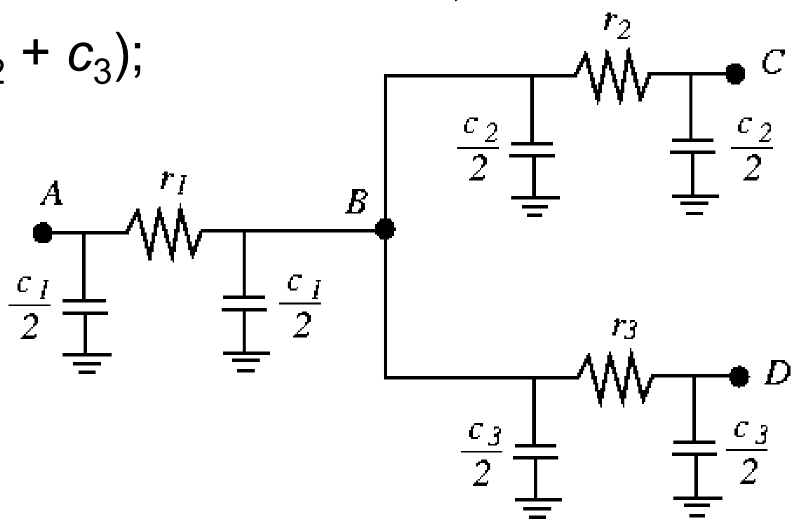


- $\pi$-model: If no capacitive loads for *C* and *D*,

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

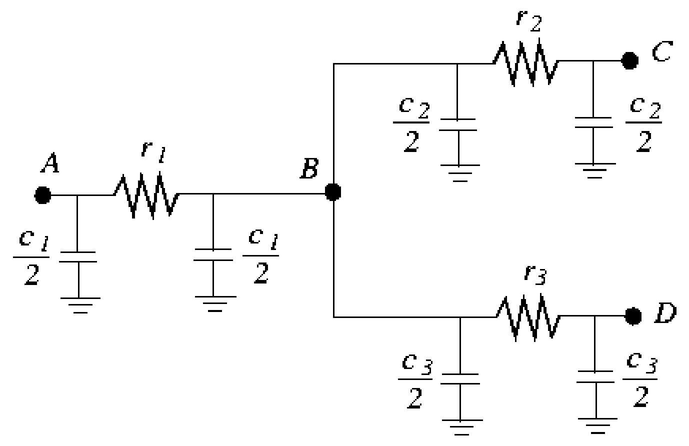
  *B* to *C*: $\delta_{BC} = r_2\ (c_2/2)$;

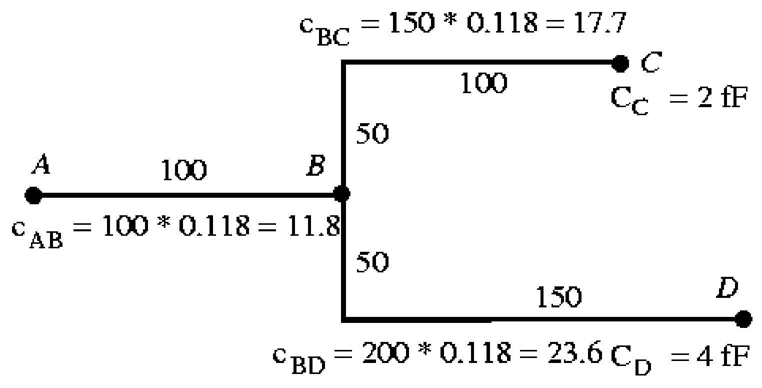
  *B* to *D*: $\delta_{BD} = r_3\ (c_3/2)$.



# Example Elmore Delay Computation

- 0.18 $\mu m$ technology: unit resistance $\hat{r} = 0.075\ \Omega / \mu m$; unit capacitance $\hat{c} = 0.118\ fF/\mu m$.
  - 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.





$c_{BC} = 150 * 0.118 = 17.7$
C
100
$C_C = 2$ fF
50
A
100
B
$c_{AB} = 100 * 0.118 = 11.8$
50
150
D
$c_{BD} = 200 * 0.118 = 23.6$
$C_D = 4$ fF

# Exact Zero Skew Algorithm

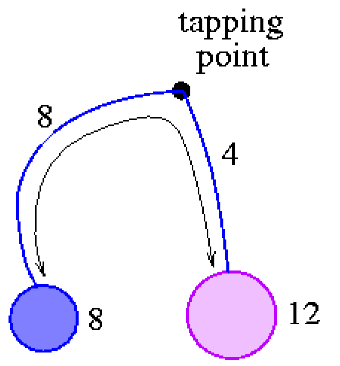
- Tsay, "Exact zero skew algorithm," ICCAD-91.
- To ensure the delay from the **tapping point** to leaf nodes of subtrees $T_1$ and $T_2$ being equal, it requires that

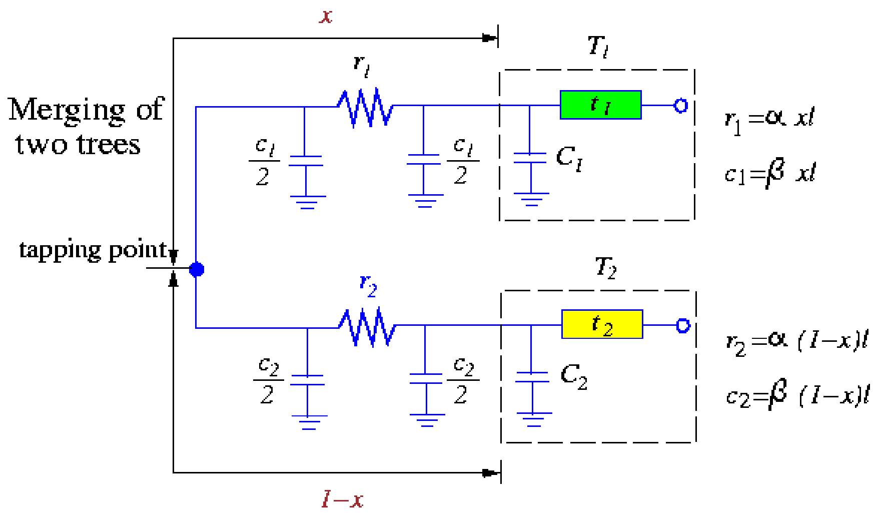
$$r_1 (c_1/2 + C_1) + t_1 = r_2 (c_2/2 + C_2) + t_2.$$

- Solving the above equation, we have

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

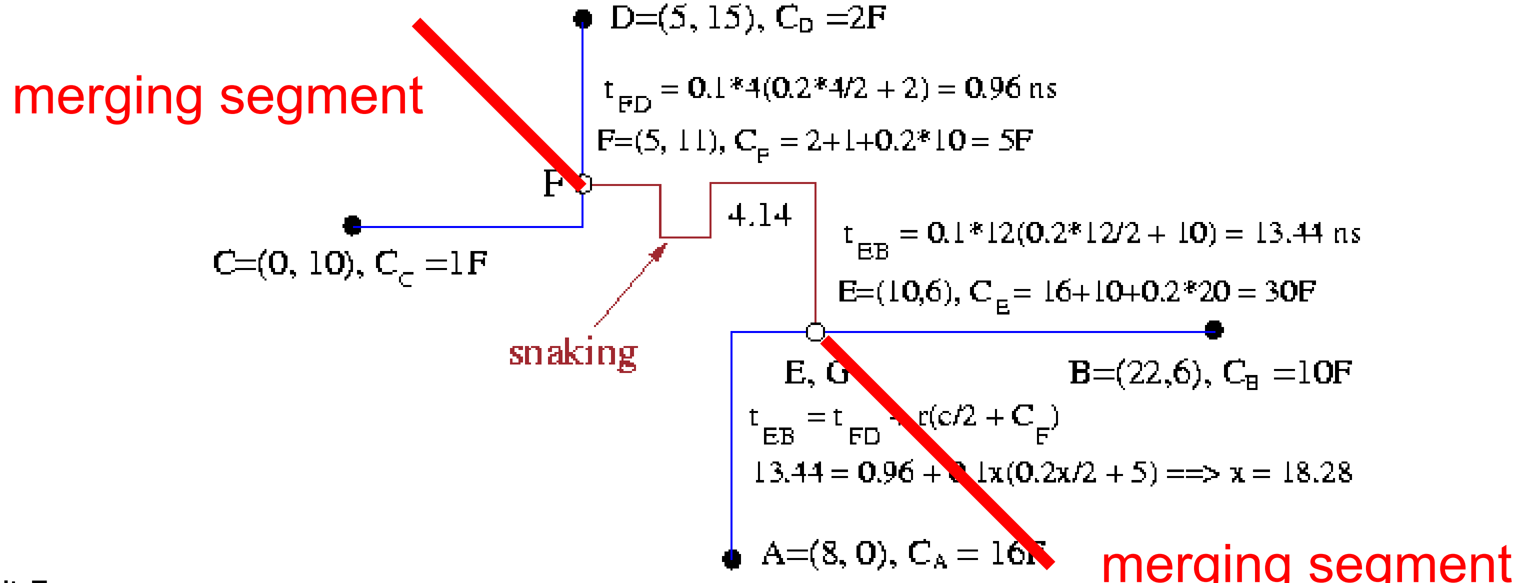
where $\alpha$ and $\beta$ are the per unit values of resistance and capacitance, $l$ the length of the interconnecting wire, $r_1 = \alpha x l$, $c_1 = \beta x l$, $r_2 = \alpha(1 - x)l$, $c_2 = \beta(1 - x)l$.





## Zero-Skew Computation

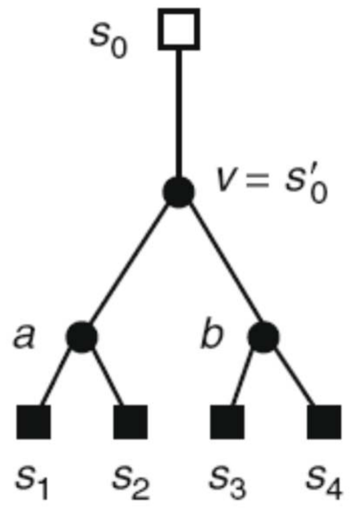
- **Balance delays:** $r_1(c_1/2 + C_1) + t_1 = r_2(c_2/2 + C_2) + t_2$.
- **Compute tapping points:** $x = \frac{(t_2 - t_1) + \alpha l \left(C_2 + \frac{\beta l}{2}\right)}{\alpha l(\beta l + C_1 + C_2)}$, $\alpha$ ($\beta$): per unit values of resistance (capacitance); $l$: length of the wire; $r_1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

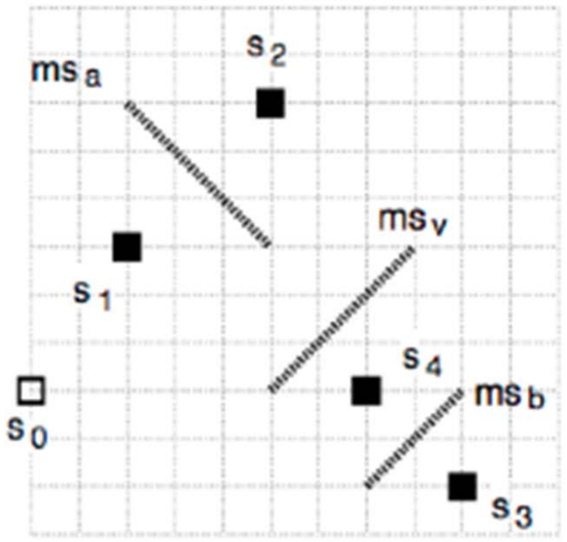


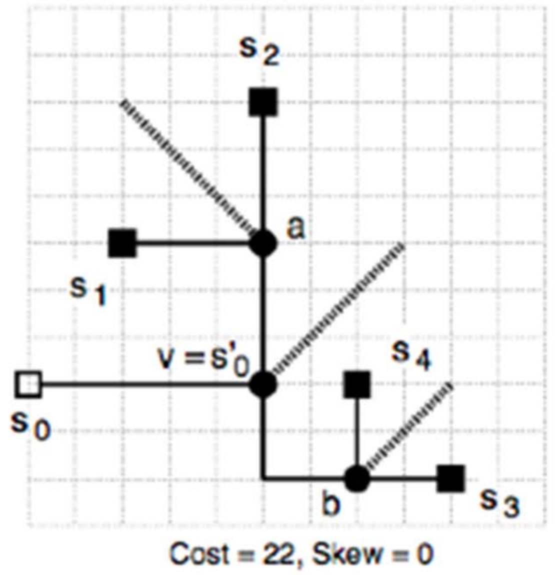
D=(5, 15), C_D =2F
merging segment
t_FD = 0.1*4(0.2*4/2 + 2) = 0.96 ns
F=(5, 11), C_F = 2+1+0.2*10 = 5F
F
4.14
C=(0, 10), C_C =1F
t_EB = 0.1*12(0.2*12/2 + 10) = 13.44 ns
E=(10,6), C_E = 16+10+0.2*20 = 30F
snaking
E, G
B=(22,6), C_B =10F
t_EB = t_FD + r(c/2 + C_F)
13.44 = 0.96 + 0.1x(0.2x/2 + 5) ==> x = 18.28
A=(8, 0), C_A = 16F
merging segment

# Deferred Merge Embedding (DME)

- Boese & Kahng, ASICON-92; Chao & Hsu & Ho, DAC-92; Edahiro, NEC R&D, 1991
- Consists of two stages: bottom-up + top-down
- Bottom-up: Build the potential embedding locations of clock sinks (i.e., a **segment** for potential tapping points)
- Top-down: Determine exact locations for the embedding

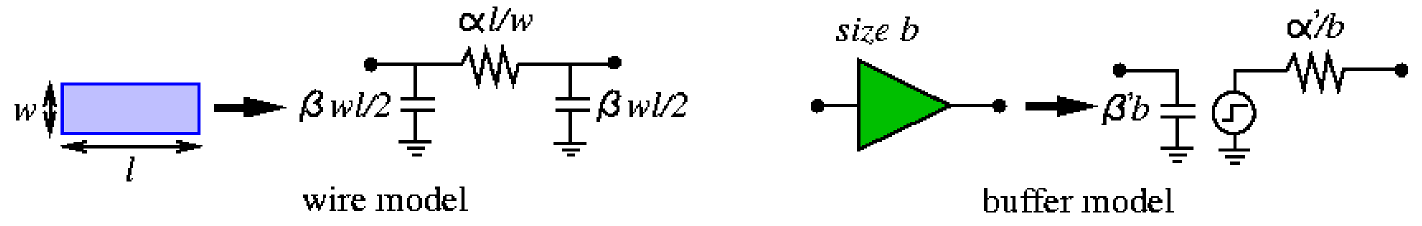


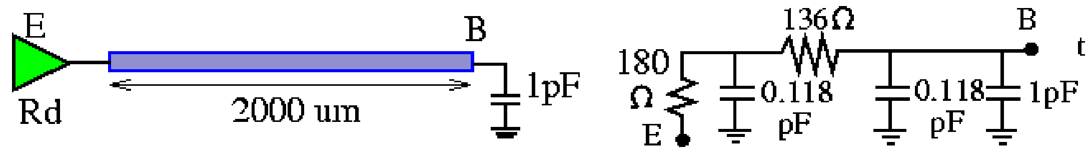




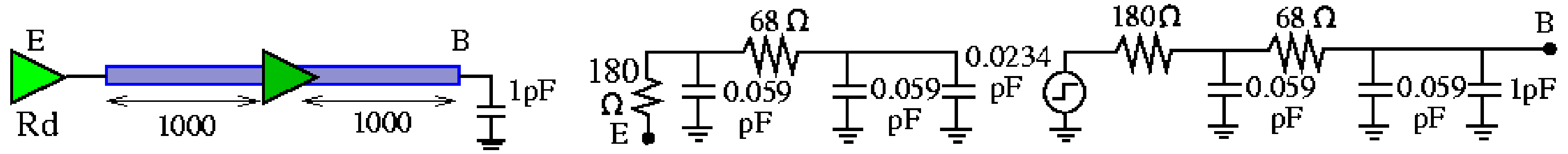
# Delay Computation for Buffered Wires

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

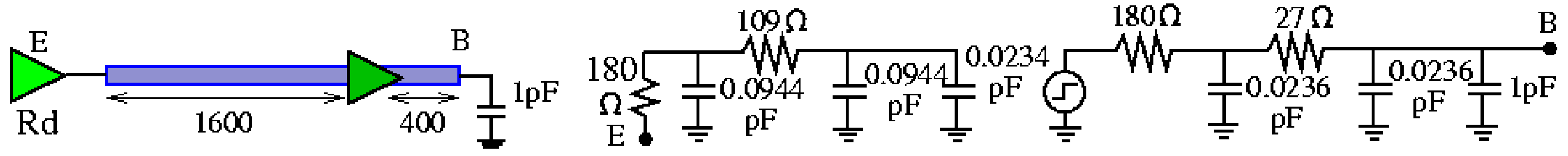




$t_{EB} = 180(0.118+0.118+1)+136(0.118+1)$
$= 375$ psec



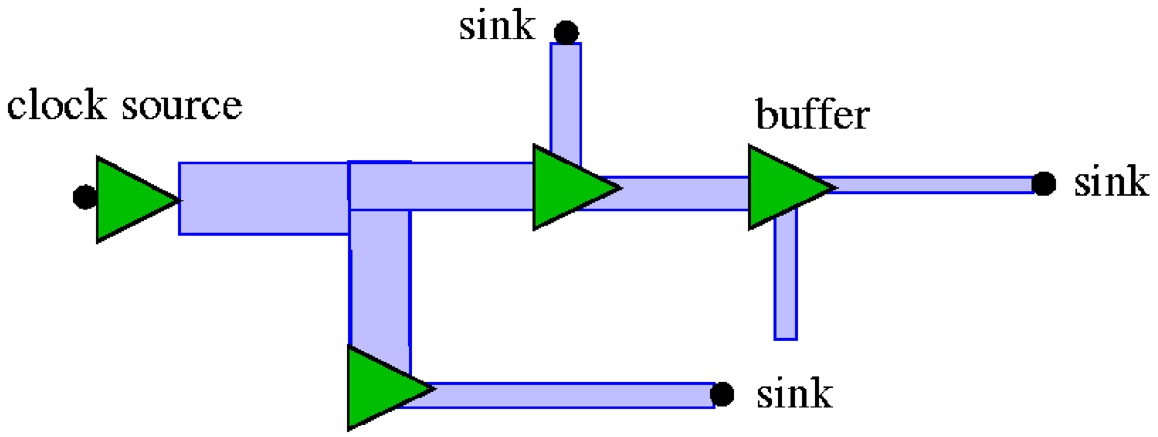
$t_{EB} = 180(0.059+0.059+0.0234)+68(0.059+0.0234)$
$+ 180(0.059+0.059+1)+68(0.059+1) = 304$ psec



$t_{EB} = 180(0.0944+0.0944+0.0234)+109(0.0944+0.0234)$
$+ 180(0.0236+0.0236+1)+27(0.0236+1) = 267$ psec

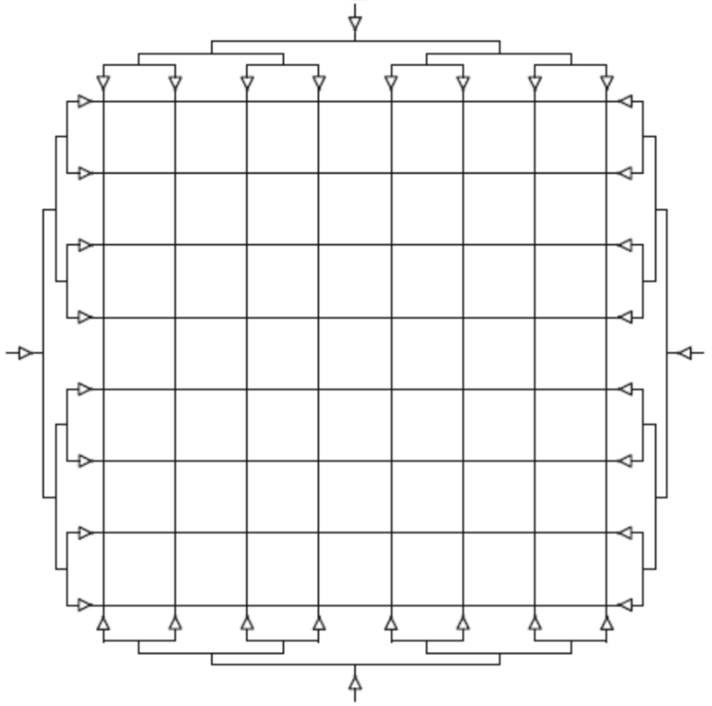
## Buffering and Wire Sizing for Skew Minimization

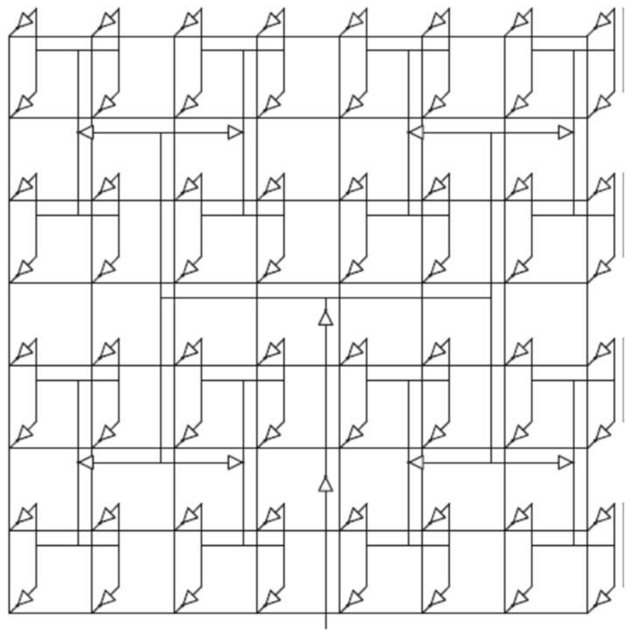
- Discrete wire/buffer sizes: dynamic programming
  - Chung & Cheng, "Skew sensitivity minimization of buffered clock tree," ICCAD-94.
- Continuous wire/buffer sizes: mathematical programming (e.g., Lagrangian relaxation)
  - Chen, Chang, Wong, "Fast performance-driven optimization for buffered clock trees based on Lagrangian relaxation," DAC-96.
  - Considers clock skew, area, delay, power, clock-skew sensitivity simultaneously.



# Clock Meshes

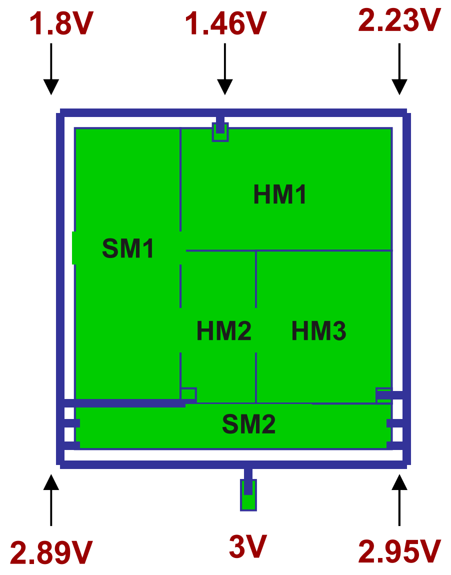
- More alternative paths to clock sinks
  - Good for high-performance circuits with stringent skew and variation constraints
- Drive mesh from the boundary or from grid points
- H-tree is a good candidate to drive mesh

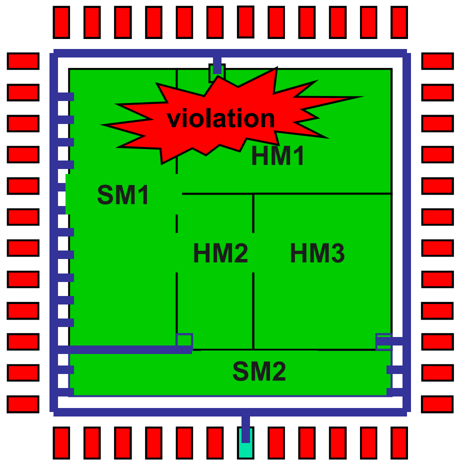

Alpha 21264 processor [Bailey *et al.* 1998]

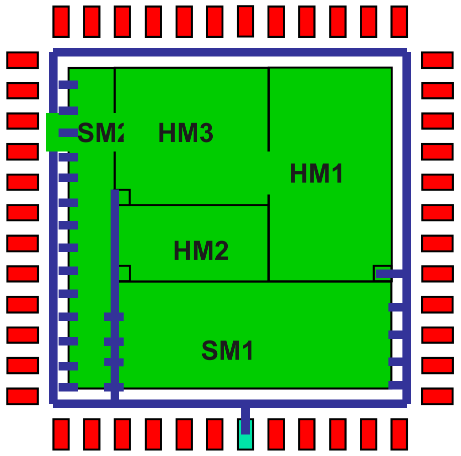

IBM Power4 processor [Anderson *et al.* 2001]

# Power Integrity: 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

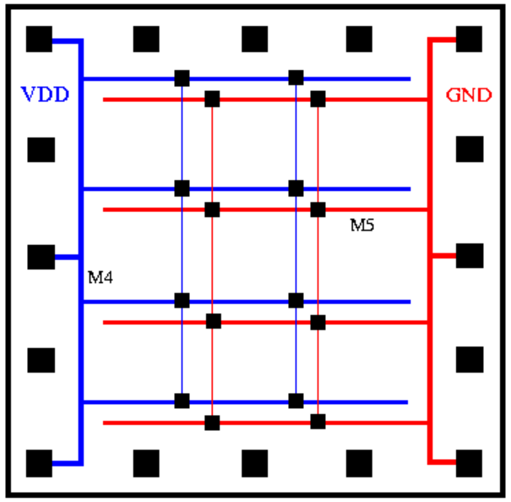




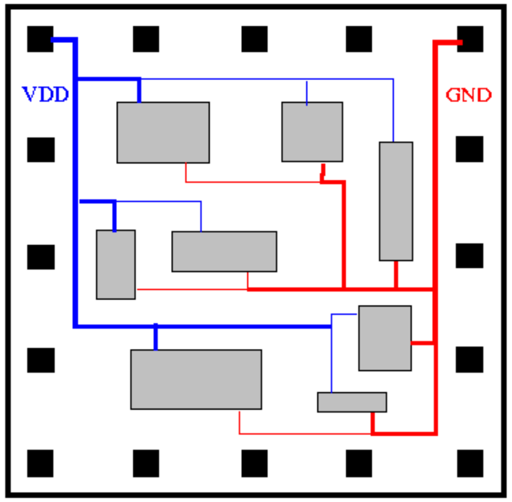


# Power/Ground (P/G) Routing

- 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 mA / $\mu$ *m*$^2$ at 25 $^o$C for 0.18 $\mu$ *m technology)*. (So area metric?)



grids



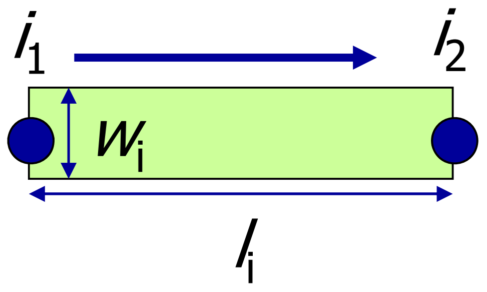
interdigitated trees

## Power/Ground Network Optimization

- Use the minimum amount of chip area for wiring P/G networks while avoiding potential reliability failures due to electromigration and excessive IR drops.
- Tan and Shi, "Fast power/ground network optimization based on equivalent circuit modeling", DAC-2001.
  - Build the equivalent models for series resistors and apply a sequence of the linear programming (SLP) method to solve the problem.
  - Size wire segments assuming the topologies of P/G networks to be fixed.
- Wu and Chang, "Efficient power/ground network analysis for power integrity driven design methodology," DAC-2004.
- Liu and Chang, "Floorplan and power/ground co-synthesis for fast design convergence," ISPD-06 (TCAD-07).
- Chang, et al., "Generating routing-driven power distribution networks with machine-learning technique," ISPD-16.

# Problem Formulation

- Let $G = \{N, B\}$ be a P/G network with $n$ nodes $N = \{1, \ldots, n\}$ and $b$ branches $B = \{1, \ldots, b\}$; branch $i$ connects two nodes: $i_1$ and $i_2$ with current flowing from $i_1$ to $i_2$.
- Let $l_i$ and $w_i$ be the length and width of branch $i$, respectively. Let $\rho$ be the sheet resistivity. Then the resistance $r_i$ of branch $i$ is $r_i = \frac{V_{i1} - V_{i2}}{I_i} = \rho \frac{l_i}{w_i}$.
- Total P/G routing area is as follows:



$$f(\mathbf{V}, \mathbf{I}) = \sum_{i \in B} l_i w_i = \sum_{i \in B} \frac{\rho I_i l_i^2}{V_{i_1} - V_{i_2}}.$$

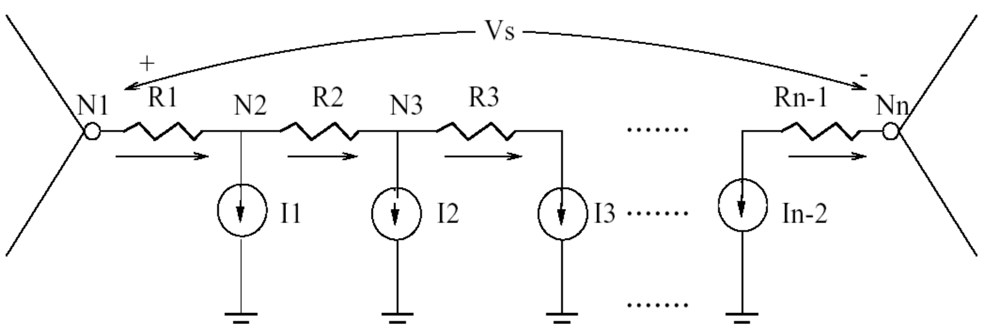
- P/G network optimization is to minimize *f(V, I)* subject to the constraints listed in the next slide.
- Relax the nonlinear objective function and then translate the constrained nonlinear programming problem into a SLP problem

# Constraints

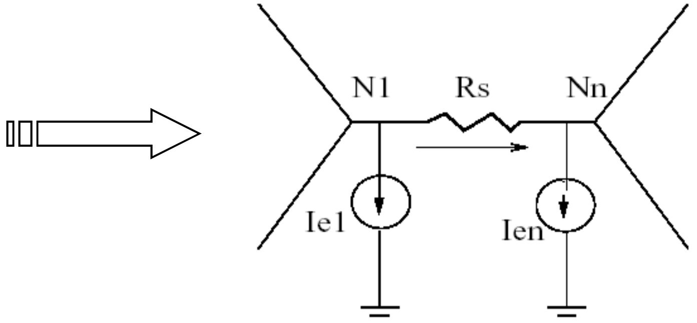
- The voltage IR drop constraints.
  - $V_i \geq V_{h,\min}$ for power networks.
  - $V_i \leq V_{l,\max}$ for ground networks.
- The minimum width constraints: $w_i = \rho \dfrac{l_i I_i}{V_{i1} - V_{i2}} \geq w_{i,\min}$
- The electro-migration constraints: $I_i/w_i \leq \sigma$ => $|V_{i1} - V_{i2}| \leq \rho l_i \sigma$
  - σ is a constant for a particular routing layer with a fixed thickness.
- Equal width constraints: $w_i = w_j$ or $\dfrac{v_{i1} - v_{i2}}{l_i I_i} = \dfrac{v_{j1} - v_{j2}}{l_j I_j}$
- Kirchoff 's current law (KCL): $\sum_{i \in B(j)} I_i = 0$
  - For each node *j* = {1, …, *n*}, *B*(*j*) is the set of indices of branches connecting to node j.

## Reducing the Problem Size with Equivalent Circuits

- Consider a series resistor chain commonly seen in the P/G network below.



Series resistor chain



Equivalent circuit

- The equivalent resistor $R_s$ is just the sum of all the resistors in series, $R_s = \sum_{i=1}^{n-1} R_i$.
- By superposition, the equivalent currents $I_{e1}$, and $I_{en}$ can be computed as follows:

$$I_{e_1} = \sum_{i=1}^{n-2} \frac{\sum_{j=i+1}^{n-1} R_j}{R_s} I_i,$$

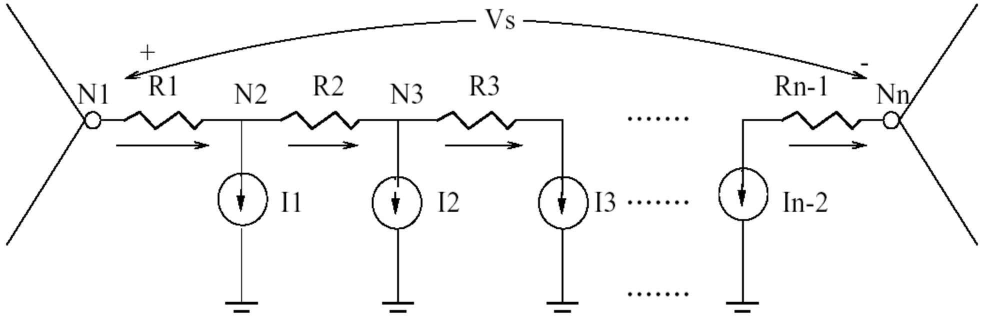
$$I_{e_n} = \sum_{i=1}^{n-2} \frac{\sum_{j=1}^{i} R_j}{R_s} I_i.$$

# Equivalent Circuit (cont'd)

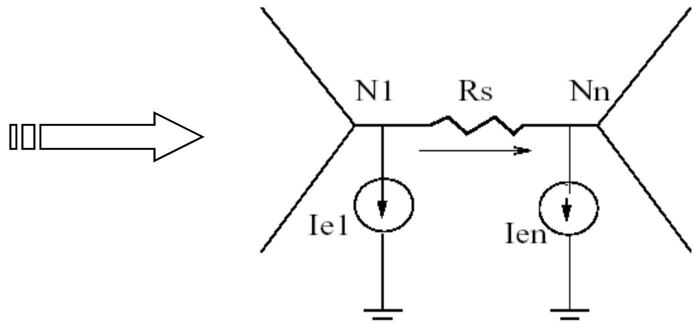
- The voltages at the intermediate nodes are calculated based on superposition as follows:

$$V_{i+1} = V_i - \frac{R_i}{R_s} V_s - R_i I_{ei}$$

$$I_{ei+1} = I_{ei} - I_i$$

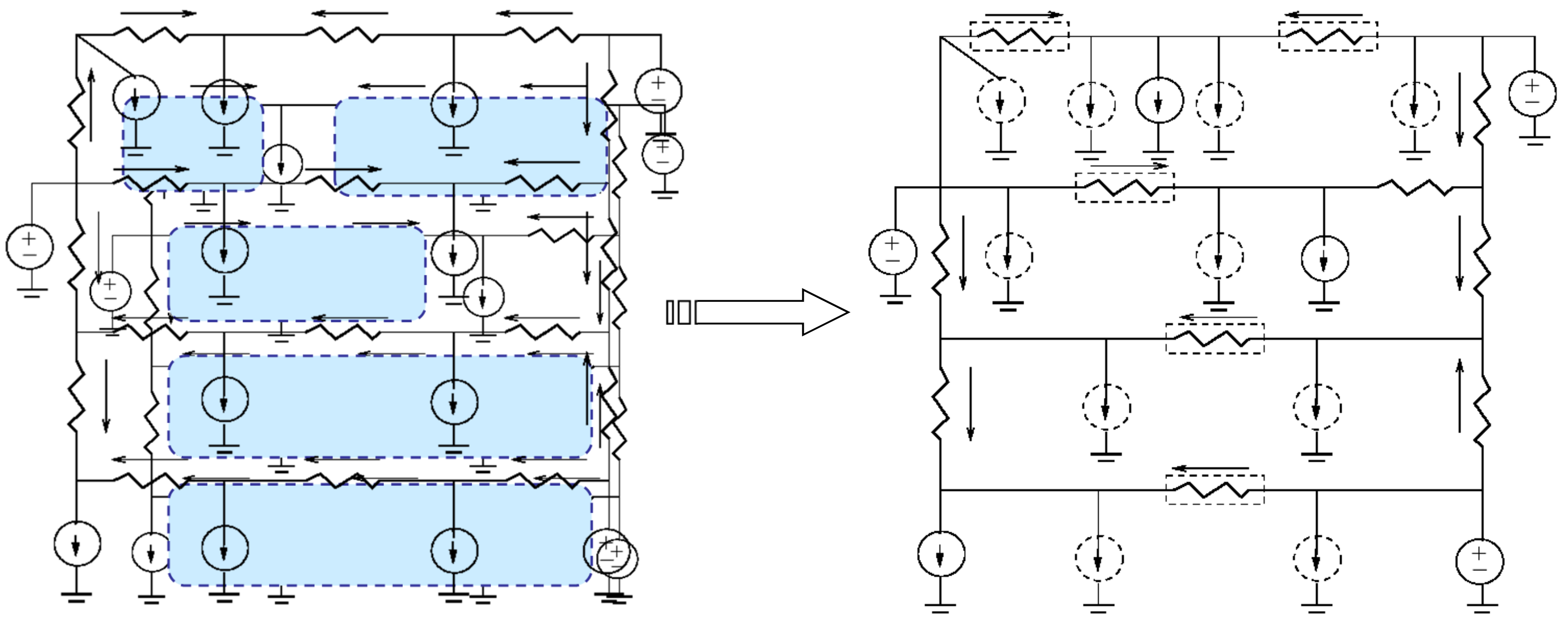


Series resistor chain



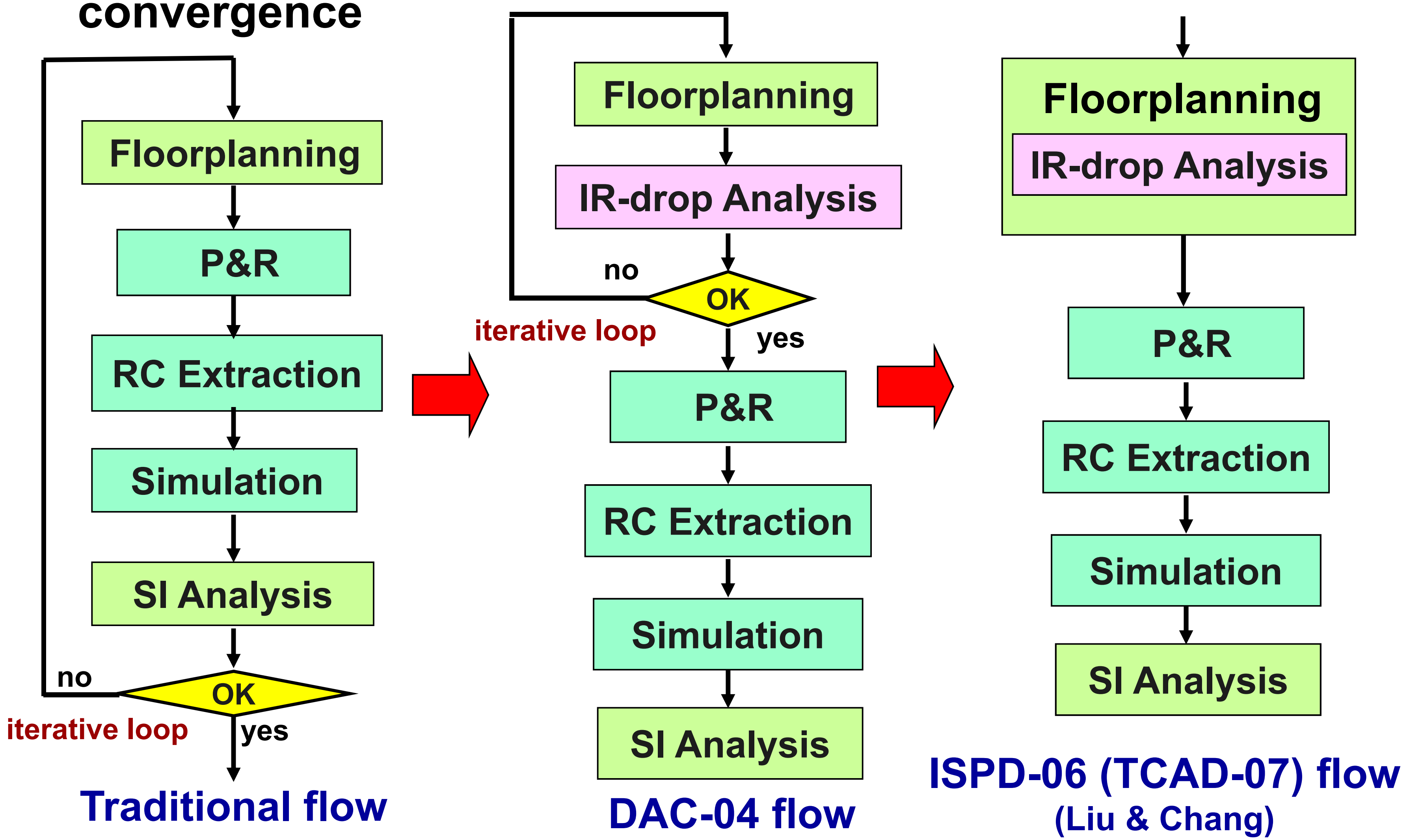
Equivalent circuit

# Equivalent Circuit Example



# Design Methodology Evolution

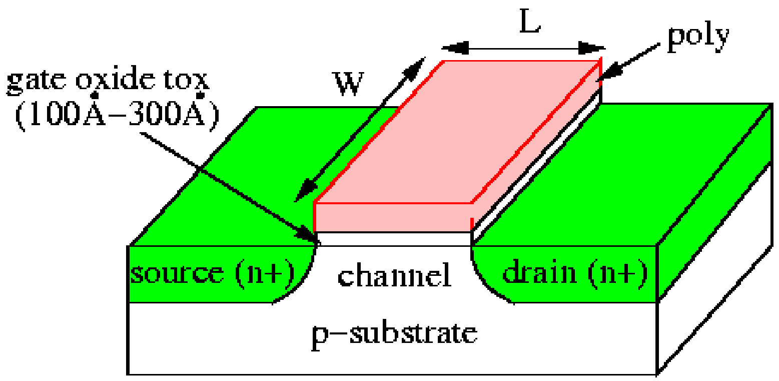
- IR-drop aware design methodology for faster design convergence



## Ideal Scaling of MOS Transistors

- Feature size scales down by *S* times:

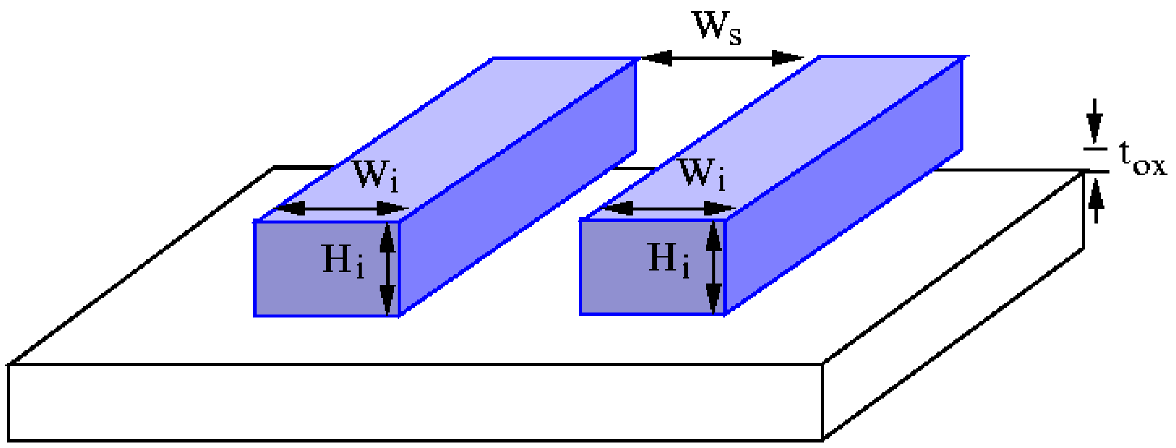
| Parameter | Scaling factor |
|---|---|
| Dimensions ($W, L, t_{ox}$, junction depth $X_j$) | $1/S$ |
| Area per device ($A = WL$) | $1/S^2$ |
| Substrate doping ($N_{SUB}$) | $S$ |
| Voltages ($V_{DD}, V_t$) | $1/S$ |
| Current per device ($I_{ds} \propto \frac{W\,\varepsilon_{ox}}{L\,t_{ox}}(V_{DD} - V_t)^2$) | $1/S$ |
| Gate capacitance ($C_g = \varepsilon_{ox} WL/t_{ox}$) | $1/S$ |
| Transistor on-resistance ($R_{tr} \propto V_{DD}/I_{ds}$) | 1 |
| **Intrinsic gate delay** ($\tau = R_{tr} C_g$) | $1/S$ |
| Power dissipation per gate ($P = IV$) | $1/S^2$ |
| Power-dissipation density ($P/A$) | 1 |



# Ideal Scaling of Interconnections

- Feature size scales down by *S* times:

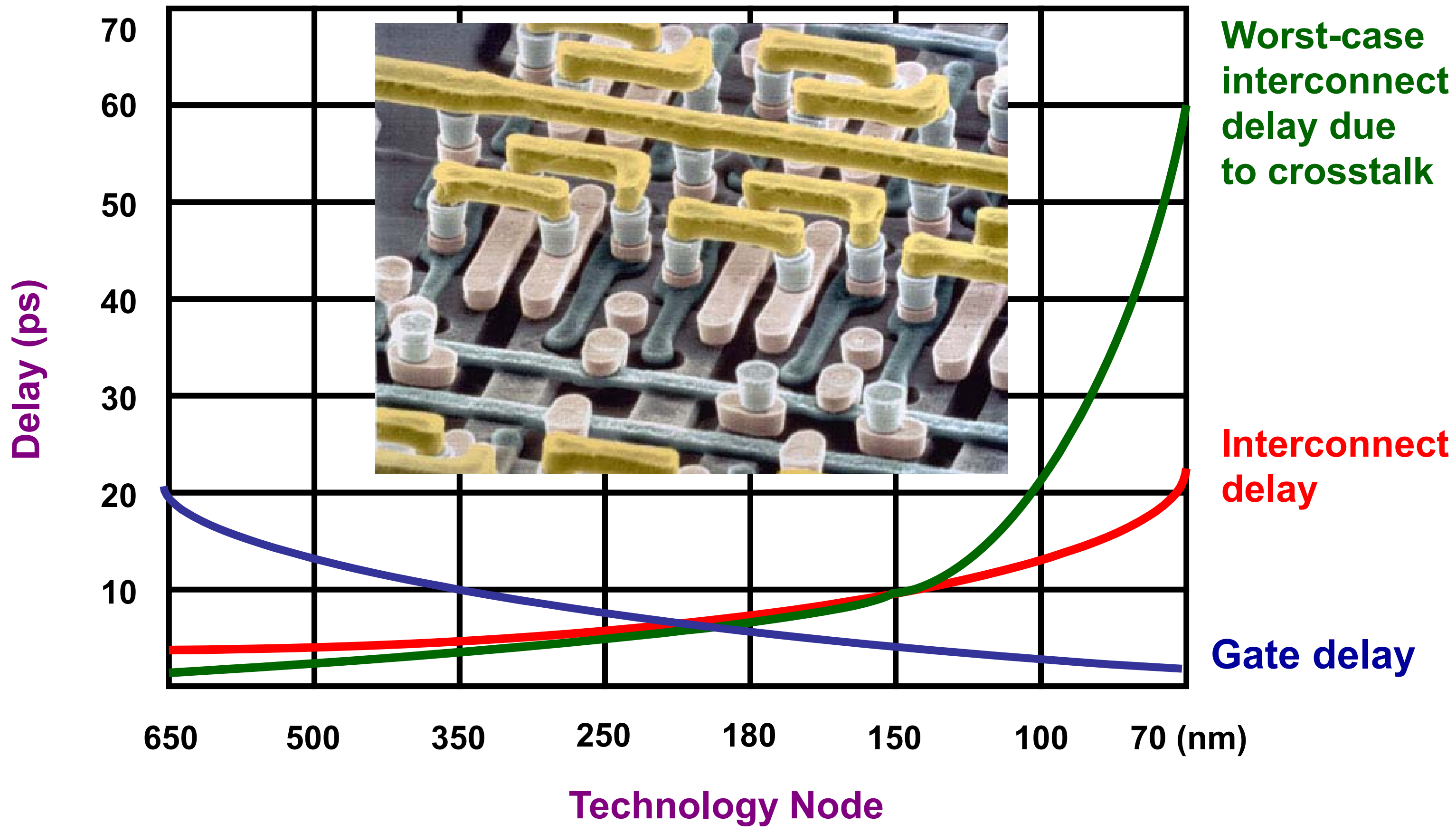
| Parameter | Scaling factor |
|---|---|
| Cross sectional dimensions ($W_i, H_i, W_s, t_{ox}$) | $1/S$ |
| Resistance per unit length ($r_0 = \rho/W_iH_i$) | $S^2$ |
| Capacitance per unit length ($c_0 = W_i\varepsilon_{ox}/t_{ox}$) | $1$ |
| RC constant per unit length ($r_0c_0$) | $S^2$ |
| Local interconnection length ($l_l$) | $1/S$ |
| Local interconnection RC delay ($r_0c_0l_l^2$) | $1$ |
| Die size ($D_c$) | $S_c$ |
| Global interconnection length ($l_g$) | $S_c$ |
| **Global interconnection RC delay** ($r_0c_0l_g^2$) | $S^2S_c^2$ |



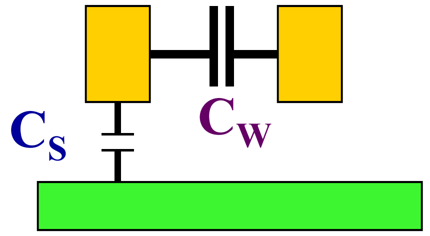
## Techniques for Higher Performance

- In very deep submicron technology, interconnect delay dominates circuit performance.
- Techniques for higher performance
  - SOI: lower gate delay.
  - Copper interconnect: lower resistance.
  - Dielectric with lower permittivity: lower capacitance.
  - **Buffering:** Insert (and size) buffers to "break" a long interconnection into shorter ones.
  - **Wire sizing**: Widen wires to reduce resistance (careful for capacitance increase).
  - **Shielding:** Add/order wires to reduce capacitive and inductive coupling.
  - **Spacing:** Widen wire spacing to reduce coupling.
  - Others: padding, track permutation, net ordering, etc.

# Interconnect Dominates Circuit Performance!!

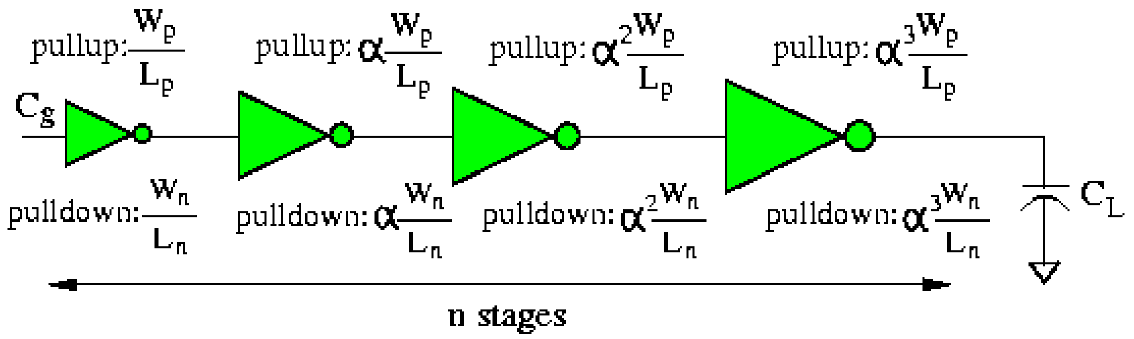


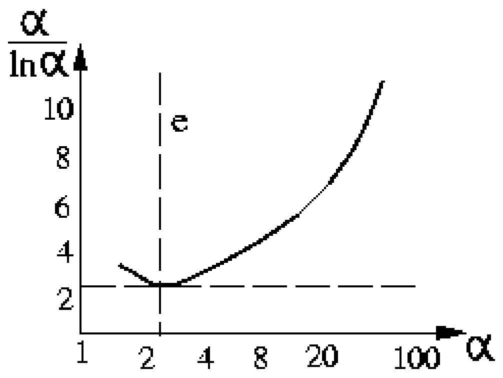
In ≦ 0.18μm wire-to-wire capacitance dominates ($C_W >> C_S$)



## Optimal Buffer Sizing w/o Considering Interconnects

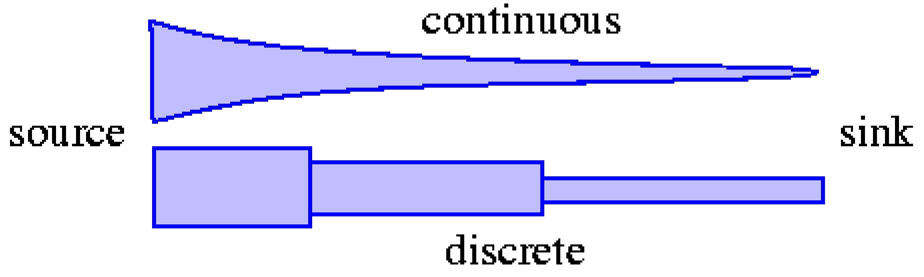
- Delay through each stage is $\alpha\, t_{min}$, where $t_{min}$ is the average delay through any inverter driving an identically sized inverter.
- $\alpha^n = C_L/C_g \Rightarrow n = \ln (C_L/C_g)/\ln \alpha$, where $C_L$ is the capacitive load and $C_g$ the capacitance of the minimum size inverter.
- Total delay $T_{tot} = n\alpha t_{min} = \frac{\alpha}{\ln \alpha} t_{min} \ln \frac{C_L}{C_g}$.
- Optimal **stage ratio:** $\frac{dT_{tot}}{d\alpha} = 0 \Rightarrow \alpha = e$.
- Optimal delay: $T_{opt} = e t_{min} \ln (C_L/C_g)$.
- Buffer sizes are exponentially tapered ($\alpha = e$).

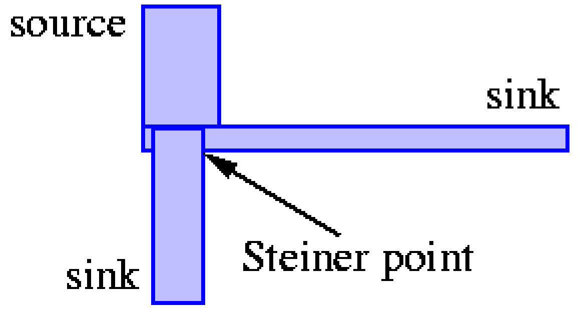

pullup: $\frac{W_p}{L_p}$
pullup: $\alpha\frac{W_p}{L_p}$
pullup: $\alpha^2\frac{W_p}{L_p}$
pullup: $\alpha^3\frac{W_p}{L_p}$
$C_g$
$C_L$
pulldown: $\frac{W_n}{L_n}$
pulldown: $\alpha\frac{W_n}{L_n}$
pulldown: $\alpha^2\frac{W_n}{L_n}$
pulldown: $\alpha^3\frac{W_n}{L_n}$
n stages



## Wire Sizing

- Wire length is determined by layout architecture, but we can choose wire width to minimize delay.
- Wire width can vary with distance from driver to adjust the resistance which drives downstream capacitance.
- Wire with minimum delay has an exponential taper.
- Can approximate optimal tapering with segments of a few widths.
- Recent research claims that buffering is more effective than wire sizing for optimizing delay, and two wire widths are sufficient for area/delay trade-off.

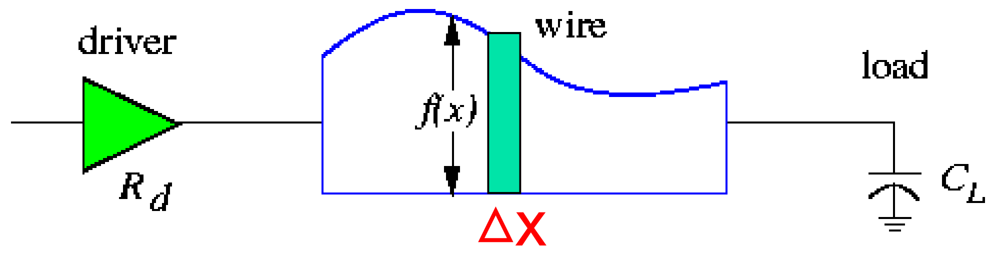


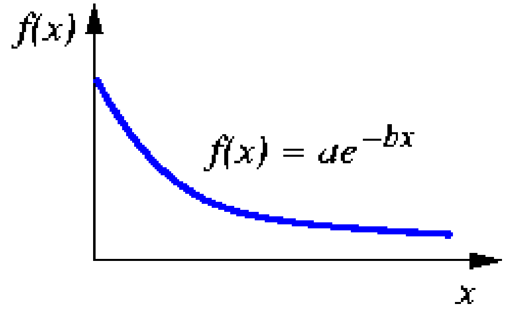


# Optimal Wire-Sizing Function

- Suppose a wire of length *L* is partitioned into *n* equal-length wire segments, each of length $\Delta x = L/n$; unit resistance and capacitance: $\hat{r}$, and $\hat{c}$.
- The respective resistance and capacitance of *i*-th wire segment can be approximated by $\hat{r}\,\Delta x / f(x_i)$ and $\hat{c}\,\Delta x\, f(x_i)$, where $f(x_i)$ is the width at position $x_i$.
- Elmore delay: $D_n = R_d\left(C_L + \sum_{i=1}^{n} \hat{c} f(x_i)\Delta x\right) + \sum_{i=1}^{n} \frac{\hat{r}\Delta x}{f(x_i)}\left(\sum_{j=i}^{n} \hat{c} f(x_j)\Delta x + C_L\right)$
- As $n \to \infty$, $D_n \to D$: $D = R_d\left(C_L + \int_0^L \hat{c} f(x)dx\right) + \int_0^L \frac{\hat{r}}{f(x)}\left(\int_x^L \hat{c} f(t)dt + C_L\right)dx$
- Optimal wire sizing function $f(x) = ae^{-bx}$, where

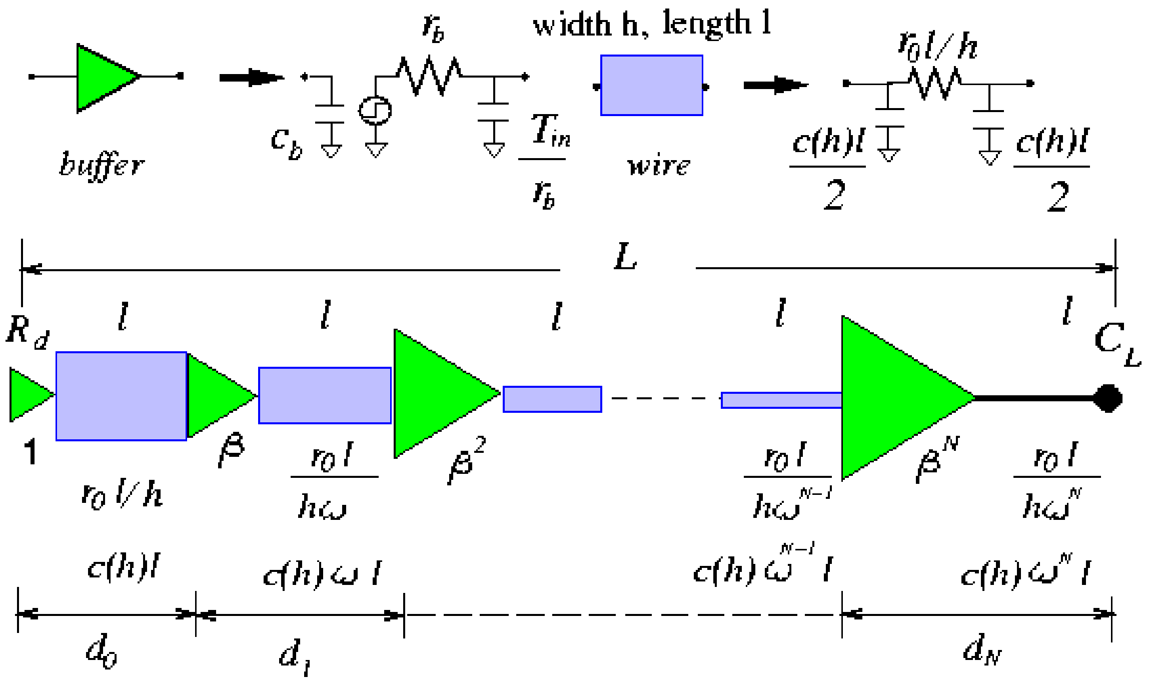
$$a = \frac{\hat{r}}{bR_d}, \quad b\sqrt{\frac{R_d C_L}{\hat{r}\hat{c}}} - e^{-bL/2} = 0.$$





# Simultaneous Wire & Buffer Sizing

- **Input:** Wire length *L*, driver resistance $R_d$, load capacitance $C_L$, unit wire area capacitance $c_0$, unit wire fringing capacitance $c_f$, unit-sized wire resistance $r_0$, unit-size capacitance of a buffer $c_b$, unit-size buffer resistance $r_b$, intrinsic buffer delay $T_{in}$, and the number of buffers *N*.
- **Objective:** Determine the stage ratio $\beta$ for buffer sizes and the stage ratio $\omega$ for wire widths such that the wire delay is minimized.

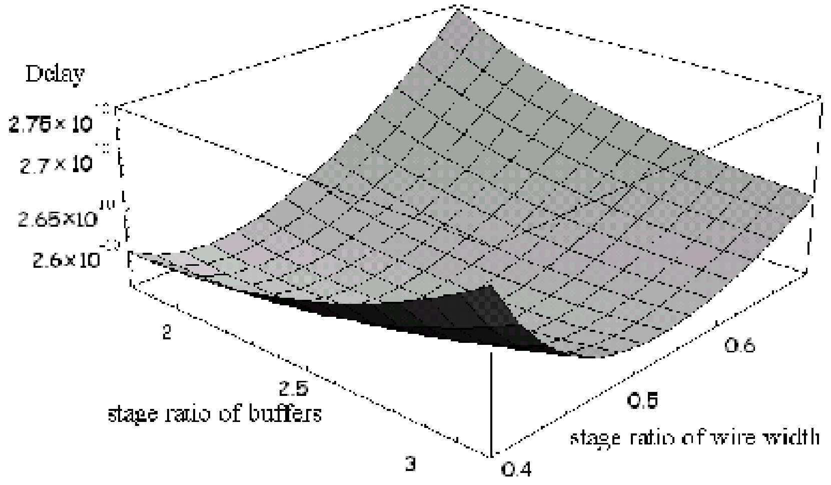


# Wire/Buffer Size Ratios for Delay Optimization

- Chang, Chang, Jiang, ISQED-2002.

$$
\begin{aligned}
D_N(\beta,\omega) \;=\;& R_d(c_0h + c_f)l + \beta R_d c_b + NT_{in} + \beta(N-1)r_bc_b + \frac{r_0 l C_L}{h\omega^N} + \frac{r_b}{\beta^N}C_L \\
+\;& \frac{1}{2}(N+1)(r_0c_0l^2 + \frac{r_0c_fl^2}{h})\sum_{i=1}^{N}\frac{r_b}{\beta^i}(c_0\omega^i hl + c_fl) + \sum_{i=1}^{N}\frac{\beta^i}{\omega^i}\frac{r_0lc_b}{h}.
\end{aligned}
$$

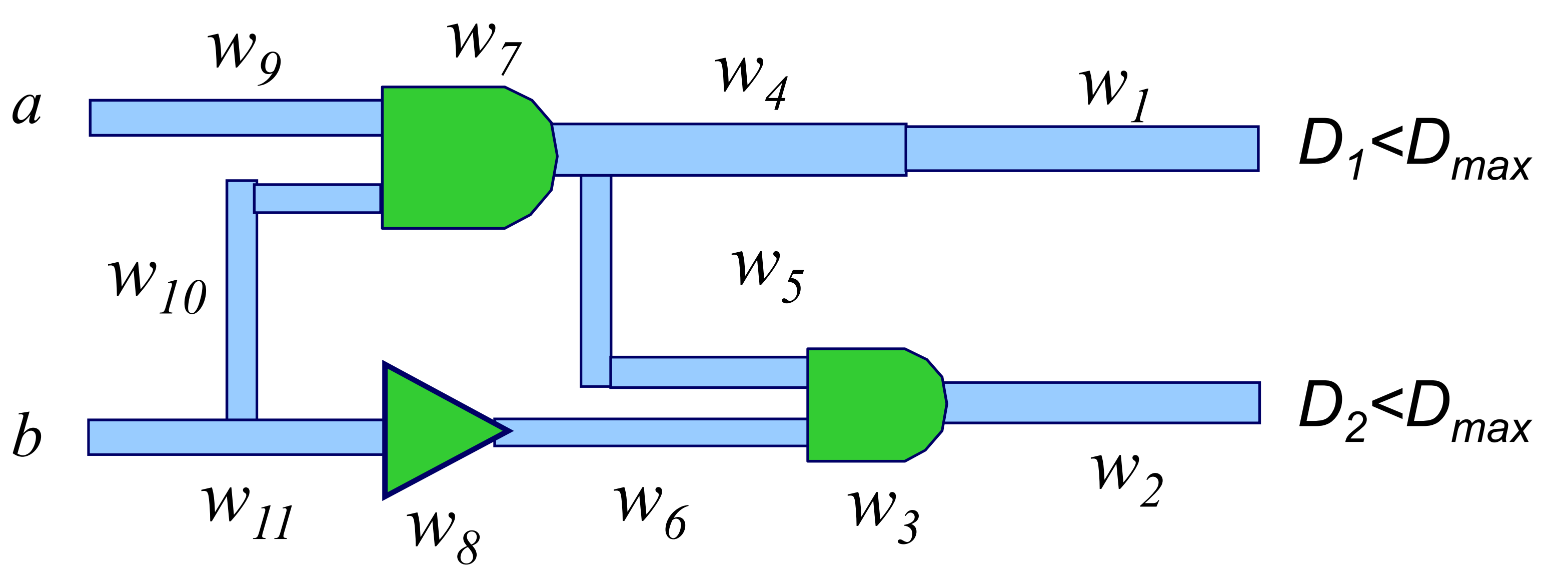
- In practice, the delay of a wire $D_N(\beta, \omega)$ is a convex function of the stage ratio $\beta$ for practical buffer sizes and the stage ratio $\omega$ for practical wire widths.
- Can apply efficient search techniques (e.g., binary search) to find the optimum ratios.



## Performance Optimization: A Sizing Problem

- Minimize the maximum delay $D_{max}$ by changing $w_1, \ldots, w_n$

$$\begin{aligned} &\textit{Minimize} \quad D_{\max} \\ &\textit{subject to} \quad D_i(\mathbf{W}) \le D_{\max},\ i = 1,\ldots,m \\ &\qquad\qquad\quad L \le w_i \le U,\ i = 1,\ldots,n \end{aligned}$$



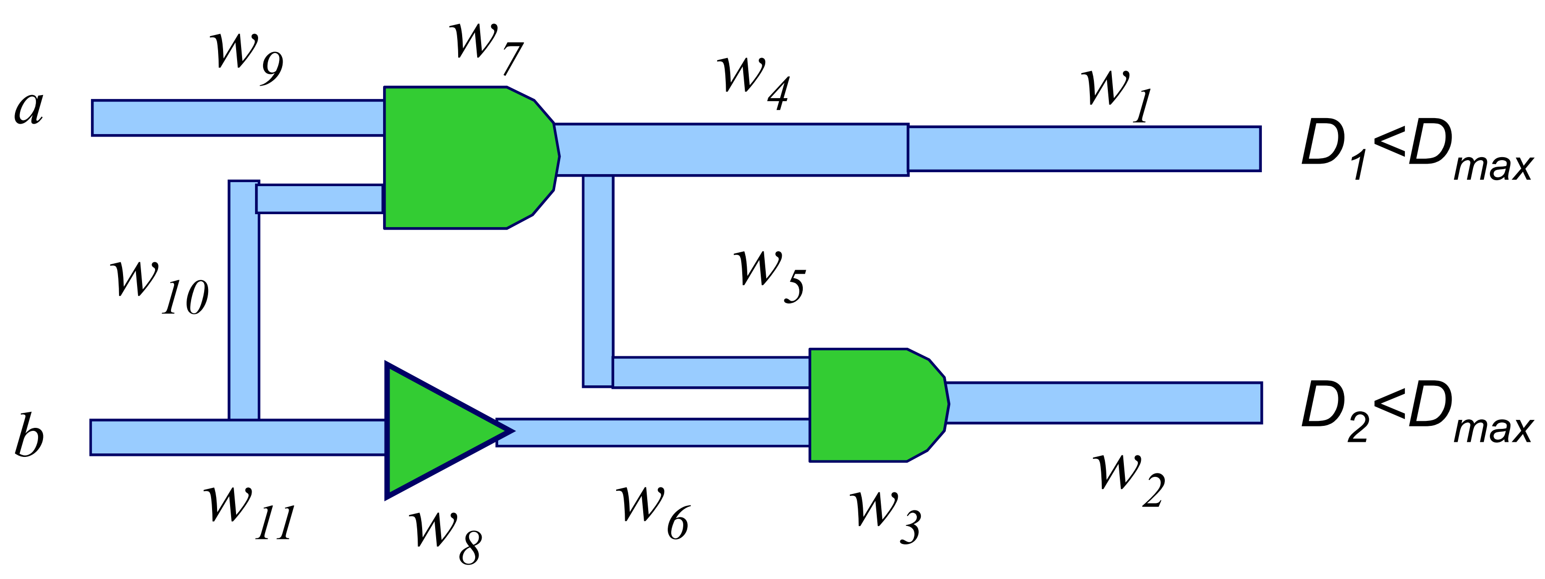
## Popular Sizing Works

- Algorithmic approaches: faster, non-optimal for general problems
  - TILOS (Fishburn, Dunlop, ICCAD-85)
  - Weighted Delay Optimization (Cong et al., ICCAD-95)
- Traditional mathematical programming: often slower, optimal
  - Geometric Programming (TILOS)
  - Augmented Lagrangian (Marple et al., 86)
  - Sequential Linear Programming (Sapatnekar et al.)
  - Interior Point Method (Sapatnekar et al., TCAD-93)
  - Sequential Quadratic Programming (Menezes et al., DAC-95)
  - Augmented Lagrangian + Adjoin Sensitivity (Visweswariah, et al., ICCAD-96, ICCAD-97)
- **Lagrangian relaxation based mathematical programming: (Chen, Chang, Wong, DAC-96; Jiang, Chang, Jou, DAC-99 [TCAD, Sept. 2000]; and many more)**
  - Fast and optimal

# TILOS: Heuristic Approach

- Finds sensitivities associated with each gate
- Up-sizes the gate with the maximum sensitivity
- Minimizes the objective function

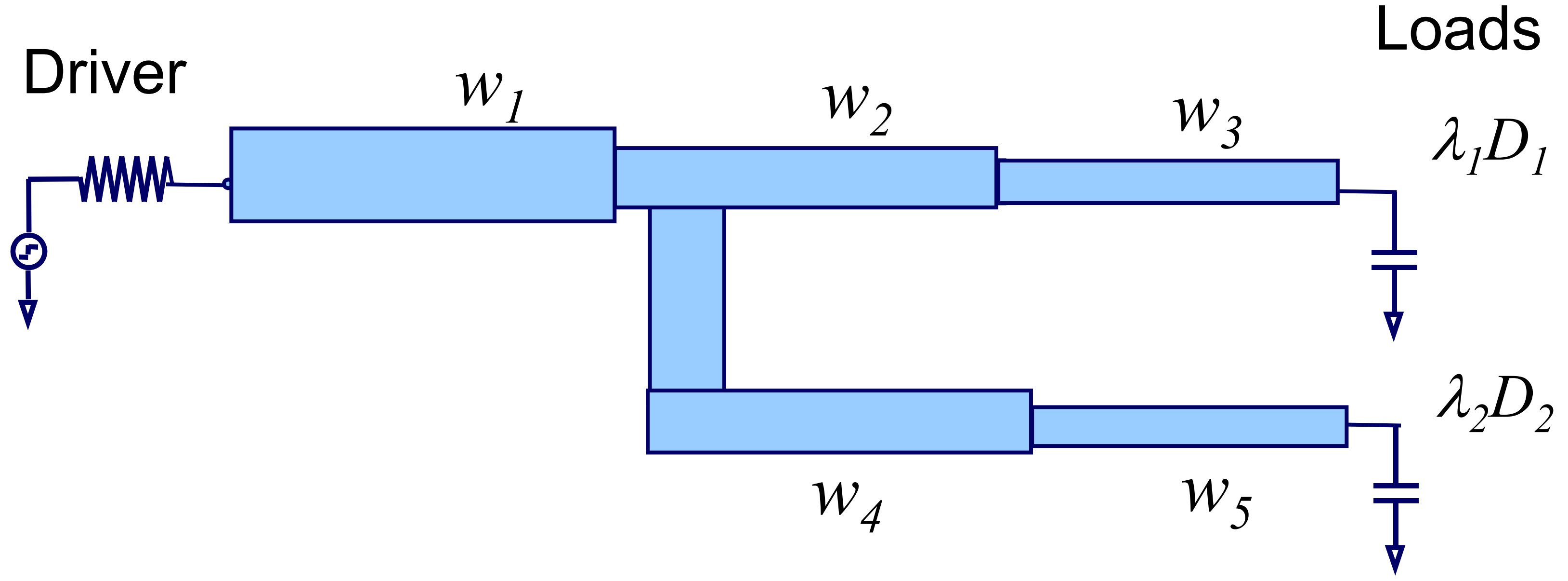
*Minimize* $D_{max}$



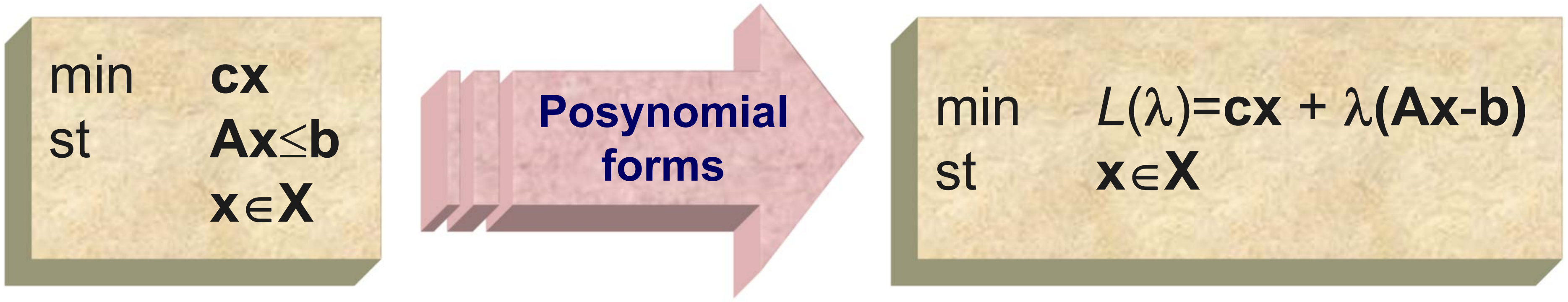
# Weighted Delay Optimization

- Cong, et. al., ICCAD-95
- Sizes one wire at a time in the DFS order
- Minimize the weighted delay
- Best weights?

$$Minimize\ \lambda_1 D_1 + \lambda_2 D_2$$



## From Mathematical Prog. to Lagrangian Relaxation



Mathematical formulation

Positive coefficient polynomials

Lagrange multipliers λ

## Mathematical Programming

- Formulation: $$\begin{aligned}&\mathit{Minimize}\quad f(x)\\&\mathit{subject\ to}\quad g_i(x) \le 0,\ i = 1..m\end{aligned}$$
- Lagrangian: $L(\boldsymbol{\lambda}) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x),\ \ \mathit{where}\ \lambda_i \ge 0$
- Optimality (Necessary) Condition (Kuhn-Tucker theorem):

$$\frac{\partial L(\boldsymbol{\lambda})}{\partial x_i} = 0 \Rightarrow \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) = 0$$

$$\lambda_i g_i(x) = 0 \text{ (Complementary Condition)}$$

$$g_i(x) \le 0, \lambda_i \ge 0 \ \text{ (Feasibility Condition)}$$

# Lagrangian Relaxation

$$\begin{aligned}&\mathit{Minimize}\quad f(x)\\ &\mathit{subject\ to}\quad g_i(x)\le 0,\quad i=1..n\\ &\qquad\qquad\quad g_i(x)\le 0,\quad i=n+1..m\end{aligned}\quad\xrightarrow{\text{LRS}}\quad\begin{aligned}&\mathit{Minimize}\quad f(x)+\sum_{i=1}^{n}\lambda_i g_i(x)\\ &\mathit{subject\ to}\quad g_i(x)\le 0,\quad i=n+1..m\end{aligned}$$

- LRS (Lagrangian Relaxation Subproblem)
- There exist Lagrangian multipliers λ that lead LRS to the optimal solution for convex programming
  - When *f(x)*, *g*$_i$(*x*)'s are all positive polynomials (posynomials)
- **The optimal solution for any LRS is a lower bound of the original problem**

# Lagrangian Relaxation

$$\text{Minimize} \quad D_{\max}$$

$$\text{subject to} \quad \cancel{D_i(\mathbf{W}) \le D_{\max}}, \quad i = 1..m$$

$$L \le w_i \le U, \; i = 1..n$$

Lagrangian Relaxation

$$\text{Minimize} \quad \cancel{D_{\max}} + \sum_{i=1}^{m} \lambda_i (D_i(\mathbf{W}) \cancel{- D_{\max}}) \quad L_\lambda$$

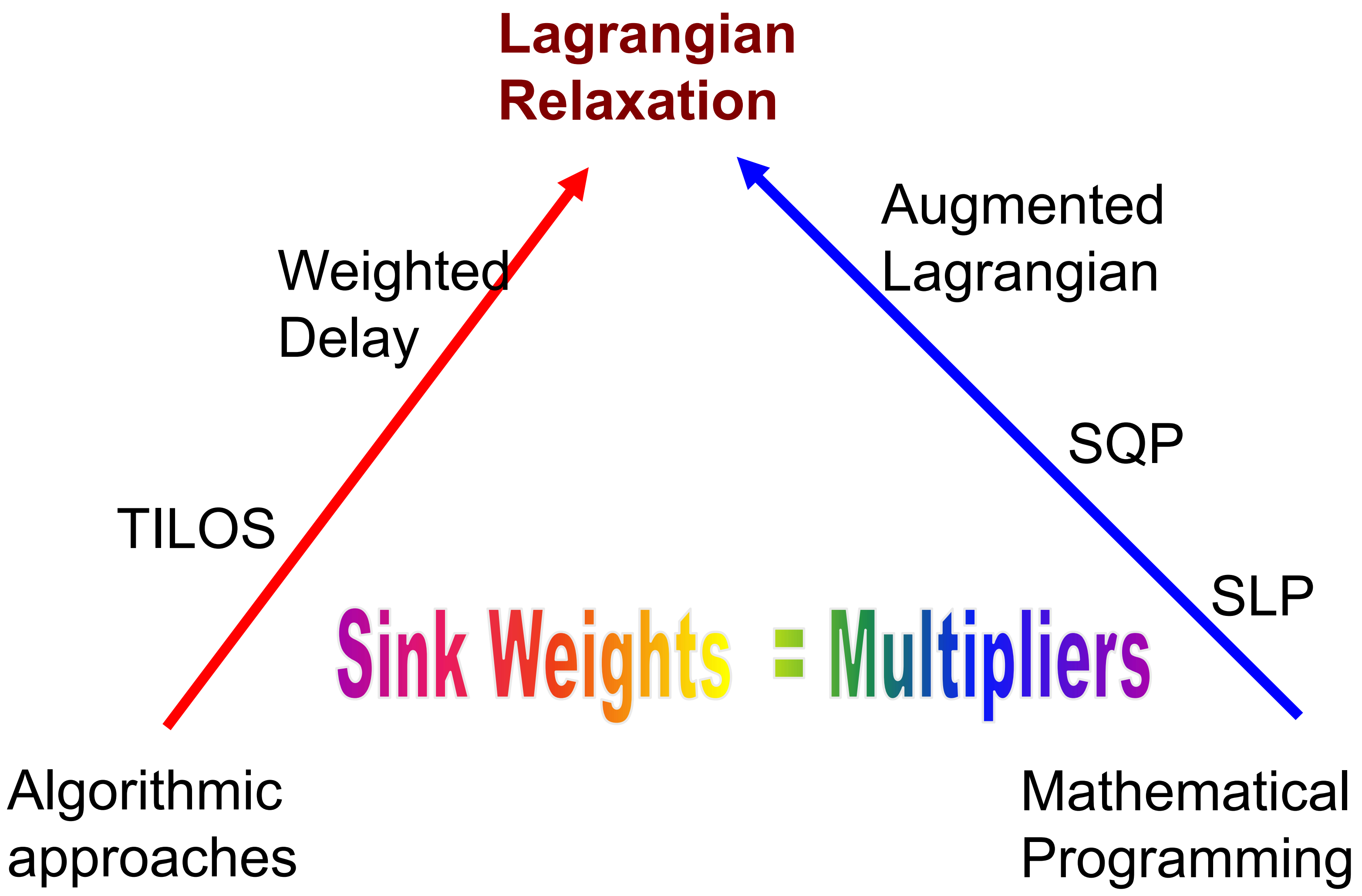
$$\text{subject to} \quad L \le w_i \le U, i = 1..n$$

$$\text{By} \quad \frac{\partial L_\lambda}{\partial D_{\max}} = 0, \quad \text{we have} \sum_{i=1}^{m} \lambda_i = 1$$

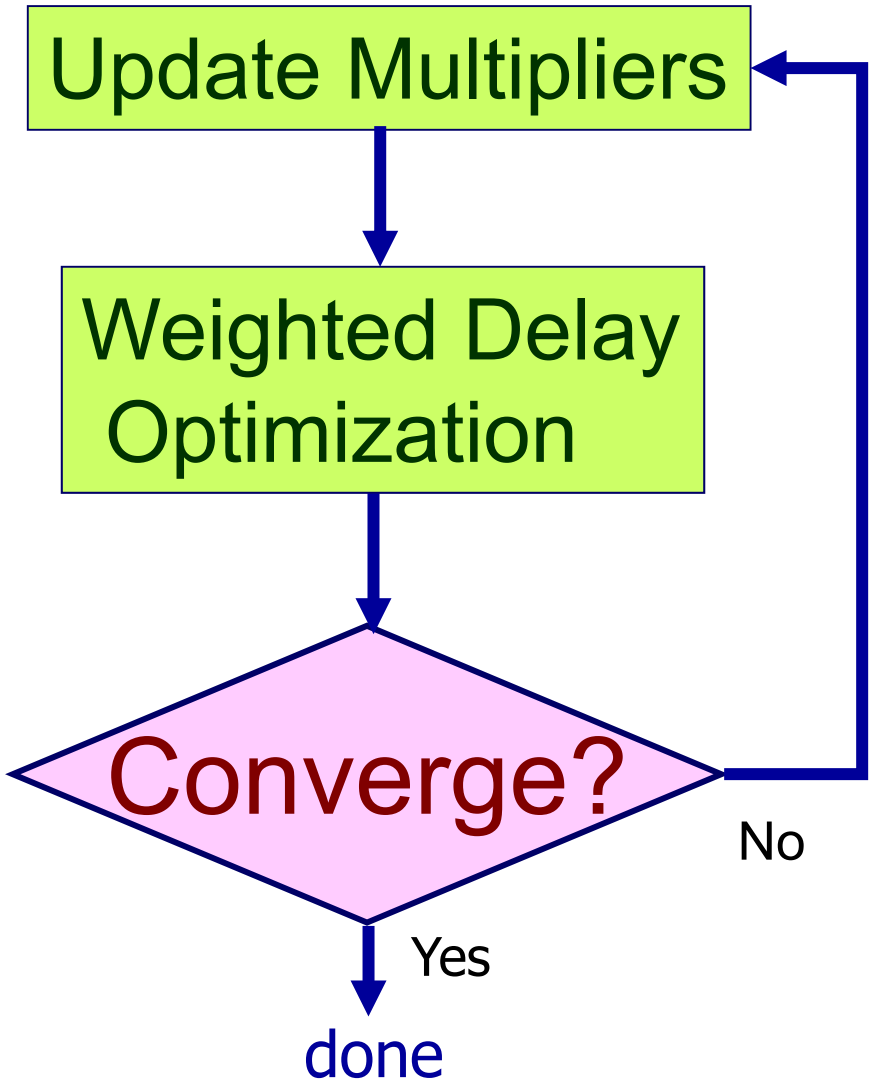
$$\text{Minimize} \quad \sum_{i=1}^{m} \lambda_i D_i(\mathbf{W})$$

$$\text{subject to} \quad L \le w_i \le U, i = 1..n$$

# Lagrangian Relaxation

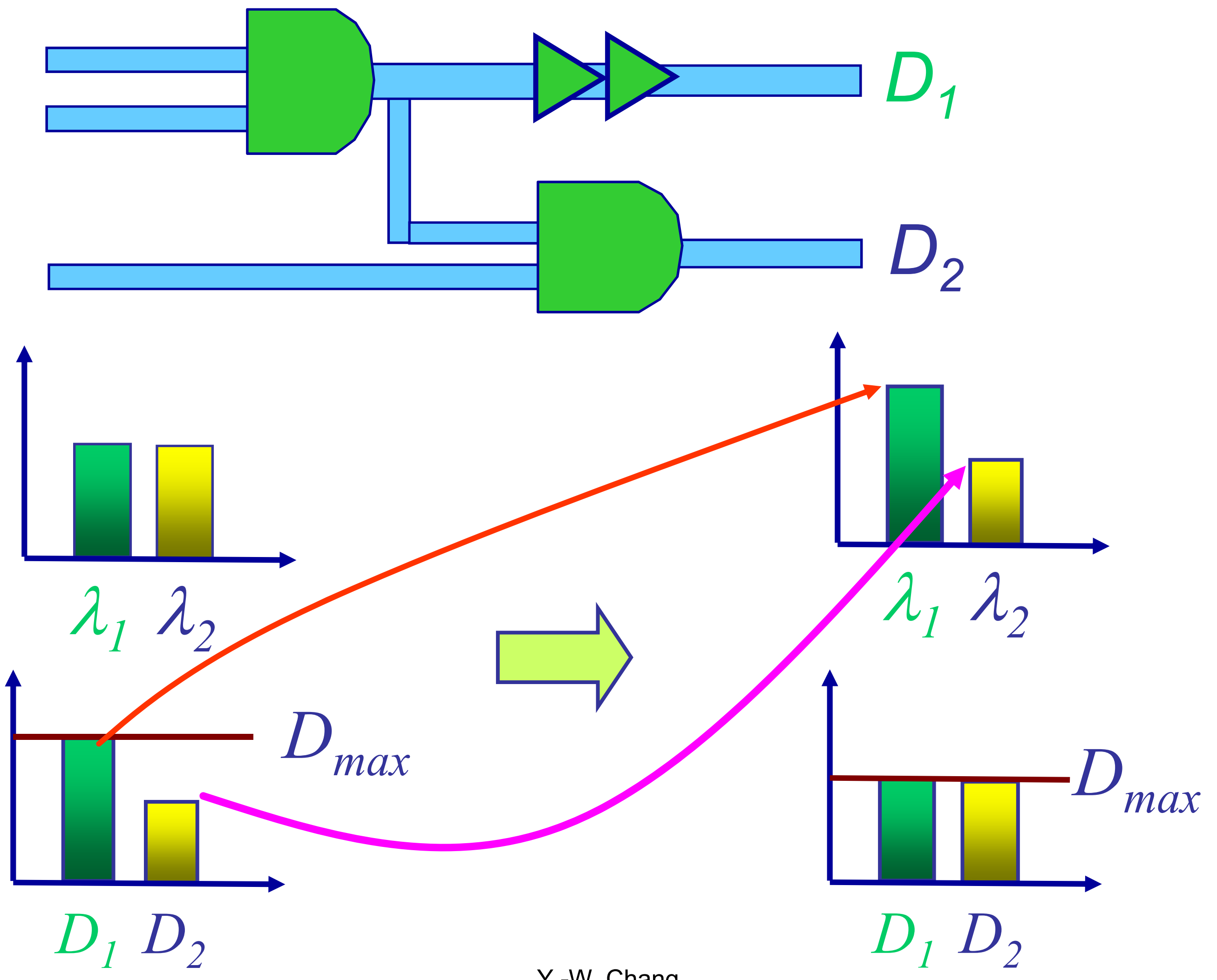


## Lagrangian Relaxation Framework



## Lagrangian Relaxation Framework

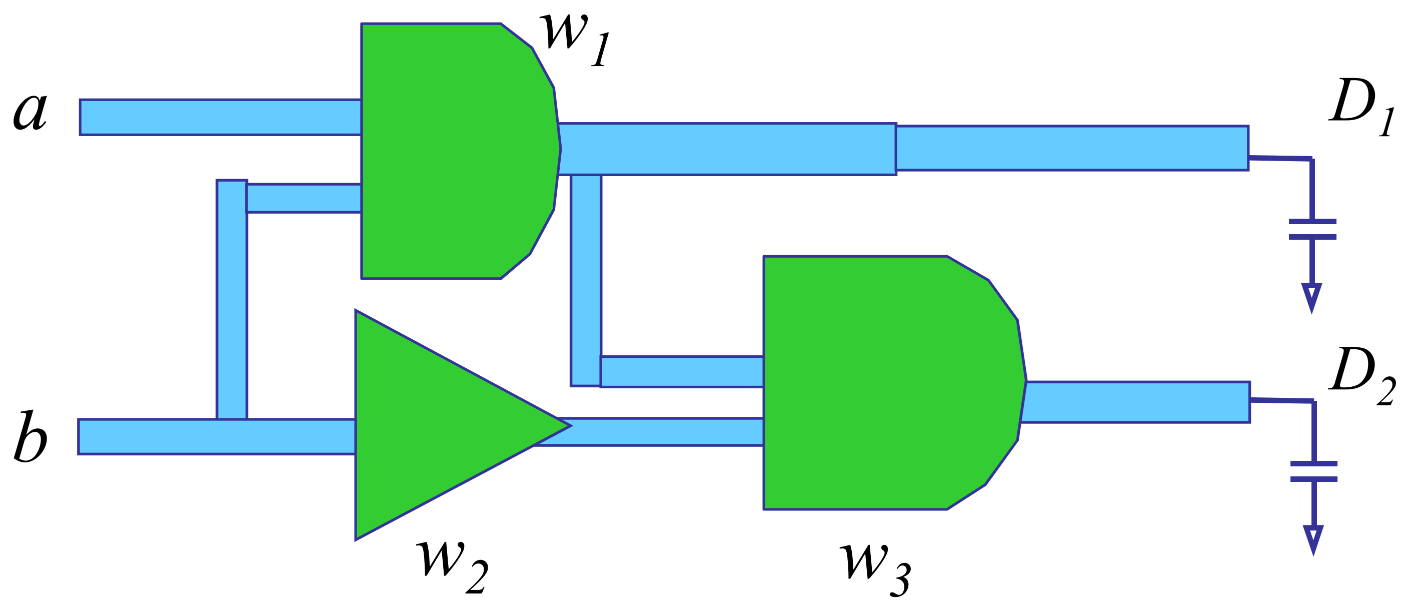
More Critical -> More Resource -> Larger Weight



## Weighted Minimization

- Traverse the circuit in the topological order
- Resize each component to minimize Lagrangian during visit

$$Minimize\ \lambda_1 D_1 + \lambda_2 D_2$$



## Multiplier Adjustment: A Subgradient Approach

$$\text{Step 1}: \lambda_i^{new} = \lambda_i^{old} + \theta_k (D_i - D_{\max}),$$

$$where \ \lim_{k\to\infty} \theta_k \to 0, \sum_{k=1}^{\infty} \theta_k \to \infty$$

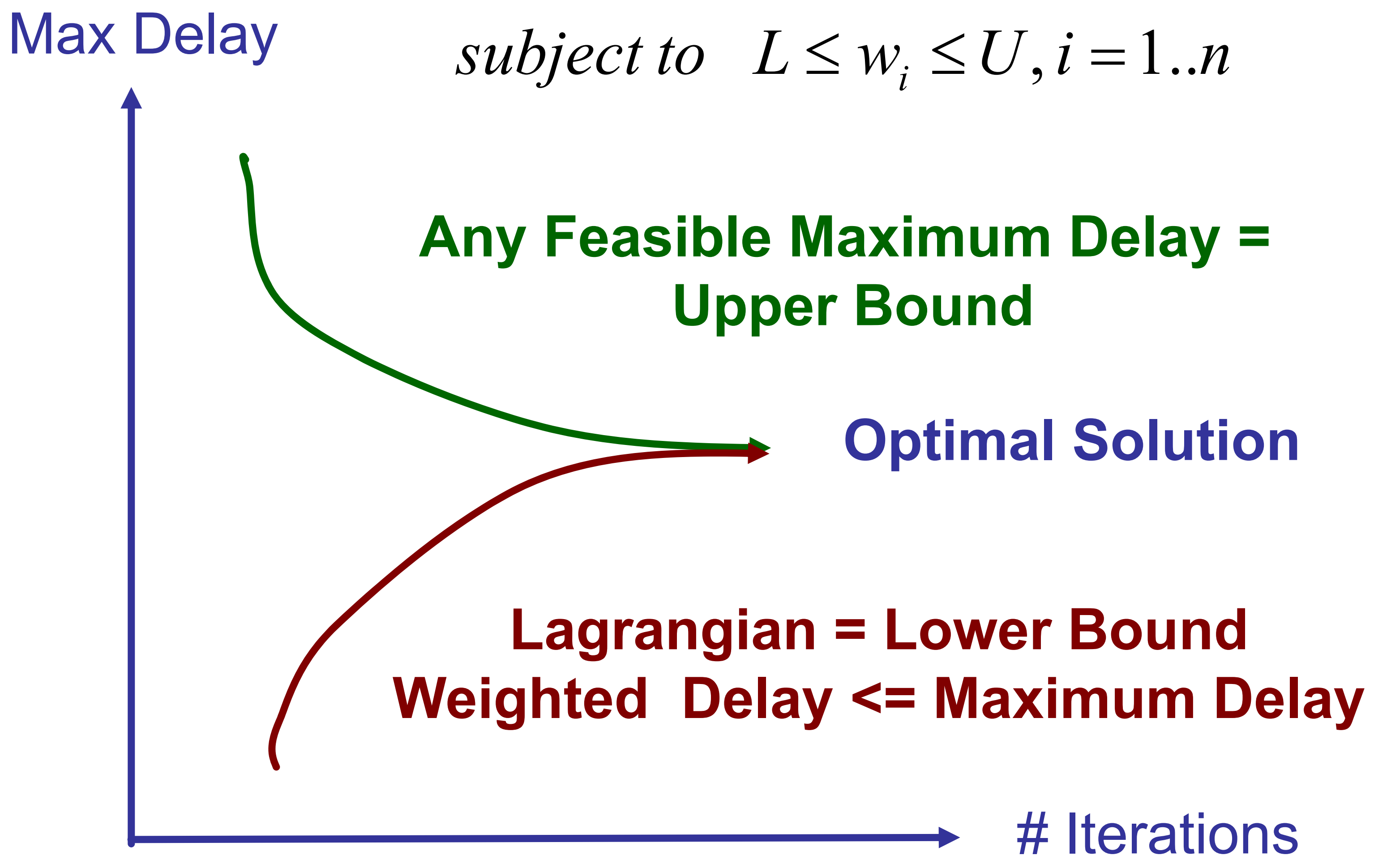
$$\text{Step 2}: \text{Project } \boldsymbol{\lambda} \text{ to the nearest feasible solution}$$

- Subgradient: An extension definition of gradient for non-smooth functions.
- Experience: Simple heuristic implementation can achieve a very good convergence rate.

## Convergence Sequence

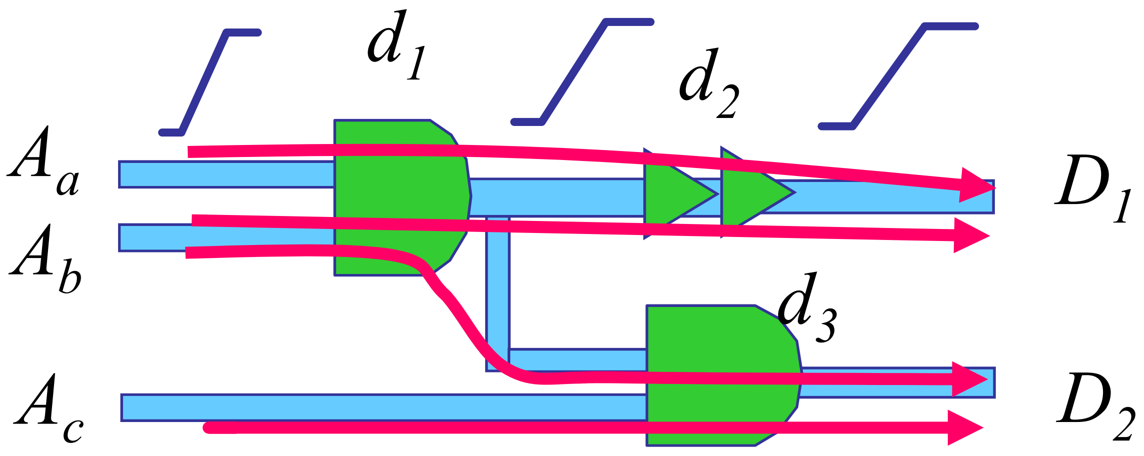
$$\textit{Minimize} \quad \sum_{i=1}^{m} \lambda_i D_i(\mathbf{w})$$

$$\textit{subject to} \quad L \leq w_i \leq U, i = 1..n$$



Max Delay
Any Feasible Maximum Delay = Upper Bound
Optimal Solution
Lagrangian = Lower Bound
Weighted Delay <= Maximum Delay
# Iterations

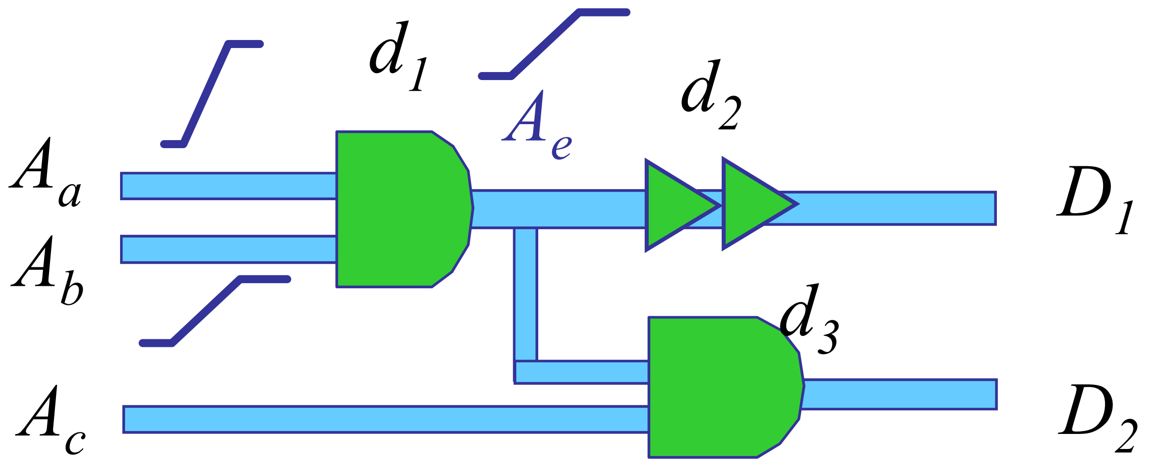
# Path Delay Formulation



$$A_a + d_1 + d_2 \le D_1$$
$$A_b + d_1 + d_2 \le D_1$$
$$A_b + d_1 + d_3 \le D_2$$
$$A_c + d_3 \le D_2$$

- Exponential growth
- More accurate
- Can exclude false paths

# Stage Delay Formulation

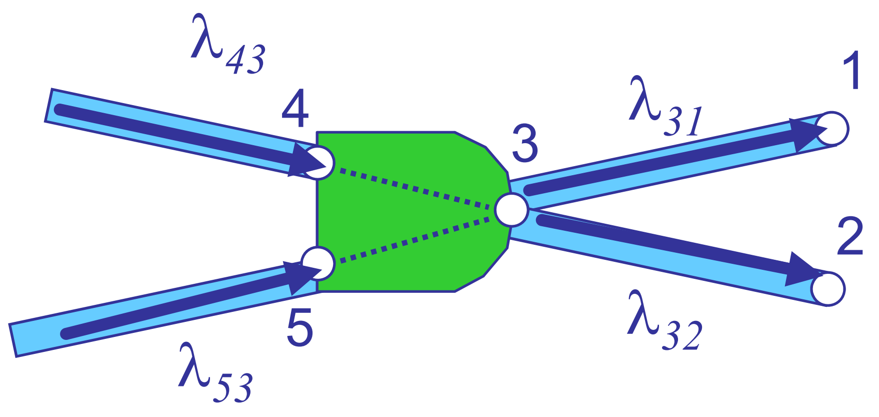


$$A_a + d_1 \le A_e$$
$$A_b + d_1 \le A_e$$
$$A_e + d_2 \le D_1$$
$$A_e + d_3 \le D_2$$
$$A_c + d_3 \le D_2$$

- Polynomial size
- Less accurate
- Contains false paths

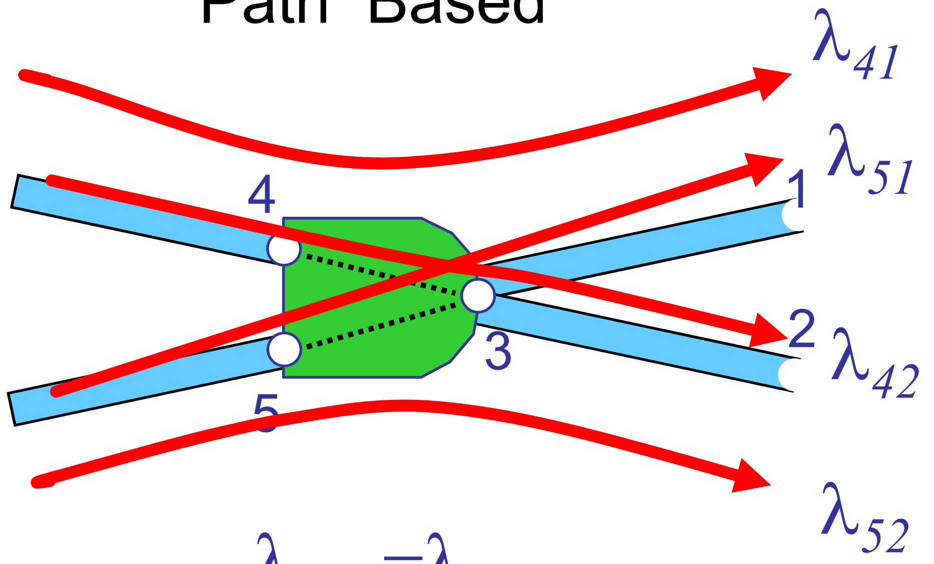
# Both Multipliers Satisfy KCL (Flow Conservation)

Stage Based



$$\lambda_{43}+\lambda_{53}=\lambda_{31}+\lambda_{32}$$

Path Based



$$\lambda_{3,in}=\lambda_{3,out}$$

$$\sum_{j\in input(i)}\lambda_{ji}=\sum_{k\in output(i)}\lambda_{ik}\ \forall i$$

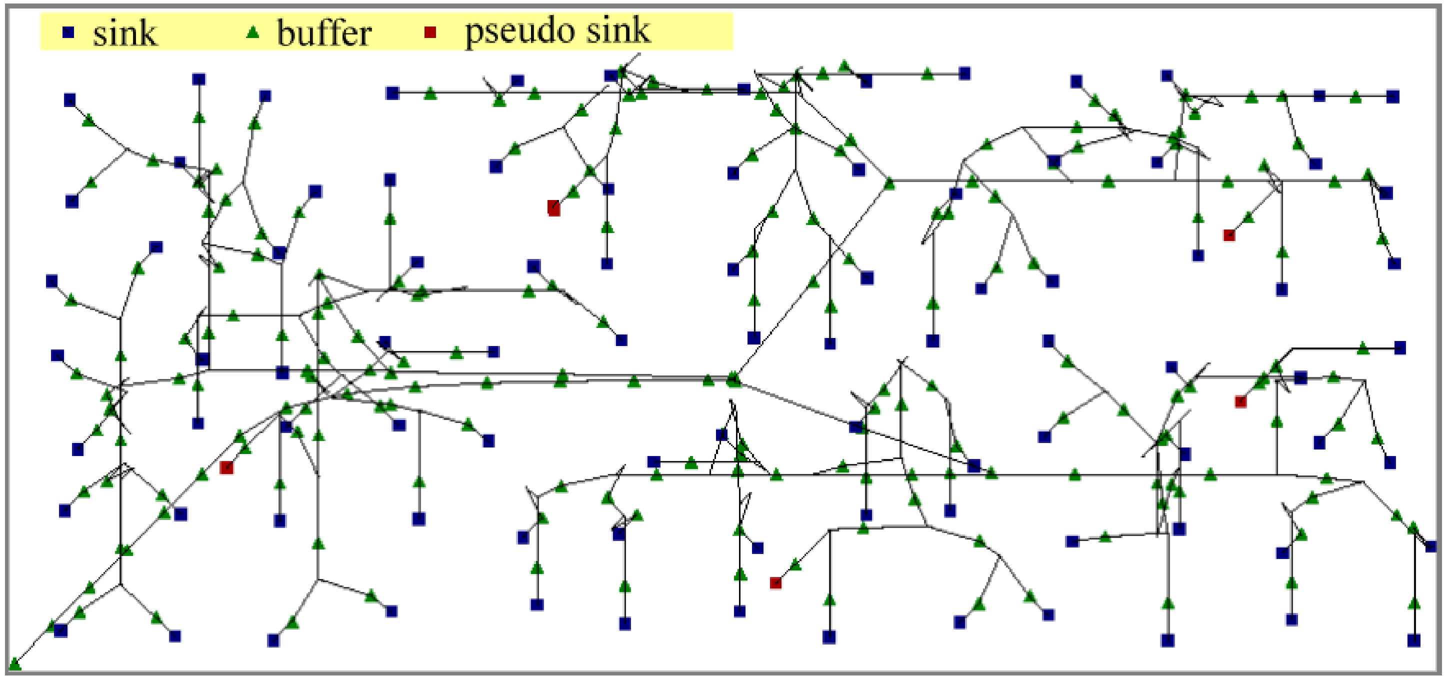
$$\sum_{j\in input(i)}\lambda_{ji}=\sum_{k\in output(i)}\lambda_{ik}\ \forall i$$

# Lagrangian Relaxation Based EDA

- Boolean optimization: Manquinho &  Marques-Silva, DATE-05
- Floorplanning: Young et al., ISPD-2K; Lee, Chang, Hsu & Yang, DAC-03; Yan & Chu, TCAD-10.
- Buffer block planning: Jiang et al., ASP-DAC-03
- Placement: Kim & Markov: DAC-12; Wu et al., ICCAD-14; Wu & Chu, TCAD-17
- Routing: Zhou & Wong, TCAD-99; Lee & Wong, ISPD-02; Ozdal & Wong, ICCAD-07; Hoo, Kumar & Ha, FPL-15.
- Gate & wire sizing & Vth assignment: Chen, Chu & Wong, TCAD-99; Flach,  Reimann, Posser, TCAD-14
- DFM: Huang & Wong, DAC-04 (OPC); Li, et al., TCAD-19 (split manufacturing)
- SSTA: Ghosh, et al., TCAD-07; Datta, et al., TVLSI-08; Gupta & Ranganathan, TVLSI-10; Ramprasath, et al. DAC-15
- Time-division multiplexing optimization for multi-FPGA systems: Pui & Young , ICCAD-19/TODAES-20

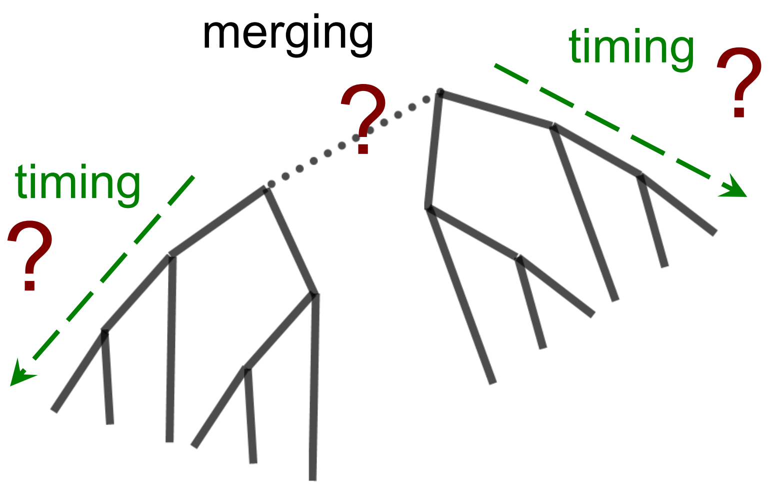
# Appendix A:

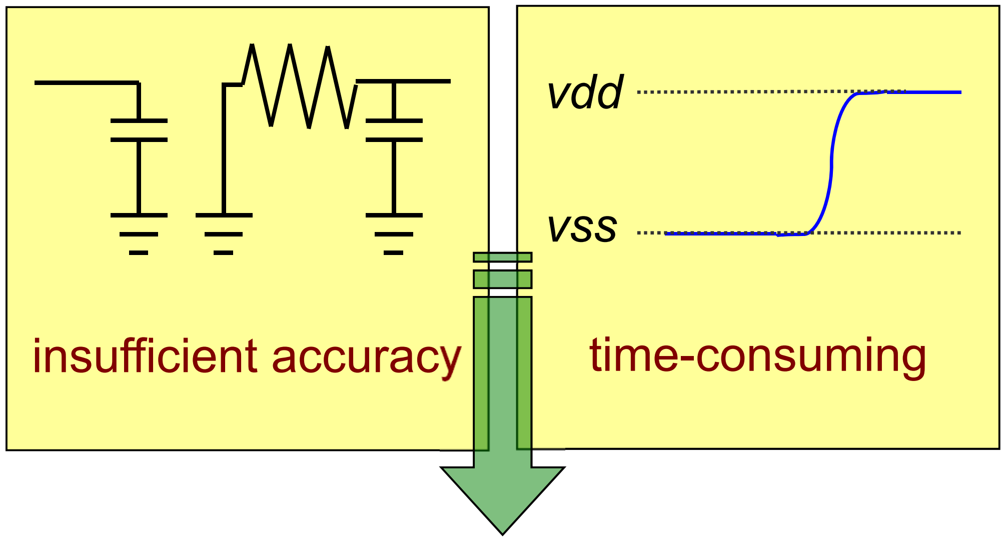
## Shih and Chang
## "Fast timing-model independent clock-tree synthesis"
## DAC-10, TCAD-12



# Introduction

- Skew-minimized buffered clock-tree synthesis plays an important role in VLSI designs for synchronous circuits
- Due to the insufficient accuracy of timing models, embedding simulation into synthesis becomes inevitable
- Runtime becomes prohibitively huge as design complexity grows

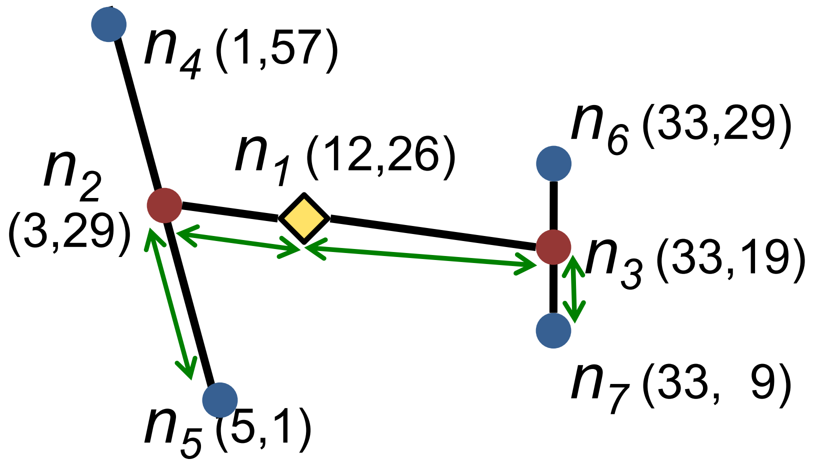




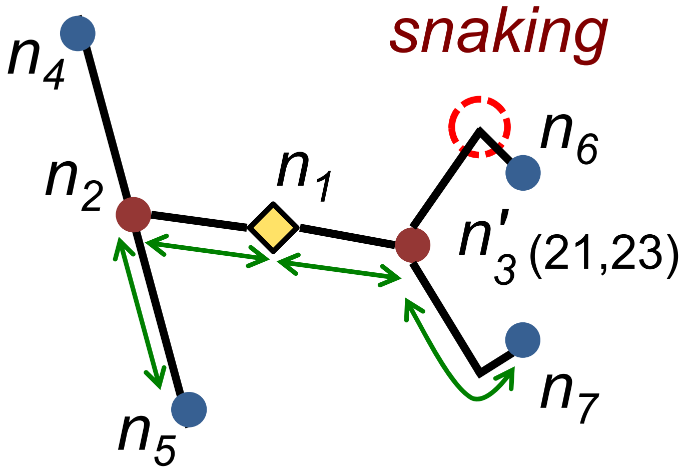
solution?

# Symmetrical Structure

- Skew is minimized by structural optimization
- Buffering and wiring of all paths are almost the same
  - Is timing-model independent
  - Do not need simulation information



0ps skew (Elmore delay)
0.123ps skew (simulation)



0ps skew (Elmore delay)
0ps skew (simulation)

## Problem Formulation

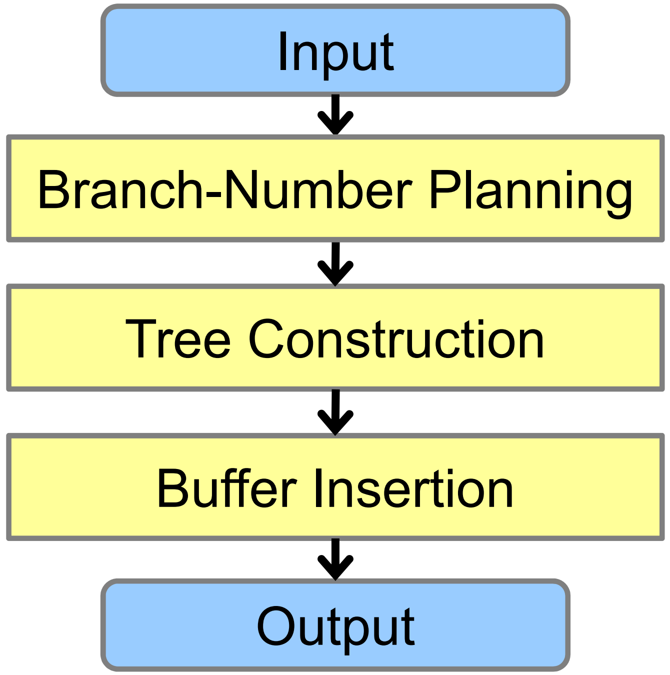
- Problem: Buffered Clock-Tree Synthesis (BCTS)

- Instance
  - Given a set of clock sinks, a slew-rate constraint, and a library of buffers

- Question
  - Construct a buffered clock tree to minimize its *skew*, subject to no slew-rate violation

## Symmetrical Clock Tree Synthesis

- Specification
  - *Number of branches, wirelength* and *inserted buffers* are the same at each level
- Flow



- Assign specific branch numbers to each tree level
- Cluster sub-trees level by level bottom-up
- Lengthen shorter connection by snaking
- Insert identical buffers along trees

## Branch-Number Planning

- Observation
  - Total branch number of some level equals the number of preceding level times its branch number
  - The multiplication sequence forms a *factorization*

$$n = f_1 \times f_2 \times \ldots \times f_q, \qquad f_i \le f_{i+1}, \forall i < q$$

***prime***

***total number of primes***

- Planning
  - Branch-Number Plan (BNP) is arranged in non-increasing order

$$B(n) = < b_1, b_2, \ldots, b_m > = < f_q, f_{q-1}, \ldots, f_1 >$$

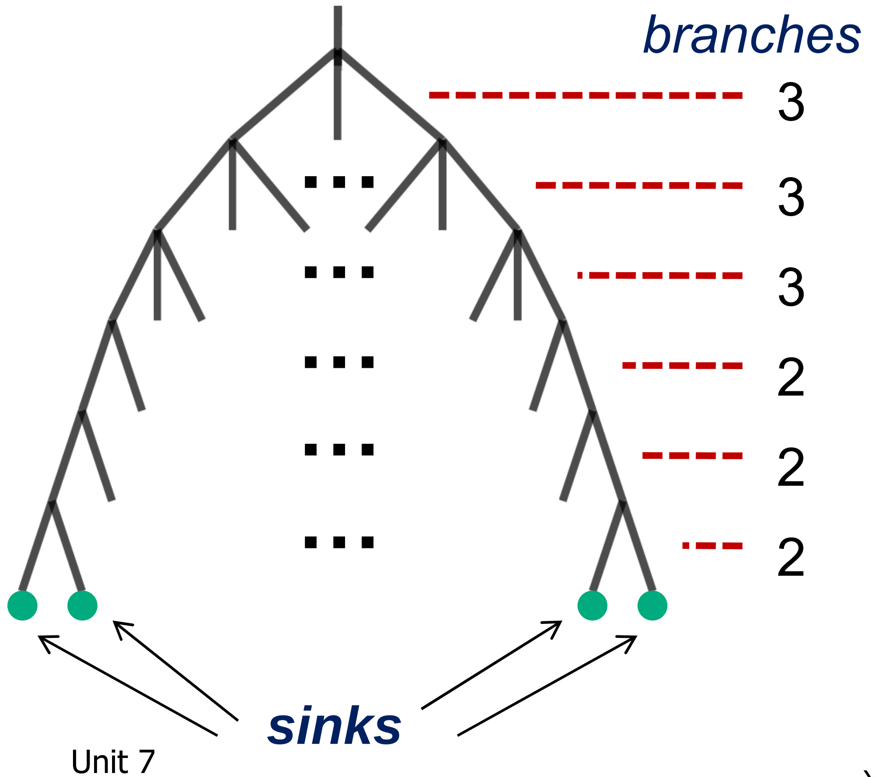
***level-1 branch number***

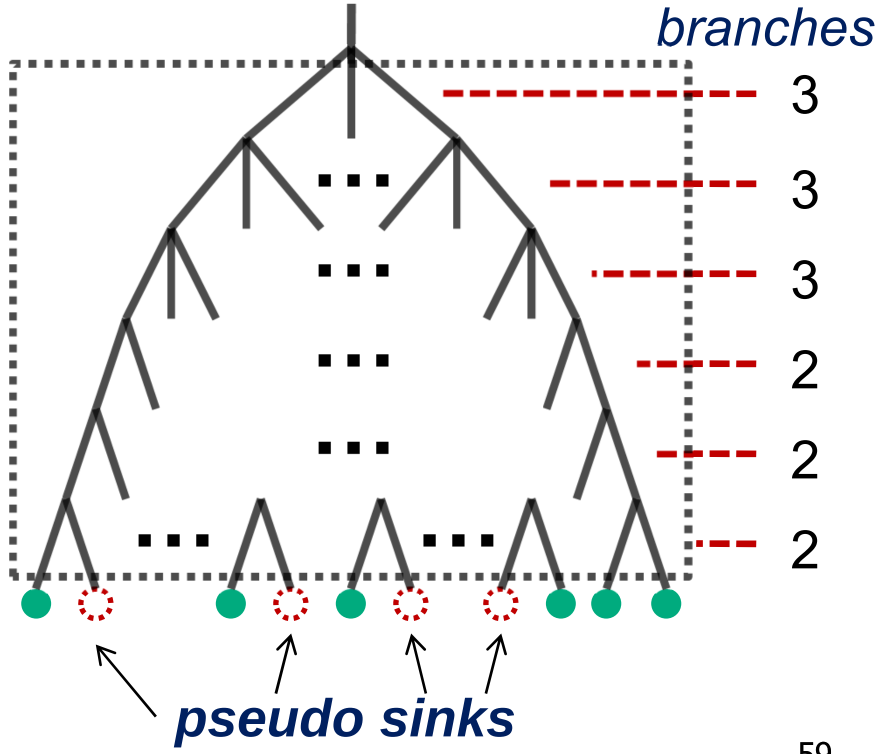
# Branch-Number Planning

- Factorization may result in a big branch number, implying a large fan-out size that could not be driven
- Pseudo sinks are added to increase the total sink number until all branch numbers are feasible

**BNP = *B*(216) = < 3, 3, 3, 2, 2, 2 >**

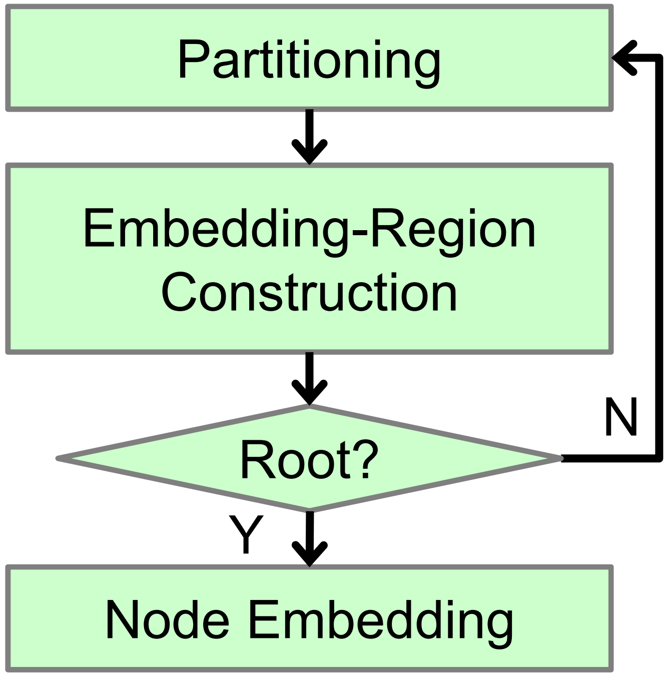
**BNP = *B*(212+4) = < 3, 3, 3, 2, 2, 2 >**





## Tree Construction

- Achieve identical wirelength in this stage
  - Cluster sub-trees level by level bottom-up
  - Lengthen shorter connection by snaking
- Flow

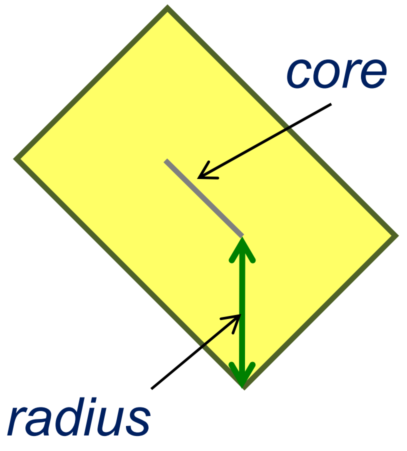


- Divide sub-trees into desired clusters
- Apply a common connection length to each cluster, and locate potential embedding positions to which snaked wires can reach
- Repeat the two stages till the embedding region of the root is built
- Find exact physical locations for nodes and route wires top-down

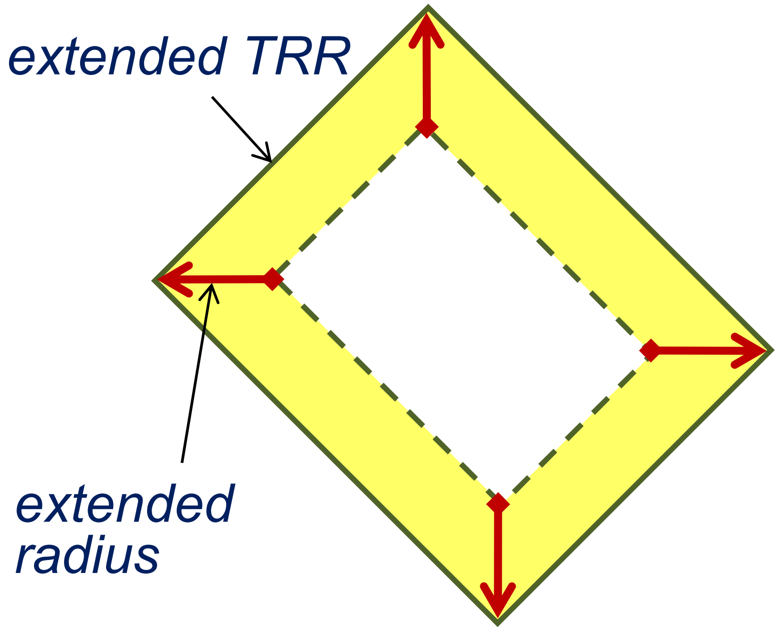
# Tilted Rectangular Region (TRR)

- Represents potential embedding positions (embedding region)
- Is a 45- or 135-degree rectangular region
  - core: a 45- or 135-degree line segment
  - radius: the Manhattan distances from the core to the region boundaries

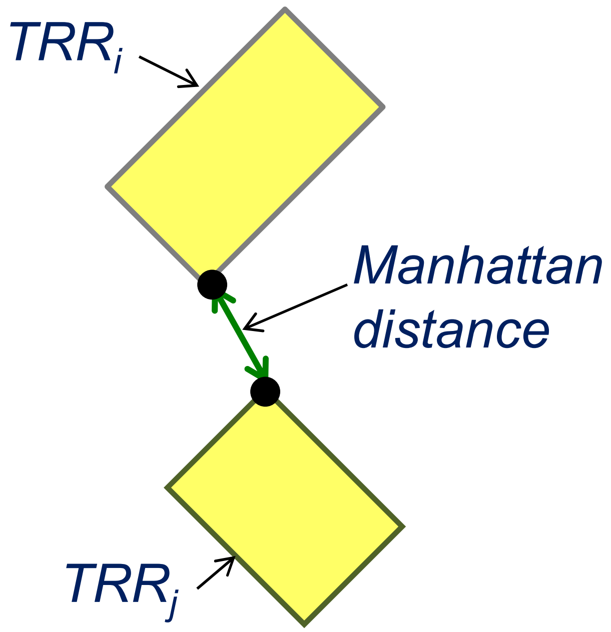
Configuration



Operation

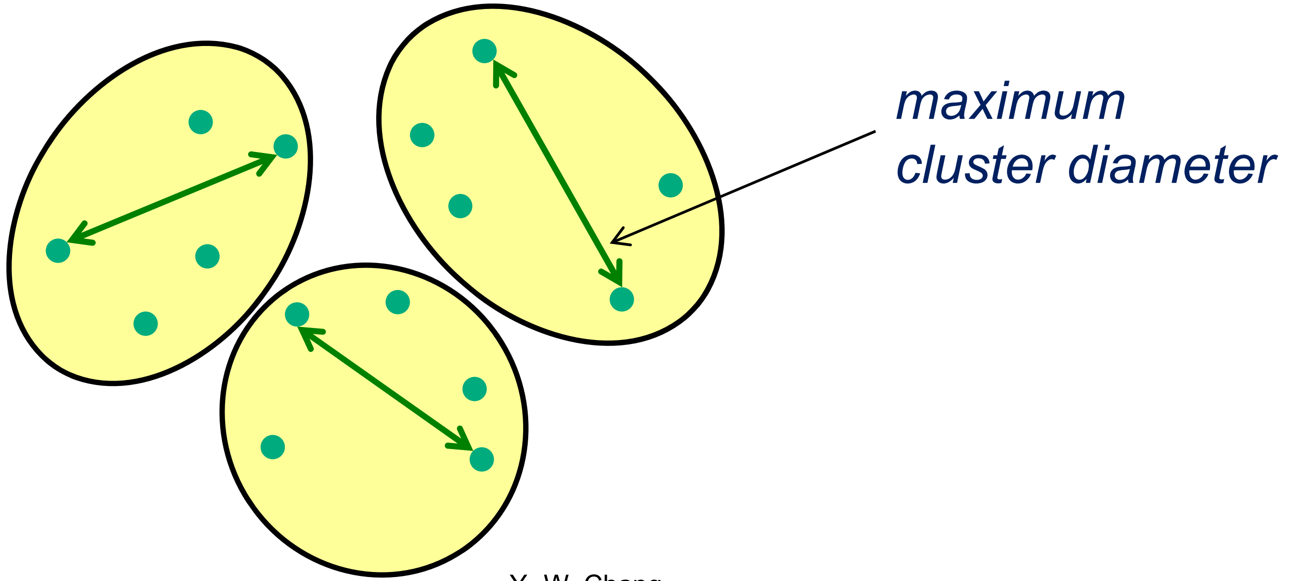


Definition



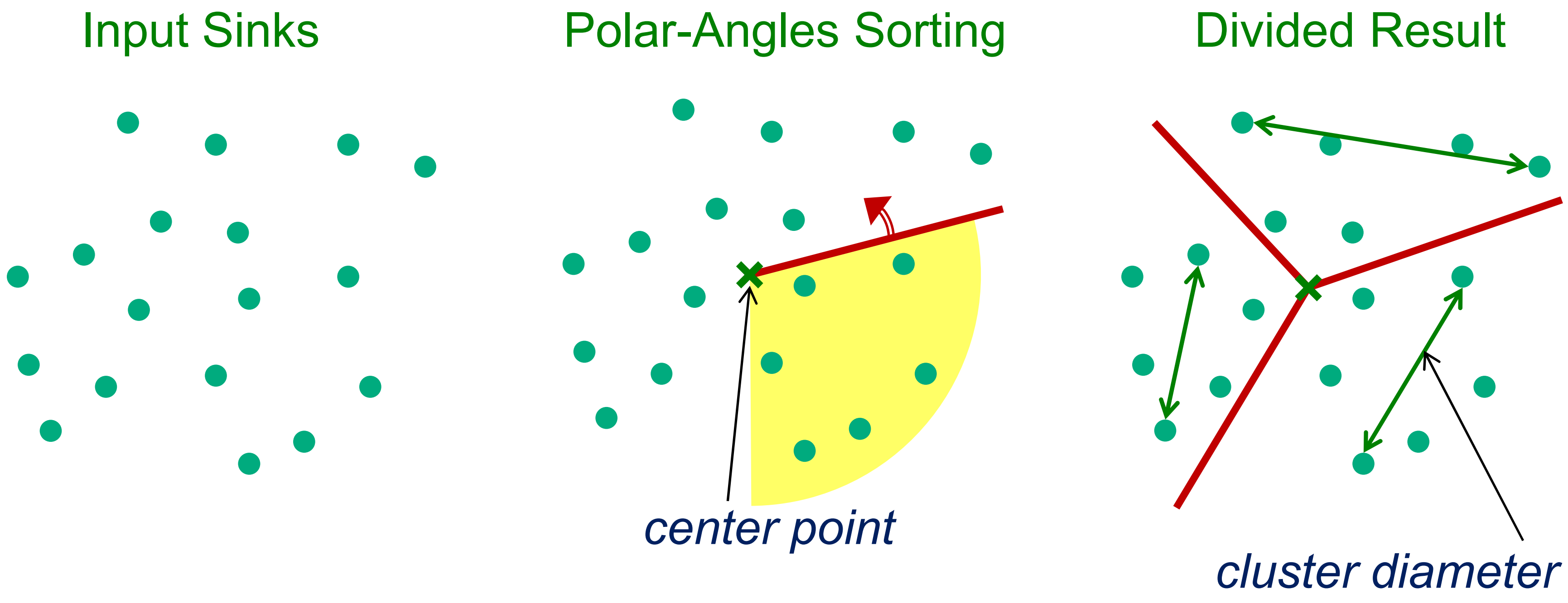
## Partitioning

- The objective is to minimize cluster diameter
  - *Cluster diameter*: the maximum distance among sub-trees within the same cluster
  - Maximum cluster diameter is the upper bound of the common connection length
- Sub-trees are divided recursively along the BNP in a top-down manner
- Non-binary tree can also be handled by this technique



## Dividing: Cake Cutting

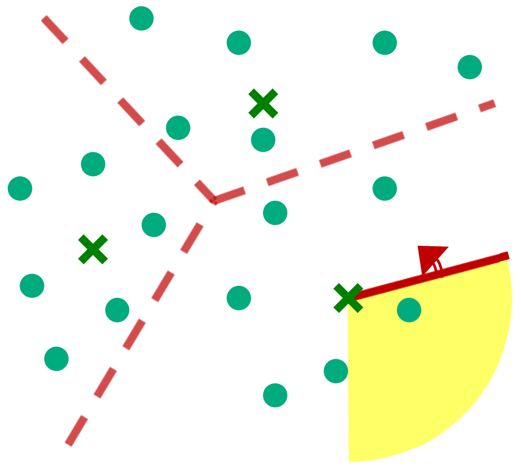
- Borrow the idea of cake cutting, i.e., slicing a cake into pieces from the center of the cake
- Sort the polar angles of sub-trees relative to the geometric center of the cluster
- Apply dynamic programming to find the minimum cluster diameter by restricting the dividing on this sorted order



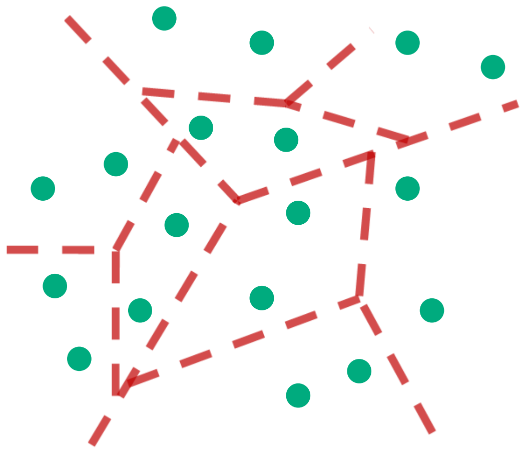
# Recursive Dividing

- For *i*-th level partitioning along the given BNP $<b_1, b_2, \ldots, b_q>$, dividing is performed recursively until $b_1 \times b_2 \times \ldots \times b_{i-1}$ clusters are derived
- Desired cluster diameter could be obtained since global sub-tree distribution is considered throughout the whole process

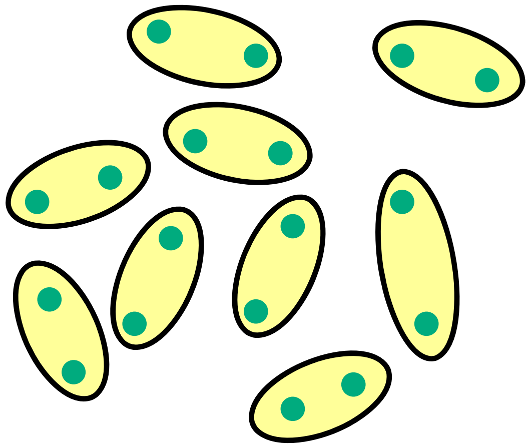
Recursive Dividing



Final Divided Result



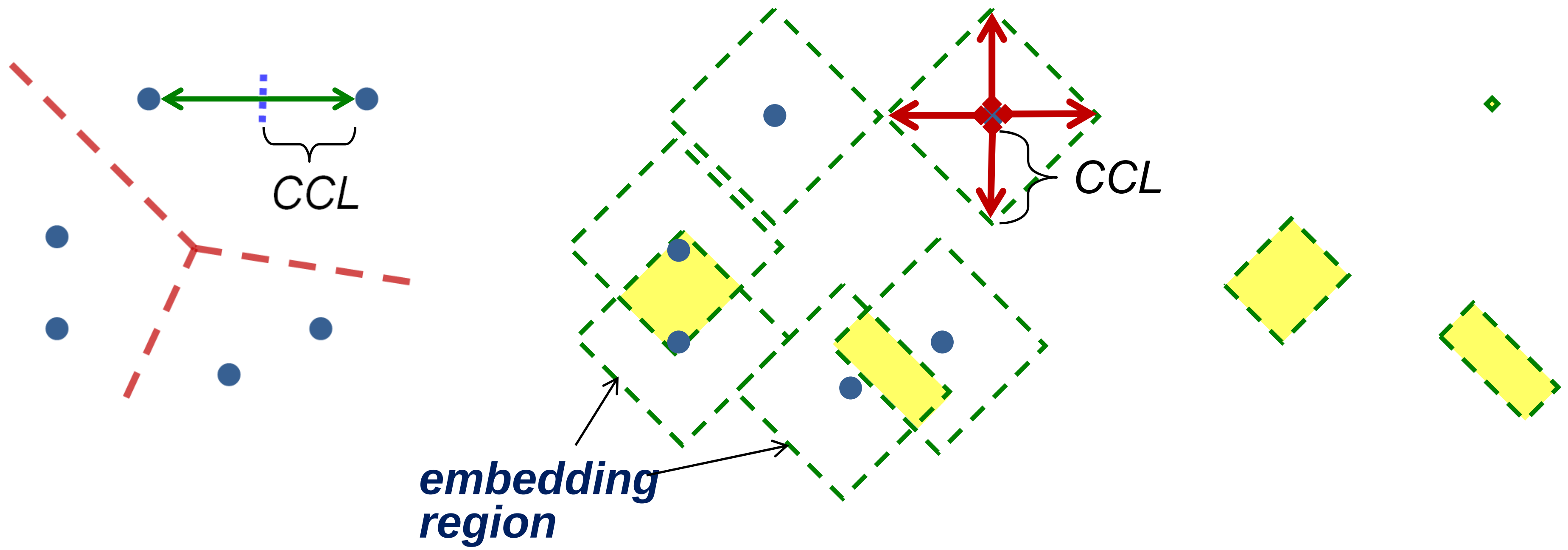
Corresponding Clusters



# Embedding-Region Construction

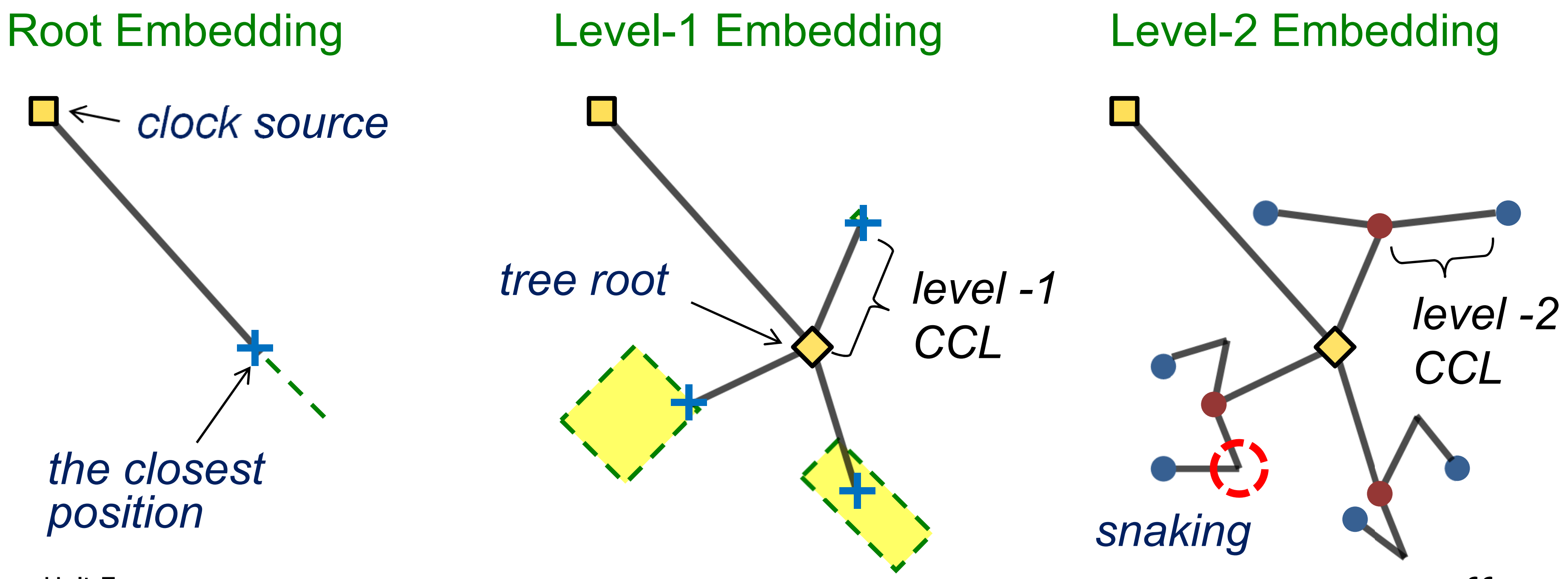
- Assign the common connection length (CCL) as the half length of the maximum cluster diameter
- Extend the TRRs of children nodes and make intersection to construct the embedding region of their parents

Given Divided Result    Region Extension/Intersection    Resulting Regions



# Node Embedding

- Set the tree root as the closest position of the embedding region w.r.t. the clock source
- Propagate embedding information level by level top-down
- Perform snaking to meet the uniform length, if necessary

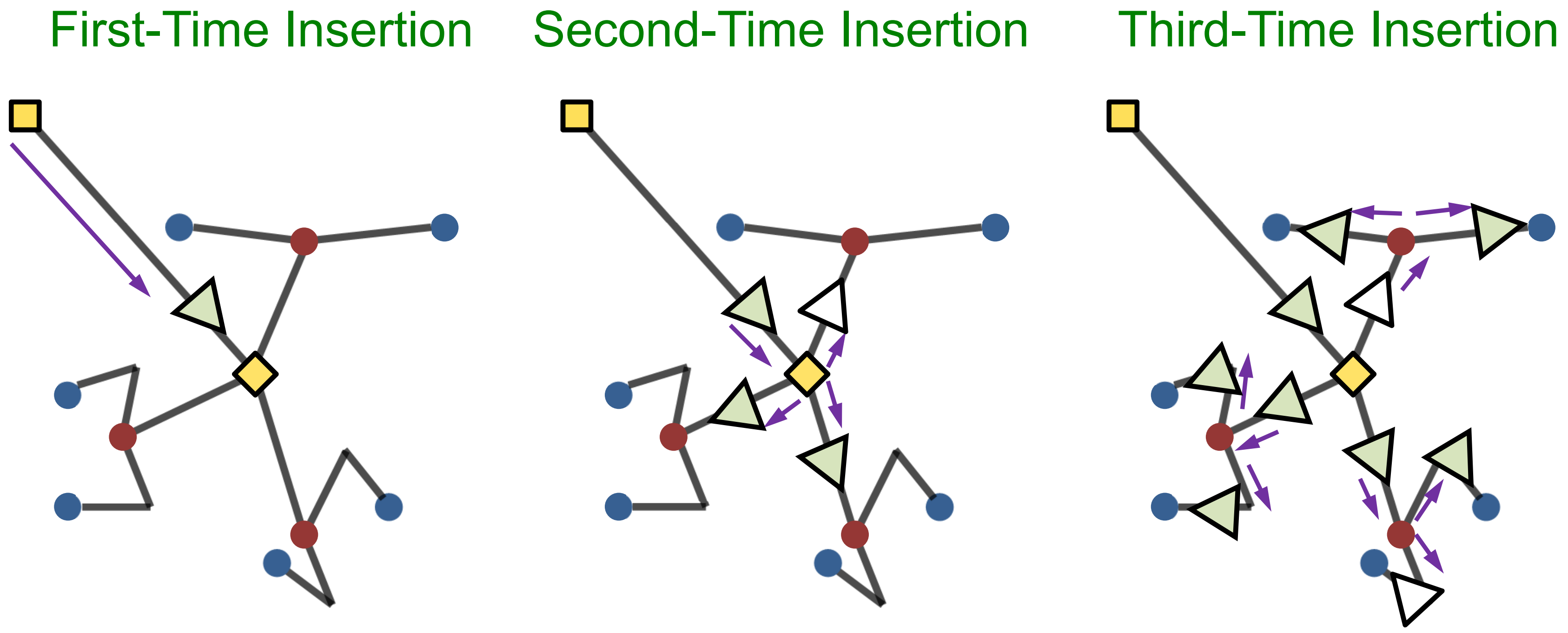


## Pseudo-Sink Handling

- For partitioning
  - Relax the sizes of clusters in a partition which can differ by at most one for the first recursion
- For embedding-region construction
  - Construct no embedding regions for pseudo sinks to reserve the flexibility of snaking
- For node embedding
  - Let the embedding regions of pseudo sinks cover entire chip
- Dangling wires can be identified and attached to proper sub-trees successfully

## Buffer Insertion

- Align buffer distribution on the symmetrical tree topology
- Insert identical buffers level by level top-down



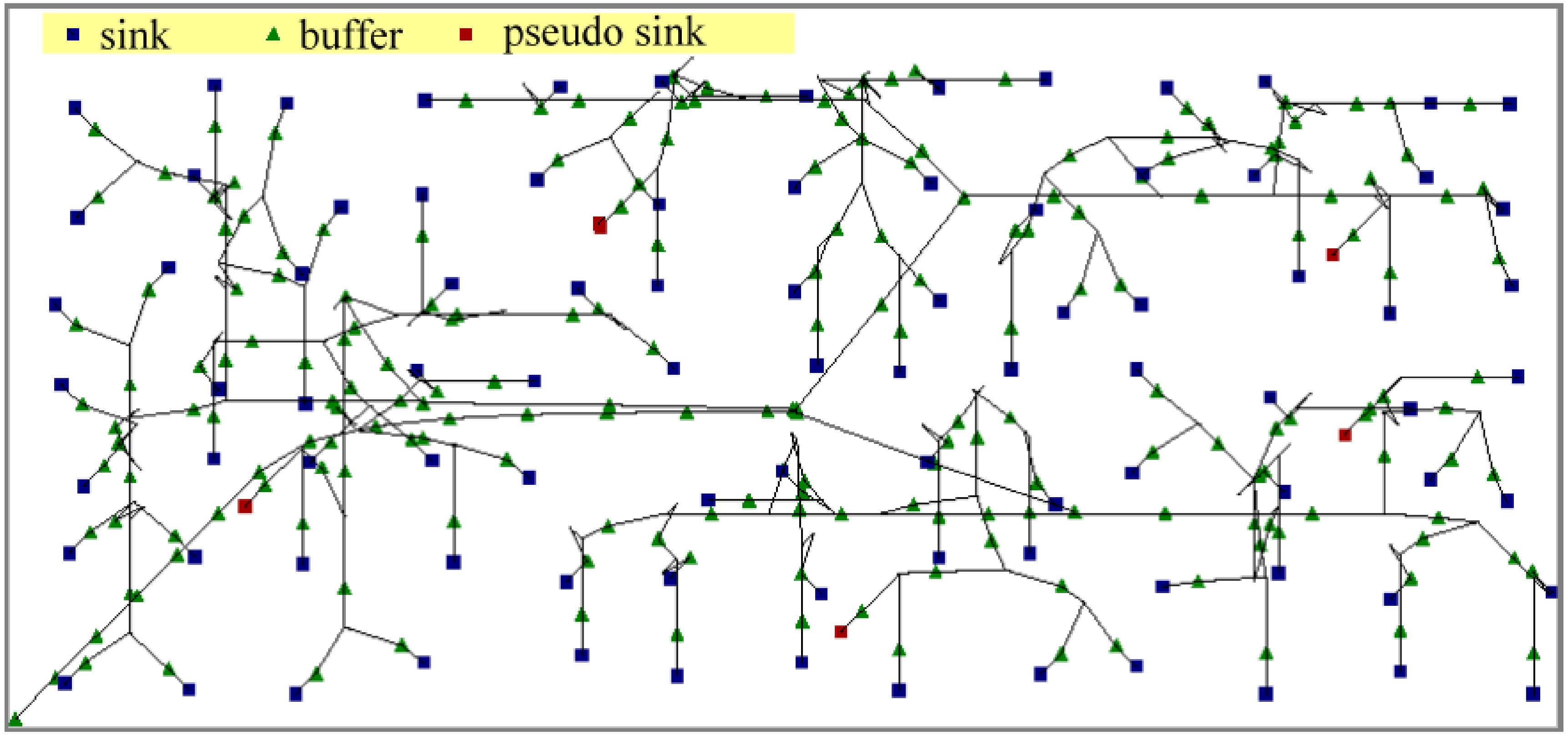
## Experimental Results on IBM Benchmarks

- Our approach can obtain much smaller skews in much shorter runtime than the state of the art, with marginal overheads of snaking for symmetry

| Circuit | # sinks | Shih et al. [ASPDAC'10] w/o simulation | | | Shih et al. [ASPDAC'10] w/ simulation | | | **Ours** | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | skew (ps) | usage (fF) | runtime (s) | skew (ps) | usage (fF) | runtime (s) | skew (ps) | usage (fF) | runtime (s) |
| r1 | 267 | 14.005 | 14001 | 2 | 5.012 | 15229 | 5126 | 1.510 | 13829 | 0.070 |
| r2 | 598 | 16.012 | 28011 | 11 | 6.421 | 29234 | 7374 | 1.770 | 31056 | 0.280 |
| r3 | 862 | 16.532 | 39123 | 26 | 5.611 | 41431 | 12739 | 2.310 | 44188 | 1.050 |
| r4 | 1903 | 17.792 | 89312 | 165 | 5.418 | 91015 | 17871 | 2.540 | 98450 | 3.350 |
| r5 | 3101 | 21.557 | 149875 | 498 | 7.028 | 156854 | 26045 | 3.010 | 171228 | 5.560 |
| **avg. comparison** | | **7.93** | **0.92** | **46.29** | **2.77** | **0.96** | **24343.13** | **1.00** | **1.00** | **1.00** |

More than 80000X faster than the ISPD-09 contest winners (simulation-based methods)

## Resulting Clock Tree: *ispd09f22*

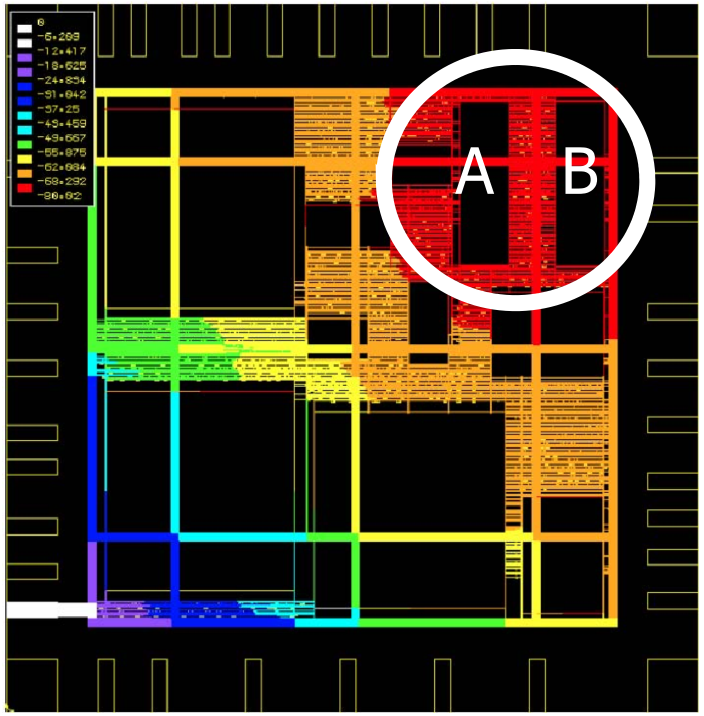


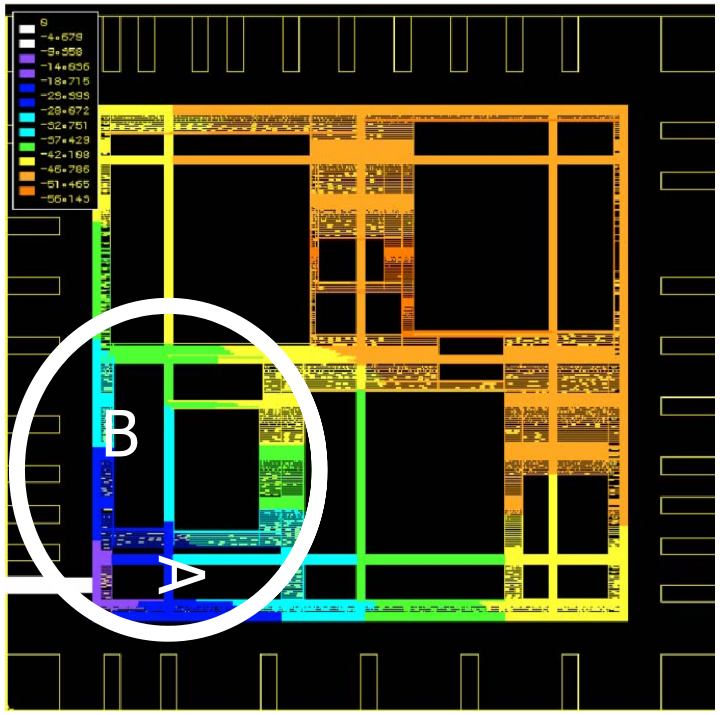
# Appendix B:

## Liu and Chang

## "Floorplan and power/ground network co-synthesis for fast design convergence"

## ISPD-06 (TCAD-07)





## Floorplan & P/G Network Co-Synthesis

- Liu and Chang, “Floorplan and power/ground network co-synthesis for fast design convergence,” ISPD-06 (TCAD-07).
- Apply the B*-tree floorplan representation and simulated annealing (SA)
- Analyze the P/G network (typical flow)
  - Circuit modeling
  - Global P/G network construction
  - P/G network modeling/reduction
  - P/G network evaluation (IR-drop computation)
- Reduce floorplan solution space

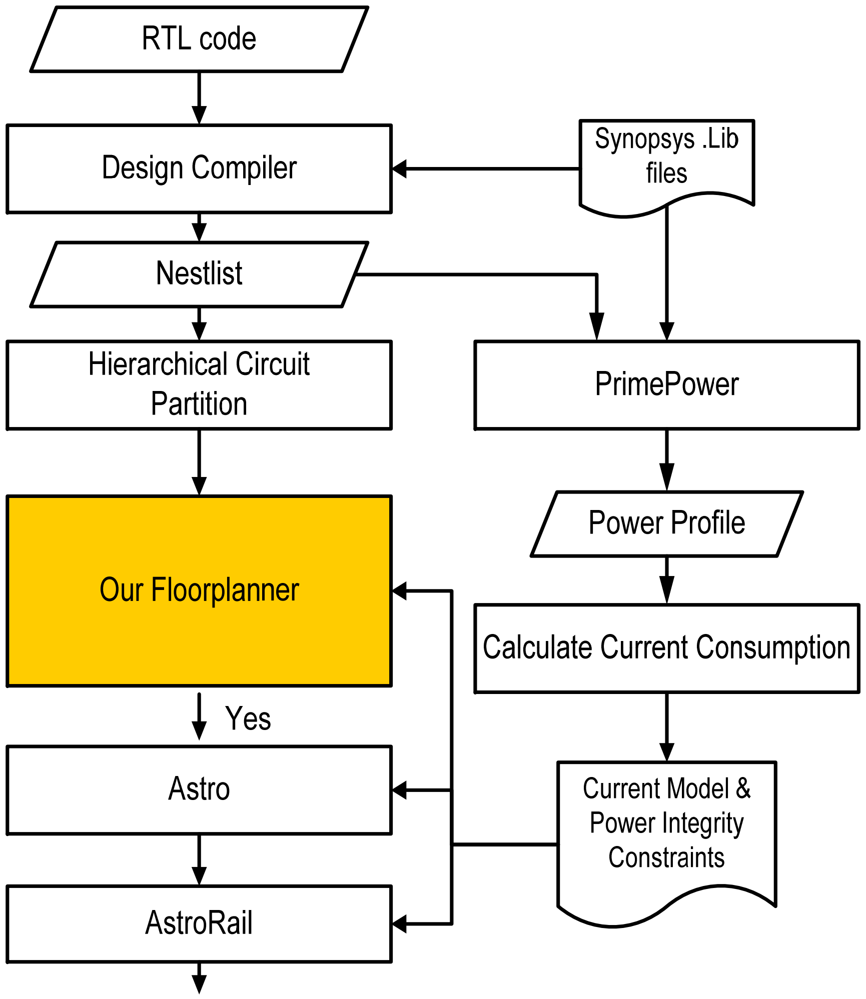
## Implementation of the Design Flow

Data preparation

- Power profile
  - Power consumption data of the modules generated by PrimePower
- Hierarchical circuit partition
  - Organize the design into hard modules and soft modules according to the hierarchy

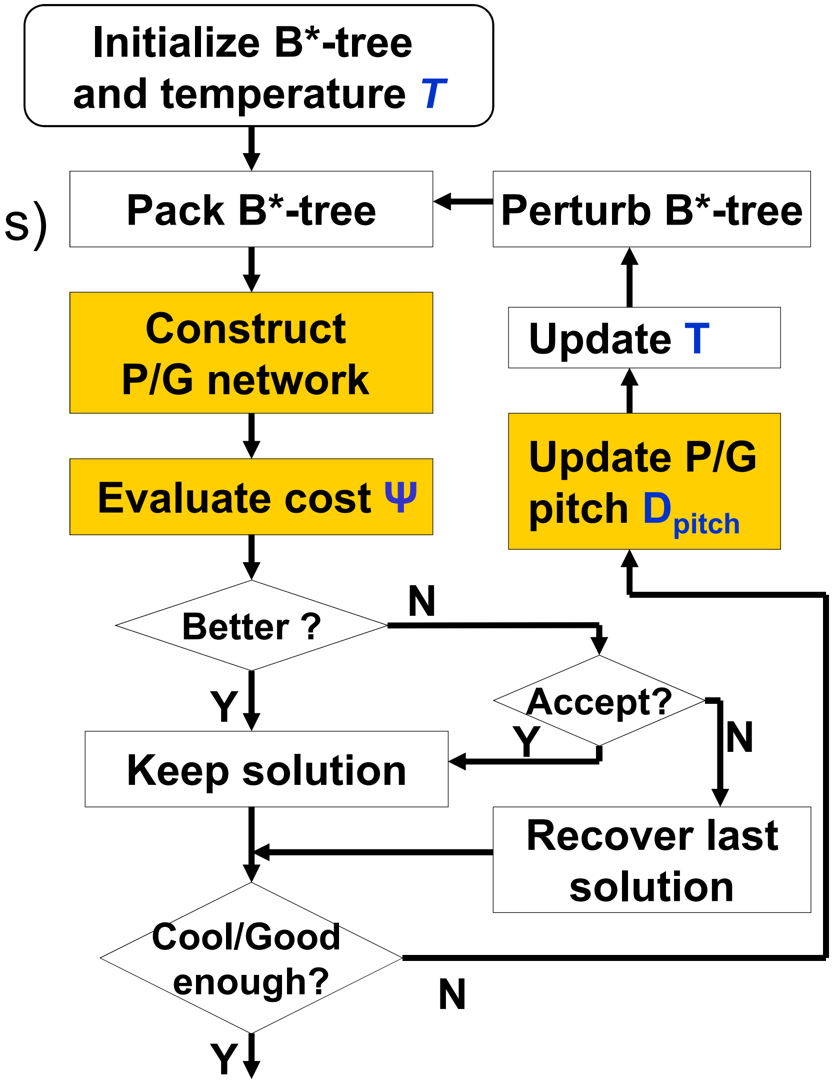
Post-layout verification

- AstroRail
  - Static cell-level P/G analysis



# Simulated Annealing Process

- Non-zero probability for up-hill climbing: $$p = \min\left(1, e^{-\frac{\Delta\Psi}{T}}\right)$$
- Perturbations (neighboring solutions)
  - Op1: Rotate a block
  - Op2: Move a node/block to another place
  - Op3: Swap two nodes/blocks
  - Op4: Resize a soft block
- The cost function Ψ is based on the floorplan cost and P/G network cost
- T is decreased every *n* cycles, where *n* is proportional to the number of blocks



## Cost Function

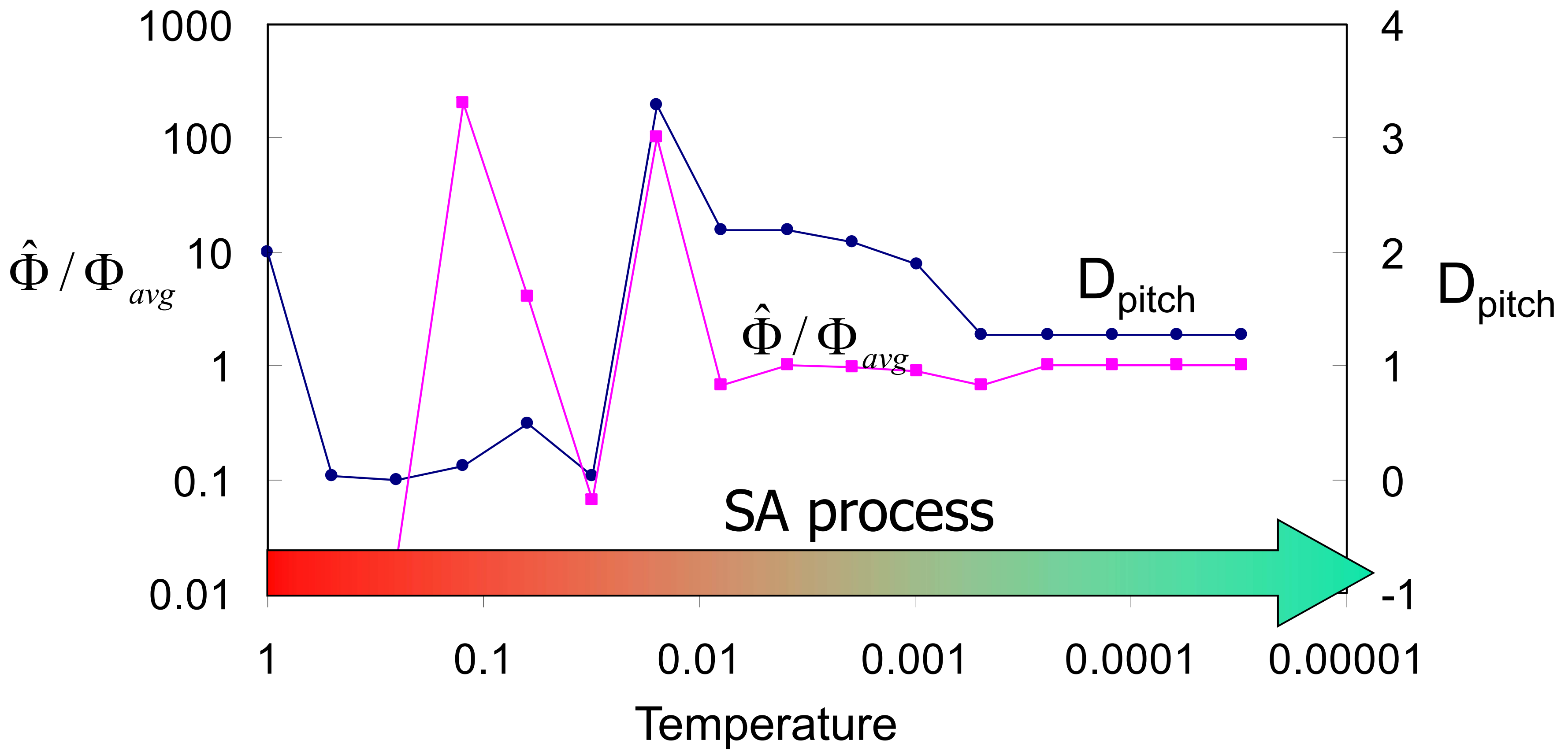
- Cost function:

$$\Psi = \overset{\text{Wirelength}}{\overset{\searrow}{\alpha}} \cdot W + \overset{\text{Area}}{\overset{\searrow}{\beta}} \cdot A + \overset{\text{P/G cost}}{\overset{\downarrow}{\gamma}} \cdot \Phi + \omega \cdot \overset{\text{P/G Density}}{\overset{\swarrow}{\frac{A}{D_{pitch}^{2}}}},$$

- W: Wirelength
- A : Area
- Φ : P/G network cost (penalty of power integrity violation)
- $D_{pitch}$: pitch of P/G network
  - Increasing power mesh density (reducing $D_{pitch}$) reduces Φ
  - Update $D_{pitch}$ by multiplying $\hat{\Phi} / \Phi_{avg}$
  - $\Phi_{avg}$ : Average P/G network cost at a temperature
  - $\hat{\Phi}$: $0 < \hat{\Phi} < 1$, a factor for adjusting the density of P/G networks
    Smaller $\hat{\Phi}$ for higher P/G density and larger one for lower P/G density

## Pitch Updating: An Example

- At the beginning of SA, $D_{pitch}$ = 2 and $\hat{\Phi} = 0.02$
- During SA process, $D_{pitch} \Leftarrow \hat{\Phi} / \Phi_{avg} \times D_{pitch}$



$\hat{\Phi} / \Phi_{avg}$ converges to 1 while temperature cools down

# P/G Network Cost

Φ: P/G network cost

$$\Phi = \overset{\text{EM cost}}{\boxed{\theta \cdot \frac{|B_{em}|}{|B|}}} + (1-\theta)\cdot \overset{\text{IR-drop cost}}{\boxed{\frac{\sum_{\forall p_{vi} \in P_v} v_{p_{vi}}}{\sum_{\forall Pi \in P} V_{\lim, pi}}}}, \quad 0 < \theta < 1$$

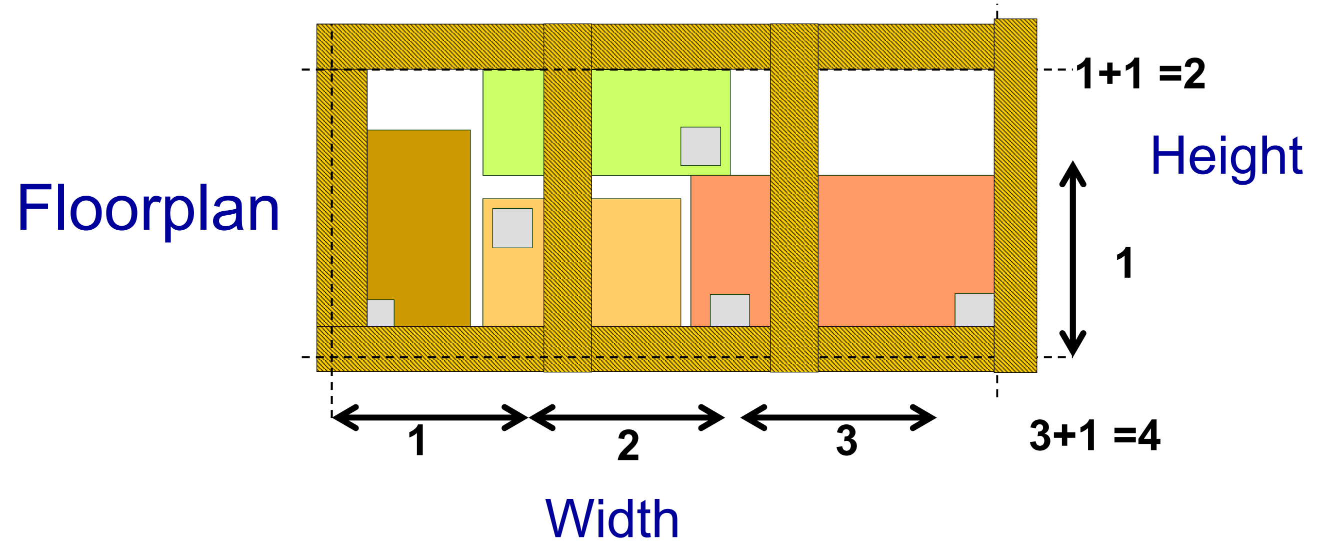
- $B_{em}$: set of branches violating electromigration constraints
- $B$ : total branches of the P/G mesh
- $v_{pvi}$: amount of the violation at the pin $p_{vi}$
- $P$ : set of all P/G pins
- $P_v$ : set of violating P/G pins
- $V_{lim,pi}$ : IR-drop constraint of the P/G pin $p_i$

## P/G Network Construction

- For each floorplan, we construct a uniform global P/G network according to $D_{pitch}$
- The number of trunks is defined by

***round[width/$D_{pitch}$]+1*** & ***round[height/$D_{pitch}$]+1***

2X4 uniform P/G network is constructed

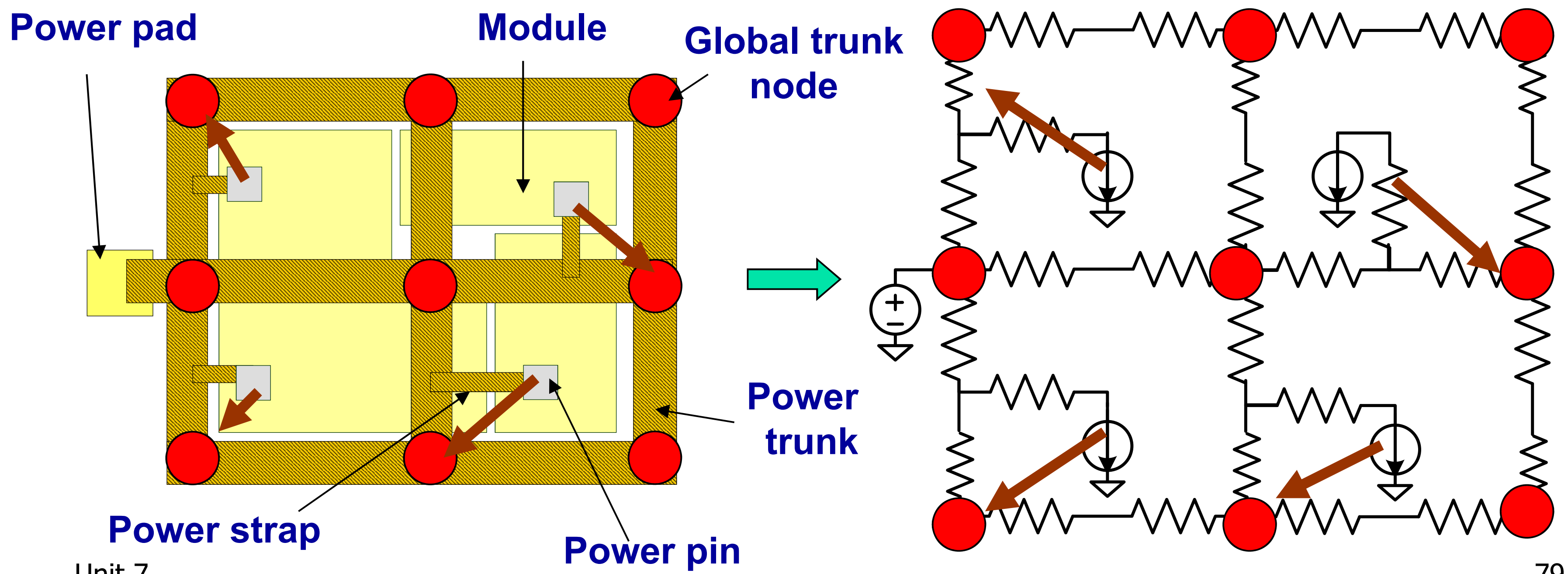


Calculate the P/G network dimension

# P/G Network Modeling

Apply static analysis for fast P/G network evaluation

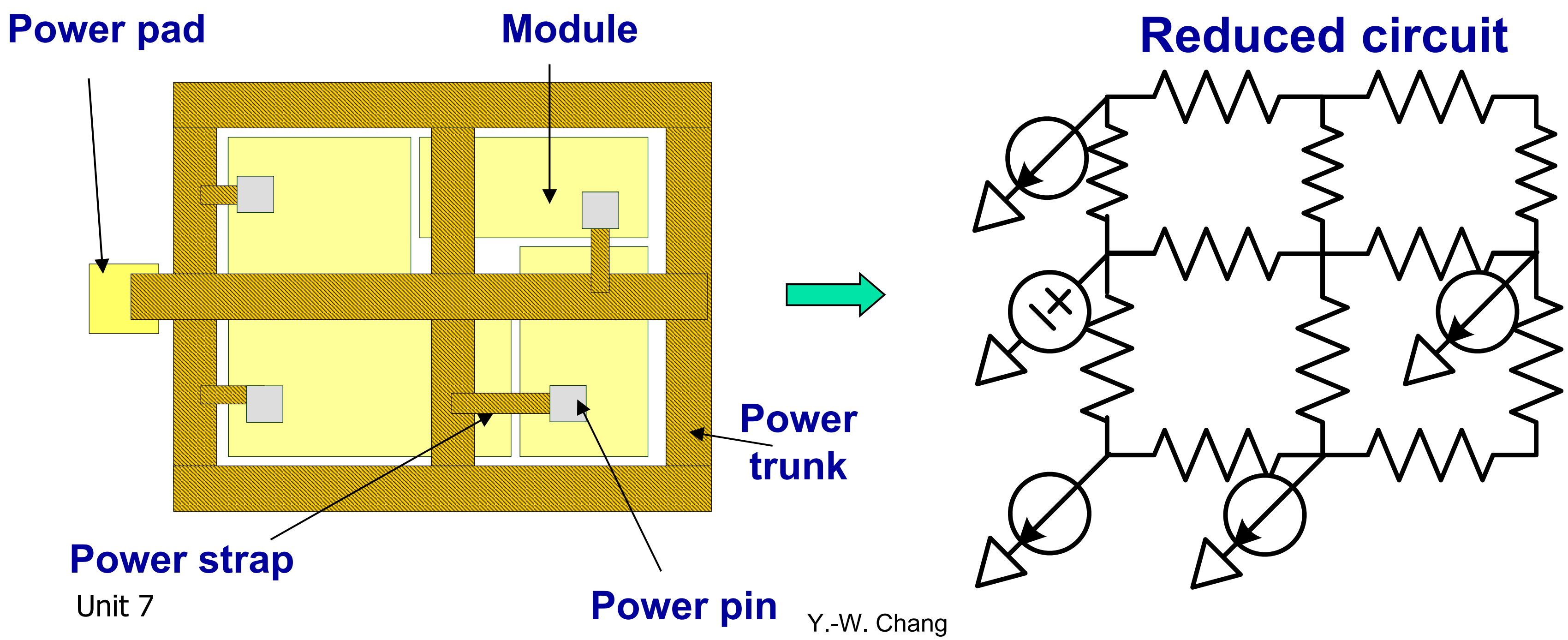
- Use resistive P/G Model
- Model P/G pins by current sources
  - Current value: maximum current drawn from P/G pins
- Reduce circuit size
  - Connect current sources to nearest global trunk nodes



## P/G Network Modeling

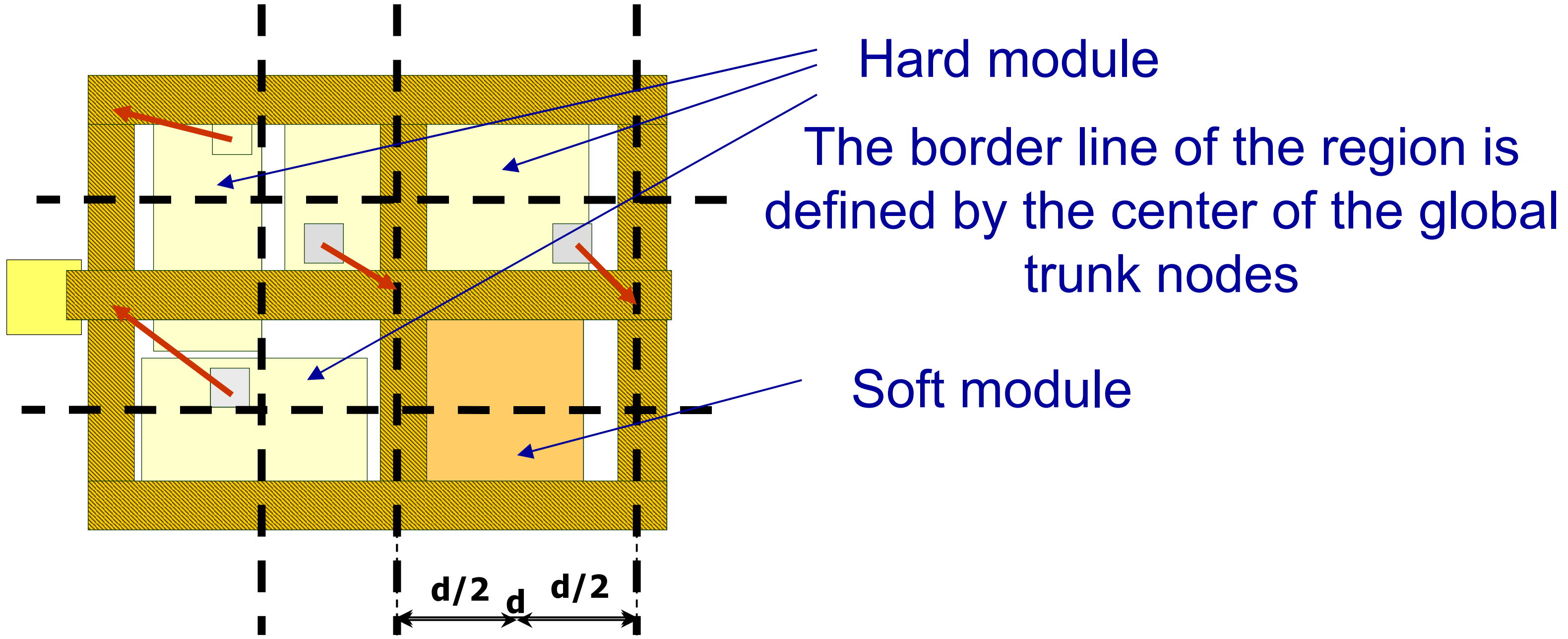
Apply static analysis for fast P/G network evaluation

- Use resistive P/G Model
- Model P/G pins by current sources
  - Current value: maximum current drawn from P/G pins
- Reduce circuit size
  - Connect current sources to nearest global trunk nodes



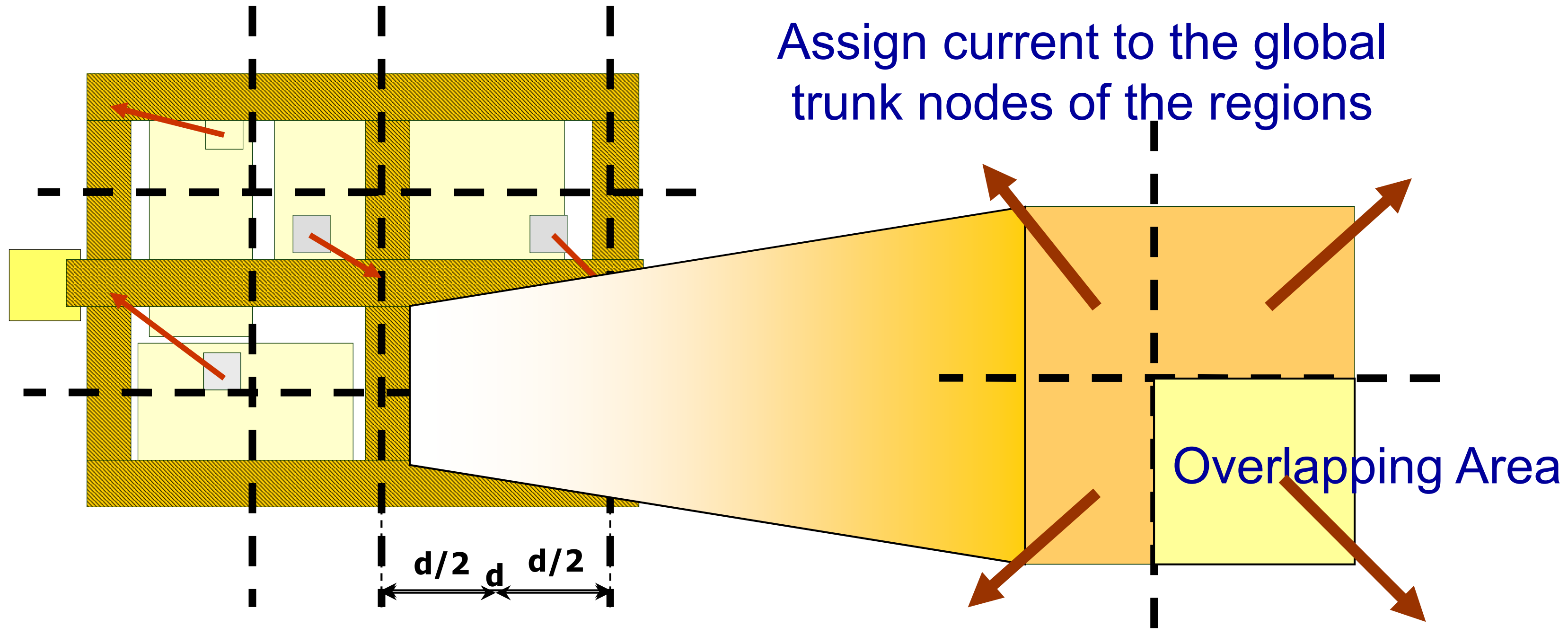
# Macro Current Modeling

- Divide the floorplan into regions
- For hard macros
  - Connect P/G pins to the nearest global trunk nodes
- For soft macros (worst-case scenario)
  - Collect the largest current drawn by standard cells in the overlapping area of the region and the soft macro


Hard module
The border line of the region is defined by the center of the global trunk nodes
Soft module
d/2
d
d/2

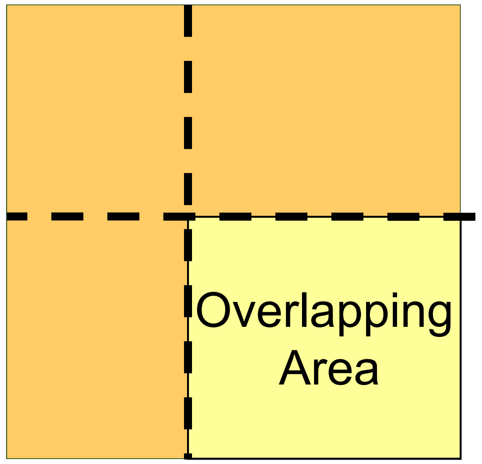
## Macro Current Modeling

- Divide the floorplan into regions
- For hard macros
  - Connect P/G pins to the nearest global trunk nodes
- For soft macros (worst-case scenario)
  - Collect the largest current drawn by standard cells in the overlapping area of the region and the soft macro

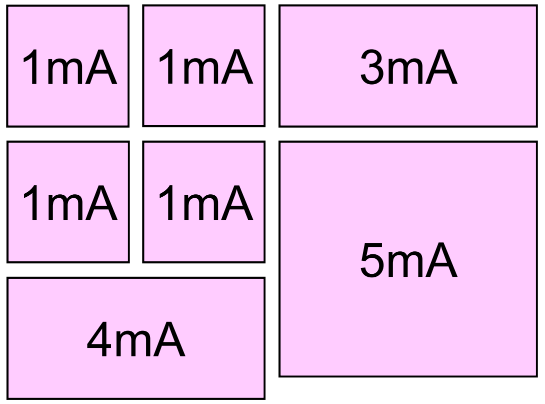


## Soft Macro Modeling

- Derive the largest current drawn by standard cells of the overlapping area
  - Maximize the current of the overlapping area
  - Constraint: total standard cell area < the overlapping area
  - The problem is known as 0-1 Knapsack Problem (NP-complete)
- Approximate it by Fractional Knapsack Algorithm
  - Assume standard cells can be broken into arbitrary smaller pieces
  - Rank cells by current to area ratio
  - Apply a greedy algorithm (complexity $O(n \lg n)$)



Standard Cells of the soft module



## Evaluation of P/G Network

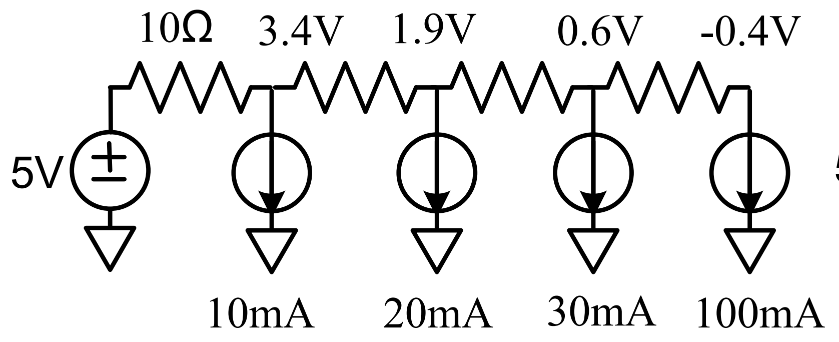
- The static analysis of a P/G network is formulated as the following modified nodal analysis (MNA) formula:

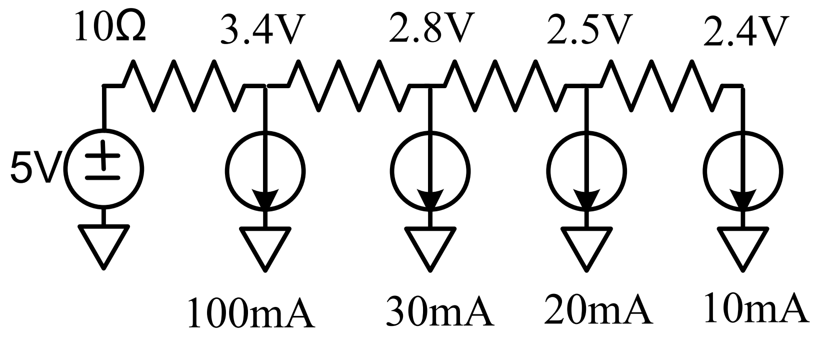
$$\mathbf{Gx = i}$$

  - **G**: conductance matrix (sparse positive definite matrix)
  - **x**: vector of node voltages
  - **i**: vector of current loads and voltage sources
  - Dimensions of G, i and x are equal to the number of nodes in the P/G network
- Solve the linear equation
  - Apply Preconditioned Conjugated Gradient (PCG) method
  - The time complexity is linear

## Idea of Solution Space Reduction

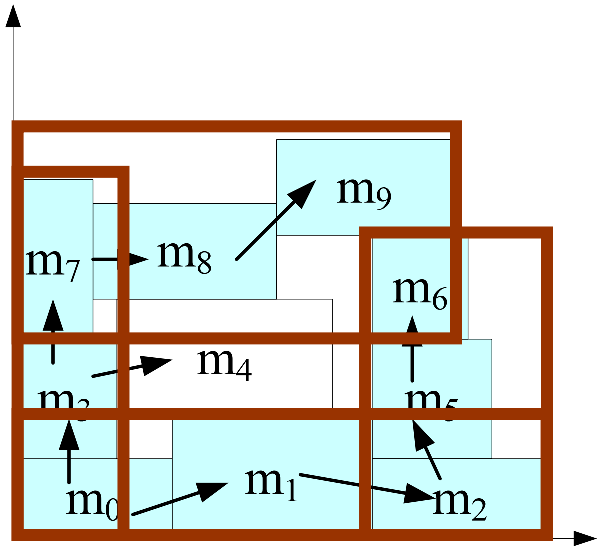
- The IR-drop of a P/G pin is proportional to the effective resistance between the P/G pin and the power pad
  - The closer the P/G pin is placed to the power pad, the smaller the IR-drop
- A technique to reduce solution space
  - Place the modules consuming larger current (power-hungry modules) near the boundary of the floorplan
  - Place power pads close to them

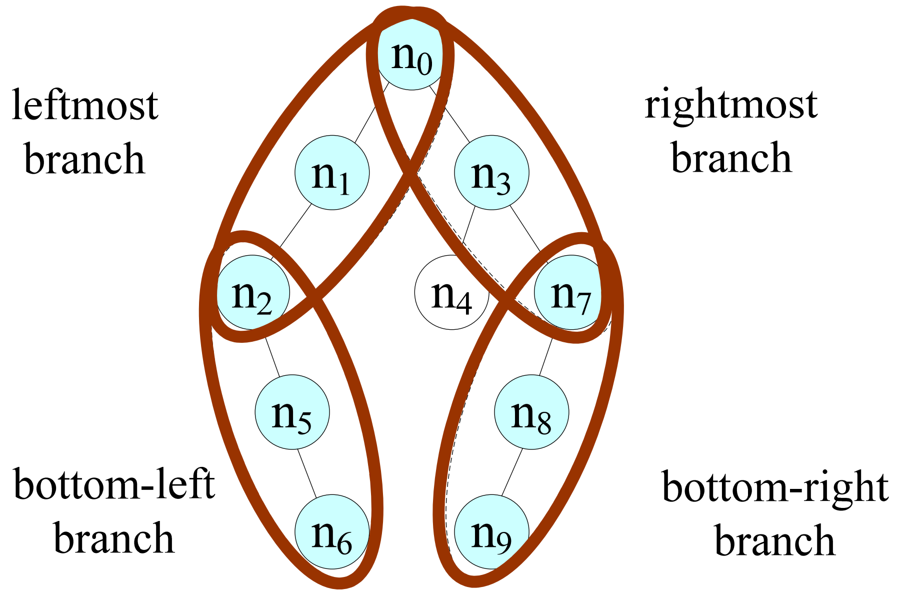




# B*tree Boundary Properties

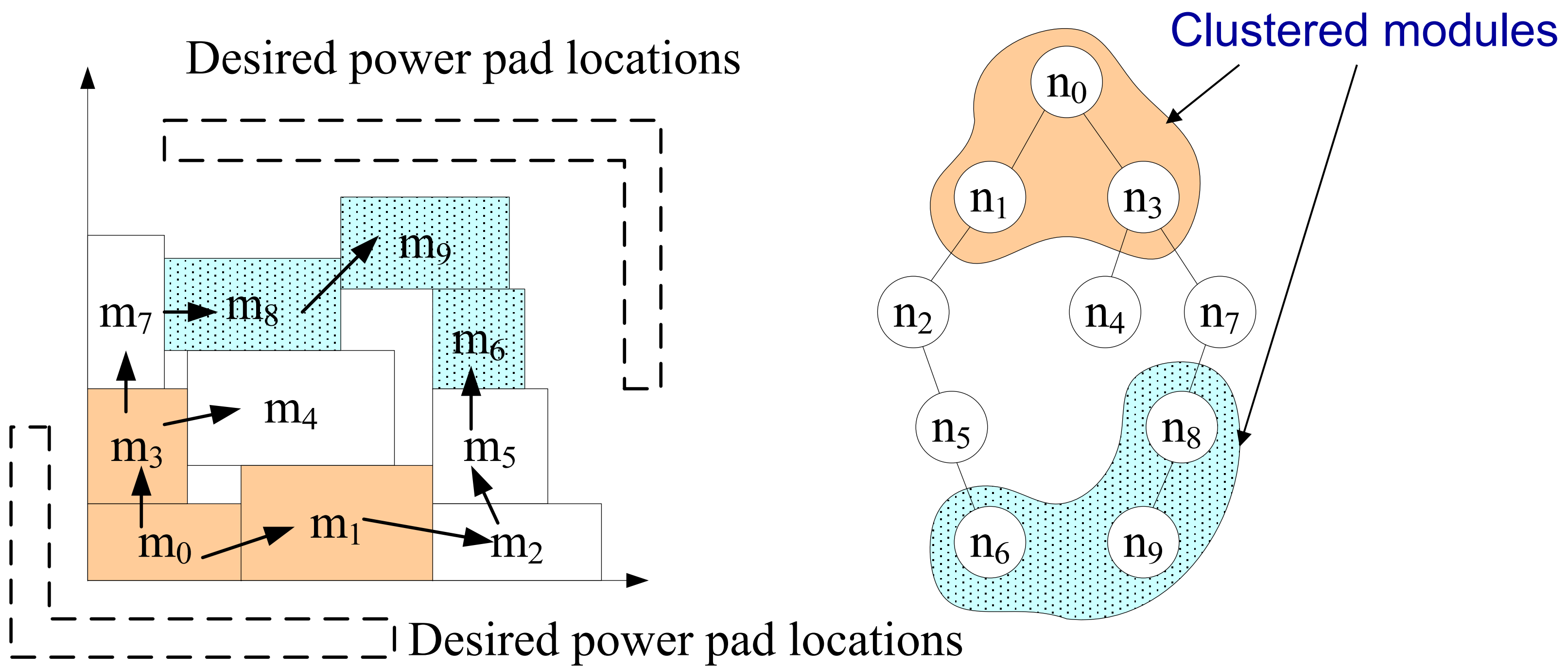
- Bottom boundary modules: the leftmost branch
- Left-boundary condition
  - Left boundary modules: the rightmost branch
- Right-boundary condition
  - Right boundary modules: the bottom-left branch
- Top-boundary condition
  - Top boundary modules: bottom-right branch





## Power-Hungry Modules Handling

- Power-Hungry Modules
  - Are clustered and restricted to satisfy the boundary property during B*-tree perturbation
  - P/G pads are placed near these modules



## Results on OpenRISC1200

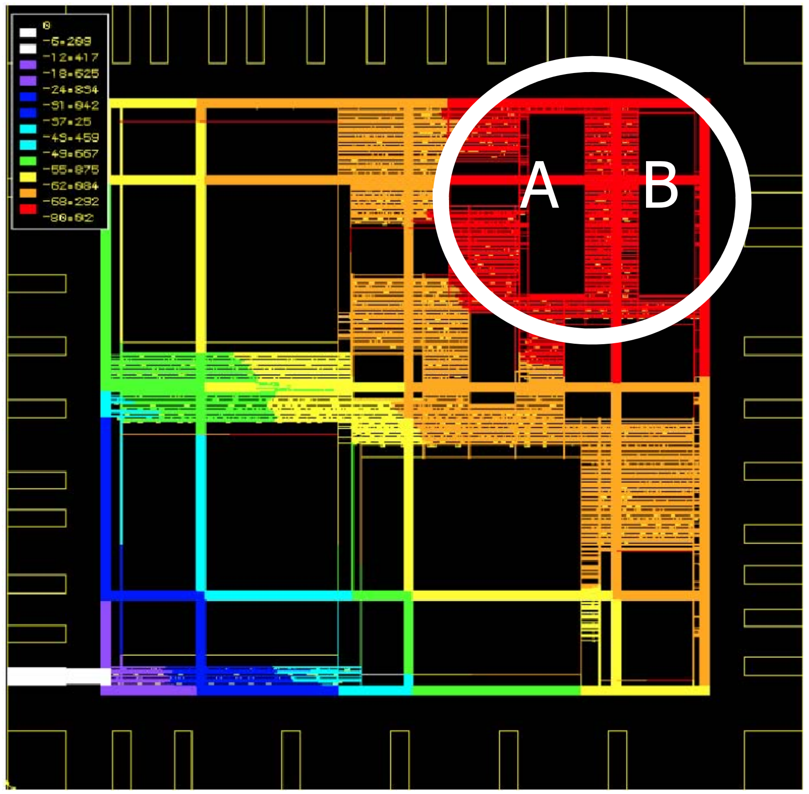
- Improve on runtime and max IR-drop with little overheads on delay & wirelength (UMC 0.18 um technology)

| OpenRISC1200 | *Astro Flow | *Astro w/ IR-drop Driven Placement | Our Flow | Our Improv. vs. Astro w/ IR-drop |
|---|---|---|---|---|
| Die Area ($mm^2$) | 3.86 | 3.86 | 3.33 | 15.9% |
| Utilization (%) | 62 | 62 | 72 | 13.9% |
| Wirelength (μm) | 1655463 | 1539125 | 1540172 | -0.1% |
| Avg. Delay (ns) | 8.62 | 8.54 | 8.55 | -0.1% |
| Max IR-drop (mv) | 80.18 | 78.20 | 55.14 | 41.8% |
| CPU Runtime (s) | 505 | 346 | 135 | 2.56X |
| Iterations | 4 | 3 | 1 | - |

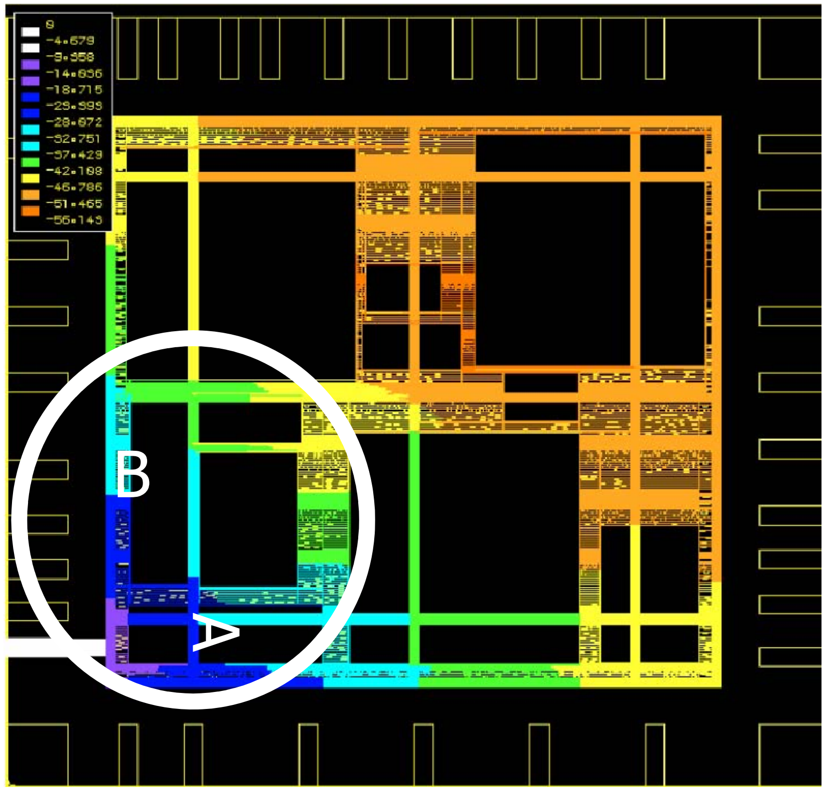
*Need iterative and manual P/G network fix

## Resulting Voltage Map

| Astro design flow | Our design flow |
|---|---|
| Power-hungry blocks (register files A&B) are placed far away from the power pad | Power-hungry blocks are placed beside the power pad |



0
-6.209
-12.417
-18.625
-24.834
-31.042
-37.25
-43.459
-49.667
-55.875
-62.084
-68.292
-80.02
A
B



0
-4.679
-9.358
-14.036
-18.715
-23.393
-28.072
-32.751
-37.429
-42.108
-46.786
-51.465
-56.143
B
A