Physical Design

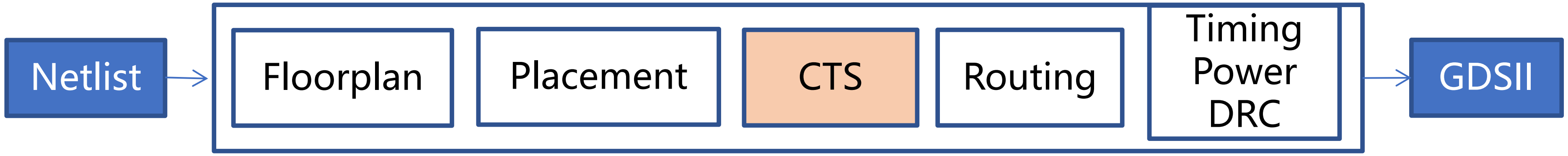
Clock Tree Synthesis

李兴权

iEDA

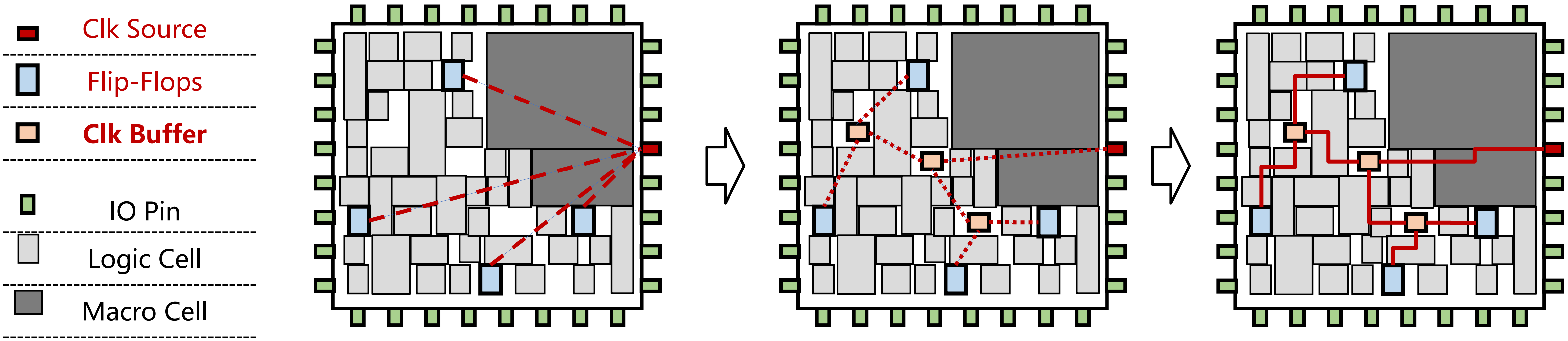
Clock Tree Synthesis (CTS)

- **CTS in Physical Design**



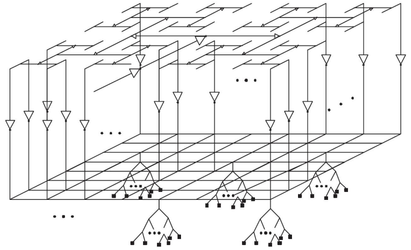
Netlist → Floorplan, Placement, CTS, Routing, Timing Power DRC → GDSII

- CTS, the bridge between Placement and Routing
- Achieving **skew balance** and **minimize design resource** (buffers, wirelength)

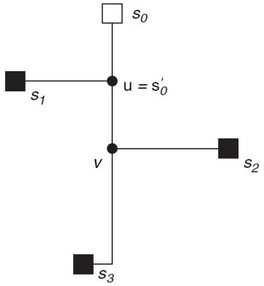


CTS

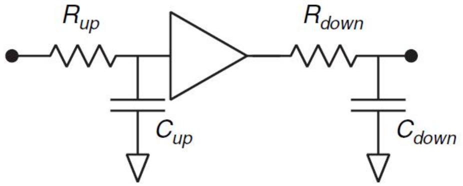
- 问题描述：时钟网络综合是希望对时钟控制单元（FF, Latch, Reg）的同步控制。确定拓扑结构和物理性状。在给定电压和时钟周期下，满足时序需要的时钟网络设计和综合。
- 输入：
  - **文件：Netlist，LEF/DEF，lib，sdc**
  - **单元库，线网，布局结果，时序约束，布线层**
- 输出
  - **文件：DEF，时钟Net**
- 目标：
  - **Skew，Latency，Transition，Power，IR drop，Capacitance, Fan out，Variations**
- 约束：
  - **实现PLL可控制所有时钟单元，Max transition, Max capacitance, Max fan-out, list of buffers and inverters**



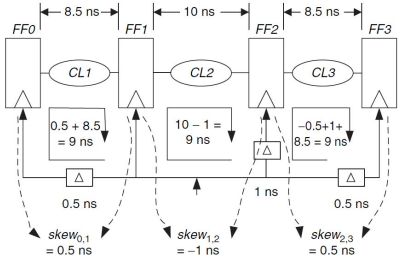
时钟网络设计：在高level级别设计时钟网络拓扑结构



时钟网络布线：为每个时钟树节点buffer到它的子节点buffer的时钟拓扑实现物理线网

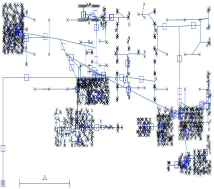


时钟网络优化：通过buffer，gating，wire，cross link等手段优化时钟目标

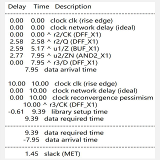


时钟网络综合：通过对整个时钟树的FF进行调度，实现更优的Skew目标

Clock Tree Synthesis (CTS)

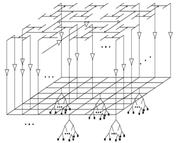


Clock Tree Synthesis
时钟网络综合



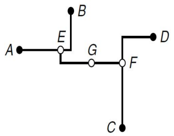
```
Delay   Time   Description
-----------------------------------------------
 0.00    0.00   clock clk (rise edge)
 0.00    0.00   clock network delay (ideal)
 0.00    0.00 ^ r2/CK (DFF_X1)
 2.58    2.58 ^ r2/Q (DFF_X1)
 2.59    5.17 ^ u1/Z (BUF_X1)
 2.77    7.95 ^ u2/ZN (AND2_X1)
 0.00    7.95 ^ r3/D (DFF_X1)
         7.95   data arrival time

10.00   10.00   clock clk (rise edge)
 0.00   10.00   clock network delay (ideal)
 0.00   10.00   clock reconvergence pessimism
        10.00 ^ r3/CK (DFF_X1)
-0.61    9.39   library setup time
         9.39   data required time
-----------------------------------------------
         9.39   data required time
        -7.95   data arrival time
-----------------------------------------------
         1.45   slack (MET)
```

Clock Constraint
时钟约束



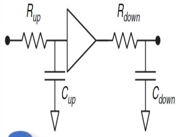
Clock Tree Design
时钟网络设计

- 树拓扑结构
- 网络拓扑结构
- 混合拓扑结构
- 聚类时钟拓扑



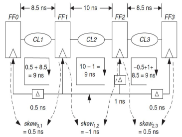
Clock Tree Routing
时钟网络布线

- 零Skew时钟布线
- 有界Skew时钟布线
- 有用Skew时钟布线
- 带障碍的时钟布线



Clock Tree Optimization
时钟网络优化

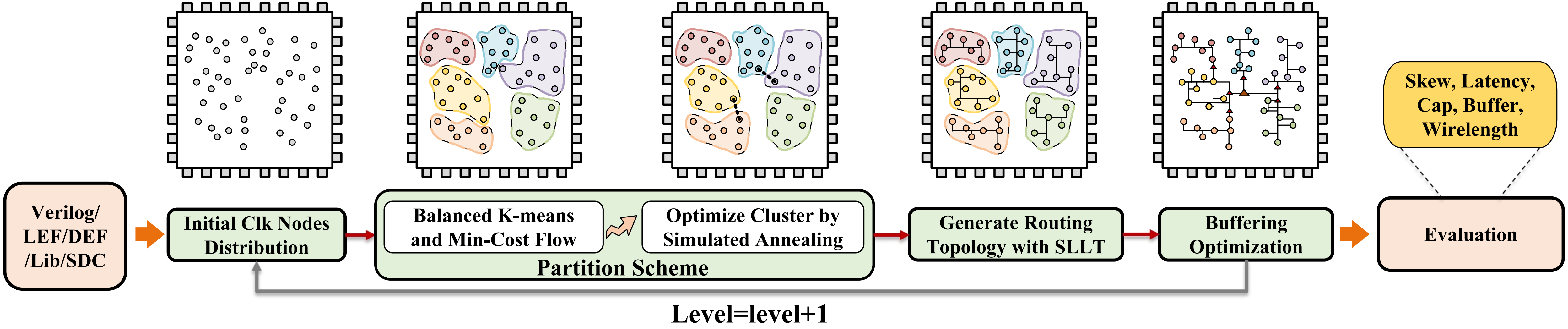
- Buffer Insertion (Transition)
- Clock Gating (Power)
- Wire Sizing (Power, Latency, IR drop, 电迁移)
- Wire Snaking (Skew)
- Cross Liking (Skew)



Clock Tree Synthesis
时钟网络综合

- Skew调度
- 分频时钟控制
- 异步时钟控制

Hierarchical Framework*



Verilog/ LEF/DEF /Lib/SDC

Initial Clk Nodes Distribution

Balanced K-means and Min-Cost Flow

Optimize Cluster by Simulated Annealing

Partition Scheme

Generate Routing Topology with SLLT

Buffering Optimization

Evaluation

Skew, Latency, Cap, Buffer, Wirelength

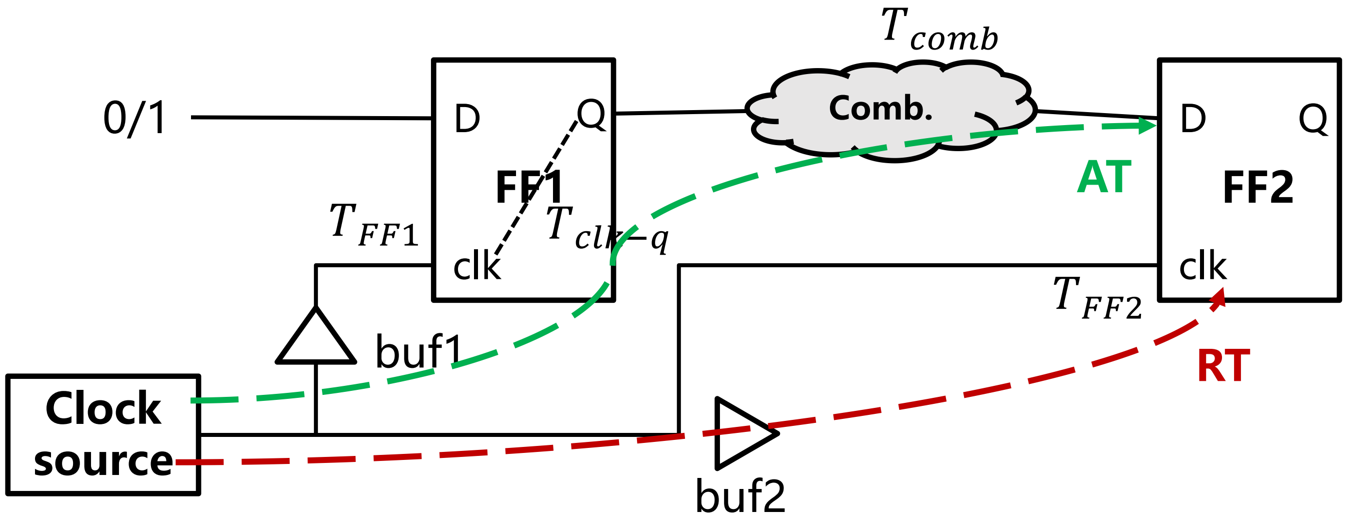
Level=level+1

Implications on Timing

- Let' s remember our famous timing constraints:
  - $\Delta_{skew} = T_{FF1} - T_{FF2}$

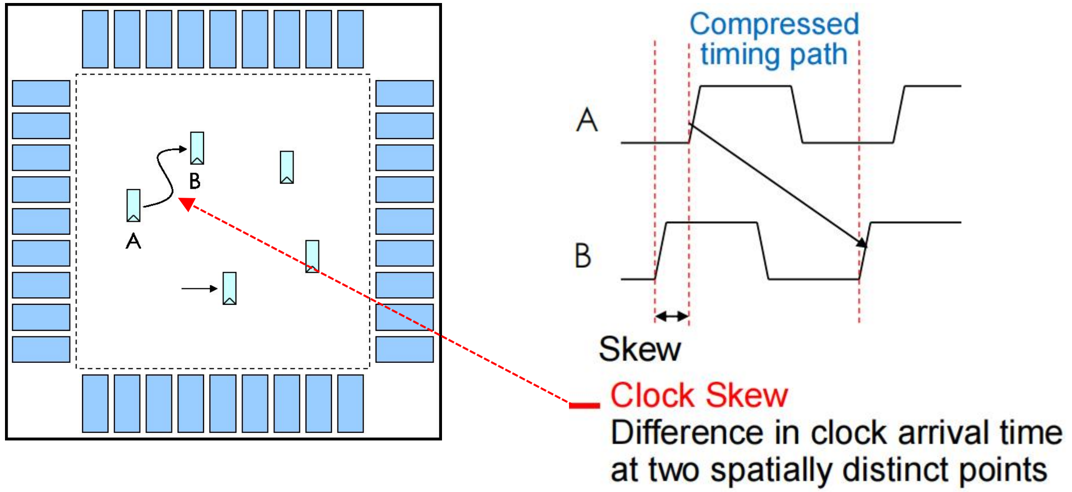
$$T_{FF1} + T_{clk-q} + T_{comb} + T_{setup} - T_{FF2} - T = T_{slack}^{late} \geq 0$$ **Setup Constraint**

$$T_{FF1} + T_{clk-q} + T_{comb} - T_{hold} - T_{FF2} = T_{slack}^{early} \geq 0$$ **Hold Constraint**



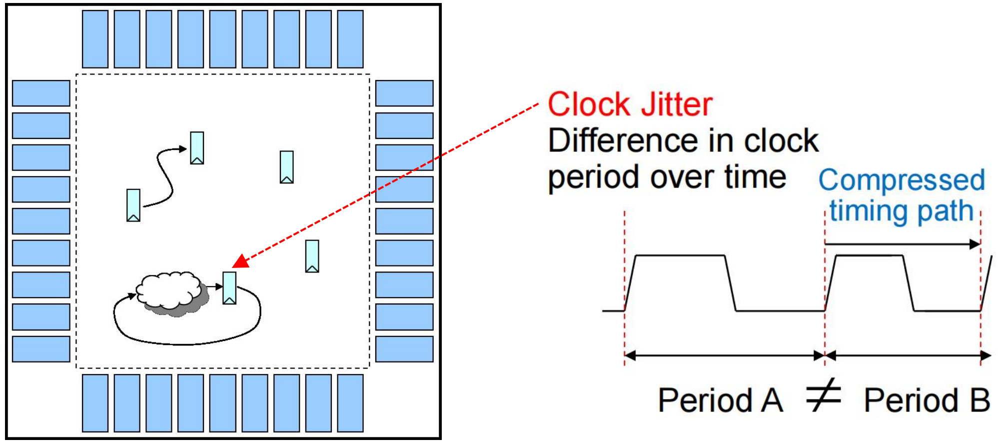
Clock Parameters

- Skew
  - Difference in clock arrival time at two different registers.
- Jitter
  - Difference in clock period between different cycles.
- Slew
  - Transition (T_{rise}/T_{fall}) of clock signal.
- Insertion Delay
  - Delay from clock source until register



Clock Skew

Difference in clock arrival time at two spatially distinct points



Clock Jitter

Difference in clock period over time

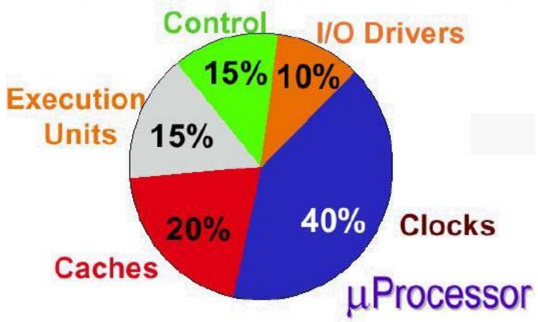
Implications on Power

- Let’ s remember how to calculate dynamic power:

$$P_{\mathrm{dyn}} = f \cdot C_{\mathrm{eff}} \cdot V_{\mathrm{DD}}^{2}$$

$$C_{\mathrm{eff}} \triangleq \alpha \cdot C_{\mathrm{total}} = \alpha_{\mathrm{clock}} \cdot C_{\mathrm{clock}} + \alpha_{\mathrm{others}} \cdot C_{\mathrm{others}}$$

  - The activity factor (α) of the clock network is 100%!
- The clock capacitance consists of:
  - Clock generation (i.e., PLL, clock dividers, etc.)
  - Clock elements (buffers, muxes, clock gates)
  - Clock wires
  - Clock load of sequential elements



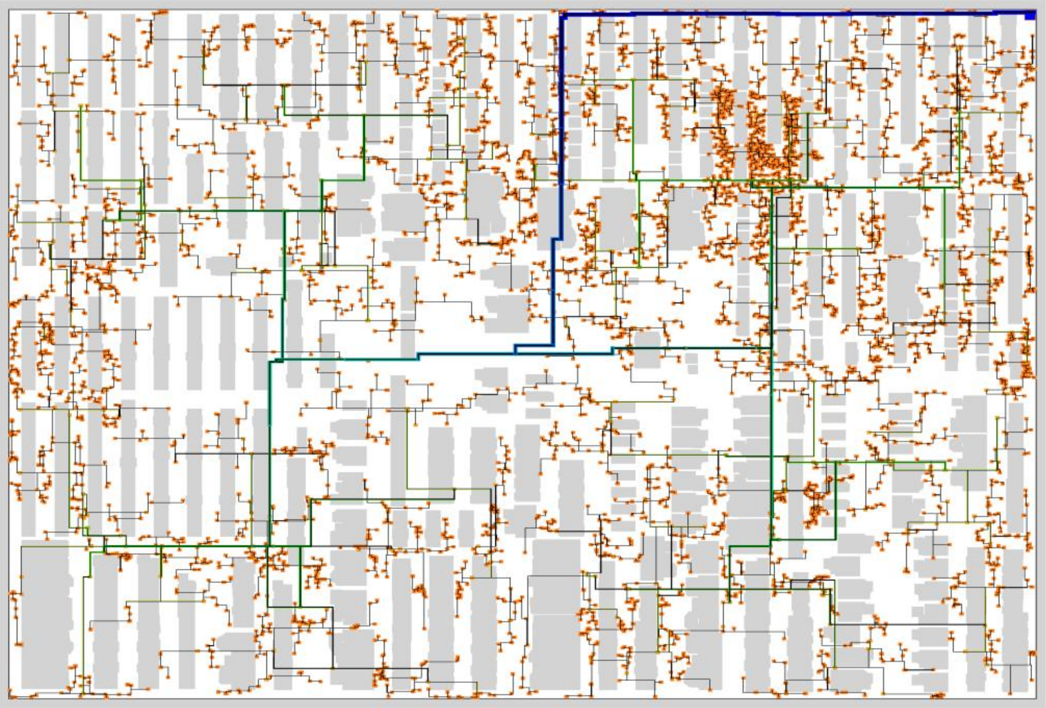
- Clock networks are huge
  - And therefore, the clock is responsible for a large percentage of the total chip power

Implications on Area

- To reiterate, clock networks consist of:
  - Clock generators
  - Clock elements
  - Clock wires
- All of these consume area
  - Clock generators (e.g., PLL) can be very large
  - Clock buffers are distributed all over the place
  - Clock wires consume a lot of routing resources
- Routing resources are most vital
  - Require low RC (for transition and power)
    - Benefit of using high, wide metals
  - Need to connect to every clock element (FF)
    - Distribution all over the chip
    - Need Via stack to go down from high metals

Intel Itanium

- 4% of M4/M5 used for clock routing



The Clock Tree Synthesis Problem

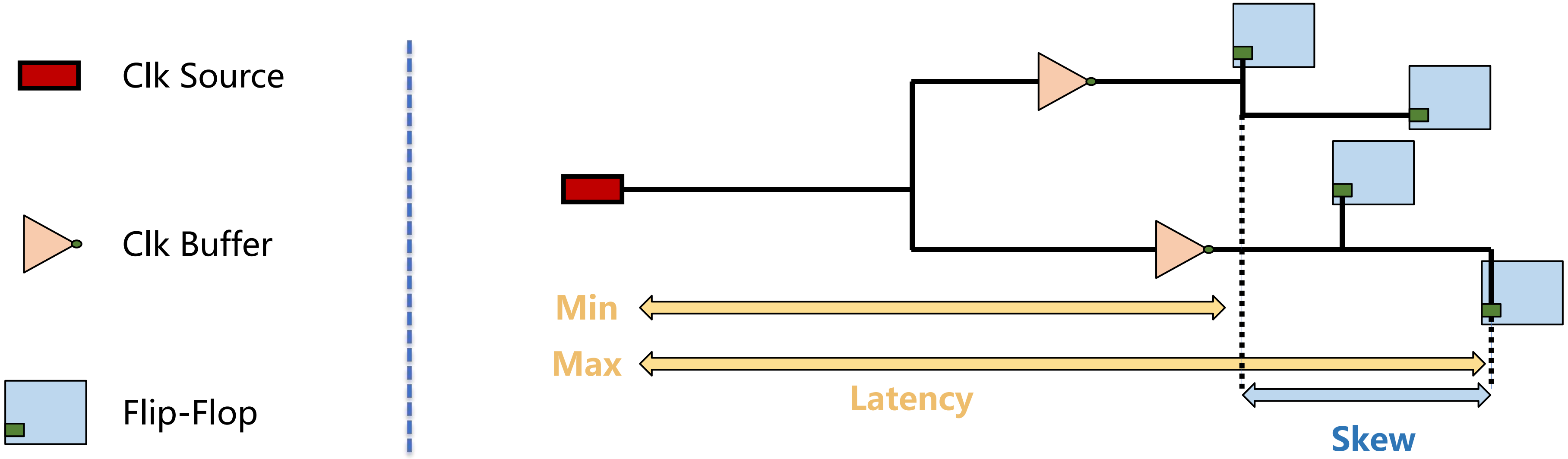
- Given a source and n sinks:
  - Connect all sinks to the source by an interconnect network so as to minimize:
    - Clock Skew = maxi,j
    - Latency (Delay) = maxi
    - Total wirelength
    - Buffer number
    - Noise and coupling effect

Clock Tree goals:
- 1) Minimize Skew
- 2) Meet target Latency (min/max)

- The Challenge:
  - Synchronize millions (billions) of separate elements
    - Within a time scale on order of ~10 ps
    - At distances spanning 2-4 cm
    - Ratio of synchronizing distance to
    - element size on order of 105
    - Reference: light travels <1 cm in 10 ps

Clock Tree constraints:
- 1) Maximum Transition
- 2) Maximum Load Cap
- 3) Maximum Fanout
- 4) Maximum Buffer Levels

Skew and Latency

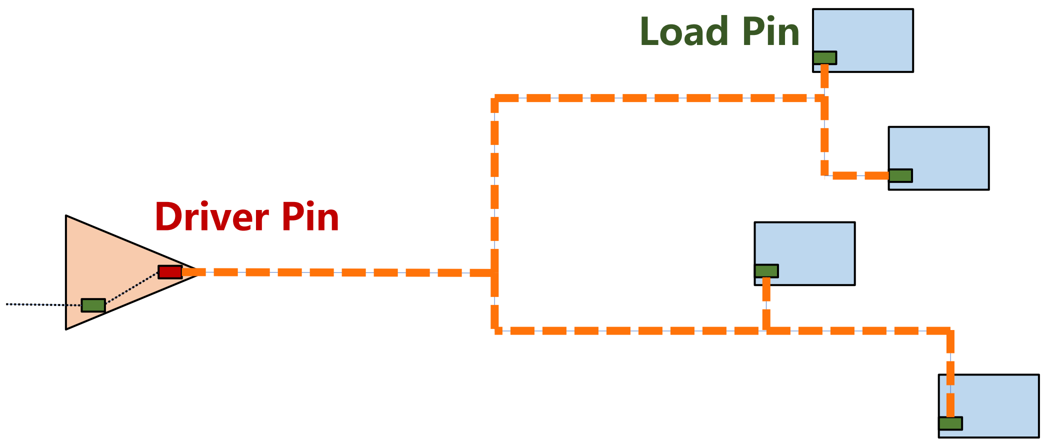


- Timing of Clock Tree
  - skew
  - latency
  - worst negative slack (WNS)
  - total negative slack (TNS)
  - …
- Design Resource
  - num of buffer insertions
  - wirelength
  - capacitance
  - power/area
  - …

What’s the operation in CTS

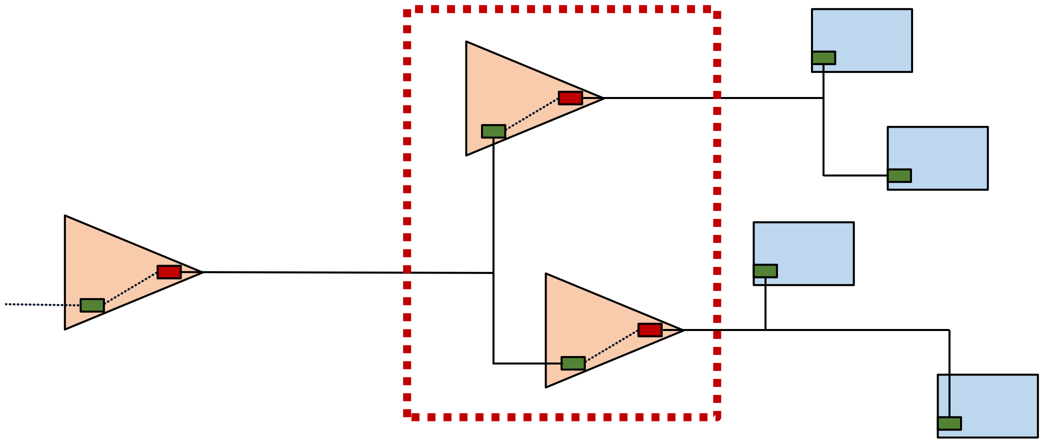
- Routing Topology
  - starting from the **driver pin**
  - connect all **load pins**
  - Steiner tree properties

  > *delay/wirelength/capacitance/…*

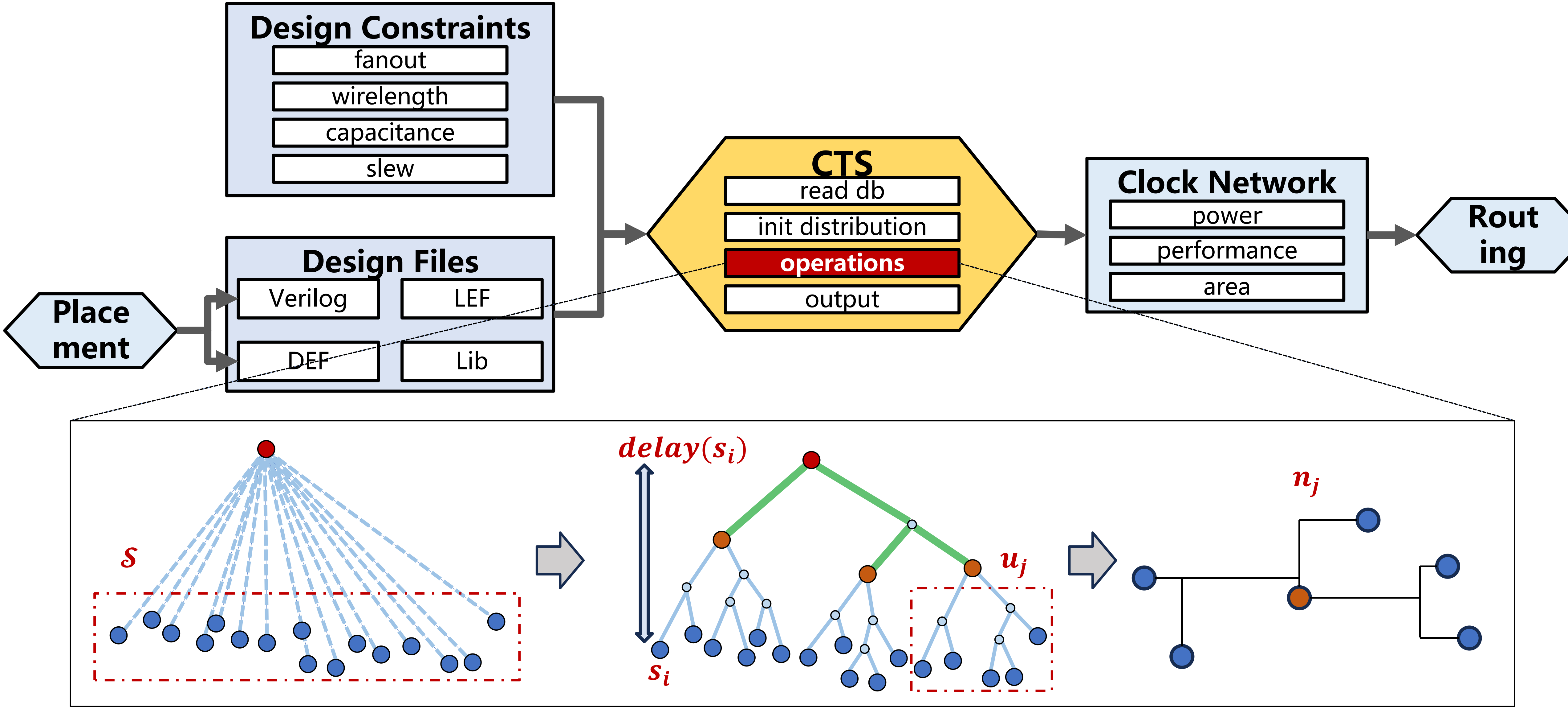


- Buffering
  - decompose the net
  - reduce fanout (4 → 2)
  - enhanced driving capability

  > *cell delay/area/power/slew/…*



CTS Flow



Design Constraints
- fanout
- wirelength
- capacitance
- slew

Design Files
- Verilog
- LEF
- DEF
- Lib

Placement

CTS
- read db
- init distribution
- **operations**
- output

Clock Network
- power
- performance
- area

Routing

Skew Constraint

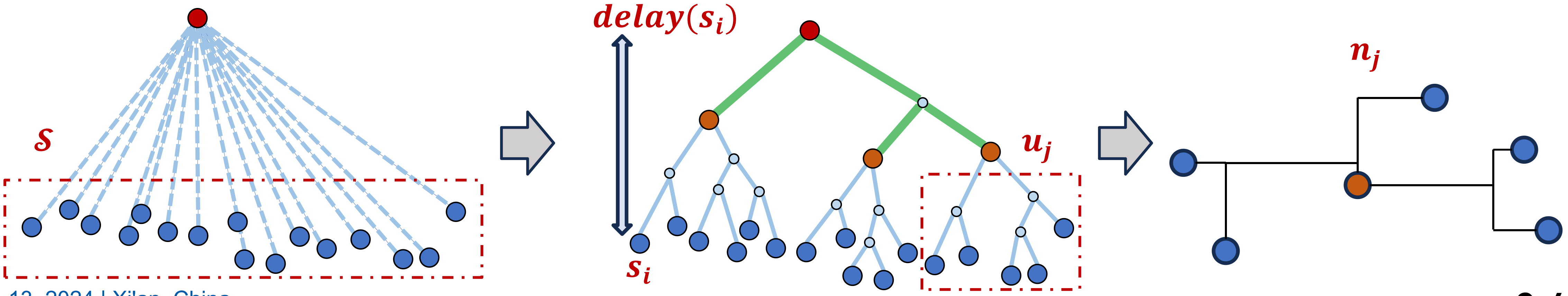
- **We define** the initial distribution of **flip-flops (sinks)** in the CTS stage as $\mathcal{S}$.

Given a clock source (root), we can establish **clock skew** constraint as follow:

$$\Delta delay(s_i) = \max_{s_i \in \mathcal{S}}\{delay(s_i)\} - \min_{s_i \in \mathcal{S}}\{delay(s_i)\} \leq skew\_bound,$$

where $delay(s_i)$ represents the delay from clock source to sink s_i (latency).

We define the set of load pins for each clock net as a **cluster**, i.e., $u_j \in \mathcal{U}$, and generate a clock net, i.e., $n_j \in \mathcal{N}$.



Fanout, Wirelength, Capcitance Constraints

- CTS needs to satisfy additional design constraints, such as the **fanout** constraint:

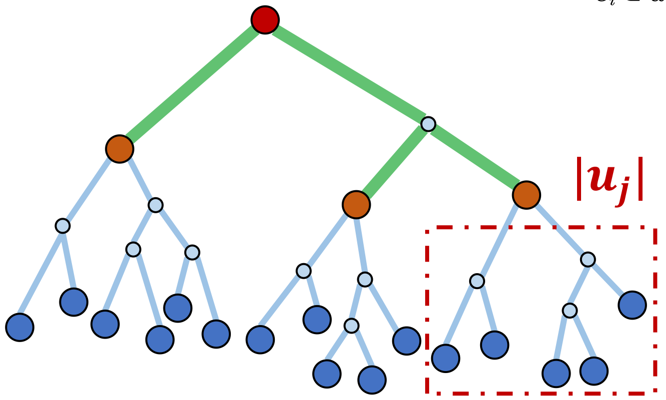
$$|u_j| \leq max\_fanout,$$

- the maximum **wirelength** constraint:

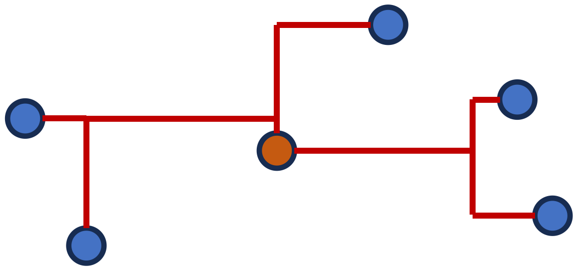
$$WL\left(n_j\right) \leq max\_length,$$

- and the maximum clock net **capacitance** constraint:

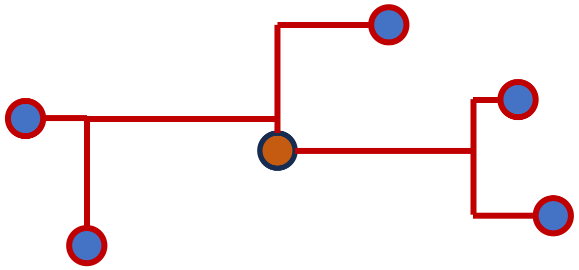
$$\sum_{s_i \in u_j} Cap_{pin}(s_i) + c \cdot WL\left(n_j\right) \leq max\_cap.$$



$WL(n_j)$



$Cap(n_j)$



CTS Problem

- CTS problem can be defined as a multi-objective optimization problem:

$$\begin{aligned} \min \quad & f(\mathcal{N},\mathcal{U},\mathcal{S}) \\ s.t. \quad & \Delta delay(s_i) \leq skew\_bound, \\ & |u_j| \leq max\_fanout, \\ & WL(n_j) \leq max\_length, \\ & \sum_{s_i \in u_j} Cap_{pin}(s_i) + c \cdot WL(n_j) \leq max\_cap, \end{aligned}$$

$f(\mathcal{N},\mathcal{U},\mathcal{S})$ represents the objective function, such as **design resources** and **power**.

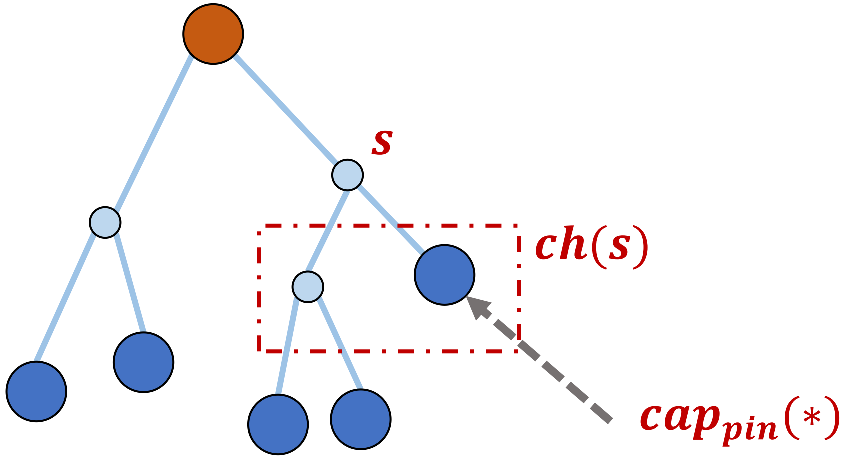
It’s a complex multi-objective optimization problem

Evaluation: Load Capacitance

- Equivalent Capacitance (Linear)

Capacitance per unit length/area

$$cap_{load}(s) = \underbrace{cap_{pin}(s)}_{\text{Pin Cap}} + \sum_{t \in ch(s)} (\underbrace{Cap_{load}(t)}_{\textbf{Tree (Recursive)}} + \underbrace{c \cdot L(s,t)}_{\text{Wire Cap}}),$$



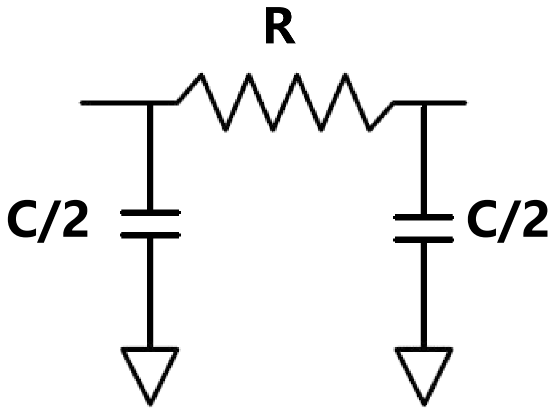
Bottom-up Calculation

Evaluation: Wire (Interconnect) Delay

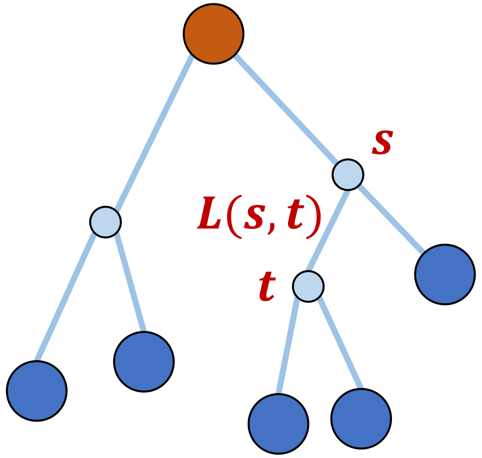
- Elmore π-Model

$$D_{wire}(s,t) = \underbrace{r \cdot L(s,t)}_{\text{Res}} \cdot \underbrace{\left[\frac{1}{2} \cdot c \cdot L(s,t) + Cap_{load}(t)\right]}_{\text{Cap}},$$

Resistance per unit length/area



Equivalent π-model of a wire segment

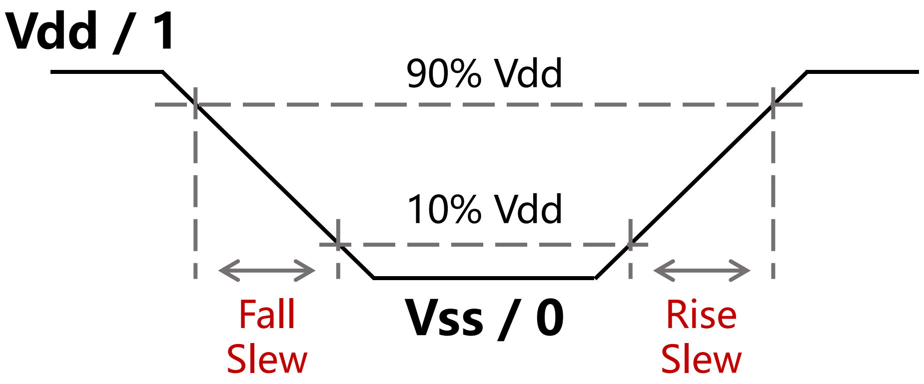


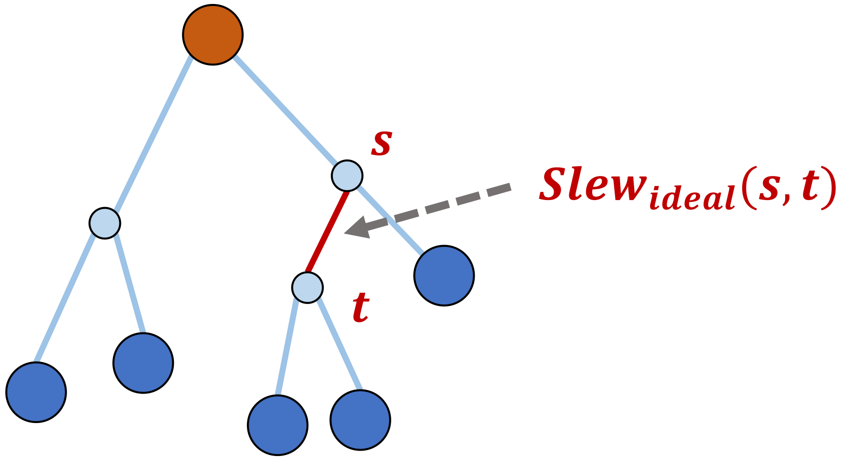
Evaluation: Slew (Transition)

- Bakoglu Metric[1] & PERI Model[2]

$$Slew_{ideal}(s,t)=\ln 9\cdot D_{wire}(s,t),$$

$$Slew_{wire}(t)=\sqrt{Slew_{wire}^{2}(s)+Slew_{ideal}^{2}(s,t)}\,.$$

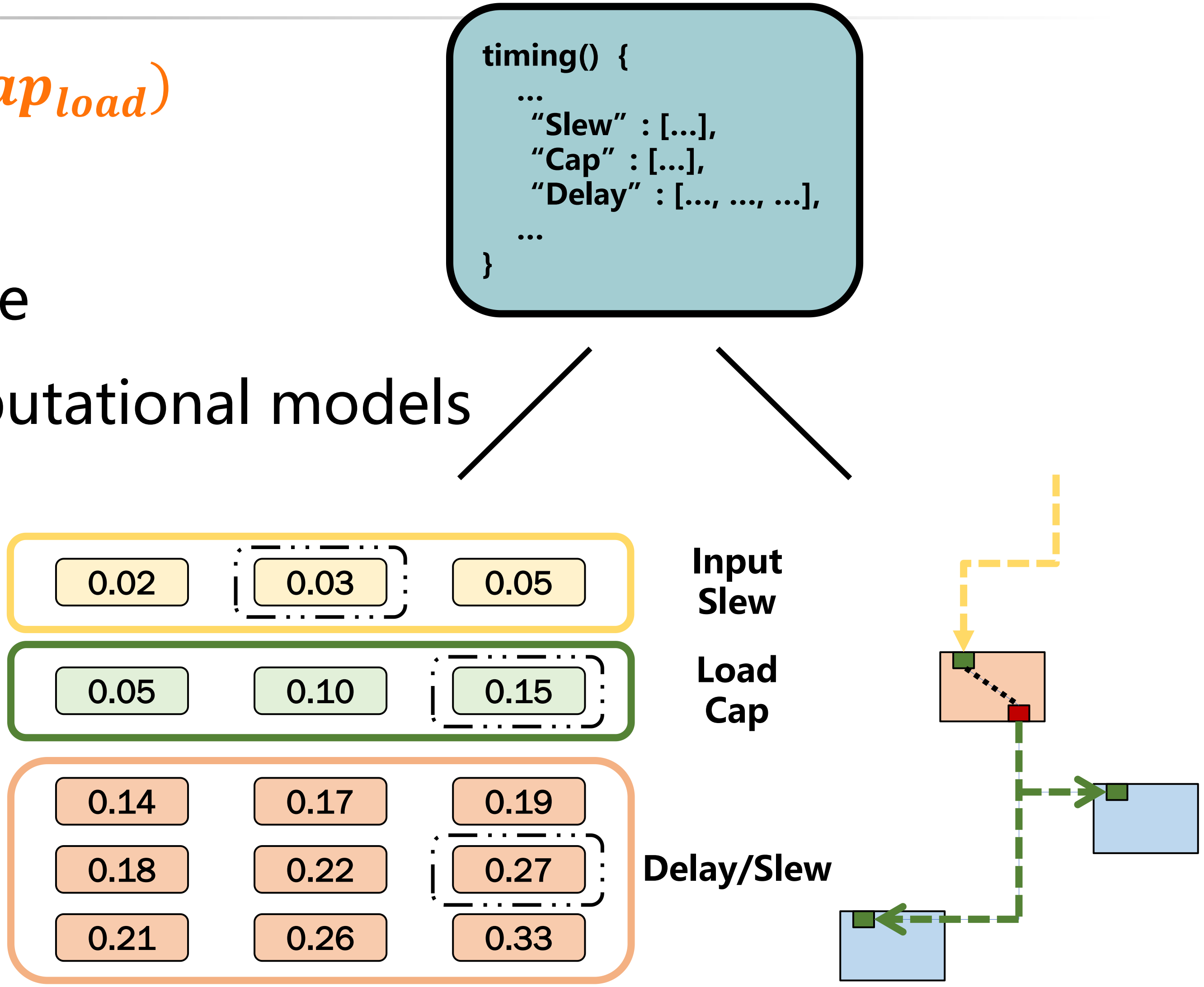




Top-down Calculation

Look-up Table (LUT)

- $Delay/Slew_{out} = LUT(Slew_{in}, Cap_{load})$
  - upstream input slew
  - downstream load capacitance
  - not reliant on complex computational models
- Model
  - bilinear interpolation
  - ML fitting…



```
timing() {
    …
     "Slew" : […],
     "Cap" : […],
     "Delay" : […, …, …],
    …
}
```

0.02	0.03	0.05	Input Slew
0.05	0.10	0.15	Load Cap
0.14	0.17	0.19	
0.18	0.22	0.27	Delay/Slew
0.21	0.26	0.33	

Evaluation: Buffer Delay

- Linear Fitting[3]

$$D_{buf}(*) = LUT_{delay}(Slew_{in}(*), Cap_{load}(*)),$$

$$\Downarrow$$

$$D_{buf}(*) = \alpha \cdot Slew_{in}(*) + \beta \cdot Cap_{load}(*) + \underbrace{\gamma}_{\text{Intrinsic Delay}}.$$

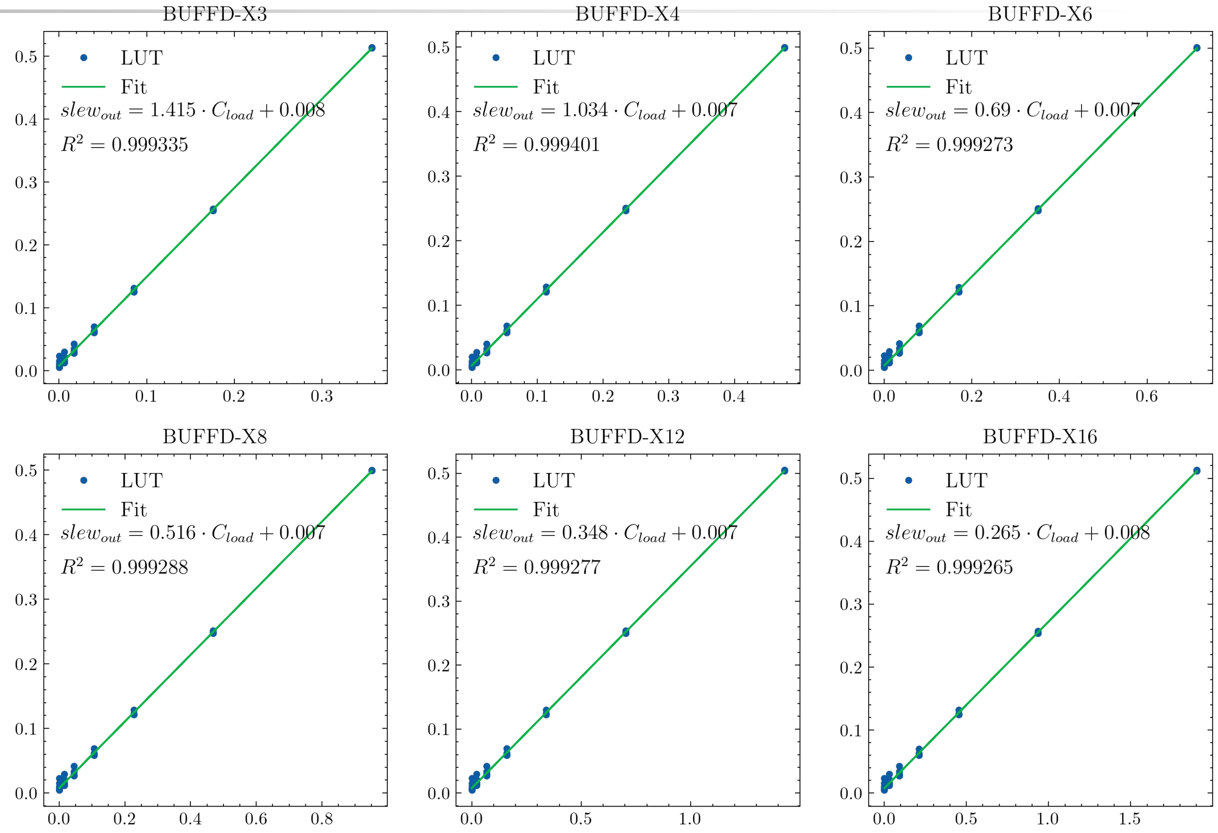
This type of modeling is only effective for clock buffers

Evaluation: Buffer Slew

- Linear Fitting[3]

$$Slew_{out}(*) = LUT_{slew}(Slew_{in}(*), Cap_{load}(*)),$$

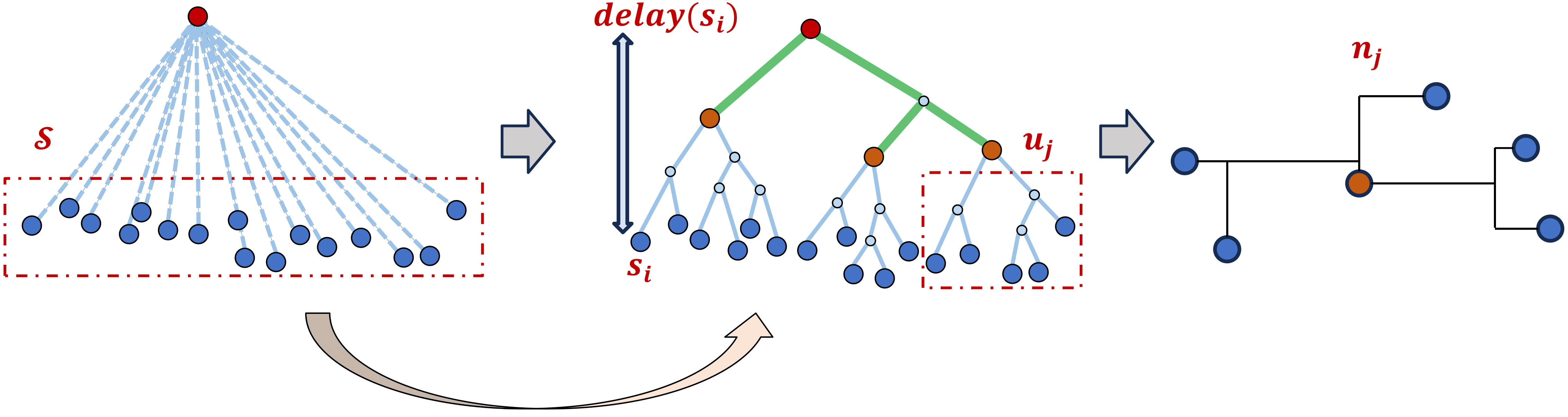
$$Slew_{out}(*) = \alpha \cdot Cap_{load}(*) + \beta.$$



Fitting results under the 28nm process library

- Benefit
  - sufficient accuracy
  - independence from the upstream information (under certain conditions)

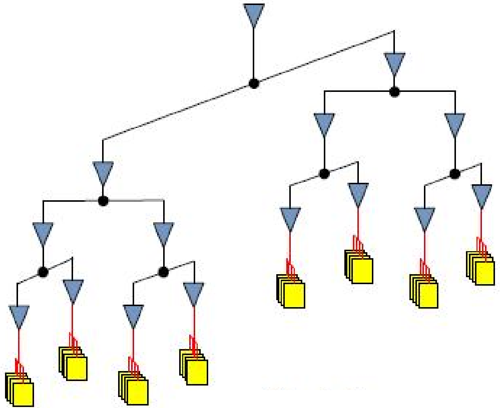
Topology Generation



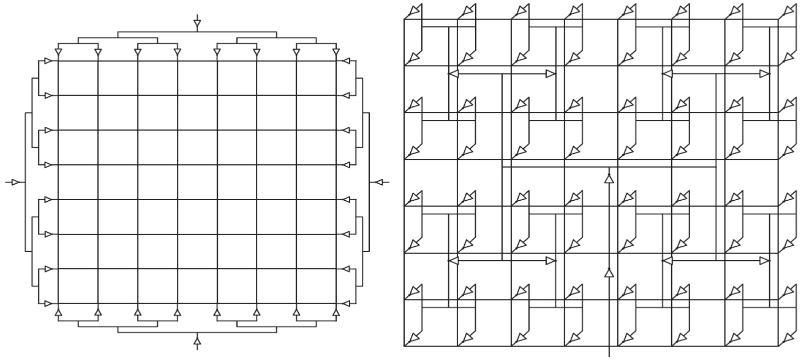
Topology Generation

Clock Topology

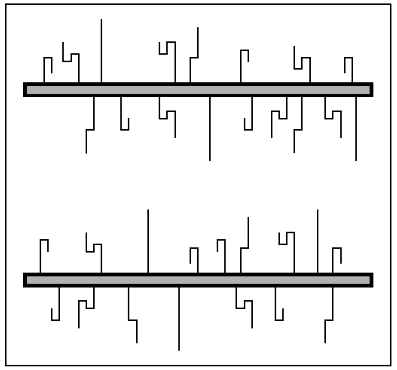
- **Common Topology**
  - Clock Tree
  - Clock Mesh (Grid)
  - Clock Spines
  - Hybrid



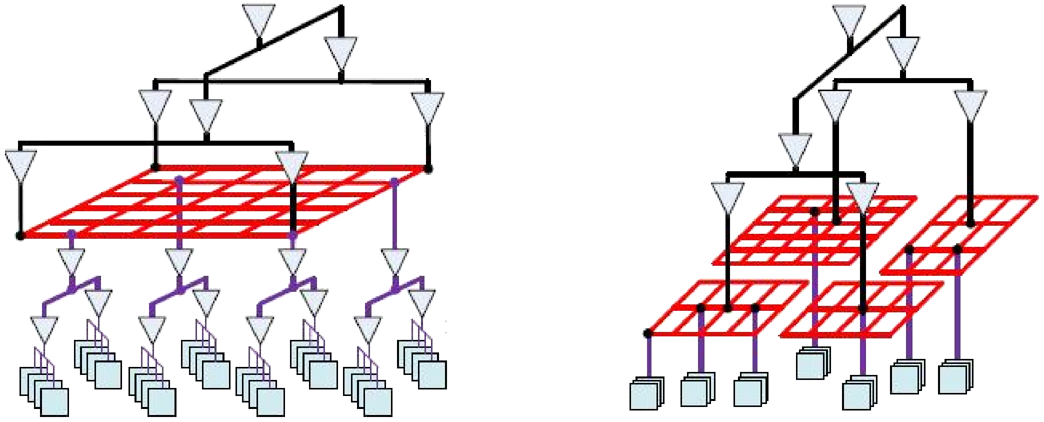
Clock Tree



Mesh Grid



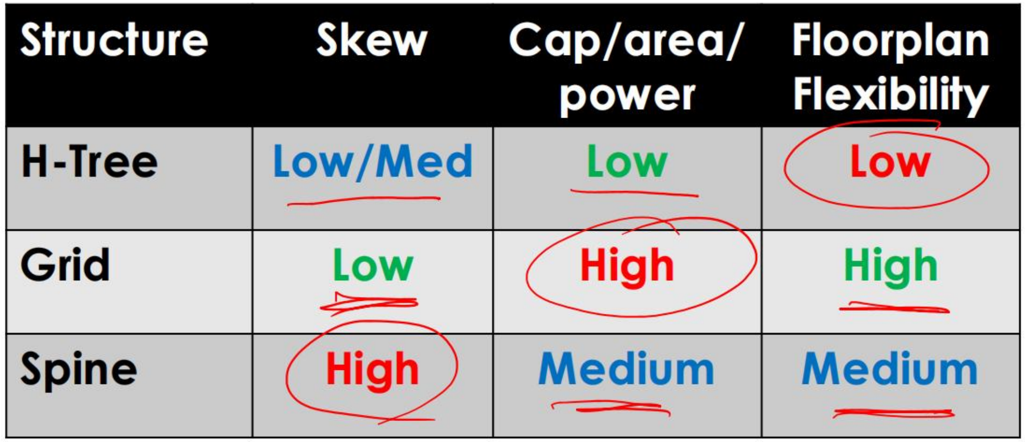
Fishbone/Spine



Hybrid Structure

Topology Comparison

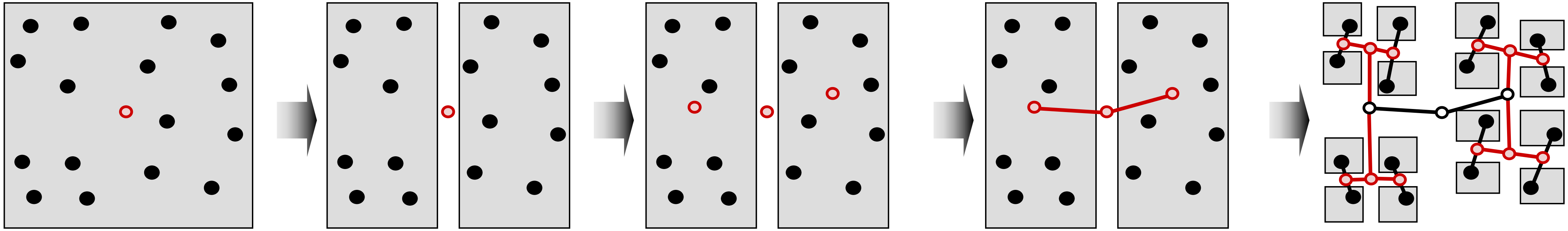
- Tree
  - Low skew, smallest routing capacitance, low power
  - Floorplan flexibility is poor.
- Grid or mesh
  - Low skew, increases routing capacitance, worse power
  - Alpha uses global clock grid and regional clock grids
- Spine
  - Small RC delay because of large spine width
  - Spine has to balance delays; difficult problem
  - Routing cap lower than grid but may be higher than H-tree



Structure	Skew	Cap/area/ power	Floorplan Flexibility
H-Tree	Low/Med	Low	Low
Grid	Low	High	High
Spine	High	Medium	Medium

Method of Means and Medians (MMM)

- Can deal with arbitrary locations of clock sinks
- Basic idea:
  - **Recursively partition the set of terminals into two subsets of equal size (median)**
  - **Connect the center of gravity (COG) of the set to the centers of gravity of the two subsets (the mean)**



Find the center of gravity

Partition *S* by the median

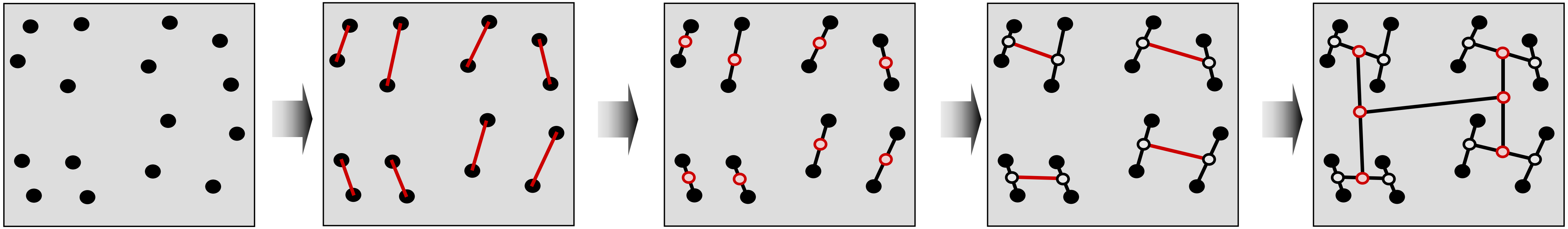
Find the center of gravity for the left and right subsets of *S*

Connect the center of gravity of *S* with the centers of gravity of the left and right subsets

Final result after recursively performing MMM on each subset

Recursive Geometric Matching (RGM)

- Recursively determine a minimum-cost geometric matching of *n* sinks
- Find a set of *n* / 2 line segments that match *n* endpoints and minimize total length (subject to the matching constraint)
- After each matching step, a balance or tapping point is found on each matching segment to preserve zero skew to the associated sinks
- The set of *n* / 2 tapping points then forms the input to the next matching step



Set of *n* sinks *S*

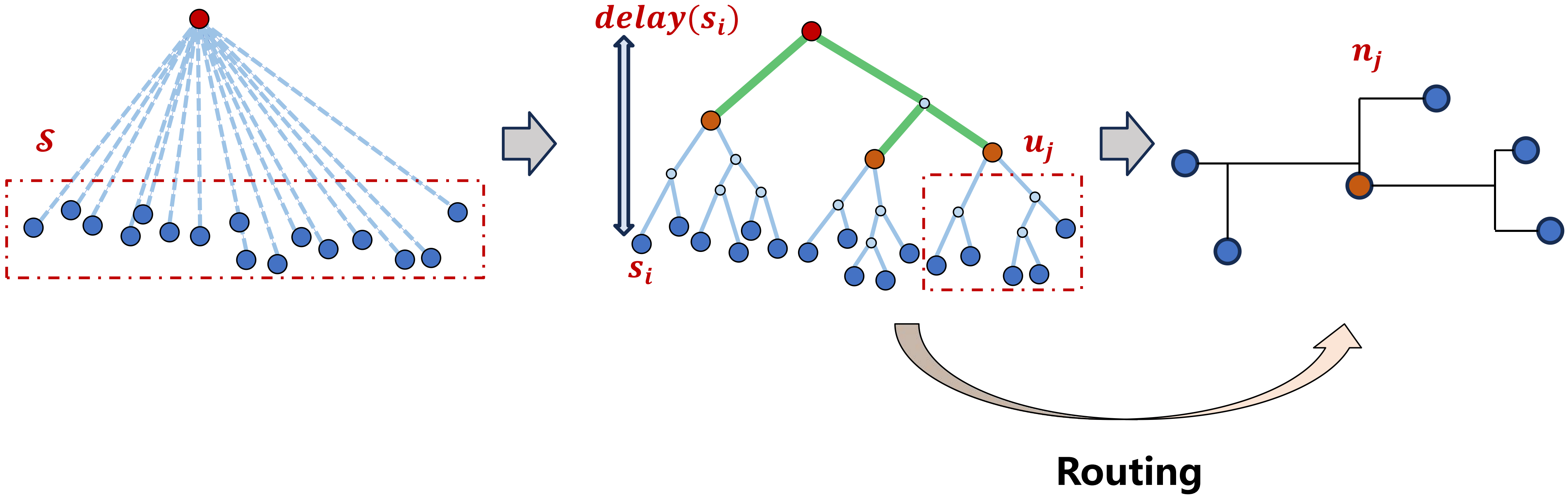
Min-cost geometric matching

Find balance or tapping points (point that achieves zero skew in the subtree, not always midpoint)

Min-cost geometric matching

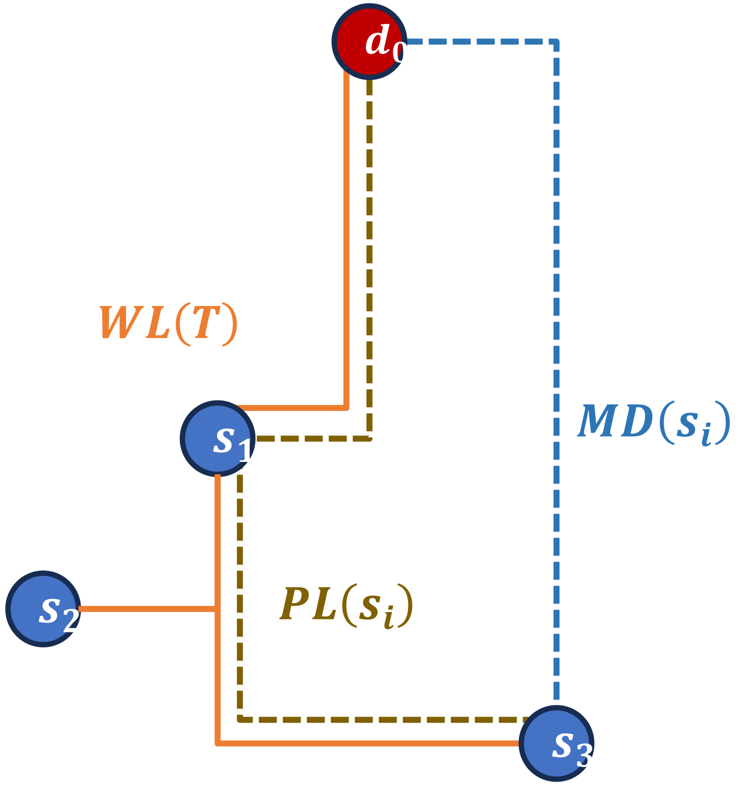
Final result after recursively performing RGM on each subset

Routing



Tree Balance of Latency-Load

- **目标：Skew(偏差)，Load Capacitance（负载），Latency (Delay)（时延)**
  - $PL(s_i)$, the path length from the source
  - $MD(s_i)$, the Manhattan distance from the source
  - $WL(T)$, the total wirelength of clock tree T
  - $WL(T_{FLUTE})$, the wirelength of *FLUTE tree*



$$\max_{s_i \in S}\left\{\frac{PL(s_i)}{MD(s_i)}\right\} \rightarrow 1$$

Latency √

$$\frac{WL(T)}{WL(T_{FLUTE})} \rightarrow 1$$

Capcitance √

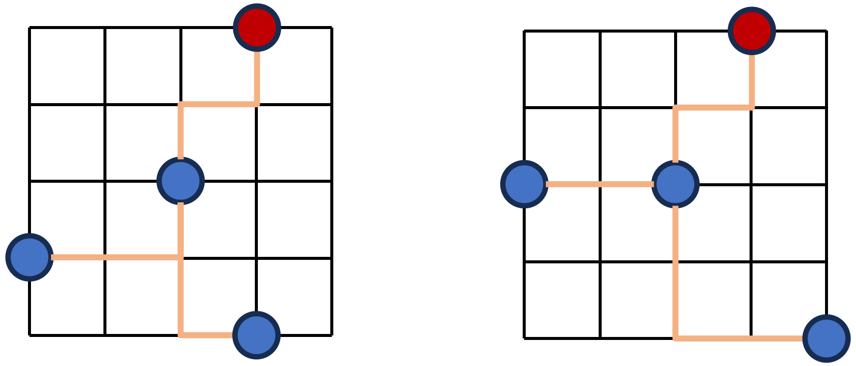
$$\frac{\max_{s_i \in S}\{PL(s_i)\}}{\overline{PL}} \rightarrow 1$$

Skew √

Skew-Latency-Load Tree (SLLT)

- ***Definition 1:*** $(\overline{\alpha},\overline{\beta}) - SLT$[1]
  - *For shallowness* $\alpha = \max_{s_i \in \mathcal{S}}\{\frac{PL(s_i)}{MD(s_i)}\}$ *and lightness* $\beta = \frac{WL(T)}{WL(T_{FLUTE})}$, *an* $(\overline{\alpha},\overline{\beta}) - SLT$, *where* $\overline{\alpha} \geq \alpha \geq 1$, *and* $\overline{\beta} \geq \beta \geq 1$

- ***Definition 2:*** ***Skewness*** γ
  - *For a Steiner tree, the skewness* $\gamma = \frac{\max_{s_i \in \mathcal{S}}\{PL(s_i)\}}{\overline{PL}}$

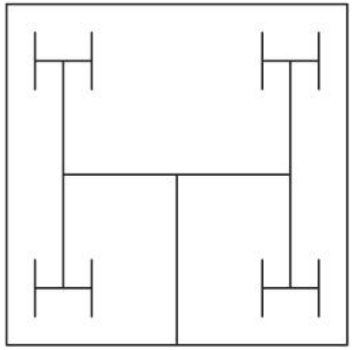


$\gamma = 6/6 = 1$ (Zero Skew Tree) **$\gamma = 7/6 \approx 1.17$ (Bound Skew Tree)**

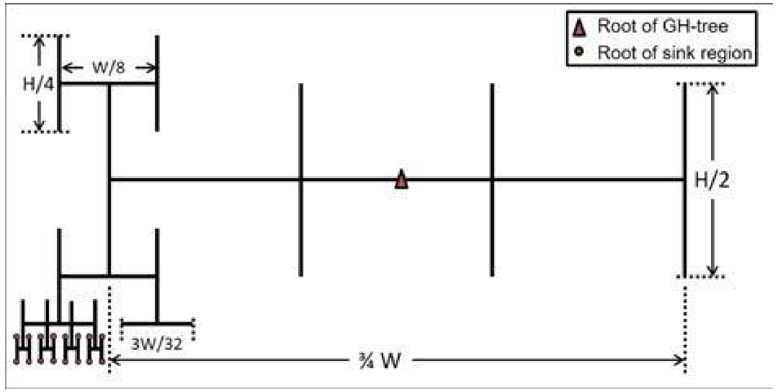
- ***Definition 3:*** $(\overline{\alpha},\overline{\beta},\overline{\gamma}) - SLLT$
  - *A rectilinear Steiner tree with shallowness* $\alpha \leq \overline{\alpha}$, *lightness* $\beta \leq \overline{\beta}$, *and skewness* $\gamma \leq \overline{\gamma}$, *is denoted as* $(\overline{\alpha},\overline{\beta},\overline{\gamma}) - SLLT$

[1] G. Chen. et. al. SALT, in ICCAD, 2018.

H-Tree & GH-Tree



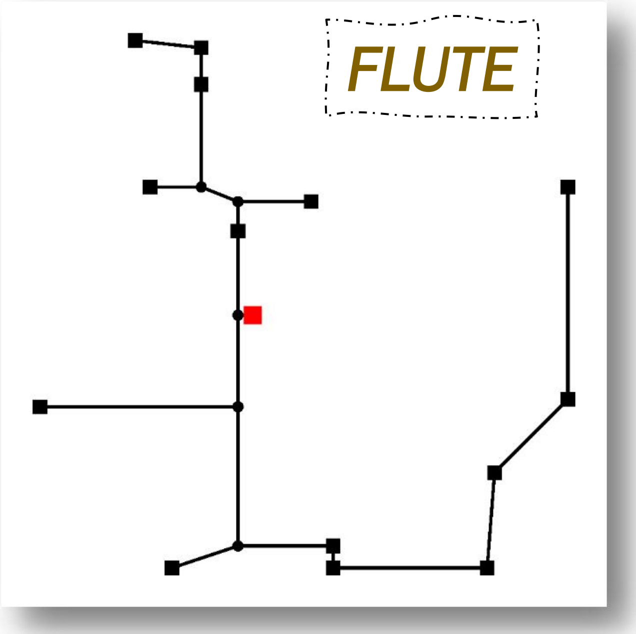
H - Tree



H-Tree

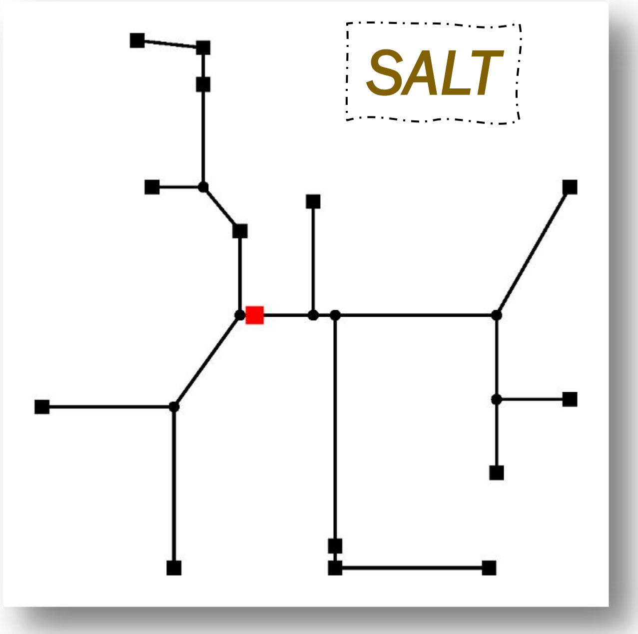
GH-Tree

已有的几种典型的 Tree



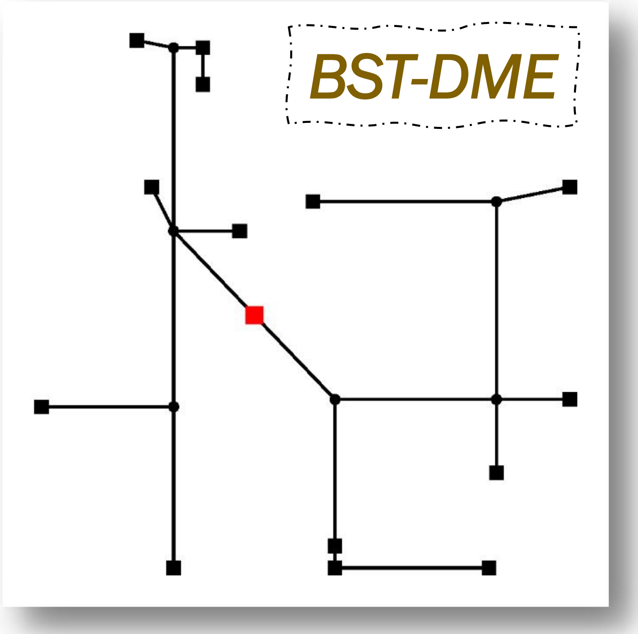
$min\{\sum WL\}$

Load Capacitance



$$\frac{PL_i}{MD_i} \leq 1$$

Latency (Delay)



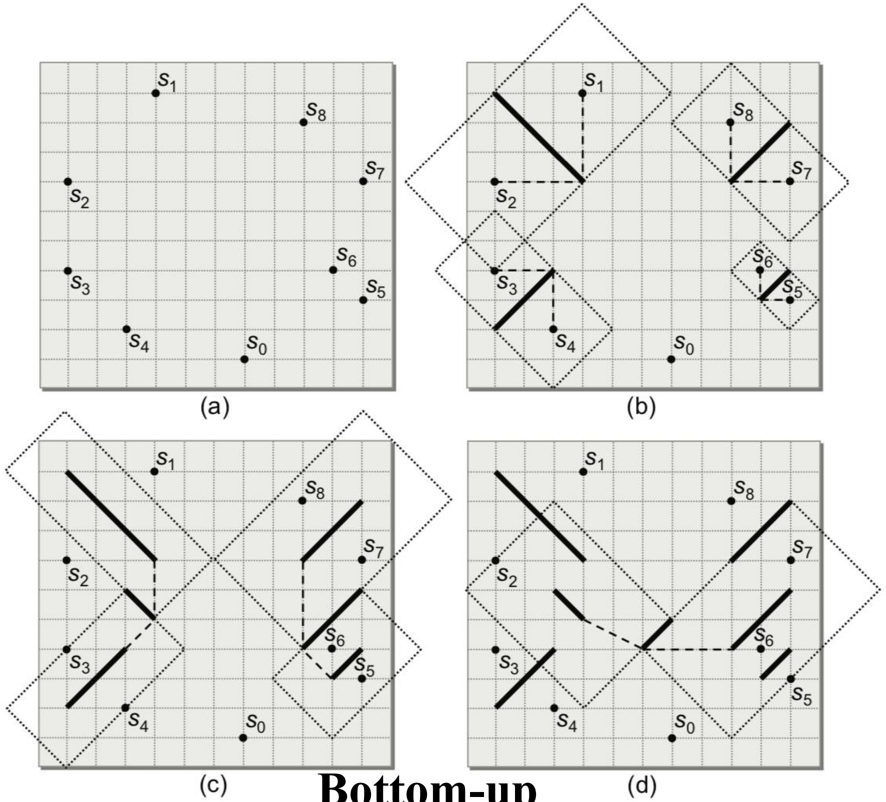
$skew \leq skew_{bound}$

?

Skew

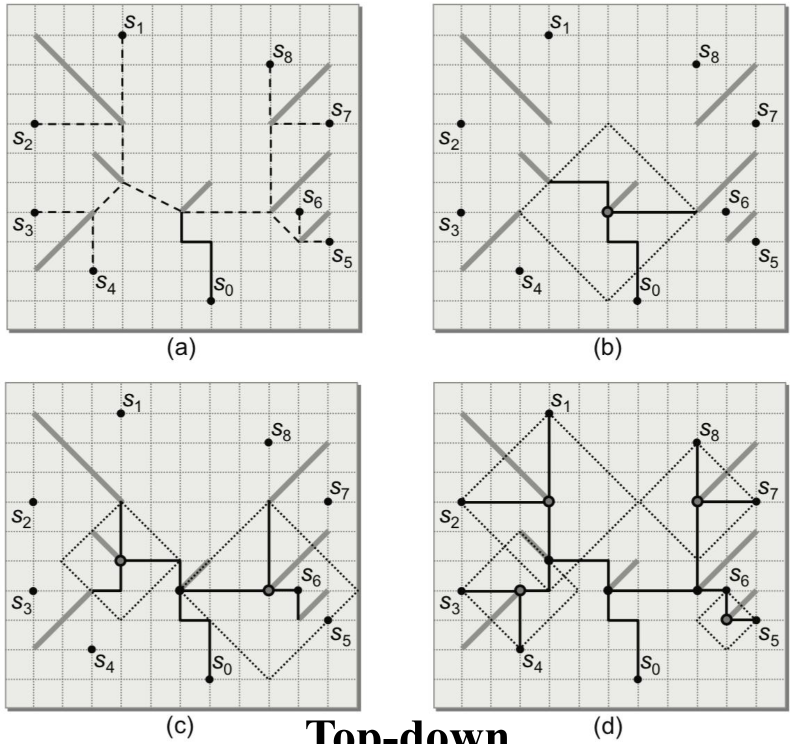
Zero Skew Tree (ZST/DME)

- **Bottom-up**
  - Determines all possible locations of internal nodes of G consistent with a minimum-cost ZST Tree of line segments, with each line segment being the locus of possible placements of an internal node of T
- **Top-down**
  - Chooses the exact locations of all internal nodes in T Output fully embedded, minimum-cost ZST with topology G



(a) (b) (c) (d)

Bottom-up

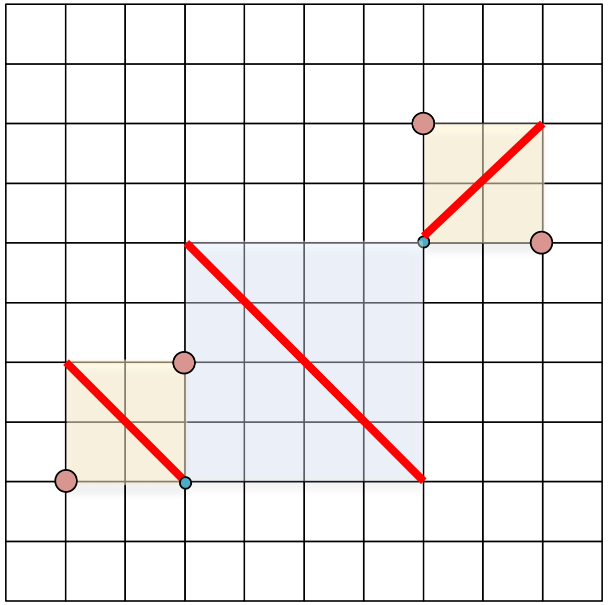


(a) (b) (c) (d)

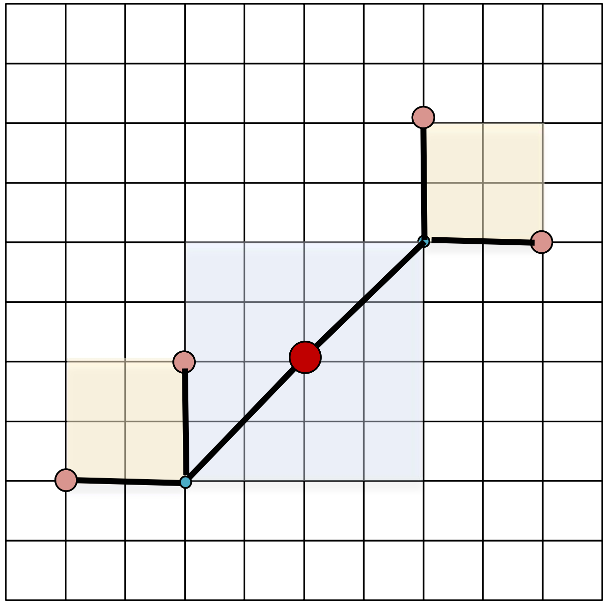
Top-down

Zero Skew Tree (ZST/DME)

- Bottom-up Stage
  - merge two sub-tree
  - determine merge region
- Top-down Stage
  - location embedding
  - nearest endpoint
- Benefit
  - zero/bound skew
  - topology based



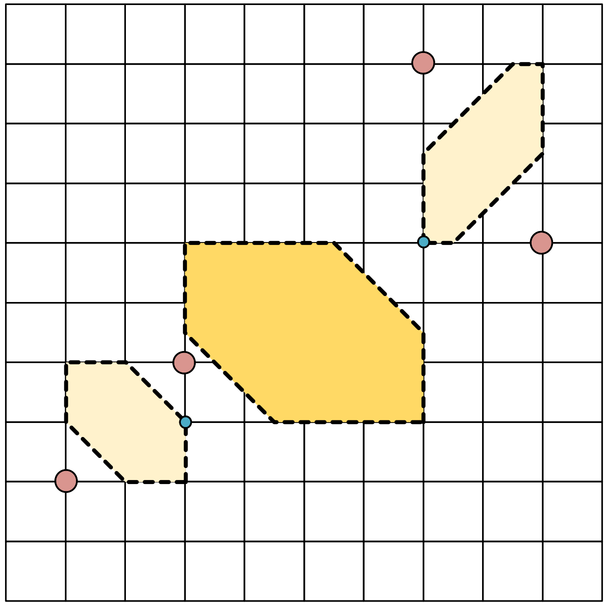
Bottom-up



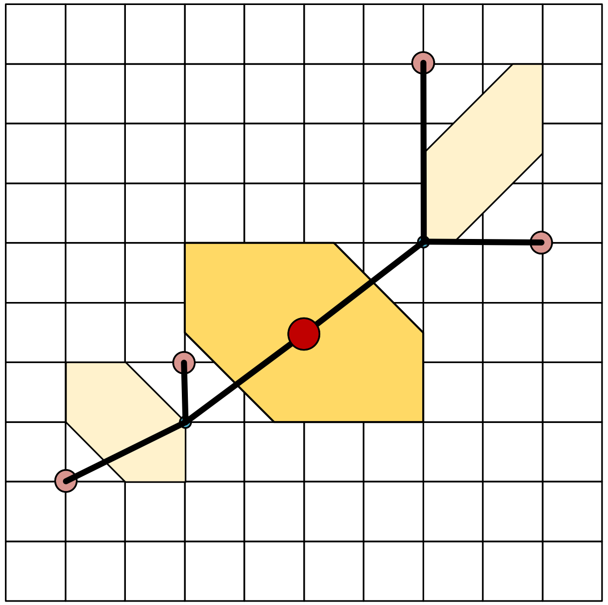
Top-down

Bound Skew Tree (BST/DME)

- Bottom-up Stage
  - merge two sub-tree
  - determine merge region
- Top-down Stage
  - location embedding
  - nearest endpoint
- Benefit
  - zero/bound skew
  - topology based



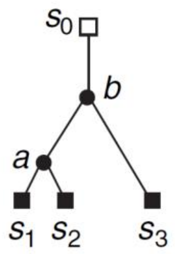
Bottom-up



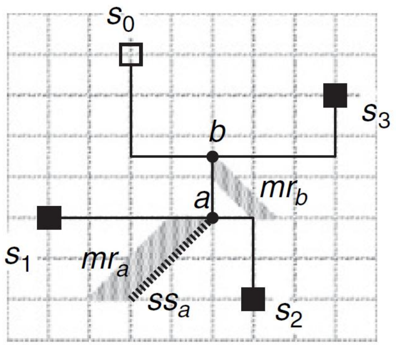
Top-down

Useful Skew Tree (UST/DME)

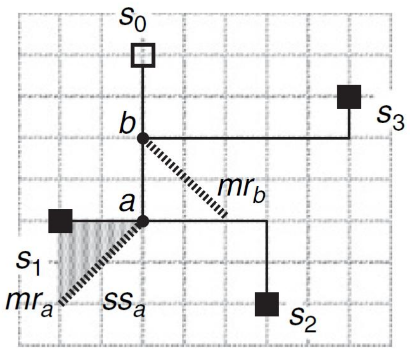
- Bottom-up Stage
  - merge two sub-tree
  - determine merge region
- Top-down Stage
  - location embedding
  - nearest endpoint
- Benefit
  - zero/bound skew
  - topology based



Topology



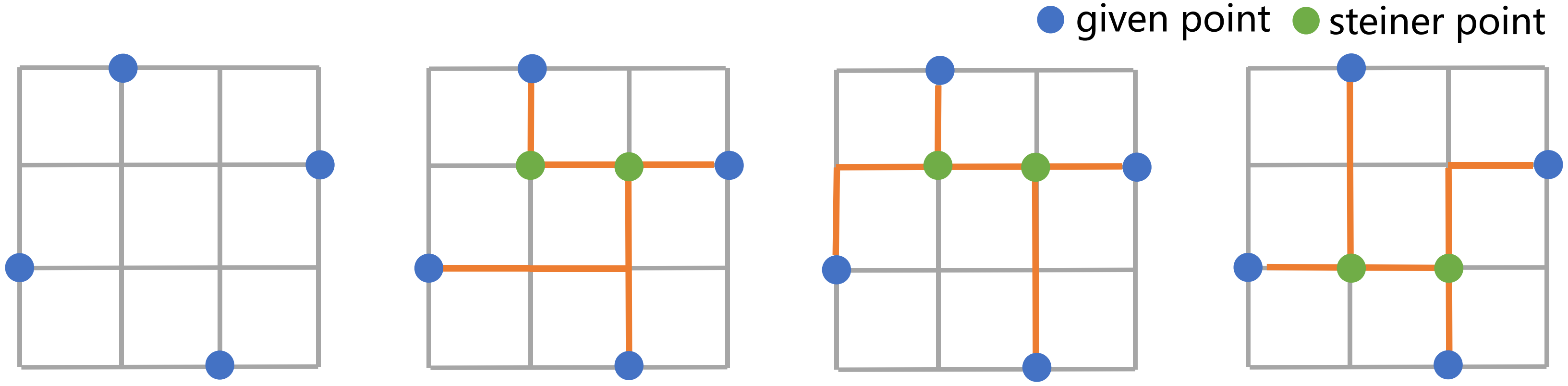
BST/DME tree $B = 1$



UST/DME tree

Rectilinear Minimum Steiner Tree (RMST)

- Rectilinear Minimum Steiner tree (RMST)
  - Given a set of points
  - Additional points (Steiner points) can be introduced
  - Limited to horizontal and vertical segments only



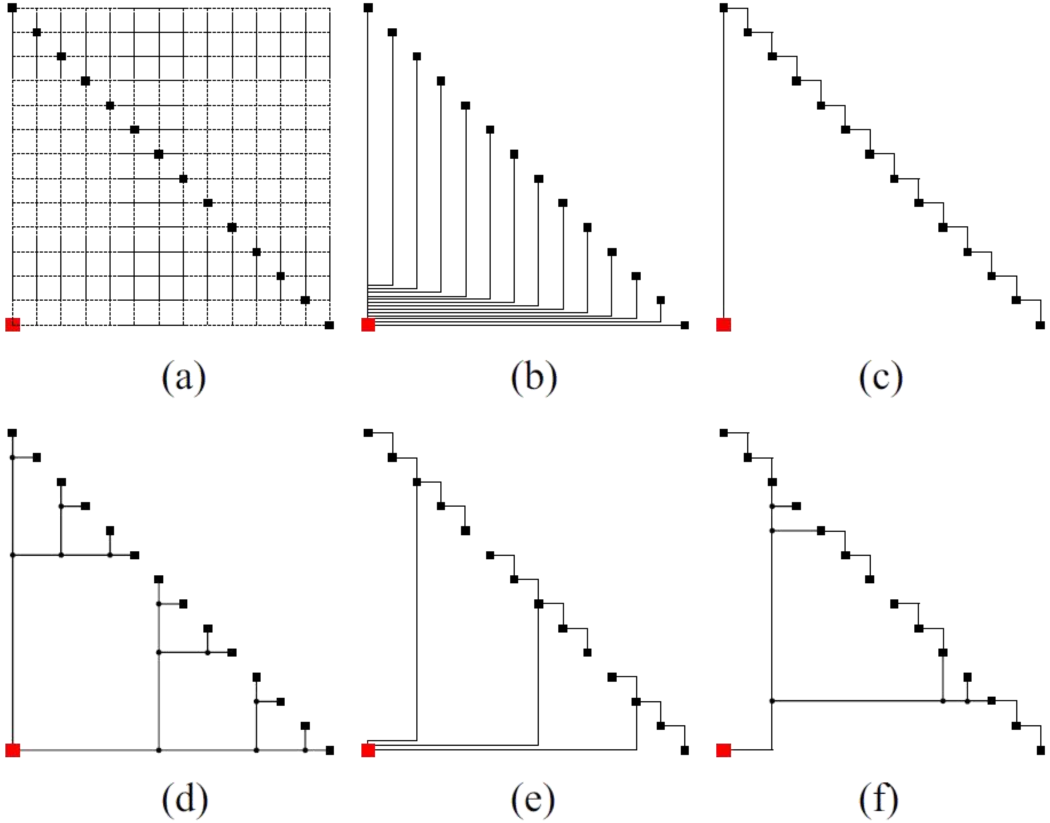
SALT

- Shallowness (path length)

$$\alpha = \max_{s_i \in \mathcal{S}} \left\{ \frac{PL(s_i)}{MD(s_i)} \right\},$$

- Lightness (tree weight)

$$\beta = \frac{WL(T)}{WL(MST(G))} \approx \frac{WL(T)}{WL(T_{FLUTE})}.$$

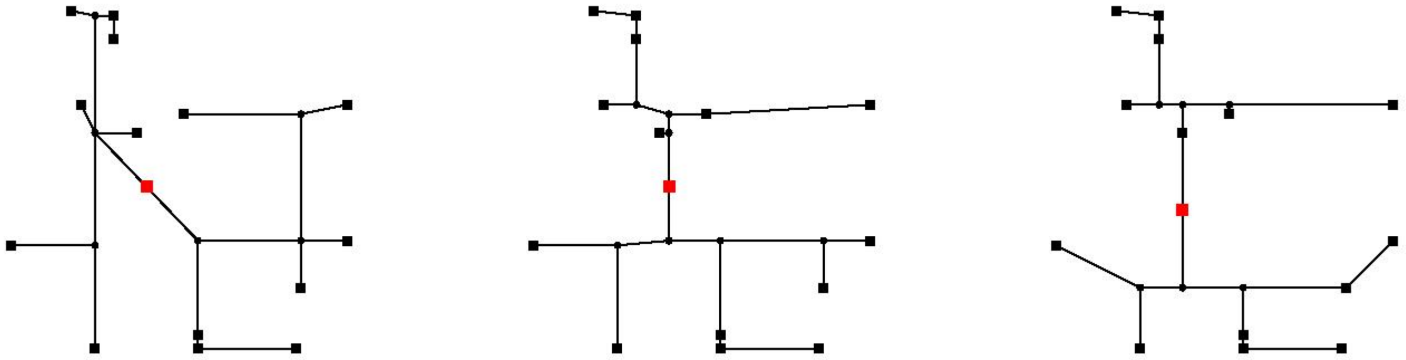


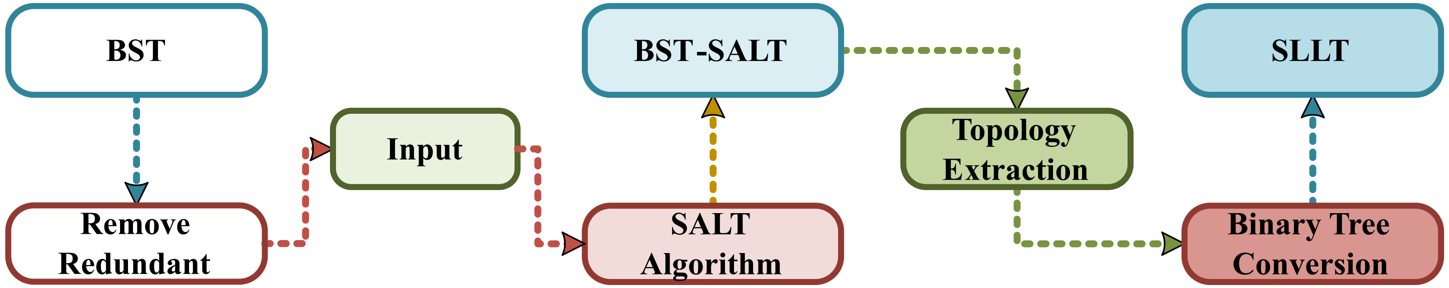
Different Shallow-Light Tree on the same net

Given ϵ, SALT realizes $\alpha \leq 1+\epsilon$ and $\beta \leq 2 + \left\lceil log \frac{2}{\epsilon} \right\rceil$

Concurrent BST and SALT (CBS)

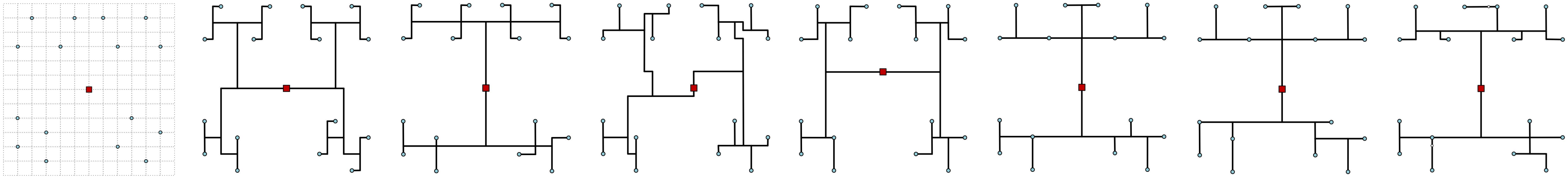
- **Bound Skew Tree (BST)**
  - **Benefit**
    - Provides bounded skew control
  - **Weakness**
    - High cost of net (wirelength, cap)
- **Steiner SLT (SALT)**
  - **Benefit**
    - Acceptable total wirelength and smaller path length (delay)
  - **Weakness**
    - Lacks the ability to balance skew





Comparison of different SLLTs

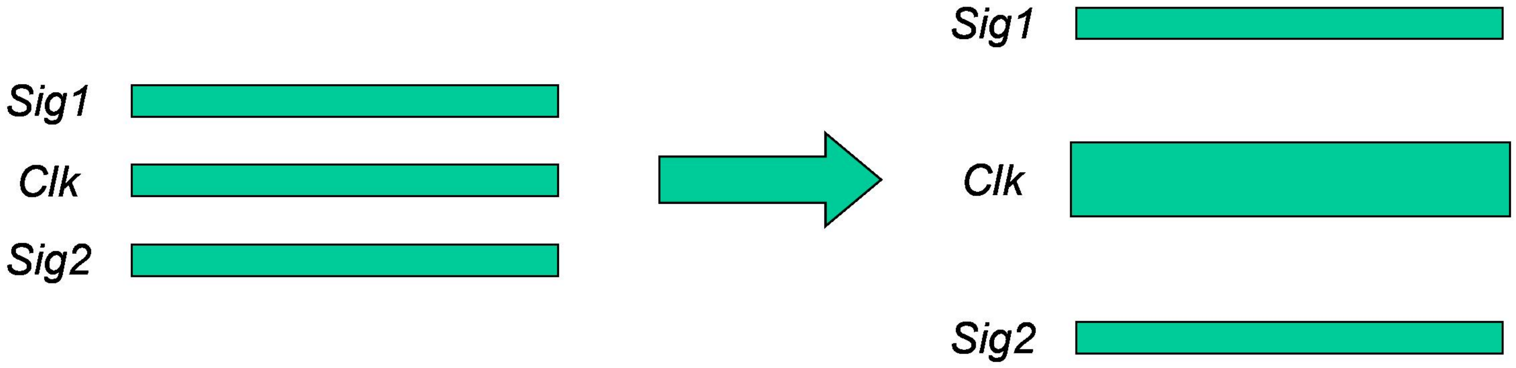
- **A rectilinear Steiner tree with $\alpha \le \bar{\alpha}$, $\beta \le \bar{\beta}$, and $\gamma \le \bar{\gamma}$, is denoted as $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}) - \mathrm{SLLT}$.**



Algorithm	Max PL	Min PL	Total WL	Mean PL	α	β	γ	Mean	Skew Control
H-tree	10	9	55.5	9.75	2	1.32	1.03	1.45	√
GH-tree	10	7	47.5	8.5	1.6	1.13	1.18	1.3	√
ZST	10.5	10.5	55.5	10.5	2.33	1.32	1.00	1.55	√
BST	10	8	50	9.25	2.25	1.19	1.08	1.51	√
FLUTE	9	5	42	7.44	1.4	1.00	1.21	1.2	×
R-SALT	9	5	43	7.06	1.00	1.02	1.27	1.10	×
CBS	9	7	45	8.13	1.4	1.07	1.11	1.19	√

Non-Default Clock Routing

- PnR Tool can route the clocks using non-default routing rules, e.g. double-spacing, double-width, shielding, and double via
- Non-default rules are often used to “harden” the clock, e.g. to make the clock routes less sensitive to Cross Talk or EM effects, which improve yield



Default Routing Rule *Effect of NDR route on Clk*

NDR Recommendations

- Always route clock on metal 3 and above
- Avoid NDR on Metal 1
  - may have trouble accessing metal 1 pins on buffers and gates
- Consider using double spacing to reduce crosstalk
- Consider double width to reduce resistance
- Consider double via to reduce resistance and improve yield

Put NDR on Pitch for Accurate RC Estimation

- Metal traces are always routed “on pitch”
- With clock NDR rules, pre-routing RC estimates of clock nets use NDR width and spacing numbers
- If NDR [spacing + width] numbers are not integer multiples of pitch (i.e. off-pitch), timing estimates pre-route may not correlate well with post-route timing
- Make sure your NDR numbers are on pitch!

01 Introduction

02 Clock Concepts and Techniques

03 Clock Tree Synthesis

04 Clock Tree Optimization

Understand Your Clock Tree Goals

- Skew Goal
  - What are the skew requirements for your design?
  - Are there different skew targets for small and large clocks?
- Insertion Delay Goal
  - What are the insertion delay specs for your block?
  - What is a reasonable target based on the size and floorplan of your block/chip?

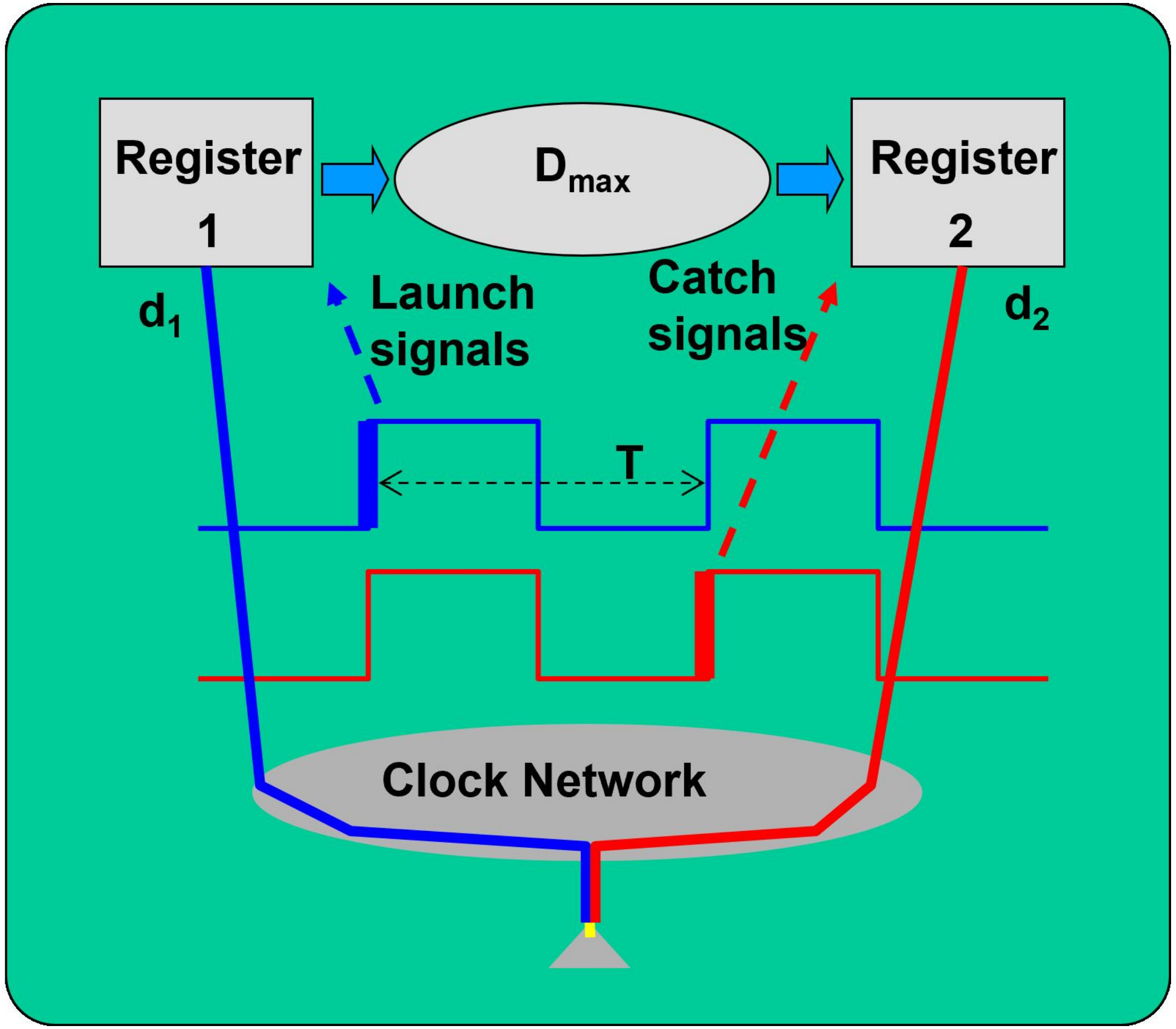
- Skew
  - 计算从PLL到各个FF的最大的与最小Latency的差值;
  - Zero Skew, Bound Skew, Useful Skew
- Latency
  - 计算信号从PLL到FF的传输时间，电阻电容
- Power
  - 在时钟线网所花费的功耗，信号切换，动态功耗极大，占总功耗40%
- Transition Time
  - 充电时间，与电压电容相关
- Area
  - Buffer布局
- IR drop
  - 电压降稳定性，电阻，逻辑翻转
- 电迁移
  - 电子从金属线迁移到半导体，致电路短路和断路

General Concepts: Clock Distribution Network

Skew = $d_1 - d_2$

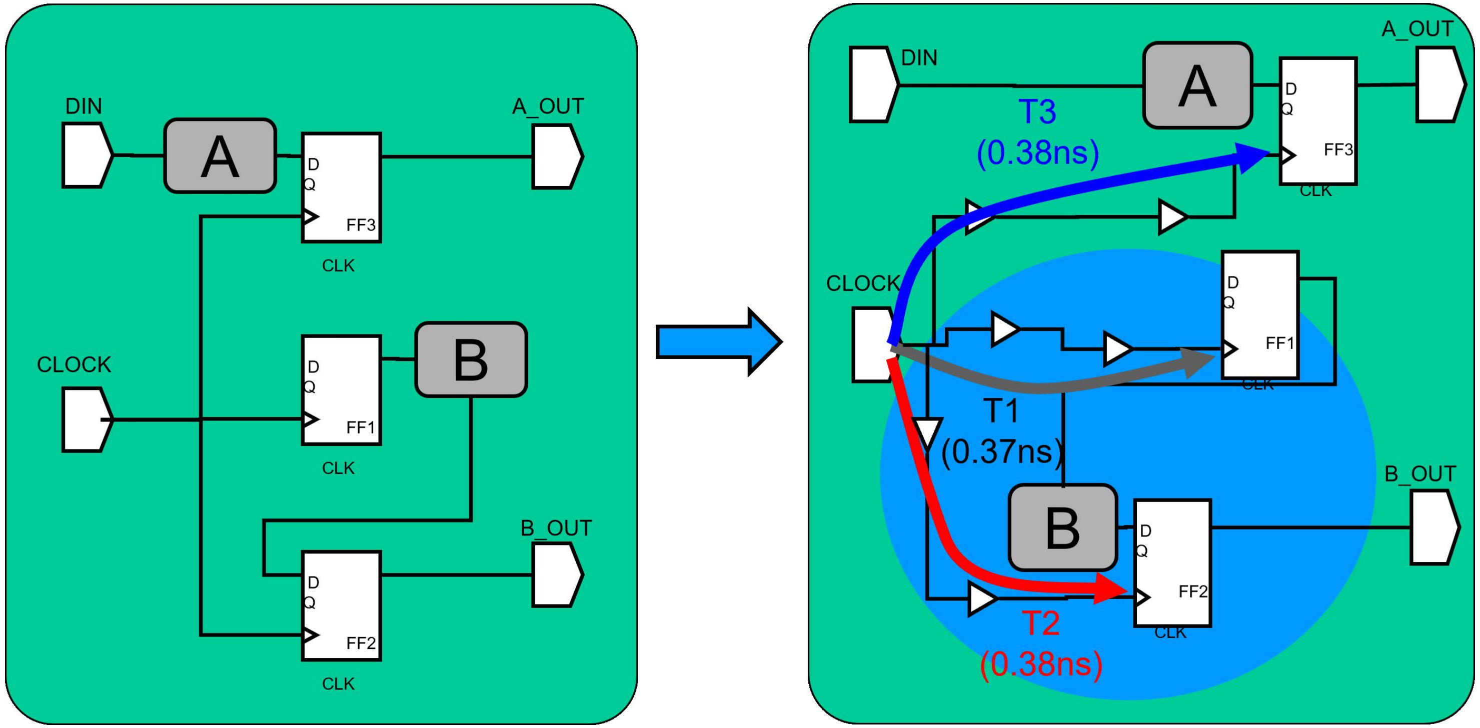
Zero skew: $d_1 = d_2$

Useful skew, $d_1 - d_2 = \delta_{12}$



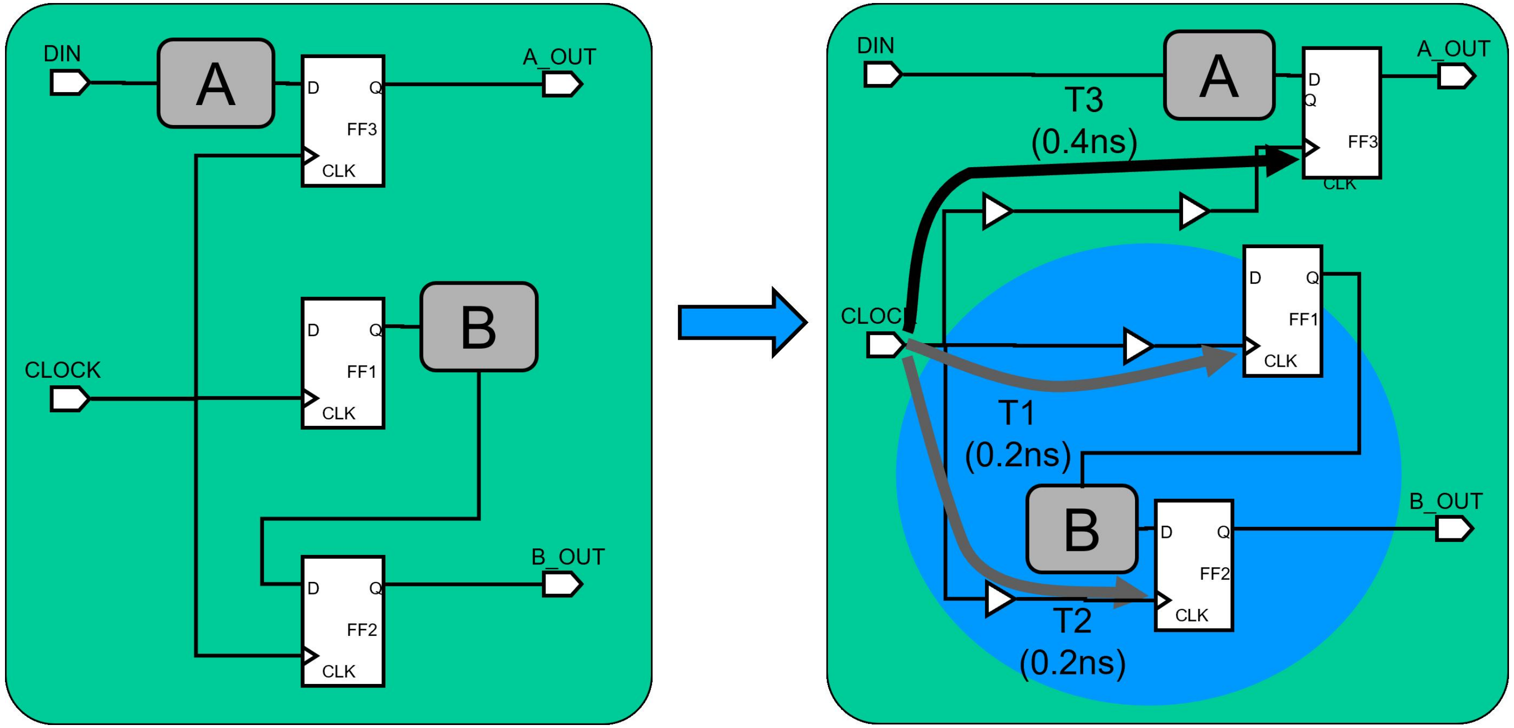
Clock Skew: Global Skew

- Global
  - Global skew is recommended - fastest, may add unnecessary buffers
- Local
  - Longer runtime, Possibly fewer buffers ” Only related FFs are balanced for skew”

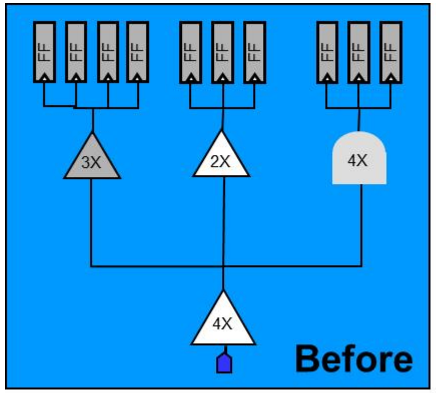


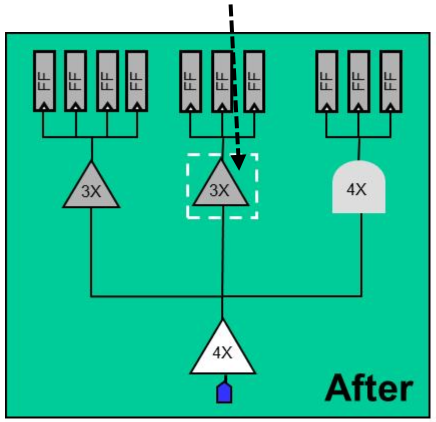
Clock Skew: Local Skew

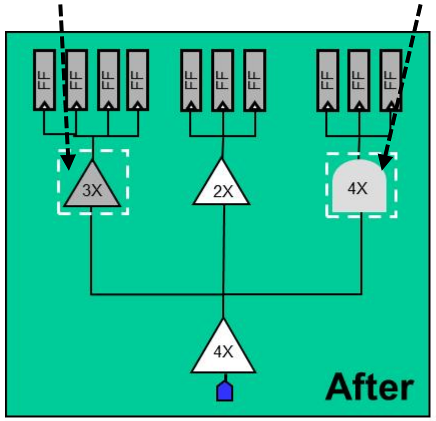
- Global
  - Global skew is recommended - fastest, may add unnecessary buffers
- Local
  - Longer runtime, Possibly fewer buffers ”   Only related FFs are balanced for skew”



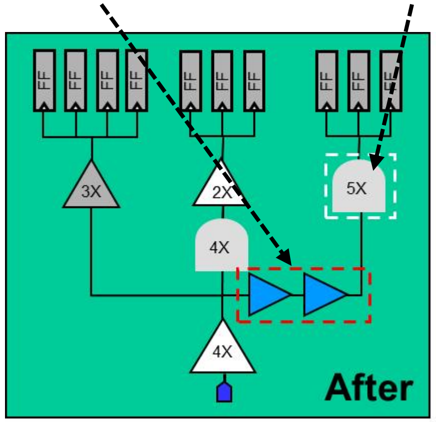
Clock Tree Optimizations




Buffer sizing


Buffer relocation

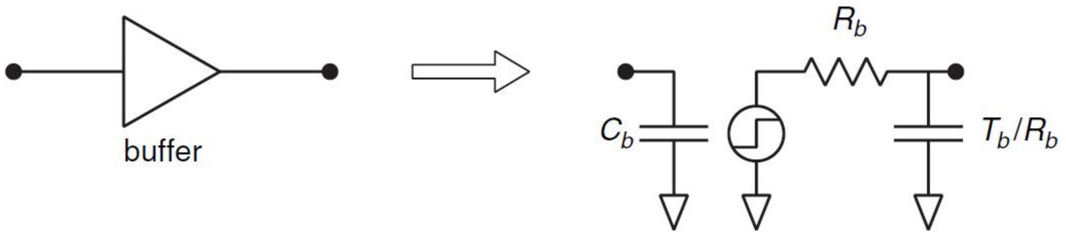
Gate relocation


Delay Insertion

Gate sizing

Buffer Insertion

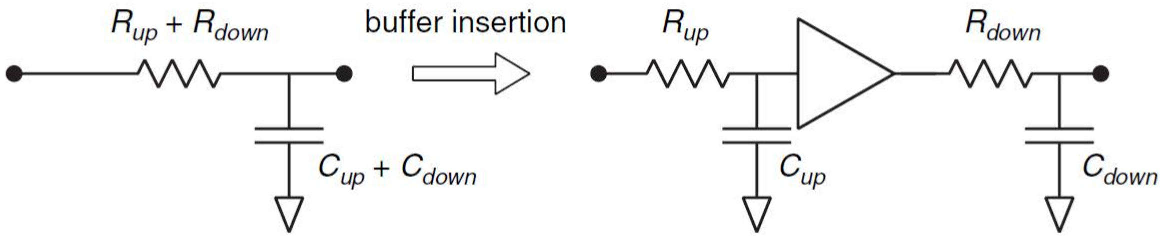
- 插入buffer，把大的时钟树变成各个层次的小时钟树；
- Global, Regional, Local
- 减少负载，增加驱动力，减少delay



Buffer model

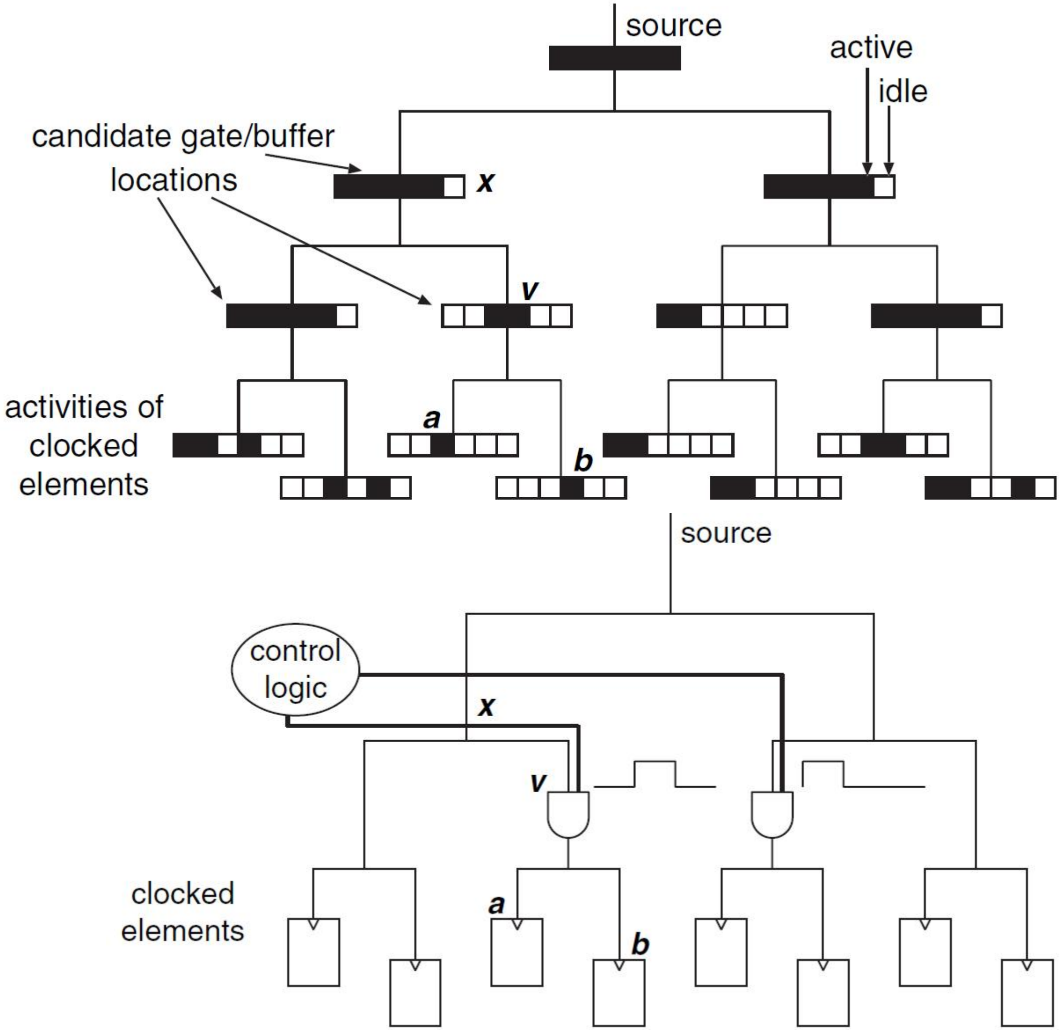
$$R_{up}C_b + T_b + R_bC_{down} < R_{up}C_{down}$$

$$R_{up}(C_{up} + C_b) + T_b + (R_b + R_{down})C_{down}$$



Clock Gating

- Clock Gating
  - 控制激活和闲置时钟
  - 减少时钟负载，减少功耗

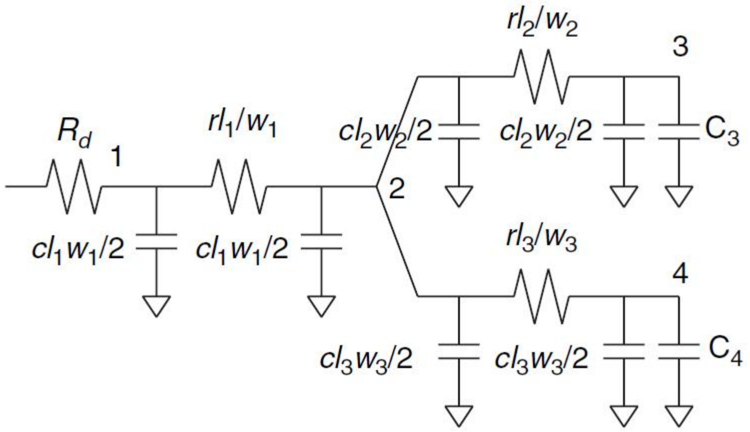


Wire Sizing

- 为线段分配合适的宽度，
- 减少Skew和delay，增加稳定性，减

$$t_i = R_d \cdot C_{T_{s_0}} + \sum_{e_v \in Path(s_0,s_i)} |e_v| \cdot \frac{r}{w_{e_v}} \cdot \left(\frac{|e_v| \cdot w_{e_v} \cdot c_a}{2} + C_{T_v}\right)$$

$$\frac{\partial t_i}{\partial w_{e_v}} = R_d \cdot c_a \cdot |e_v| + \sum_{e_u \in Ans(e_v) \cap path(s_0,s_i)} \frac{|e_u| \cdot r \cdot c_a \cdot |e_v|}{w_{e_u}}$$

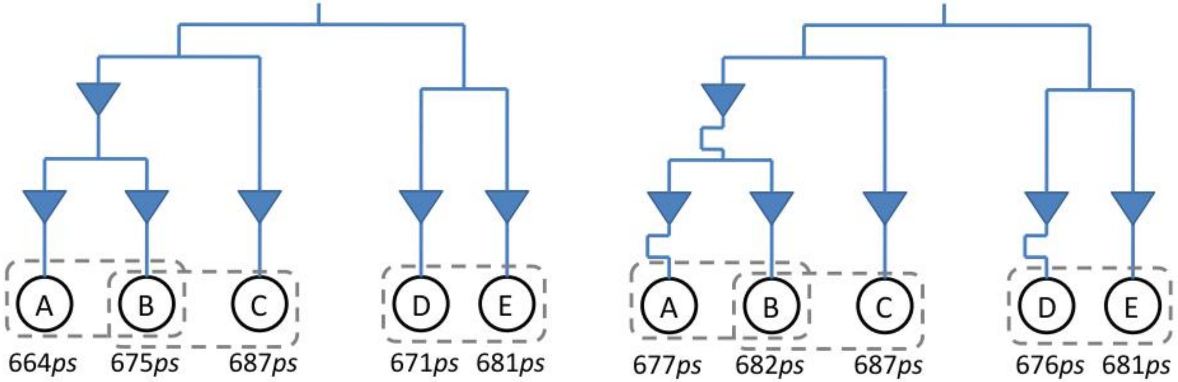


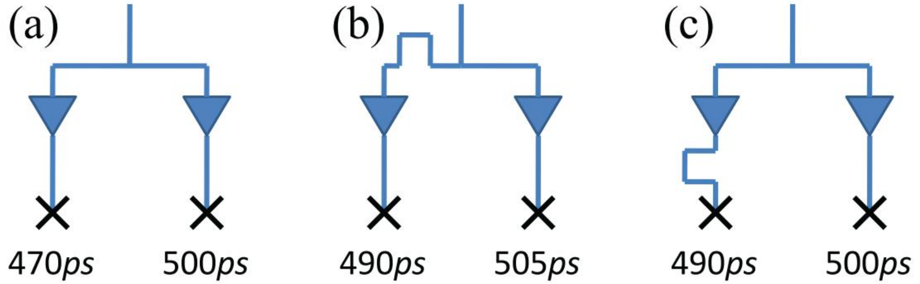
$$\frac{\partial t_3}{\partial w_2} = R_d \cdot c \cdot l_2 - \frac{l_2 \cdot r \cdot C_3}{w_2^2} + \frac{l_1 \cdot r \cdot c \cdot l_2}{w_1}$$

$$\frac{\partial t_4}{\partial w_2} = R_d \cdot c \cdot l_2 + \frac{l_1 \cdot r \cdot c \cdot l_2}{w_1}$$

Wire Snaking

- **局部增加线长，减少skew**

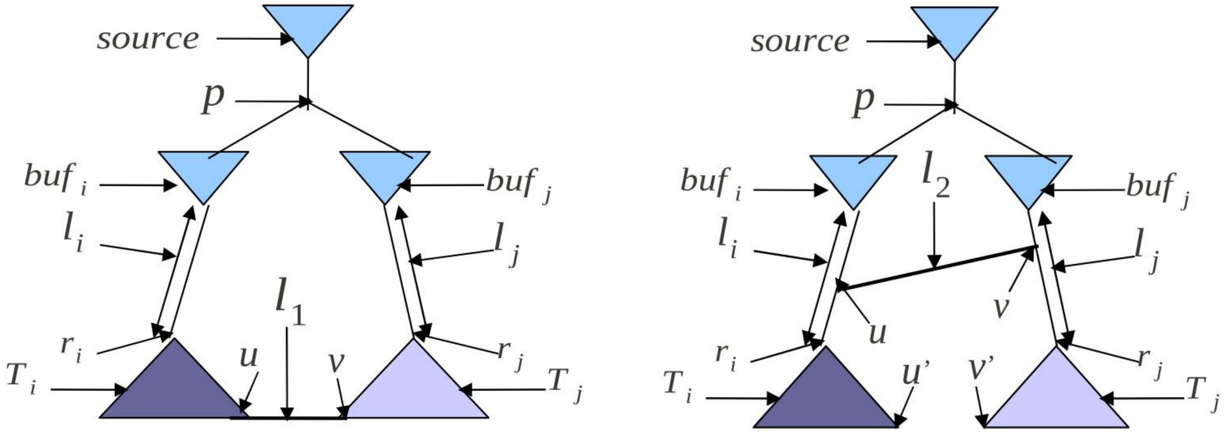




$$\alpha \leq \frac{T^i_{actual}(e)}{T^i_{target}(e)} \leq 1.0$$

Cross Linking

- 在FF之间连接边，或者在中间级buffer拆入边
- 减少skew，减少latence
  - **优点:** Skew小，层次少，功耗小，层次多，
  - **缺点：** 设计难度大



$l_2 << l_1$ satisfies $\alpha_2<\alpha_1$ & $\beta_2<\beta_1$

$\overline{q}_{u,v}= \alpha q_{u,v} +\alpha\beta$

where

$\overline{q}_{u,v}$=skew after link addition

$q_{u,v}$=skew before link addition

$\alpha=R_l/R_{loop}$

$\boldsymbol{\beta=Cl/2(R_{u,u}-R_{v,v})}$

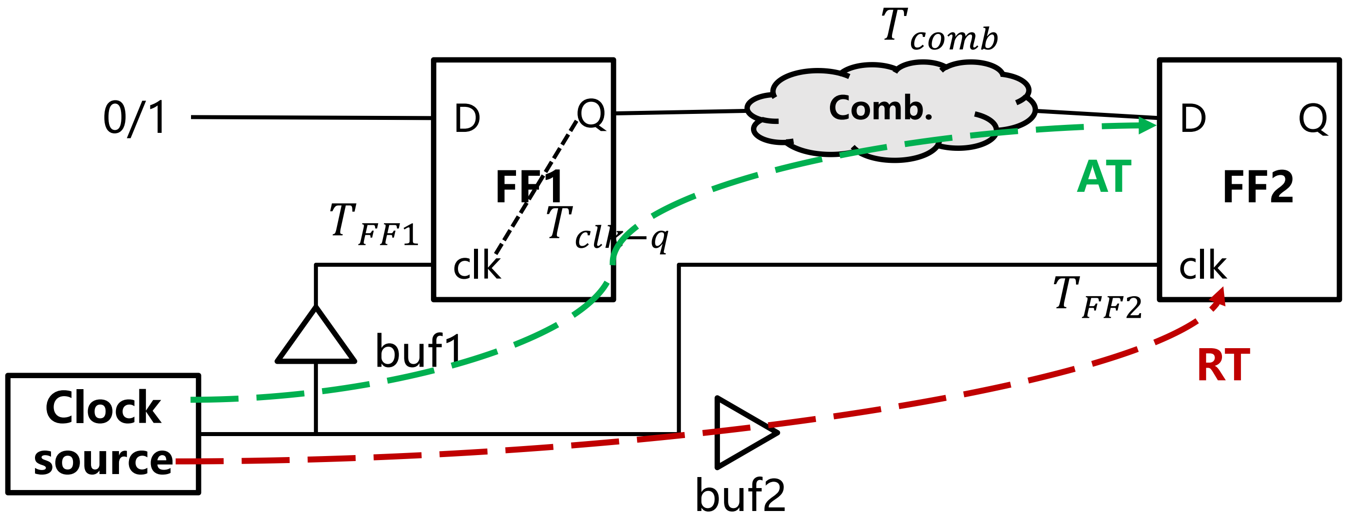
- **01** Introduction
- **02** Clock Tree Synthesis
- **03** Clock Tree Optimization

Useful Skew

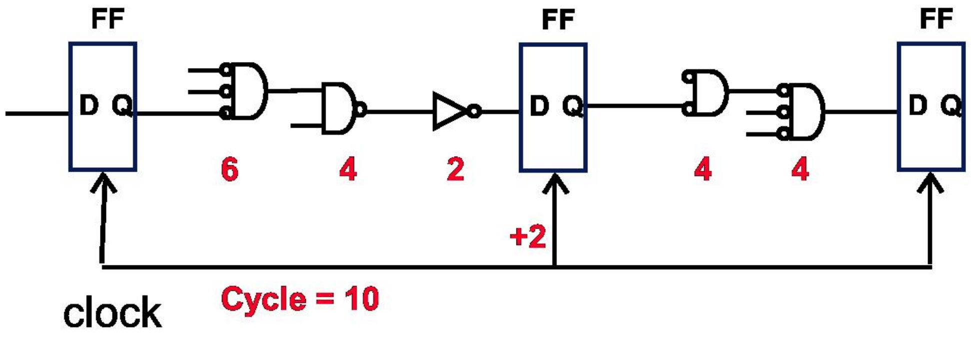
- Let’ s remember our famous timing constraints:
  - $\Delta_{skew} = T_{FF1} - T_{FF2}$

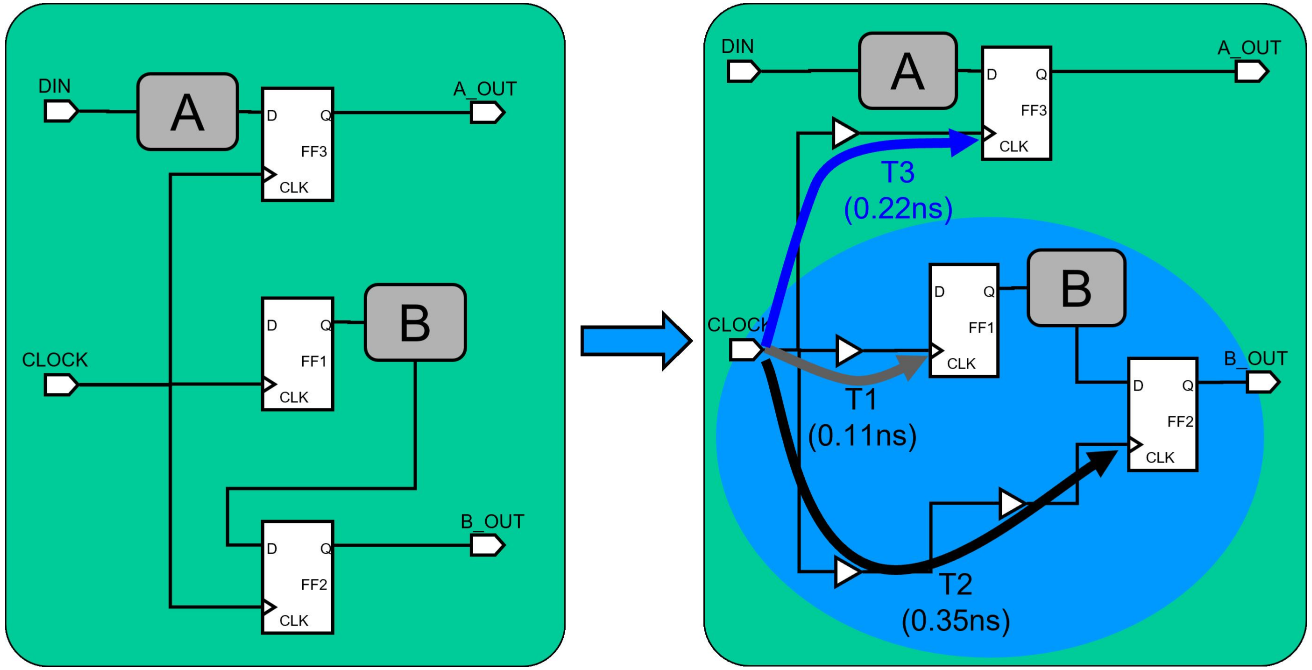
$$T_{FF1} + T_{clk-q} + T_{comb} + T_{setup} - T_{FF2} - T = T_{slack}^{late} \geq 0$$ **Setup Constraint**

$$T_{FF1} + T_{clk-q} + T_{comb} - T_{hold} - T_{FF2} = T_{slack}^{early} \geq 0$$ **Hold Constraint**



Useful Skew


FF
D Q
FF
D Q
FF
D Q
6
4
2
4
4
+2
clock
Cycle = 10


DIN
A
D Q
FF3
CLK
A_OUT
CLOCK
D Q
FF1
CLK
B
D Q
FF2
CLK
B_OUT
DIN
A
D Q
FF3
CLK
A_OUT
T3
(0.22ns)
CLOCK
D Q
FF1
CLK
B
T1
(0.11ns)
B_OUT
D Q
FF2
CLK
T2
(0.35ns)

Clock Schedule

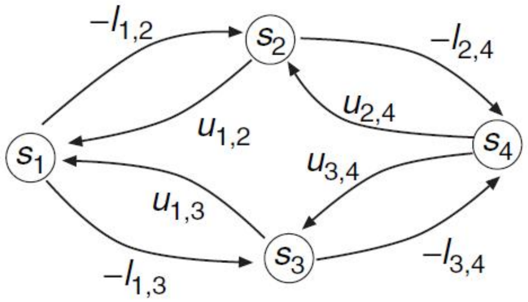
$$skew_{i,j} \leq C_p - t_{pFF}^{\max} - t_{logic}^{\max} - t_{setup}^{\max} - (T_i^{jitter} + T_j^{jitter}) - \delta_u$$

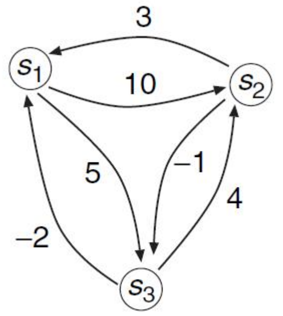
$$skew_{i,j} \geq t_{hold}^{\max} - t_{pFF}^{\min} - t_{logic}^{\min} + (T_i^{jitter} + T_j^{jitter}) + \delta_l$$

$$l_{i,j} \leq skew_{i,j} = t_i - t_j \leq u_{i,j}$$

$$skew_{1,2} + skew_{2,4} = (t_1 - t_2) + (t_2 - t_4) = t_1 - t_4 \leq u_{1,2} + u_{2,4}$$
$$skew_{1,3} + skew_{3,4} = (t_1 - t_3) + (t_3 - t_4) = t_1 - t_4 \leq u_{1,3} + u_{3,4}$$

$$skew_{1,4} = t_1 - t_4 \leq \min\{u_{1,2} + u_{2,4}, u_{1,3} + u_{3,4}\}$$





Constraint graph G_C

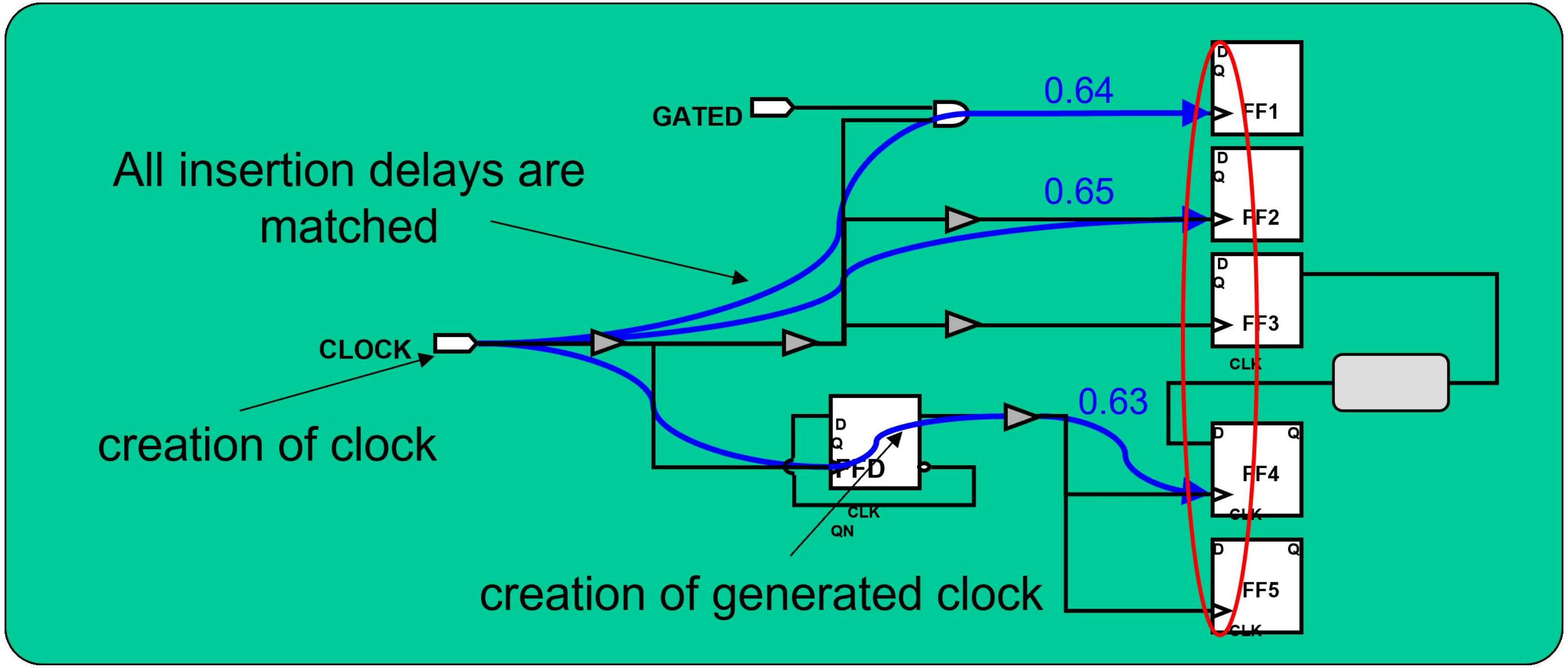
$$\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ s_1 & 0 & 9 & 5 \\ s_2 & -3 & 0 & -1 \\ s_3 & -2 & 4 & 0 \end{array}$$

All-pairs shortest distance matrix D

$$\begin{array}{c|ccc} & s_1 & s_2 & s_3 \\ s_1 & 0 & 3 & 2 \\ s_2 & -3 & 0 & -1 \\ s_3 & -2 & 1 & 0 \end{array}$$

Updated matrix D after committing $skew_{1,2} = -3$

Generated and Gated Clocks



All insertion delays are matched

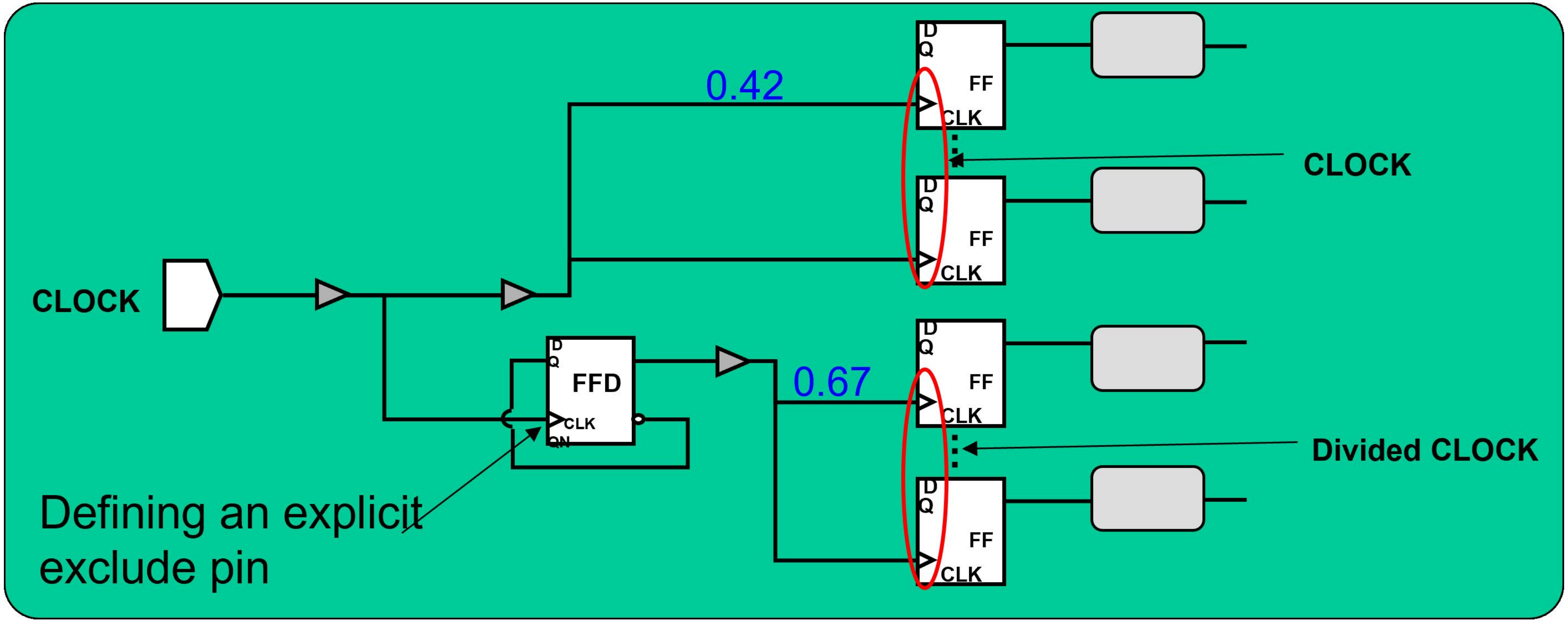
creation of clock

creation of generated clock

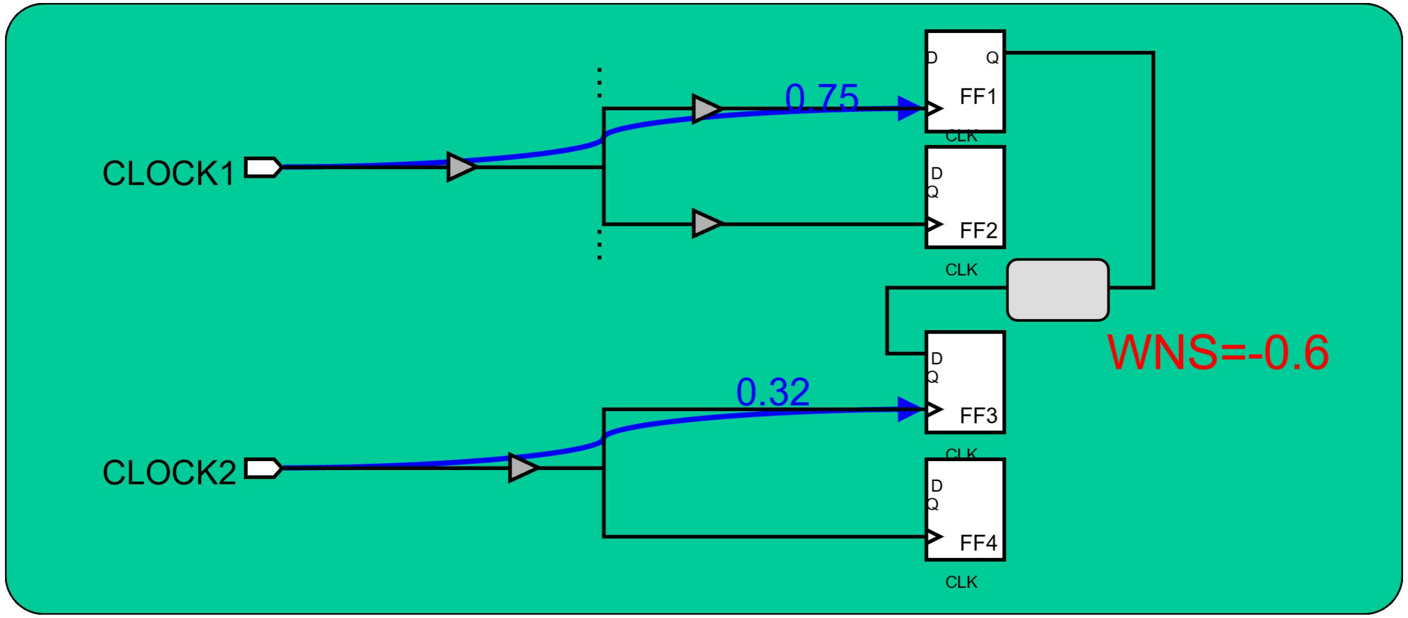
Skew will be balanced globally within each clock domain across all clock-pins for both master and generated clock.

Skew Balancing not Required

If the divided clock domain is independent of the master domain (no paths), then skew balancing may not be important.



No Inter-Clock Skew Balancing by Default

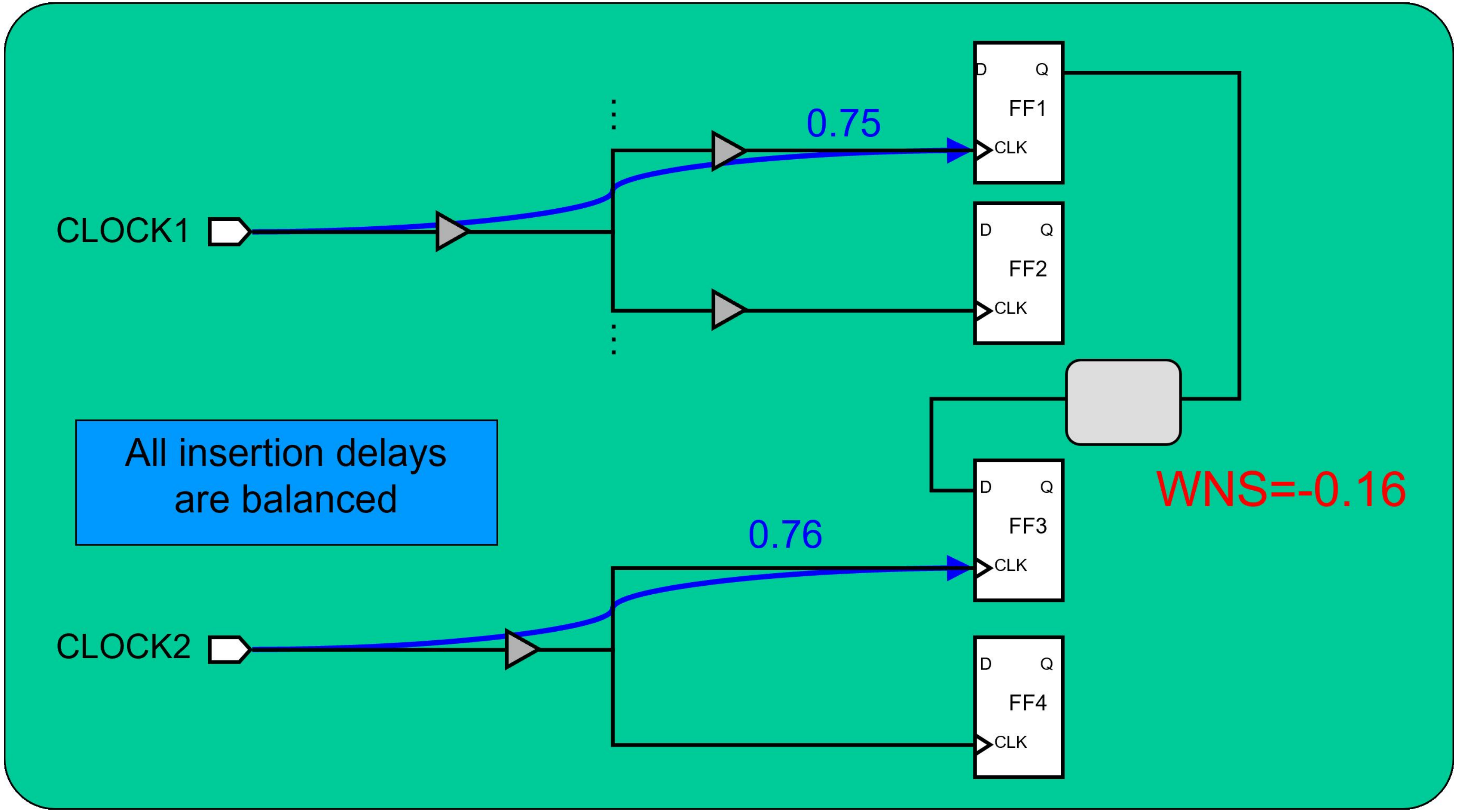


By default CTS does not perform inter-clock skew balancing → May result in worse setup timing violations

The path from FF1 to FF3 will have an additional setup penalty of

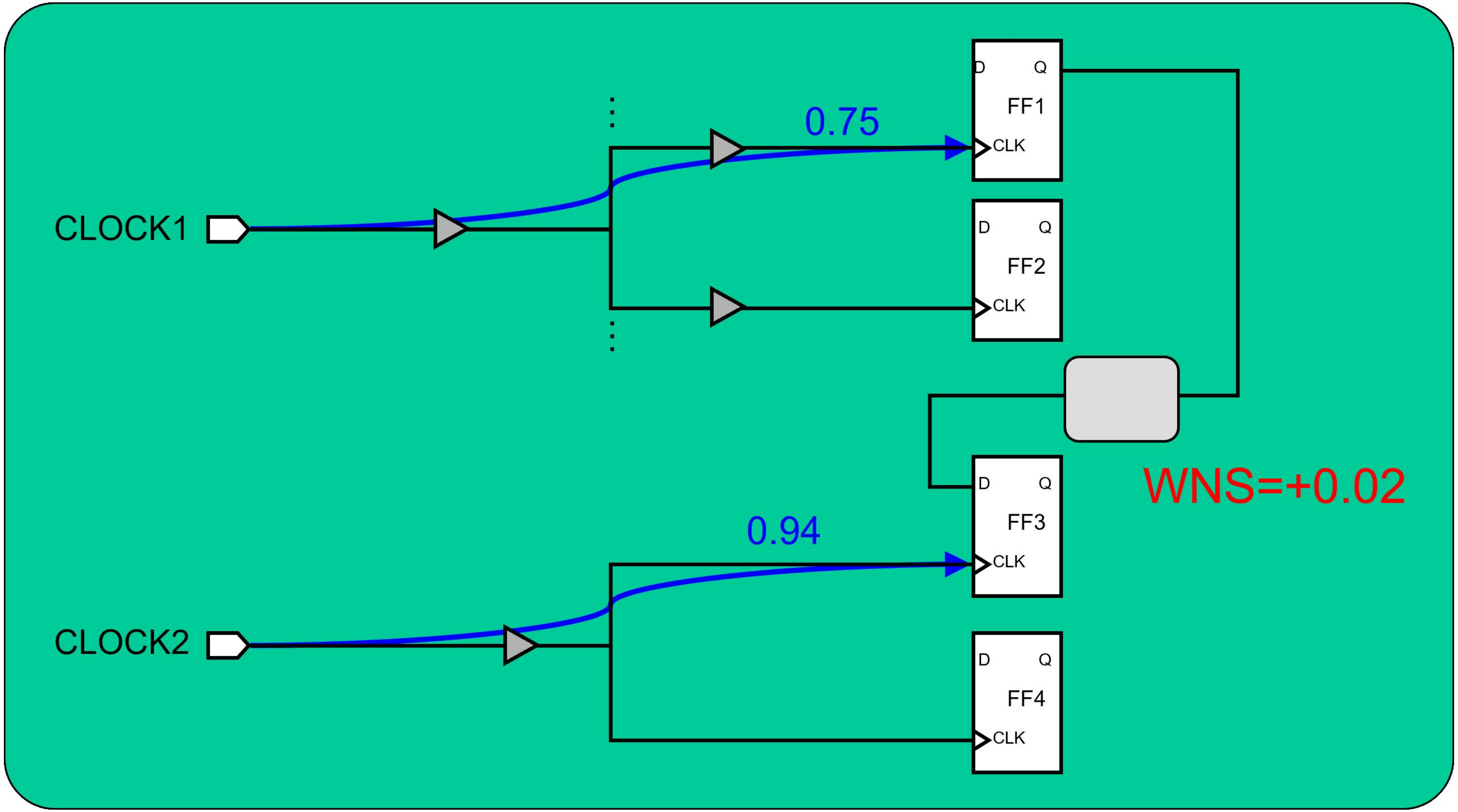
0.75-0.32=0.43

Inter-Clock Delay Balancing



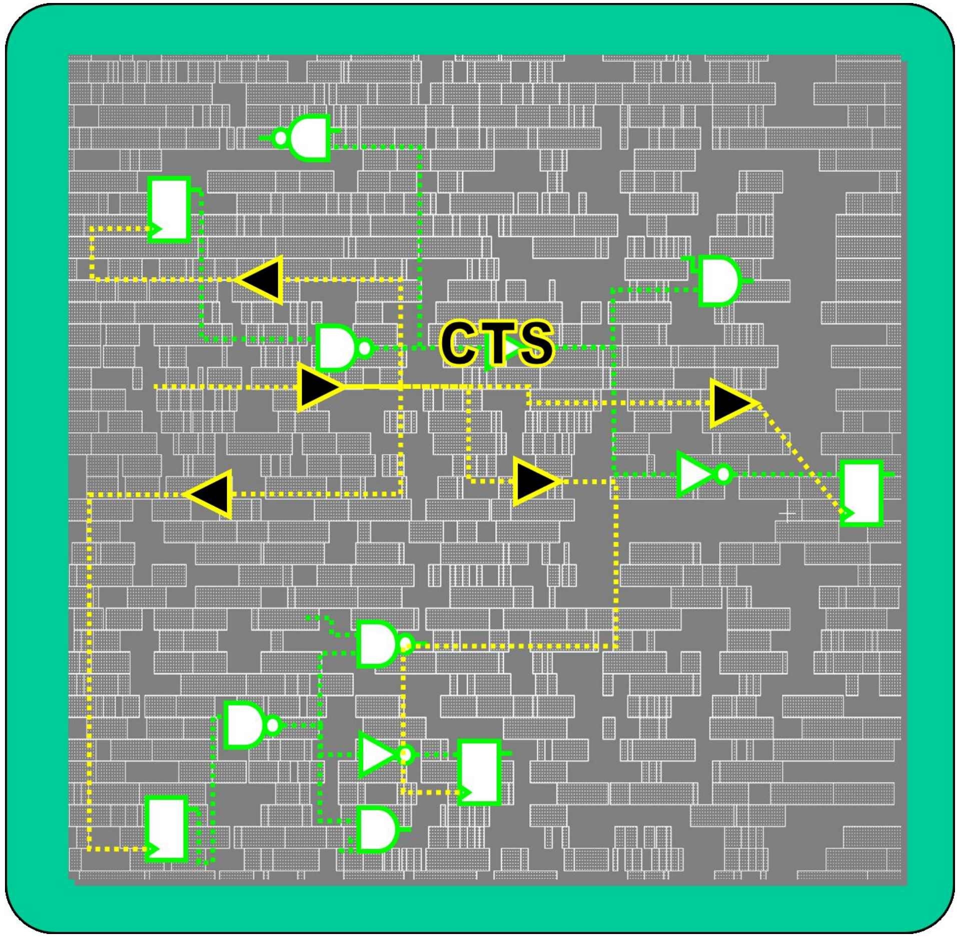
All insertion delays are balanced

Inter-Clock Delay Balancing: With Offset



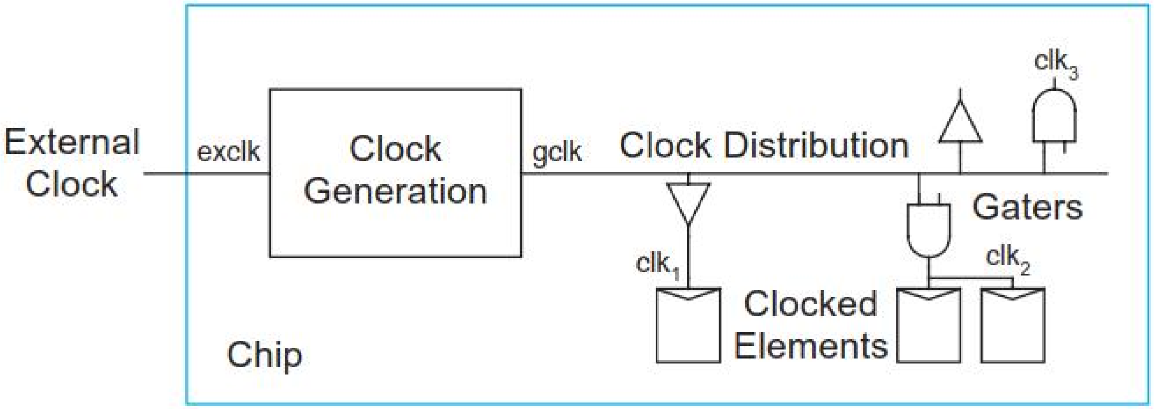
Effects of Clock Tree Synthesis

- Clock buffers added
- Congestion may increase
- Non clock cells may have been moved to less ideal locations
- Inserting clock tress can introduce new timing and max tran/cap violations, which will be checked in the next stages



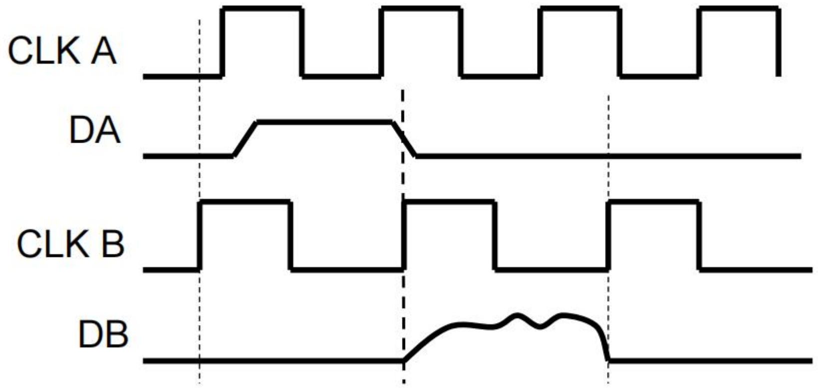
Clock Generation

- Where does the clock come from?
  - • The easiest way to generate a clock is by using a ring oscillator or some other non-stable circuit, but these are susceptible to PVT variations.
  - • Therefore, clocks are generally generated off-chip using a crystal and an oscillation circuit.
  - • However, usually only one off-chip clock can be used (a single frequency) and the frequency that can be input to the chip is limited to around 100 MHz.
  - • Therefore, on-chip local clock generation is employed, usually with a PLL or DLL.



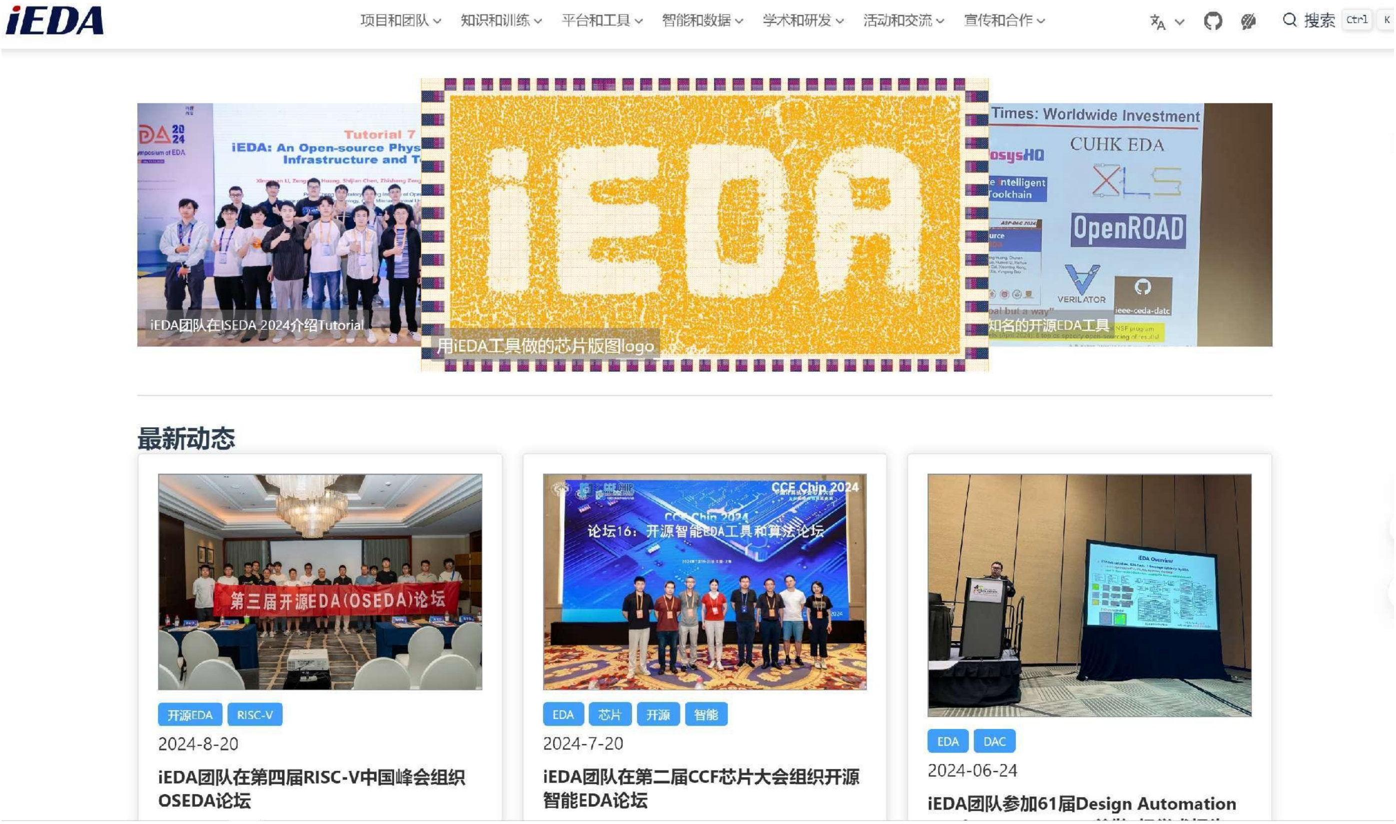
Multi Clock Domains

- Multi Clock Domains
  - • Most system-on-chips have several components that communicate with different external interfaces and run on different clock frequencies.
    - • In other words, they have multiple asynchronous clock domains.
  - • Asynchronous clocks cannot communicate with each other in a straightforward fashion:



Conclusions

- Clock Tree Problem
- Clock Concepts
  - Objectives： Skew, Latency, Power, Wirelength, Buffer Area
  - Constraints： Fanout, Slew, Load,
- Clock Topology
  - Tree, Grid, Spine, Hybrid
- Clock Routing
  - H-tree, GH-tree, RMST, SALT, ZST/BST/UST， CBS
- Clock Optimization
  - Buffer Insertion, Wire Sizing, Gate Sizing, Wire Snake, Wire Linking, Clock Gate
  - Clock and Data Concurrent Optimation (CCOPT)



iEDA团队在ISEDA 2024介绍Tutorial

用iEDA工具做的芯片版图logo

知名的开源EDA工具

最新动态

开源EDA RISC-V
2024-8-20
iEDA团队在第四届RISC-V中国峰会组织OSEDA论坛

EDA 芯片 开源 智能
2024-7-20
iEDA团队在第二届CCF芯片大会组织开源智能EDA论坛

EDA DAC
2024-06-24
iEDA团队参加61届Design Automation

Thanks

iEDA website: ieda.oscc.cc

李兴权（Xingquan Li）
fzulxq@gmail.com