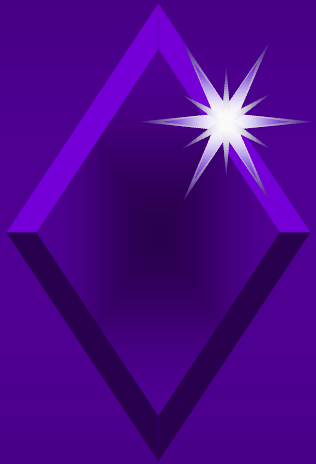


# Performance-Driven Interconnect Optimization



*Charlie Chung-Ping Chen*

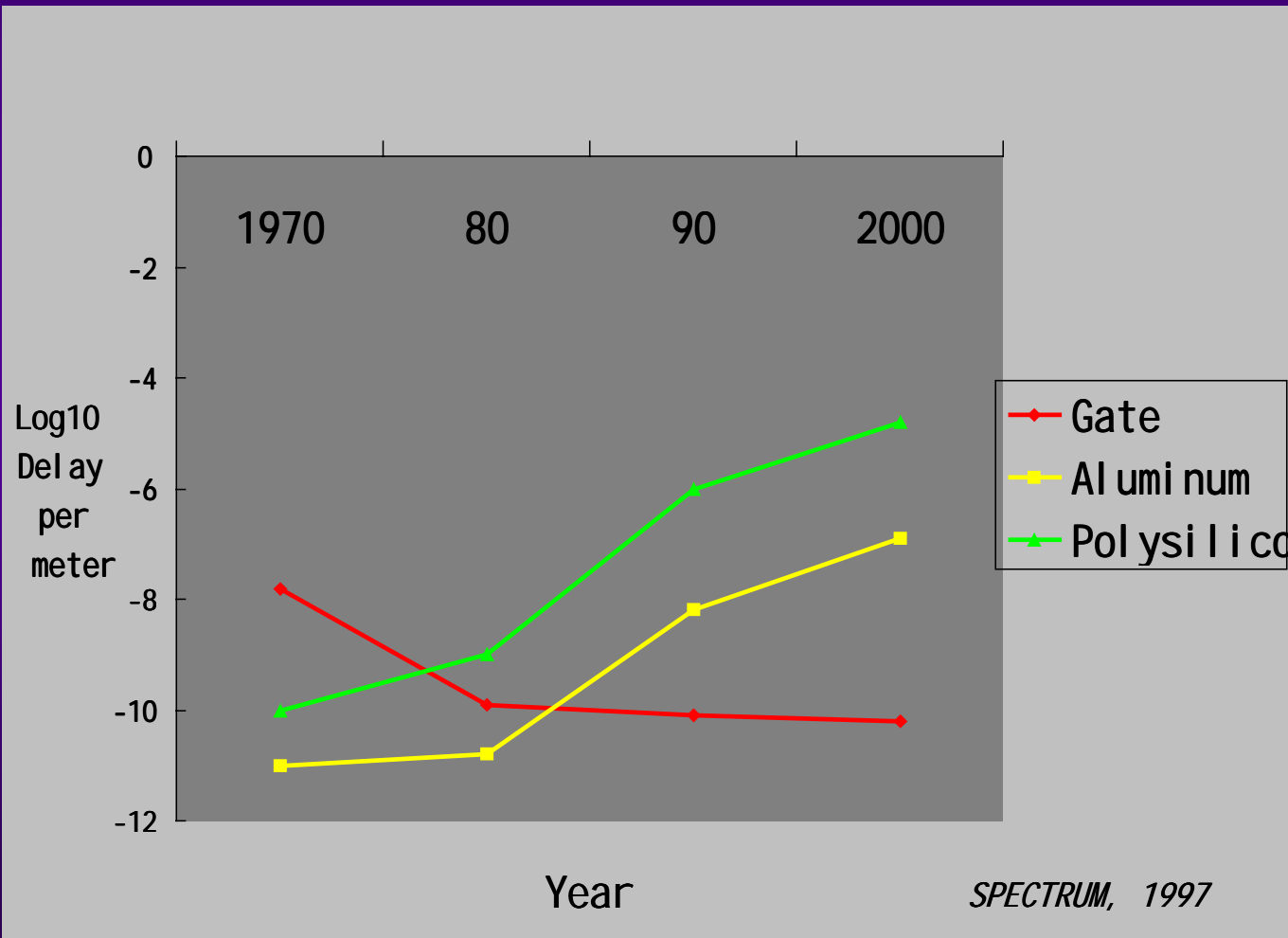
# *Publications*

- ◆ A Fast algorithm for Optimal Wire-Sizing Under Elmore Delay Model, ISCAS, 1995.
- ◆ Optimal Wire-Sizing Formula Under the Elmore Delay Model, DAC, 1996.
- ◆ Optimal Wire-Sizing Formula Under the Elmore Delay Model, ACM Physical Design Work Shop, 1996.
- ◆ Performance-Driven Buffered Clock Tree Optimization Based on Lagrangian Relaxation , DAC, 1996.
- ◆ Optimal Non-Uniform Wire-Sizing for Routing Trees, ICCAD, 1996.
- ◆ Optimal Wire-Sizing Function with Fringing Capacitance Consideration, DAC 1997.
- ◆ Spec-Based Buffer Insertion and Wire-Sizing for RC Nets, DTTC, 1997.
- ◆ Fast and Exact Simultaneous Transistor and Wire-Sizing by Lagrangian Relaxation, ICCAD, 98.

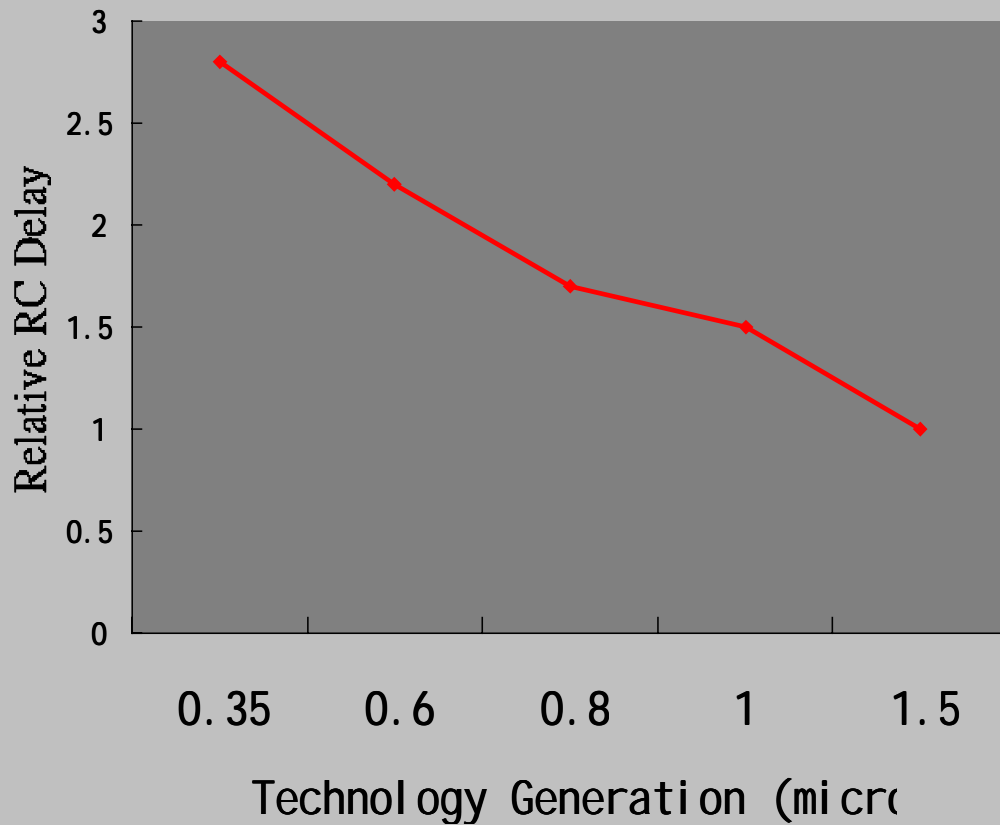
# *Outline*

- ◆ Interconnect Optimization
- ◆ Thesis
  - ◆ Wire Sizing
  - ◆ Buffer Sizing
  - ◆ **Buffer Insertion**
  - ◆ **Interconnect Simulation**

# Interconnect Delay Trend



# *Interconnect Delay Trend*



*I EDM 95*

# SPEED / PERFORMANCE ISSUE *The Technical Problem*

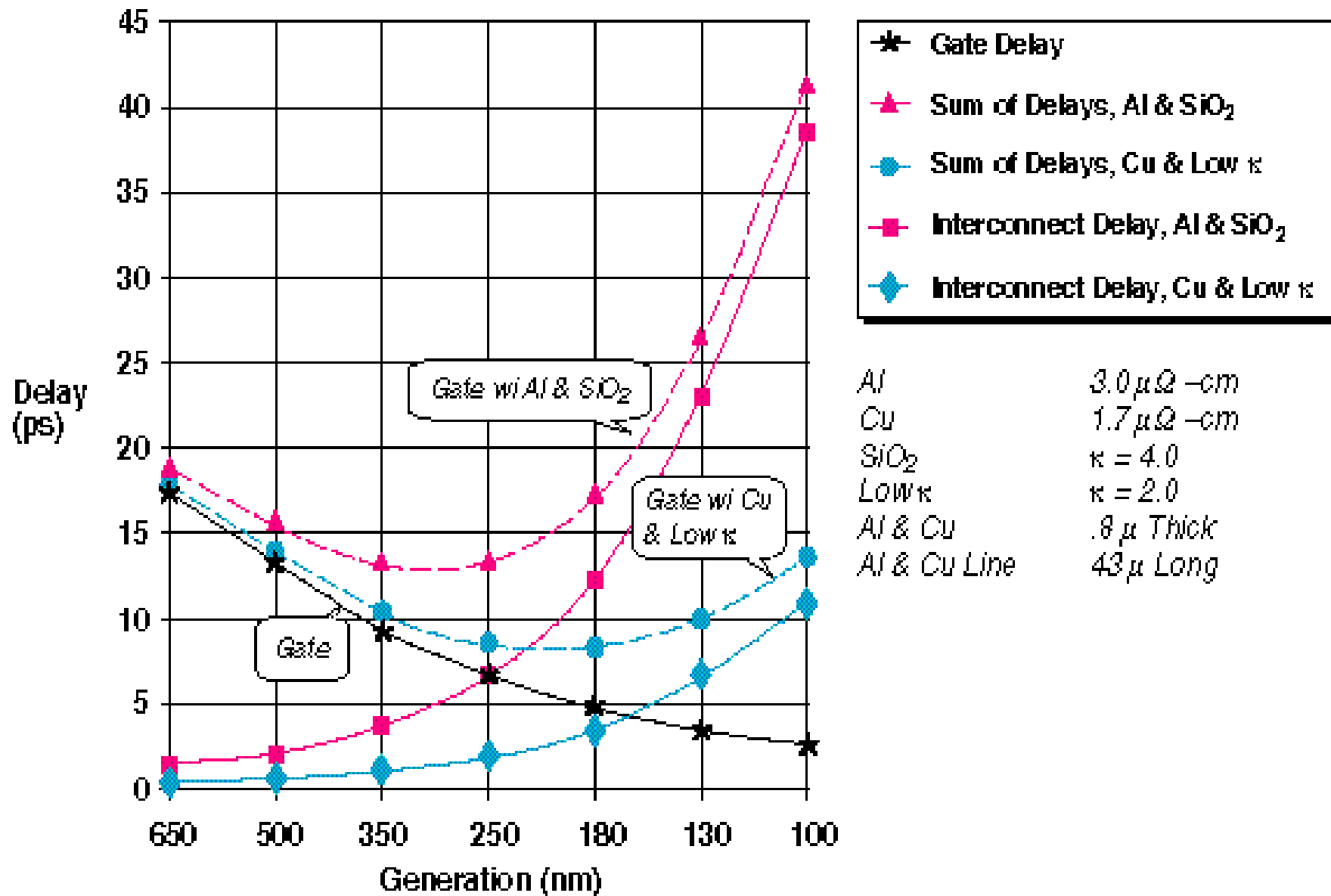
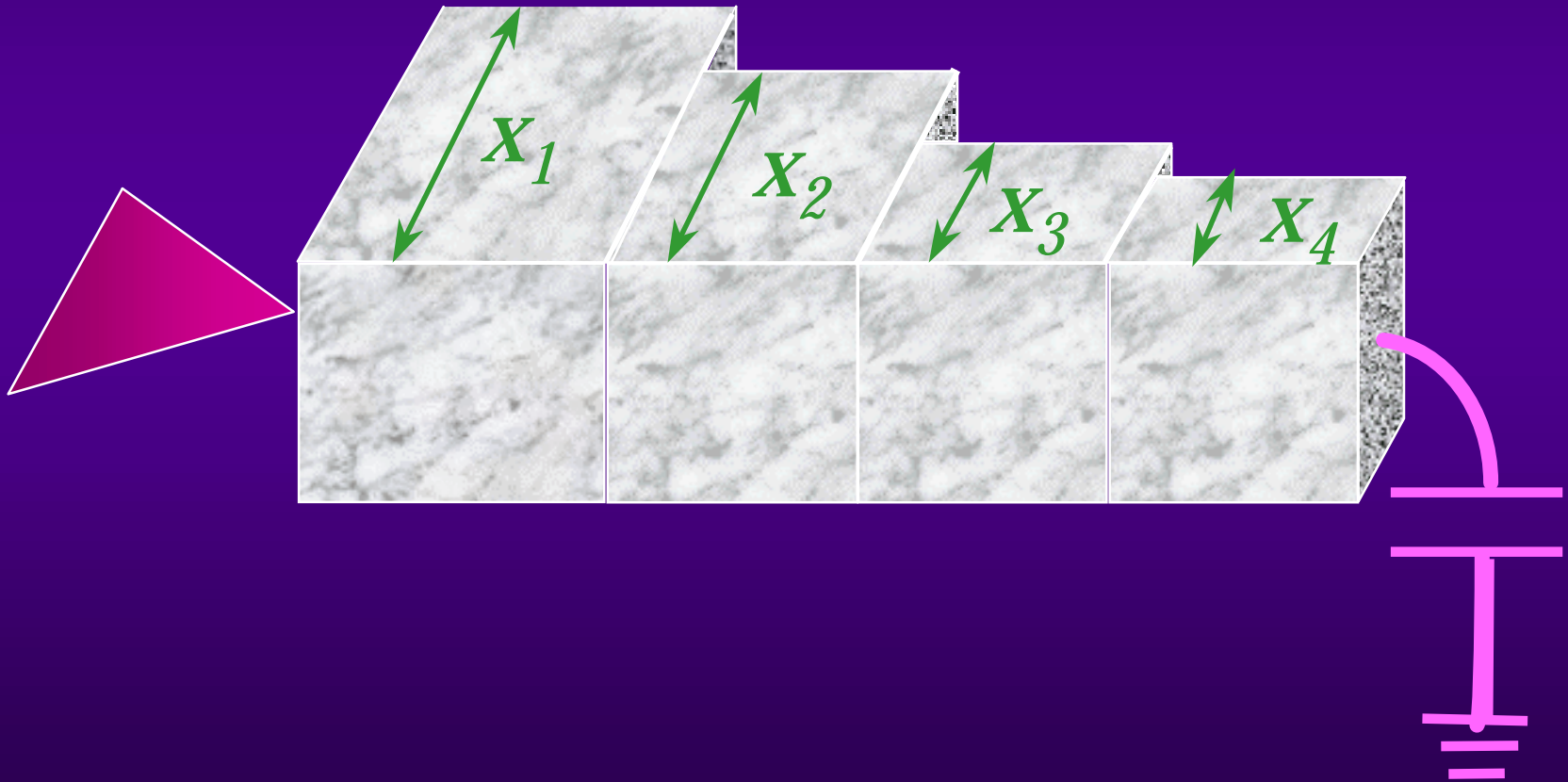
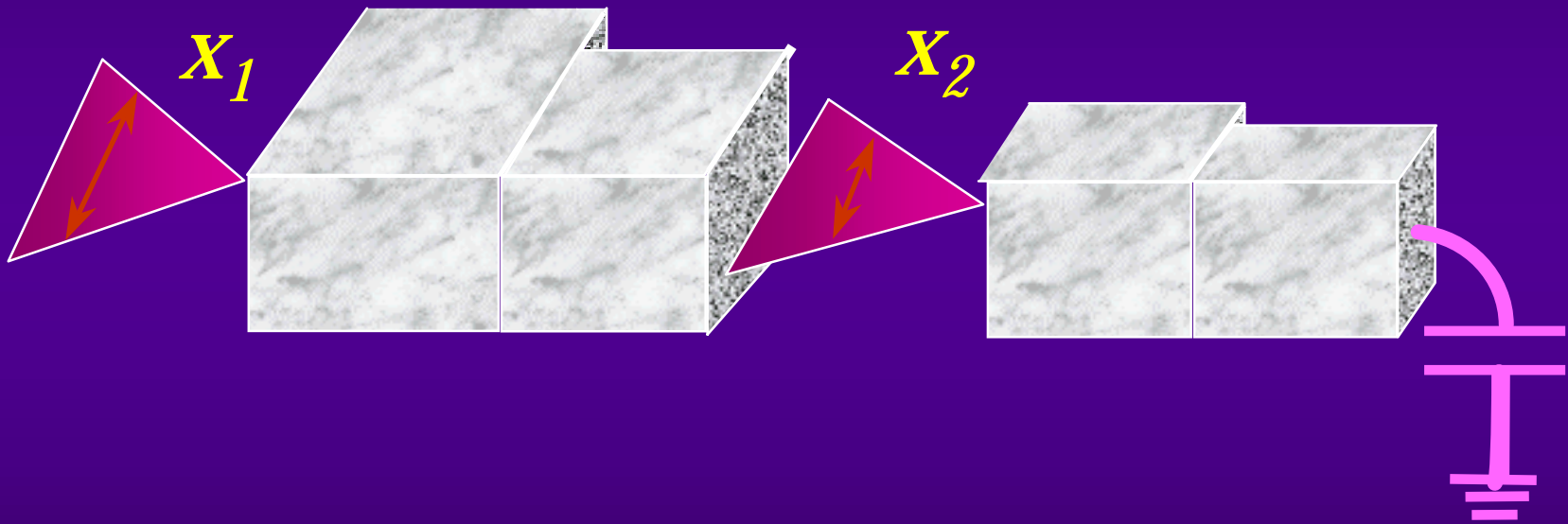


Figure 3 Calculated Gate and Interconnect Delay versus Technology Generation

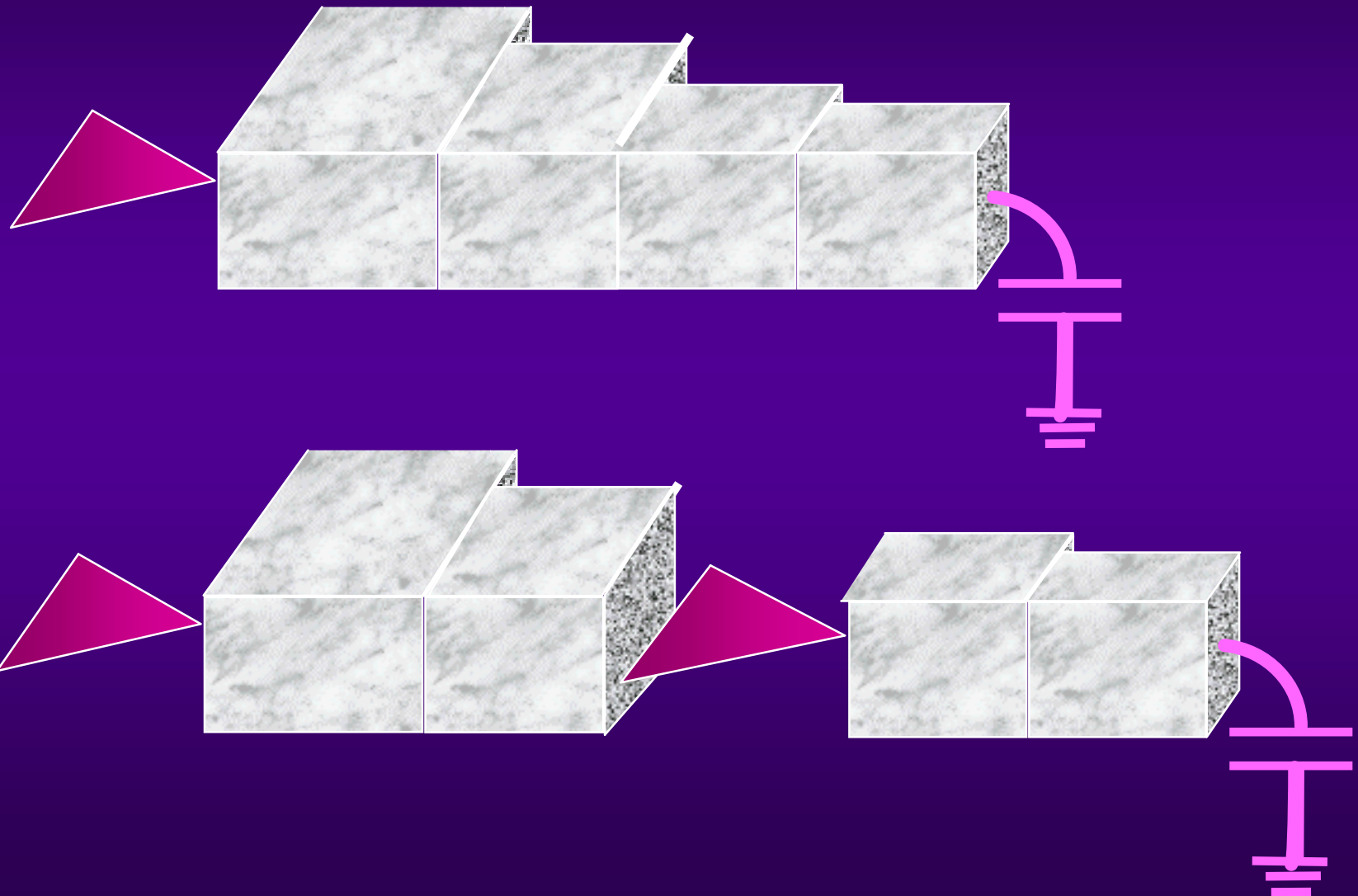
# Wire-Sizing



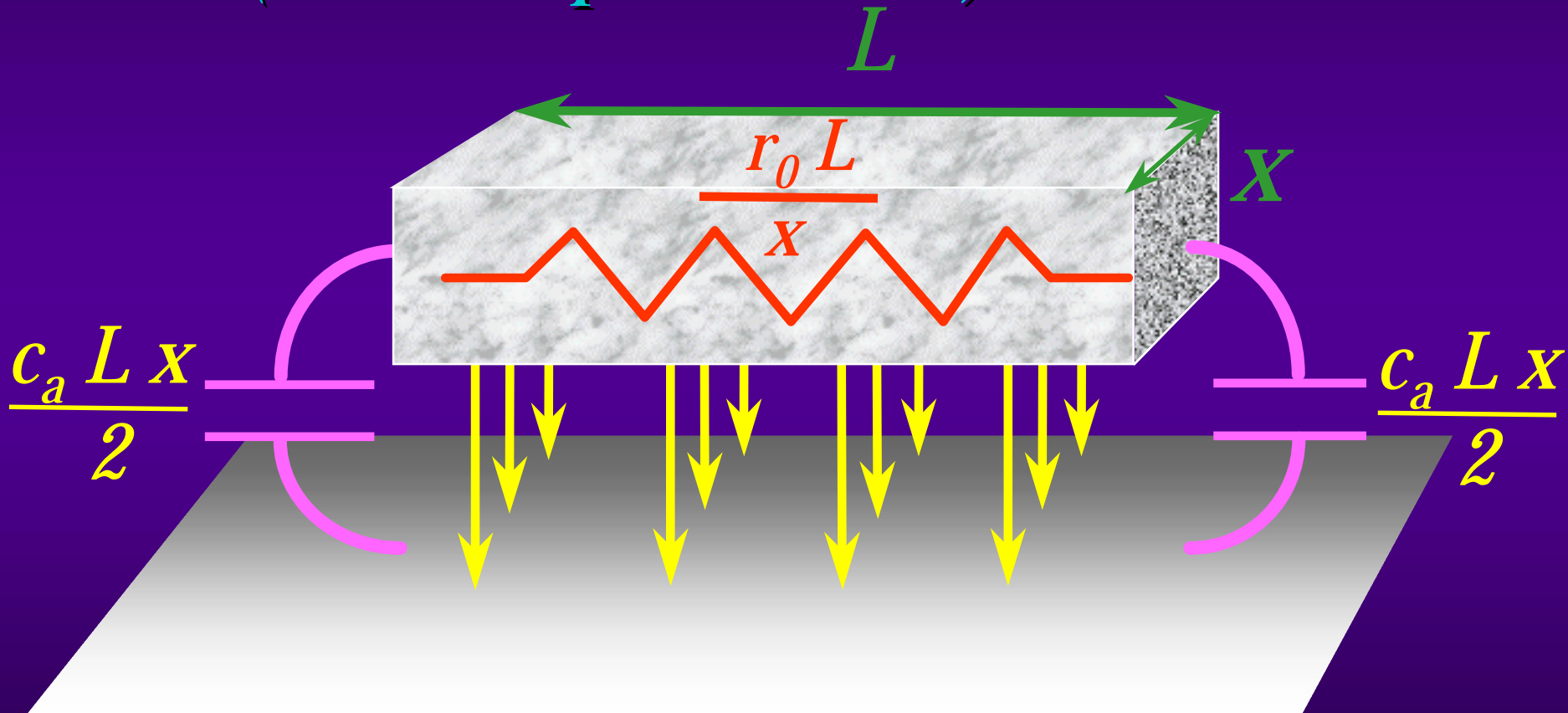
# *Buffer Sizing*



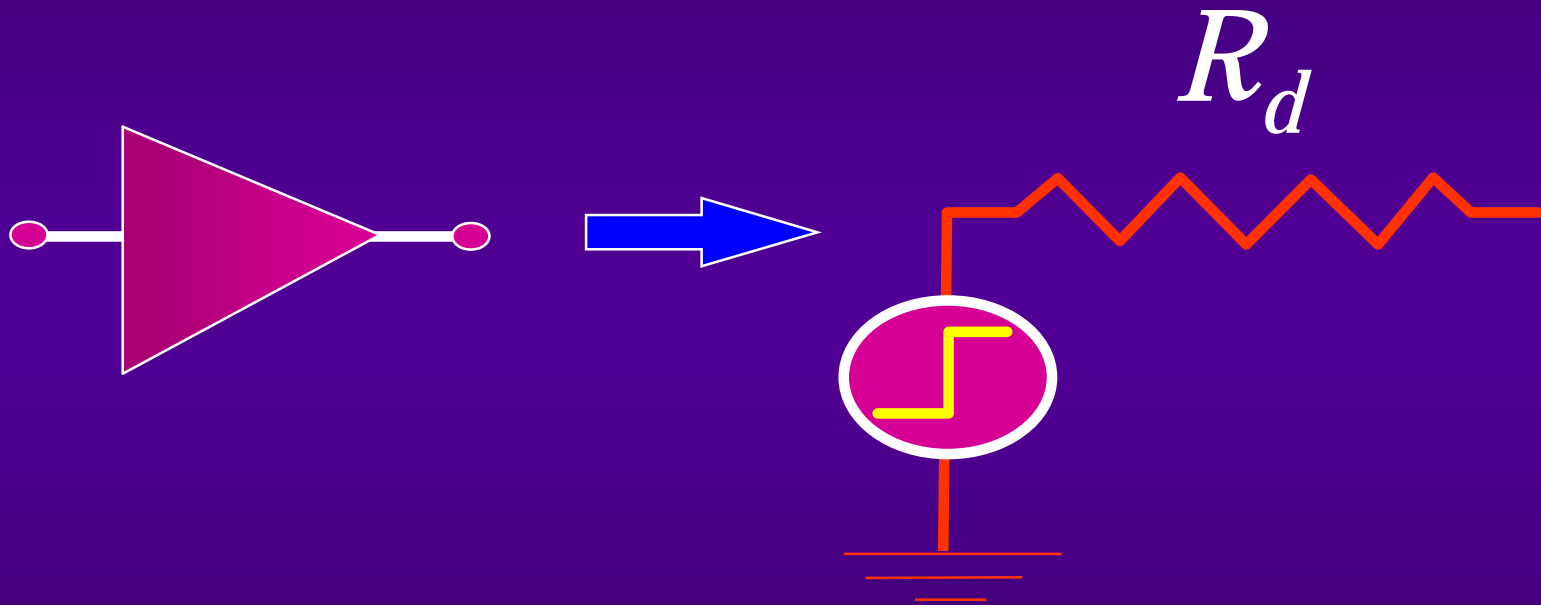
# *Buffer Insertion*



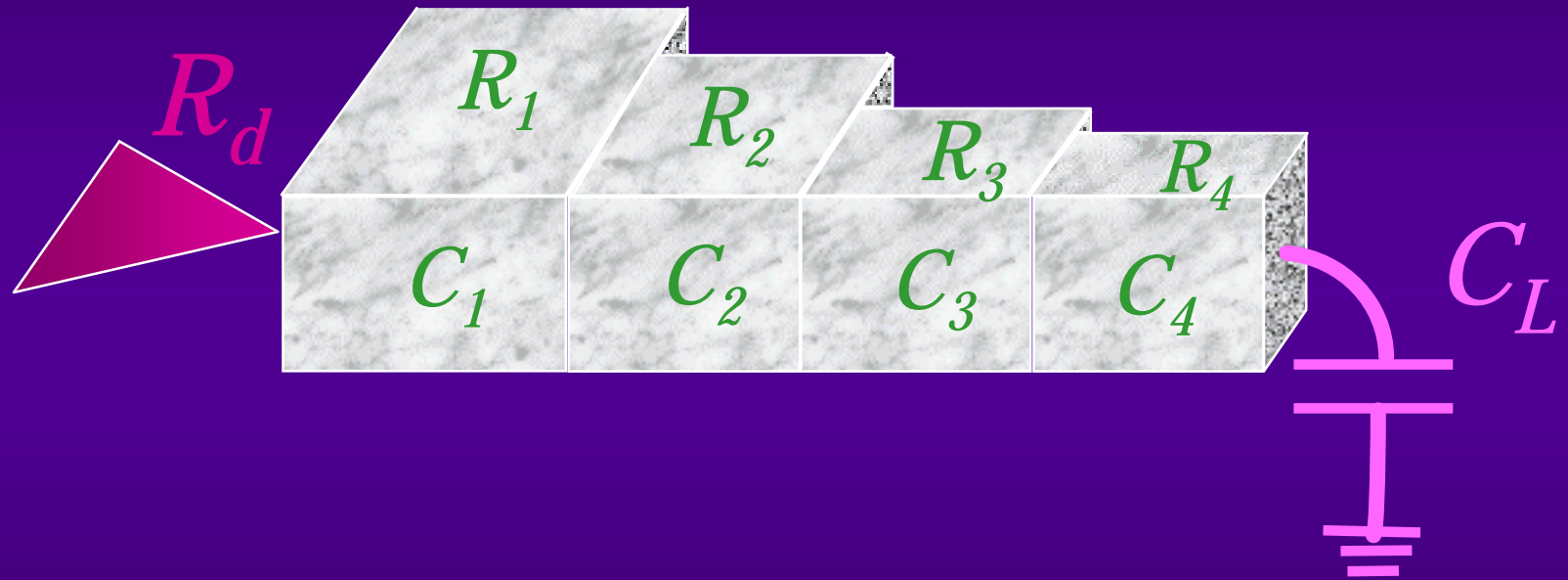
# Interconnect Model (area capacitance)



# *Driver Model*

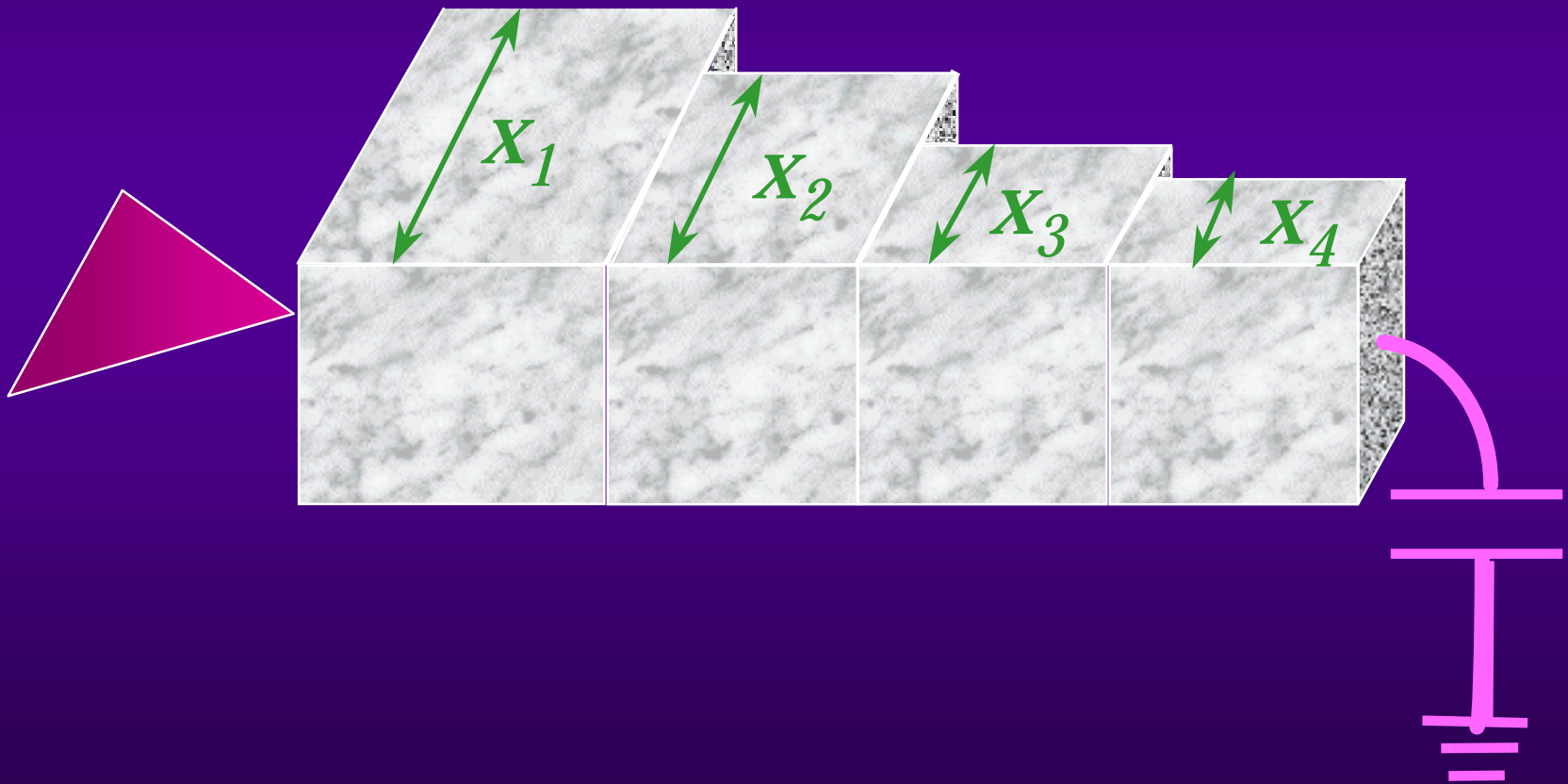


# Elmore Delay Computation

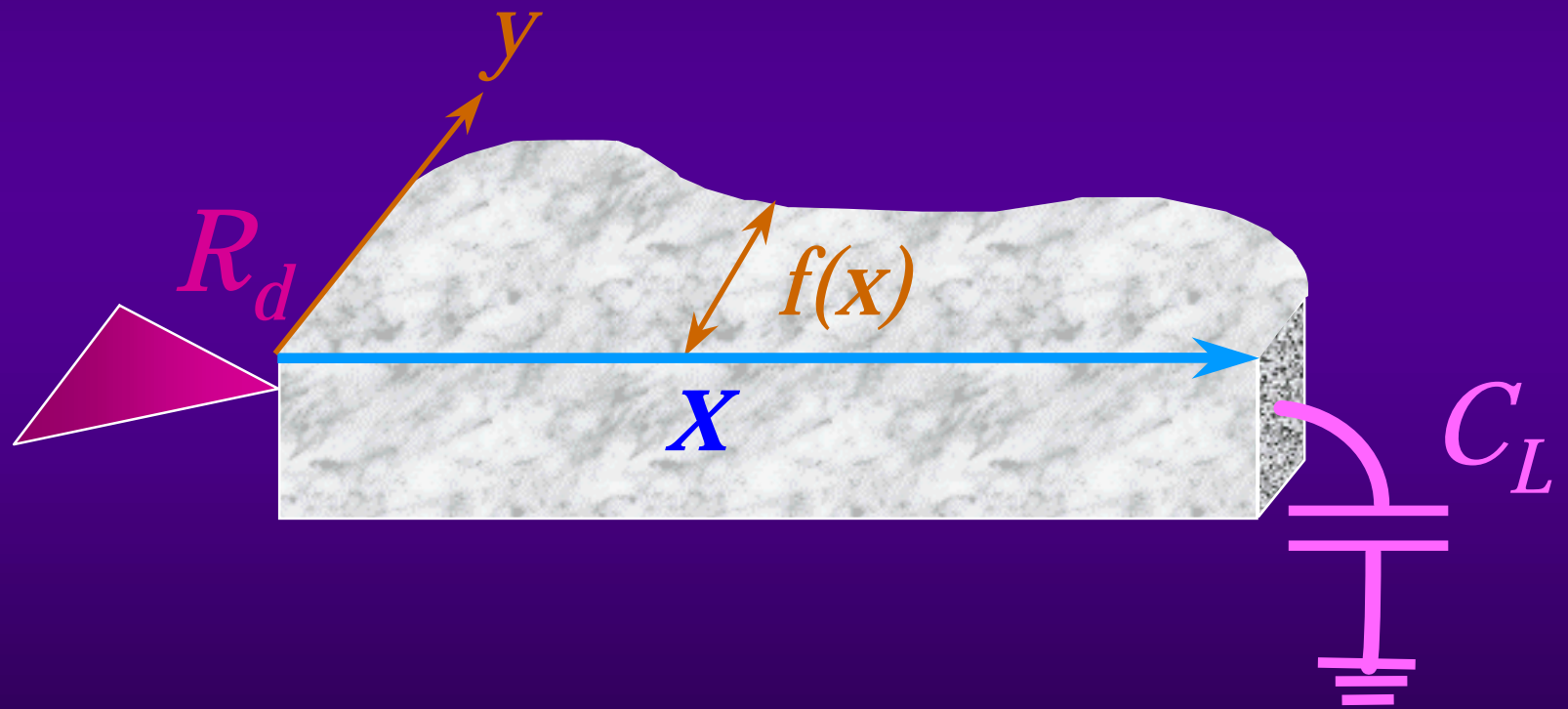


$$D = R_d(C_1 + C_2 + C_3 + C_4 + C_L) + R_1(C_1/2 + C_2 + C_3 + C_4 + C_L) + R_2(C_2/2 + C_3 + C_4 + C_L) + R_3(C_3/2 + C_4 + C_L) + R_4(C_4/2 + C_L)$$

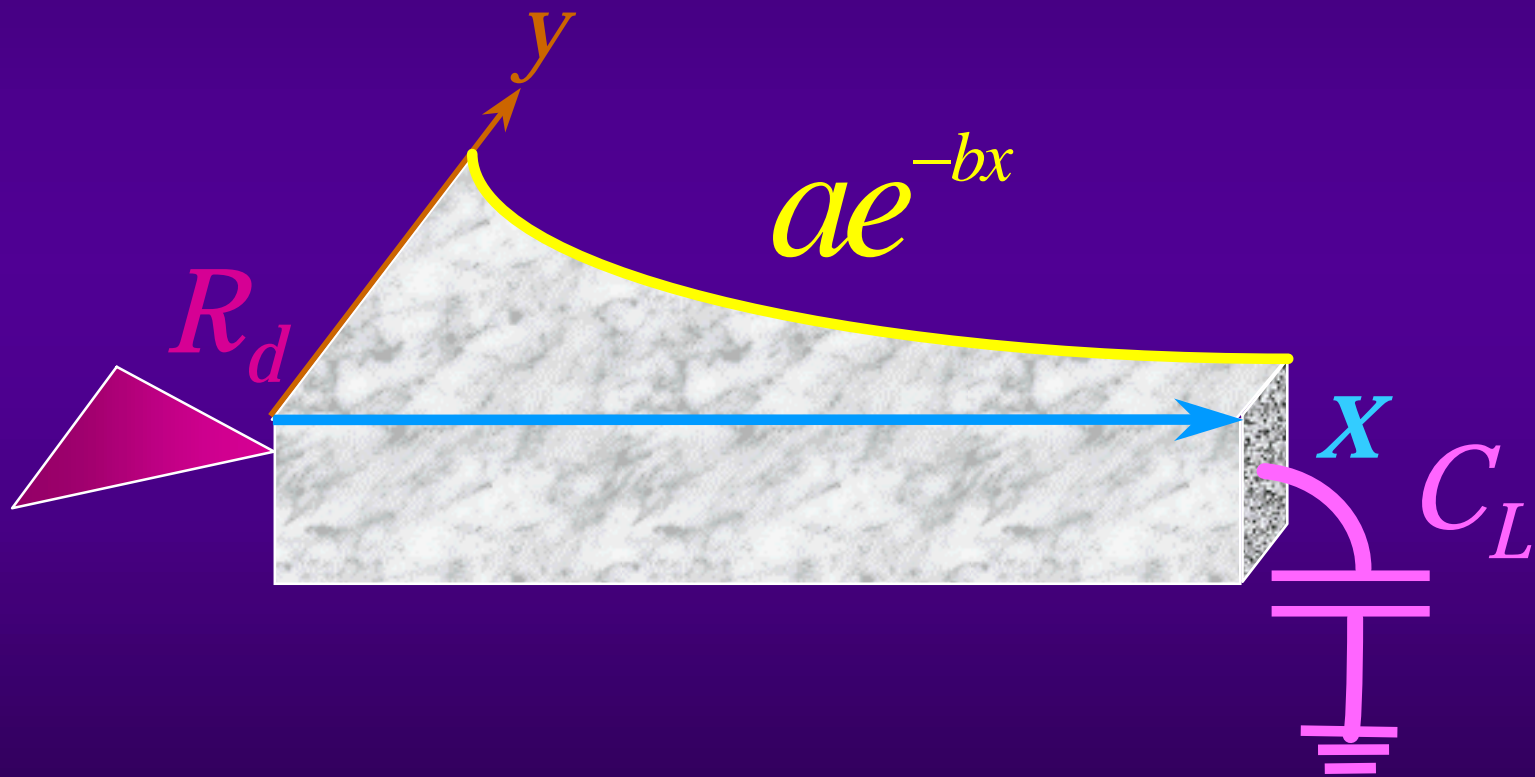
# *Uniform Wire-Sizing*



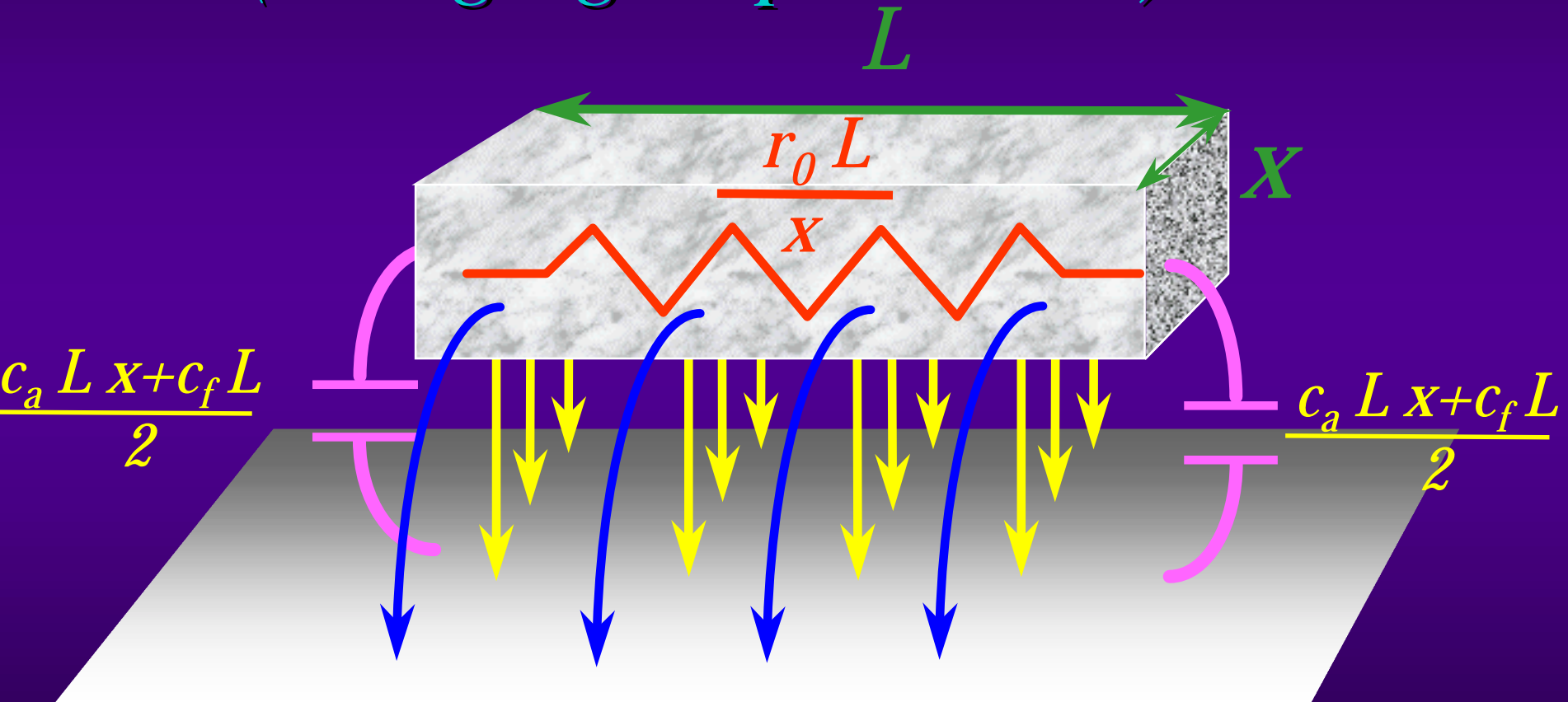
# *Non-uniform Wire-Sizing*



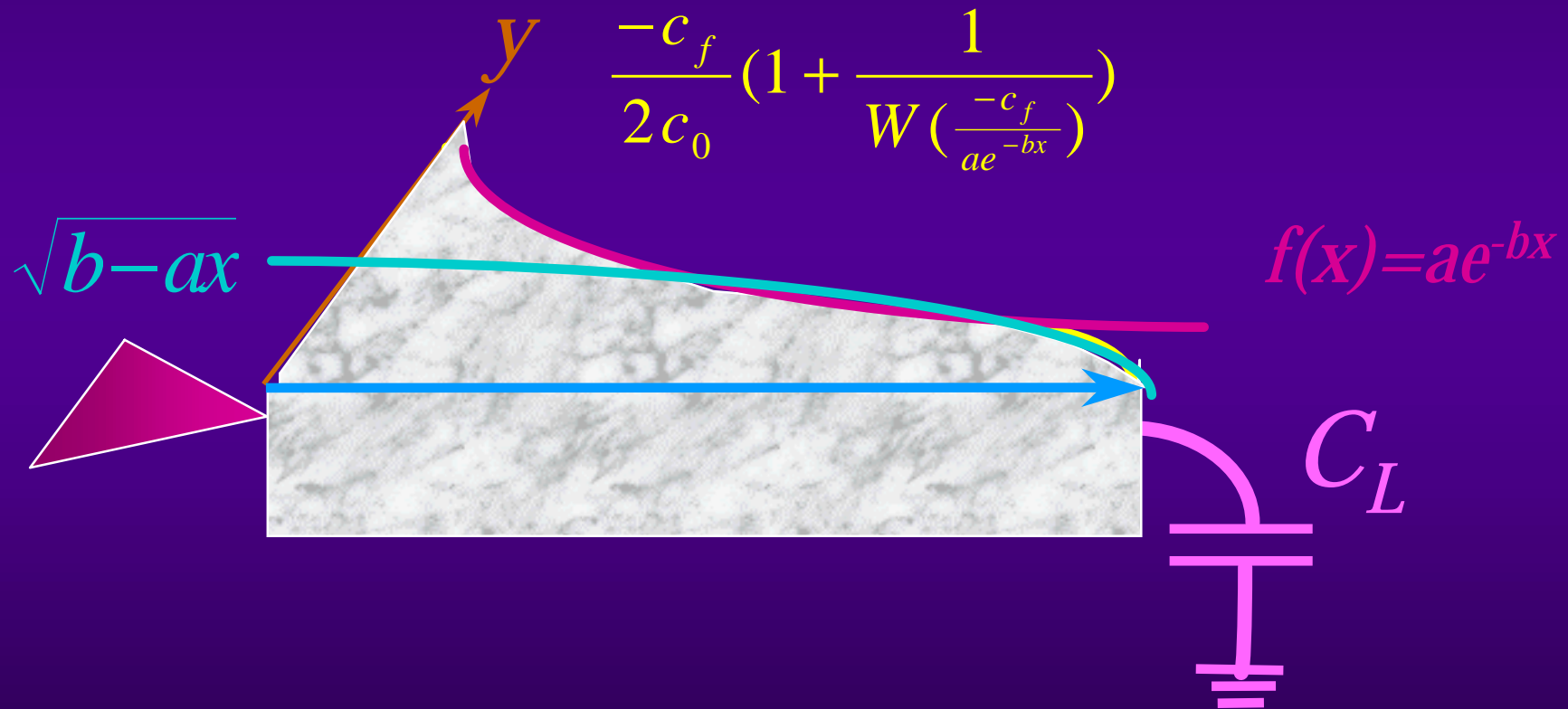
# *Optimal Wire-Sizing Function: Exponential Tapering*



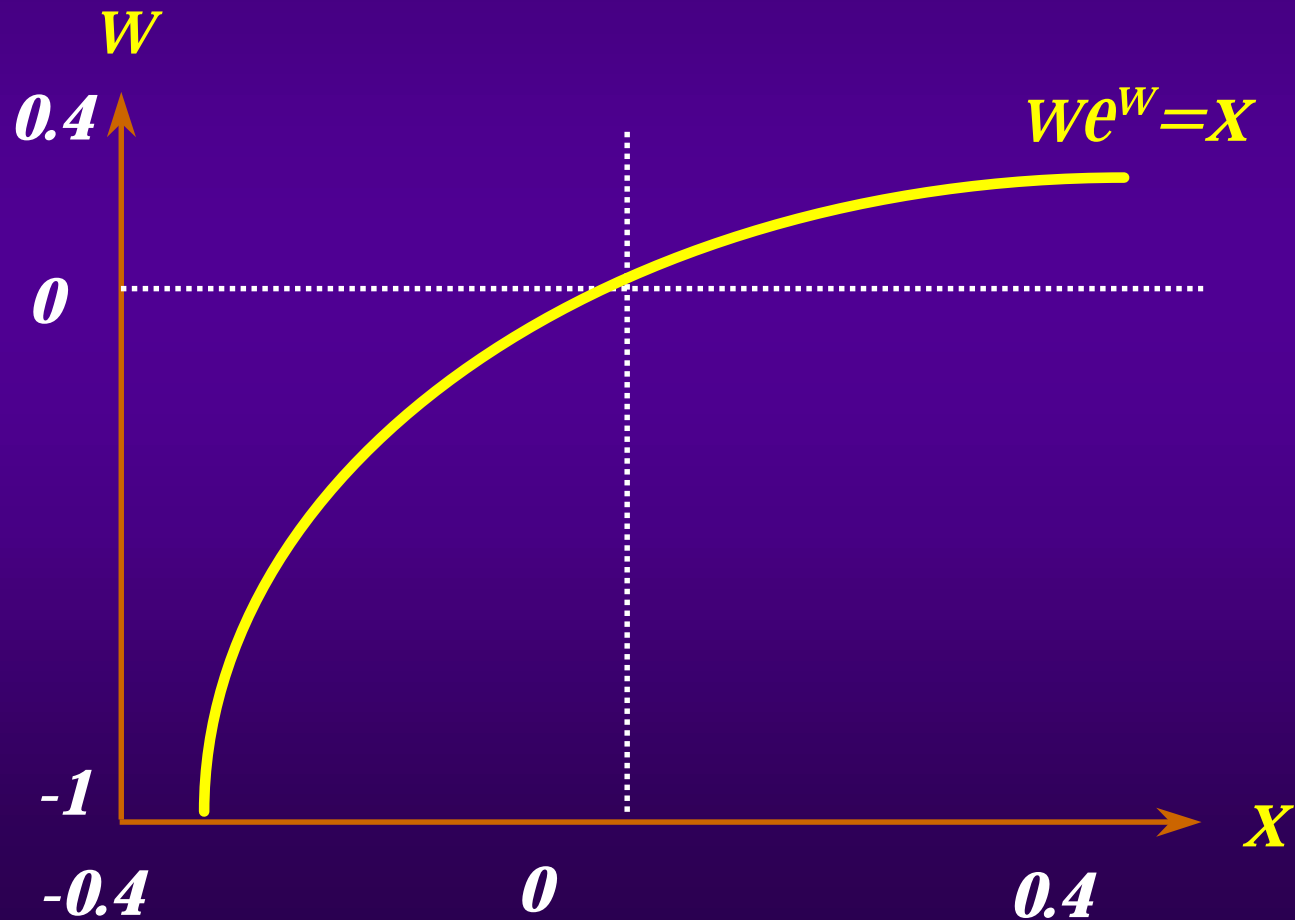
# Interconnect Model (Fringing Capacitance)



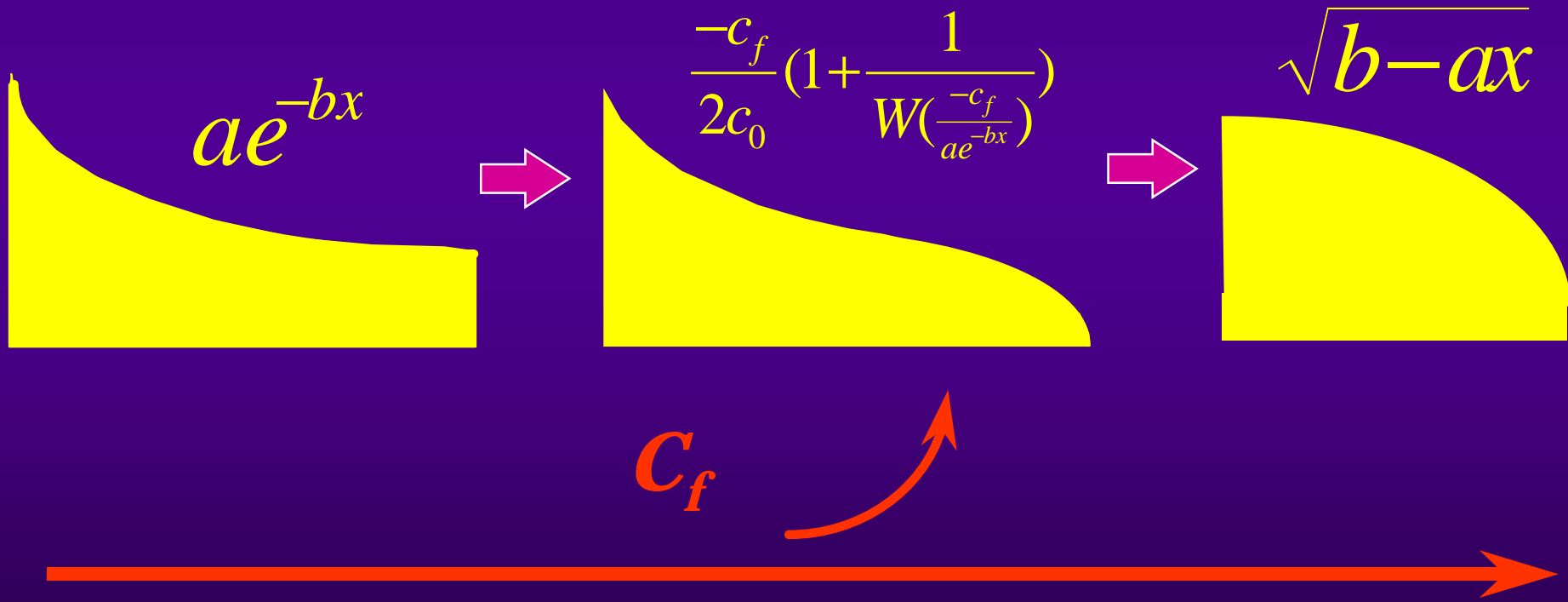
# Optimal Wire-Sizing Function: *W-function tapering*



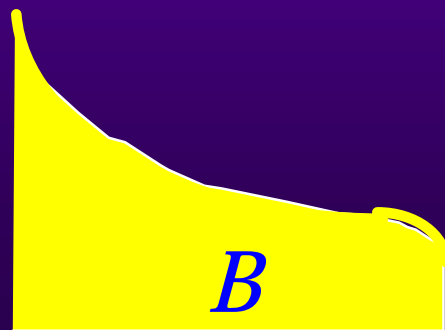
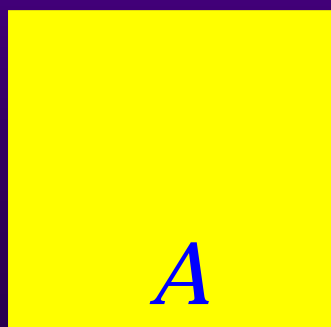
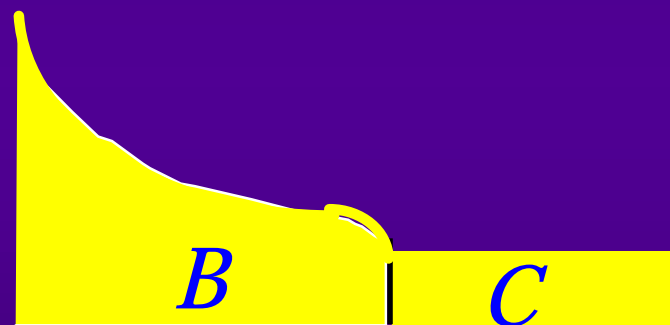
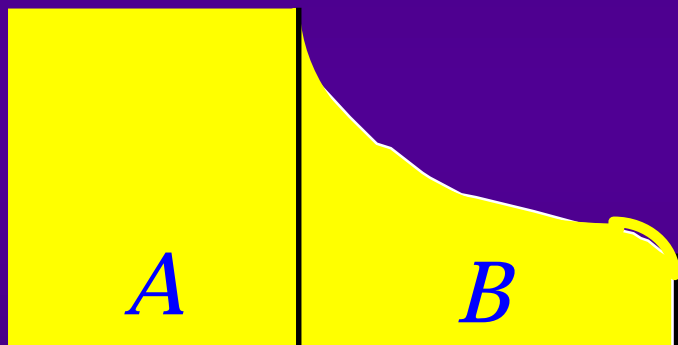
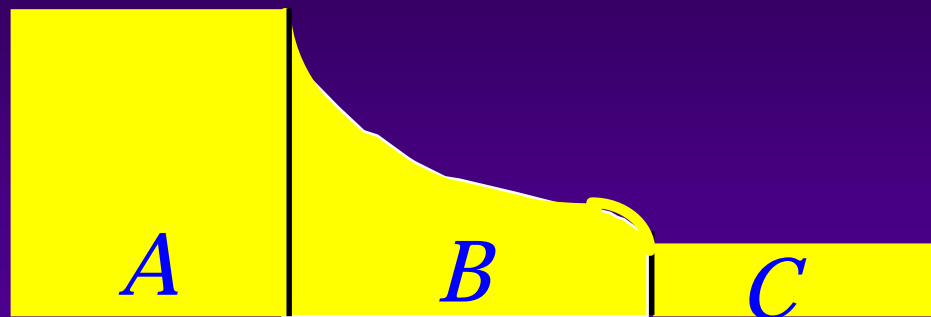
# Lambert's $W$ function



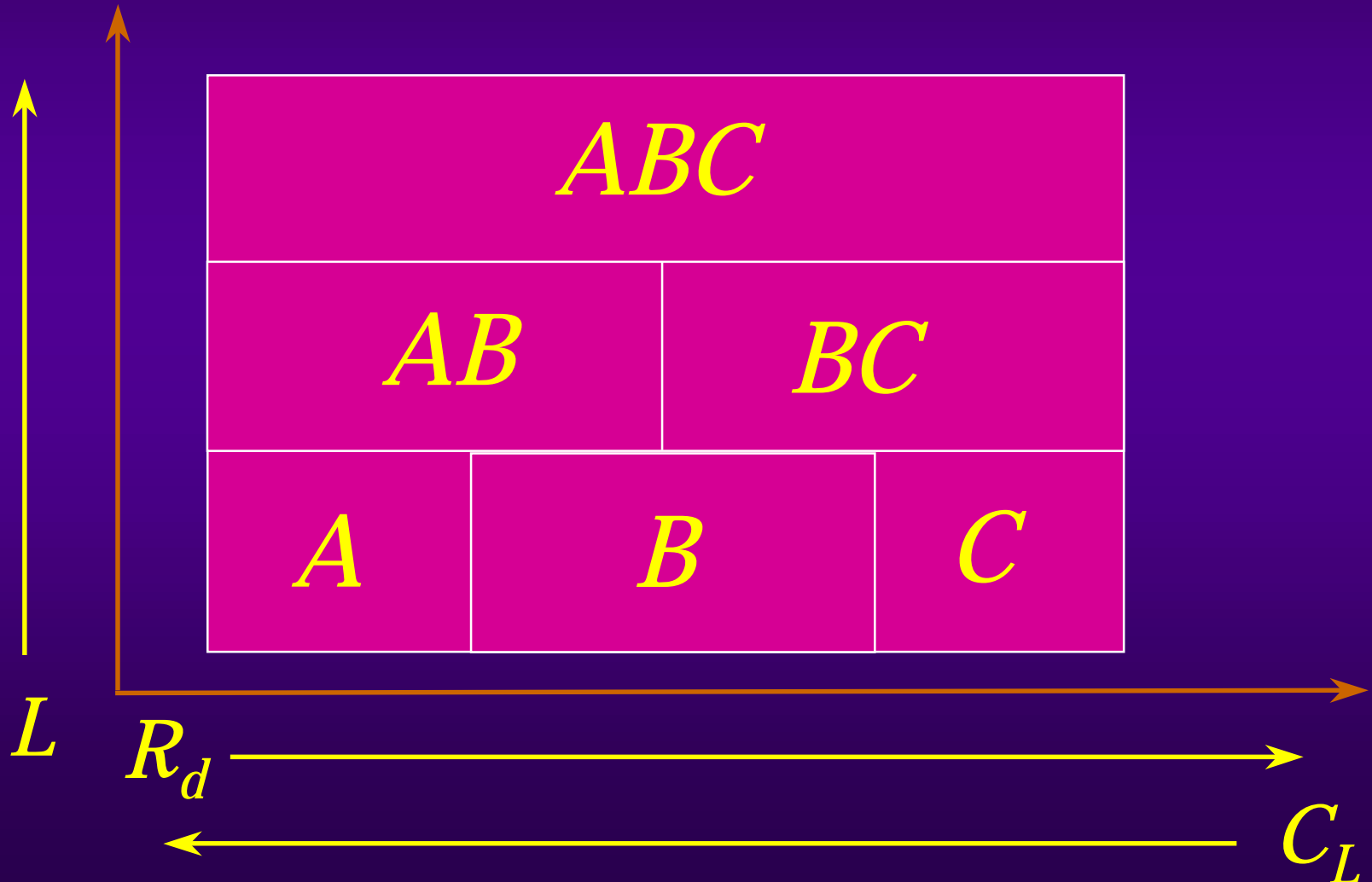
# *Optimal Wire-Sizing Function*



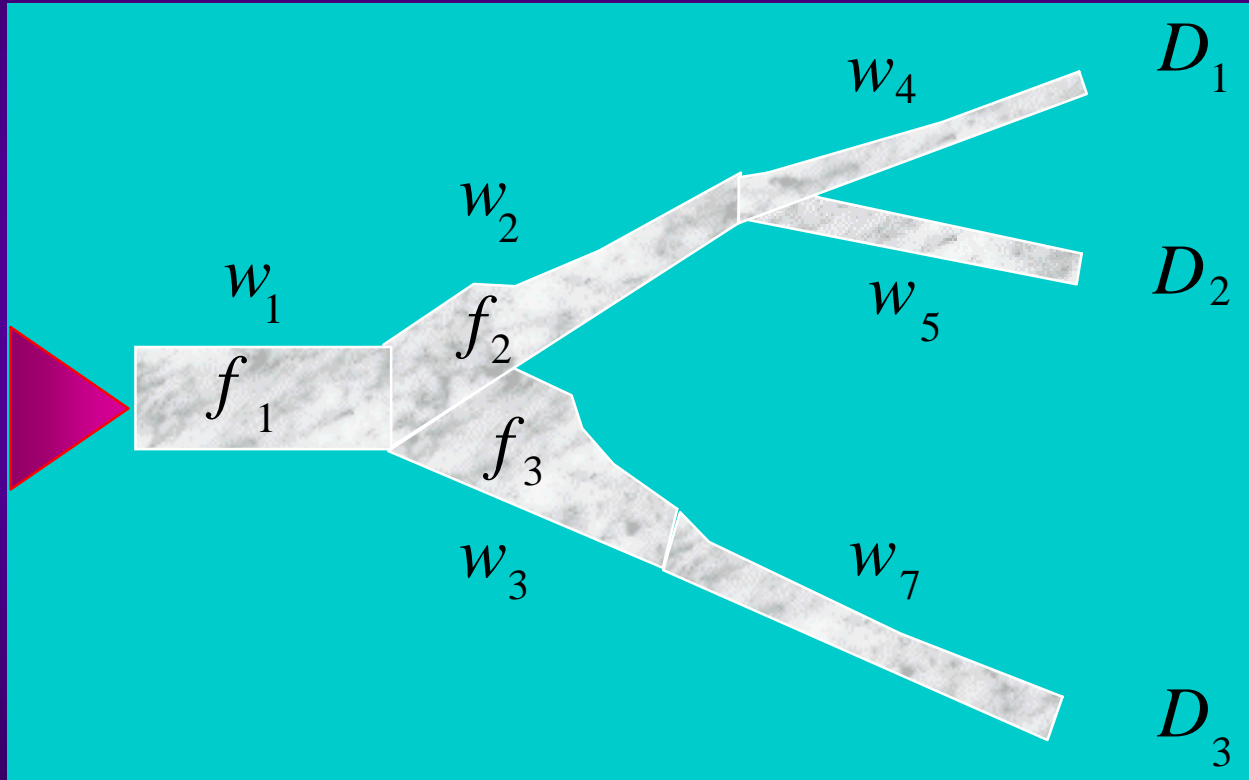
# *Constrained Wire-Sizing* $L \leq f(x) \leq U$



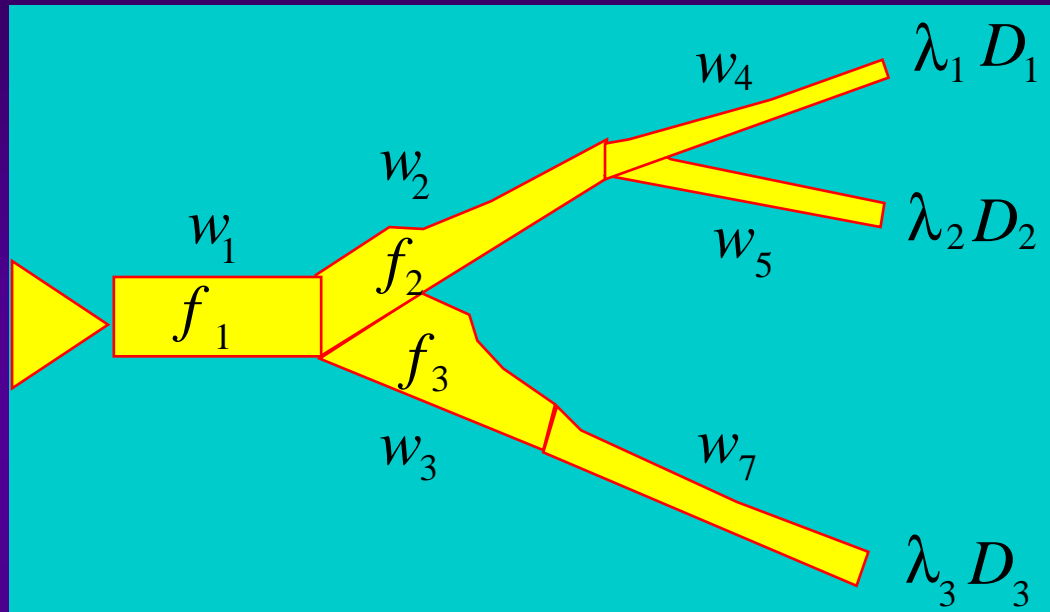
# *Relations among the six types of functions*



# Wire-Sizing for Routing Trees



# Weighted Sink Delay Optimization



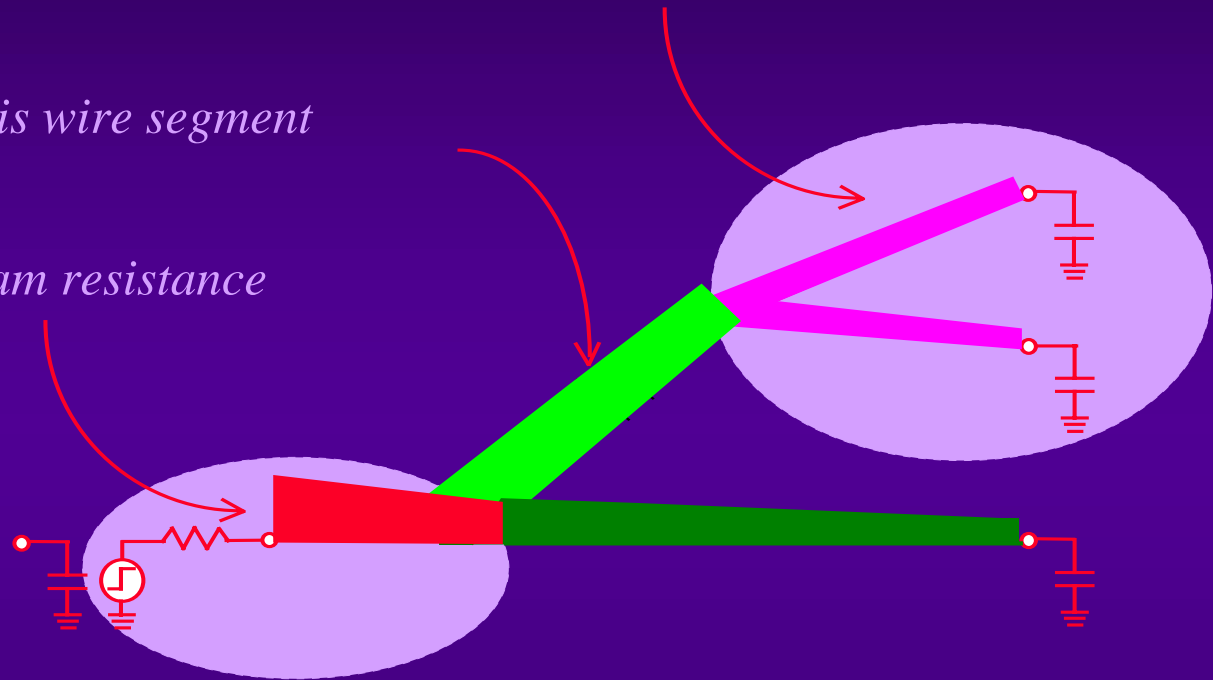
$$\begin{aligned} \text{Minimize} \quad & D(f) = \sum_{i=1}^s \lambda_i D_i \\ \text{Subject to} \quad & L_i \leq f_i \leq U_i, \quad 1 \leq i \leq n, \\ \text{where} \quad & \sum_{i=1}^s \lambda_i = 1. \end{aligned}$$

# Optimally Resizing One Segment

*downstream capacitance*

*while processing this wire segment*

*weighted upstream resistance*

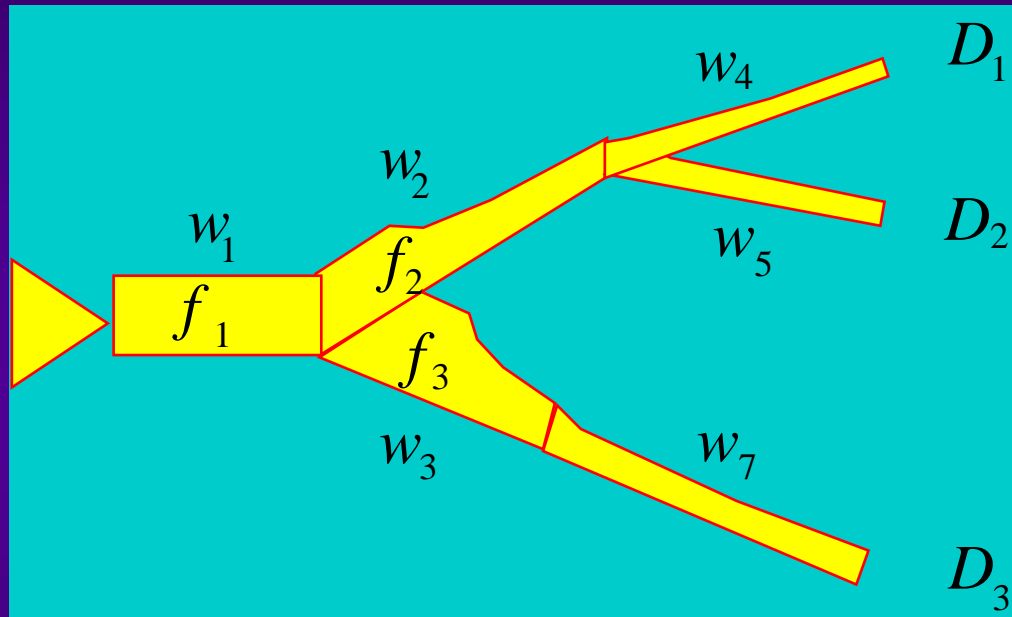


**Theorem:**

For each  $w_i$ , the optimal wire - sizing formula for a single line can be used to optimally resize  $w_i$  by the following

transformation:  $R_d = R_i, C_L = C_i, L = L_i, \sigma = \mu_i \sigma$ .

# Minimizing Area with Delay Constraints



*Minimize*

$A$

*Subject to*

$$D_i(\mathbf{f}) \leq B_i, \quad 1 \leq i \leq s,$$

$$L_i \leq f_i \leq U_i, \quad 1 \leq i \leq n.$$

# Minimizing Area with Delay Constraints

$$\begin{array}{ll} \text{Minimize} & A \\ \text{Subject to} & D_i(\mathbf{f}) \leq B_i, \quad 1 \leq i \leq s, \\ & L_i \leq f_i \leq U_i, \quad 1 \leq i \leq n. \end{array}$$



## Lagrangian Relaxation Subproblem

$$\begin{array}{ll} \text{Minimize} & D'(\mathbf{f}) = A + \sum_{i=1}^s \lambda_i (D_i(\mathbf{f}) - B_i) \\ \text{Subject to} & L_i \leq f_i \leq U_i, \quad 1 \leq i \leq n. \end{array}$$

# *Minimizing Area with Delay Constraints*

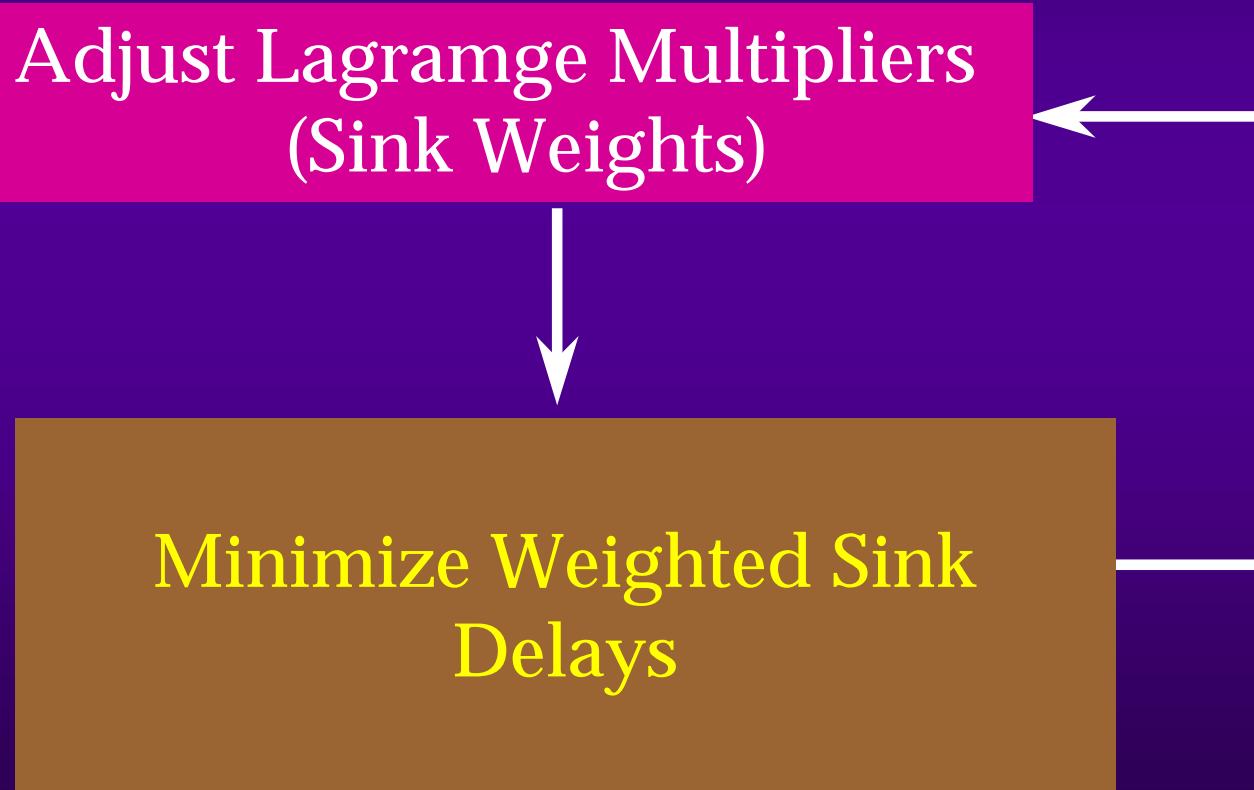
$$\begin{array}{ll} \text{Minimize} & D'(f) = A + \sum_{i=1}^s (\lambda_i D_i(f) - B_i) \\ \text{Subject to} & L_i \leq f_i \leq U_i, \quad 1 \leq i \leq n. \end{array}$$

**Theorem:**

Solving this Lagrangian subproblem is equivalent to solving the weighted sink delay problem with

driver resistance changed to  $R_d + \frac{1}{\xi\lambda}$ , where  $\lambda = \sum_{i=1}^s \lambda_i$ .

# *Algorithm Framework*



# *Minimizing Maximum Delay*

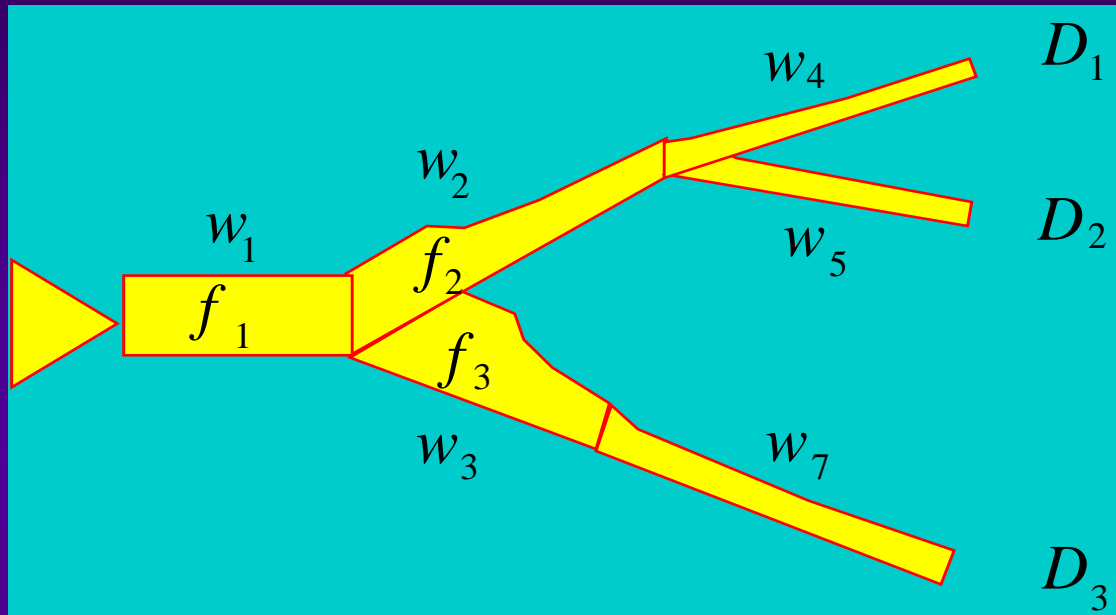
$$\begin{array}{ll} \text{Minimize} & D_{\max} \\ \text{Subject to} & D_i(\mathbf{x}) \leq D_{\max}, \quad 1 \leq i \leq s, \\ & L_i \leq f(x_i) \leq U_i, \quad 1 \leq i \leq n, \\ & D_{\max} > 0. \end{array}$$



*Lagrangian Relaxation Subproblem*

$$\begin{array}{ll} \text{Minimize} & D_{\max} + \sum_{i=1}^s \lambda_i (D_i - D_{\max}) \\ \text{Subject to} & L_i \leq f(x_i) \leq U_i, \quad 1 \leq i \leq n, \\ & D_{\max} > 0. \end{array}$$

# Delay/Power/Area Minimization



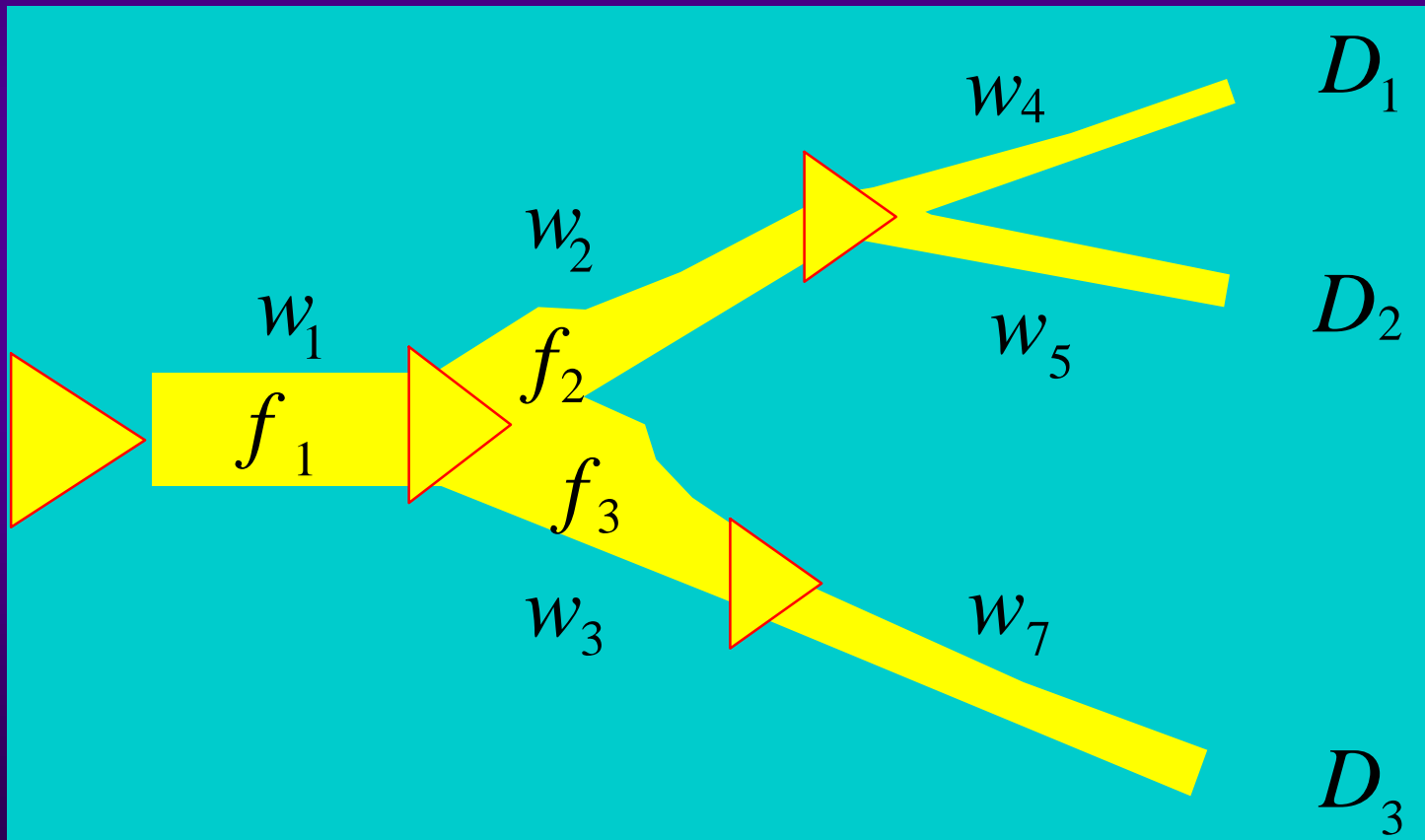
Minimize  $\alpha D_{\max} + \beta \text{Power} + \gamma \text{Area}$

Subject to  $D_i(\mathbf{x}) \leq D_{\max}, 1 \leq i \leq s,$

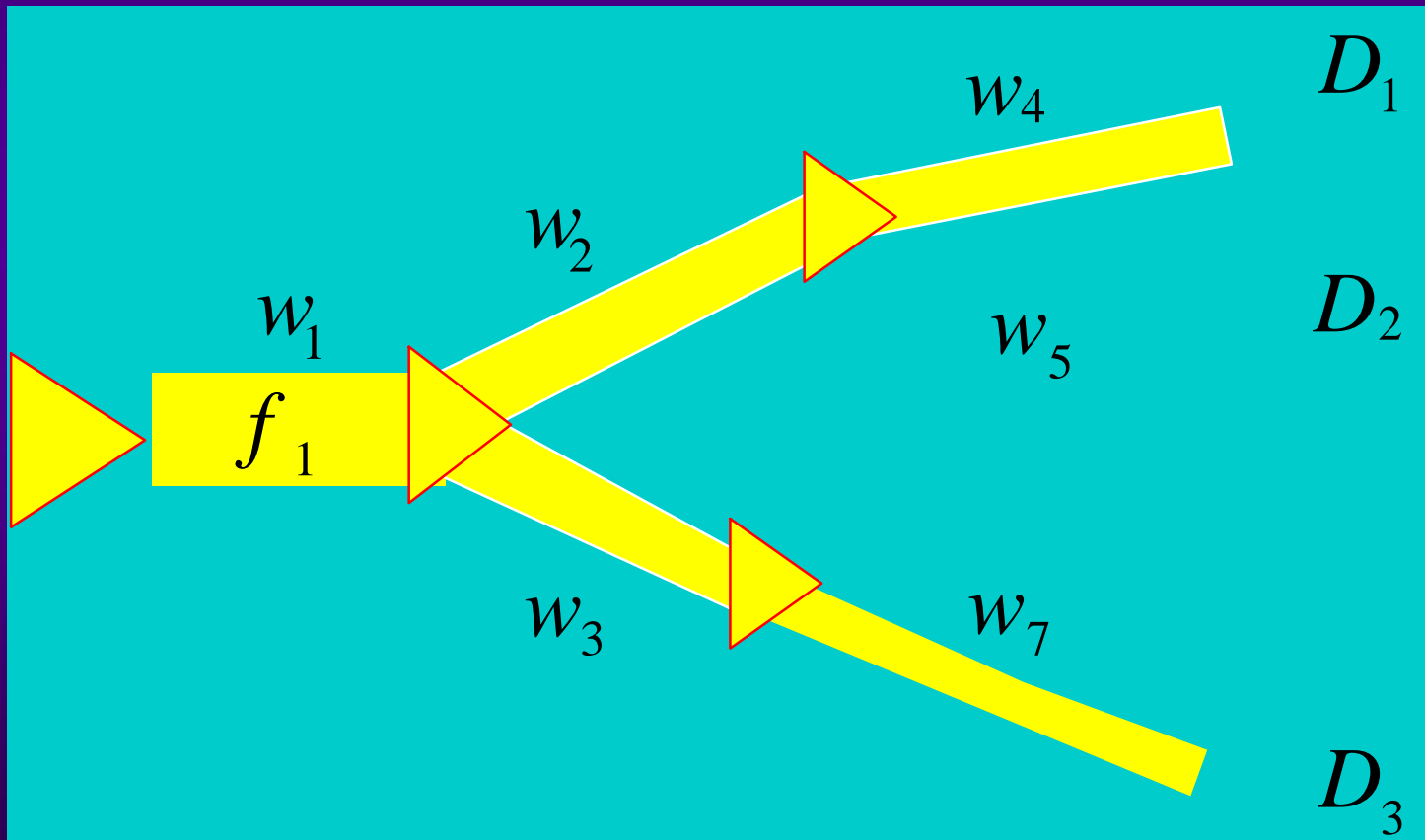
$L_i \leq x_i \leq U_i, 0 \leq i \leq n + m,$

$D_{\max} > 0.$

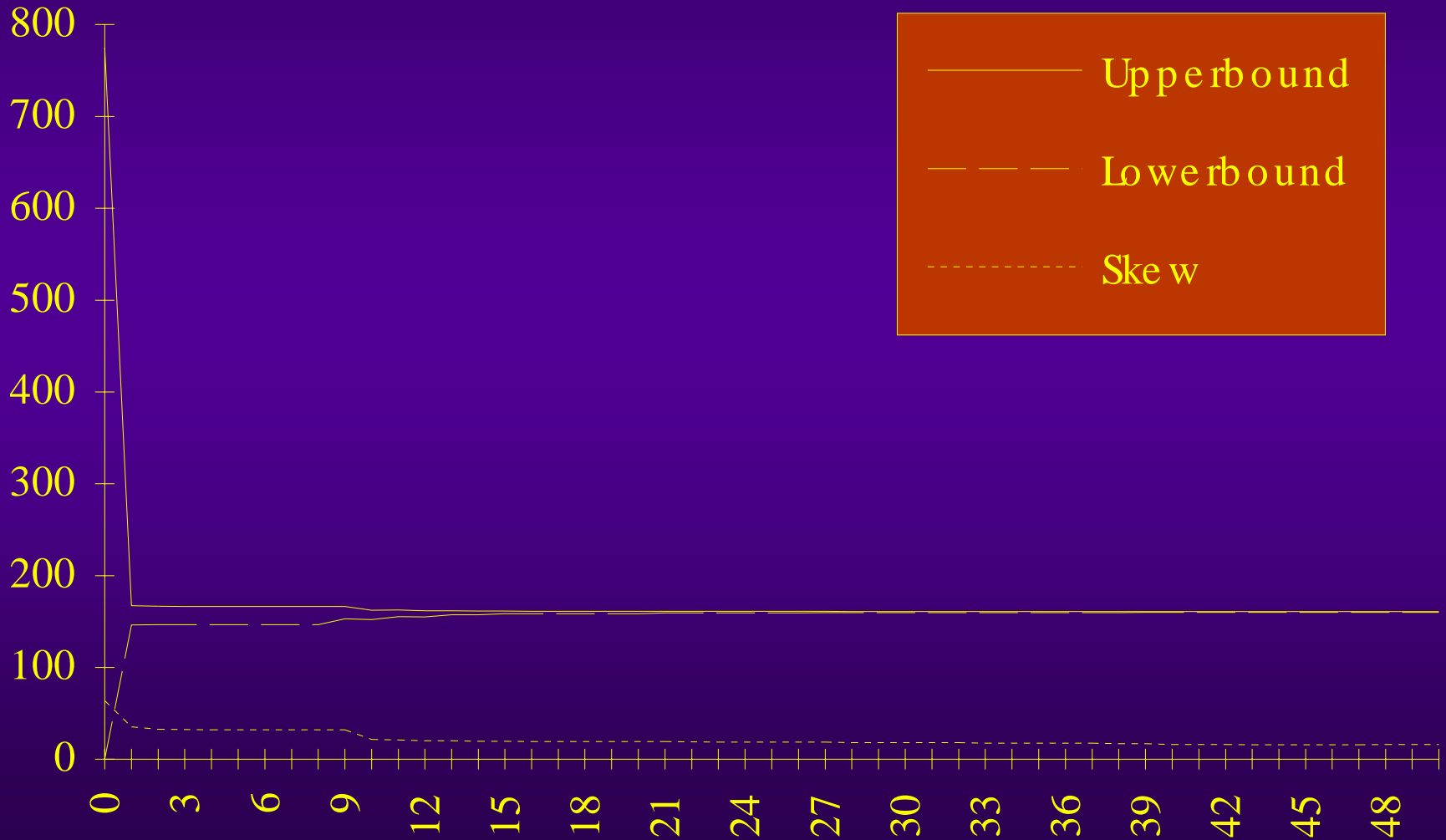
# *Simultaneous Buffer and Wire-Sizing*

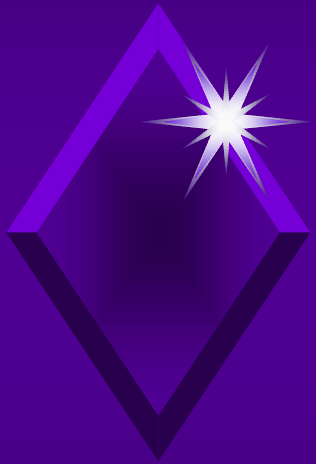


# Uniform Wire-Sizing



# *Upperbound, Lowerbound, Skew*

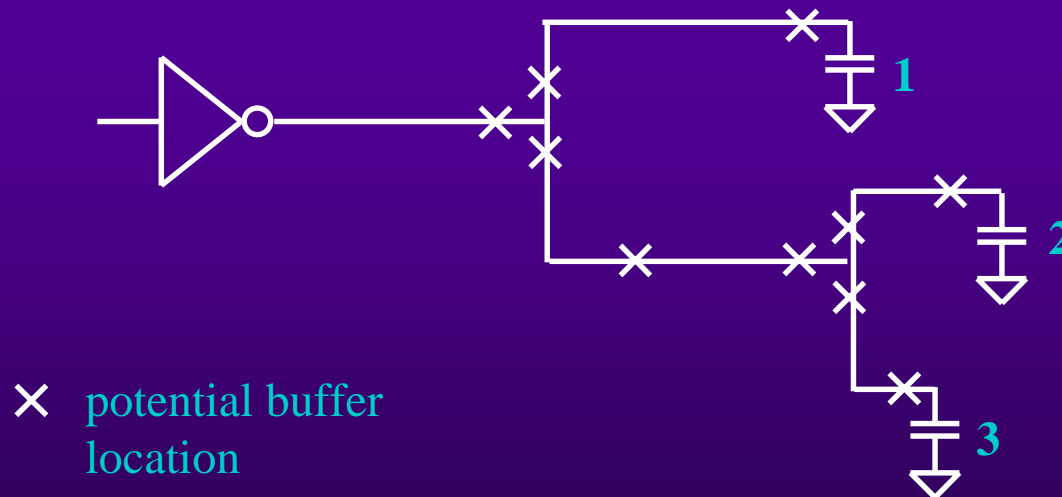




# *Buffer Insertion*

# Spec-based Buffer Insertion

- Given a routing tree, possible buffer insert location, required arrival times at receivers, max slope constraint, and polarity requirement
- A library of buffers:  $B_1, B_2, \dots, B_n$
- Insert buffers to satisfy the spec (maximum delay, delay bounds at each receiver and maximum slope)



# *Problem formulation*

- Combinations of the following:
  - Buffer Insertion
  - Buffer-Sizing
  - Wire-Sizing
- **Goals and Constraints**
  - Minimize the Maximum Delay
  - Satisfy delay constraints at each receiver
  - Repeater Insertion Location Constraints
  - Maximum Slope constraint
  - Polarity constraints.

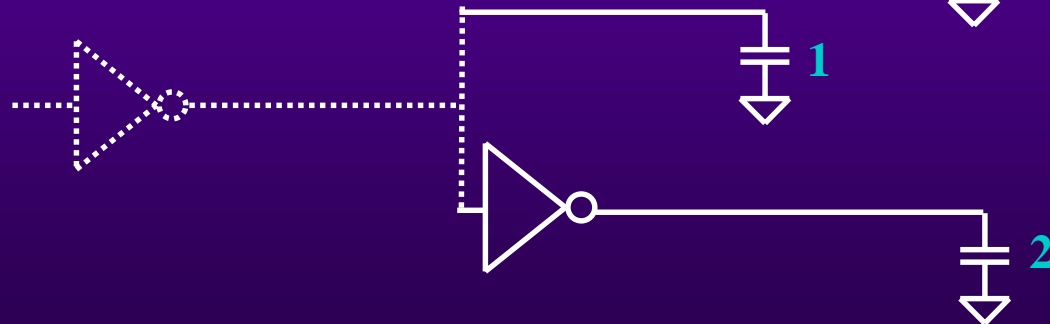
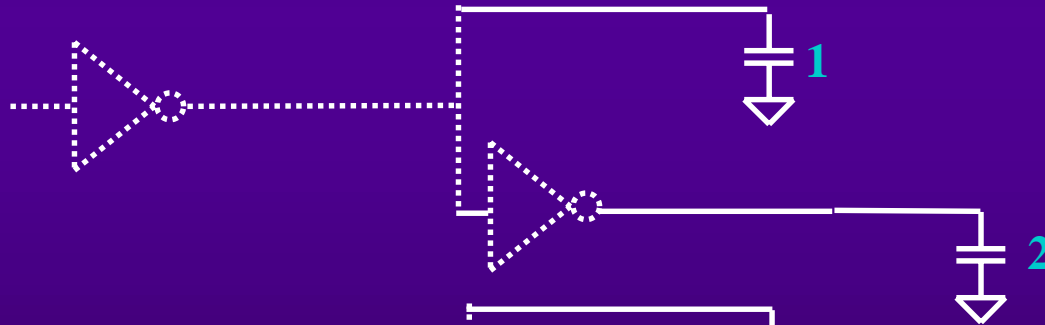
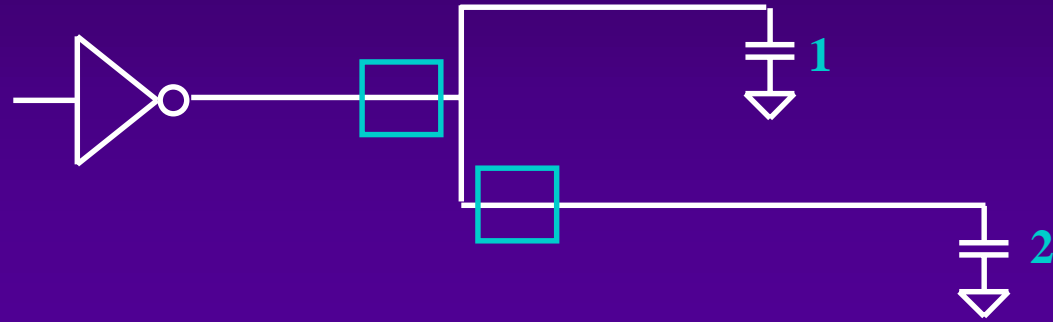
# *Current Issues*

- Previous solutions
  - Exhaustive enumeration-> **Exponentially Growing**
  - First Ginneken, and then, Lillis, suggested a dynamic programming approach which can get optimal solution for delay under the Elmore delay model .
  - Provides very useful information like **power-delay curves**
  - Problems: **not accurate, doesn't consider reliability issues, runtime and storage already high.**

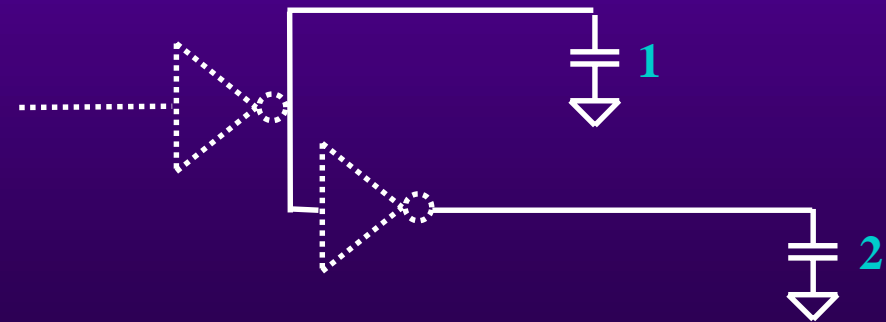
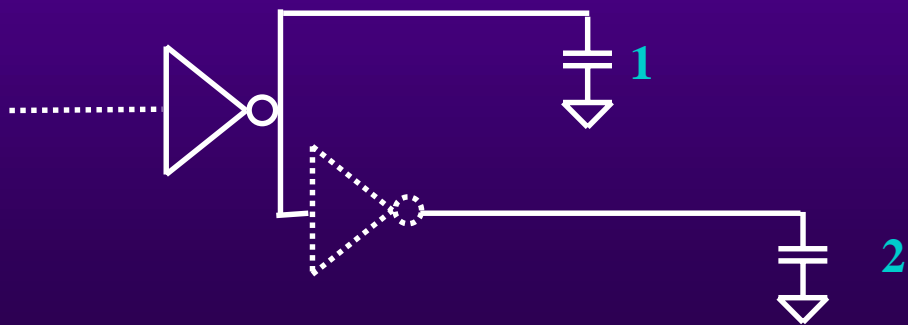
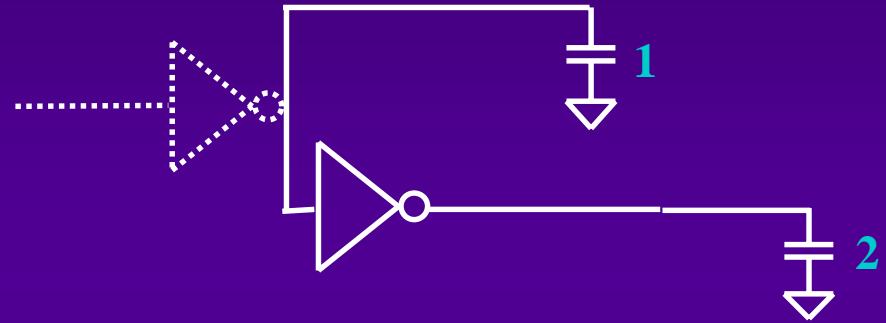
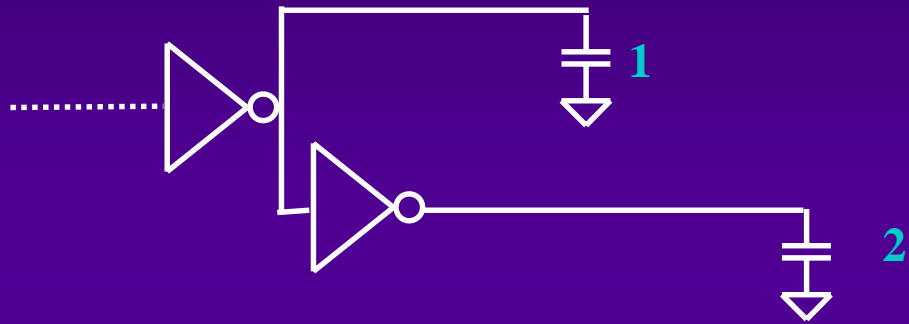
# *Brief Algorithm Description*

- **Algorithm:**
  - Traverse circuit in a bottom-up manner
  - Enumerate all the possible solutions and prunes out sub-optimal solutions dynamically.
- **How do we know which solution to kill?**
  - Violate the constraints
  - If there is another solution cost less and achieve more in all aspects?
    - Number of buffers
    - Maximum delay
    - Polarity
    - Area, Power ...

# Example



# *Example*

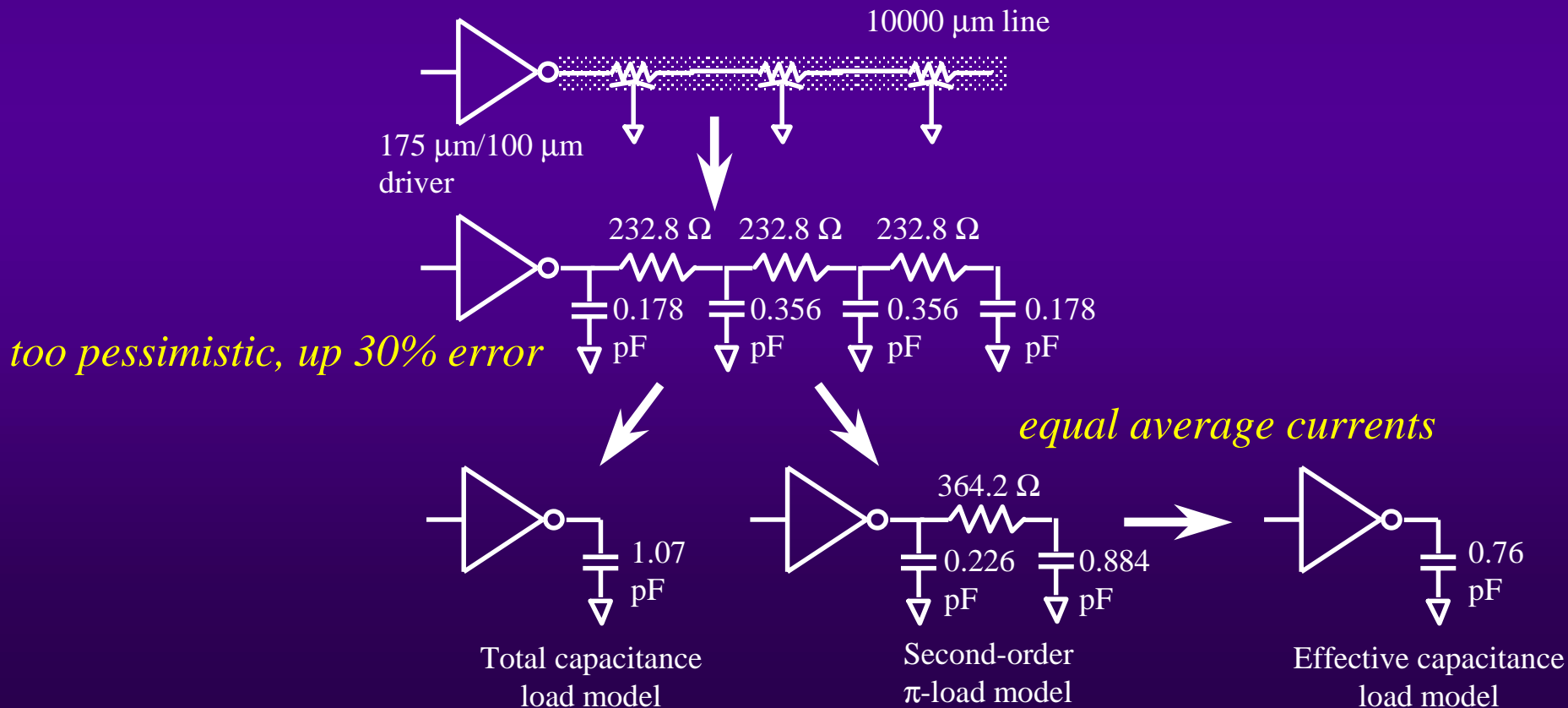


# *Problems*

- How to calculate the gate and interconnect delay accurately and efficiently?
  - A naive approach: Repeatedly calculate the delay of the subtree by calling AWE or SPICE (causing  $O(N)$  penalty for each solution). However, the runtime is already high (proportional to  $N^2$ ).
  - An efficiently hierarchical delay computation method is needed
- How to include slope into consideration?

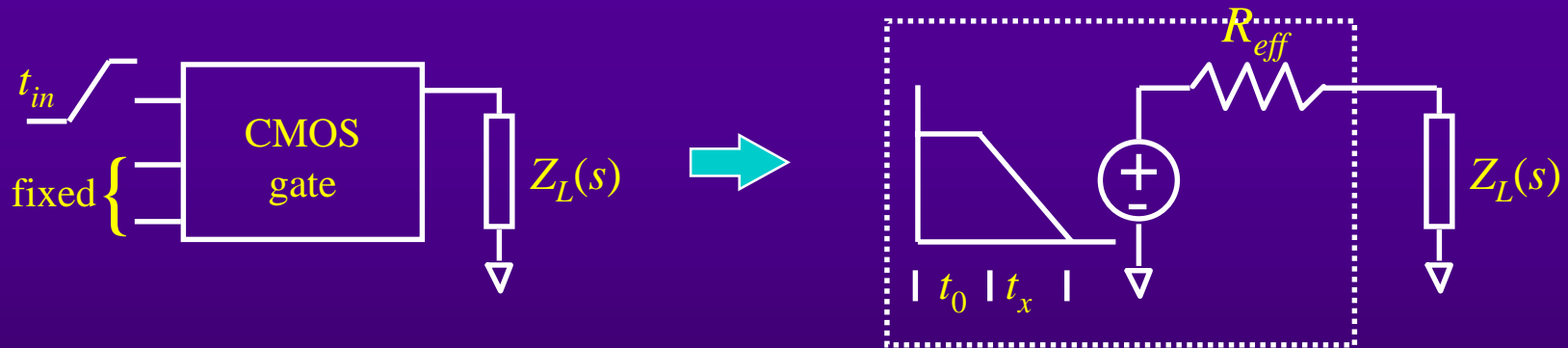
# Accurate Load Model (Effective Capacitance)

- The total-net capacitance is no longer a valid load model -- the second-order  $\pi$ -load driving-point admittance approximation is more accurate



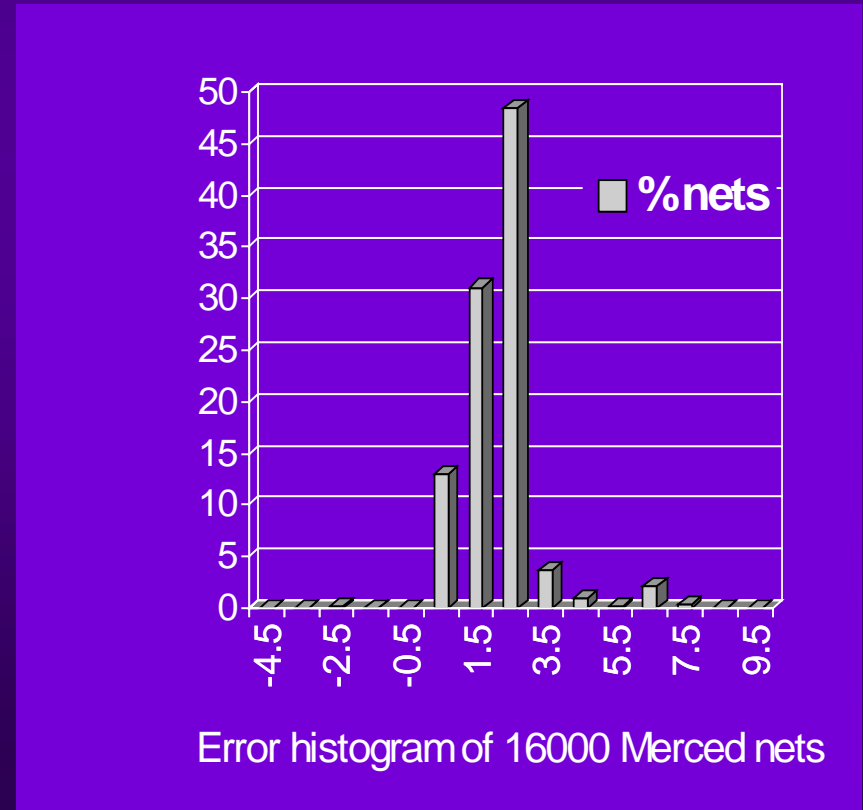
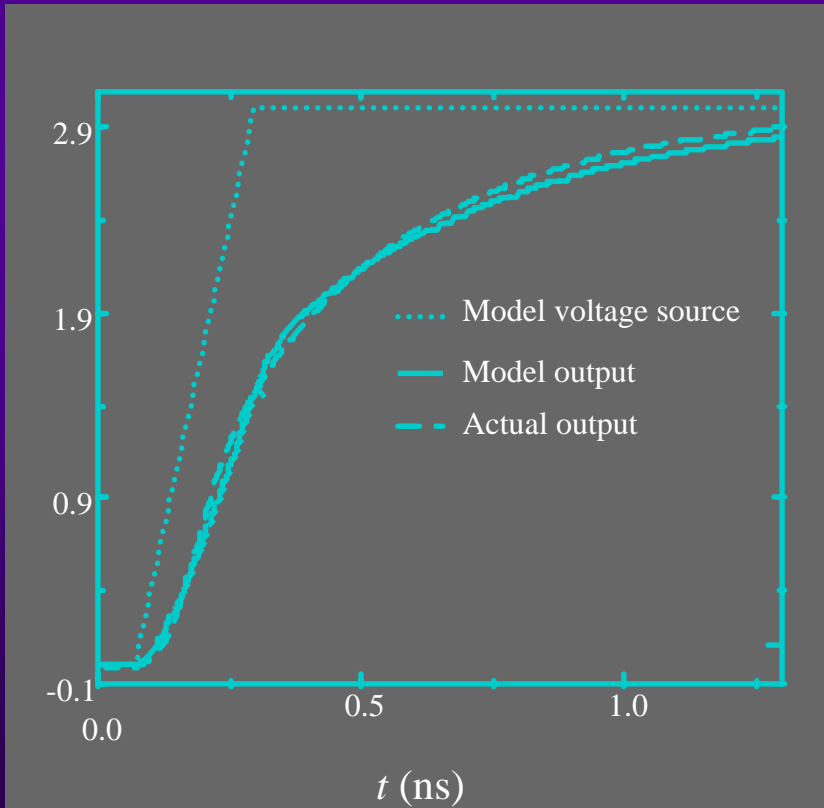
# Accurate Repeater Model (Voltage Ramp)

- Several timing analyzers model the gate by a single resistor
  - Errors of up to 30% have been reported
- The proposed gate delay model is a fixed-resistor driven by a ramp voltage source



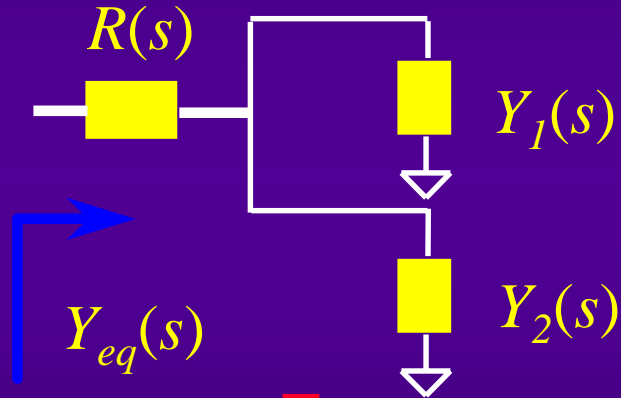
- Voltage ramp parameters,  $t_0$  and  $t_x$ , are determined from the Gate characteristic equations

# Accuracy of Voltage Ramp Model



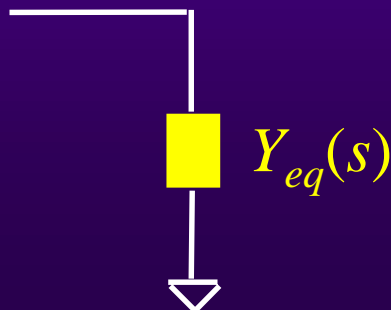
# How to calculate $\pi$ -load hierarchically

- There exists a simple way to calculate the  $\pi$  load (actually it can handle arbitrary higher order approximation) hierarchically.



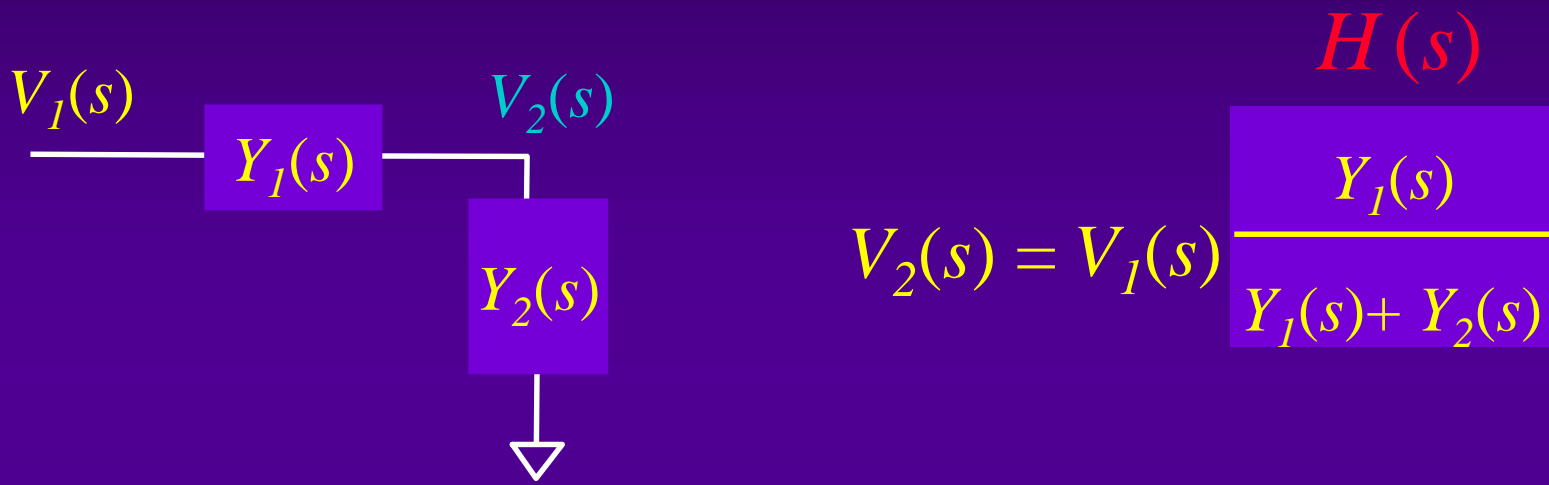
$$Y_{new}(s) = \frac{1}{R(s) + \frac{1}{Y_1(s) + Y_2(s)}}$$

Taylor Expansion:



$$Y_{eq}(s) = y_1s + y_2s^2 + y_3s^3 + y_4s^4 + y_5s^5 + y_6s^6$$

# What about wires?



$$V_2(s) = V_1(s) \frac{Y_1(s)}{Y_1(s) + Y_2(s)}$$

$$H(s) = \frac{Y_1(s)}{Y_1(s) + Y_2(s)}$$

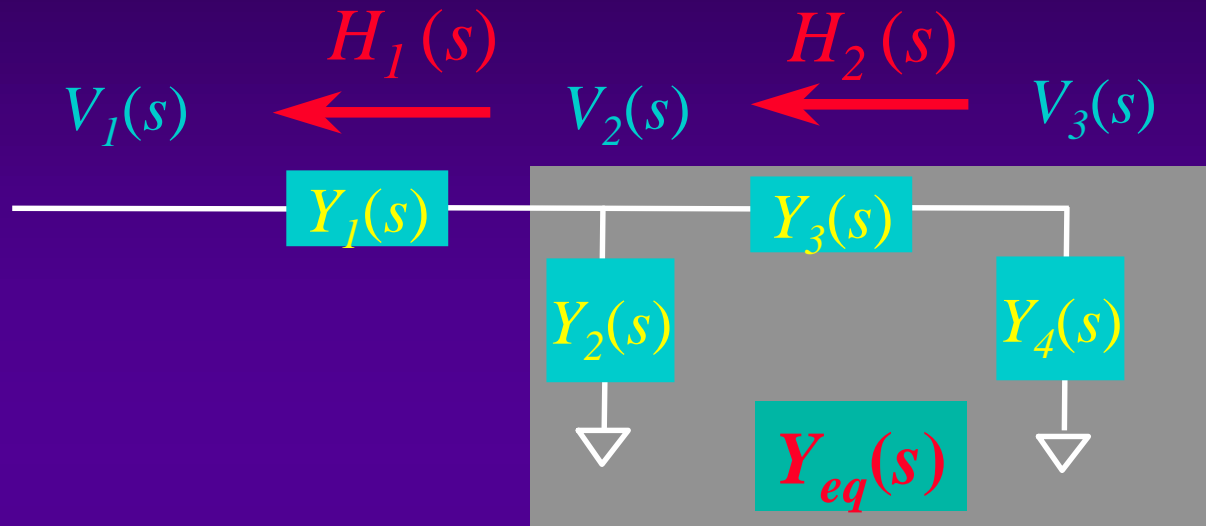
Taylor Expansion:



$$H(s) = m_0 + m_1s + m_2s^2 + m_3s^3 + m_4s^4 + m_5s^5 + m_6s^6$$

**Transfer function computation**

# Many Stages

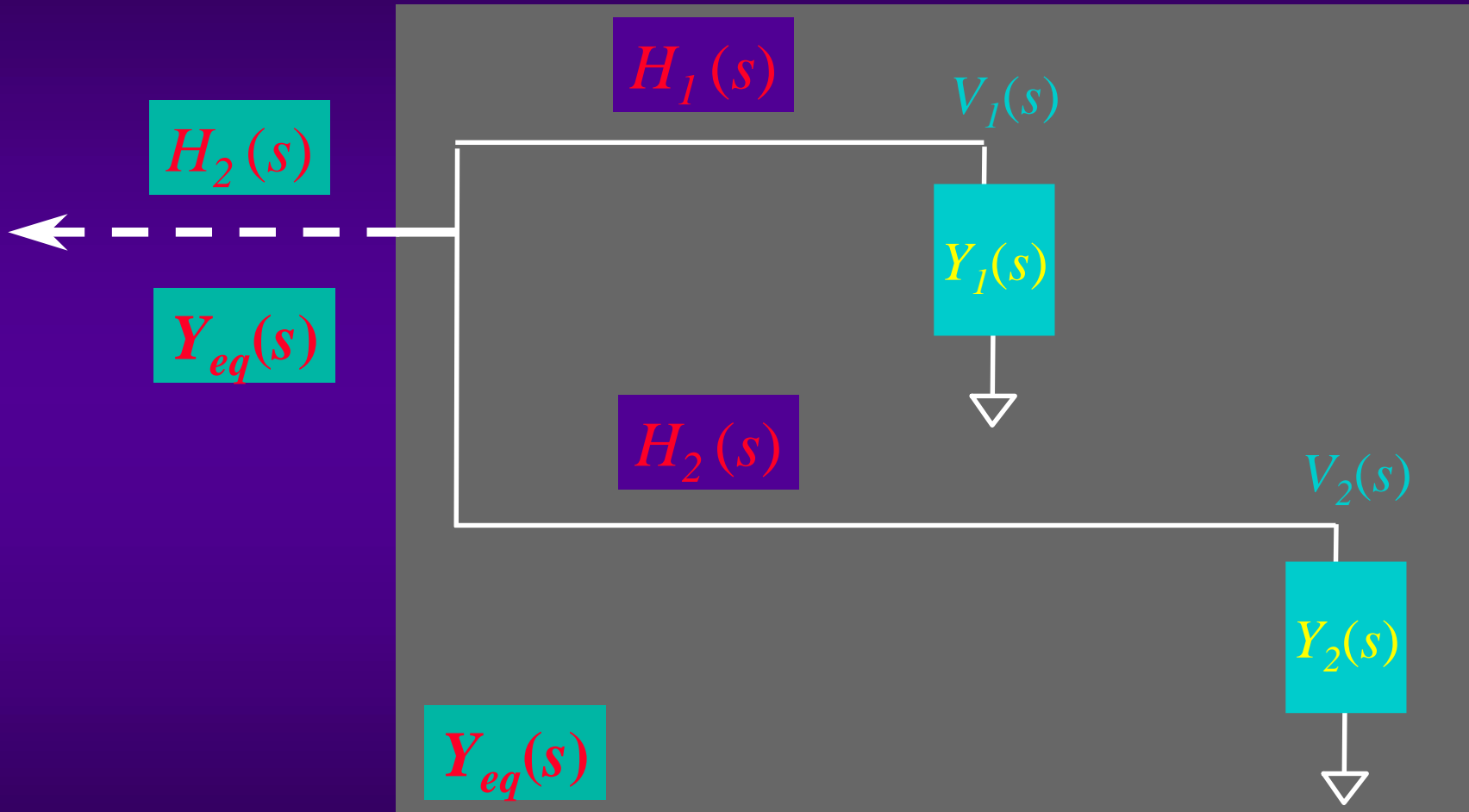


$$V_3(s) = V_1(s) \frac{Y_1(s)}{Y_1(s) + Y_{eq}(s)} \frac{Y_3(s)}{Y_3(s) + Y_4(s)}$$

$V_2(s)$

*Transfer function computation*

# What about Trees?



*Keep track the worse sink's transfer function*

# Hierarchical moment computation -- REX

- Assume in general

- $H(s) = 1/[b_0 + b_1s + b_2s^2 + b_3s^3 + b_4s^4 + b_5s^5]$
- $Y(s) = y_1s + y_2s^2 + y_3s^3 + y_4s^4 + y_5s^5 + y_6s^6$

- Across a capacitor

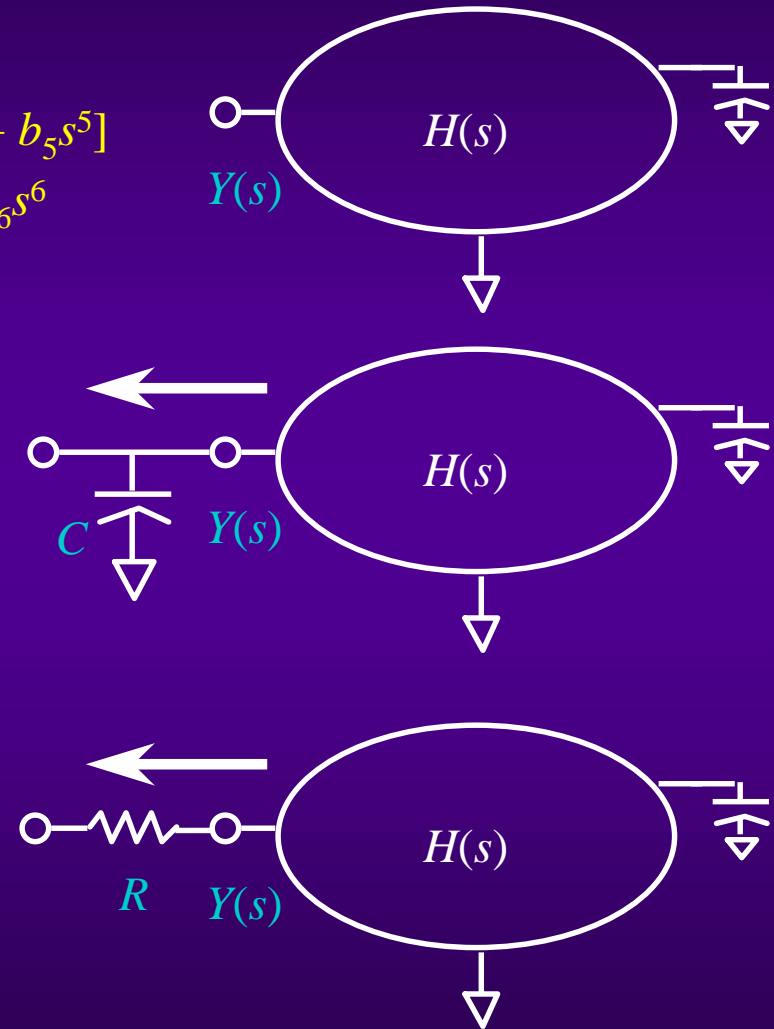
- $H'(s) = H(s)$
- $Y'(s) = Y(s) + Cs$

- Across a resistor

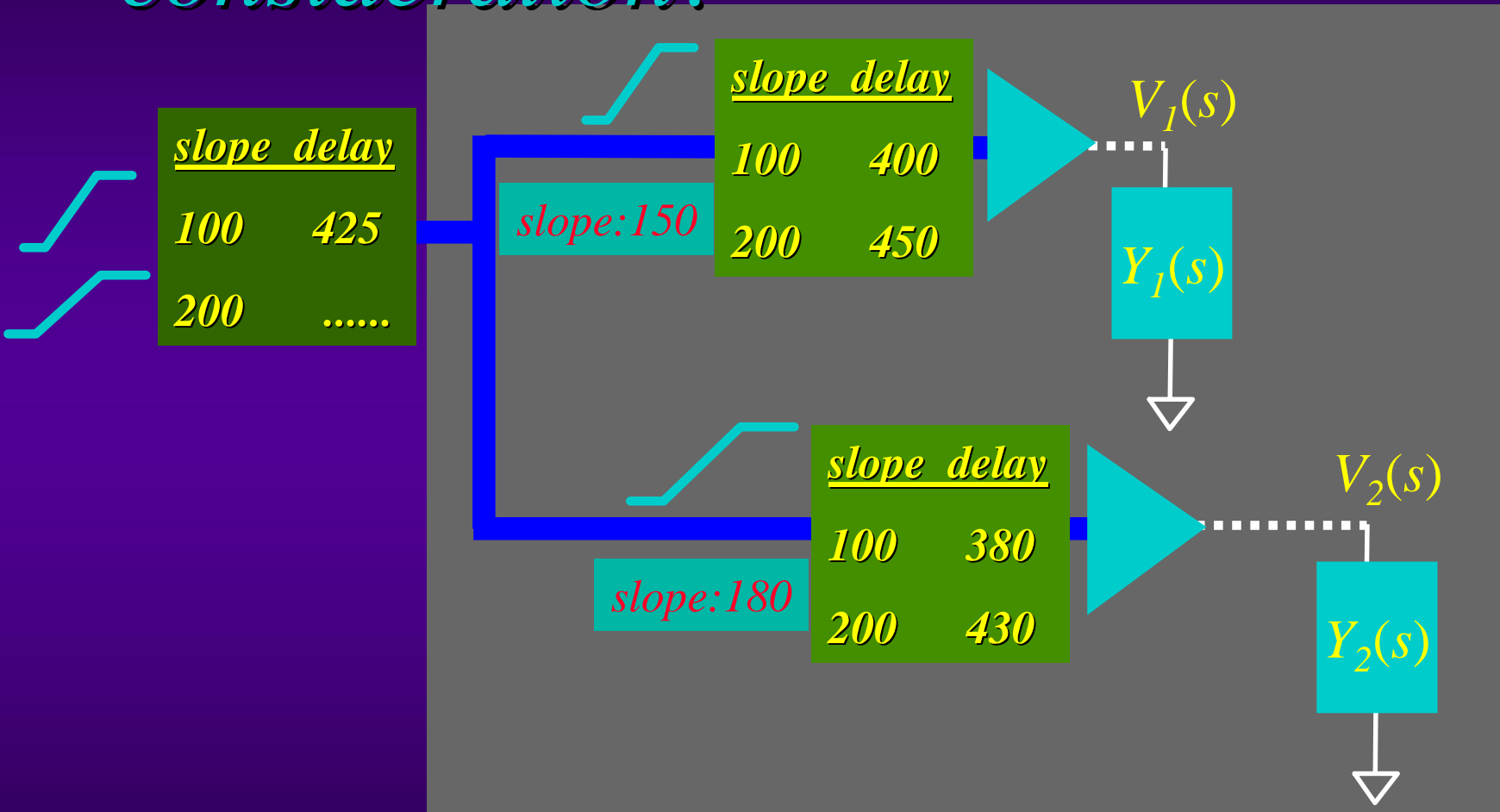
- $H'(s) = H(s)/[1 + R Y(s)]$
- $Y'(s) = Y(s)/[1 + R Y(s)]$

- Base case at the receiver

- $H(s) = 1$
- $Y(s) = C_Ls$



# *With buffer inserted and slope consideration?*



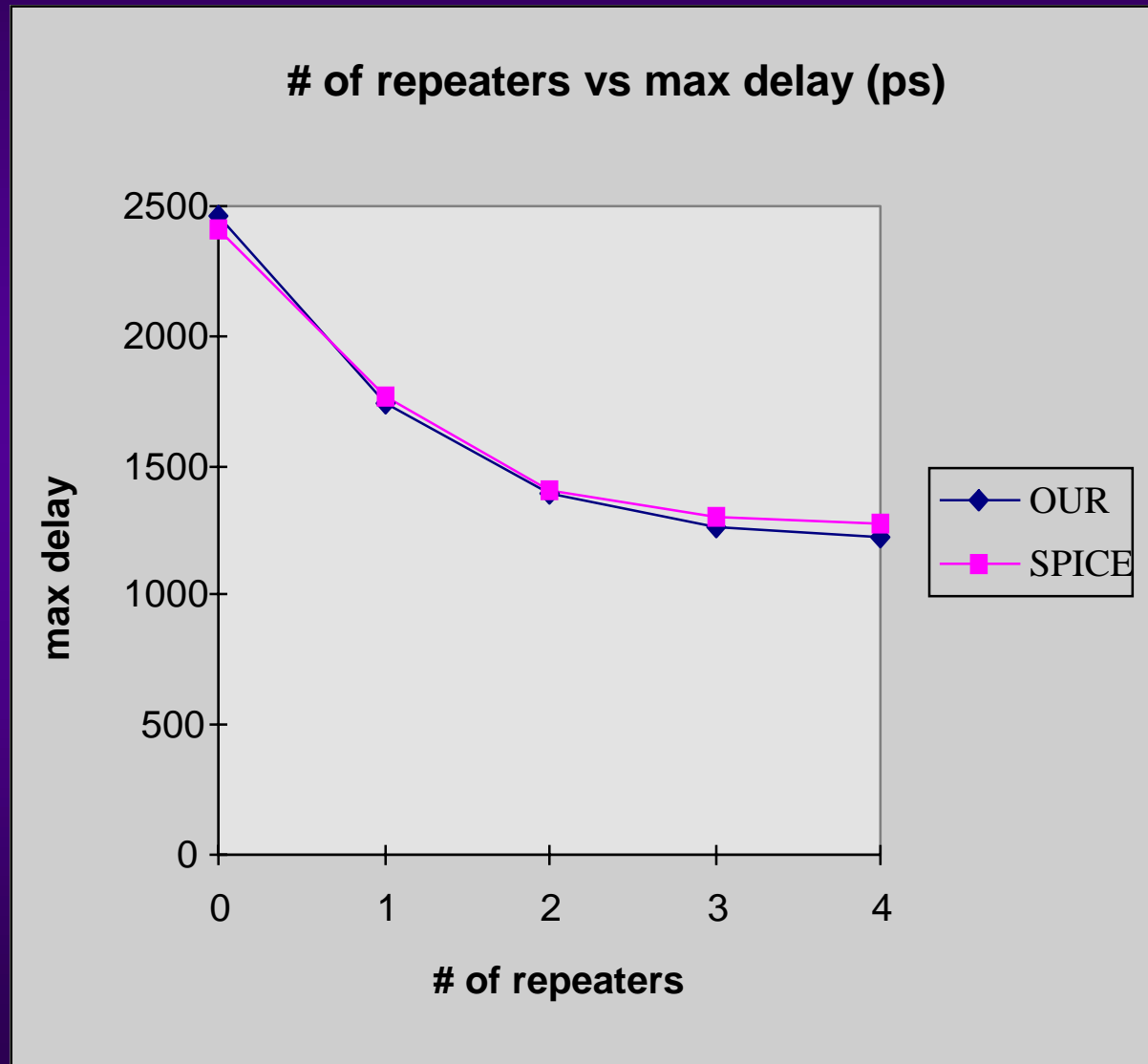
*Interpolate and extrapolate the delay at receivers*

# Results

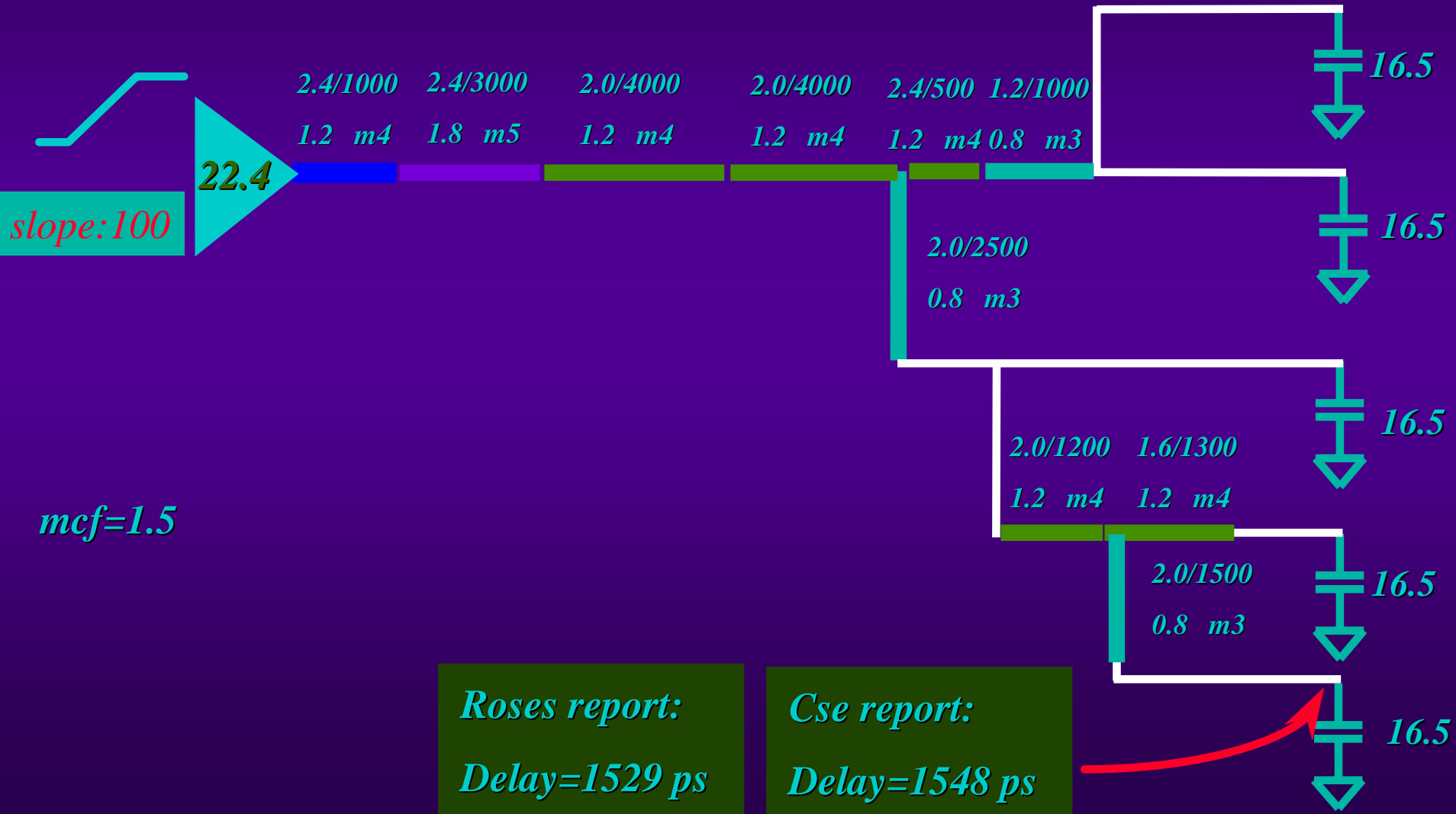
- Sample net -- 10000  $\mu\text{m}$  line on m1pm2 broken into 40 segments
- Delay before optimization -- 2405 ps
- Time for optimization -- 27 seconds on RS6K

Stages	OUR	SPICE	% error
0	2467	2405	2.5
1	1736	1761	-1.4
2	1388	1404	-1.1
3	1267	1301	-2.6
4	1218	1274	-4.3

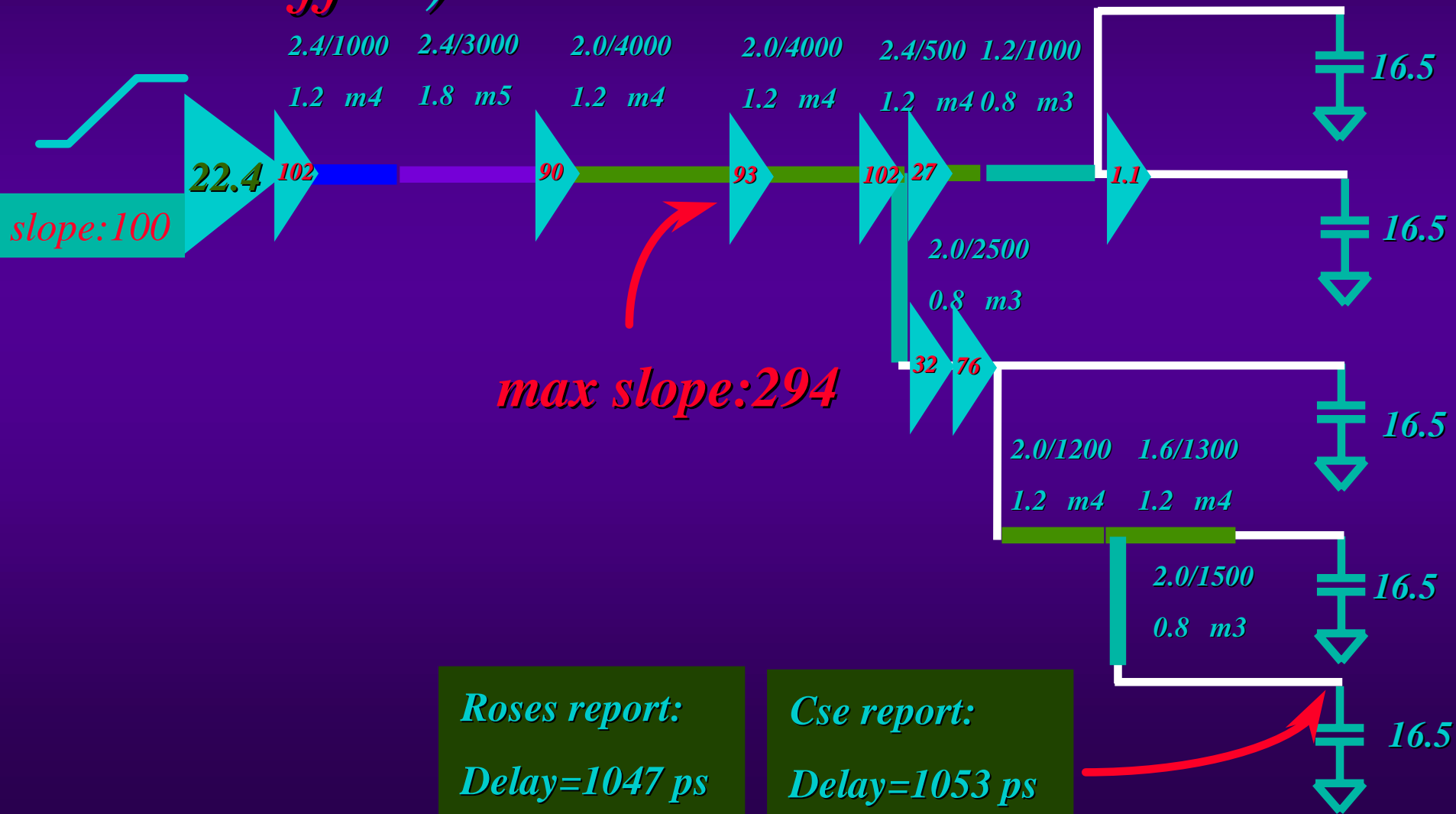
# Cost-Performance Curves



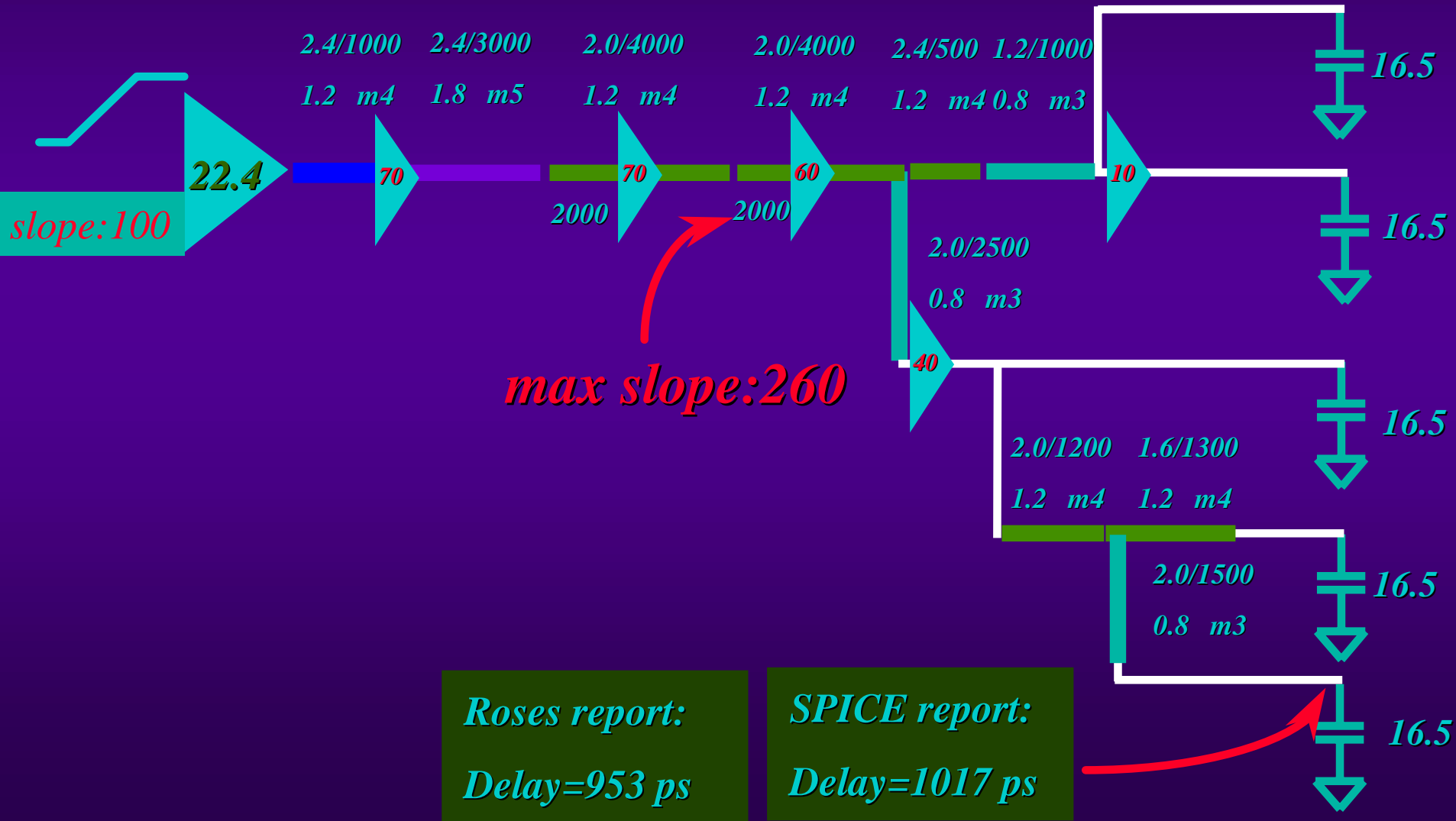
# Case Study:



# Case Study: Manual Result (8 buffer)

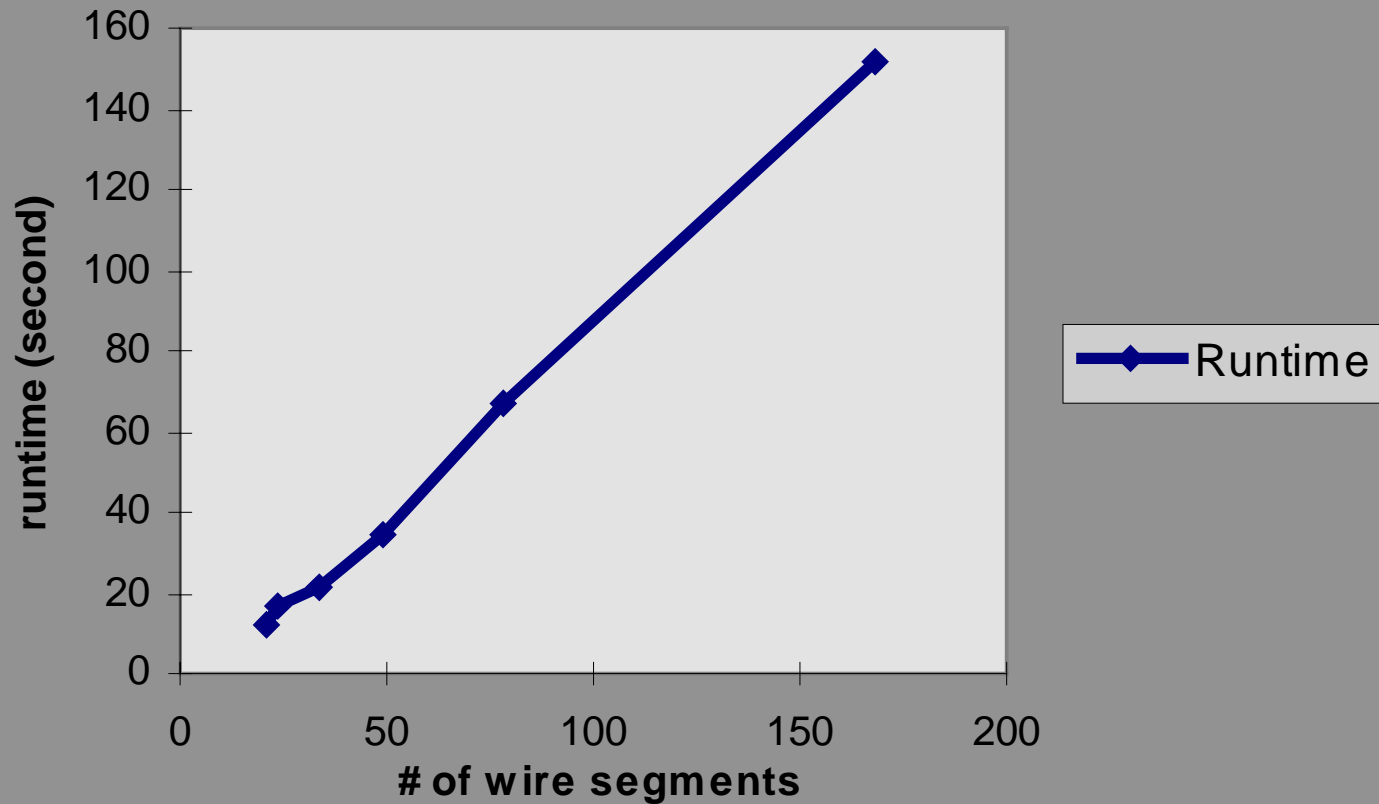


# Case Study: Our Result (5 buffers)



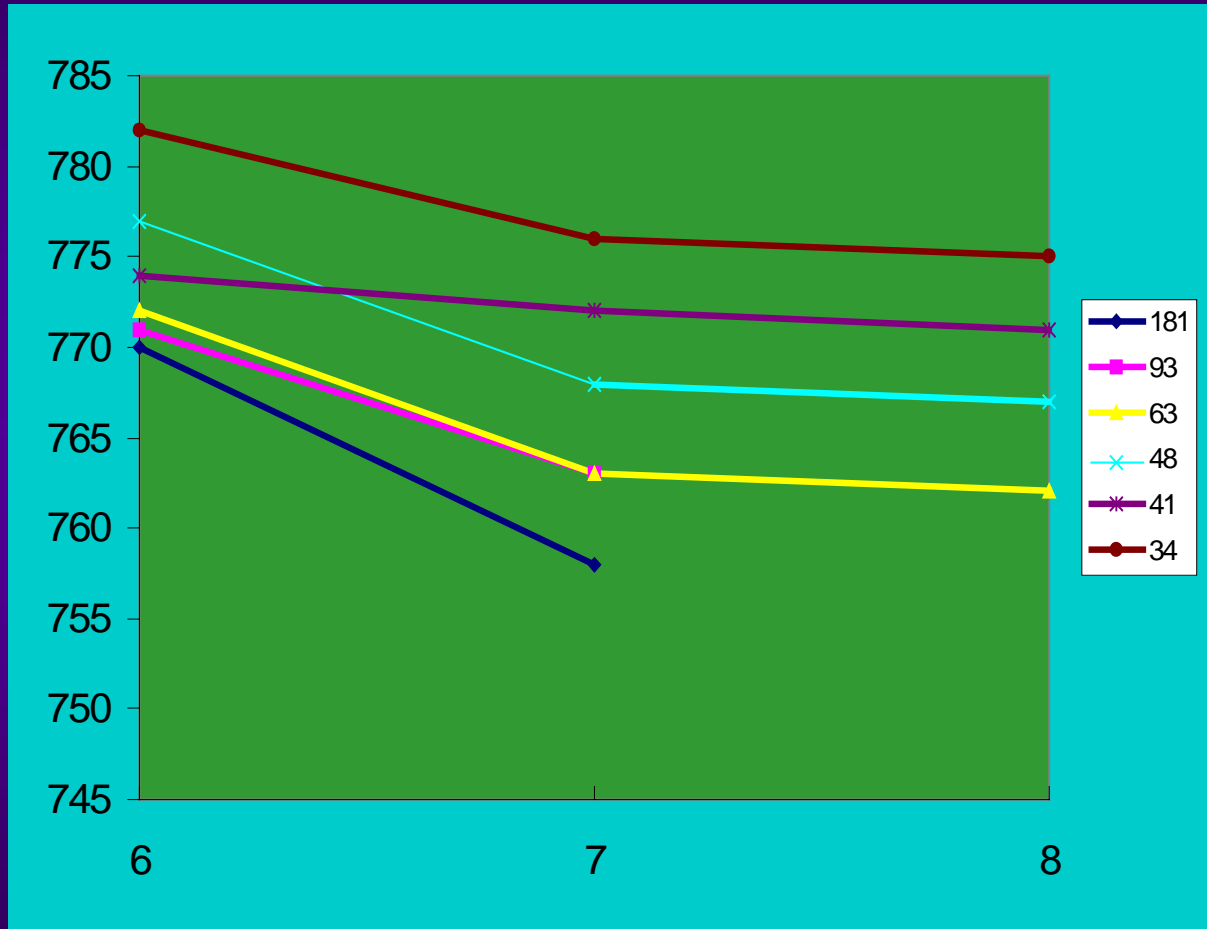
# *Runtime Report*

**# of wire segments vs runtime**



# *# of wire segment vs maximum delay*

*Maximum  
delay*



*# of repeaters*

# *Conclusion*

- Buffer model provides about 5~7% accuracy relative to SPICE
  - The total net capacitance is no longer a valid load approximation
  - Using accurate models aid
- Hierarchical moments computation of RC delays and slopes
  - New Moment-matching methods provide efficient and accurate delay calculation for RC nets especially for hierarchical moment computation
- Dynamic programming approaches applied to buffer insertion
  - Hierarchical moment methods for efficient RC delay computation