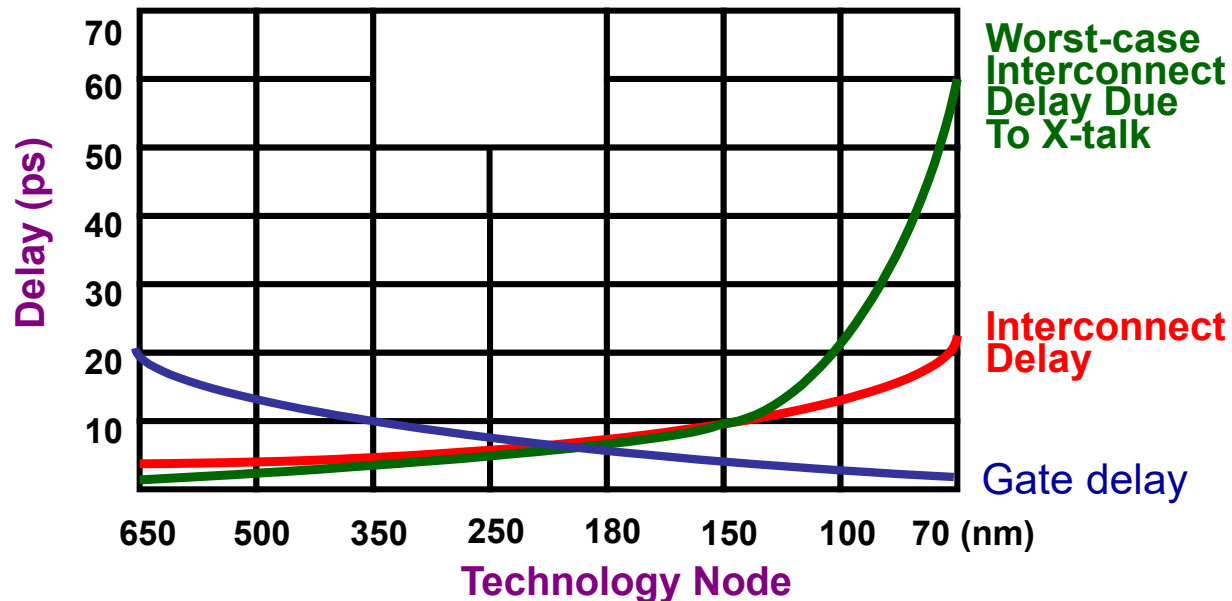


Modern Design Closure Techniques

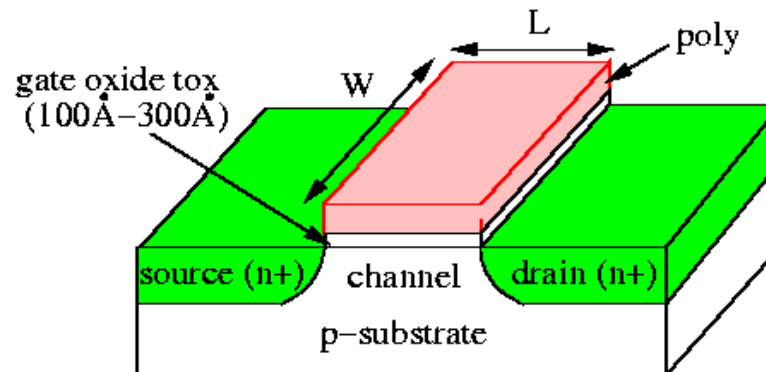
- Course contents:
 - Introduction to post layout optimization
 - Physical synthesis and buffering interconnect
 - Engineering change order



Ideal Scaling of MOS Transistors

- Feature size scales down by S times:

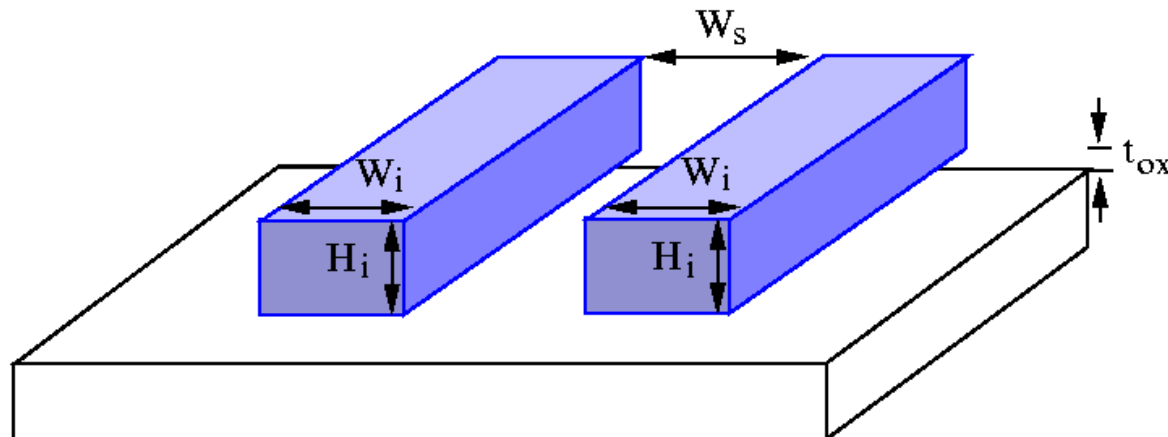
Parameter	Scaling factor
Dimensions (W, L, t_{ox} , junction depth X_j)	$1/S$
Area per device ($A = WL$)	$1/S^2$
Substrate doping (N_{SUB})	S
Voltages (V_{DD}, V_t)	$1/S$
Current per device ($I_{ds} \propto \frac{W \epsilon_{ox}}{L t_{ox}} (V_{DD} - V_t)^2$)	$1/S$
Gate capacitance ($C_g = \epsilon_{ox} W L / t_{ox}$)	$1/S$
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:

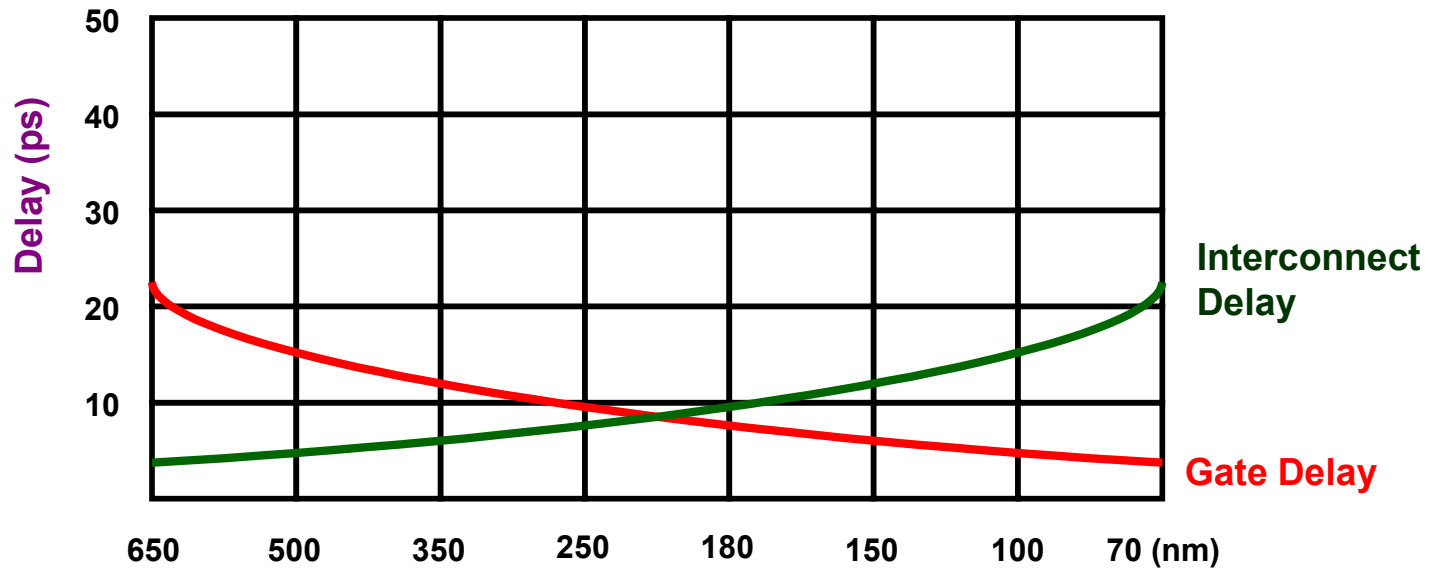
Parameter	Scaling factor
Cross sectional dimensions (W_i, H_i, W_s, t_{ox})	$1/S$
Resistance per unit length ($r_0 = \rho/W_i H_i$)	S^2
Capacitance per unit length ($c_0 = W_i \epsilon_{ox}/t_{ox}$)	1
RC constant per unit length ($r_0 c_0$)	S^2
Local interconnection length (l_l)	$1/S$
Local interconnection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



Techniques for Higher Performance

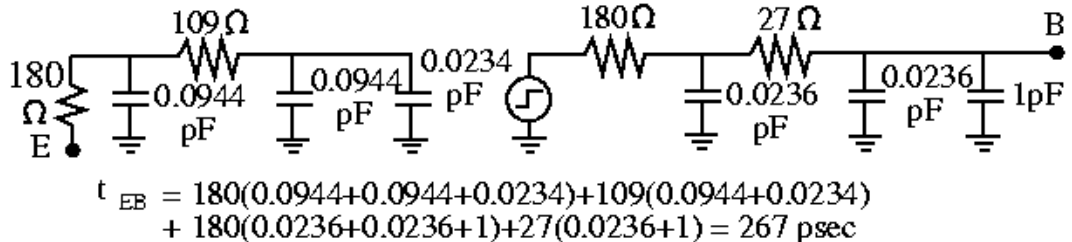
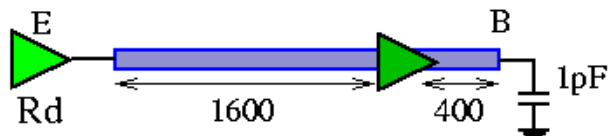
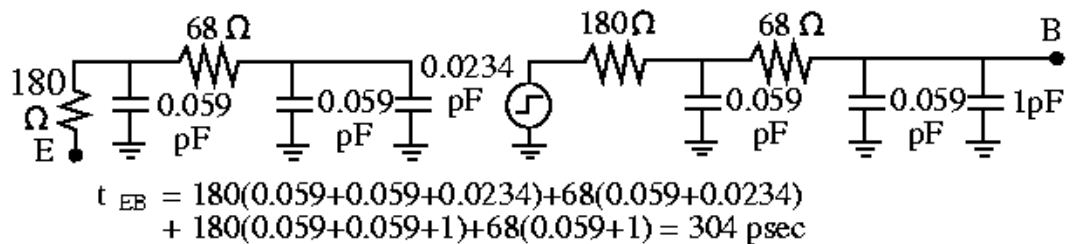
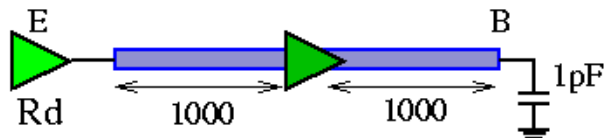
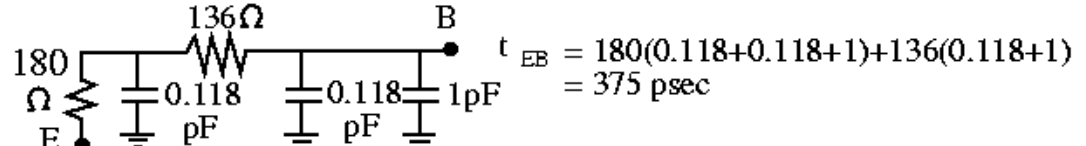
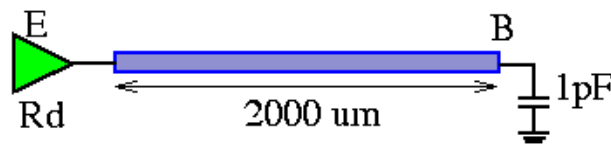
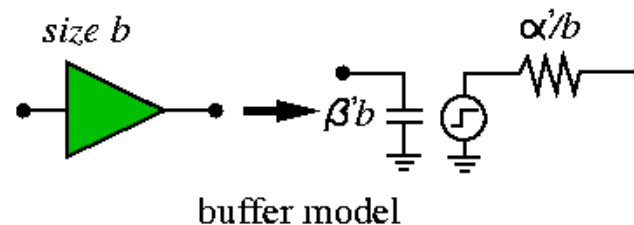
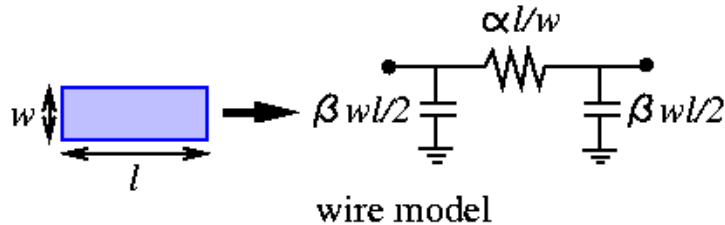
- In very deep submicron technology, interconnect delay dominates circuit performance.
- Techniques for higher performance
 - SOI: lower gate delay.
 - Copper interconnect: lower resistance.
 - Dielectric with lower permittivity: lower capacitance.
 - **Buffering**: Insert (and size) buffers to “break” a long interconnection into shorter ones.
 - 35% of all cells will be intra-block repeaters for 45nm node
 - **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 Delay Dominance



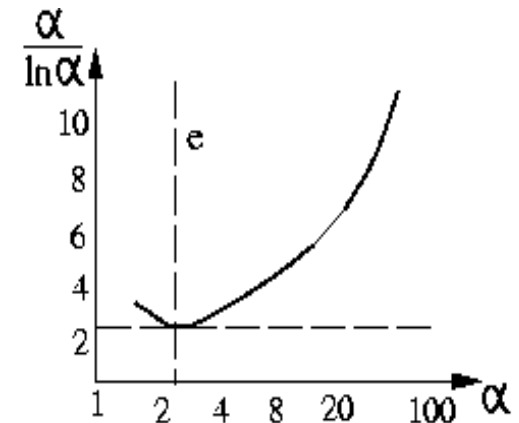
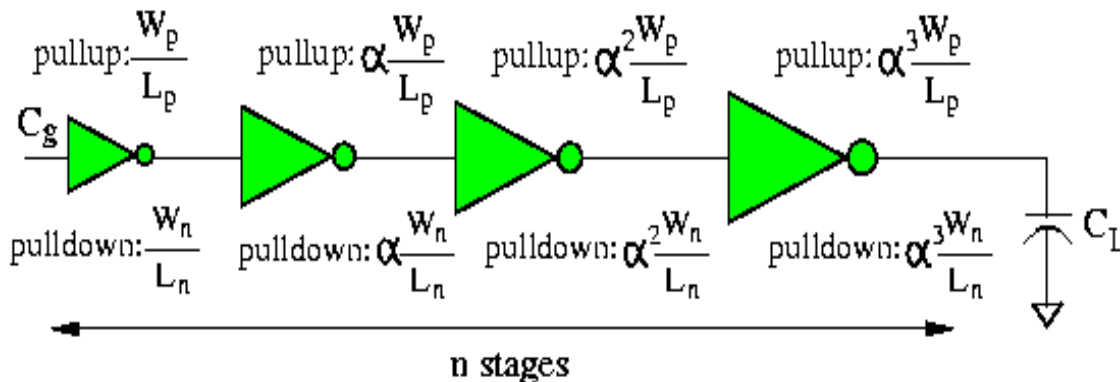
Improving Delay by Buffering

- 0.18 μm technology: 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.



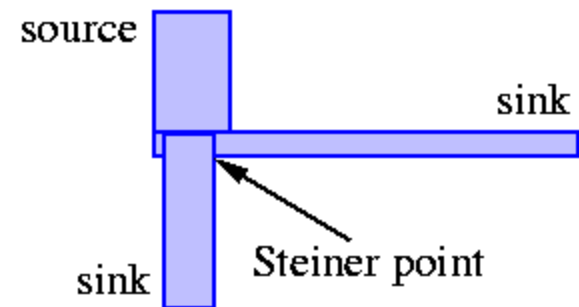
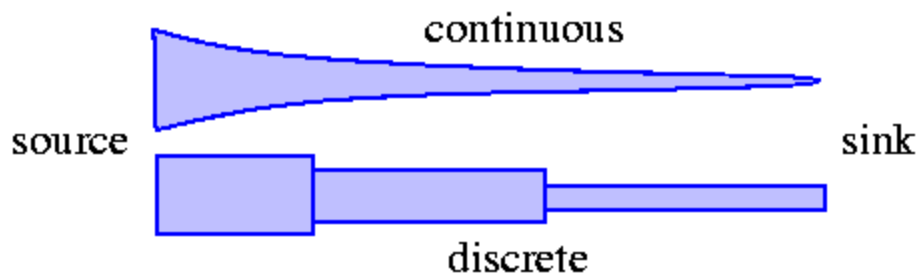
Optimal Buffer Sizing w/o Considering Interconnects

- Delay through each stage is αt_{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 ($\alpha = e$).



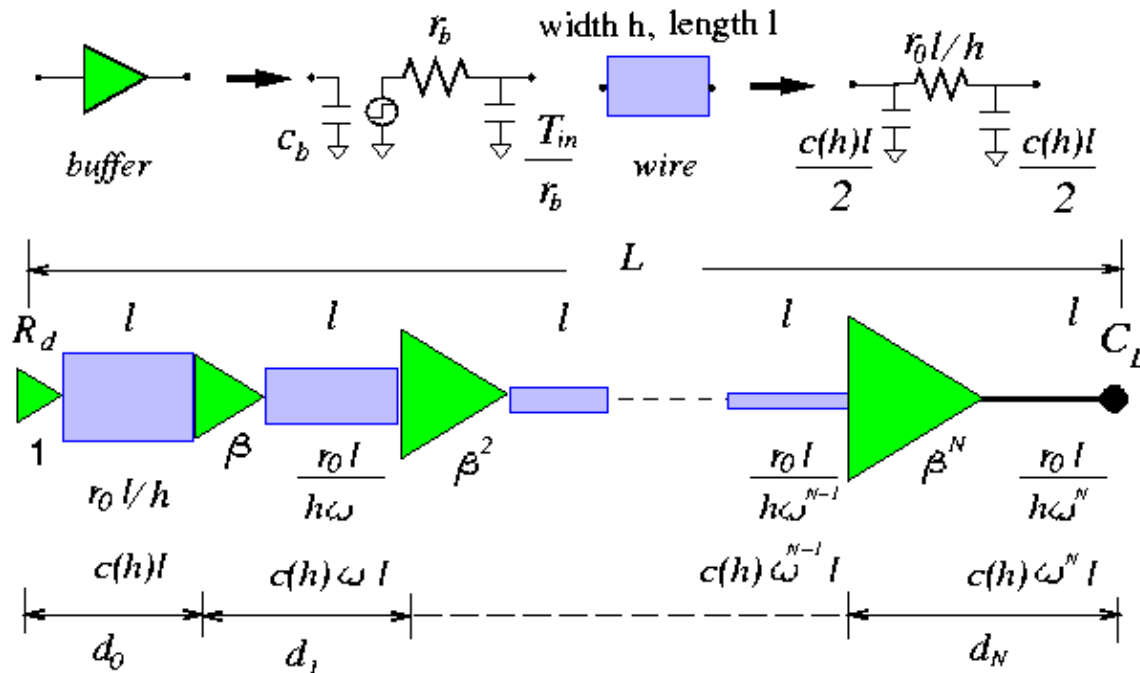
Wire Sizing

- 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.



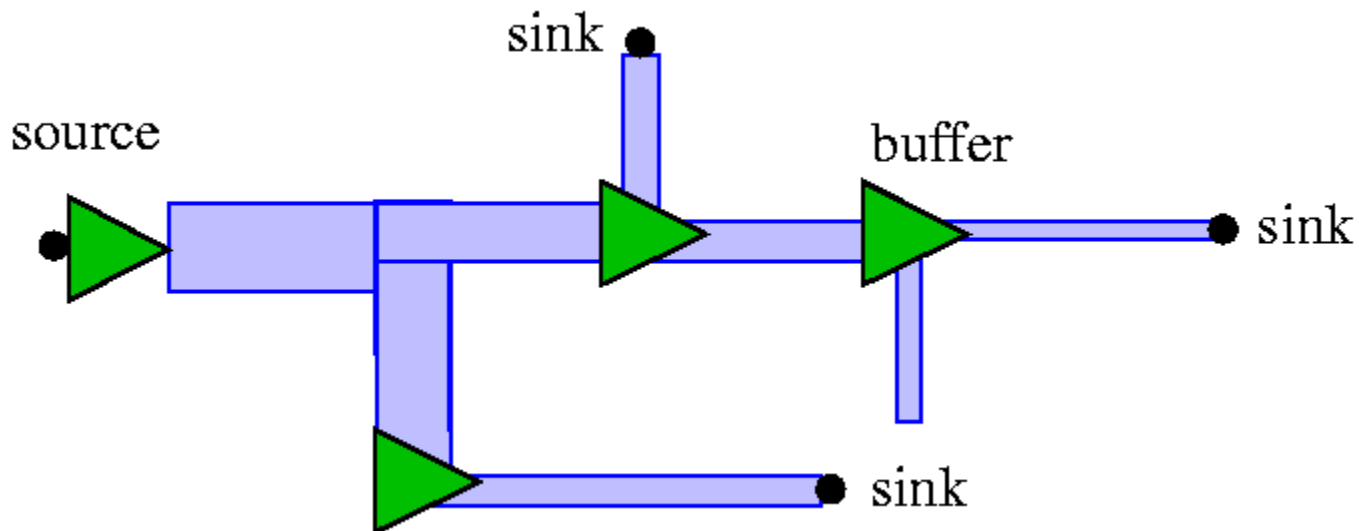
Simultaneous Wire & Buffer Sizing

- **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 β for buffer sizes and the stage ratio ω for wire widths such that the wire delay is minimized.



Wire Sizing & Buffering for Delay Optimization

- Size wires and buffers to optimize delay.
- Trade-off among area, delay, power, clock skew, clock-skew sensitivity, crosstalk, crosstalk sensitivity, etc.
- Popular techniques:
 - Discrete wire/buffer sizing: dynamic programming
 - Continuous wire/buffer sizing: mathematical programming (e.g., Lagrangian relaxation)



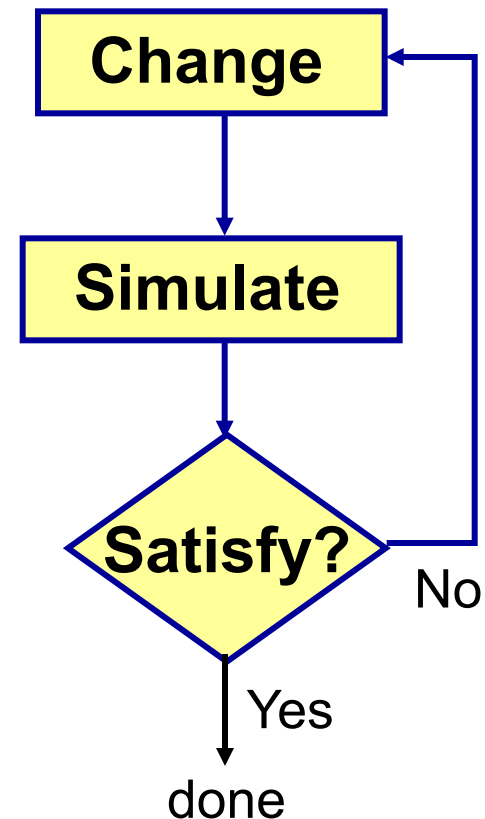
Post-layout Optimization

- “Moore's” law implication: Double the work load and design complexity every 18 months.
- Trends
 1. **Interconnect delay dominates circuit performance.**
 2. Increased custom design for high performance.
 3. Aggressive tuning for performance optimization.
 4. Severe interconnect effects.
 5. Signal integrity issues.
 6. Shorter time-to-market.
- Post-layout circuit component tuning can significantly improve circuit performance and signal integrity without major modification.

Manual Sizing

- Pros
 - Takes advantage of human experience
 - Can simultaneously combine with other optimization techniques
- Cons
 - Is slow, tedious, limited, and error-prone
 - Relies too much on experience, requires solid training
 - Cannot guarantee optimality (don't know when to stop)

Many iterations



Automatic Circuit Tuning

- Pros
 - Is fast
 - Can achieve the best performance
 - Can explore other objectives (power/delay/noise tradeoff)
 - Can boost productivity
 - Can guarantee optimality for convex problems
 - Can insure timing and reliability
- Cons
 - Complicated tool development and support (\$\$)
 - Tool testing, integration, and training

Good Tuning Algorithms

- Efficient
- Simple (easy to develop and maintain)
- Easy to use
- Versatile
- Optimal (for convex problems)
- Solution quality index (error bound to the optimal solution)

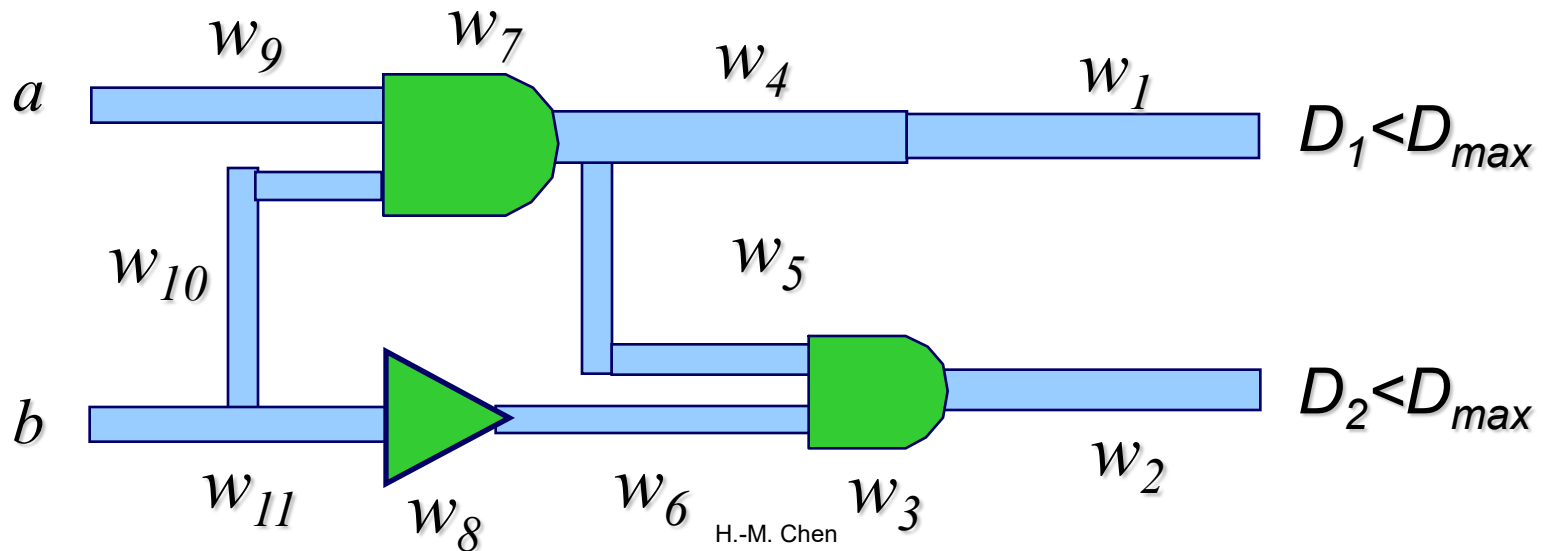
A Simple Sizing Problem

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

Minimize D_{\max}

subject to $D_i(\mathbf{W}) \leq D_{\max}, i = 1..m$

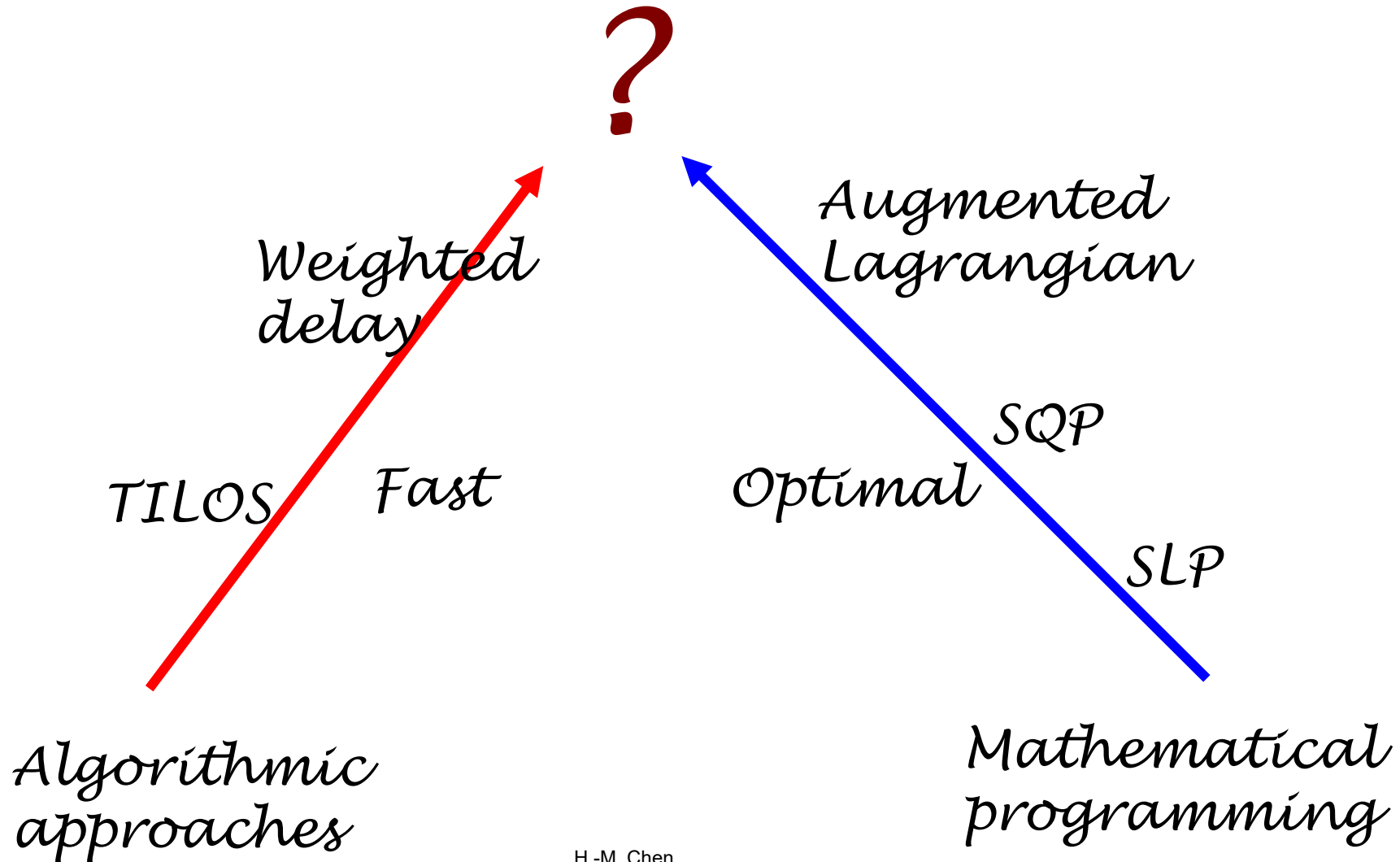
$L \leq w_i \leq U, i = 1..n$



Popular Sizing Works

- Algorithmic approaches: faster, non-optimal for general problems
 - TILOS (Fishburn, Dunlop, ICCAD-85)
 - Weighted Delay Optimization (Cong et al., ICCAD-95)
- 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 (Chen, Chang, Wong, DAC-96; Jiang, Chang, Jou, DAC-99 [TCAD, Sept. 2000])**
 - Fast and optimal

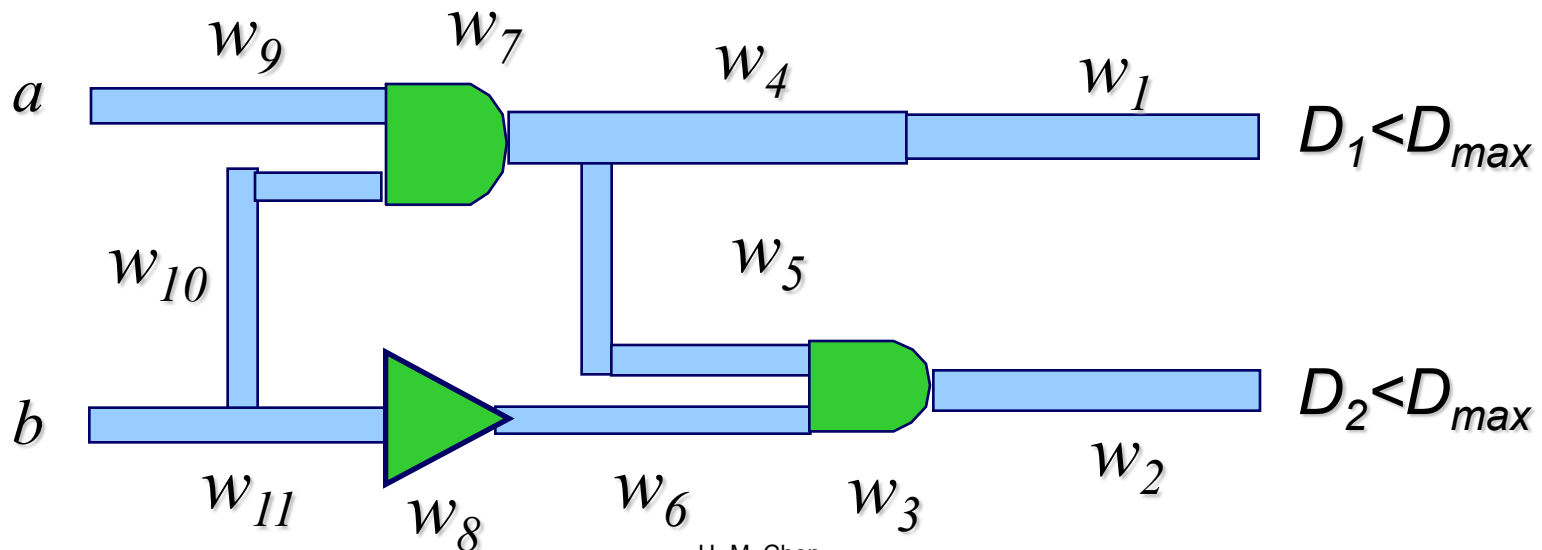
Converge?



TILOS: Heuristic Approach

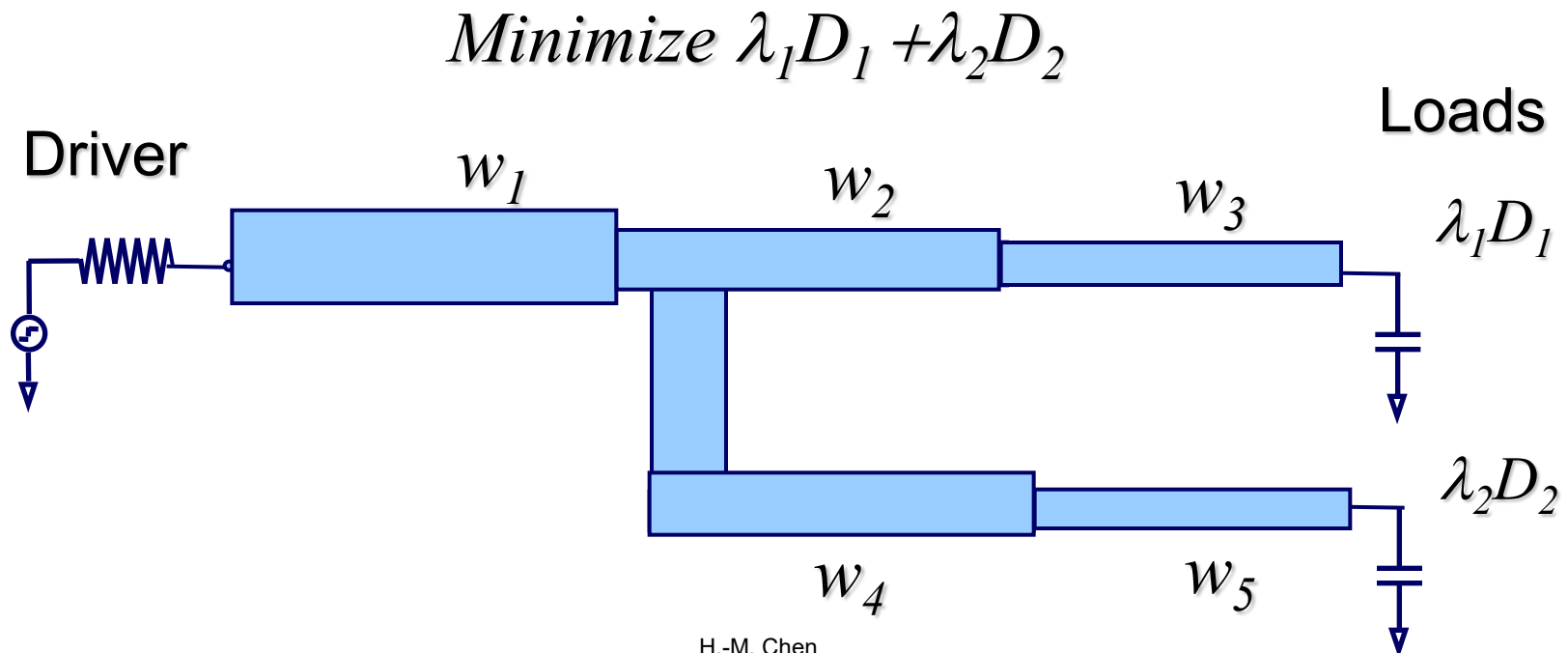
- Finds sensitivities associated with each gate
- Up-sizes the gate with the maximum sensitivity
- Minimizes the objective function

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?



Mathematical Programming

- **Formulation:**

$$\begin{aligned} & \text{Minimize } f(x) \\ & \text{subject to } g_i(x) \leq 0, \quad i = 1..m \end{aligned}$$

- **Lagrangian:**

$$L(\lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x), \quad \text{where } \lambda_i > 0$$


- **Optimality (Necessary) Condition (Kuhn-Tucker theorem):**

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

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

$$g_i(x) \leq 0, \lambda_i \geq 0 \quad (\text{Feasibility Condition})$$

Lagrangian Relaxation (1/3)

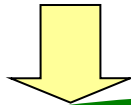

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & \cancel{g_i(x) \leq 0}, i = 1..n \\ & g_i(x) \leq 0, i = n..m \end{array} \xRightarrow{\text{LRS}} \begin{array}{ll} \text{Minimize} & f(x) + \sum_{i=1}^n \lambda_i g_i(x) \\ \text{subject to} & g_i(x) \leq 0, i = n..m \end{array}$$

- 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 (2/3)

$$\begin{aligned} &\text{Minimize} && D_{\max} \\ &\text{subject to} && D_i(\mathbf{W}) \leq D_{\max}, \quad i = 1..m \end{aligned}$$

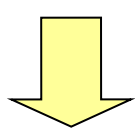
$$L \leq w_i \leq U, \quad i = 1..n$$



Lagrangian Relaxation

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

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



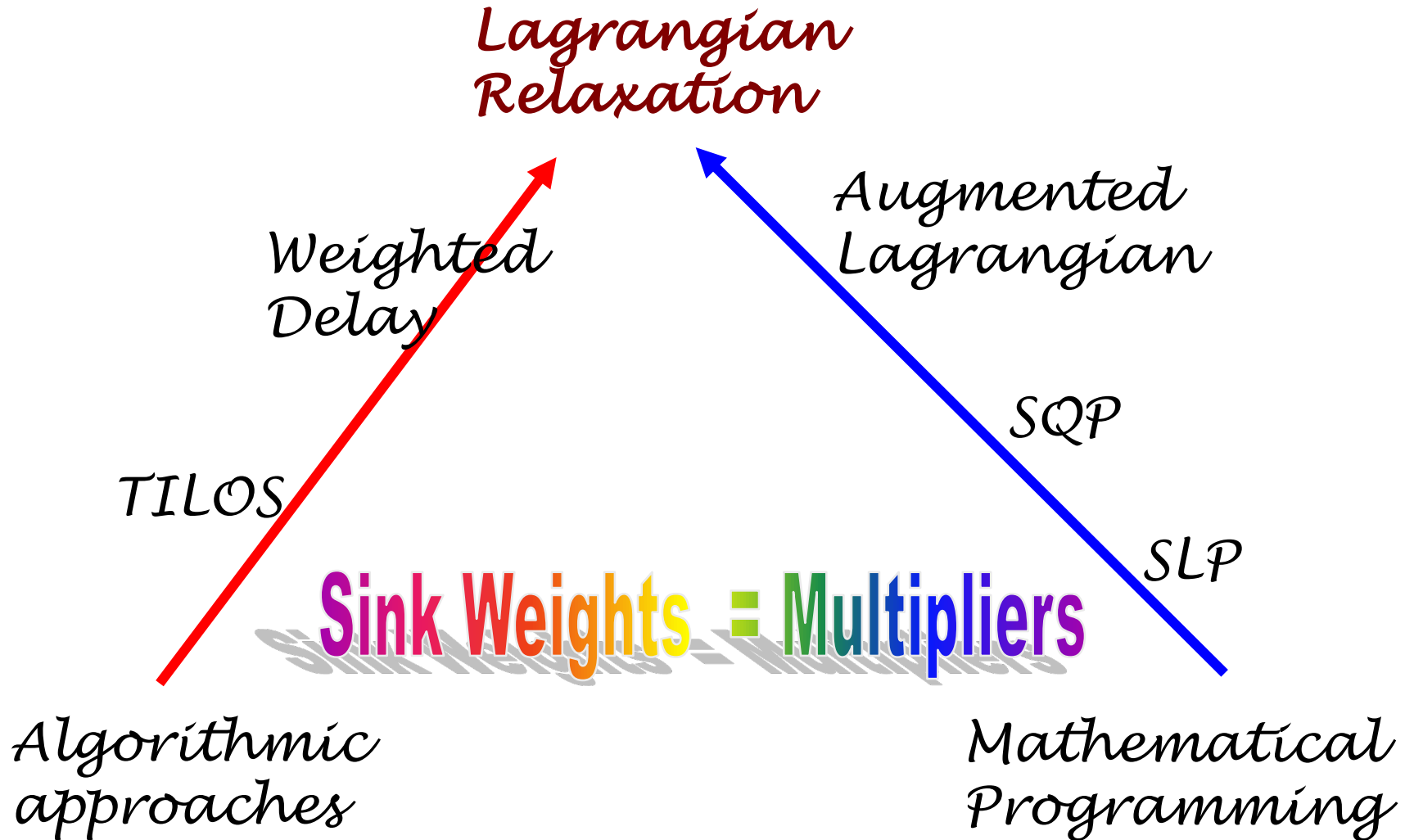
By $\frac{\partial L}{\partial D_{\max}} \stackrel{\lambda}{=} 0$, we have $\sum_{i=1}^m \lambda_i = 1$

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

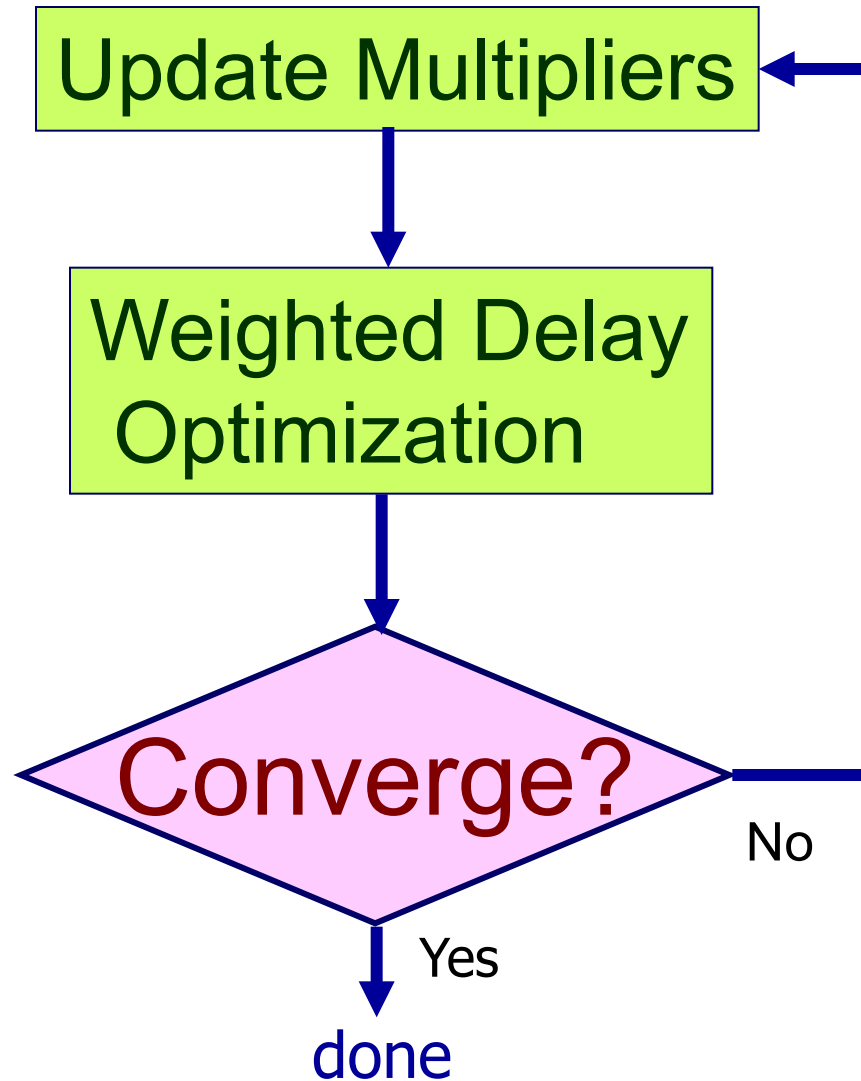
Weighted Delays!

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

Lagrangian Relaxation (3/3)

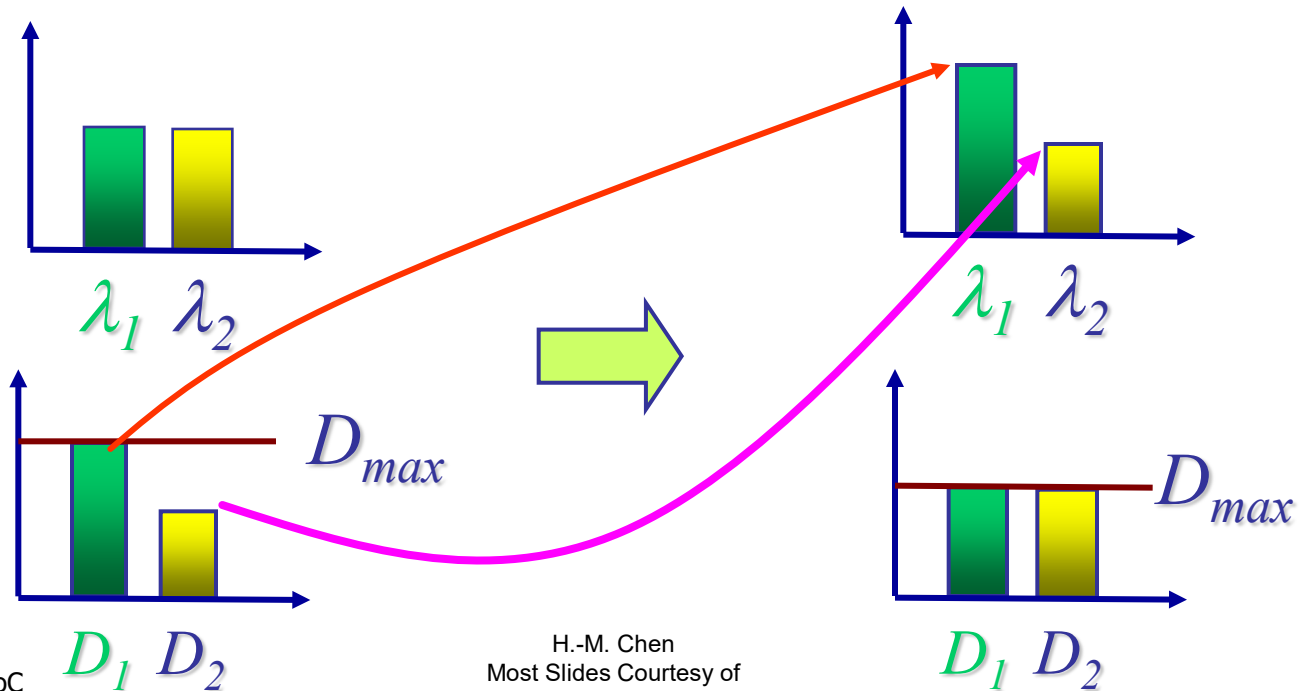
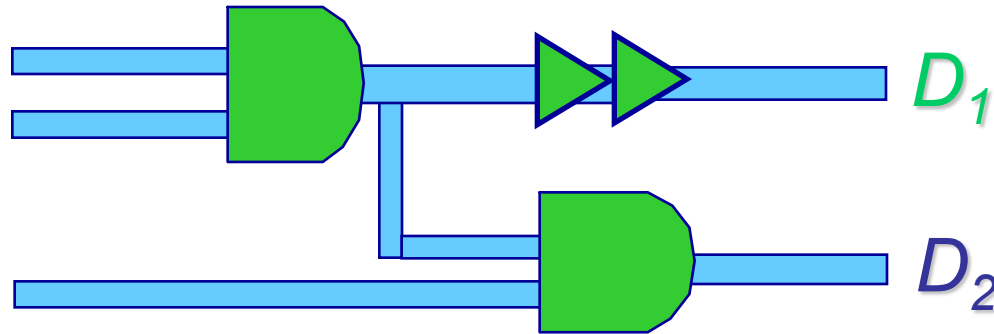


Lagrangian Relaxation Framework



Lagrangian Relaxation Framework in Sizing

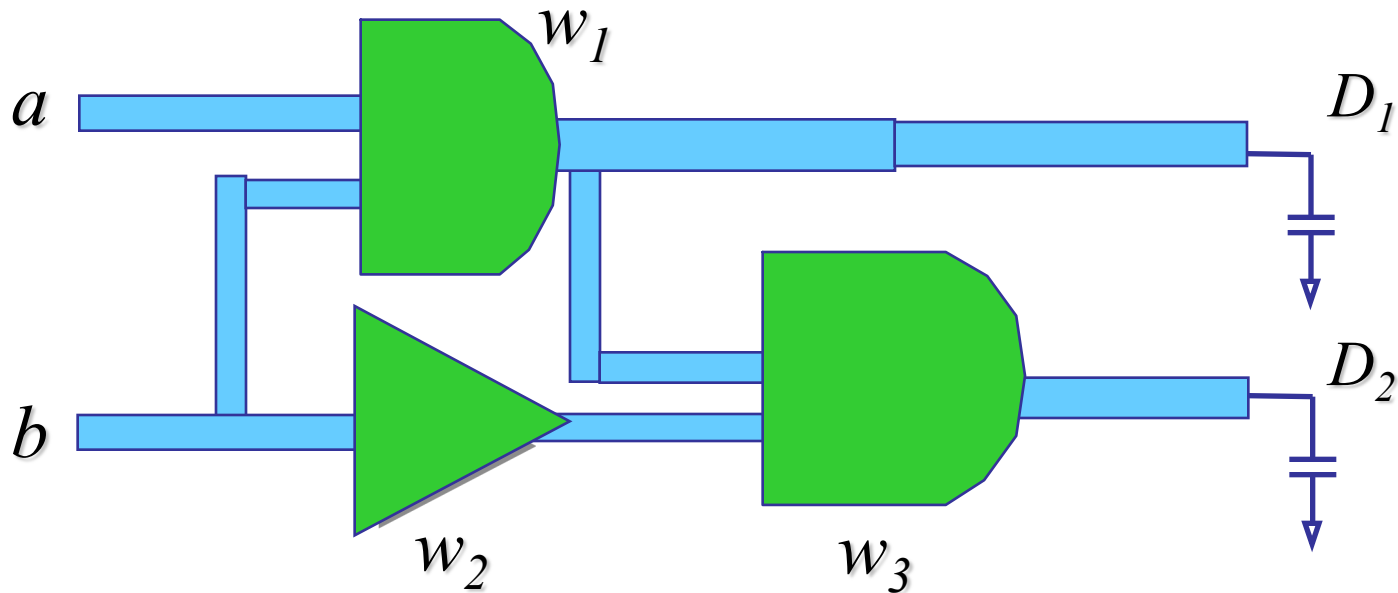
More Critical -> More Resource -> Larger Weight



Weighted Minimization

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

$$\text{Minimize } \lambda_1 D_1 + \lambda_2 D_2$$



Multiplier Adjustment: A Subgradient Approach

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

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

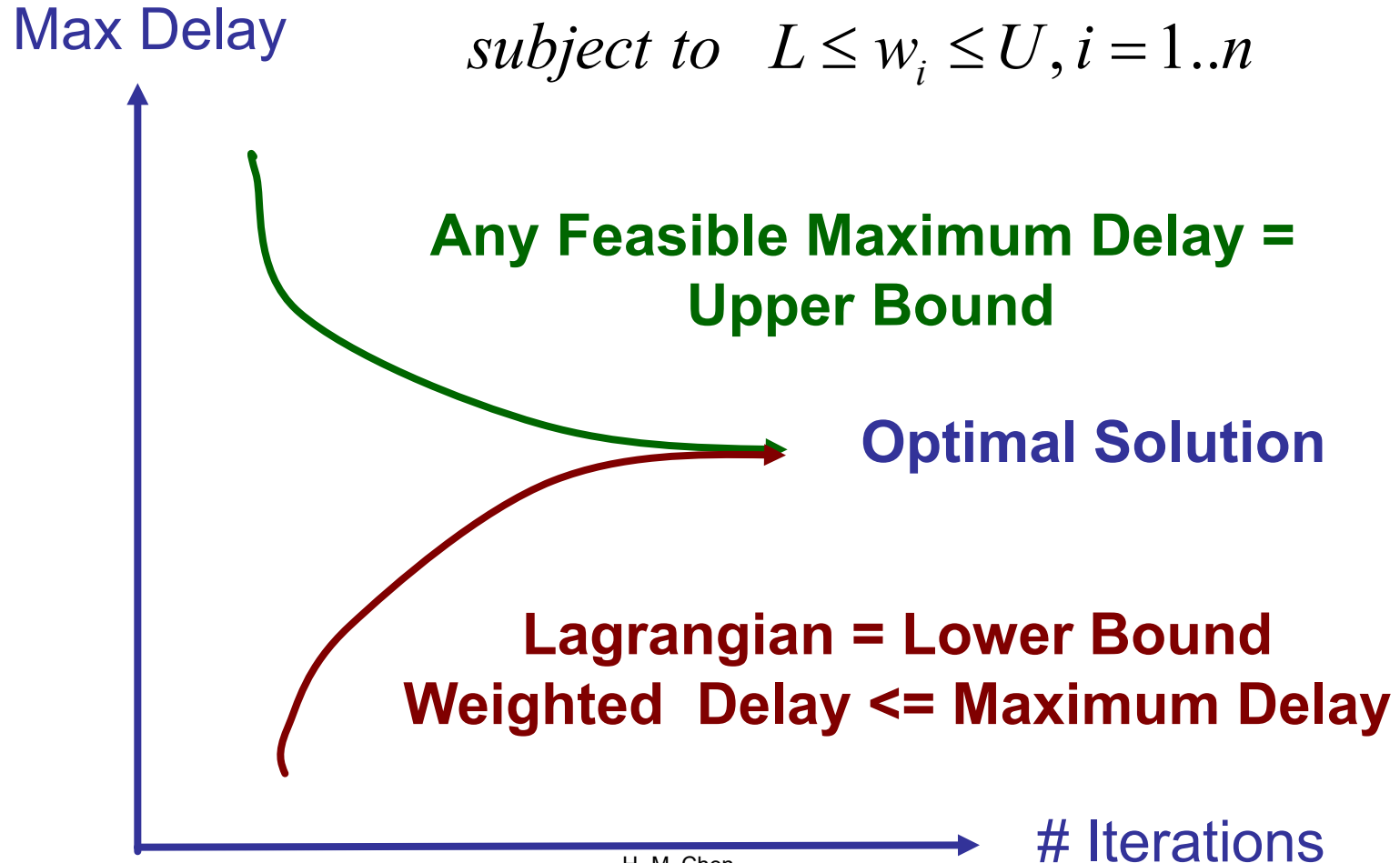
Step 2: Project λ to nearest feasible solution

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

Convergence Sequence

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

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



Physical Synthesis Concept

- It begins with
 - A mapped netlist generated by logic synthesis
- It generates
 - A new optimized netlist and a corresponding layout
- It can be considered as
 - A wrapper around traditional P&R
- “It has to be much more cognizant of not just **timing closure**, but of creating routable designs”
 - New optimizations that try to spread cells, refactor logic, and find alternative buffering strategies are key to achieving **routing closure**

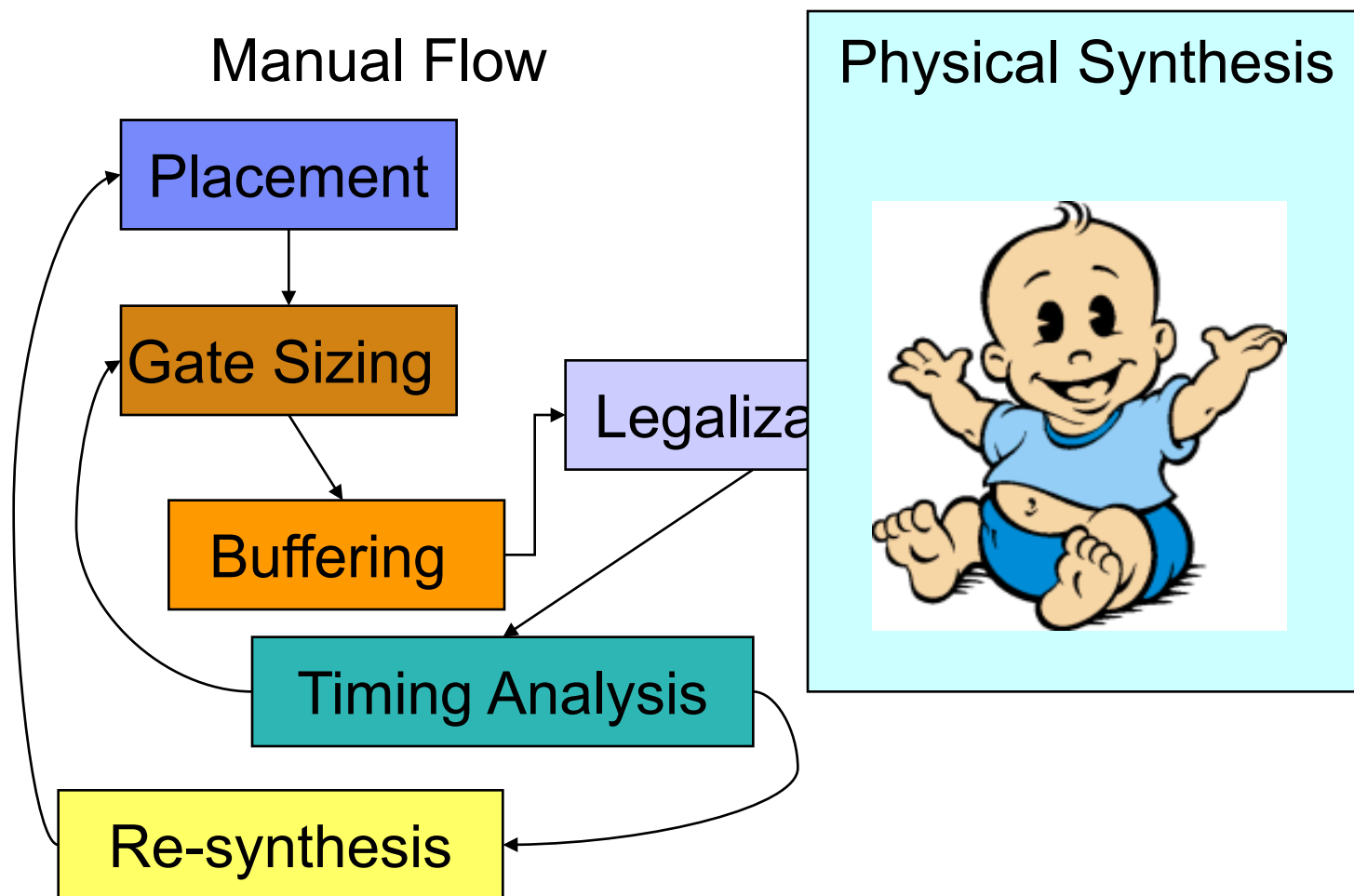


November 6, 2007

Physical Synthesis Comes of Age

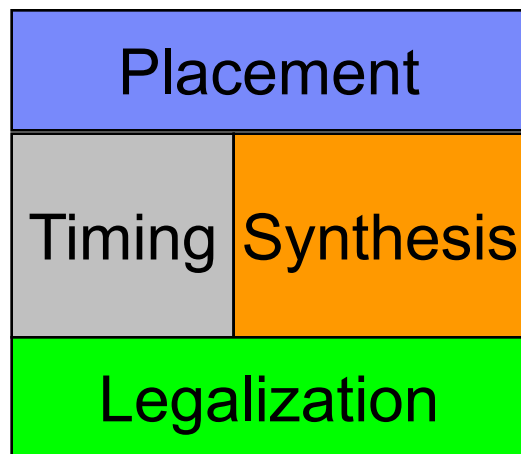
Chuck Alpert, IBM Corp.
Chris Chu, Iowa State
Paul Villarrubia, IBM Corp.

The Birth of Physical Synthesis

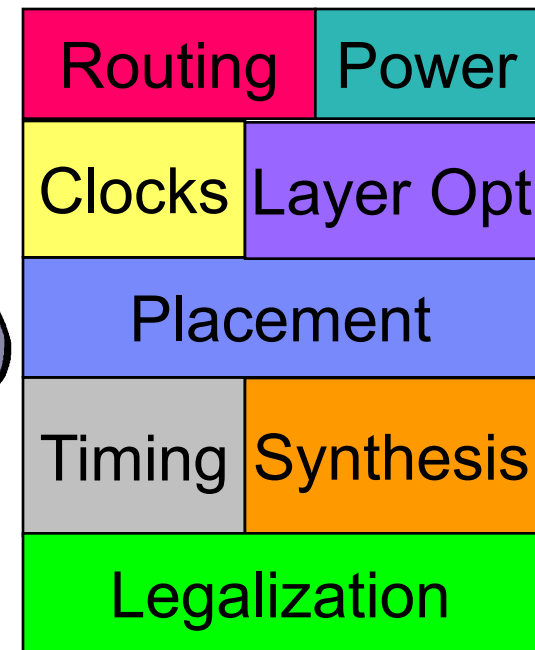
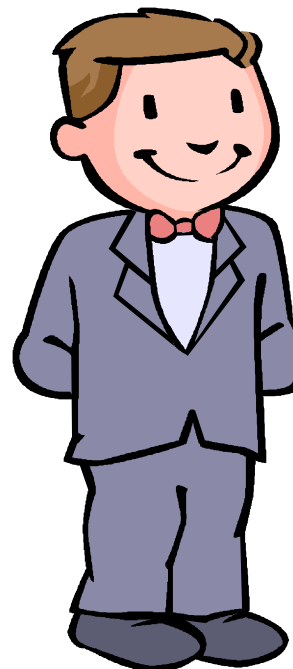


Coming of Age

Back then



Today

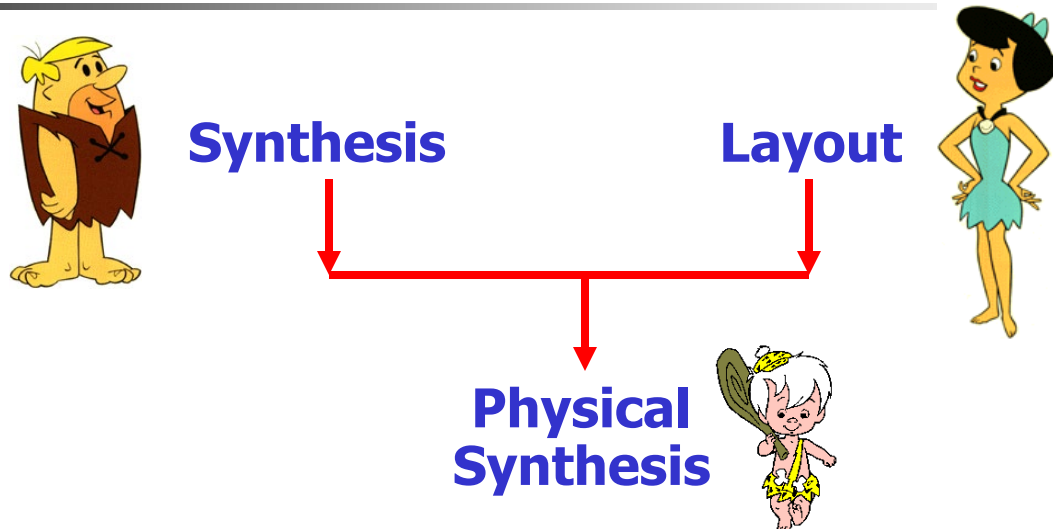




Challenges this Young Man Must Face

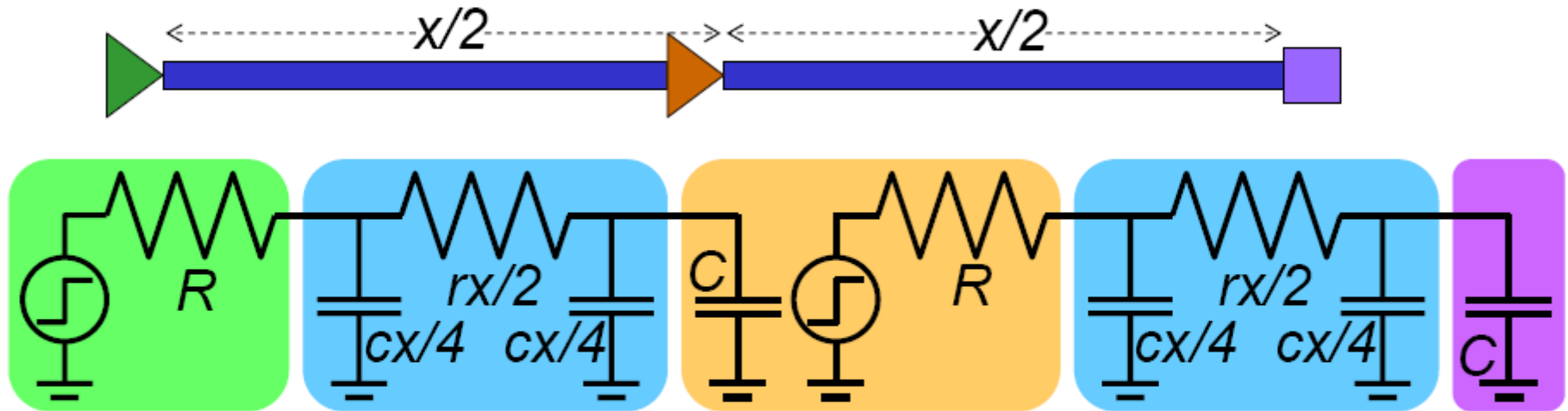
- **Increasing design complexity**
- **Routability**
- **Multi-cycle paths**
- **Hierarchical design styles**
- **Buffer explosion**
- **Efficient use of thick metal**
- **Power management**

Physical Synthesis Family Tree



- Roles of layout as a parent:
 - Clean up the mess created by physical synthesis
(Implement the netlist generated by physical synthesis)
 - Provide guidance to physical synthesis
so that it will do things right
- Is layout mature enough to serve the role?
- Is there still room for layout to grow?

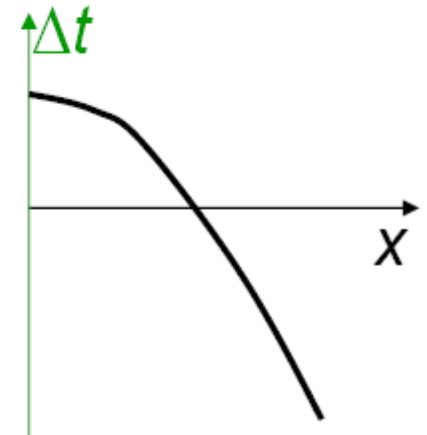
Buffering Concept



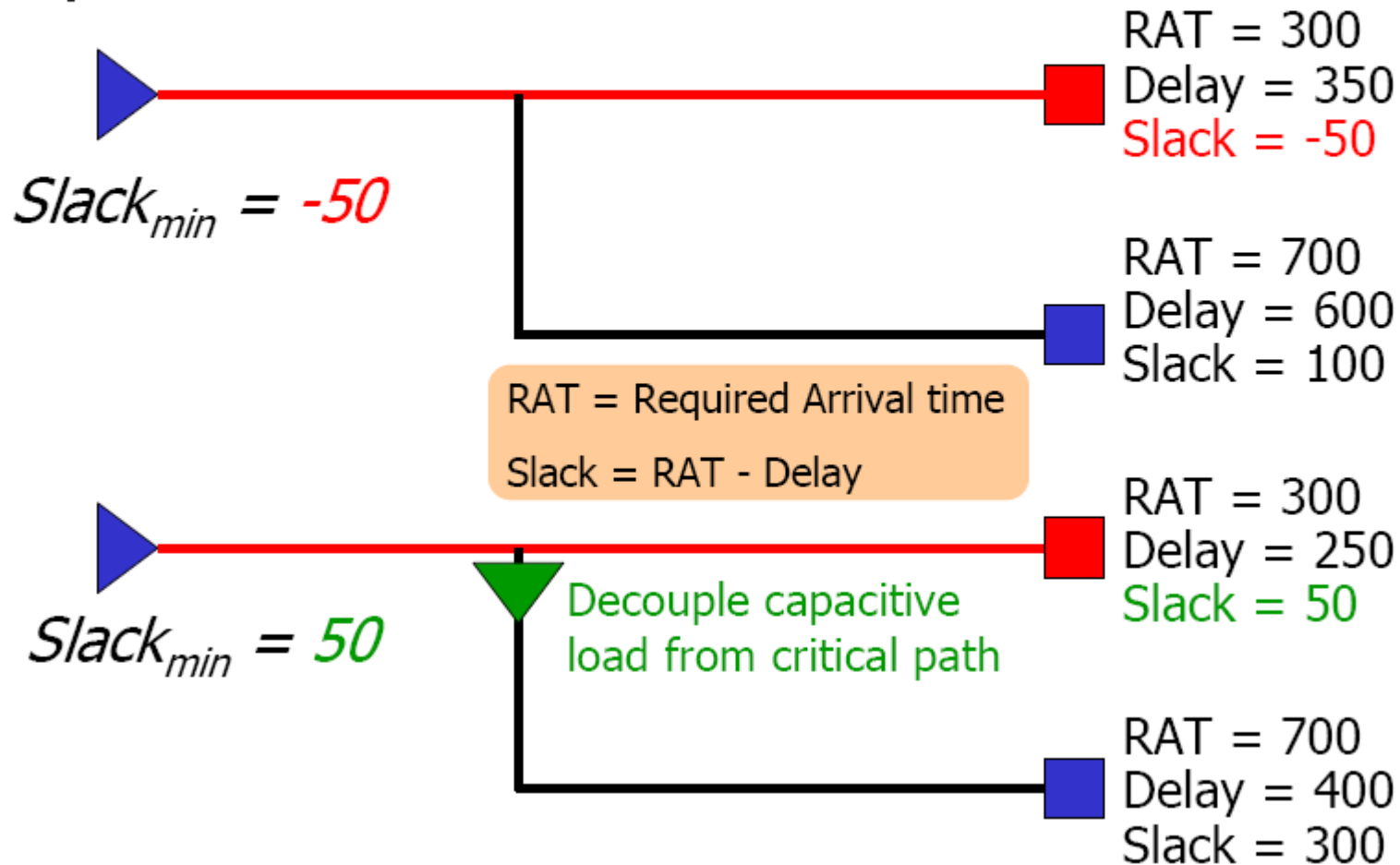
$$t_{unbuf} = R(cx + C) + rx(cx/2 + C)$$

$$t_{buf} = 2R(cx/2 + C) + rx(cx/4 + C) + t_b$$

$$t_{buf} - t_{unbuf} = RC + t_b - rcx^2/4$$



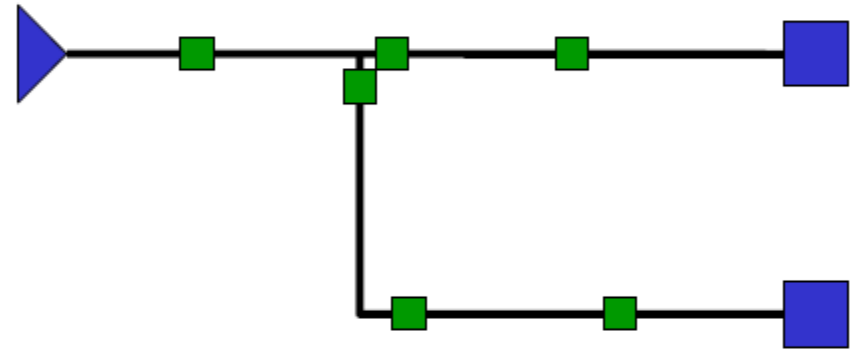
Buffers Improve Slack



Problem Formulation

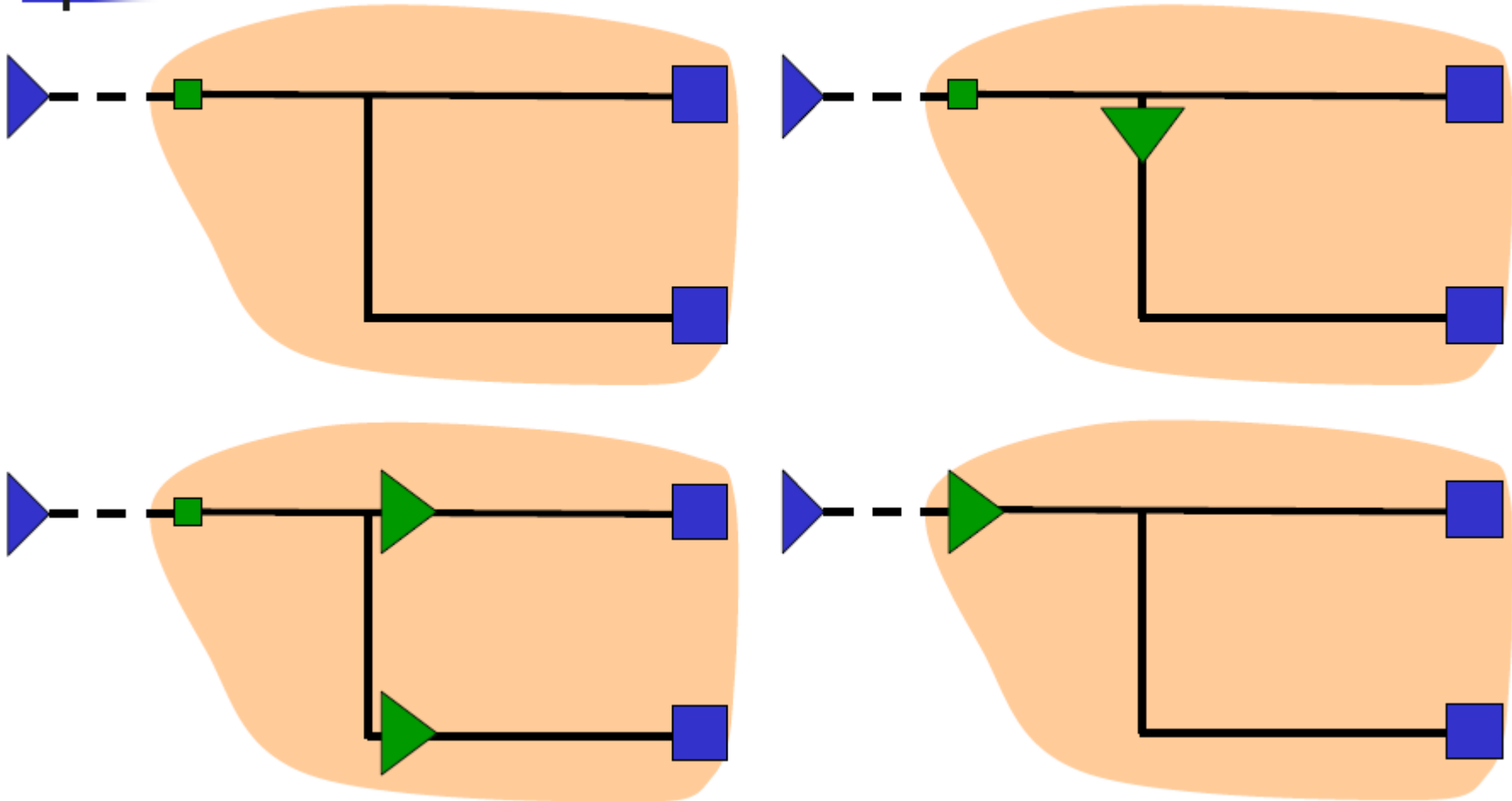
- Given

- A Steiner tree
- RAT at each sink
- A buffer type
- RC parameters
- Candidate buffer locations



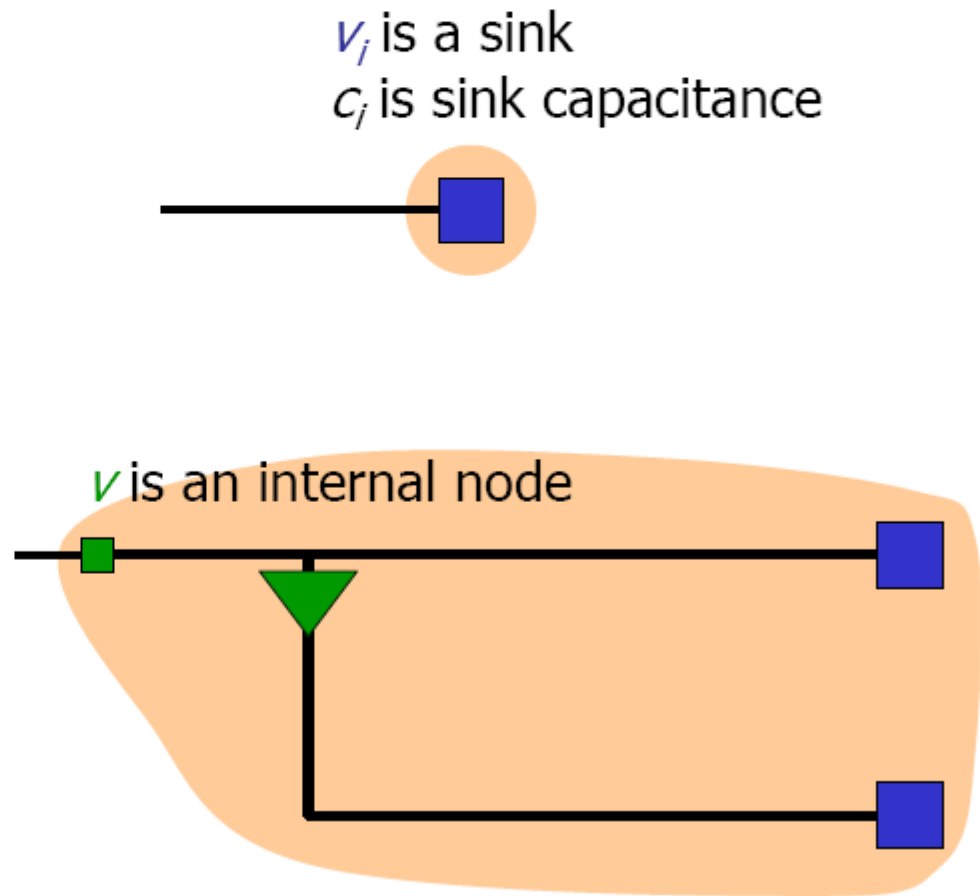
- Find buffer insertion solution such that the slack_{\min} is maximized

Candidate Buffering Solutions

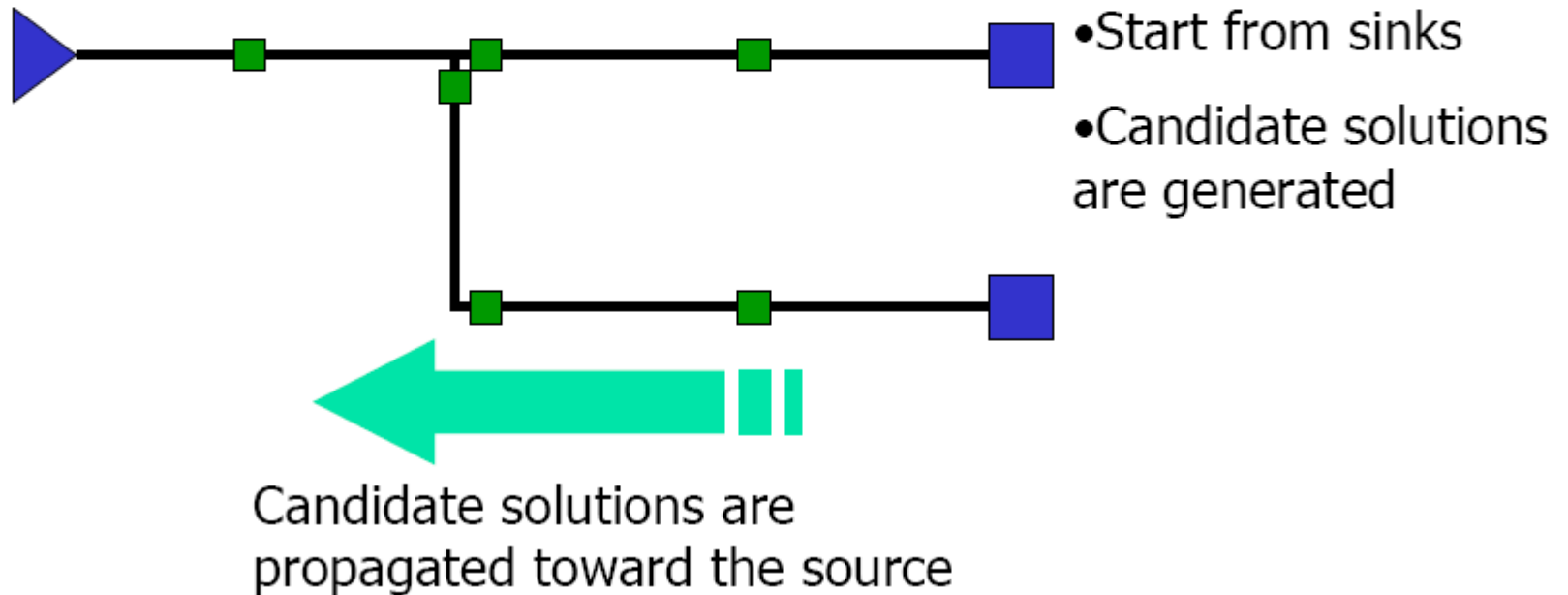


Candidate Solution Characteristics

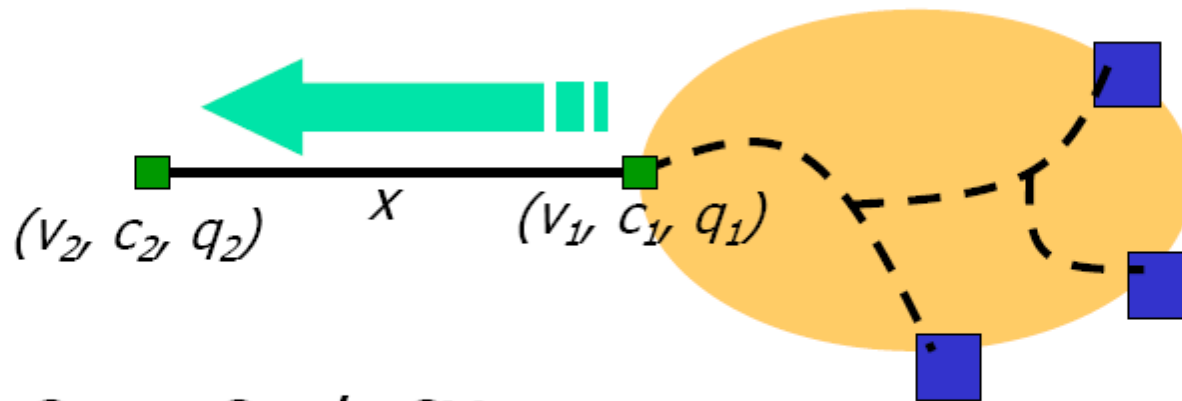
- Each candidate solution is associated with
 - v_i : a node
 - c_i : downstream capacitance
 - q_i : RAT



Van Ginneken's Algorithm

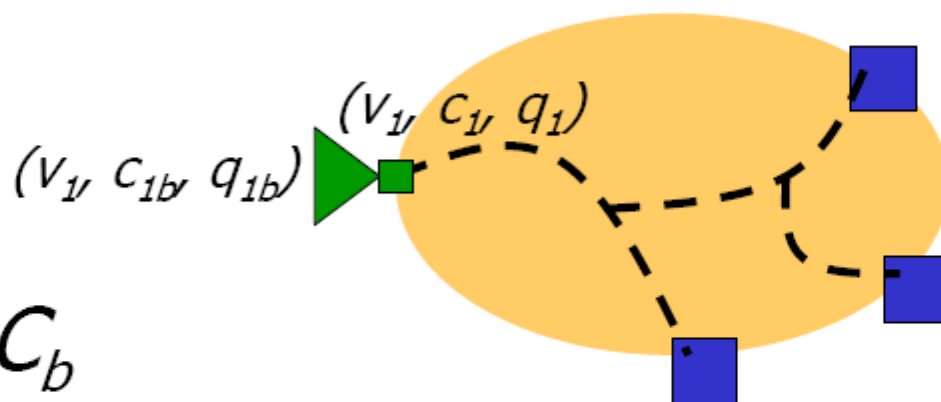


Solution Propagation: Add Wire



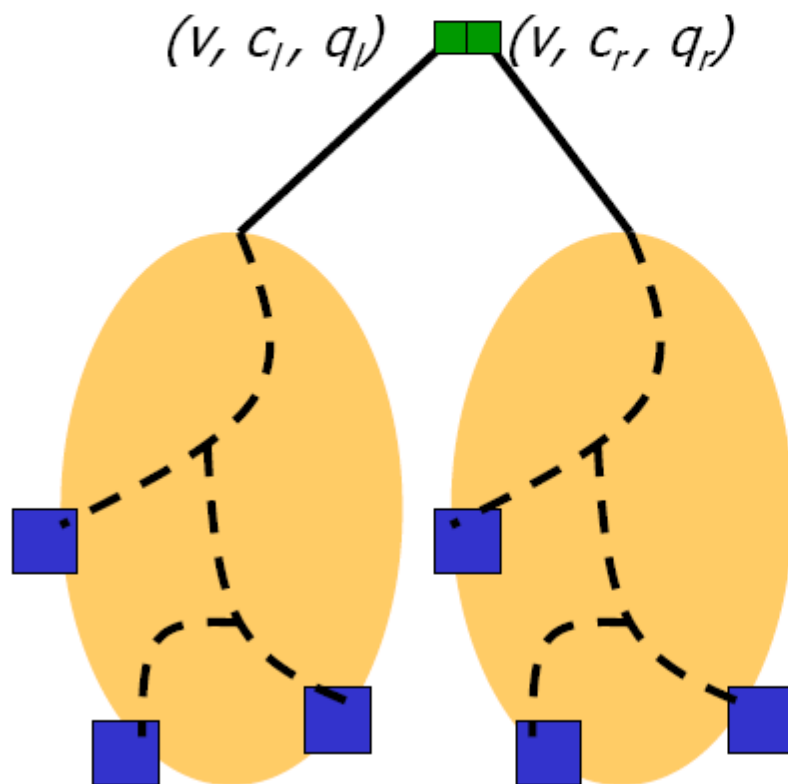
- $c_2 = c_1 + cX$
- $q_2 = q_1 - rcX^2/2 - rxc_1$
- r : wire resistance per unit length
- c : wire capacitance per unit length

Solution Propagation: Insert Buffer



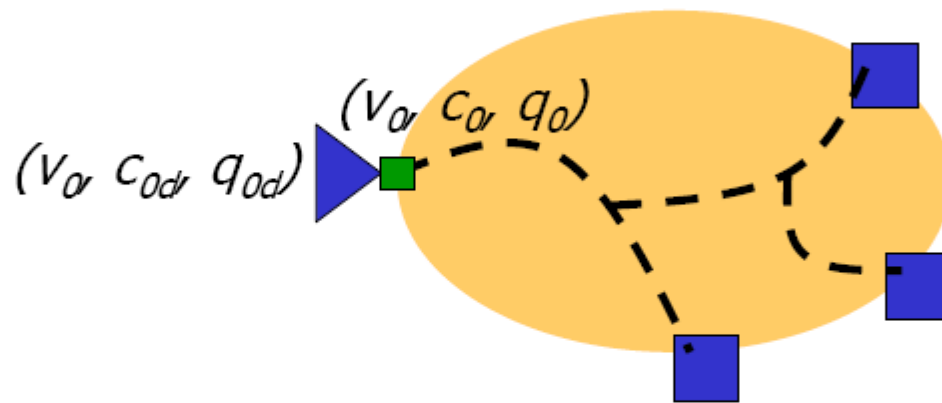
- $c_{1b} = C_b$
- $q_{1b} = q_1 - R_b c_1 - t_b$
- C_b : buffer input capacitance
- R_b : buffer output resistance
- t_b : buffer intrinsic delay

Solution Propagation: Merge



- $c_{merge} = c_l + c_r$
- $q_{merge} = \min(q_l, q_r)$

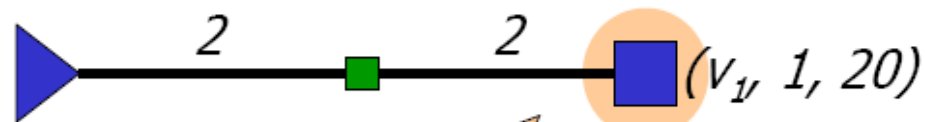
Solution Propagation: Add Driver



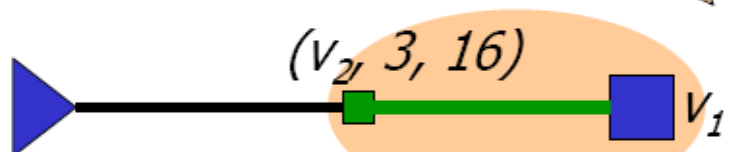
- $q_{od} = q_o - R_d c_o = \text{Slack}_{min}$
- R_d : driver resistance
- Pick solution with max Slack_{min}

Example of Solution Propagation

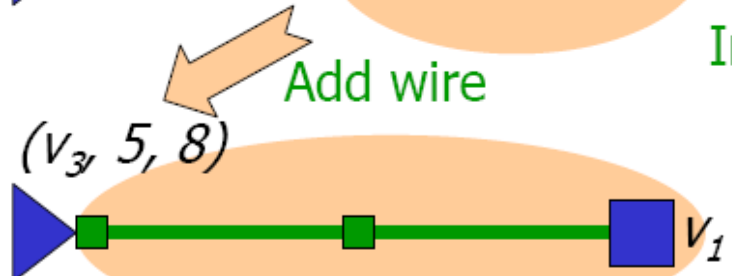
- $r = 1, c = 1$
- $R_b = 1, C_b = 1, t_b = 1$
- $R_d = 1$



Add wire

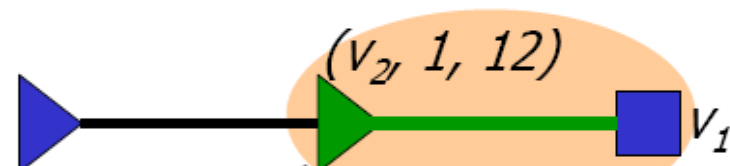


Insert buffer



Add wire

slack = 3 Add driver



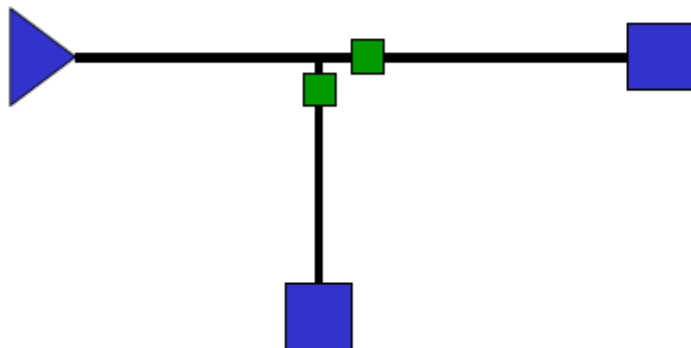
Add wire



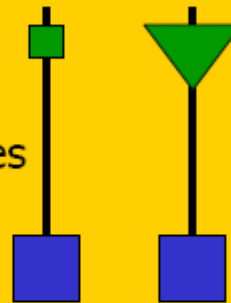
slack = 5 Add driver



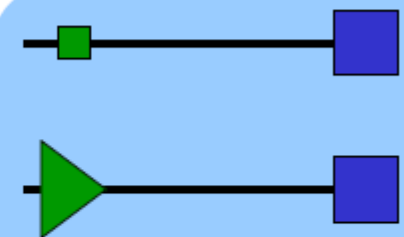
Example of Merging



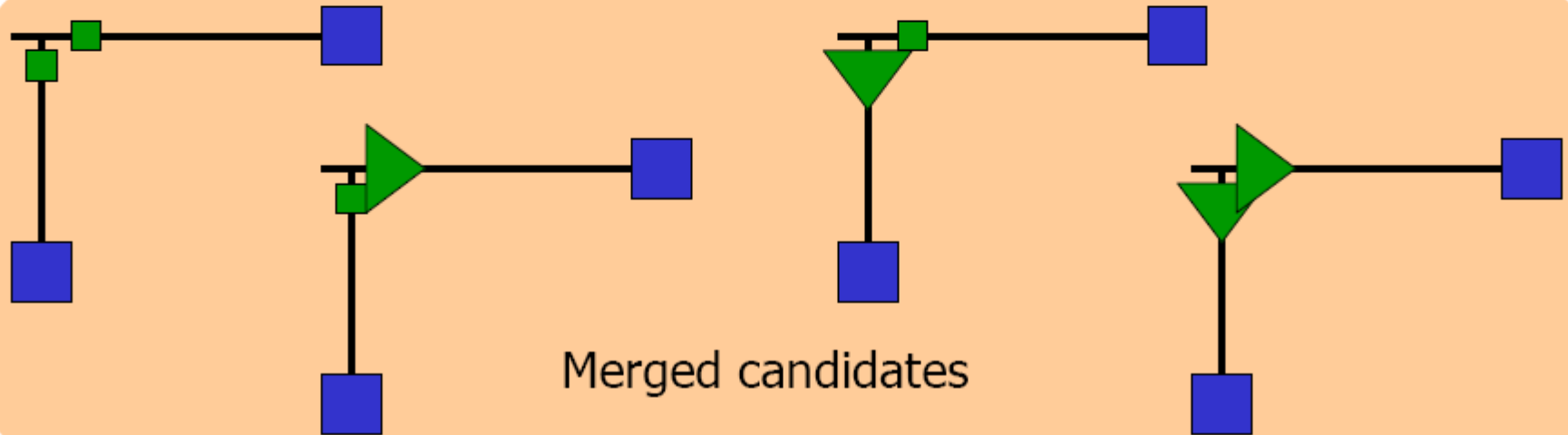
Left
candidates



Right candidates



Merged candidates





Solution Pruning

- Two candidate solutions
 - (V, c_1, q_1)
 - (V, c_2, q_2)
- Solution 1 is inferior if
 - $c_1 > c_2$: larger load
 - and $q_1 < q_2$: tighter timing



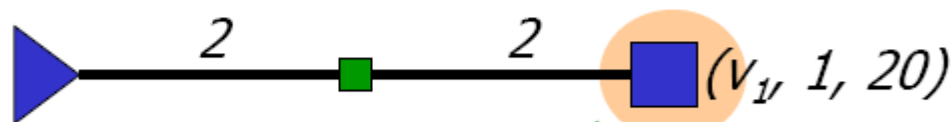
Van Ginneken Recap

- Generate candidates from sinks to source
- Quadratic runtime
 - Adding a wire does not change #candidates
 - Adding a buffer adds only one new candidate
 - Merging branches additive, not multiplicative
 - Linear time solution list pruning
- Optimal for Elmore delay model

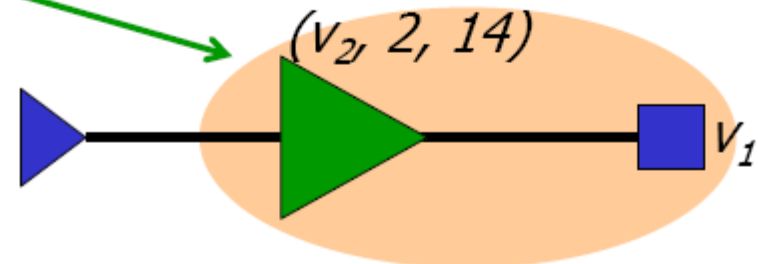
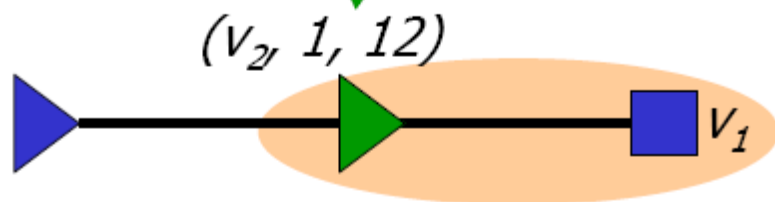
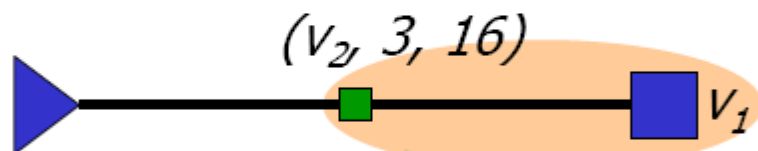
Enhancements in Buffering

- Buffer library
- Inverter/polarity
- Slew/cap constraints
- Driver/wire sizing
- Buffering blockage

Multiple Buffer Types



- $r = 1, c = 1$
- $R_b = 1, C_b = 1, t_b = 1$
- $R_{b2} = 0.5, C_{b2} = 2, t_{b2} = 0.5$
- $R_d = 1$



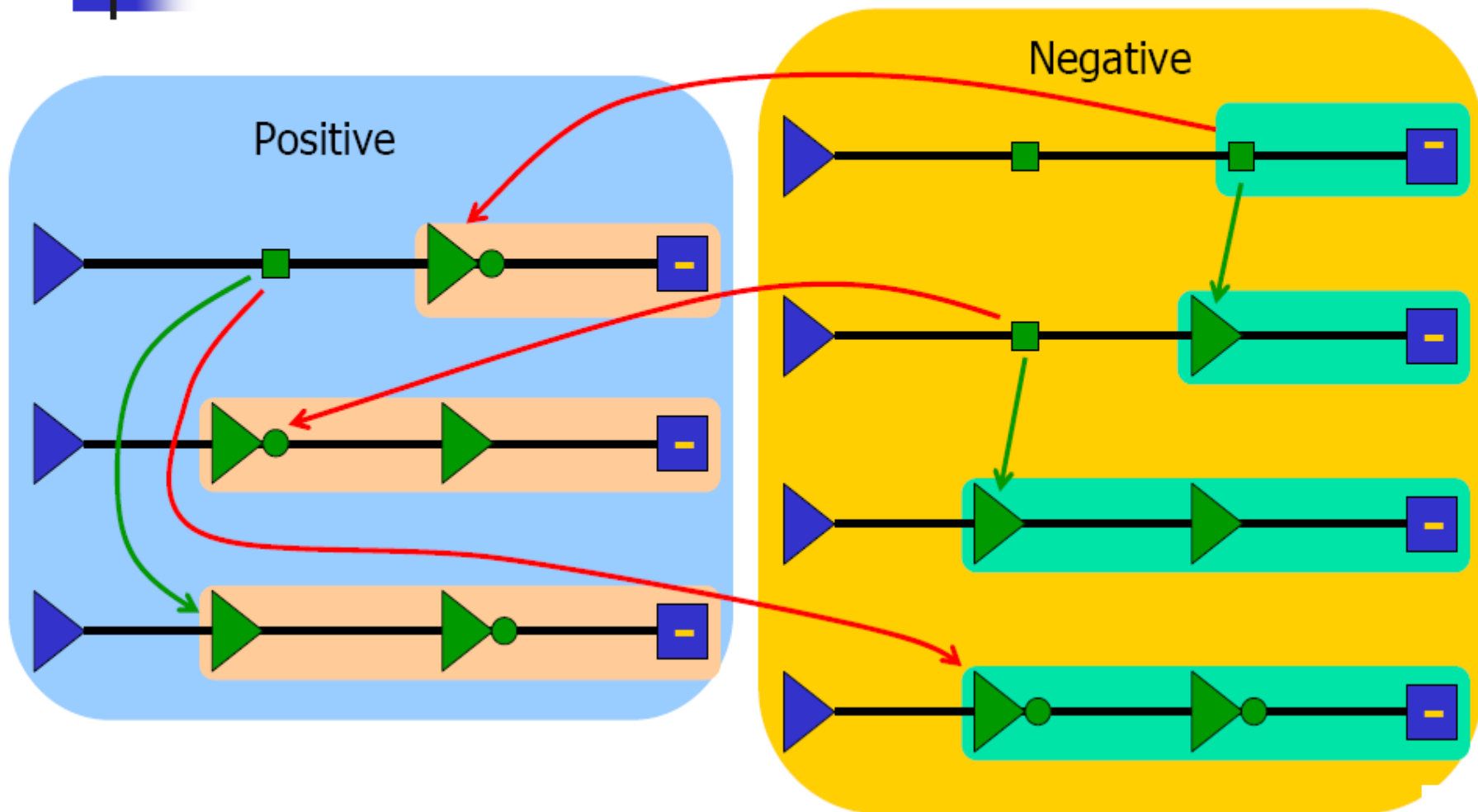


Using Inverters



Less cost

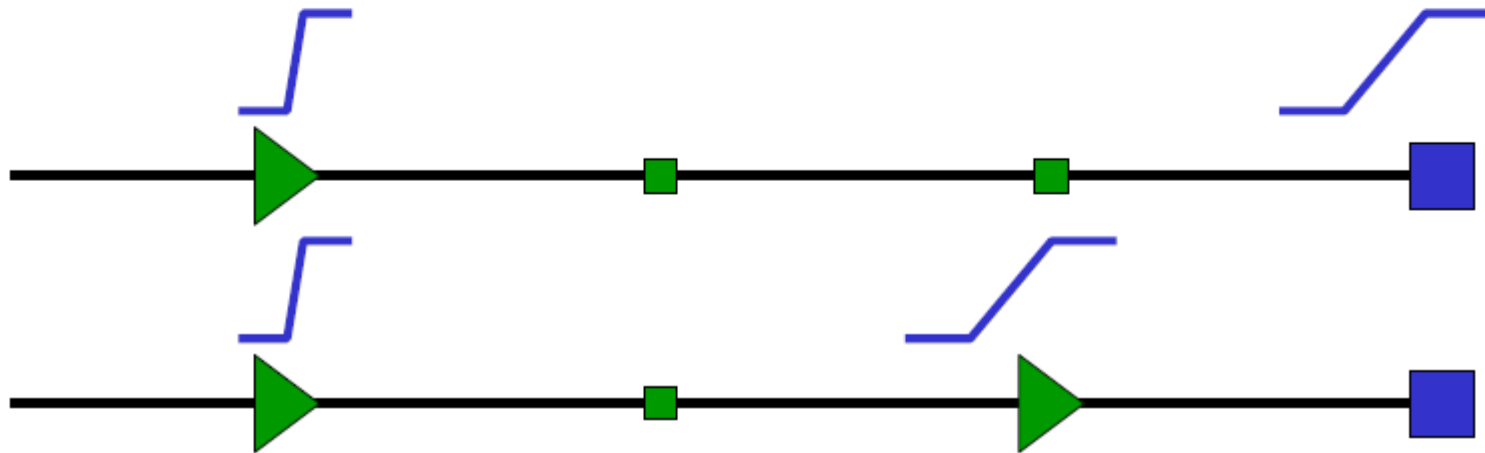
Handle Polarity





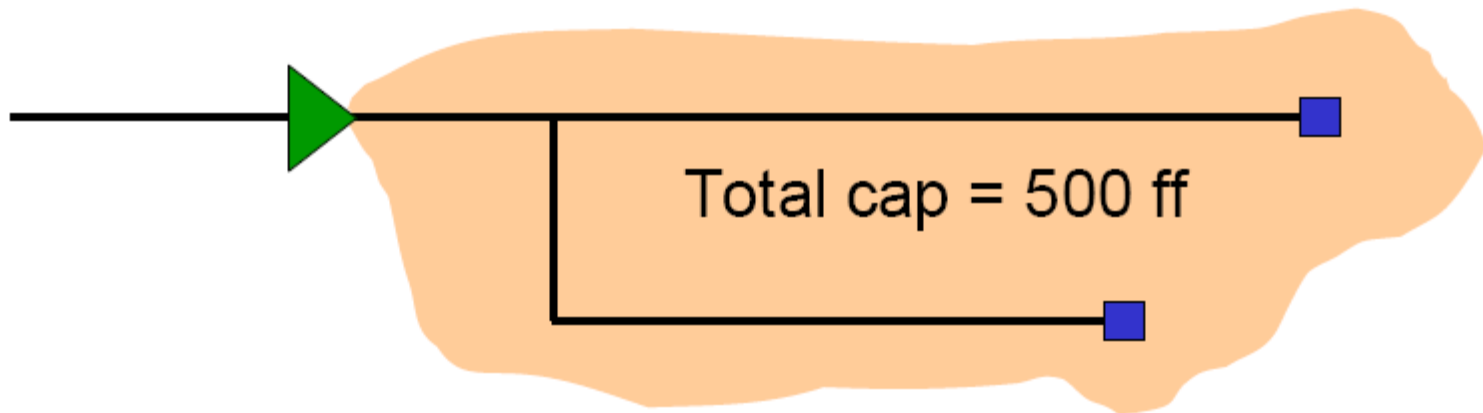
Slew Constraints

- When a buffer is inserted, assume ideal slew rate at its input
- Check slew rate at downstream buffers/sinks
- If slew is too large, candidate is discarded



Capacitance Constraints

- Each gate g drives at most $C(g)$ capacitance
- When inserting buffer g , check downstream capacitance.
- If $> C(g)$, throw out candidate



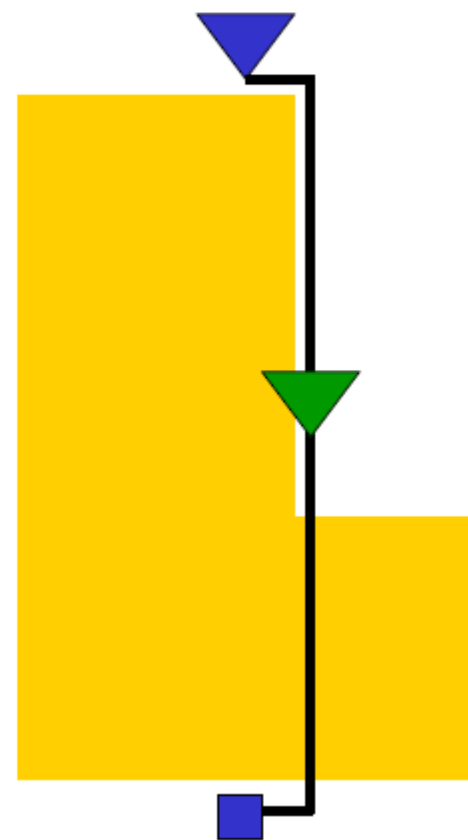
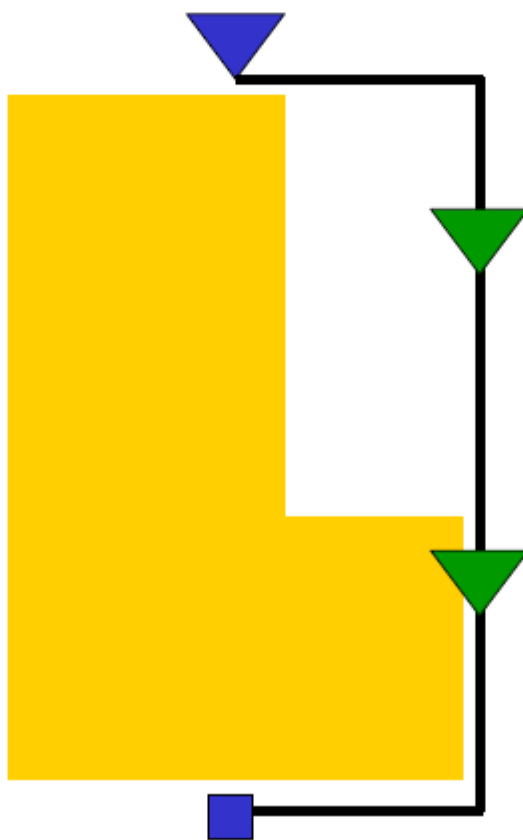
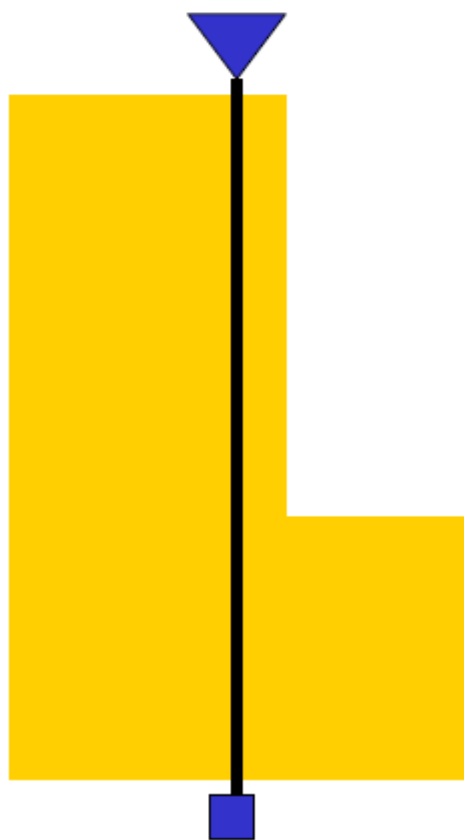


Buffer Blockage

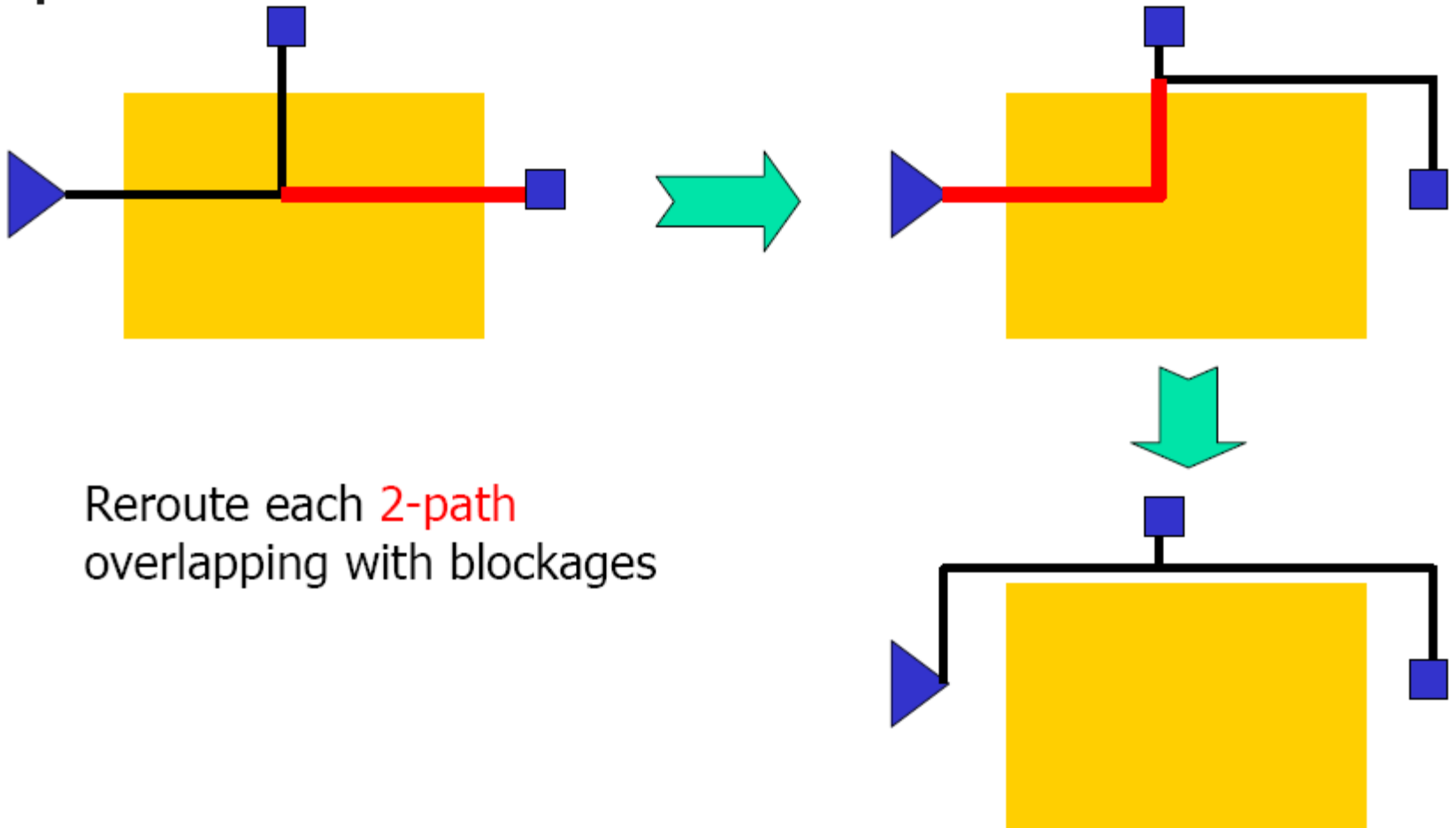


- Macro, IP core, cache ...
- Wires are allowed
- Buffers are not allowed

Wire Detour?



Rerouting to Avoid Blockages



Reroute each **2-path**
overlapping with blockages

What Makes a Design Difficult to Route?

- **Bad floorplan**
- **Over packing dense logic**
- **Inefficient area minimization**
- **Over weighting for timing-driven placement**
- **Buffering too packed**
- **Over-constraining router**
- **Not capturing local routing issues**
- **Even if successful, don't mess up timing too much**

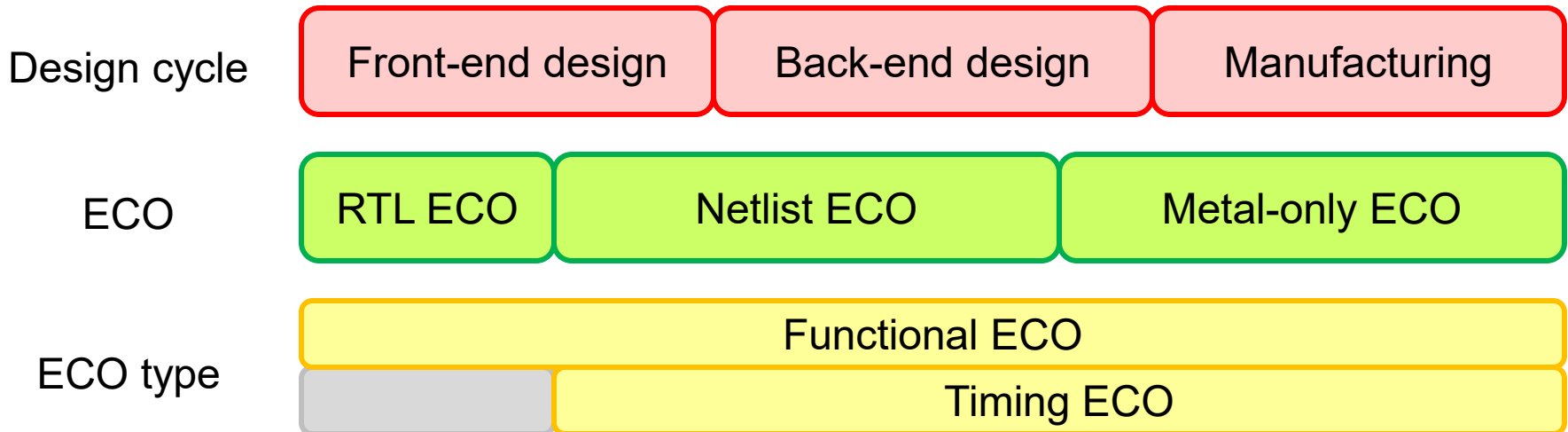
Engineering Change Order (1/3)

60

IRIS H.-R. JIANG

□ What?

- Applies incremental design changes **outside the normal flow**
- Types
 - **Functional** ECO: bug fixing, specification revision
 - **Timing** ECO: slew, loading, delay improvement



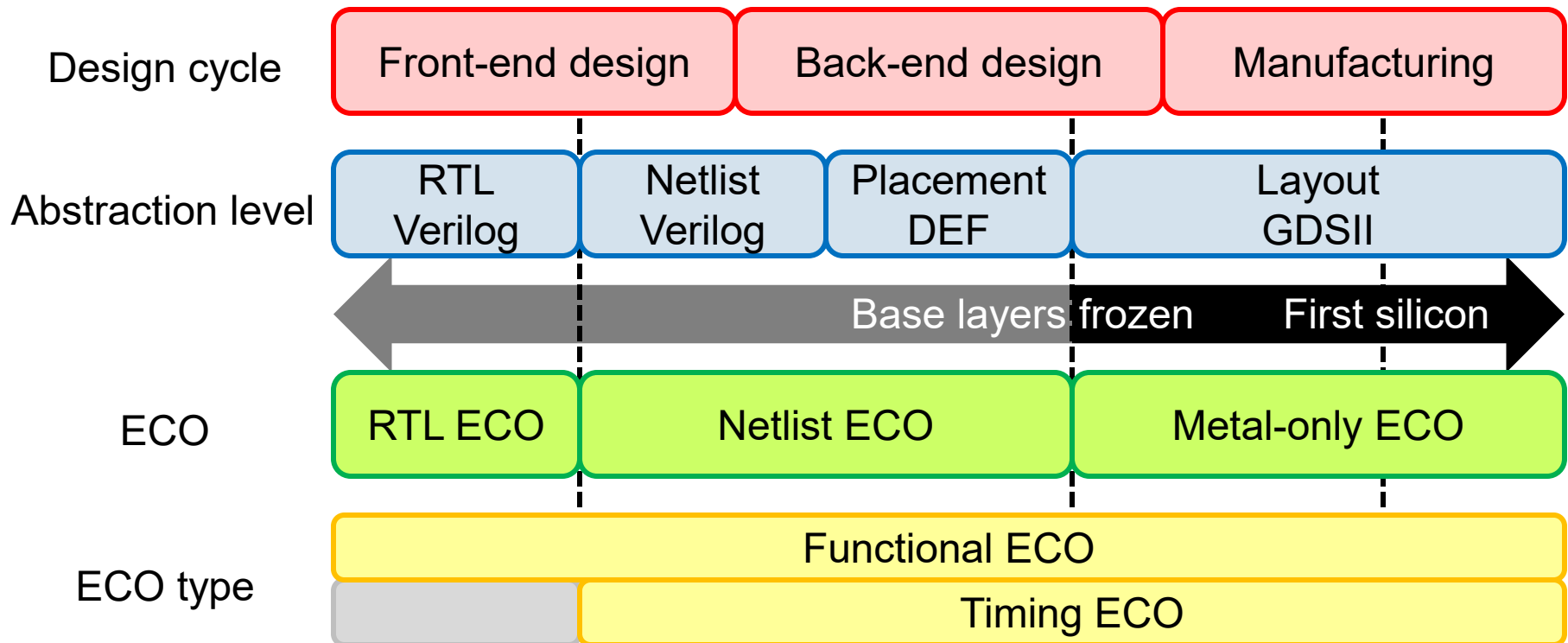
Engineering Change Order (2/3)

61

IRIS H.-R. JIANG

□ When?

- ▣ **Before** base layers are frozen: RTL / Netlist ECO
- ▣ **After** base layers are frozen: metal-only ECO
 - **How?** Inject spare cells at placement



Engineering Change Order (3/3)

62

IRIS H.-R. JIANG

□ Why?

▣ All about the money

■ **Revenue**: Shorten time-to-market (Time is money)

■ **Cost**: Reduce the photomask cost

■ The photomask cost rapidly increases

Year of Production	2007	2008	2009	2010	2011	2012	2014	2016	2018	2020	2022
Normalized mask cost from public and IDM data	1	1.3	1.7	2.3	3	3.9	6.6	11.4	19.6	33.6	57.7

■ Cells dominate! (base layer plus low metal layers)

Mask layer	M1	V1	M2	V2	M3	V3	M4	V4	M5	V5	M6	V6	M7	V7	M8
130nm	1.00	1.03	0.63	1.03	0.63	1.03	0.63	1.03	0.63	1.03	0.63	0.29	0.29	0.29	0.29
90nm	1.00	1.00	0.77	0.67	0.77	0.67	0.77	0.67	0.77	0.67	0.28	0.28	0.28	0.28	0.09

■ **Metal-only ECO** projects are popular

■ ECO projects : new projects = 118 : 74

Good Metal-Only ECO

63

IRIS H.-R. JIANG

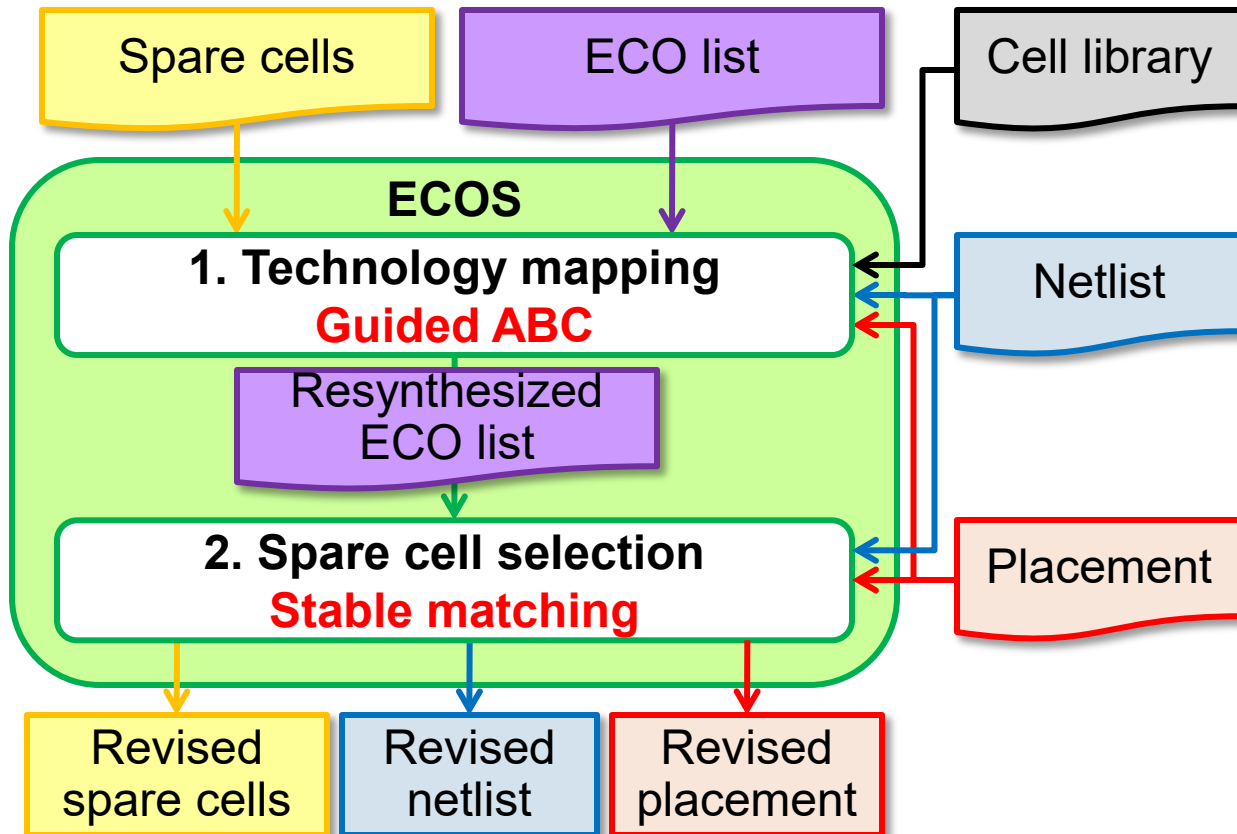
- **Facts of metal-only ECO**
 - ▣ The spare cells are **limited** both in **number** and in **cell type**
- **A good metal-only ECO relies on**
 - ▣ Sufficient and evenly sprinkled spare cells
 - Z.-W. Jiang, et al. [DAC-09]
 - ▣ A good incremental ECO-router
 - J.-Y. Li and Y.-L. Li [ISPD-05]
 - ▣ A powerful ECO synthesizer
 - **Timing** ECO: cannot extend to functional ECO
 - Y.-P. Chen, et al. [ICCAD-07]
 - C.-P. Lu, et al. [ISPD-09]
 - **Functional** ECO: cannot consider timing
 - Y.-M. Kuo, et al. [ICCAD-07]
 - **Both?**

Overview of ECOS [DAC'09]

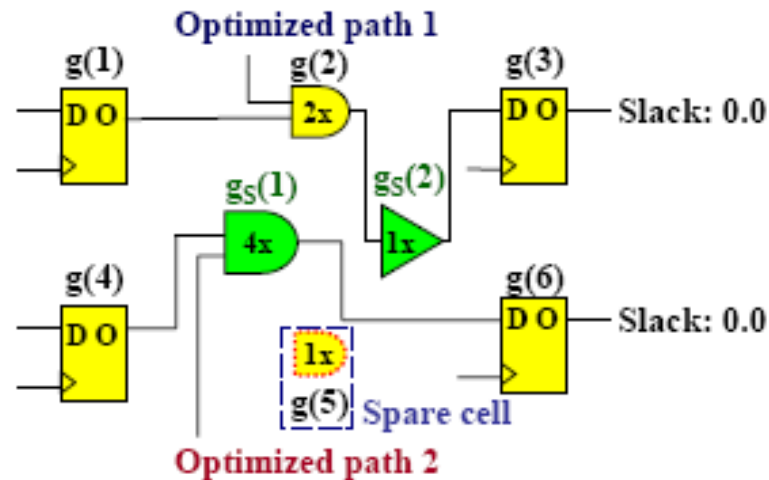
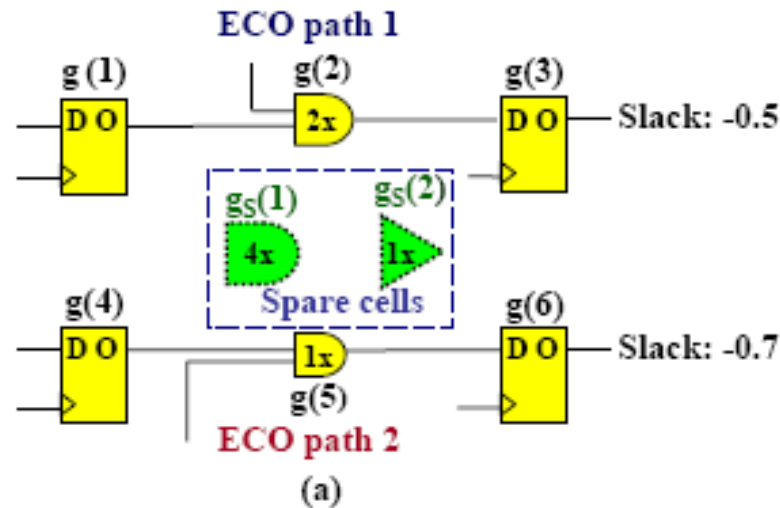
64

IRIS H.-R. JIANG

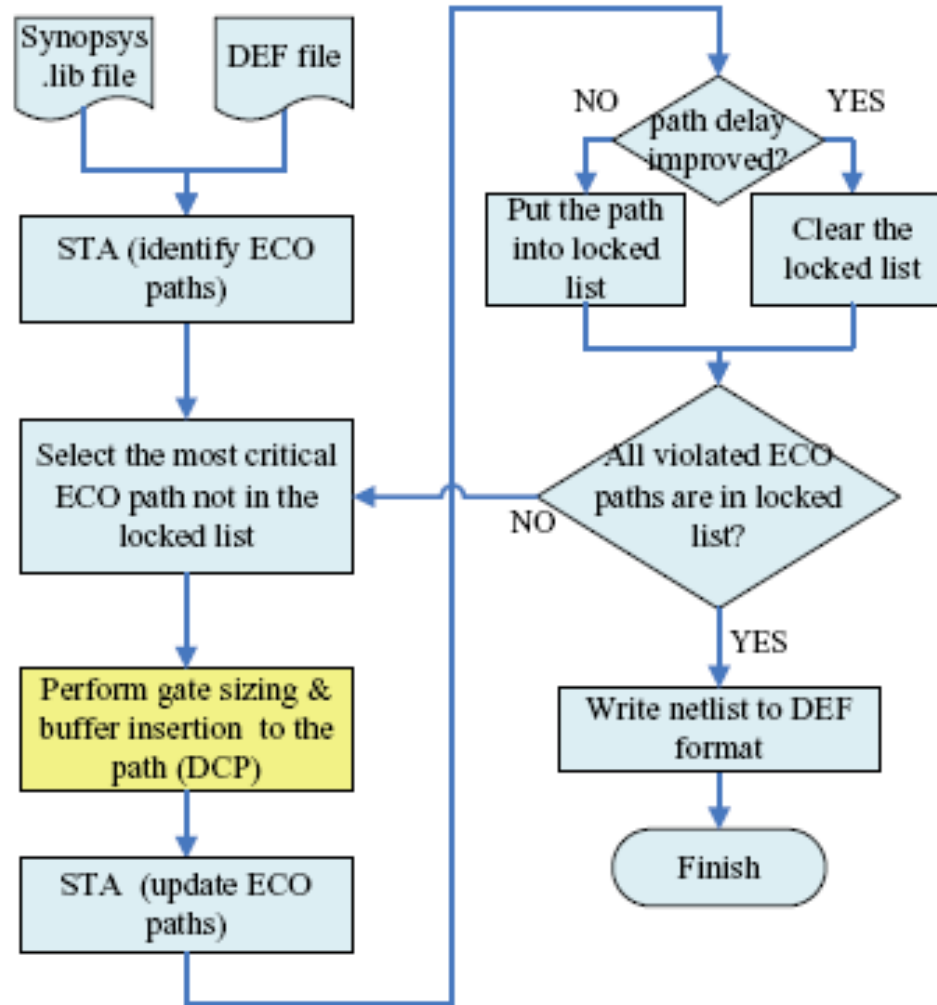
- **Goal: ECO functionality w/o sacrificing **timing** and **routability****



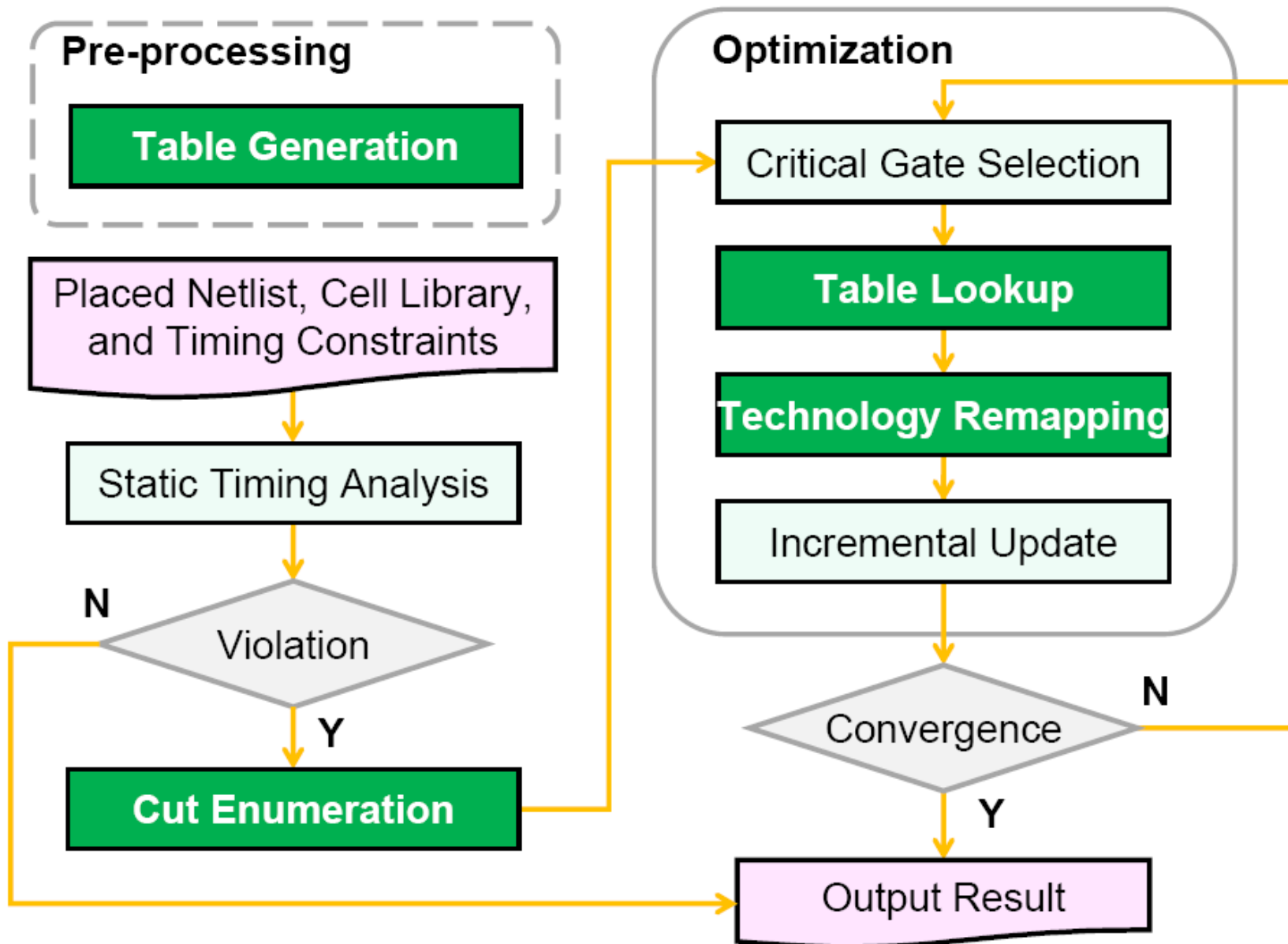
ECO Timing Optimization using Spare Cells - ICCAD'07



ECO Timing Optimization using Spare Cells



Timing ECO through Technology Remapping - ASPDAC'10



Algorithm of Technology Remapping

Technology Remapping

Rip-up & Map



Iterative wirelength reduction



Iterative timing optimization

Rip-up the target sub-circuit

Map to a new circuit with a given pattern graph randomly

Reduce total wirelength

Optimize timing of the new circuit