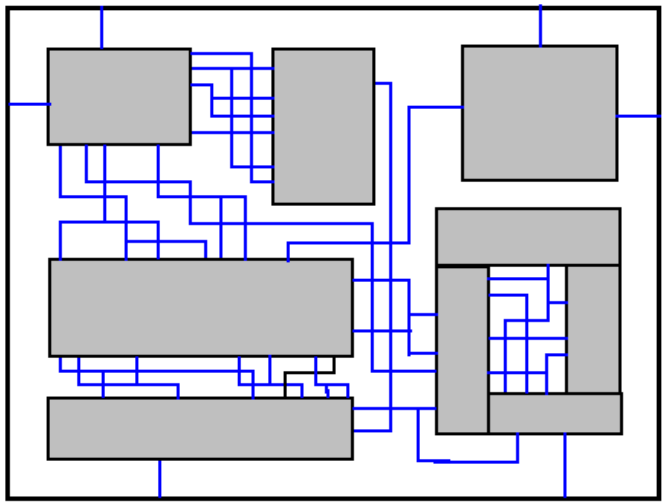
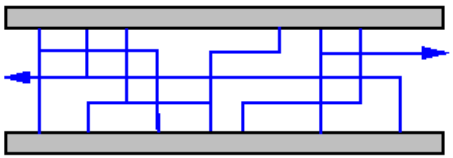
Unit 5E: Channel, Clock, and Power/Ground Routing

- Course contents
  - Channel routing (for old nodes, but good to learn its algorithms)
  - Clock routing
  - Power/ground routing
  - (Bus routing)
- Readings
  - Chapters 9.3 and 9.4

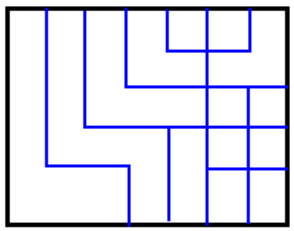


Detailed routing



channel routing

Pins are fixed at top & bottom sides



switchbox routing

Pins are fixed at adjacent sides (i.e., not opposite)
→ Not much flexibility

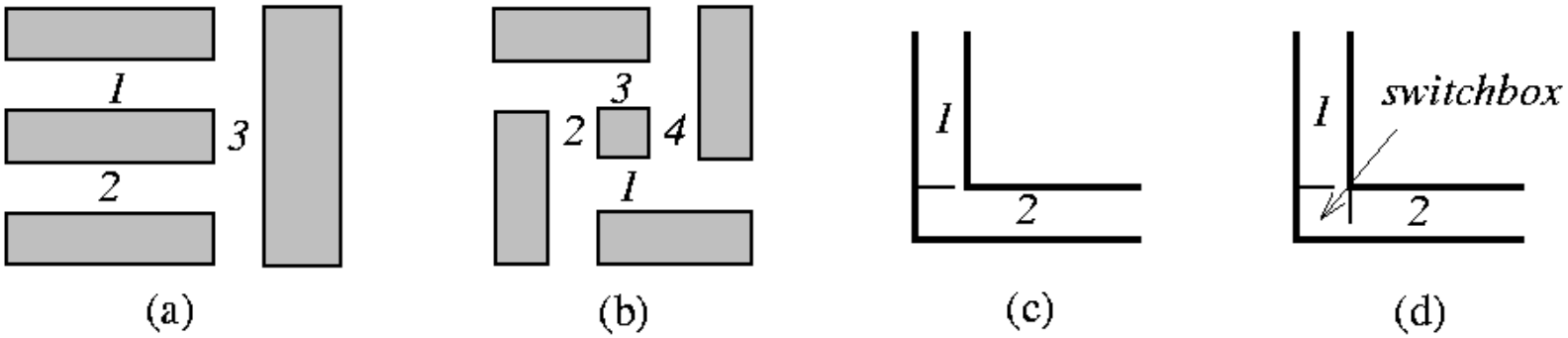
Order of Routing Regions and L-Channels

(a) No conflicts in case of routing in the order of 1, 2, and 3.

(b) No ordering is possible to avoid conflicts.

(c) The situation of (b) can be resolved by using L-channels.

(d) An L-channel can be decomposed into a channel and a switchbox.



Routing Considerations

- Number of terminals (two-terminal vs. multi-terminal nets)
- Net widths (power and ground vs. signal nets)
- Via restrictions (stacked vs. conventional vias)
  - Cut used to be square; now can be rectangle (called bar via)
- Boundary types (regular vs. irregular)
- Number of layers (two vs. three, more layers?)
- Net types (critical vs. non-critical nets)

- Non-default routing (NDR) rules (certain nets have their own constraints, their own spacing, width, and contact types)
  - These are design dependent. Usually they are for critical nets

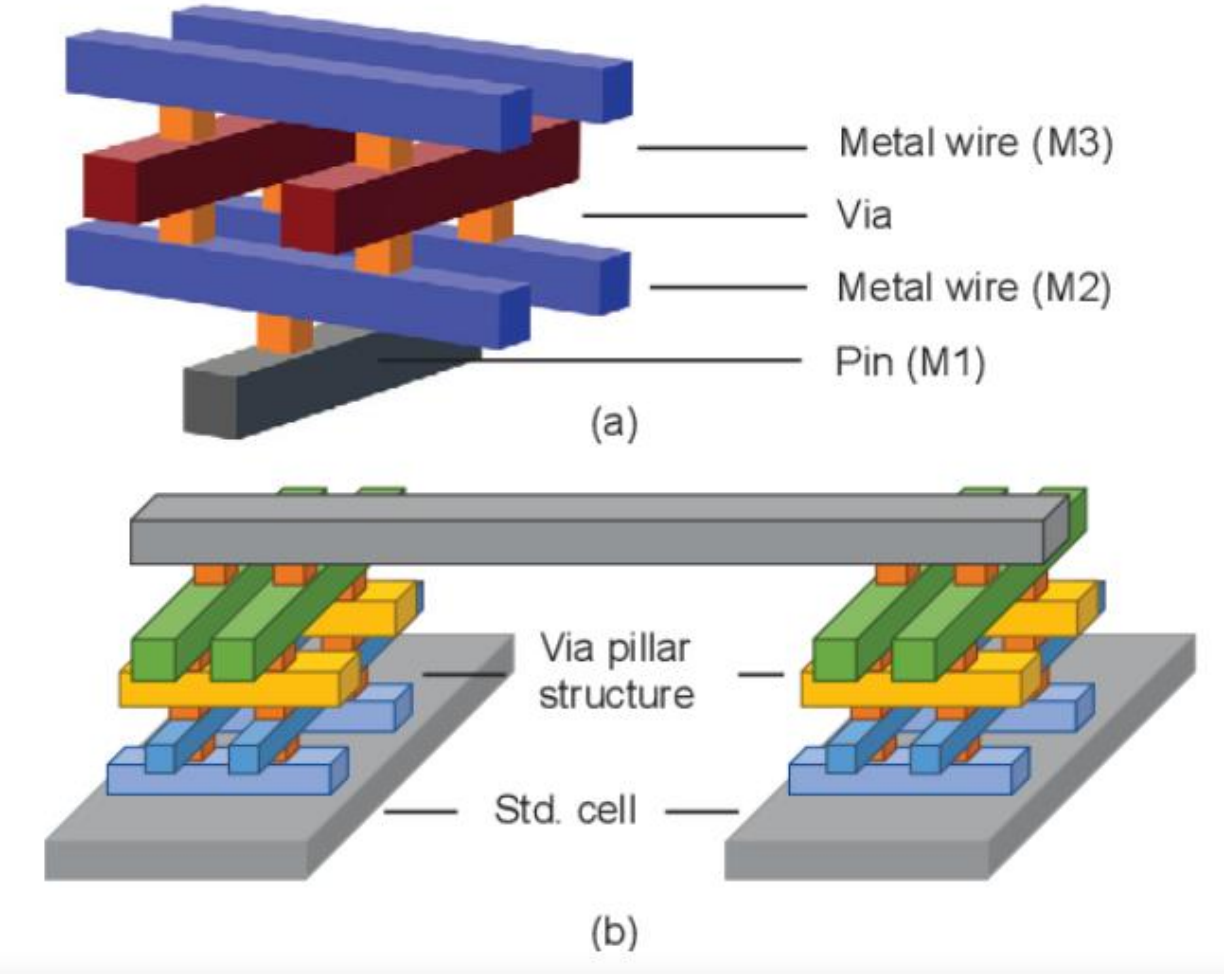
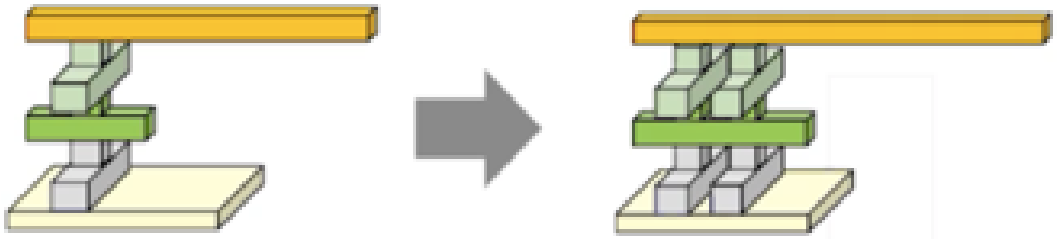


Figure 2: An illustration of via pillar and its structure. (a) A via pillar consists of parallel metal wires and multiple vias. (b) Two via pillars are built on standard cells, and connected by a metal wire on a higher metal layer.

Via modeling and NDR will impact global routing

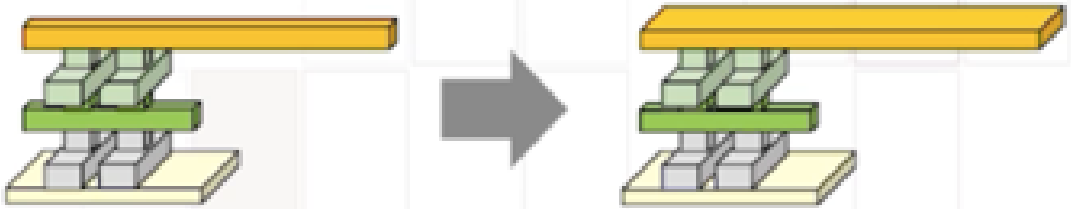
Via Pillar Insertion

Add parallel VIAs on source side of timing critical nets for R reduction



AutoNDR Routing

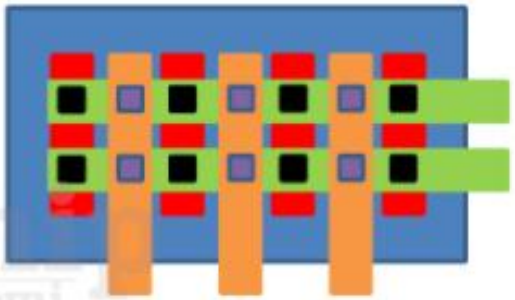
Enlarge metal width and spacing of timing critical nets (non-default routing, NDR) for RC reduction



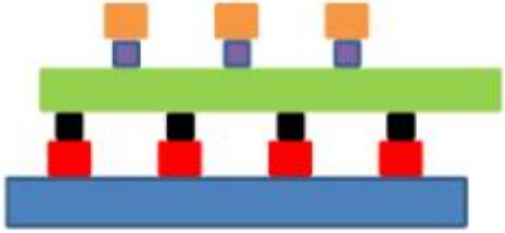
Wire Engineering and Via Ladders

- Team used selective layer optimization, buffering, pre-routes, and via ladders to exploit the fast layers for critical signals
- Two types of via ladders
  - High Performance: for large buffers driving long wires
  - EM: for high-activity gates (e.g., clock drivers)
  - Mitigated EM issues on large fanout nodes with high activity

Top Via Ladder View



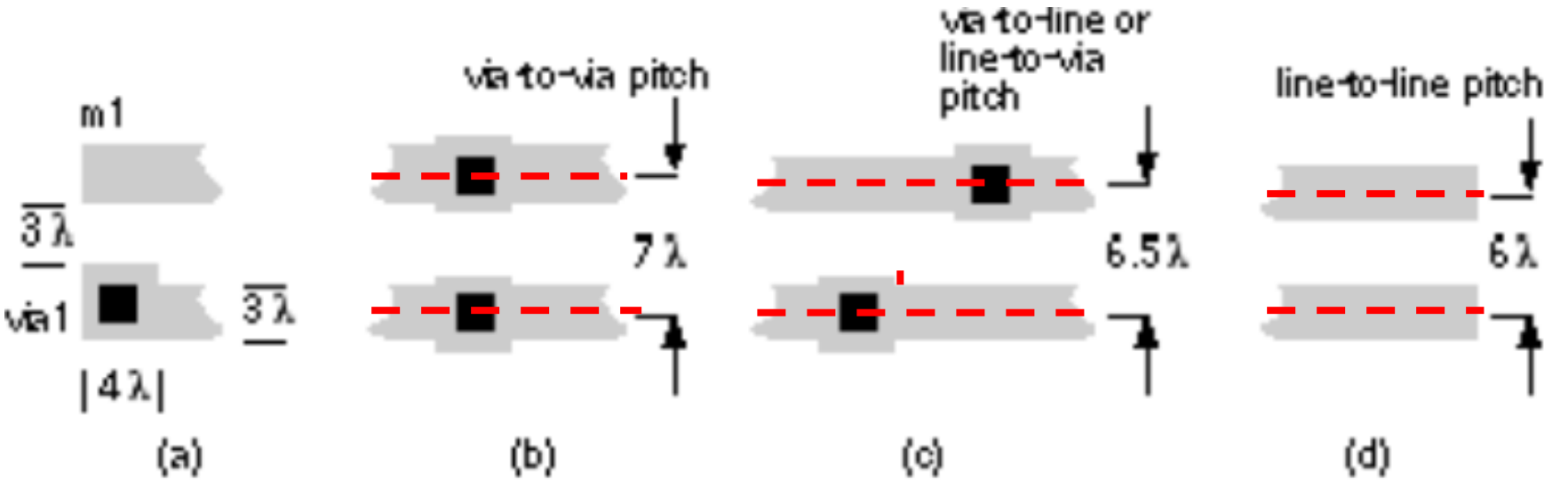
Side Via Ladder View



- How about extracting R/C for via ladders? Is it pre-characterized?

Routing Models

- **Grid-based model:**
  - A grid is super-imposed on the routing region.
  - Wires follow paths along the grid lines.
  - Pitch: distance between two grid lines (usually wire width +spacing)
    - But if contact is bigger, pitch likely will be



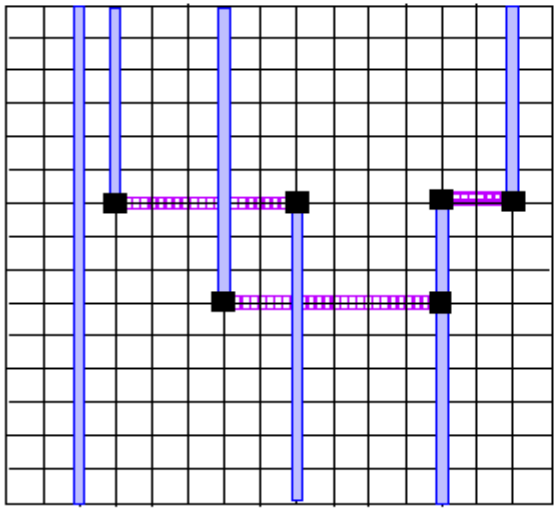
2(half contact width) + spacing

Line-to-line + ½(contact width – line width)

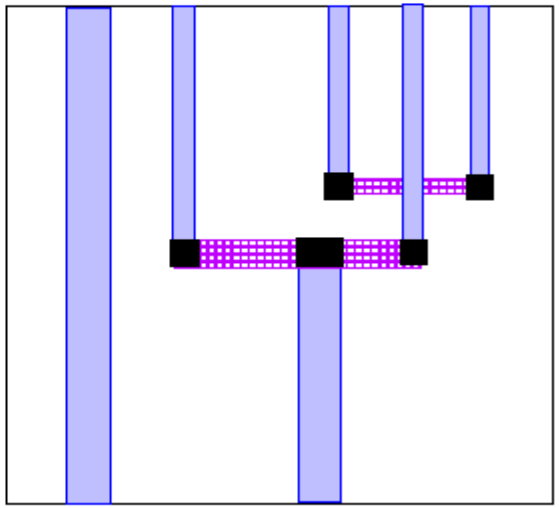
2(half width) + spacing

Routing Models

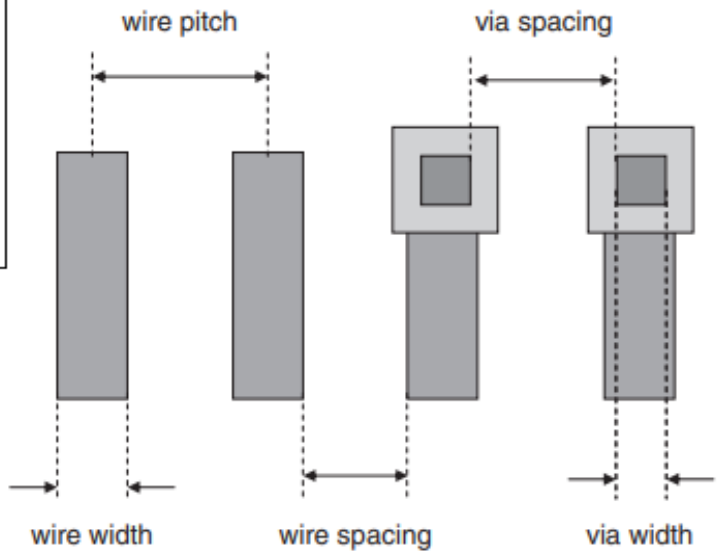
- **Gridless model:**
  - Any model that does not follow this “gridded” approach.
  - Area can be more compact; but routing programs became harder



grid–based

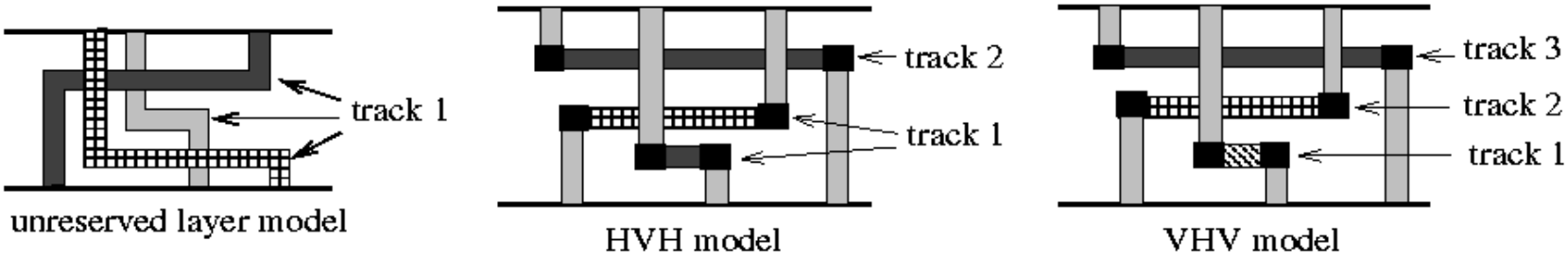


gridless



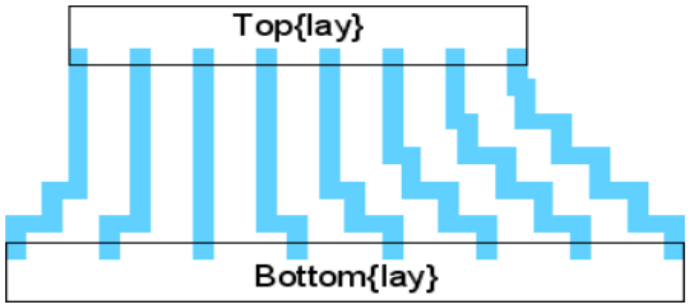
Models for Multi-Layer Routing

- **Unreserved layer model:** Any net segment is allowed to be placed in any layer.
- **Reserved layer model:** Certain type of segments are restricted to particular layer(s).
  - Two-layer: HV (horizontal-Vertical), VH
  - Three-layer: HVH, VHV



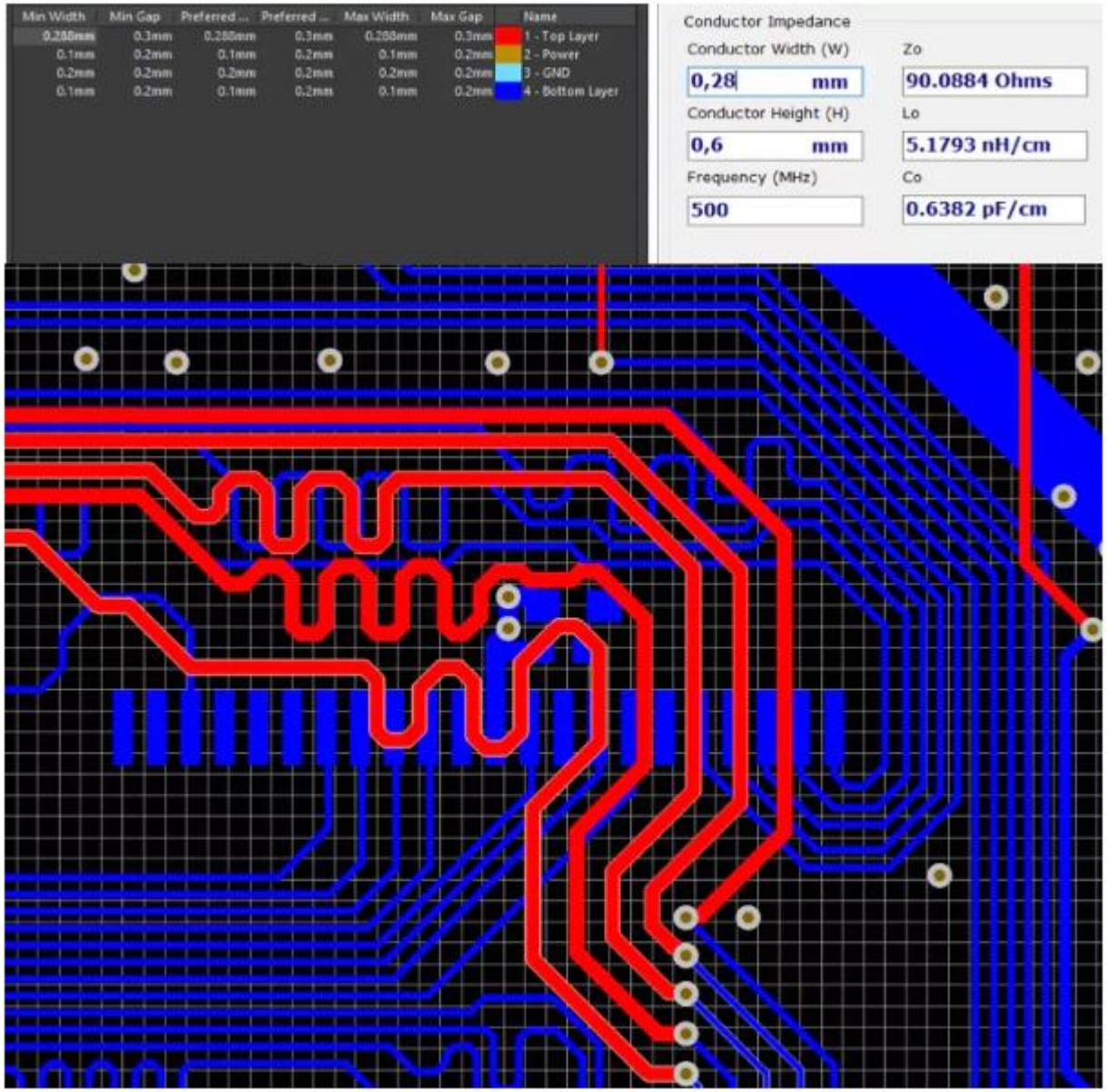
3 types of 3–layer models

River Routing
(single layer)



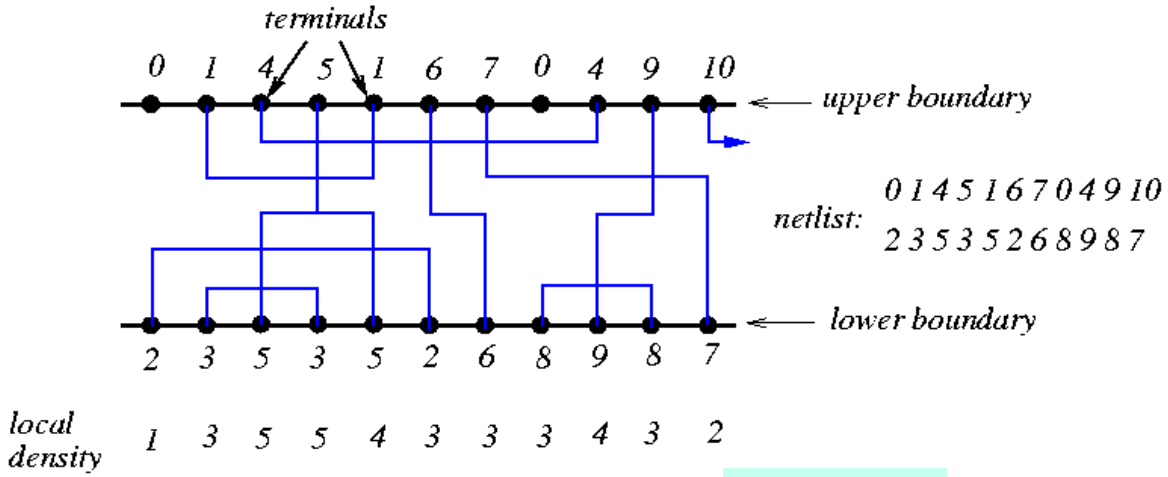
https://www.integrasources.com/blog/high-speed-pcb-design-guidelines/

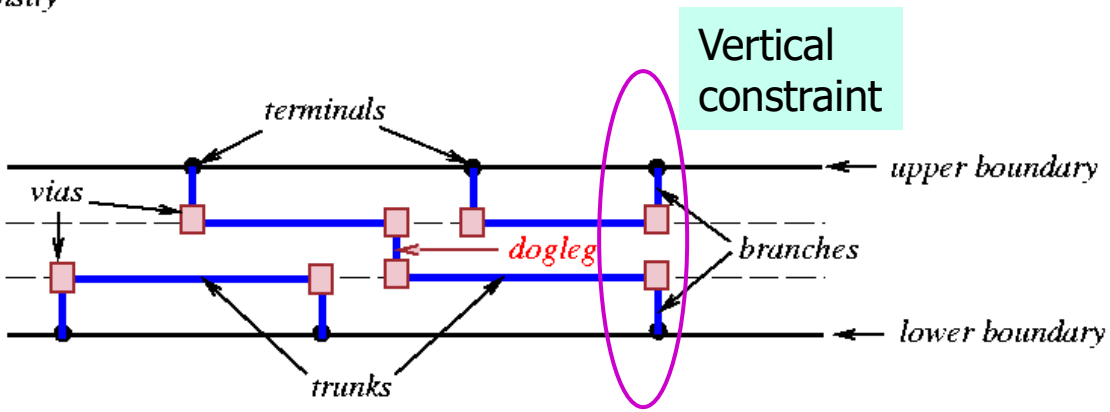
Special routing: equal resistance (or length)

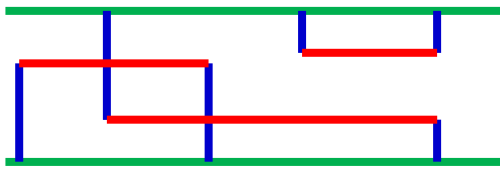


The picture shows one and the same impedance depending on the distance to the GND layer. The longer the distance, the thicker the track.

Terminology for Channel Routing







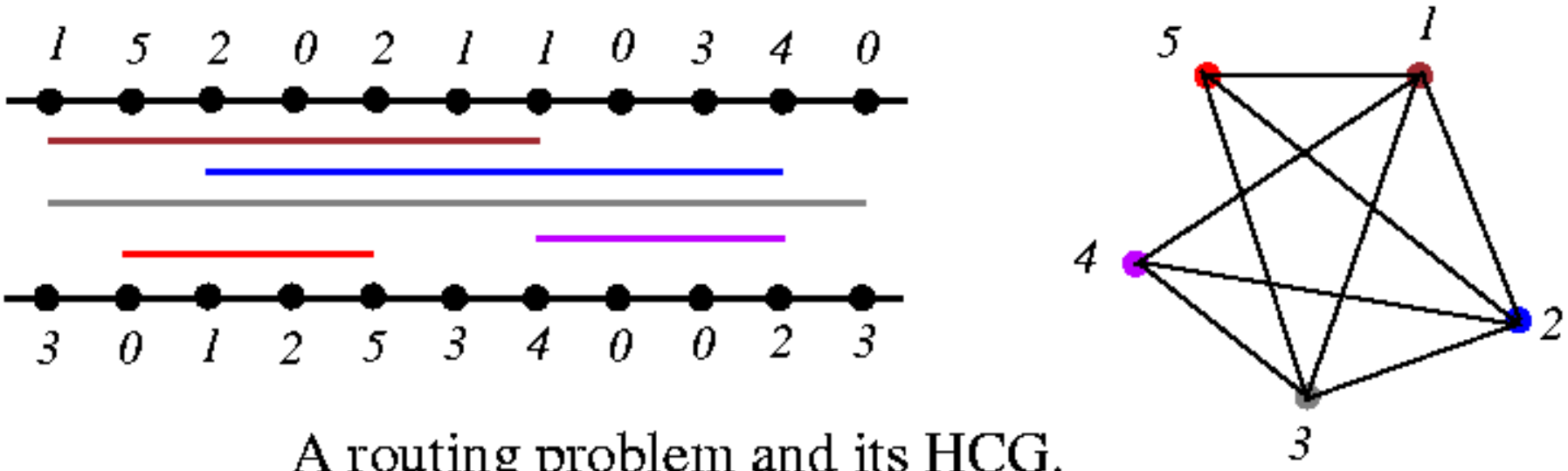
- Local density at column *i*, *d(i)*: total # of nets that crosses column *i*.
- **Channel density:** maximum local density
  - # of horizontal tracks required ≥ channel density.

Channel Routing Problem

- **Assignments of horizontal segments of nets to tracks**
- **Assignments of vertical segments to connect**
  - horizontal segments of the same net in different tracks, and
  - the terminals of the net to horizontal segments of the net.
- **Horizontal and vertical constraints must not be violated**
  - Horizontal constraints between two nets: the horizontal span of two nets overlaps each other.
  - Vertical constraints between two nets: there exists a column such that the terminal on top of the column belongs to one net and the terminal on bottom of the column belongs to another net.
- **Objective: Channel height is minimized** (i.e., channel area is minimized).

Horizontal Constraint Graph (HCG)

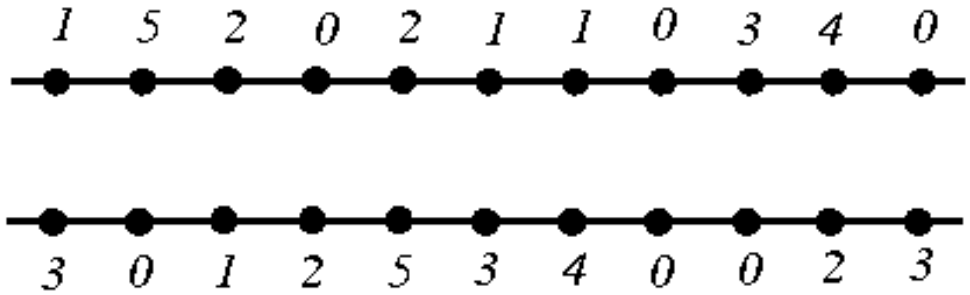
- HCG $G = (V, E)$ is **undirected** graph where
  - $V = \{ v_i \mid v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j) \mid$ a horizontal constraint exists between n_i and $n_j\}$.
- For graph G: vertices $\Leftrightarrow$ nets; edge $(i, j) \Leftrightarrow$ net i overlaps net j

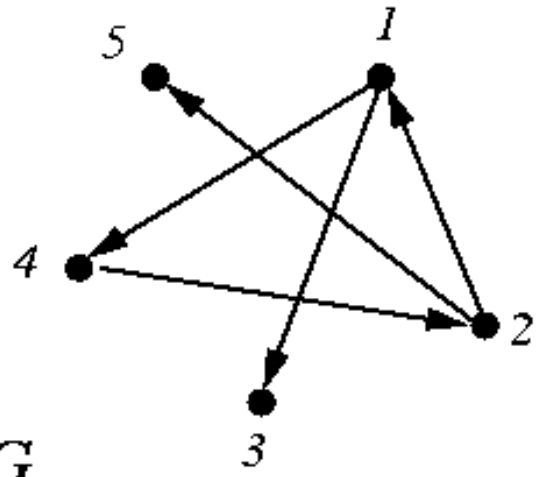


A routing problem and its HCG.

Vertical Constraint Graph (VCG)

- VCG $G = (V, E)$ is **directed** graph where
  - $V = \{ v_i \mid v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j) \mid$ a vertical constraint exists between n_i and $n_j\}$.
- For graph G: vertices $\Leftrightarrow$ nets; edge $i \rightarrow j \Leftrightarrow$ net i must be above net j.





A routing problem and its VCG.

2-L Channel Routing: Basic Left-Edge Algorithm

- Hashimoto & Stevens, “Wire routing by optimizing channel assignment within large apertures,” DAC-1971.
- **No vertical constraint.**
- HV-layer model is used.
- **Doglegs are not allowed.**
- Treat each net as an interval.
- Intervals are sorted according to their left-end *x*-coordinates
- Intervals (nets) are routed one-by-one according to the order.
- For a net, tracks are scanned from top to bottom, and the first track that can accommodate the net is assigned to the net.

- Optimality: produces a routing solution with the minimum # of tracks (if no vertical constraint).

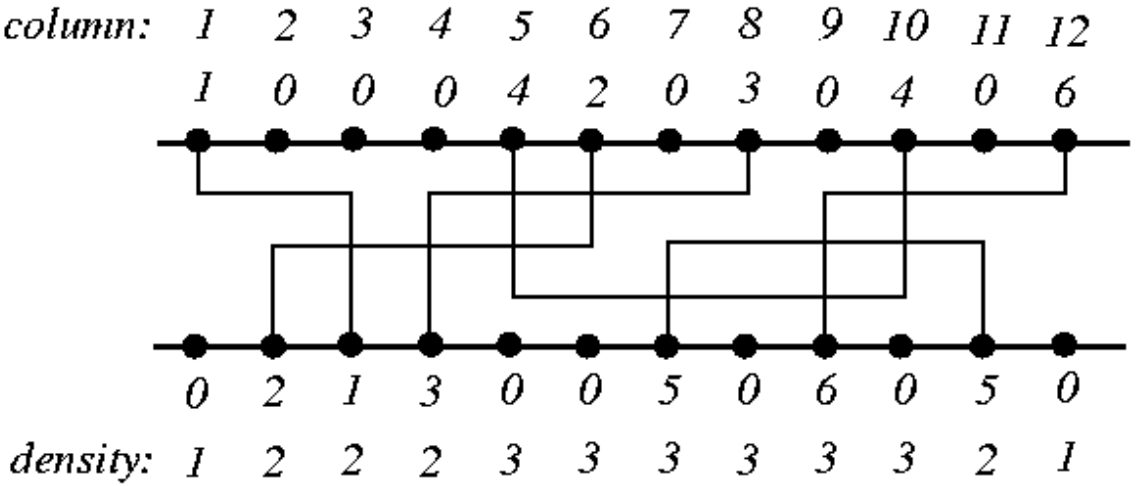
Basic Left-Edge Algorithm

Algorithm: Basic_Left-Edge(*U, track*[*j*])
U: set of unassigned intervals (nets) $I_1, \ldots, I_n$;
$I_j=[s_j, e_j]$: interval *j* with left-end *x*-coordinate s_j and right-end e_j;
track[*j*]: track to which net *j* is assigned.

```
1  begin
2  U ← {I1, I2 , …, In};
3  t ← 0;
4  while (U ≠ ∅ ) do
5      t ← t + 1;
6      watermark ← 0;
7      while (there is an Ij ∈ U s.t. sj > watermark) do
8          Pick the interval Ij ∈ U with sj > watermark,
           nearest watermark;
9          track[j] ← t;
10         watermark ← ej;
11         U ← U - {Ij};
12 end
```

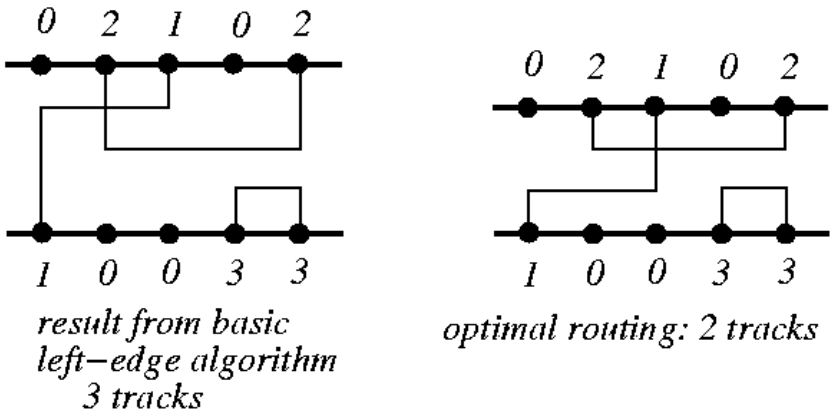
Basic Left-Edge Example

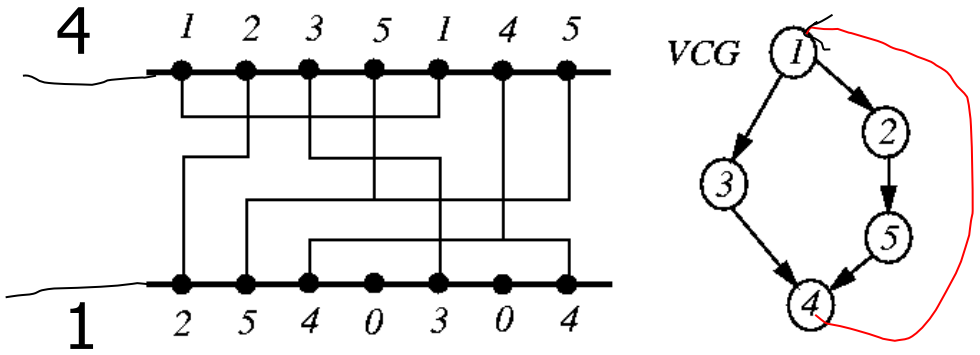
- $U = \{I_1, I_2, \ldots, I_6\}$; $I_1 = [1, 3]$, $I_2 = [2, 6]$, $I_3 = [4, 8]$, $I_4 = [5, 10]$, $I_5 = [7, 11]$, $I_6 = [9, 12]$.
- t =1:
  - Route I_1: *watermark* = 3;
  - Route I_3 : *watermark* = 8;
  - Route I_6: *watermark* = 12;
- t = 2:
  - Route I_2 : *watermark* = 6;
  - Route I_5 : *watermark*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



Basic Left-Edge Algorithm

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





When a red edge is added, a cycle is formed

Constrained Left-Edge Algorithm

Algorithm: Constrained_Left-Edge(U, $track[j]$**)**

U: set of unassigned intervals (nets) I_1, …, I_n;

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

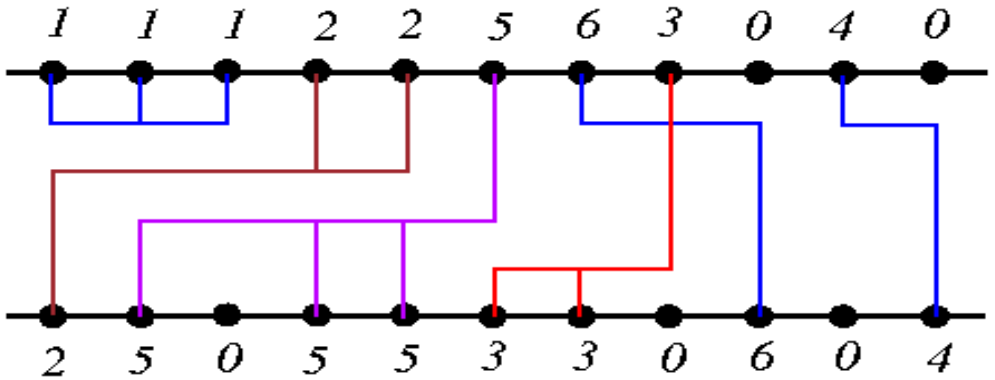
$track[j]$: track to which net j is assigned.

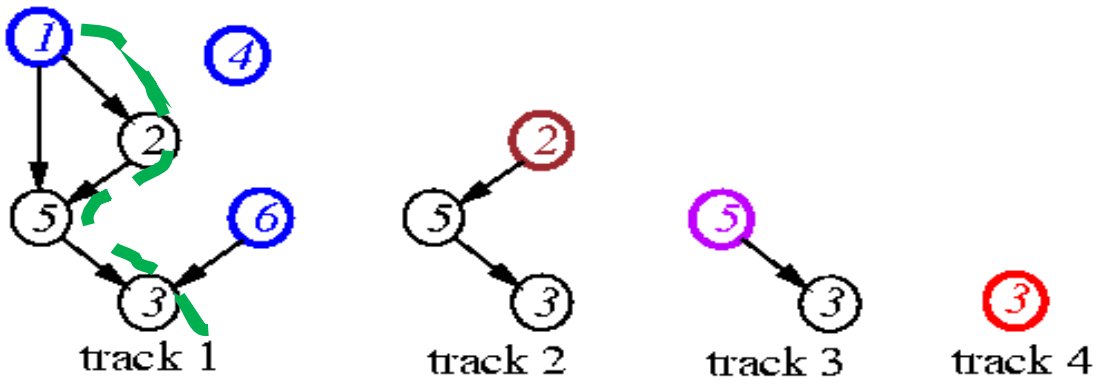
```
1 begin
2 U ← { I1, I2, …, In};
3 t ← 0;
4 while (U ≠ ∅) do
5     t ← t + 1;
6     watermark ← 0;
7      while (there is an unconstrained I_j ∈ U s.t. s_j > watermark)
      do
8     Pick the interval I_j ∈ U that is unconstrained,
          with s_j > watermark, nearest watermark;
9         track[j] ← t;
10        watermark ← e_j;
11        U ← U - {I_j};
12 end
```

Constrained Left-Edge Example

- I_1 = [1, 3], I_2 = [1, 5], I_3 = [6, 8], I_4 = [10, 11], I_5= [2, 6], I_6 = [7, 9].
- Track 1: Route I_1 (cannot route I_3); Route I_6; Route I_4.
- Track 2: Route I_2; cannot route I_3.
- Track 3: Route I_5.
- Track 4: Route I_3.

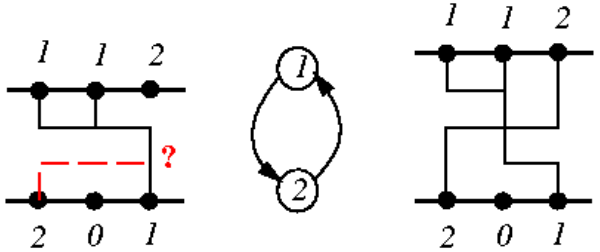
Can we improve?



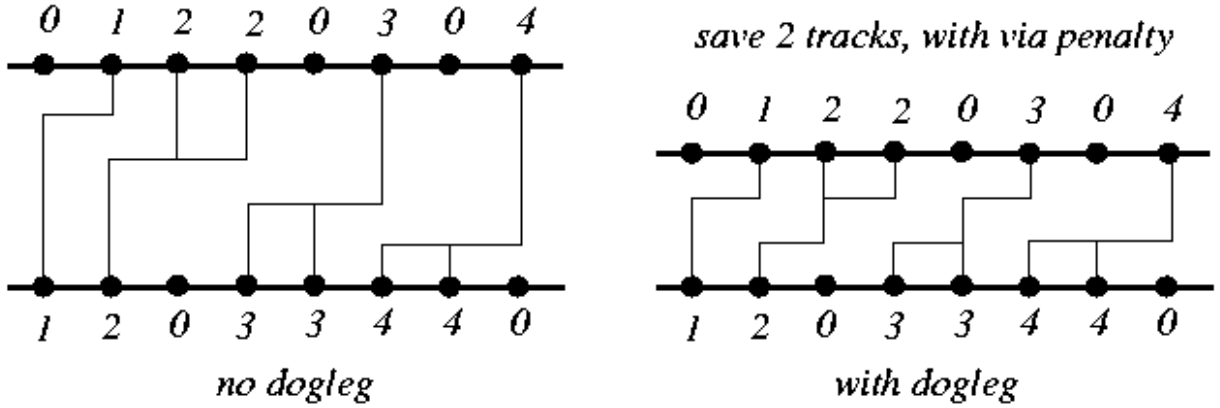


Dogleg Channel Router

- Deutch, “A dogleg channel router,” 13rd DAC, 1976.
- **Drawback of Left-Edge: cannot handle the cases with constraint cycles.**
  - **Doglegs** are used to resolve constraint cycle.

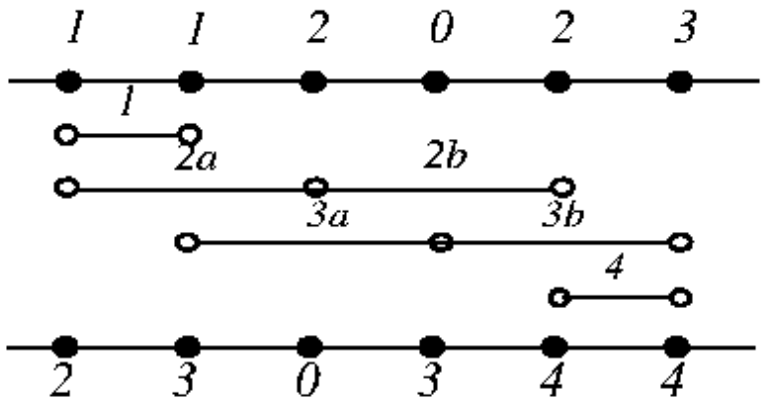


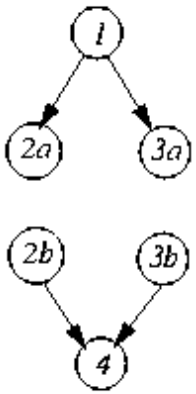
- **Drawback of Left-Edge: the entire net is on a single track.**
  - **Doglegs** are used to place parts of a net on different tracks to minimize channel height.
  - Might incur penalty for additional vias.

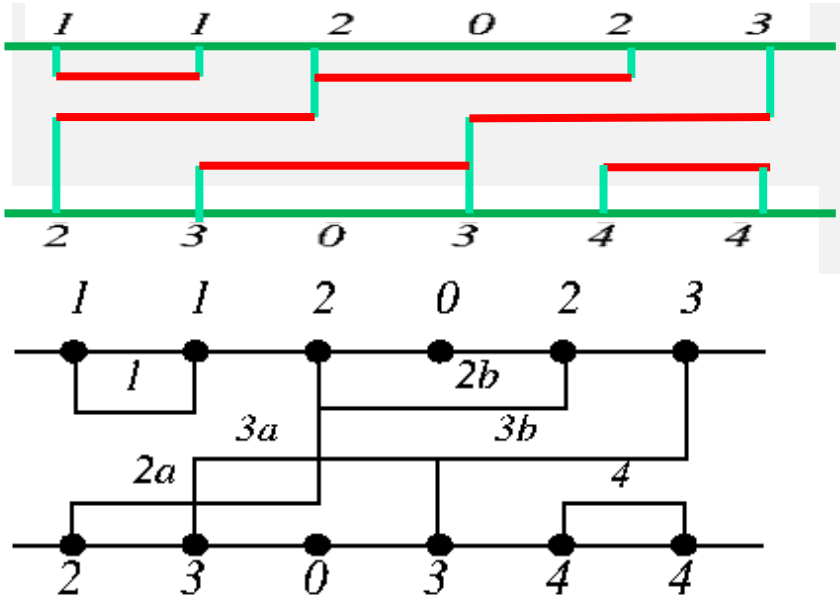


Dogleg Channel Router

- Each multi-terminal net is broken into a set of 2-terminal nets.
- Two parameters are used to control routing:
  - Range: Determine the # of consecutive 2-terminal subnets of the same net that can be placed on the same track.
  - Routing sequence: Specifies the starting position and the direction of routing along the channel.
- Modified Left-Edge Algorithm is applied to each subnet.

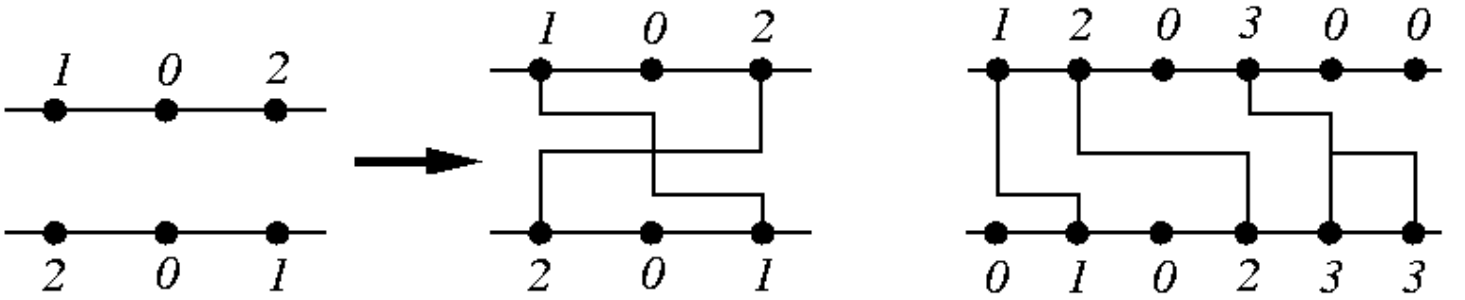






Restricted vs. Unrestricted Doglegging

- **Unrestricted doglegging:** Allow a dogleg even at a position where there is no pin.
- **Restricted doglegging:** Allow a dogleg only at a position where there is a pin belonging to that net.
- The dogleg channel router does not allow unrestricted doglegging.



dogleg channel router will fail! *Solution exists!* *restricted doglegging*

dogleg splits a net into subnets.

Robust Channel Router

- Yoeli, “A robust channel router,” IEEE TCAD, 1991.
- Alternates between top and bottom tracks until the center is reached.
- The working side is called the *current side*.
- Net weights are used to guide the assignment of segments in a track, which
  - favor nets that contribute to the channel density;
  - favor nets with terminals at the current side;
  - penalize nets whose routing at the current side would cause vertical constraint violations.
- Allows unrestricted doglegs by rip-up and re-route.

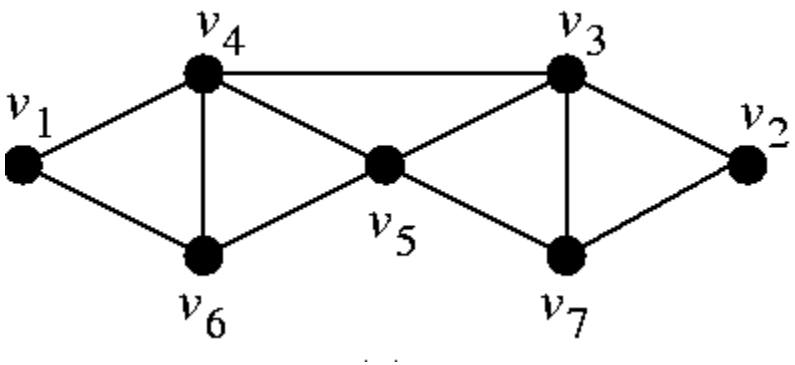
Robust Channel Router

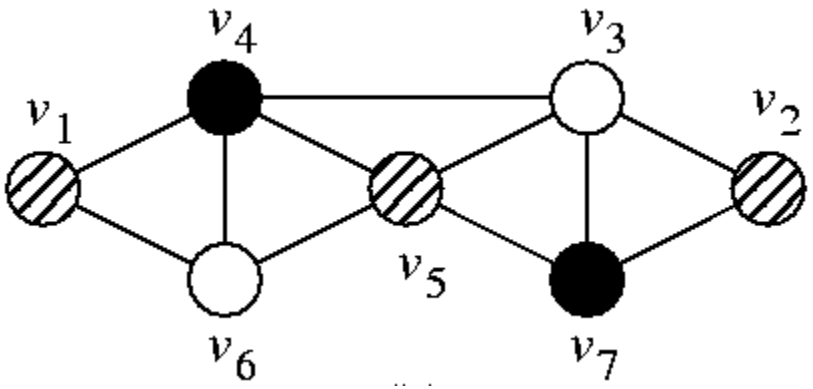
- Select the set of nets for the current side by solving the maximum weighted independent set problem for interval graphs.
  - In graph theory, an **independent set**, or **anti-clique** is a set of vertices in a graph, no two of which are adjacent
  - NP-complete for general graphs, but can be solved efficiently for interval graphs using dynamic programming.

Interval Graphs

- There is a vertex for each interval.
- Vertices corresponding to overlapping intervals are connected by an edge.
- Solving the track assignment problem is equivalent to finding a **minimal vertex coloring** of the graph.
  - Using the least # of colors to label vertices so that no edge connects to two same colored vertices
  - Below is using 3 colors

Black nodes/nets assigned to one track; white nodes assigned to another track





Robust Channel Router

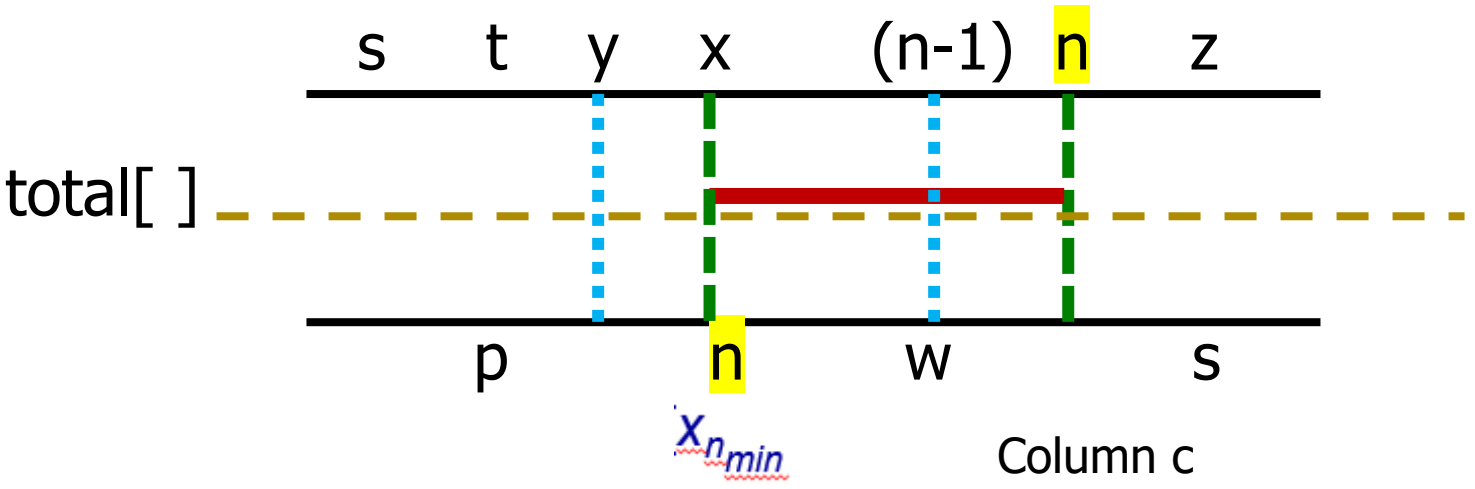
1. For all nets i whose intervals contain the columns of maximal density, add a large number B to the weights w_i. This stimulates the decrease of the density of the channel routing problem in the next iteration.

2. For each net i that has a current-side terminal at the column positions x, add to w_i the local density $d(x)$ for all x. This encourages the selection of nets with terminals at the current side.

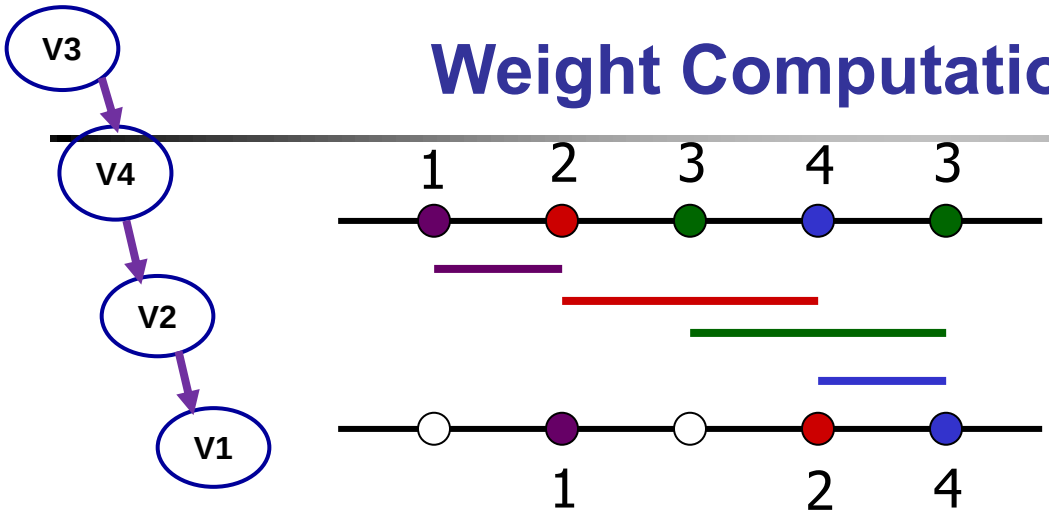
3. For each column x for which an assignment of some net i to the current side will create a vertical constraint violation, subtract $Kd(x)$ from w_i; K is a parameter that should have a value between 5 and 10. This discourages the creation of vertical constraint violations.

Robust Channel Router

- Main ideas:
  - The interval for net *i* is denoted by $[x_{i_{min}}, x_{i_{max}}]$; its weight is w_i.
  - Process channel from left to right column; the optimal cost for position *c* is denoted by total[*c*];
  - A net *n* with a rightmost terminal at position *c* is taken into the solution if total[*c* – 1] < w_n + total[$x_{n_{min}}$ – 1].



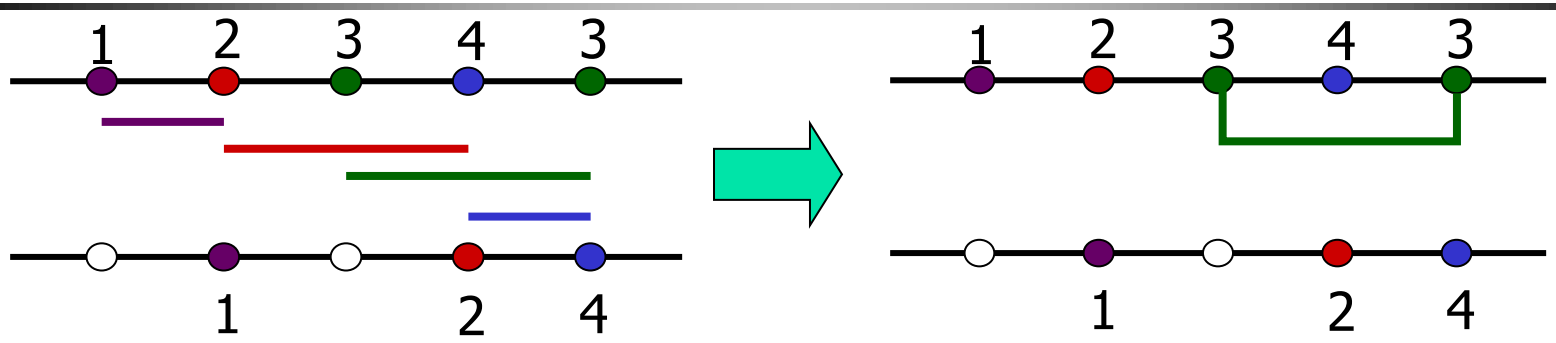
Weight Computation (cont'd)



d(1) = 1
d(2) = 2
d(3) = 2
d(4) = 3 (nets 2, 3, 4)
d(5) = 2

- Computation of the weight w_i for net i:
  1. favor nets that contribute to the channel density: add a large B to w_i.
  2. favor nets with current side terminals at column x: add d(x) to w_i.
  3. penalize nets whose routing at the current side would cause vertical constraint violations: subtract Kd(x) from w_i, K = 5 ~ 10.
  - Assume B = 1000 and K = 5 in the 1st iteration (top side):
    - w_1 = (0) + (1) + (-5 * 2) = -9 (its span does not cover max density (0), column 1 has density 1, route 1 causes vertical vio (-5), column 2 has density 2.)
    - w_2 = (1000) + (2) + (-5 * 3) = 987 (net2 has one term at current side with density 2)
    - w_3 = (1000) + (2+2) + (0) = 1004 (net3 has two terminals at current top)
    - w_4 = (1000) + (3) + (-5 * 2) = 993 (net4 has 1 term at current top with density 3)

Top-Row Net Selection

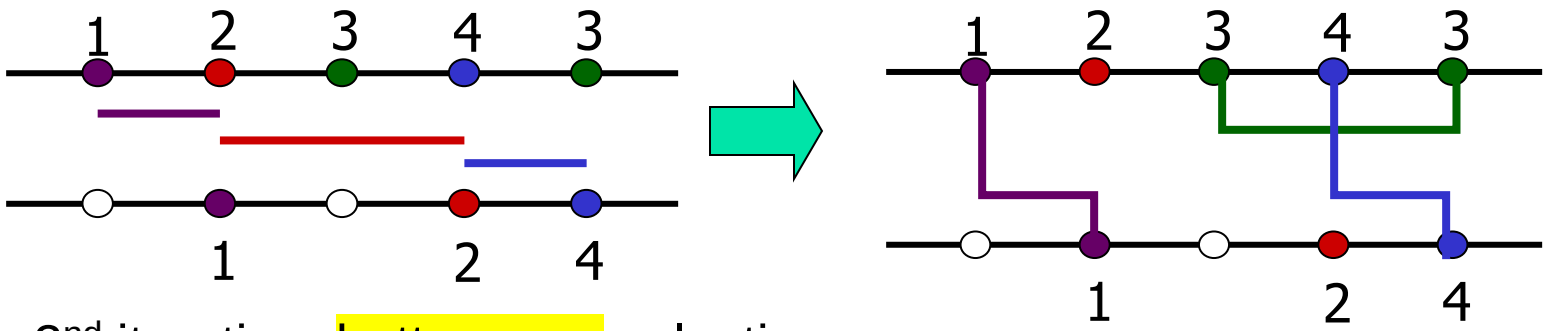


- $w_1 = -9$, $w_2 = 987$, $w_3 = 1004$, $w_4 = 993$.
- A net *n* with a rightmost terminal at position *c* is taken into the solution if: total[$c - 1$] < w_n + total[$x_{n_{min}} - 1$].

total[1] = 0	selected_net[1] = 0
total[2] = max(0, 0-9) = 0 (net1 rightmost@2)	selected_net[2] = 0
total[3] = 0	selected_net[3] = 0
total[4] = max(0, w_2+total[1]) = 987	selected_net[4] = 2
total[5] = max(987, 0+1004, 0+993) = 1004	selected_net[5] = 3

- Select nets backwards from right to left and with no horizontal constraints: Only net 3 is selected for the top row. (Net 2 is not selected since it overlaps with net 3.)

Bottom-Row Net Selection

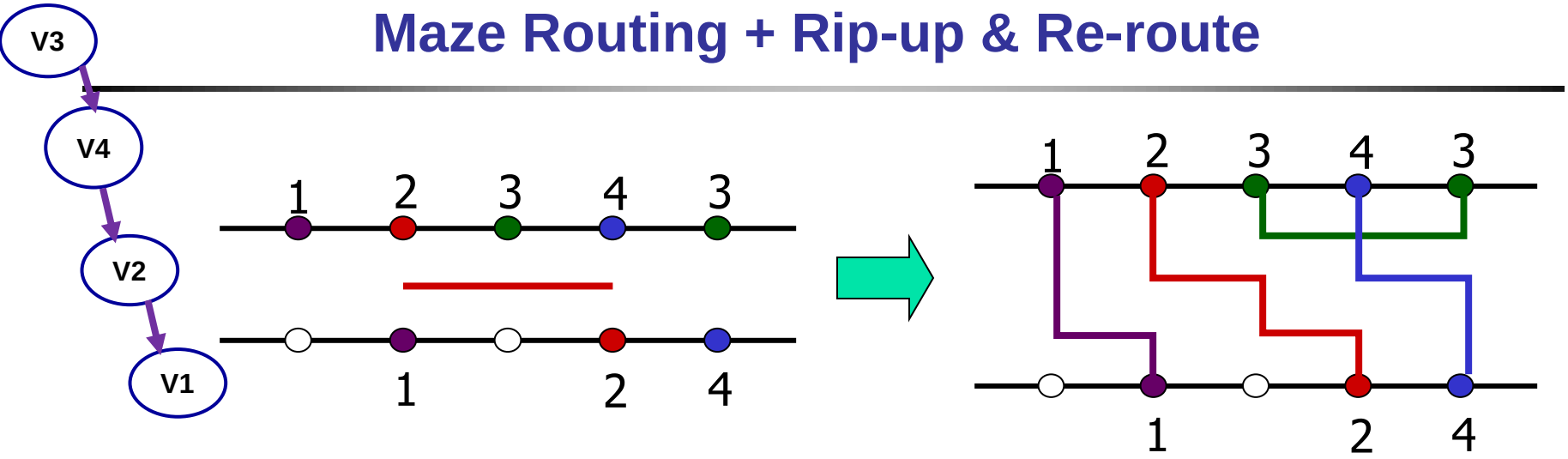


- 2nd iteration: bottom-row selection
  - w_1 = (1000) + (2) + (0) = 1002
  - w_2 = (1000) + (2) + (-5 * 2) = 992
  - w_4 = (1000) + (1) + (-5 * 2) = 991

total[1] = 0	selected_net[1] = 0
total[2] = max(0, 0+1002) = 1002	selected_net[2] = 1
total[3] = 1002	selected_net[3] = 0
total[4] = max(1002, 0+992) = 1002	selected_net[4] = 0
total[5] = max(1002, 1002+991) = 1993	selected_net[5] = 4

- Nets 4 and 1are selected for the bottom row.

Maze Routing + Rip-up & Re-route



- 3rd iteration
  - Routing net 2 in the middle row leads to an infeasible solution.
  - Apply maze routing and rip-up and re-route nets 2 and 4 to fix the solution.
- From vertical constraints, we know if no dogleg, we need 4 tracks. Now with bends, we can pack them onto 3 tracks

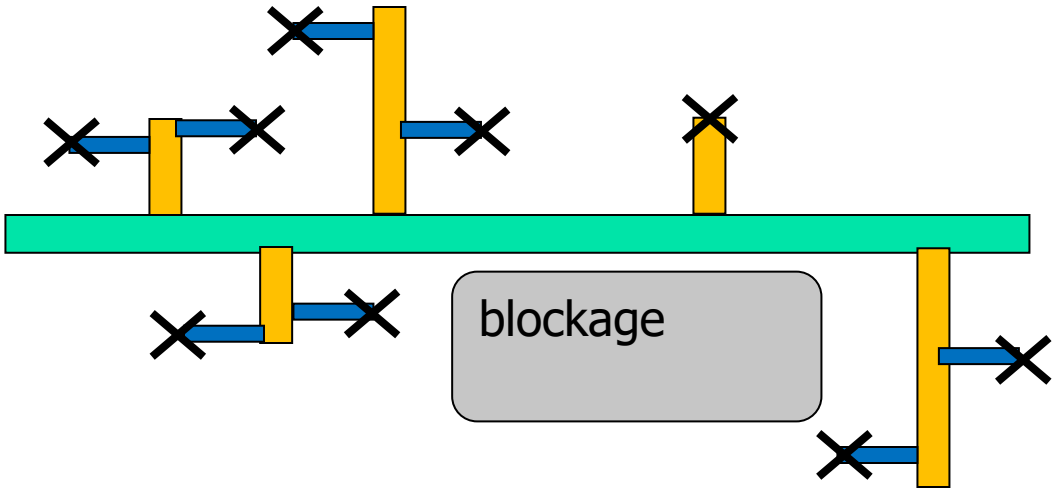
Robust Channel Router

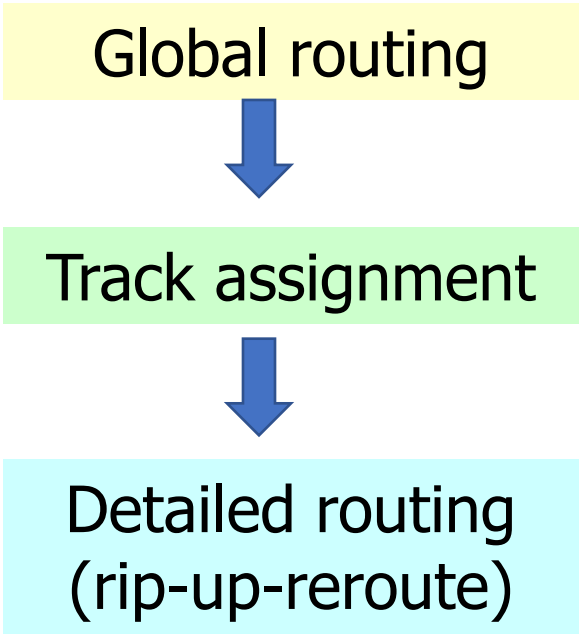
```
robust_router (struct netlist N)
{
  set of int row;
  struct solution S;
  int total[channel_width + 1], selected_net[channel_width
  int top, height, c, r, i;

  top ← 1;
  height ← density(N);
  for (r ← 1; r ≤ height; r ← r + 1) {
    for all "nets i in netlist N"
      w_i ← compute_weight(N, top);
    total[0] ← 0;
    for (c ← 1; c ≤ channel_width; c ← c + 1) {
      selected_net[c] ← 0;
      total[c] ← total[c − 1];
      if ("some net n has a top terminal at position c")
        if (w_n + total[x_{n_min} − 1]) > total[c]) {
          total[c] ← w_n + total[x_{n_min} − 1]);
          selected_net[c] ← n;
        }
      if ("some net n has a bottom terminal at position c")
        if (w_n + total[x_{n_min} − 1]) > total[c]) {
          total[c] ← w_n + total[x_{n_min} − 1]);
          selected_net[c] ← n;
        }
    }
    row ← ∅;
    c ← channel_width;
    while (c > 0)
      if (selected_net[c]) {
        n ← selected_net[c];
        row ← row ∪ {n};
        c ← x_{n_min} − 1;
      }
      else
        c ← c − 1;
    solution ← solution ∪ {row};
    top ← !top;
    N ← "N without the nets selected in row"
  }/* for */
  "apply maze routing to eliminate possible vertical constraint violation
}
```

However, Channel Route Usage Dwindled

- But trunk (or spine) routing is popularly used
  - Interval graph concept is useful
  - If one trunk/spine is hard to achieve, ok to break into two
  - Often need to consider multiple layers
  - Used in memory and custom layout, clock tree routing, …




Global routing
Track assignment
Detailed routing
(rip-up-reroute)

Track Assignment

- Nowadays for digital SoC routing, track assignment is deployed between global routing and detailed routing

http://users.ece.northwestern.edu/~haizhou/publications/iccad02.pdf
TrackAssign: a desirable intermediate step between GR and DR

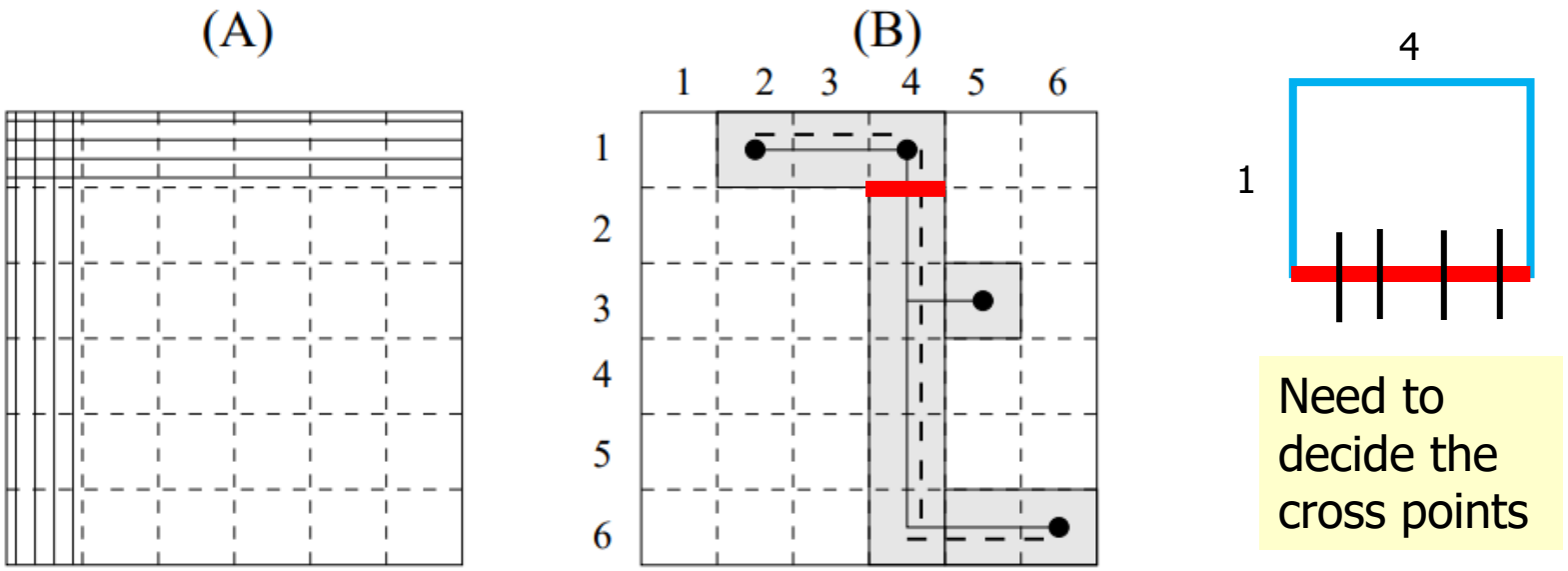


Fig. 1. Global cells and markings of horizontal and vertical grid lines in row and column respectively. Marking for a net shown in shaded GCs.

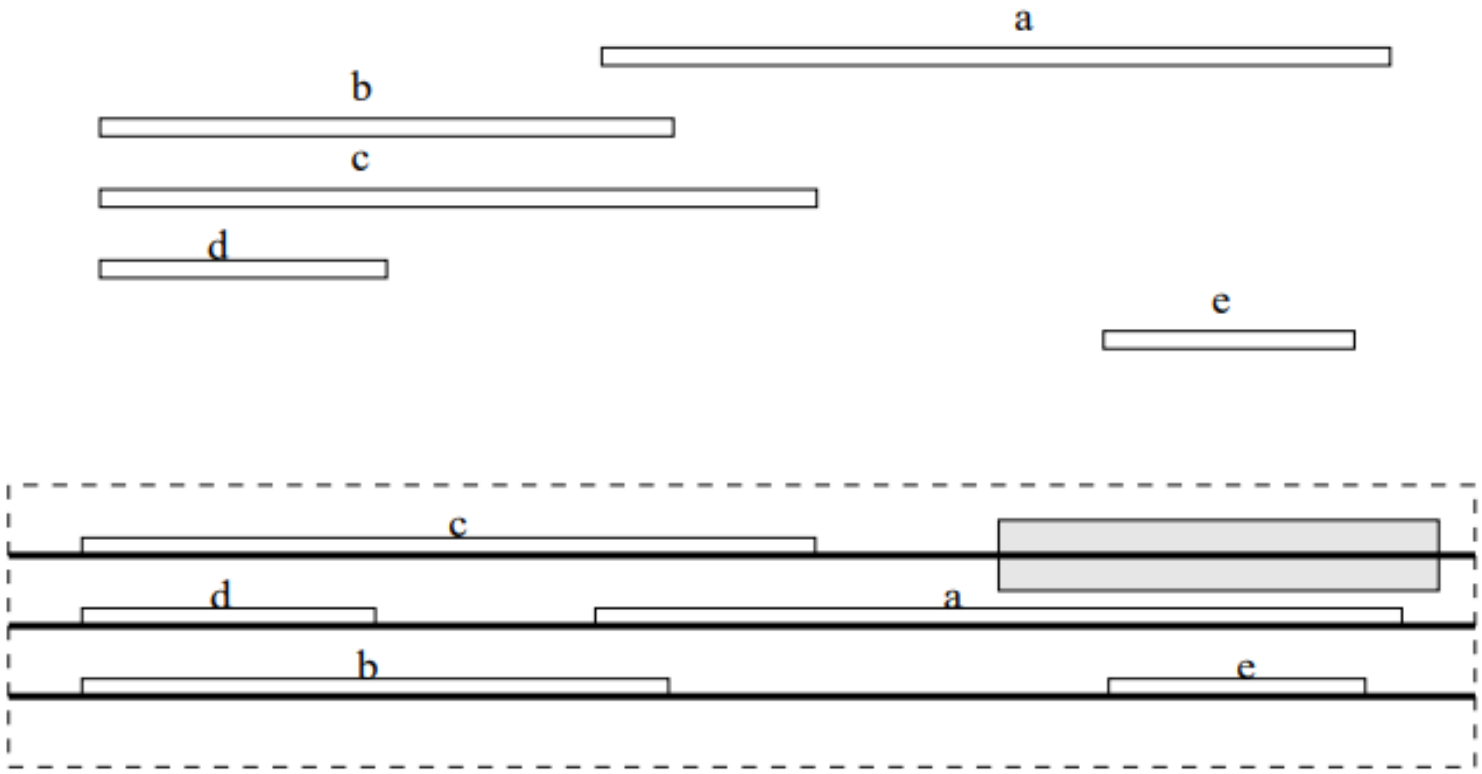
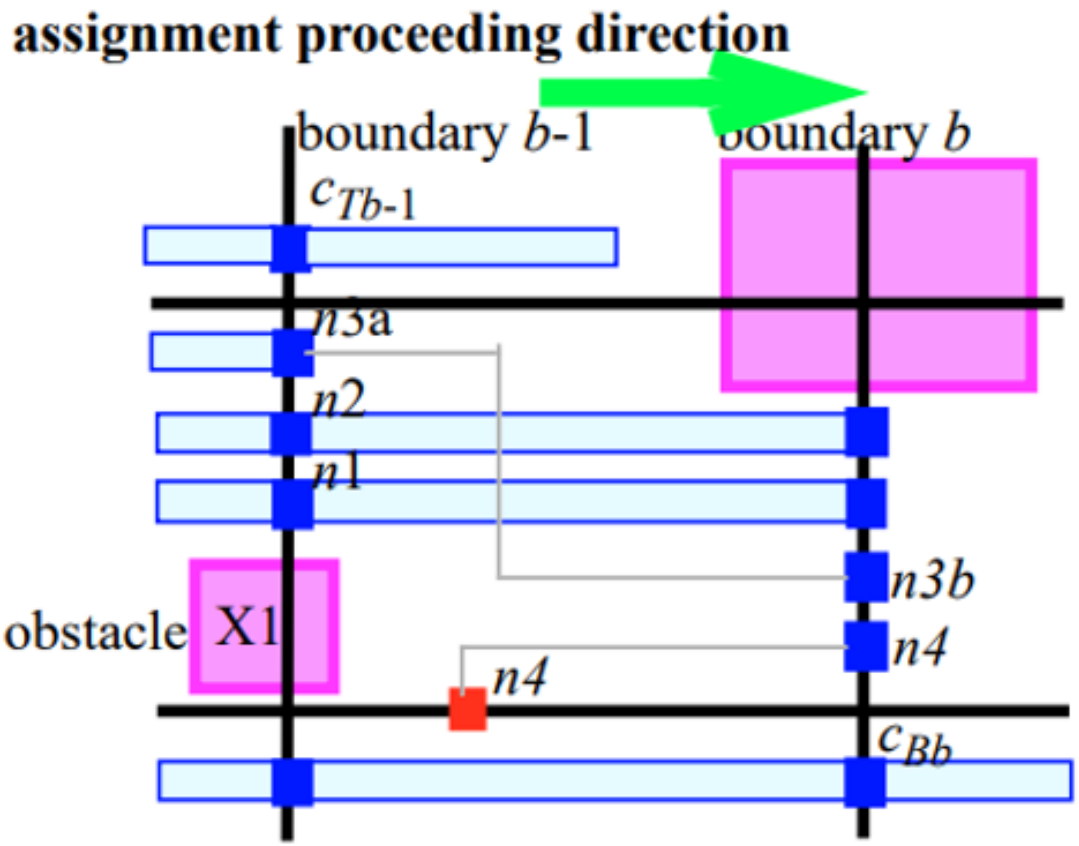


Fig. 6. Solution to the track assignment problem given in Figure 3 with lookahead.

Another Picture Related To Pin Assignment



(a) proceed to boundary *b* after *b*-1

- It is fine to have short, design rule errors in Track Assignment.
- Detailed router does rip-up-and-reroute (RR) to clean them up
- It is an area based RR; area's size can be changed in iterations. In the beginning, area tends to be smaller; then, gradually enlarged (to get more freedom to resolve DRC err.)

- A tough one, especially want to fix as many errors as possible

Timing and Crosstalk Driven Area Routing

Hsiao-Ping Tseng
University of Washington
Dept. of Electrical Engineering
Seattle, WA 98195
206-685-8678
hptseng@twolf.ee.washington.edu

Louis Scheffer
Cadence Design Systems, Inc.
555 River Oaks Parkway, Bldg. 2, MS2B2
San Jose, CA 95134
408-944-7114
lou@cadence.com

Carl Sechen
University of Washington
Dept. of Electrical Engineering
Seattle, WA 98195
206-685-8756
sechen@twolf.ee.washington.edu

We present a timing and crosstalk driven router for the chip assembly task that is applied **between global and detailed routing**. Our new approach aims to process the crosstalk and timing constraints by ordering nets and tuning wire spacing in a quantitative way. Our graph-based optimizer pre-routes wires on the global routing grids incrementally in two stages - net order assignment and space relaxation. The **timing delay of each critical** path is calculated taking into account **interconnect coupling capacitance**. The objective is to reduce the delays of critical nets with negative timing slack values, by tuning net ordering and adding extra wire spacing. It shows a remarkable 8.4-25% delay reduction for MCNC benchmarks for wire geometric ratio=2.0, against a 33% delay reduction if interconnect interference disappear.

https://www.academia.edu/26876513/Timing_and_crosstalk_driven_area_routing?email_work_card=title

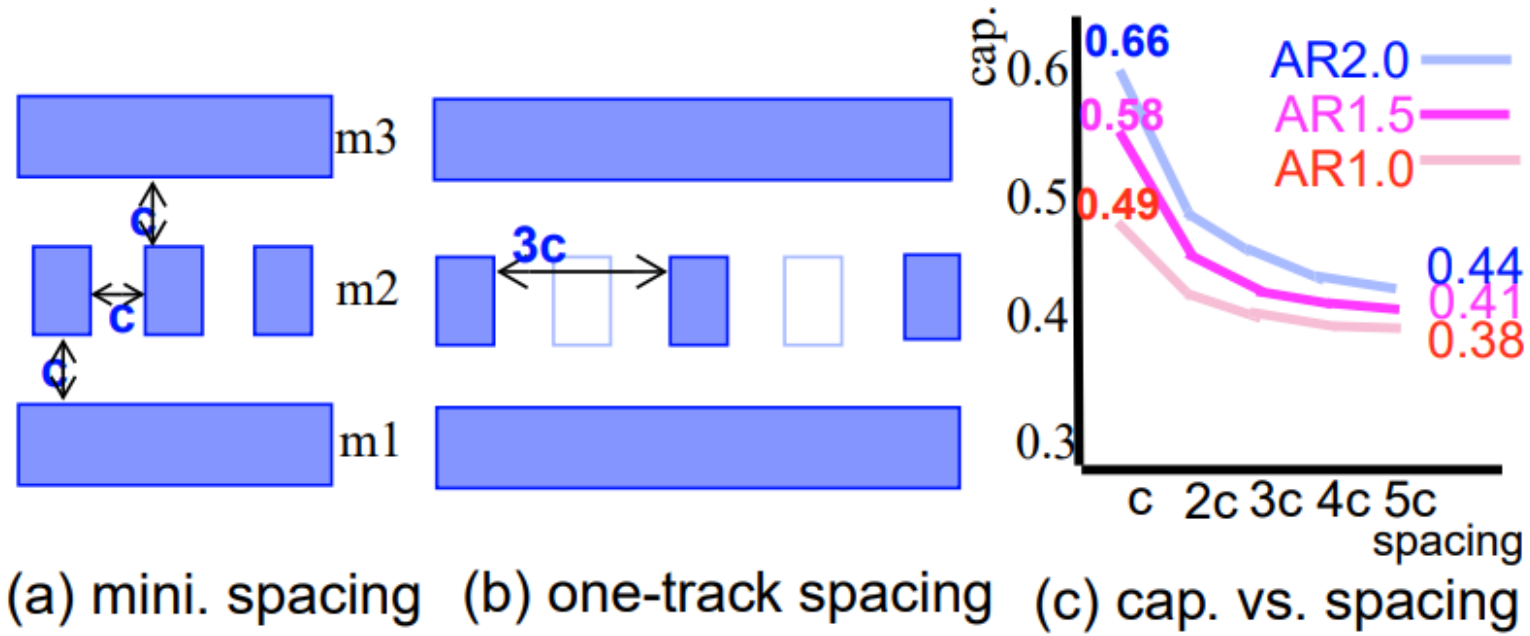


Figure 2: Coupling capacitance vs. wire spacing (c: minimum wire width)

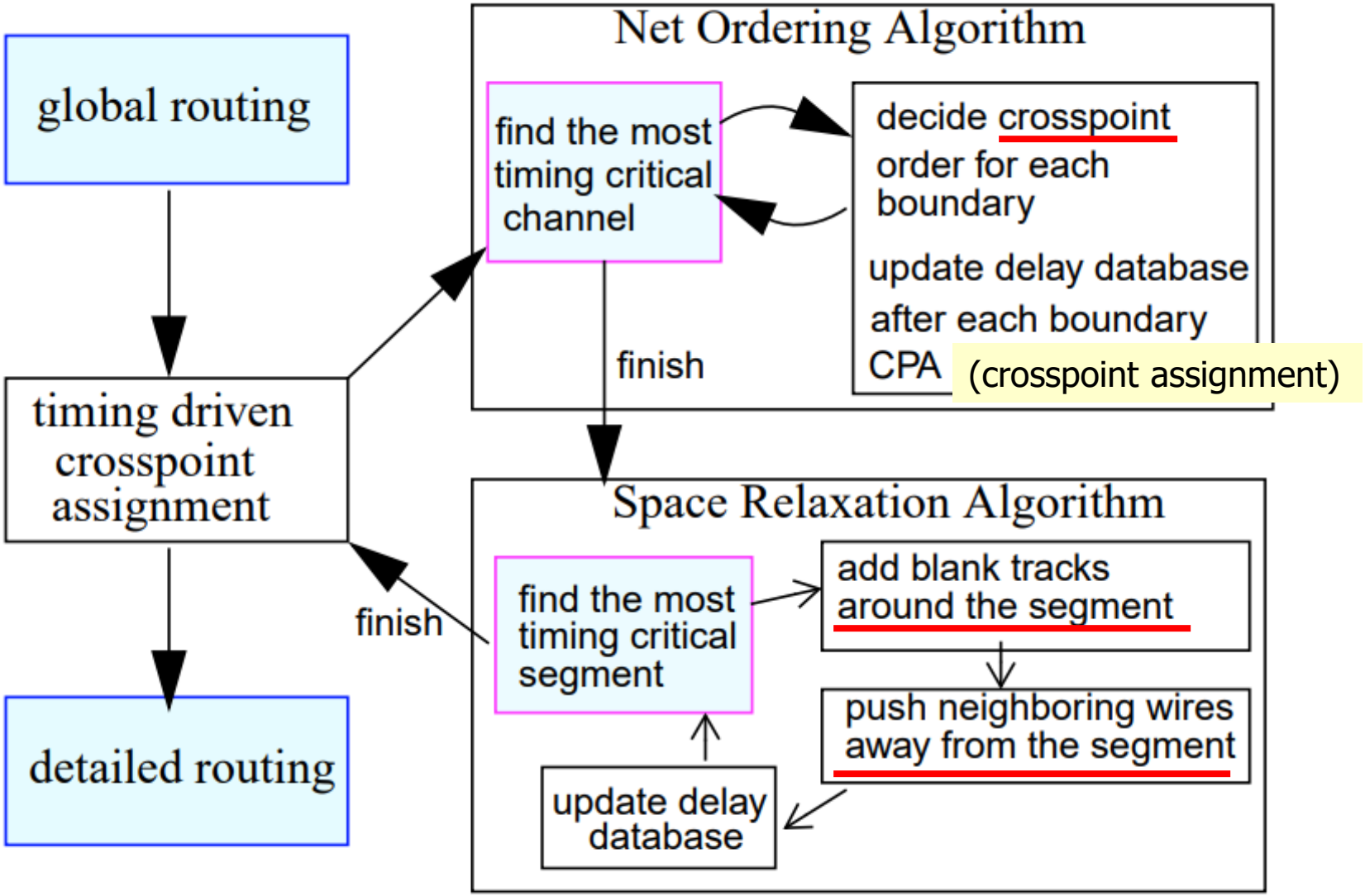


Figure 1: Flow of the algorithms

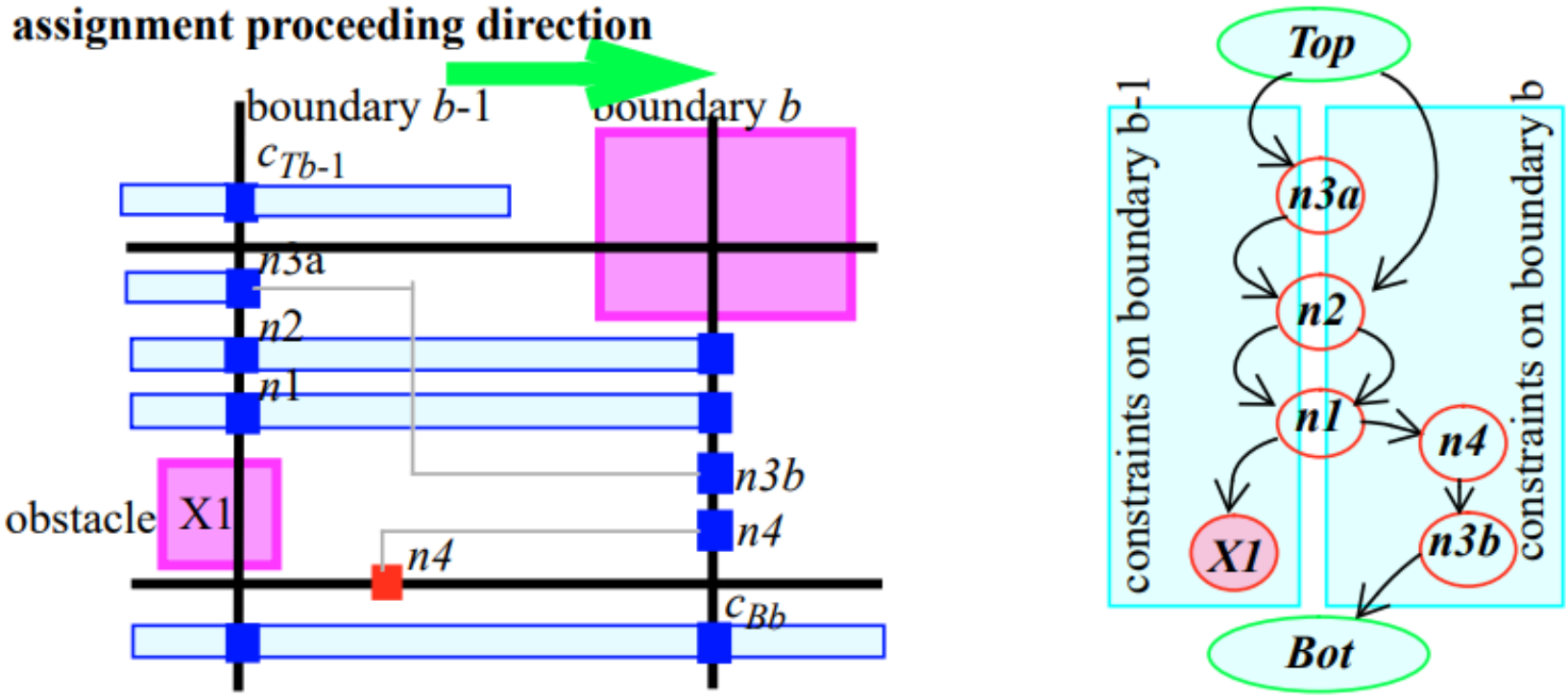


Figure 3: Crosspoint assignment

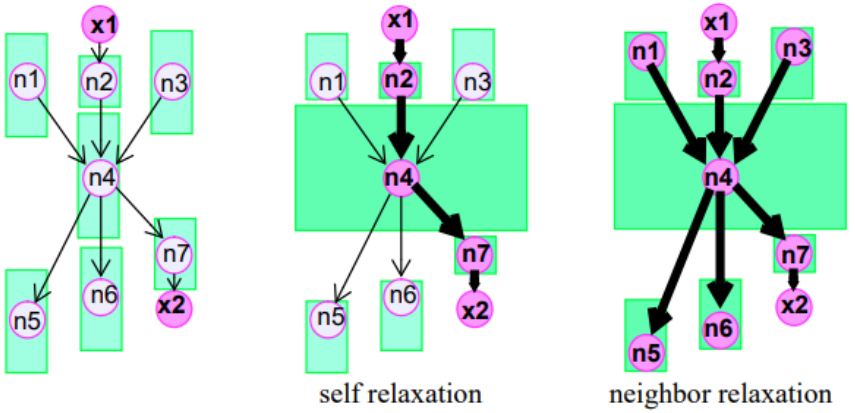
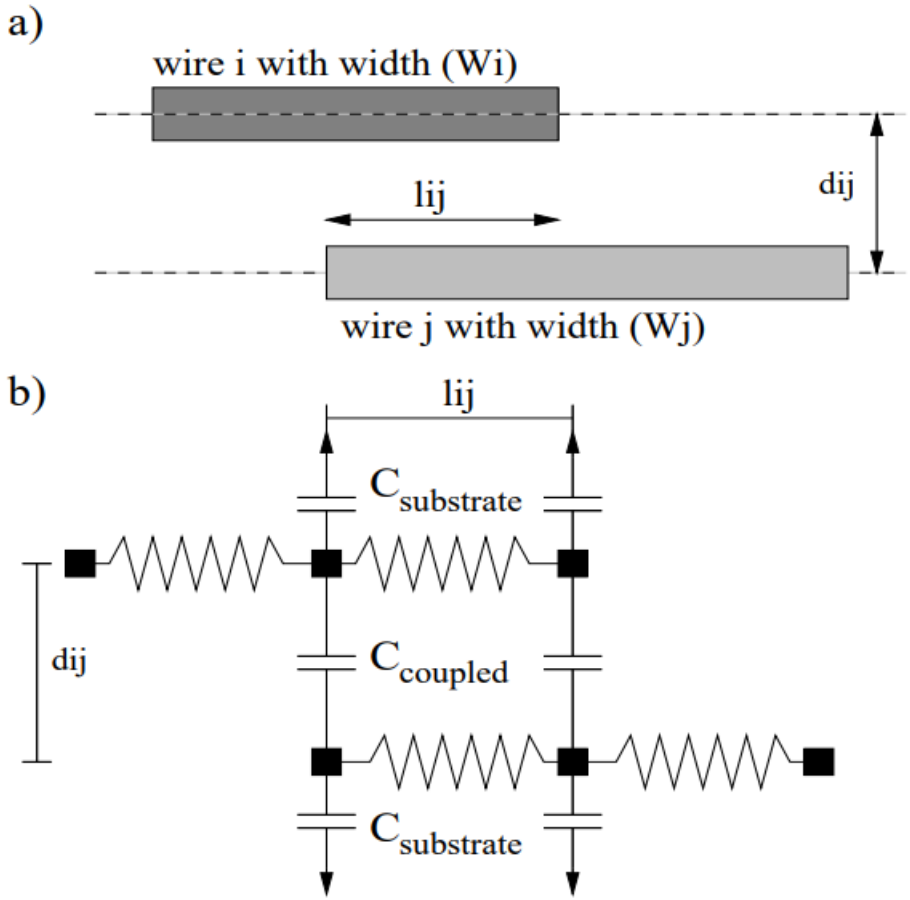


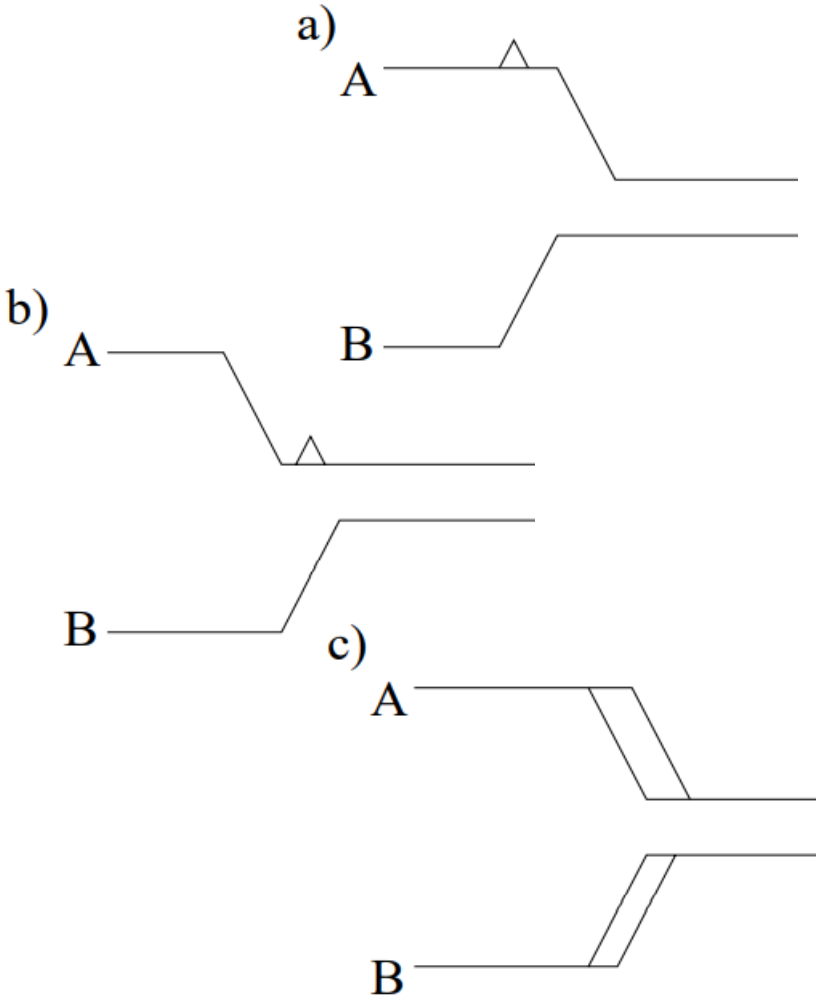
Figure 4: Self relaxation of node 4 and its neighbor relaxation

Coupling Reduction (R. Kastner's MS Thesis)



Net A - sensitive line (weak driver)

Net B - noisy line (strong driver)



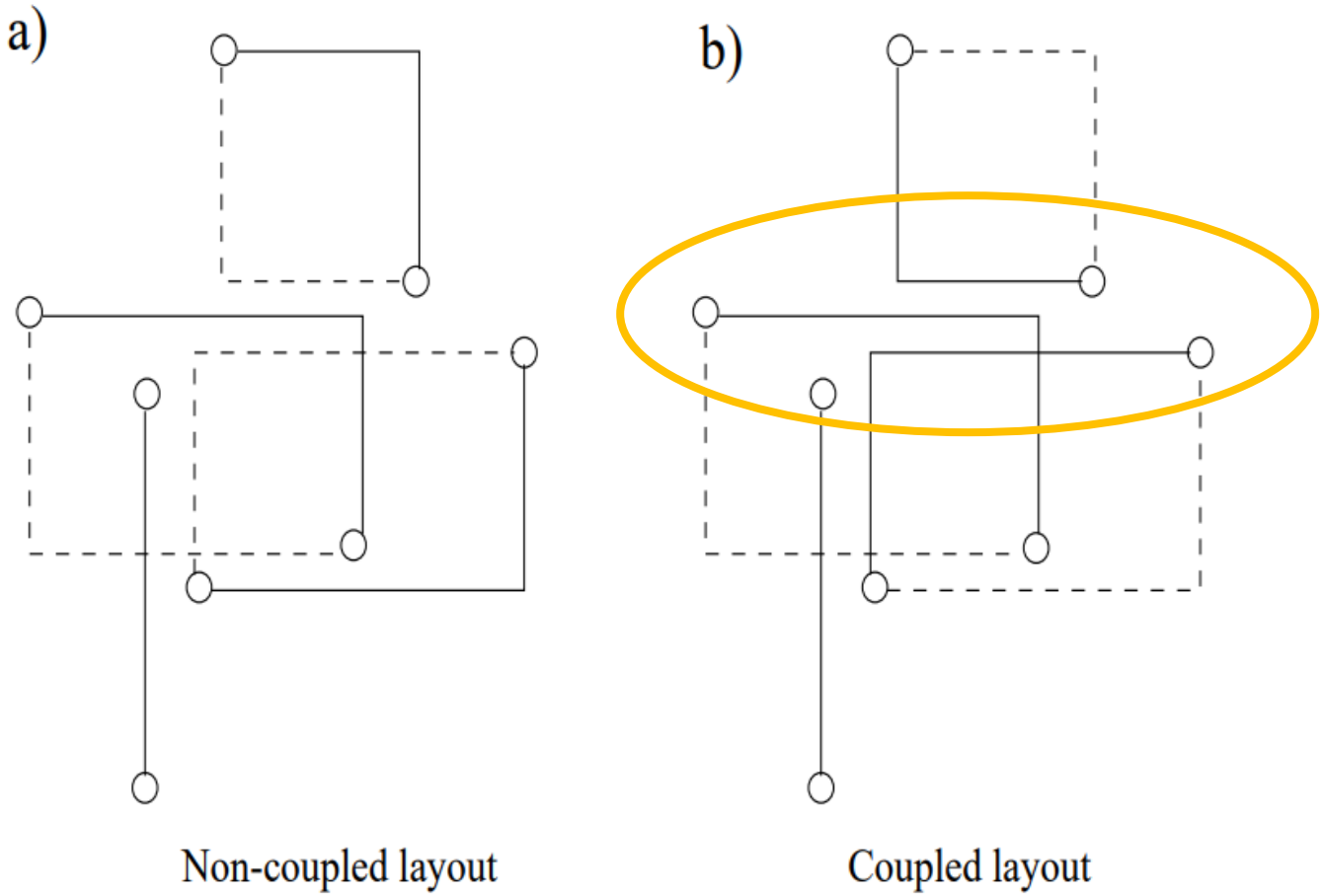


Figure 2.7: a) Coupling-free routings b) Non-coupling-free routings

What Aprisa (A Commercial Tool) Said

https://blogs.sw.siemens.com/aprisa/2022/06/28/webinar-arm-finds-success-with-siemens-aprisa-place-and-route/

This is where Aprisa enters the story. Arm began a trial of the Aprisa IC place-and-route tool from Siemens, and looked at how Aprisa resolved a few key implementation challenges, including:

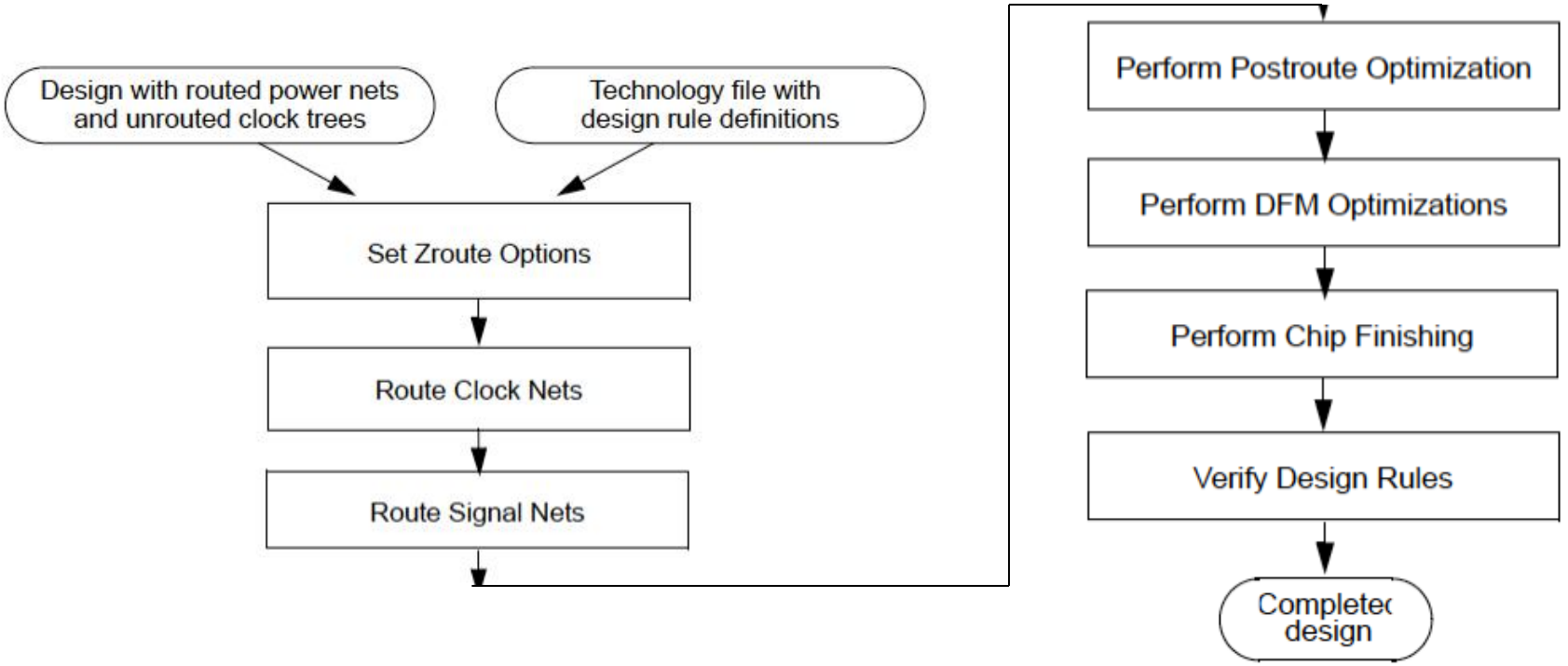
- Global register placement
- Global routing and Track Assignment to reduce SI
- Co-optimizing buffer/layer/NDR

Detailed Routing

- It is doing design rule checking and fixing.
- Rip-up-and-reroute requires many expertise. It takes time to build a good detailed router

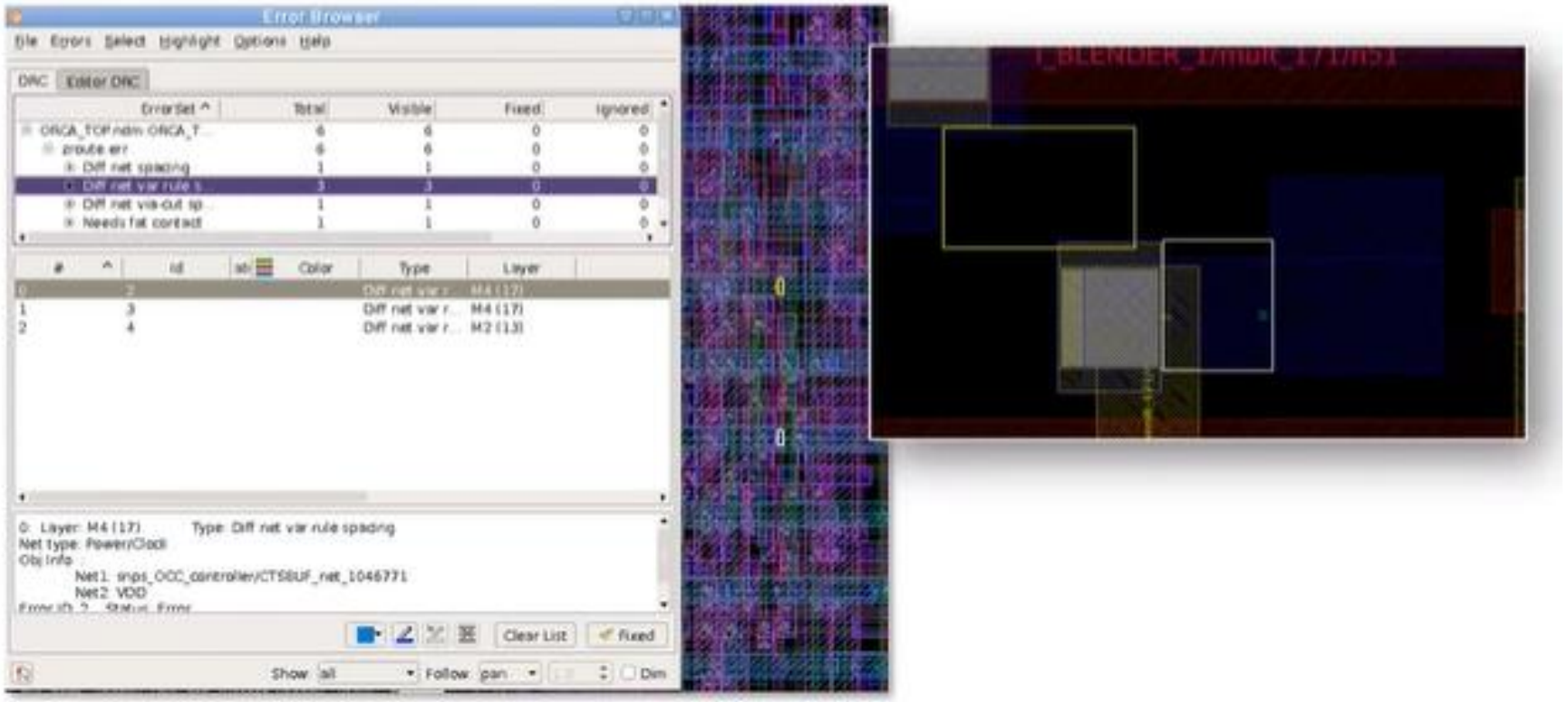
https://inst.eecs.berkeley.edu/~ee290c/sp17/lectures/Lecture26.pdf

Basic Zroute Flow (Synopsys flagship router)

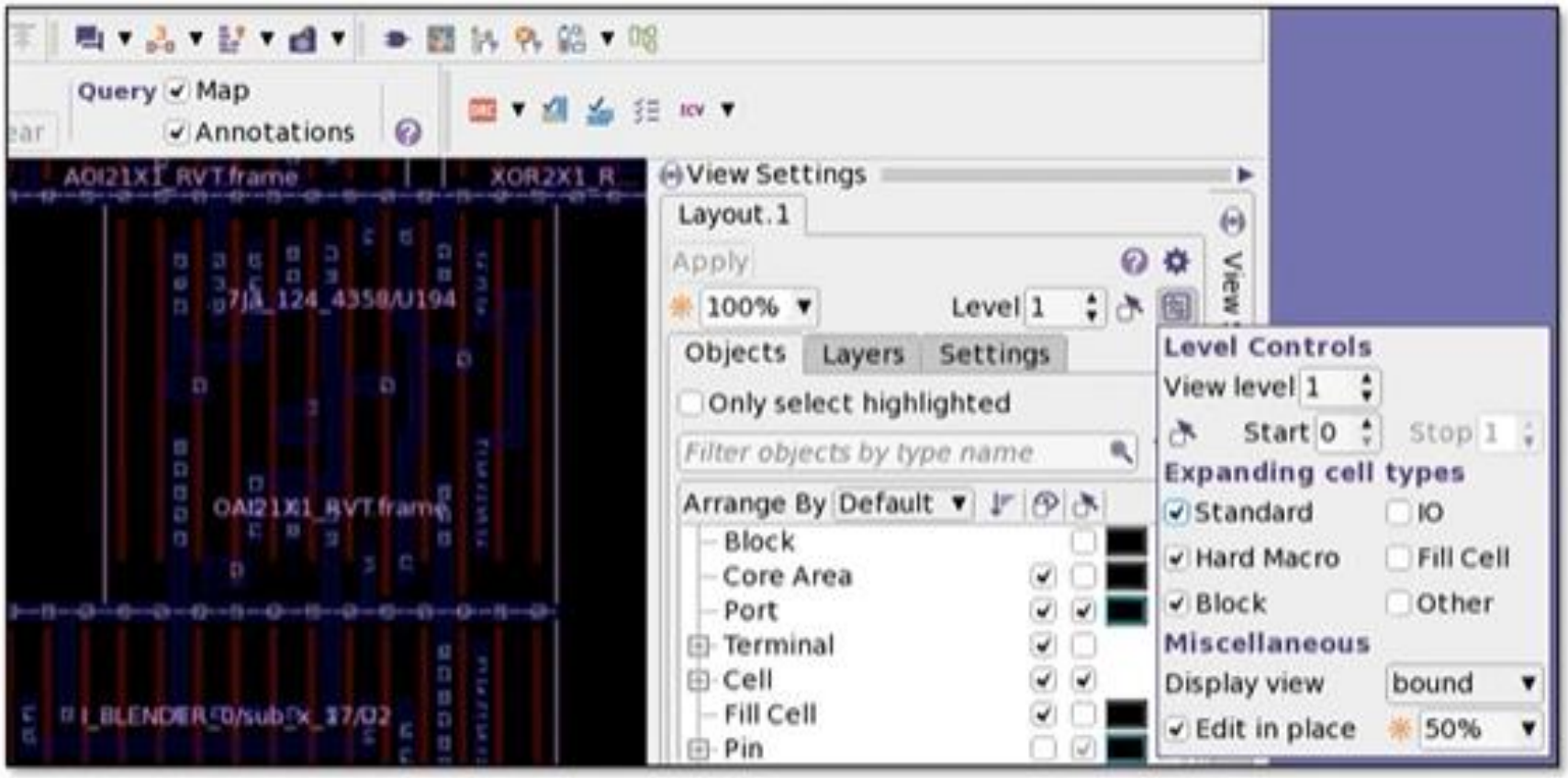


Detailed Routing

- Browsing router DRC errors



In Debugging Routing DRC Errors



Latest Attempts In School For Detailed Routing

https://github.com/The-OpenROAD-Project/TritonRoute/blob/master/README.md

TritonRoute: The Open Source Detailed Router (GitHub, Prof. Khang)

https://github.com/cuhk-eda/dr-cu

Dr. CU

Dr. CU is a VLSI detailed routing tool developed by the research team supervised by Prof. Evangeline F.Y. Young in The Chinese University of Hong Kong (CUHK). Different from global routing, detailed routing takes care of many detailed design rules and is performed on a significantly larger routing grid graph. In advanced technology nodes, it becomes the most complicated and time-consuming stage in the VLSI physical design flow. To tackle the challenges, we design and implement several efficient and effective data structures and algorithms under a holistic framework:

- A set of two-level sparse data structures
- An optimal correct-by-construction path search
- An efficient bulk synchronous parallel scheme
- ...

Redundant Vias (also called double vias) (For Yield, Reliability)

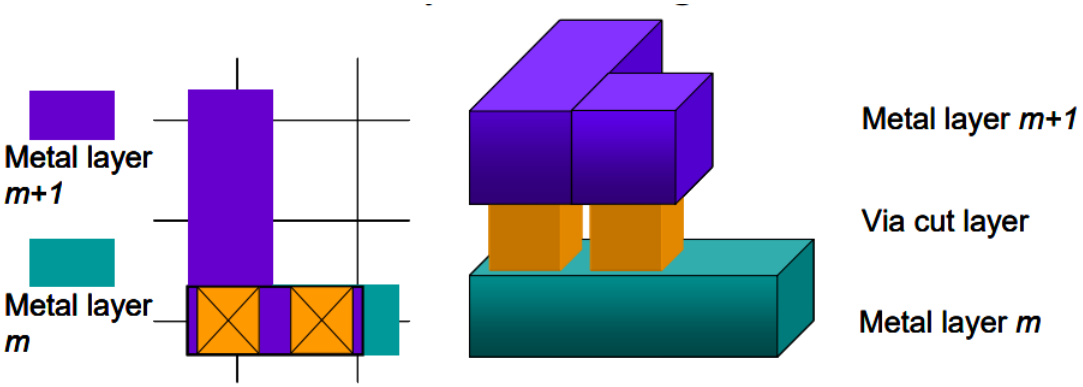


Fig. 1 Illustration for redundant via insertion.

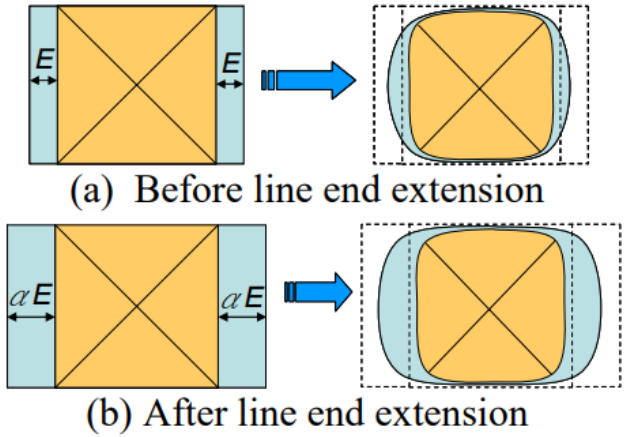


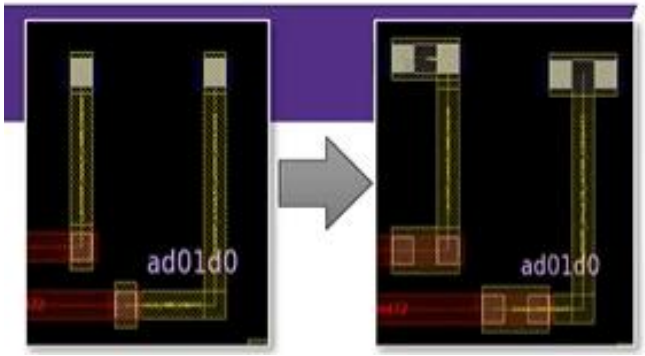
Fig. 2. Via pattern distortions.

After inserting redundant vias into a design, the maximum via density rule should be re-verified.

Line end extension is to broaden the extension distance and thus can ease the cut misalignment problem caused by line-end shortening, as shown in Fig. 2(b)

Latest Status Of Redundant Vias

- Redundant via insertion may not be used for advanced nodes in the digital SoC flow; but I still saw this in routing steps:



- Analog layout flow may still use it since it can have wider wires

About timing delay

https://www.slideserve.com/mead/layer-assignment-algorithm-for-rlc-crosstalk-minimization

Layer Assignment Algorithm for RLC Crosstalk Minimization

Bin Liu, Yici Cai, Qiang Zhou, Xianlong Hong

Tsinghua University

2014

Motivations

- Layer assignment is an ideal step to address crosstalk.
  - Global routing: flexible but inaccurate
  - Detailed routing: accurate but lacks flexibility
  - As an intermediate stage, layer assignment combines accuracy and flexibility

Different Kind Of Routing – Flip Chip, InFO

https://dl.acm.org/doi/pdf/10.1145/1837274.1837298?casa_token=XmMdZ8q3YUMAAAAA:zfcMVUQxhO9_fqu57Y8XKKdHYnY7pdbcnVWHRCms9HuZtA3JCsJOShtkcZ1lj24bKUTDtHabxJKZIA

Global Routing and Track Assignment for Flip-Chip Designs

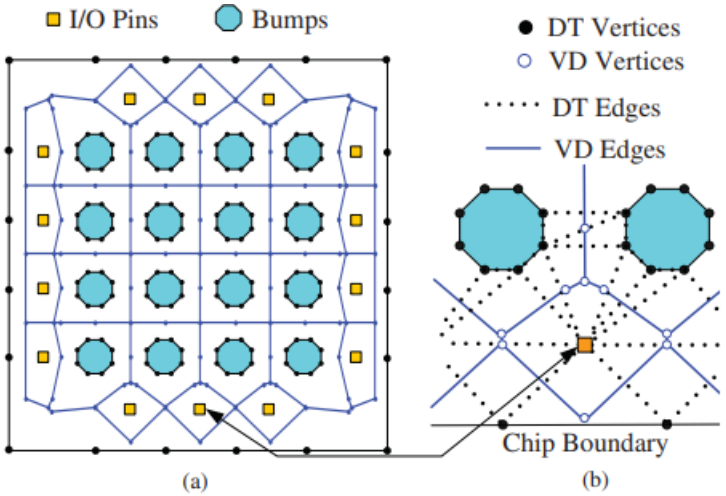
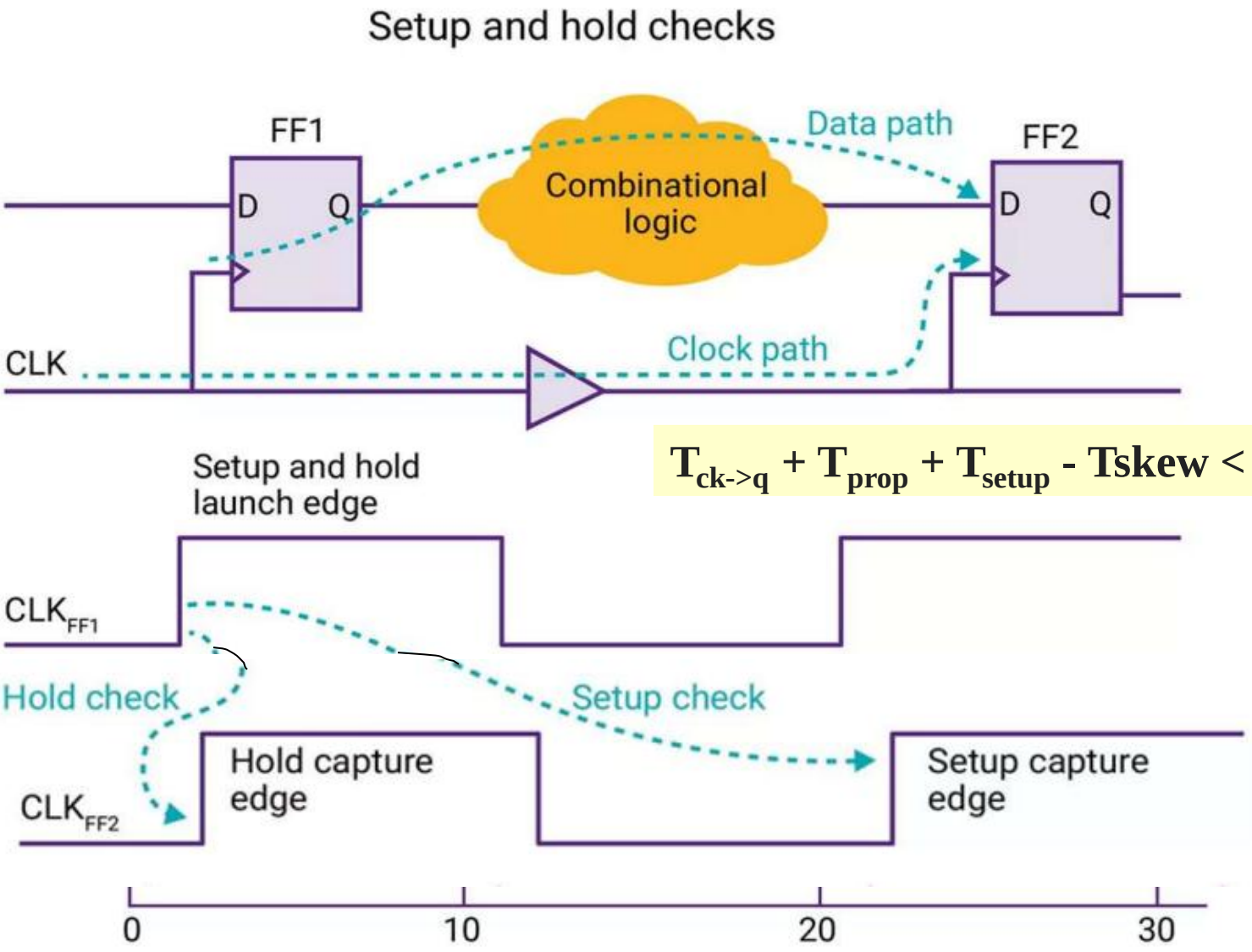


Figure 4: (a) Global routing channel graph of Figure 2(a). (b) Partial DT and VD for global routing.

Clock Routing

Setup Time & Hold Time Check

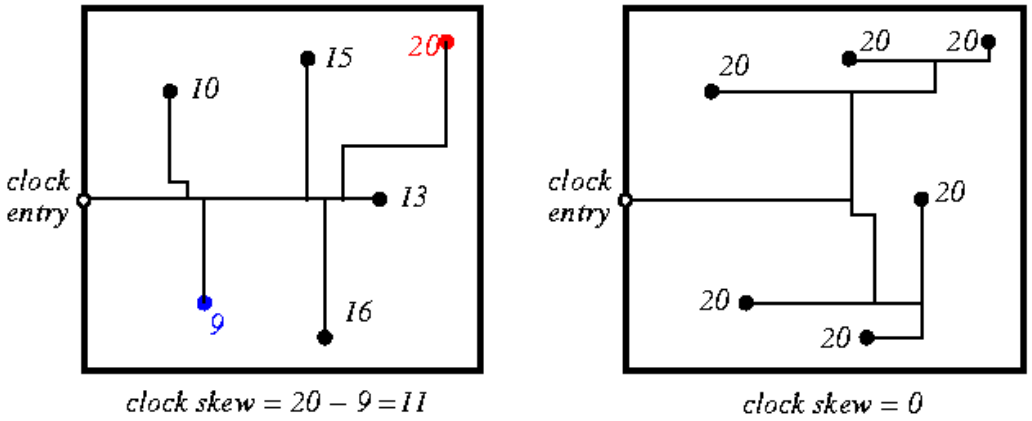


$$T_{ck->q} + T_{prop} + T_{setup} - Tskew < T_{period}$$

$$T_{ck->q} + T_{prop} > T_{hold} + T_{skew}$$

The Clock Routing Problem (CRP)

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



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

Clock Routing Problem

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

- Pathlength-based approaches
  1. *H*-tree: Dhar, Franklin, Wang, ICCD-84; Fisher & Kung, 1982. Geometric matching: Cong, Kahng, Robins, DAC-91.
- RC-delay based approaches:
  1. Exact zero skew: Tasy, ICCAD-91.
  2. Lagrangian relaxation: Chen, Chang, Wong, DAC-96.

Clock Tree Routing

Several algorithms exist that are trying to achieve this goal (to minimize the skew).

- H-Tree
- X-Tree
- Method of Mean and Median
- Geometric Matching Algorithms
- Zero skew clock routing

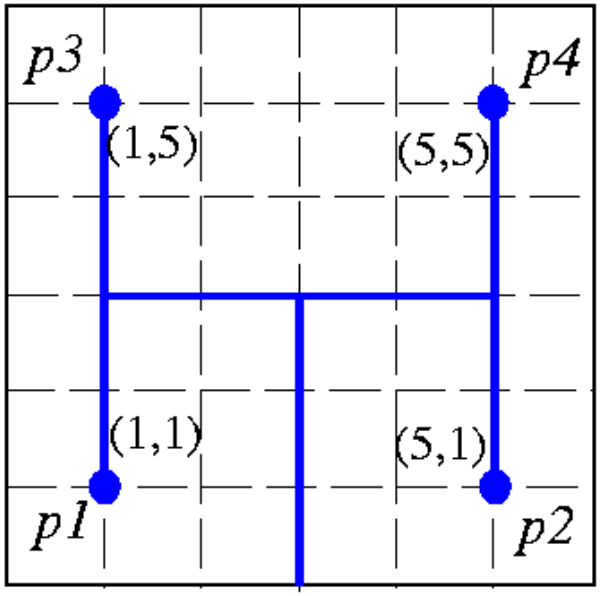
The first to fourth algorithm techniques are trying to make minimize the length and the last one is to use the actual interconnect delay in making the skew is zero.

https://vlsibegin.blogspot.com/p/cts-clock-tree-synthesis.html (worthwhile)

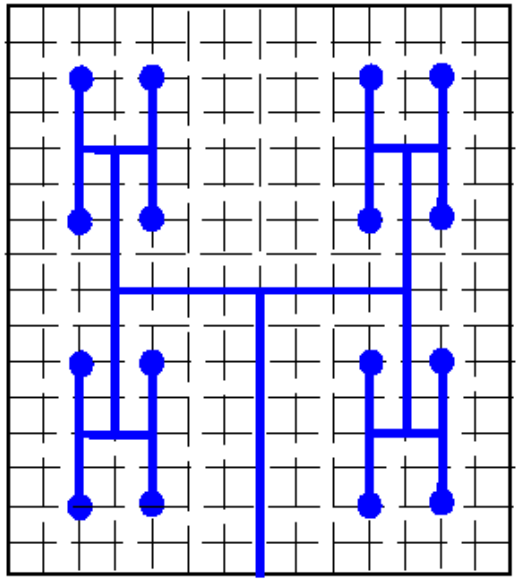
https://www.physicaldesign4u.com/2020/03/clock-tree-routing-algorithms.html

H-Tree Based Algorithm

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



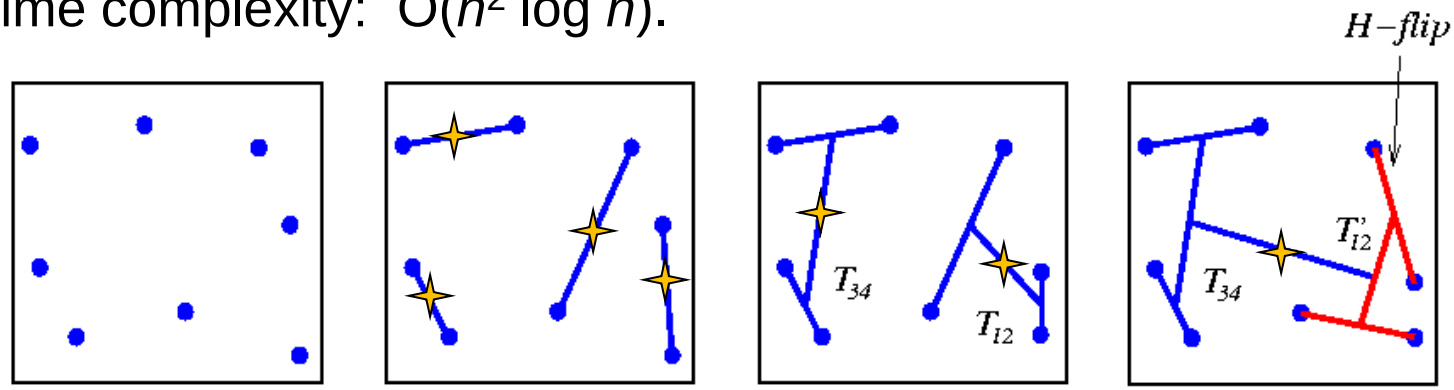
H–tree over 4 points



H–tree over 16 points

The Geometric Matching Algorithm

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



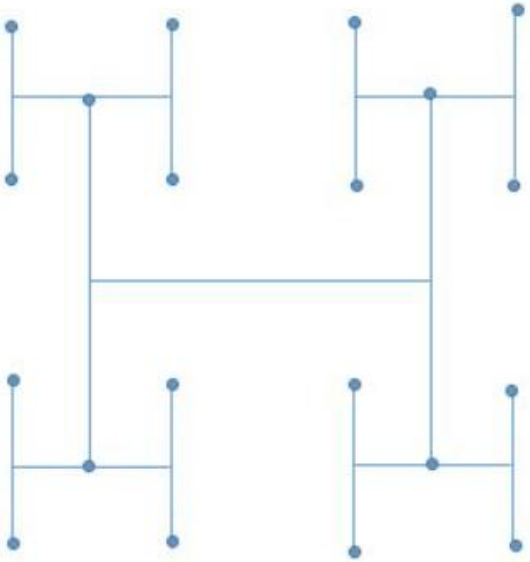


fig: H tree with 16 sink points

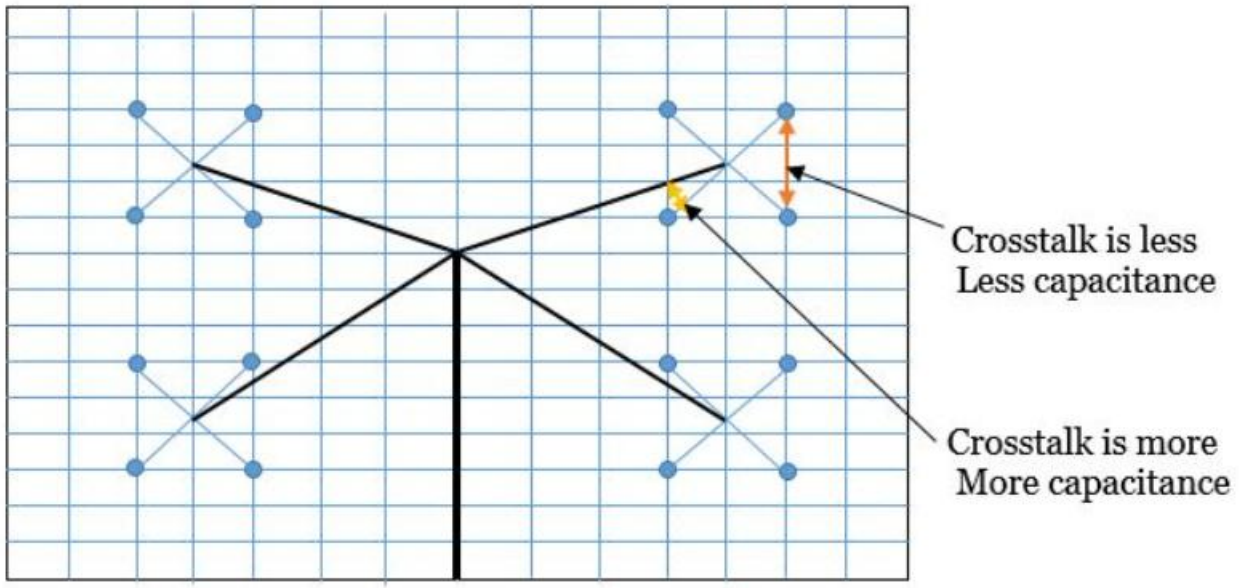


Fig: X tree with 16 points

Method of Mean and Medium:
A **top-down** approach

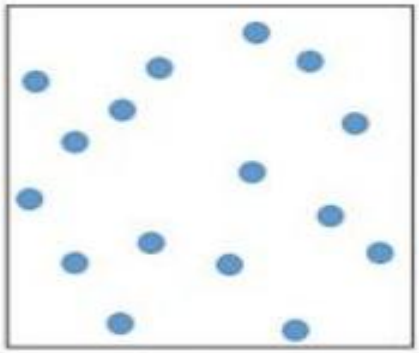


Fig a

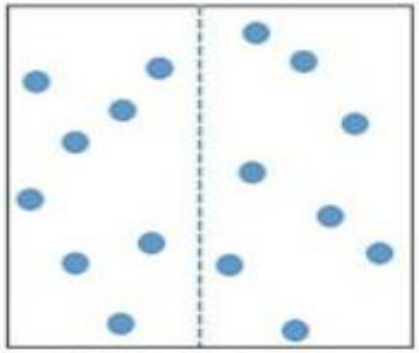


fig b: find the center of mass and then partition

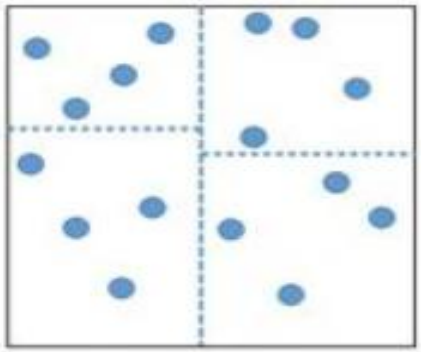


Fig c: find the center of mass for the left and right Subset

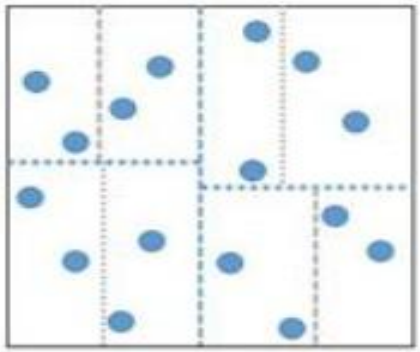


fig d: find the center of mass for up and down from the left and right subset

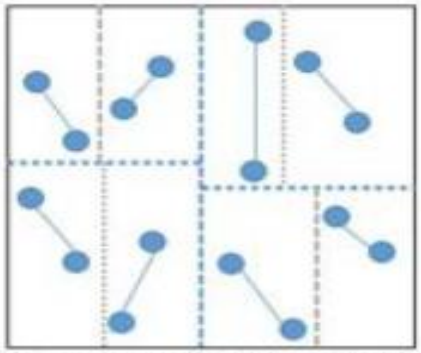


Fig e: connect all the subset in their respective region

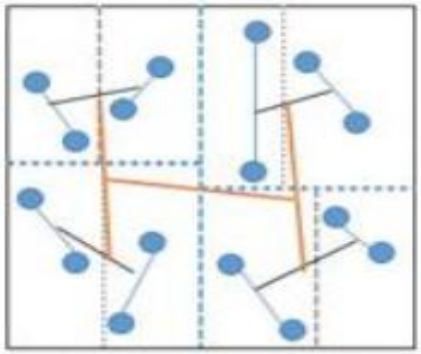


fig d: finally connect all the centers of subset

fig: MMM algorithm

Recursive geometric Matching Algorithm (RGM)

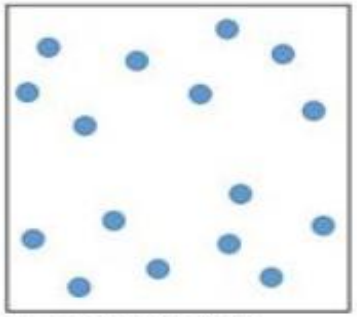


Fig a: set of n sinks

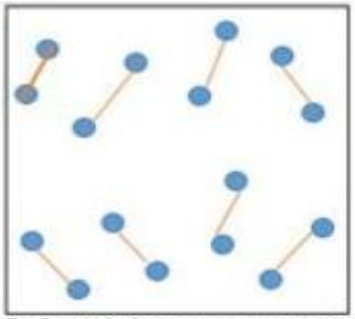


fig b: minimum cost geometric matching

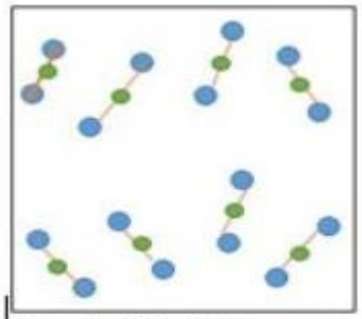


Fig c: find the balance or Tapping points (the point that Achieve zero skew in the Sub tree not always a mid-point)

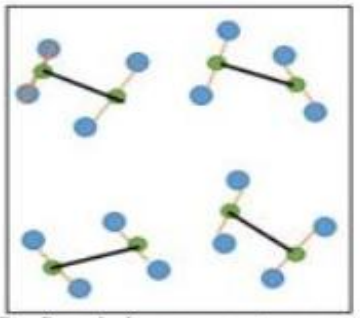


fig d: minimum cost geometric matching

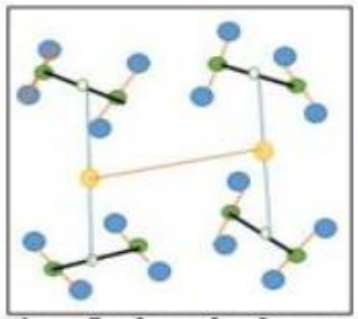


Fig e: final result after recursively Performing RGM on each subset

This **bottom-up** approach gives a better result than a top-down approach.

Fig:- RGM algorithm

Exact Zero Skew Algorithm

- Tasy, "Exact zero skew algorithm," ICCAD-91 (Prof. Ren-Song Tsay)
- (review Elmore delay)
- To ensure the delay from the **tapping point** to leaf nodes of subtrees T_1 and T_2 being equal, it requires that

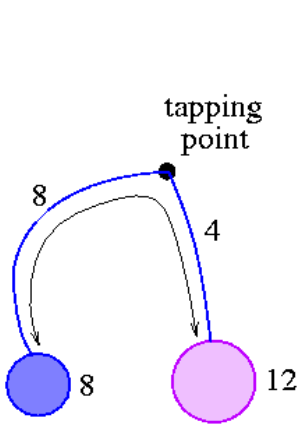
  $r_1 (c_1/2 + C_1) + t_1 = r_2 (c_2/2 + C_2) + t_2$.

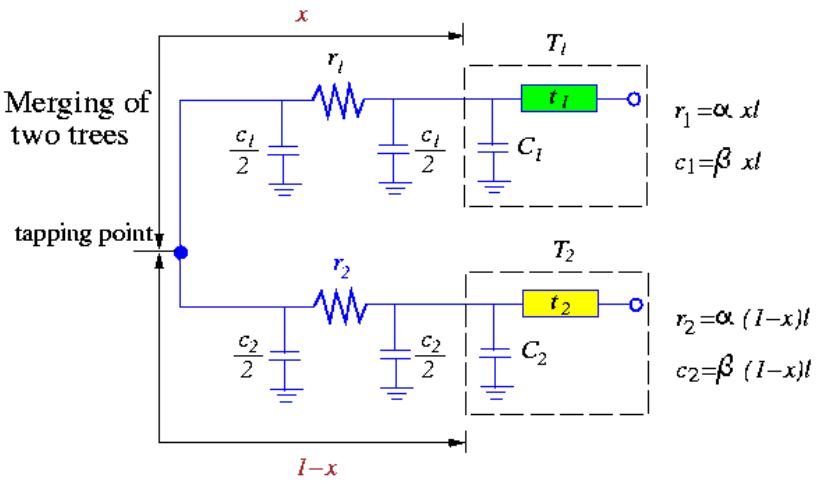
- Solving the above equation, we have

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

  where α and β are the per unit values of resistance and capacitance, l the length of the interconnecting wire,

  $r_1 = \alpha x l$, $c_1 = \beta x l$, $r_2 = \alpha(1 - x)\, l$, $c_2 = \beta(1 - x) l$.

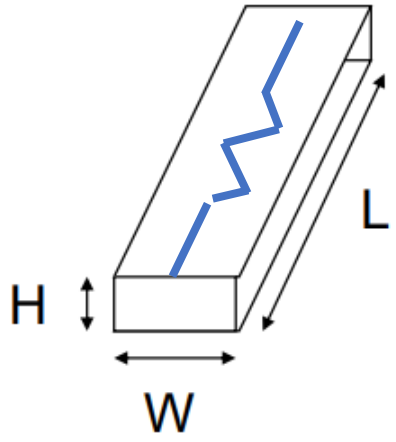




Elmore Delay & Skew

Elmore Delay (For Wiring)

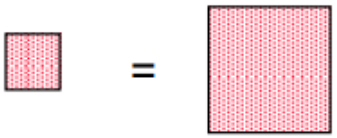
Wire Resistance



$$R = \frac{\rho L}{A} = \frac{\rho}{H} \frac{L}{W}$$

Sheet Resistance $R_{\square}$

$R_{1\square} = R_{2\square}$

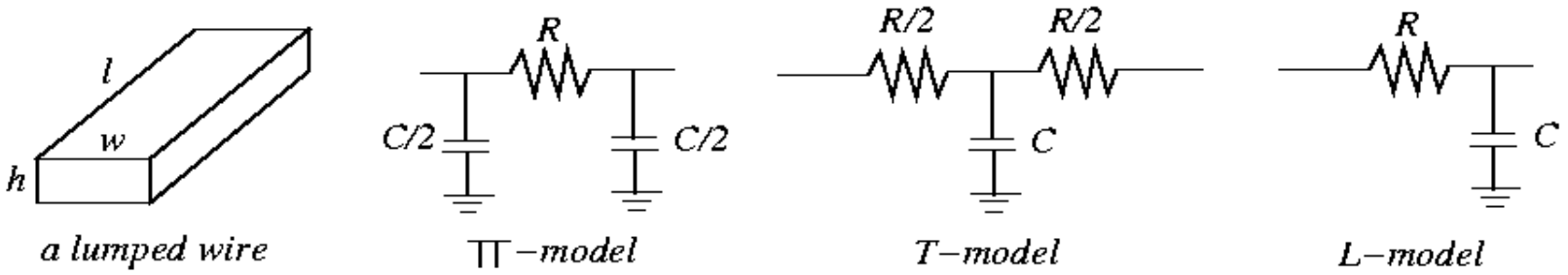


Material	ρ(Ω-m)
Silver (Ag)	1.6×10^{-8}
Copper (Cu)	1.7×10^{-8}
Gold (Au)	2.2×10^{-8}
Aluminum (Al)	2.7×10^{-8}
Tungsten (W)	5.5×10^{-8}

Material	Sheet Res. (Ω/□)
n, p well diffusion	1000 to 1500
n+, p+ diffusion	50 to 150
n+, p+ diffusion with silicide	3 to 5
polysilicon	150 to 200
polysilicon with silicide	4 to 5
Aluminum	0.05 to 0.1

Wire Models

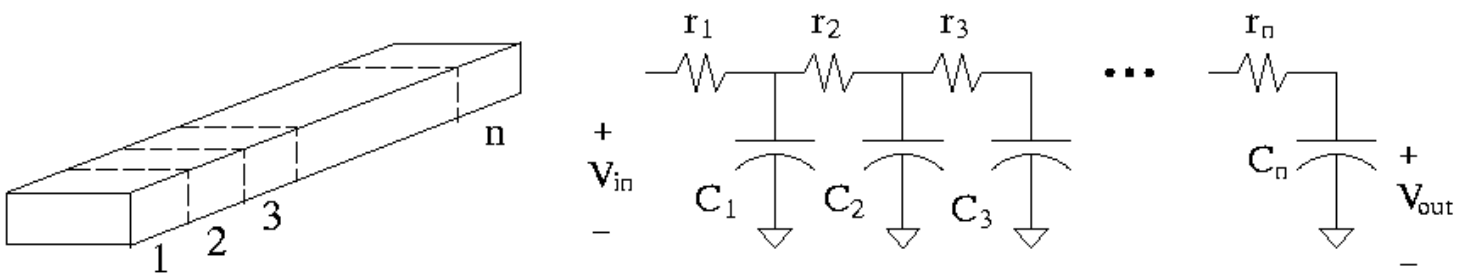
- Lumped circuit approximations for distributed RC lines: **π-model** (most popular), *T*-model, *L*-model.



Elmore Delay: Nonlinear Delay Model

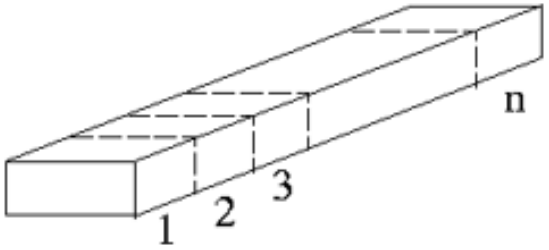
- Parasitic resistance and capacitance start to dominate delay in deep submicron wires.
- Resistor r_i must charge all downstream capacitors.
- **Elmore delay:** Delay can be approximated as sum of sections: resistance ✕ downstream capacitance.

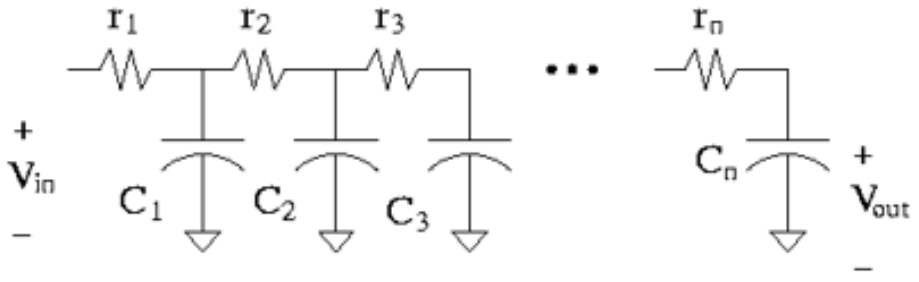
$$\delta = \sum_{i=1}^{n} \left(r_i \sum_{k=i}^{n} c_k \right) = \sum_{i=1}^{n} r(n-i+1)c = \frac{n(n+1)}{2} rc.$$



- Delay (delta) grows as **square** of wire length.

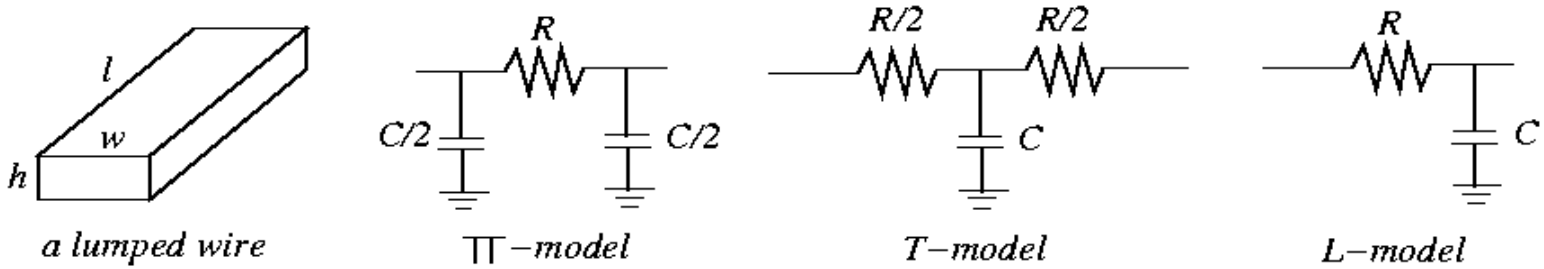
r1 * (c1+c2+…+cn) +
r2 * (c2+…+cn) +
r3 * (c3+...+cn)
= r1 * nc +
r2 * (n-1)c +
r3 * (n-2)c
= rc * (n + n-1 + n-2 +..)





Wire Models

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

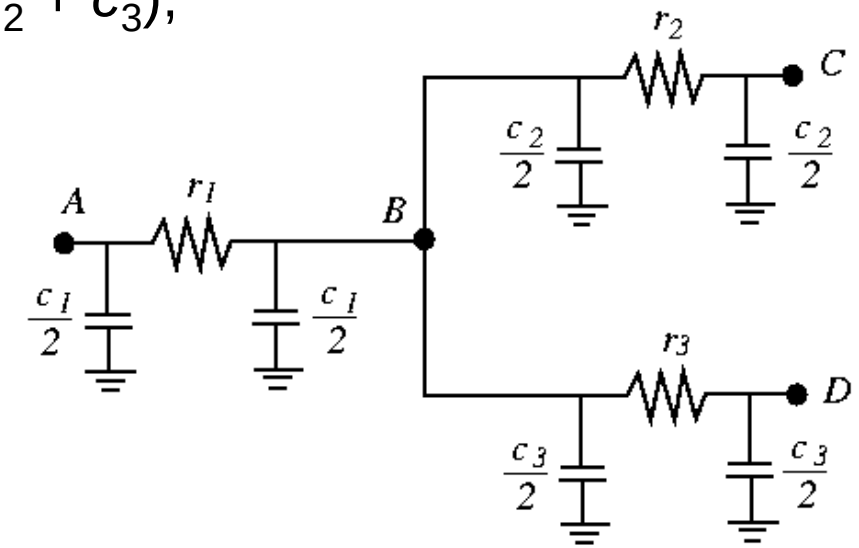


- π-model: If no capacitive loads for *C* and *D*,

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

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

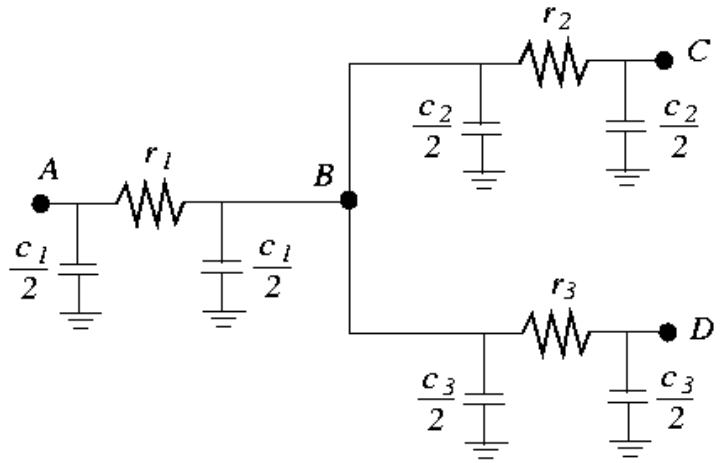
  B to *D*: $\delta_{BD} = r_3 (c_3/2)$.

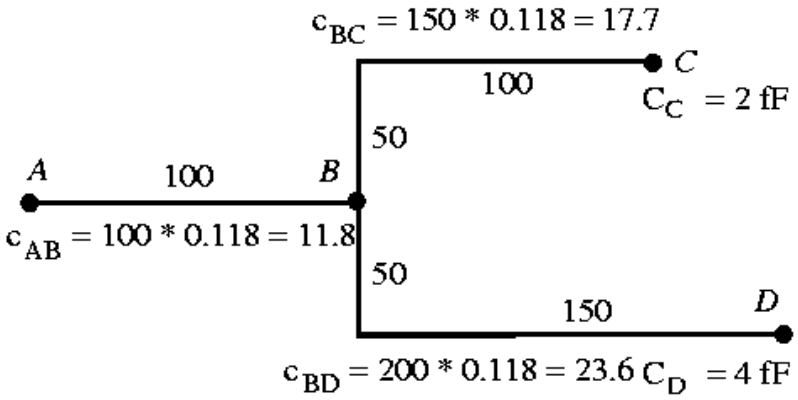


Example Elmore Delay Computation

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

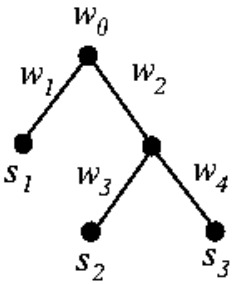
B-to-C: wire length 150um
Cbc=150um*0.118=17.7



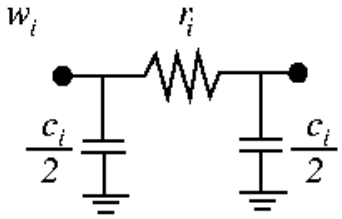


Delay Calculation for a Clock Tree

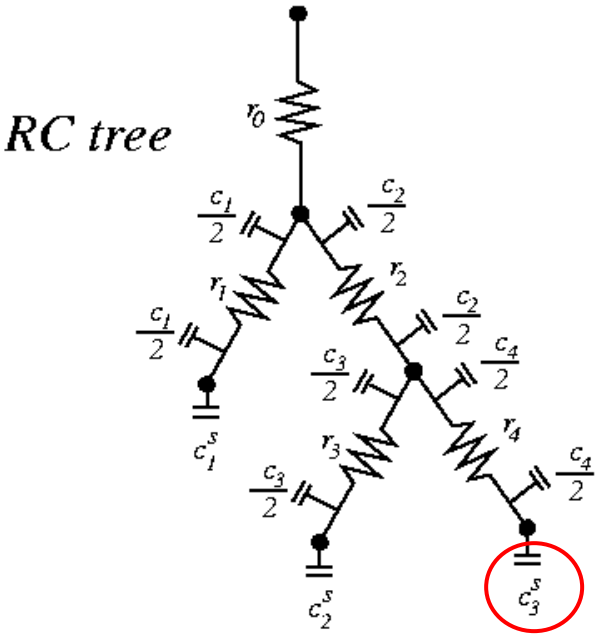
- Let T be an RC tree with points $P = \{p_1, p_2, \ldots, p_n\}$, c_i the capacitance of p_i, r_i the resistance of the edge between p_i and its immediate predecessor.
- The subtree capacitance at node i is given as $C_i = c_i + \sum_{j \in Si} C_j$, where S_i is the set of all the immediate successors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



clock tree

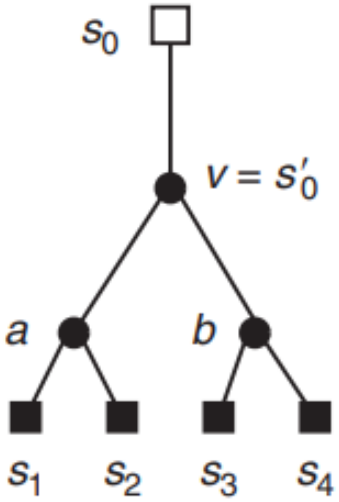


delay model

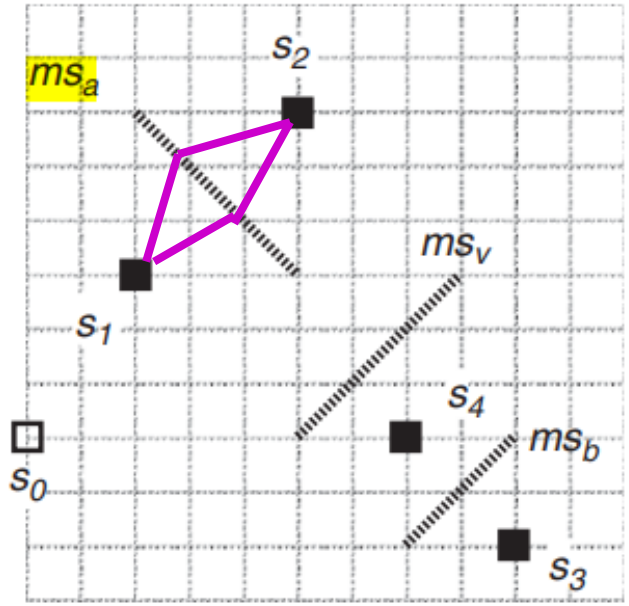


any point l_a on the line segment ms_a is equidistant from sinks s1 and s2

Page-33 in Chap-13



Topology
(a)



Bottom-up Merging phase
(b)

FIGURE 13.23

(Deferred merge embedding)

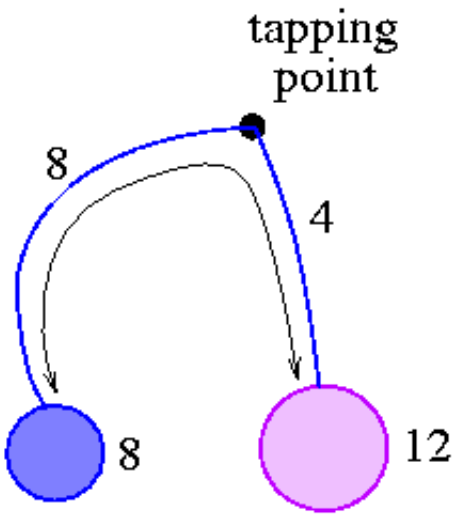
An illustration of the bottom-up phase of the DME algorithm: (a) Topology of a clock source s_0 and 4 sinks $s_{1..4}$. (b) Merging segments of internal nodes *a*, *b* and $v = s_0$.

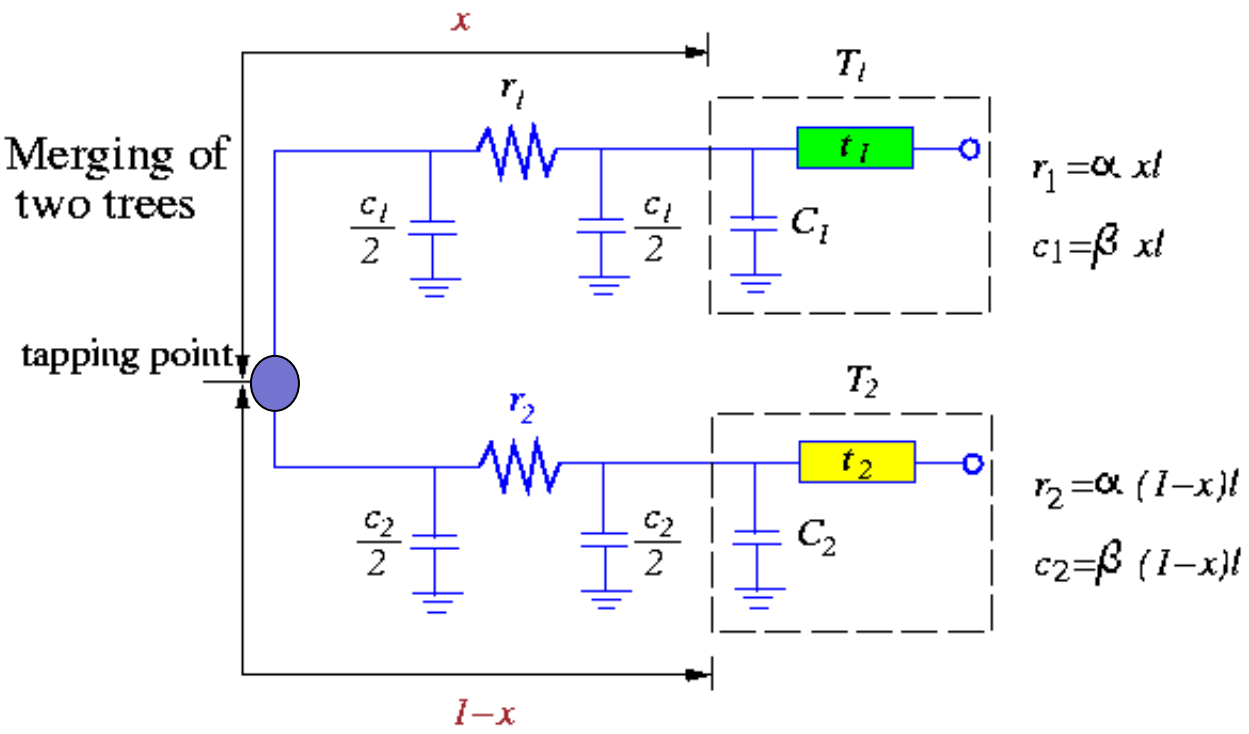
Exact Zero Skew Algorithm - 1

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

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

Using delay to decide the tapping point



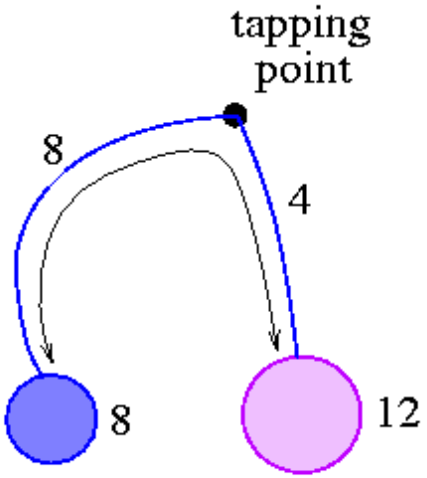


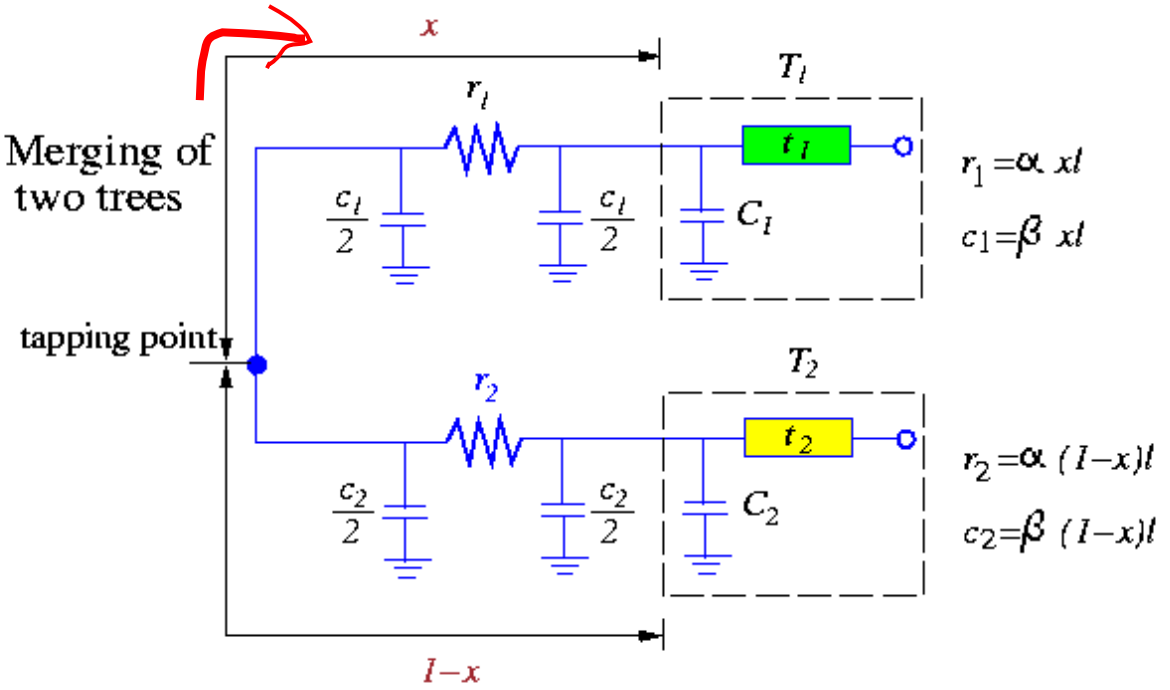
Exact Zero Skew Algorithm - 2

- $r_1 (c_1/2 + C_1) + t_1 = r_2 (c_2/2 + C_2) + t_2$.
- Solving the above equation, we have

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

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

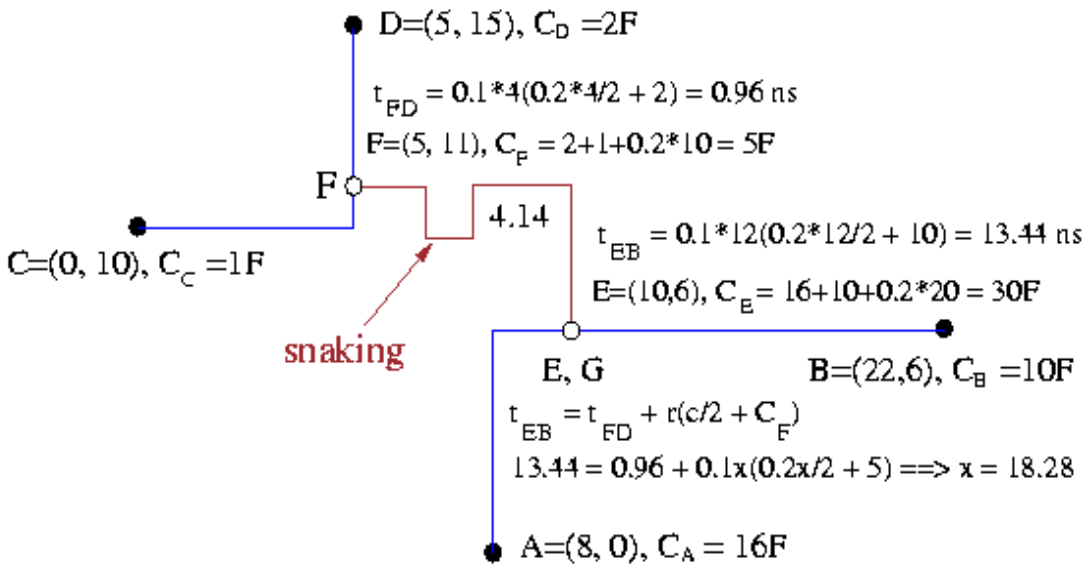




Zero-Skew Computation

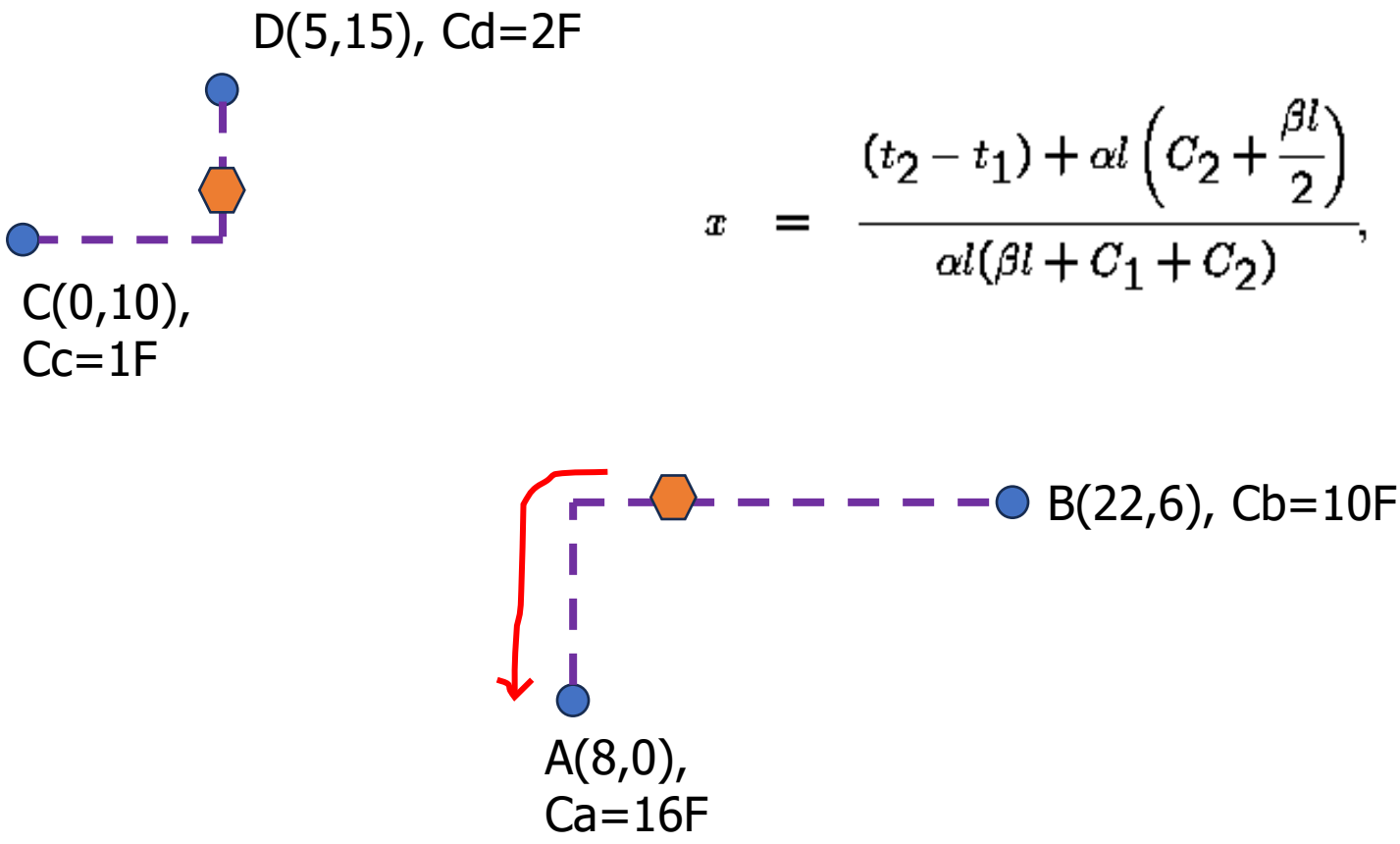
- **Balance delays:** $r_1(c_1/2 + C_1) + t_1 = r_2 (c_2/2 + C_2) + t_2$.
- **Compute tapping points:** $x = \dfrac{(t_2 - t_1) + \alpha l \left(C_2 + \frac{\beta l}{2}\right)}{\alpha l(\beta l + C_1 + C_2)}$, α (β): per unit values of resistance (capacitance); l: length of the wire; $r_1 = \beta x l$, $c_1 = \beta x l$; $r_2 = \alpha(1 - x) l$, $c_2 = \beta (1 - x) l$.
- If $x \notin [0, 1]$, need **snaking** to find tapping point (due to (t2 - t1) unbalance)
- Exp (page-38): $\alpha = 0.1\ \Omega$ /unit, $\beta = 0.2\ F$ /unit. (Find tapping points E for A and B, F for C and D, and G for E and F.)

Chap-13



L(A,B)=(22-8)+(6-0)=20
x=((0-0)+0.1*20(10+0.2*20/2)) / 0.1*20(0.2*20 + 16 + 10)
= (10+2)/(4 + 16 + 10) = 0.4
Loc(E) from length 8 (20*0.4) = (8+2, 0+6)

(F, E)→unbalance delays (0.96 vs 13.44)
L(F,E)=(10-5 + 11-6)=10
Tapping point G: snaking
18.28 – 10 = 8.28 → 4.14 zigzag



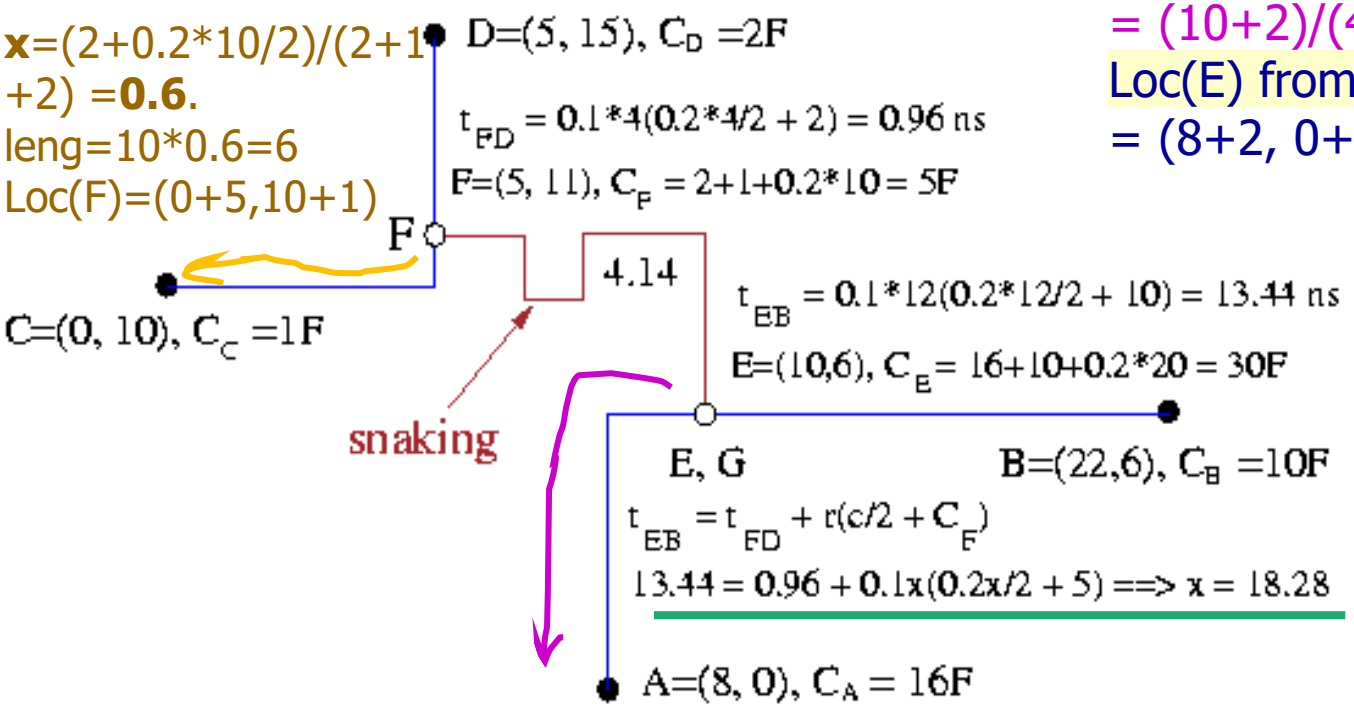
$$x = \frac{(t_2 - t_1) + \alpha l\left(C_2 + \frac{\beta l}{2}\right)}{\alpha l(\beta l + C_1 + C_2)},$$

Zero-Skew Computation

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

L(A,B)=(22-8)+(6-0)=20
Compute E:
x=((0-0)+ 0.1*20(10+0.2*20 /2)) / 0.1*20 (0.2*20 + 16 + 10)
= (10+2)/(4 + 16 + 10) = **0.4**
Loc(E) from length 8 (=20*0.4)
= (8+2, 0+6)

x=(2+0.2*10/2)/(2+1+2) =**0.6**.
leng=10*0.6=6
Loc(F)=(0+5,10+1)

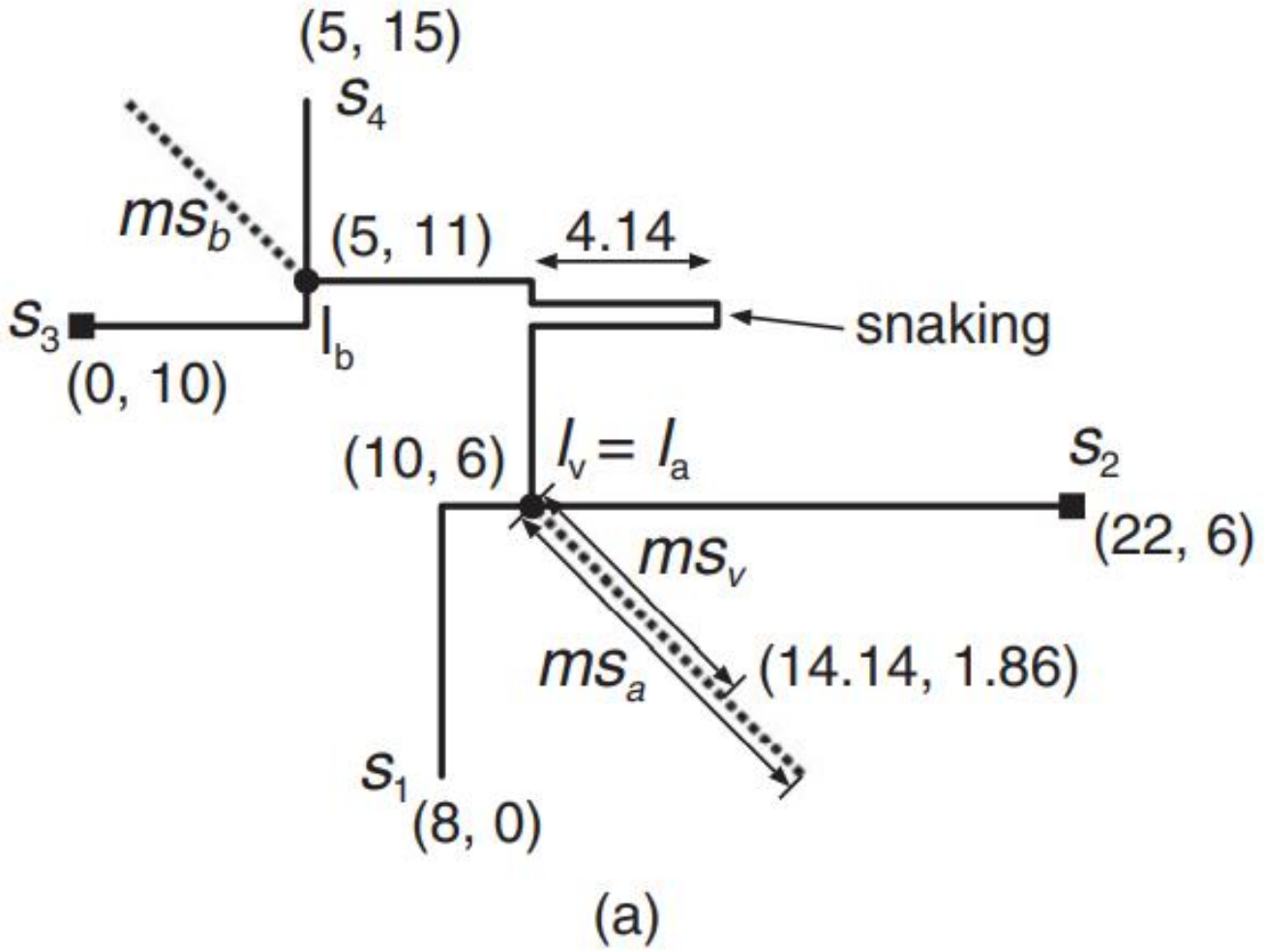


(F, E)→unbalance delays (0.96 vs 13.44)
L(F,E)=(10-5 + 11-6)=10
Tapping point G: snaking
18.28 – 10 = 8.28 → 4.14 zigzag

This x is not the one in tapping point equ.
It is simply the length from G to F

Another Solution

Page-42 in Chap-13



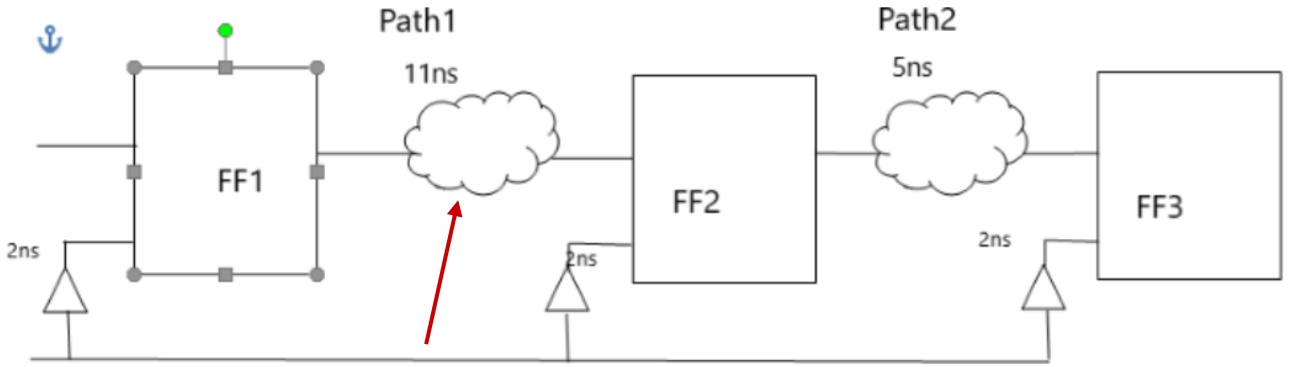
(a)

Useful Skew

https://vlsi.pro/useful-skew/

- However, **meeting zero skew in some large designs can be unnecessarily costly**. Can we leverage the clock signal in meeting timing, with acceptable margins?

Given below is a design with zero skew, but setup timing violation.



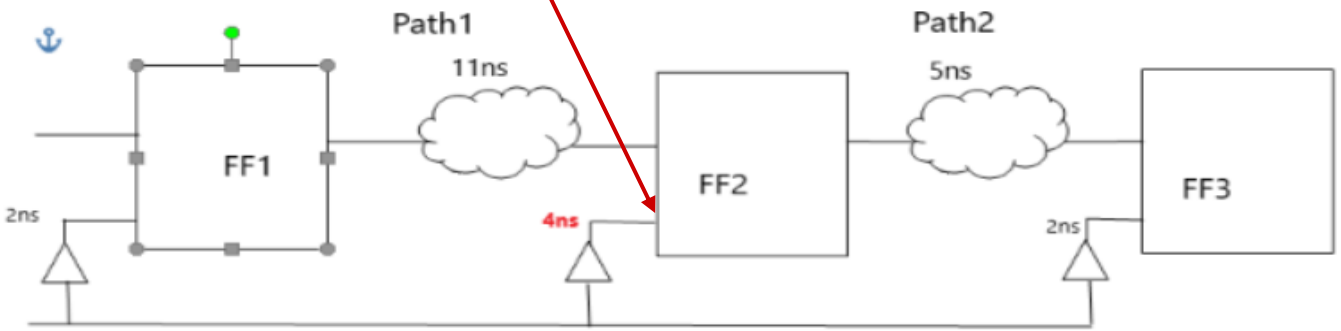
Clock period 10ns; path1 has setup violation.
Longer data path delay -> clock to FF2 can arrive late

setup timing requirement is `2ns(clock path delay to FF1/CP) + 11ns < 4ns(clock path delay to FF2/CP)+**10ns(clock period)**`
`13ns < 14ns`-> *Pass*

Path2 timing now changed to:
`4ns(clock path delay to FF2/CP)+5ns < 2ns+**10ns**`
`9ns < 12ns` --> not violated, even though margin is now less.

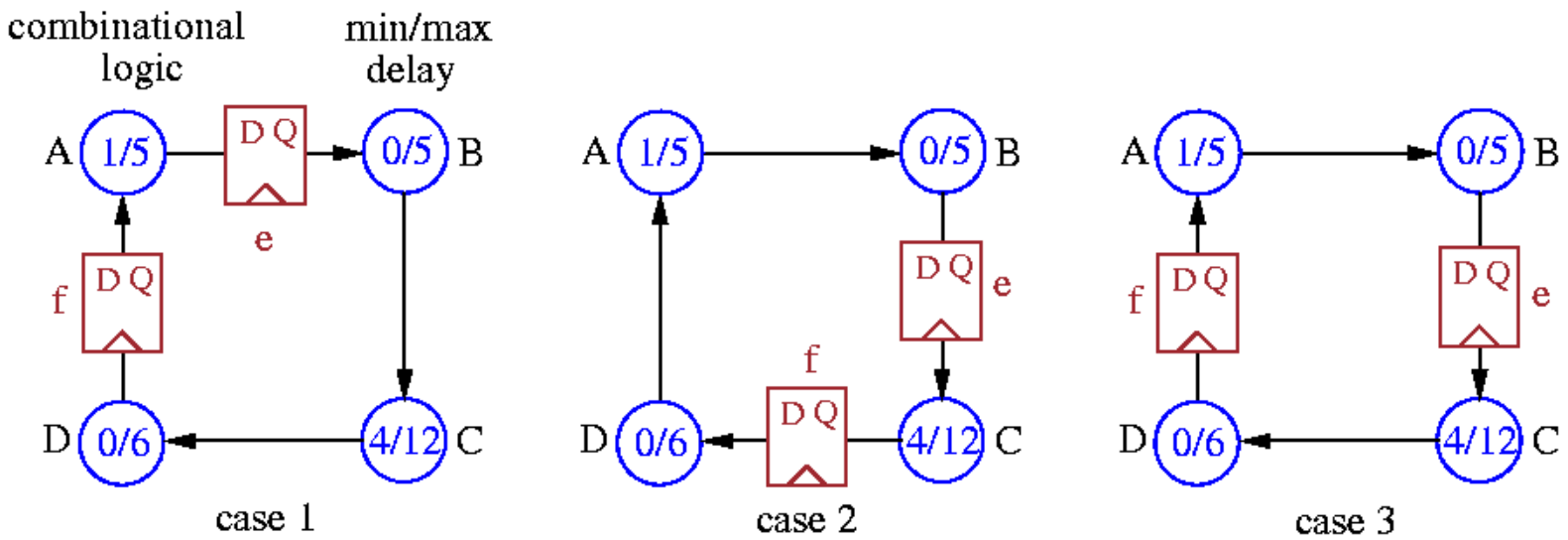
Useful Skew

Sini Mukundan May 8, 2018 2 Comments



Simultaneous Retiming and Clock Skew Scheduling

- Liu, Papaefthymiou, Friedman: Simultaneous retiming and **useful** clock skew scheduling can further reduce clock period, DAC-99 (https://dl.acm.org/doi/pdf/10.1145/309847.309919)
- Case 1
  - Zero clock skew: clock period $\phi = 23\tau$. (=5+12+6)
  - Schedule *e* 4τ earlier than *f*: Clock period $\phi = 19\tau$. (data path to f is longer – useful skew)



Retiming and Clock Skew Scheduling

- Case 1
  - Zero clock skew: clock period $\phi = 23\tau$.
  - Schedule *e* 4τ earlier than *f*: clock period $\phi = 19\tau$. (=23-4)

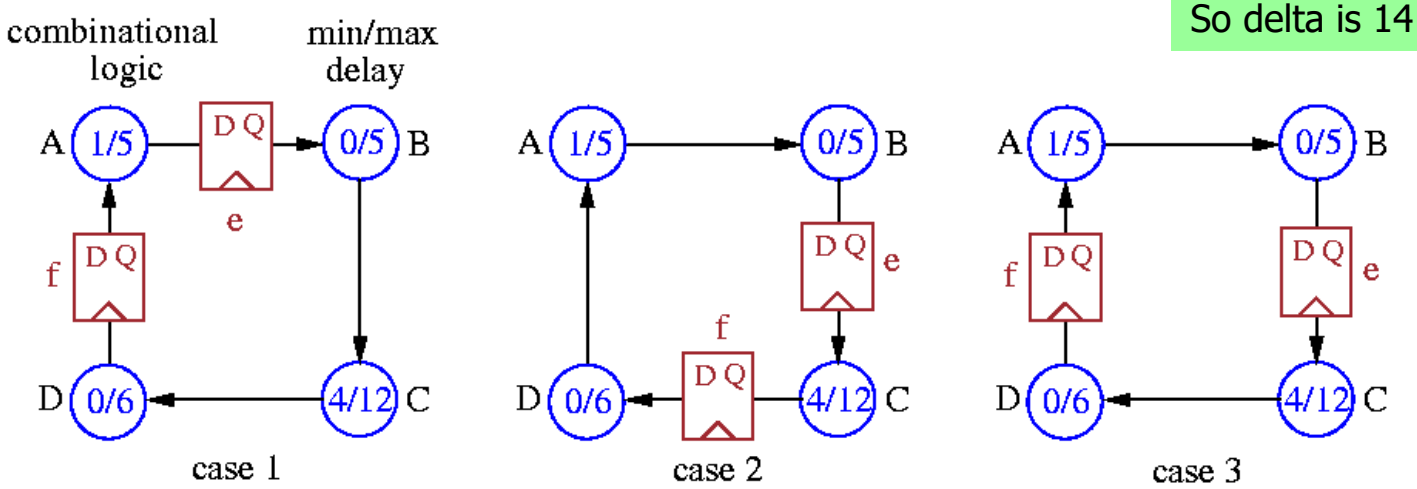
BCD path delay (4/23).
So delta is 19

- Case 2
  - Zero clock skew: clock period $\phi = 16\ \tau$.
  - Schedule *f* 1τ earlier than *e*: clock period $\phi = 15\tau$. (=16-1)

DAB path delay (1/16).
So delta is 15

- Case 3: optimal effective clock period? No!! Optimal case?
  - Zero clock skew: clock period $\phi = 18\tau$.
  - Schedule *e* 4τ earlier than *f*: clock period $\phi = 14\tau$. (total 2 data path delays = 28 -> div by 2 = 14)

CD path delay (4/18).
So delta is 14



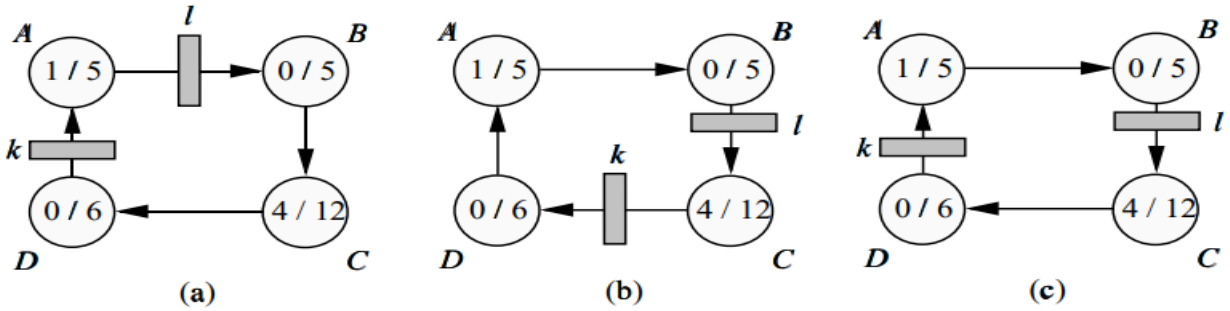


Figure 1: Simultaneous retiming and clock skew scheduling. (a) Original circuit. (b) Fastest retimed circuit with zero skew. (c) Fastest retimed circuit with nonzero skew.

BCD path delay (4/23).
So delta is 19

The original circuit depicted in Figure 1(a) achieves a clock period of $\Phi = 23$ tu (time units) when the clock skew is zero. If register l "sees" the clock edge by 4 tu earlier than register k, the circuit can function correctly with a clock period of $\Phi = 19$ tu. This time is the shortest clock period that can be achieved by scheduling the clock skews without introducing any signal races, since the propagation delay difference along the critical path BCD is 19 tu.

The retimed circuit in Figure 1(b) is obtained from the original circuit by shifting register k backward across block D and register l forward across block B. With zero clock skew, this circuit is optimally retimed and achieves a clock period of $\Phi = 16$ tu. If the clock edge arrives at register k earlier than at register l by 1 tu, this retimed circuit can achieve a shorter clock period of $\Phi = 15$ tu. No further clock period improvements are possible because the propagation delay difference along path DAB is 15 tu.

DAB path delay (1/16).
So delta is 15

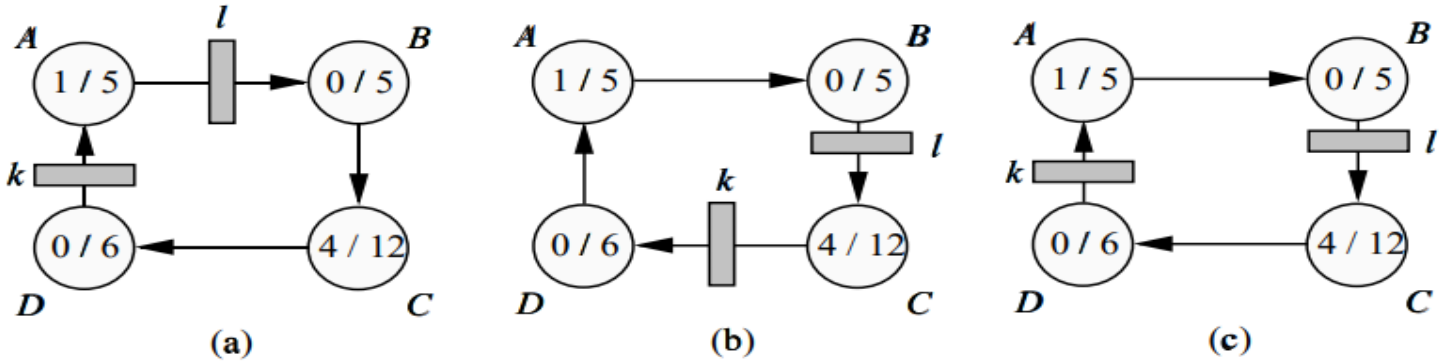


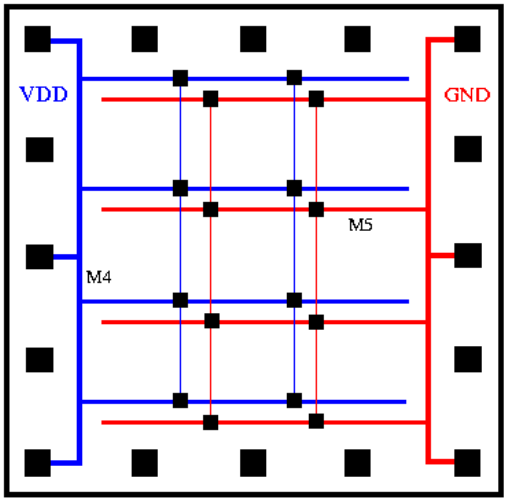
Figure 1: Simultaneous retiming and clock skew scheduling. (a) Original circuit. (b) Fastest retimed circuit with zero skew. (c) Fastest retimed circuit with nonzero skew.

In Figure 1(c), another retimed version of the original circuit that is obtained by shifting register l across block B is shown. For this circuit, the shortest clock period that will not result in any timing violations when the clock skew is zero is $\Phi = 18$ tu. If the clock edge arrives at register k later than at register l by 4 tu, however, this circuit can function correctly with a clock period of $\Phi = 14$ tu. This time is the shortest clock period that can be achieved by applying both retiming and clock scheduling, since the total propagation delay around the cycle is 28 tu which must be distributed between two combinational paths.

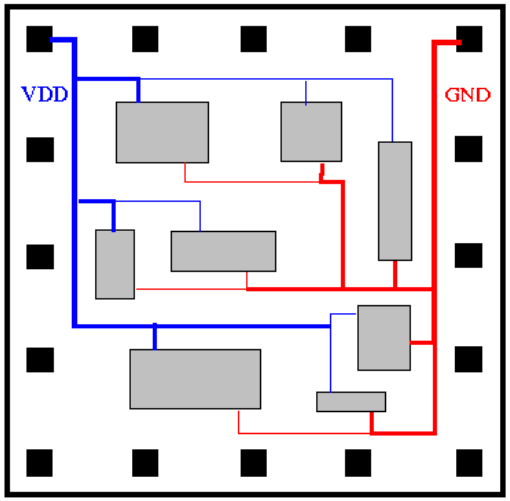
P/G Routing

Power/Ground Routing

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



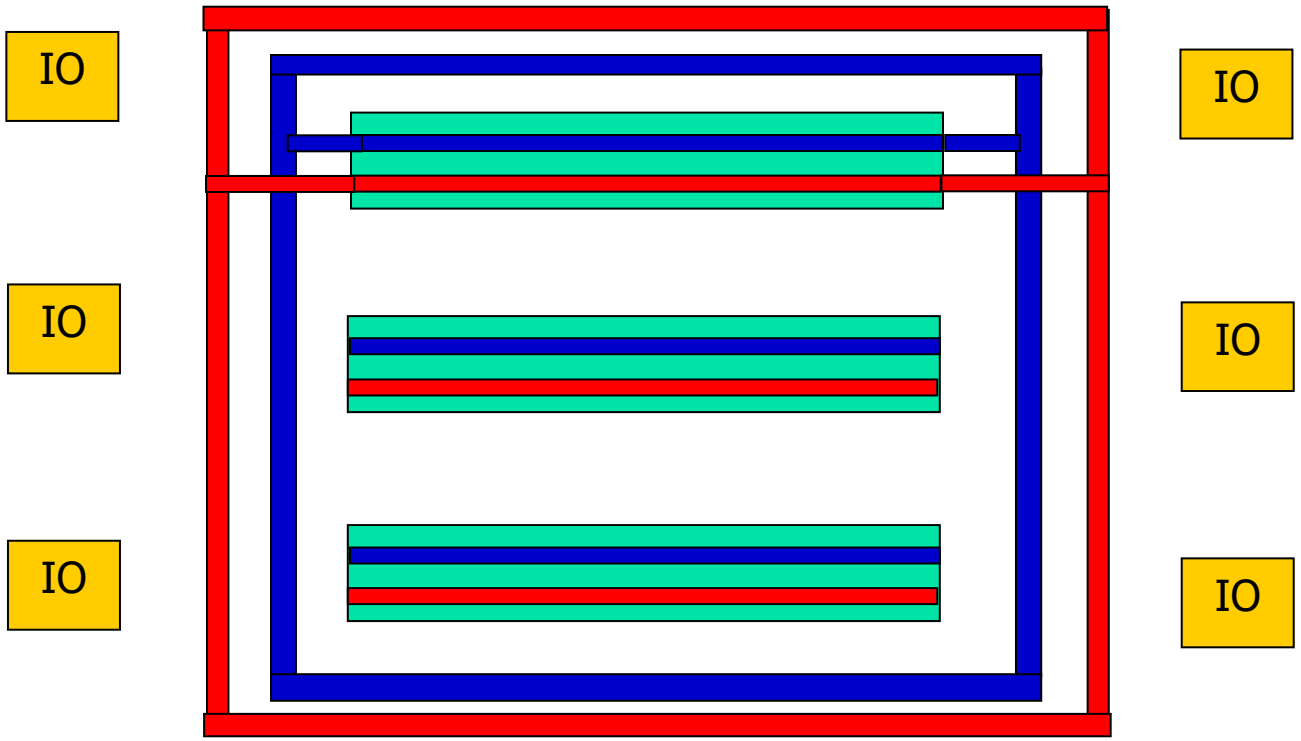
grids

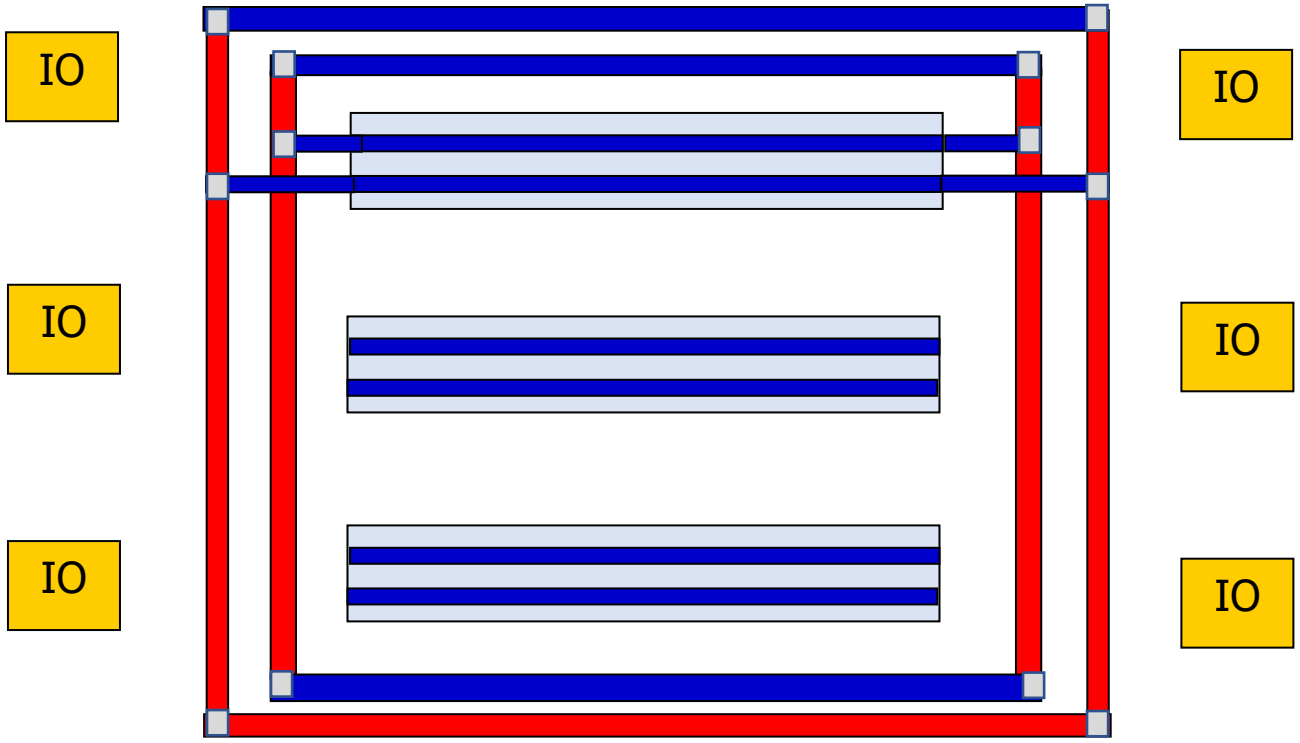


interdigitated trees

More About P/G Wiring

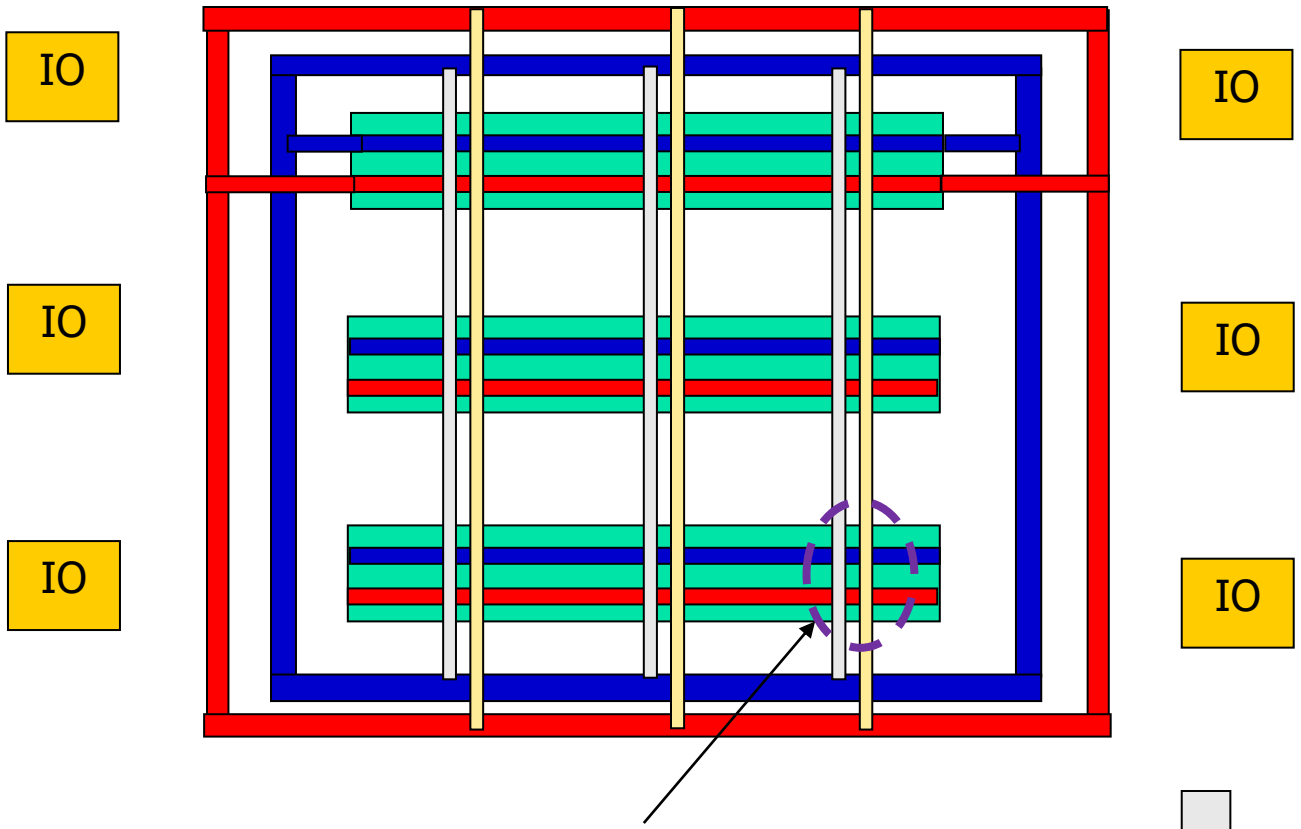
- The previous slide shows the simple P/G wiring. Nowadays it is much more complicated
  - In standard cell rows, we do
    - P/G-follow-pin, P/G meshes, P/G rings around cell area
  - In I/O area, outer rings, inner rings



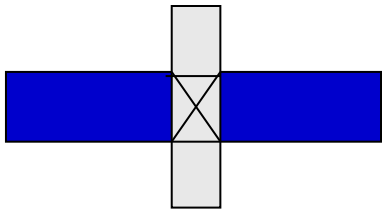

IO
IO
IO
IO
IO
IO

More About P/G Wiring (2)

- P/G Meshes to reduce IR drop

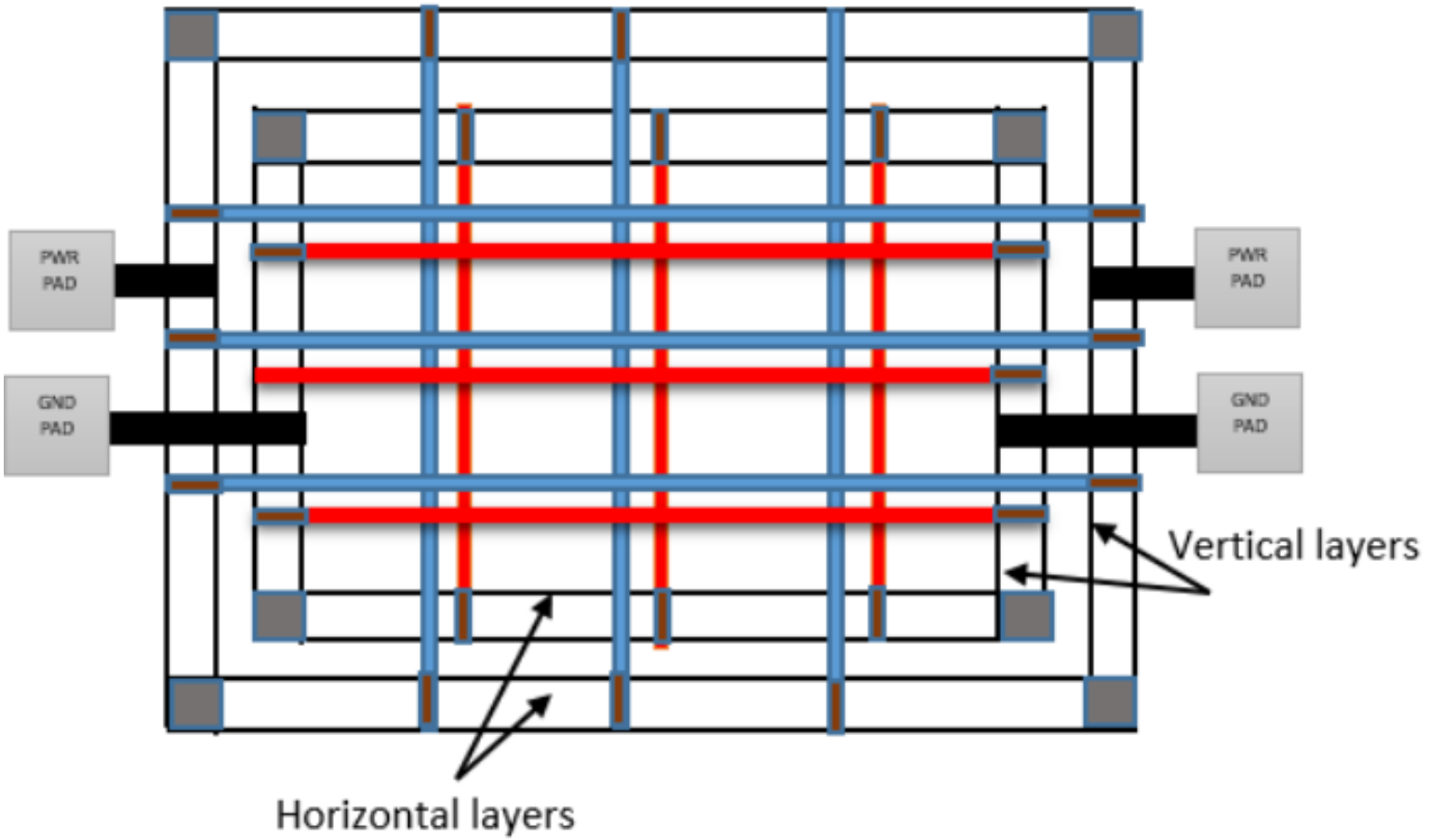


- Via connections are critical



https://inst.eecs.berkeley.edu/~cs250/fa09/lectures/lec08.pdf
CS250 @ UCB

https://www.physicaldesign4u.com/2020/01/power-planning.html



- Build a power network

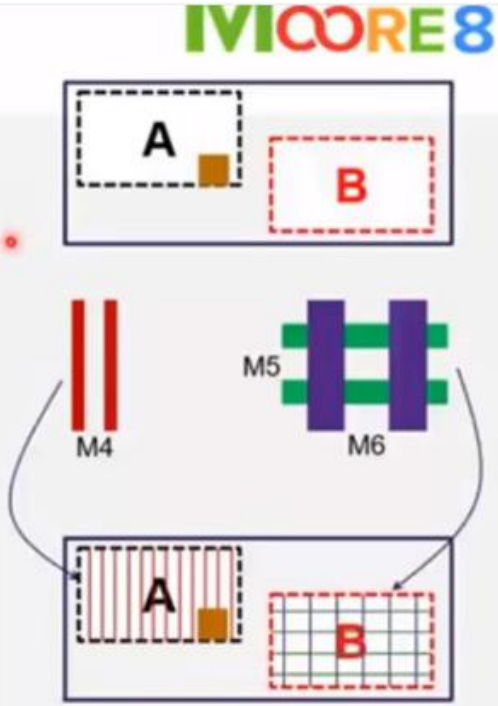
Pattern_Based Power Network Synthesis

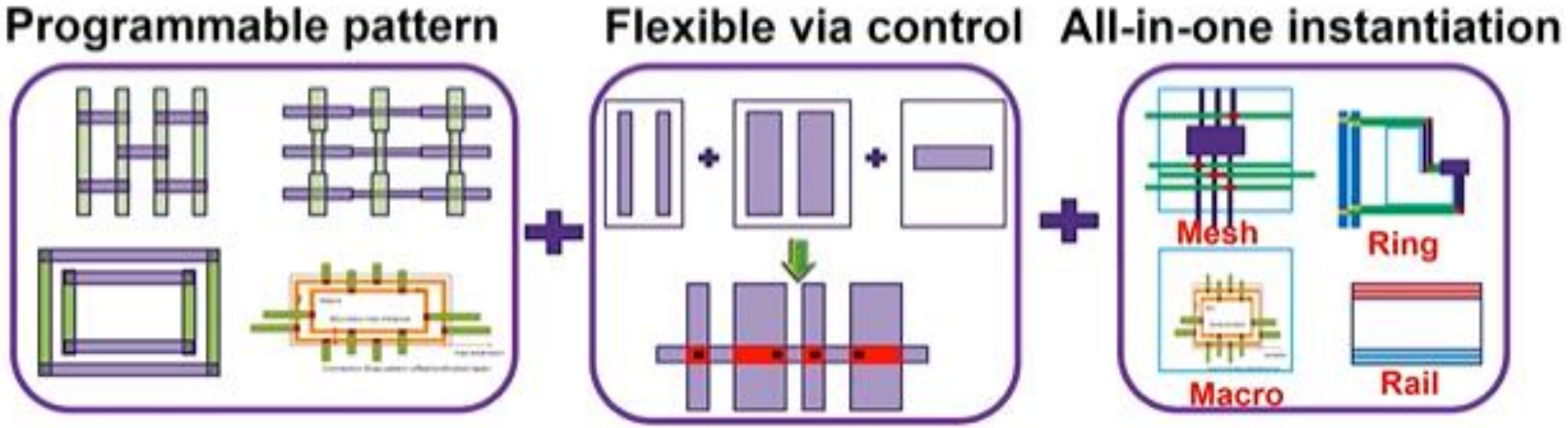
1. **Define PG Regions**

2. **Define Pattern: Structure**
   - Defines metal layers, spacing, width, …

3. **Define Strategy: P/G topology**
   - Applies specific pattern to specific PG region, voltage areas or bounds and specific power/ground nets
   - Flexible to floorplan change (no fixed coordinates)

4. **Compile power network**

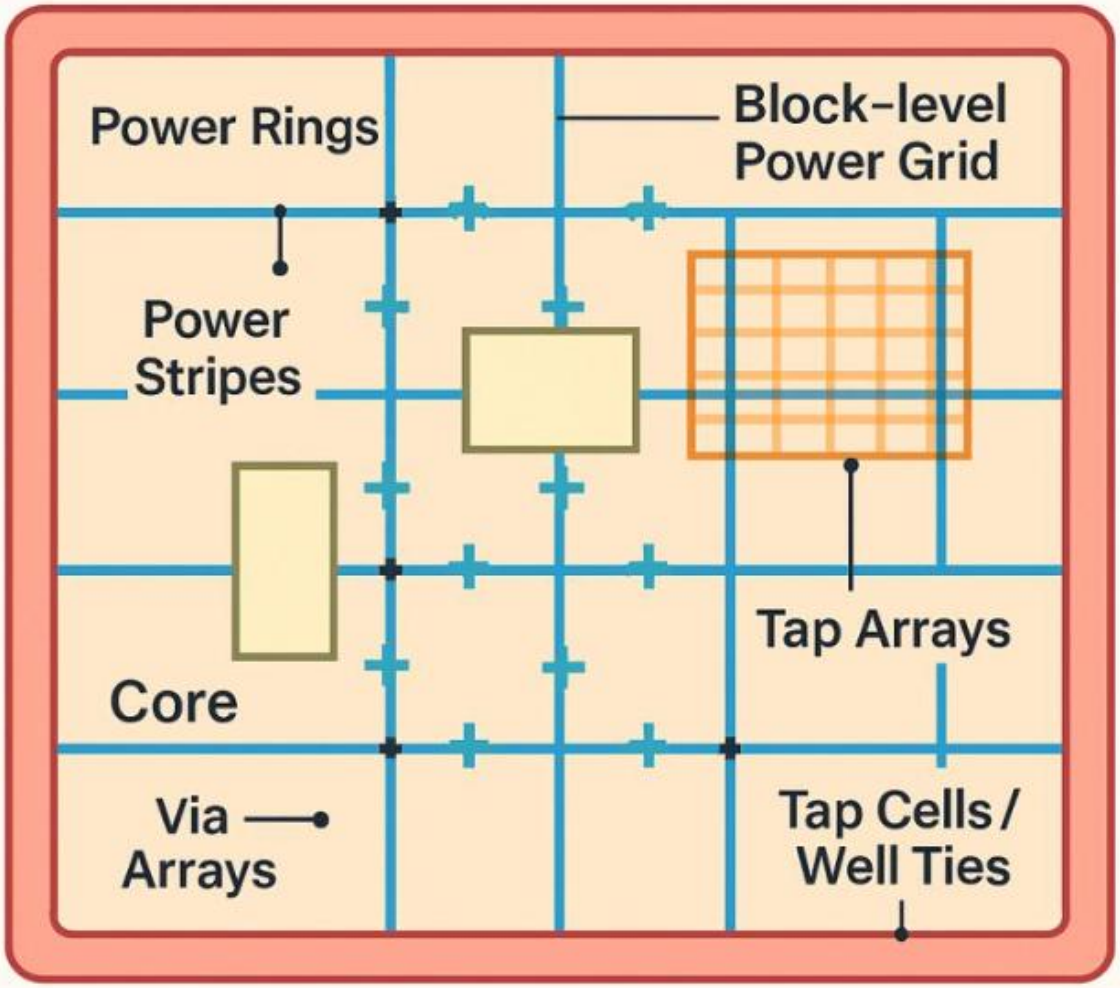





```
StrapConfig : PG_MESH1 {
   target_area: core
   category: mesh
   nets: VDD  VSS
   layer:  M3
      widths: 0.24  0.36
      direction: vertical
      spacings: minimum | interleaving | {0.24}
      pitch: 6.912
   }
}
```

New Simple User Interface

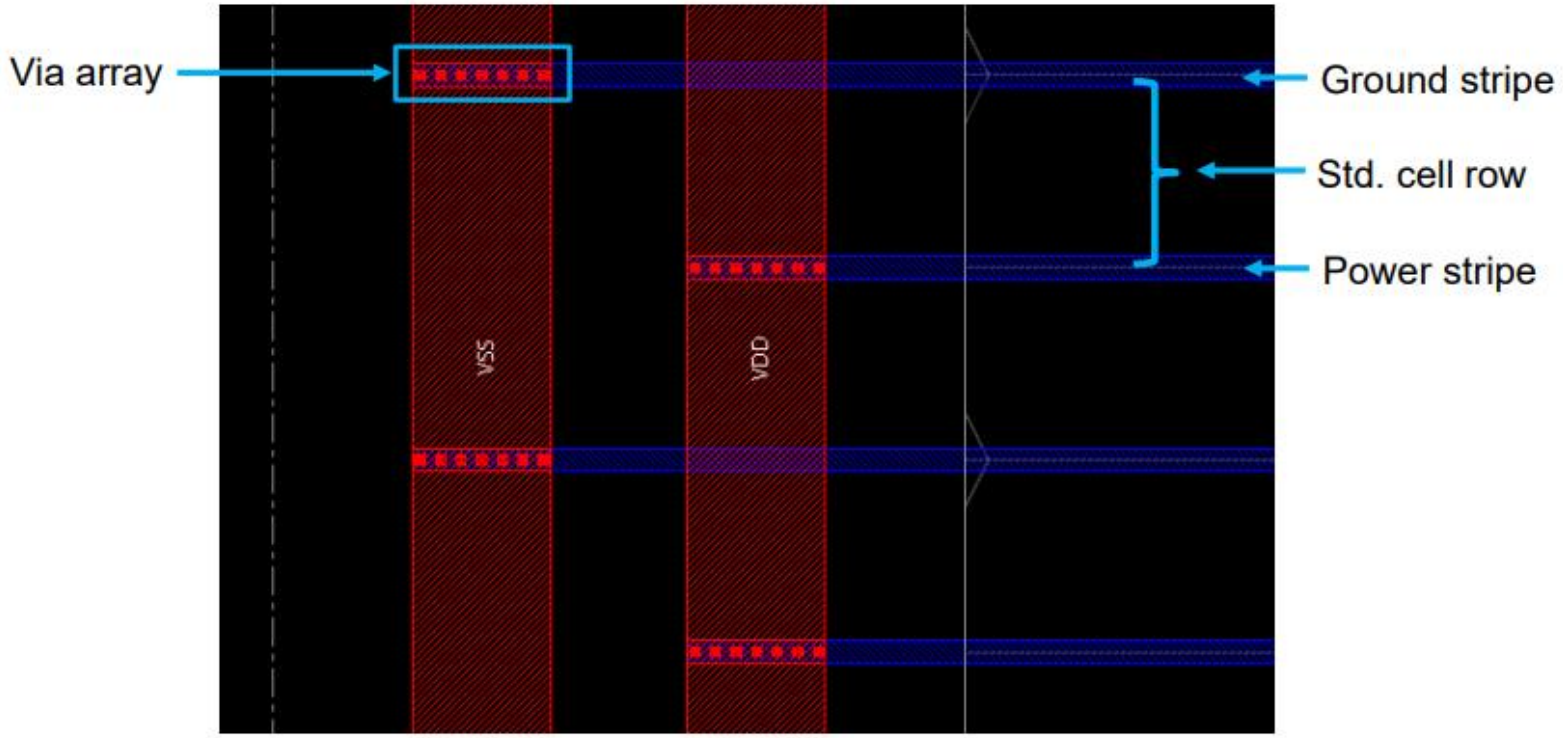
Power Planning



Power Stripes

2. P/G Network Design

- As you see, the P/G stripes are alternating between VDD and VSS. See the vias.



https://studylib.net/doc/25704408/ic-compiler-ii-design-planning-user-guide

To define the power plan strategy, use the `set_pg_strategy` command as in the following example.

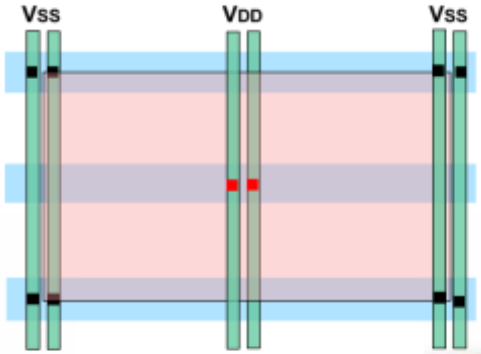
```
icc2_shell> set_pg_strategy ring_strat -core \
   -pattern {{name: ring_pattern} {nets: {VDD VSS}} \
             {offset: {3 3}} {parameters: {M7 10 2 M8 10 2 true}}} \
   -extension {{stop: design_boundary}}
```

P/G Wiring Affecting Placement, Routability

Logic Density Improvement (I)
Power Plan Optimization

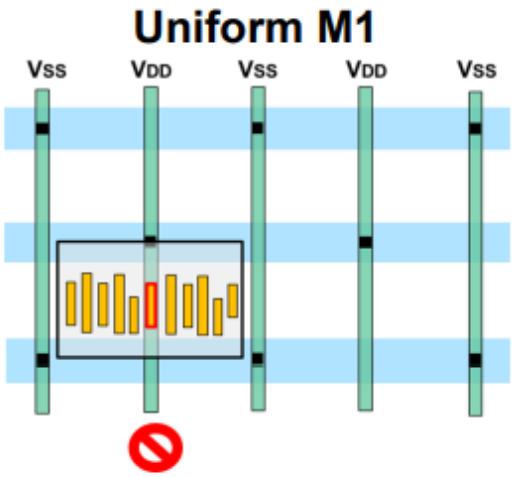
- To improve power plan IR, PG via count is increasing over technology generations

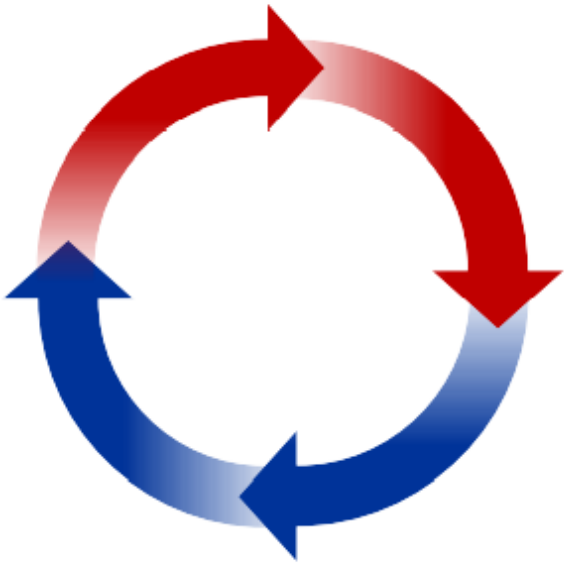
- However, shrinking PG pitch impacts routing resources. So different and new PG design approaches evolved

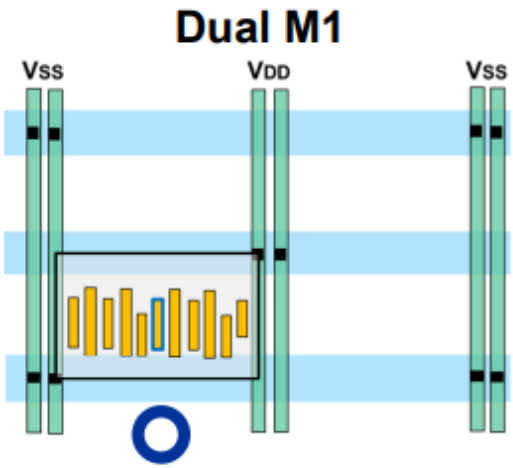


Allows better cell placement freedom

Logic Density Improvement (II)
Power Plan and Cell Layout Co-optimization

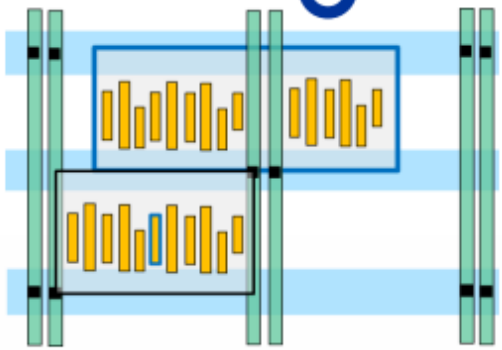




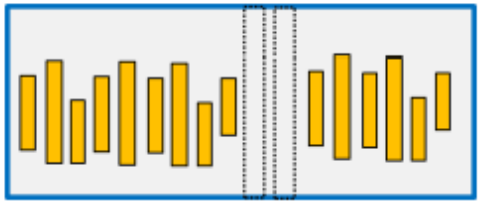


PG design has to be friendly to cell architecture (Uniform → Dual)

PG and Cell Optimized Co-Design



Cell re-Design



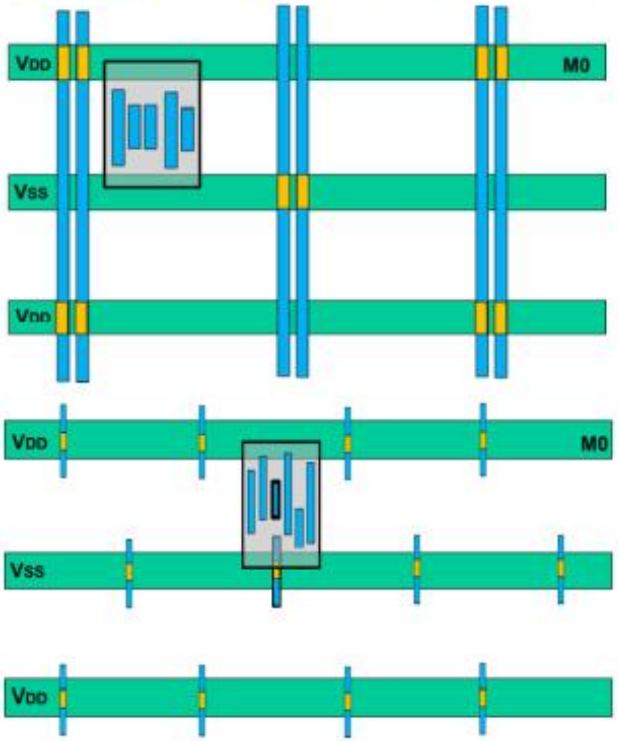
Big cell design has to be redesigned to be compatible with Dual PG architecture

ISPD 2017

Open Innovation Platform™

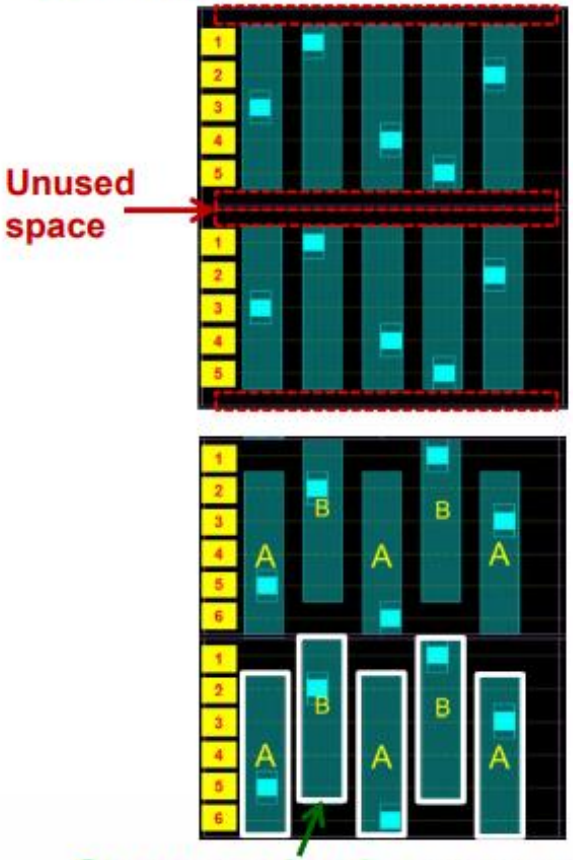
Logic Density Improvement (III)
Power Plan and Cell Layout Push to Limit

Power strap:
Less cells placed under PG



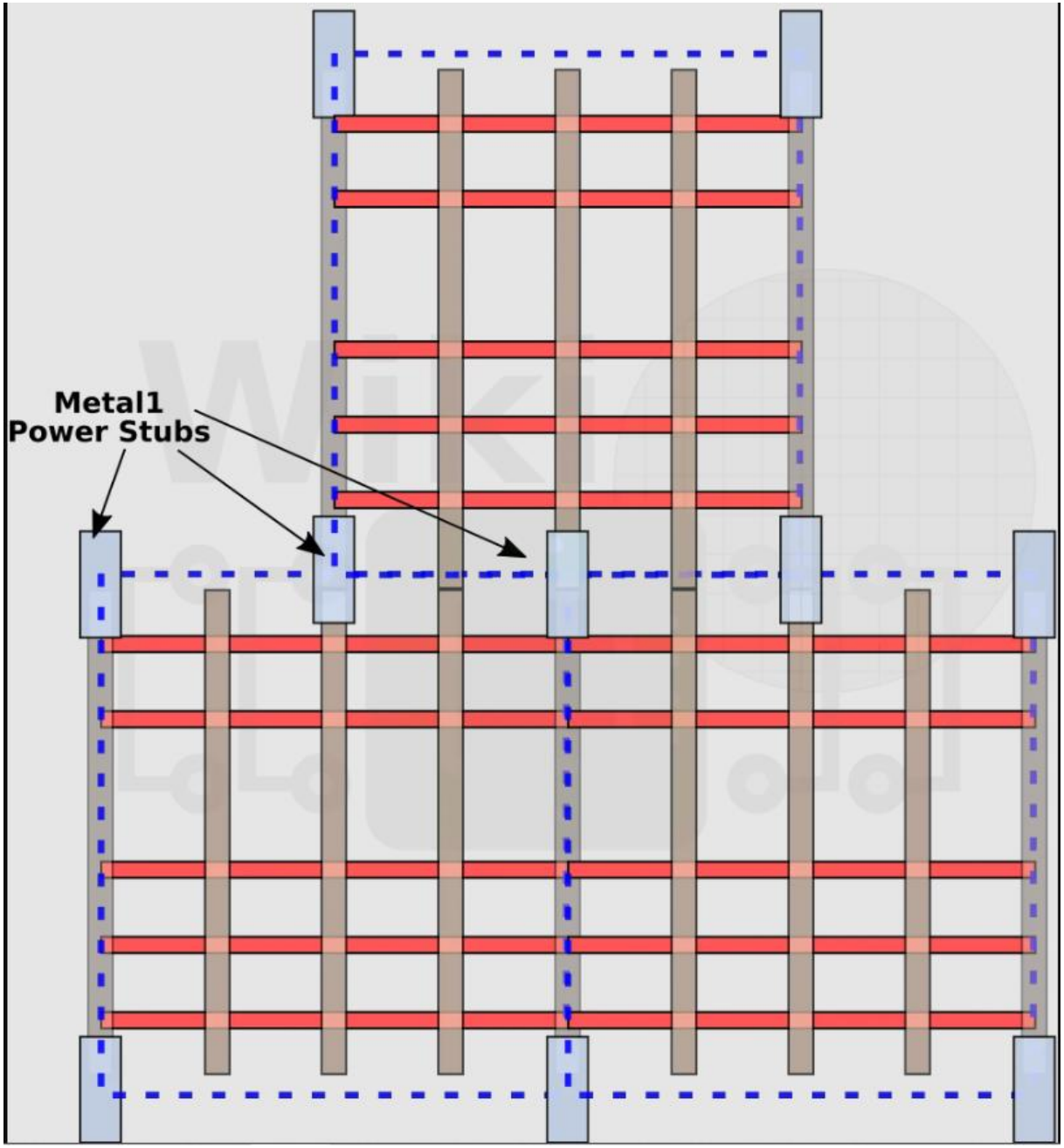
Power stub:
More cells placed under PG

No stagger pin: 5 access points

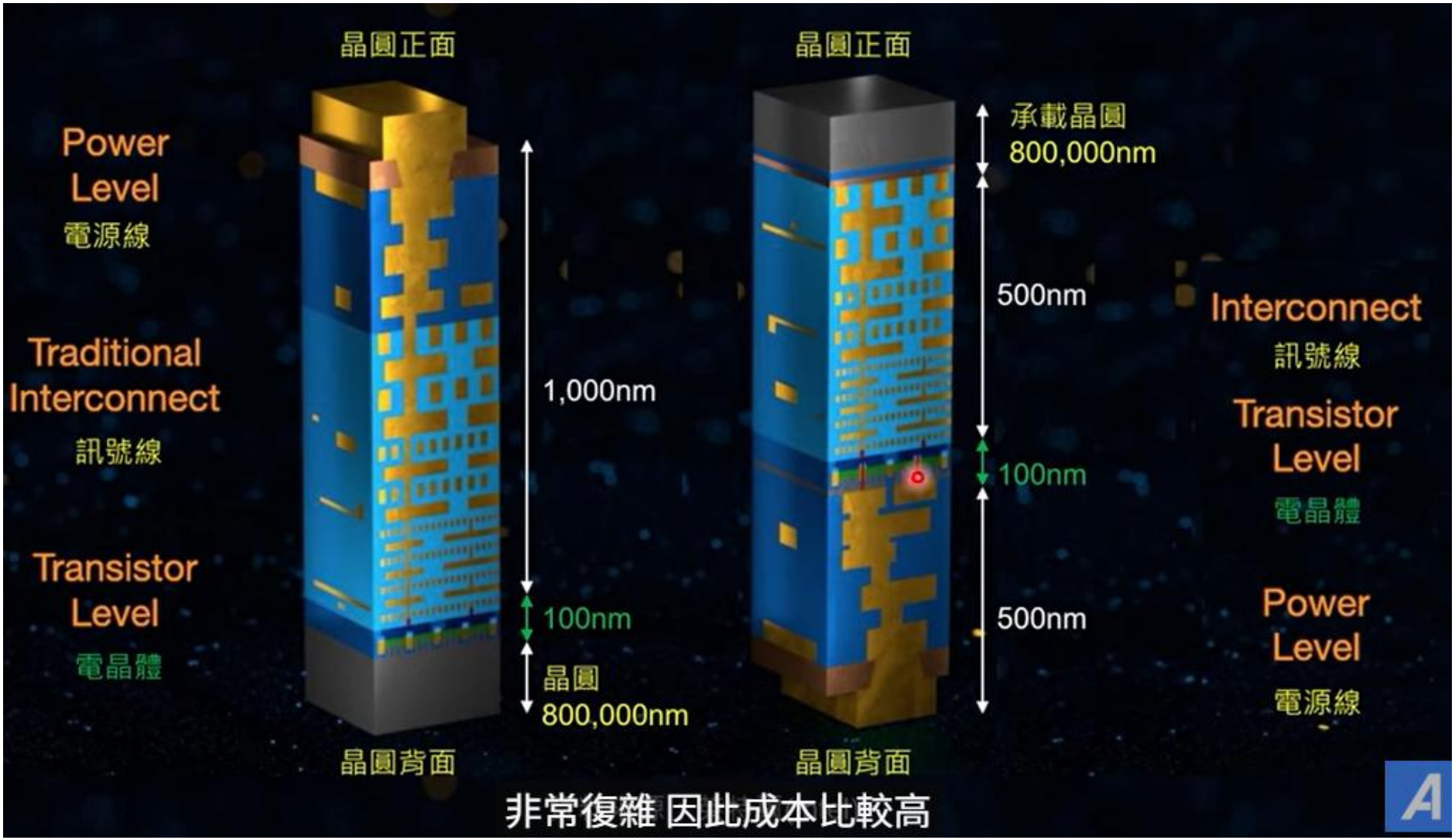


Pin access is critical for routing. So more tracks available is better

10nm cell power stubs multi block level



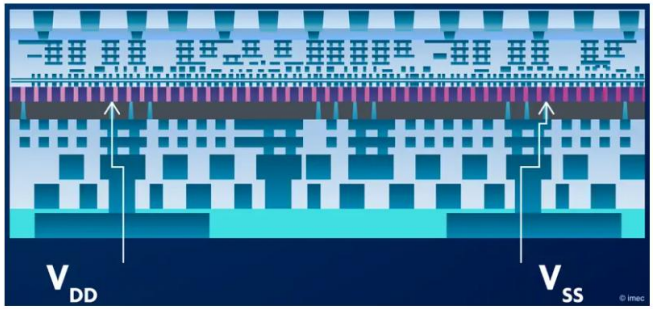
Backside Power Delivery

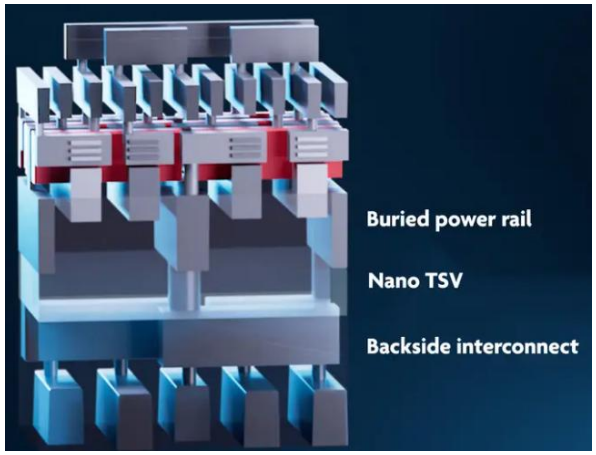


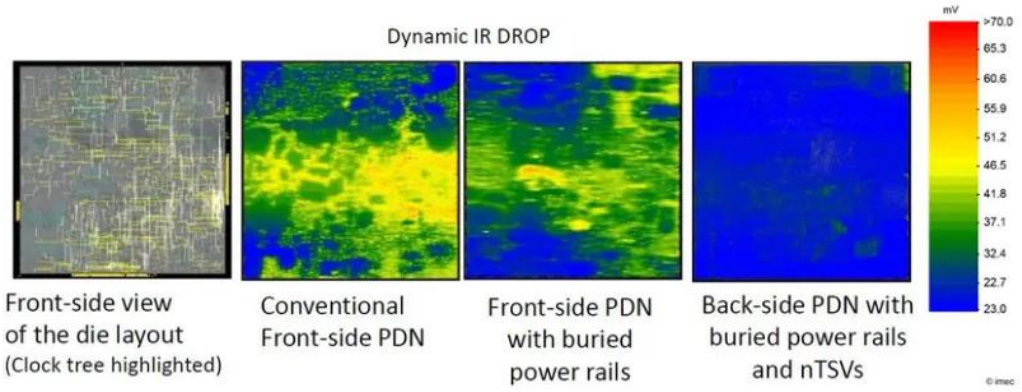
https://www.imec-int.com/en/articles/how-power-chips-backside

Impacts From **Backside** Power Delivery

- Cell library
- Pin access
- P/G Wiring
- Tie-high/low
- DRC/LVS/RC
- IR-drop
- GDS/OASIS out (or DEF?)
- …
- Many CAD programs to be enhanced



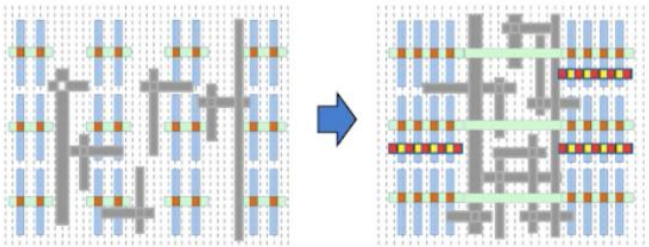




Examples Of Enhancements For DTCO

+1% Speed Gain with Same IR on Cortex-A72 DS Design

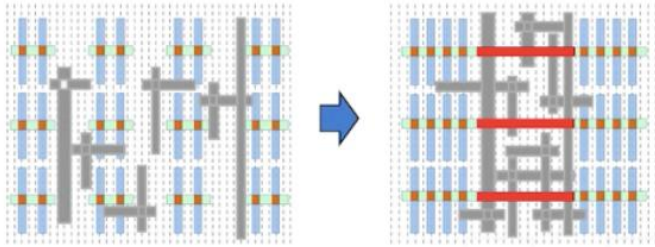
Inserting PG Vias after Route



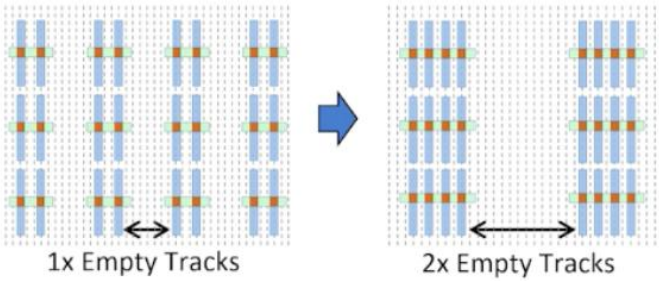
IR Recovery
PG Via Insertion

IR Recovery
PG Wire Patching

Connecting PG Wires after Route



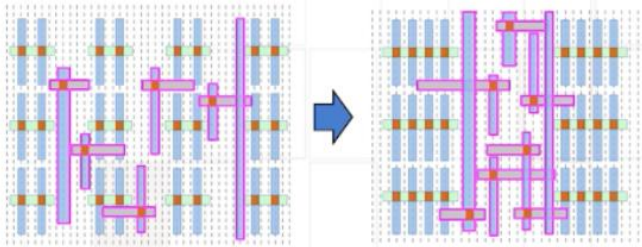
PG Grouping : 1x → 2x



Speed Gain

PG Grouping
Opt

Clock Net Grouping and Skew Reduction for Speedup



1.5~2% Speed Gain on Cortex-A78 IS_INT and A72 ID/DS

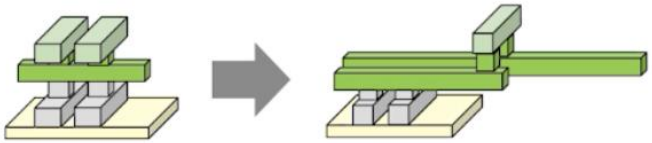
Low-Layer Metal R Reduction

Via R Reduction

High-Layer Metal R Reduction

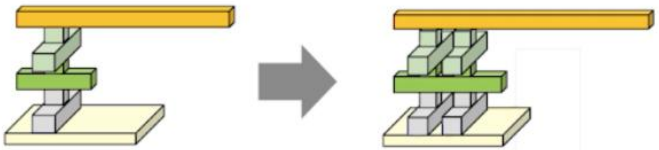
Partial Parallel Routing

Add parallel wires on timing critical nets to utilize low-layer metals for R reduction



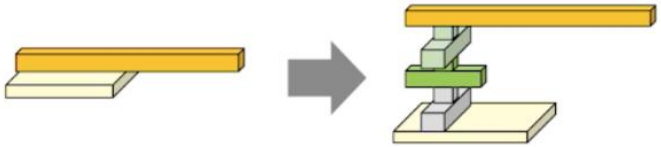
Via Pillar Insertion

Add parallel VIAs on source side of timing critical nets for R reduction



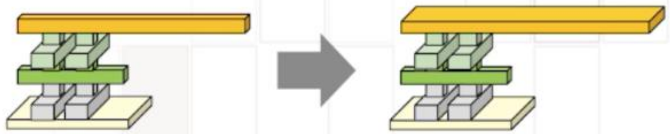
Layer Promotion

Route timing critical nets on upper metal layers for R reduction

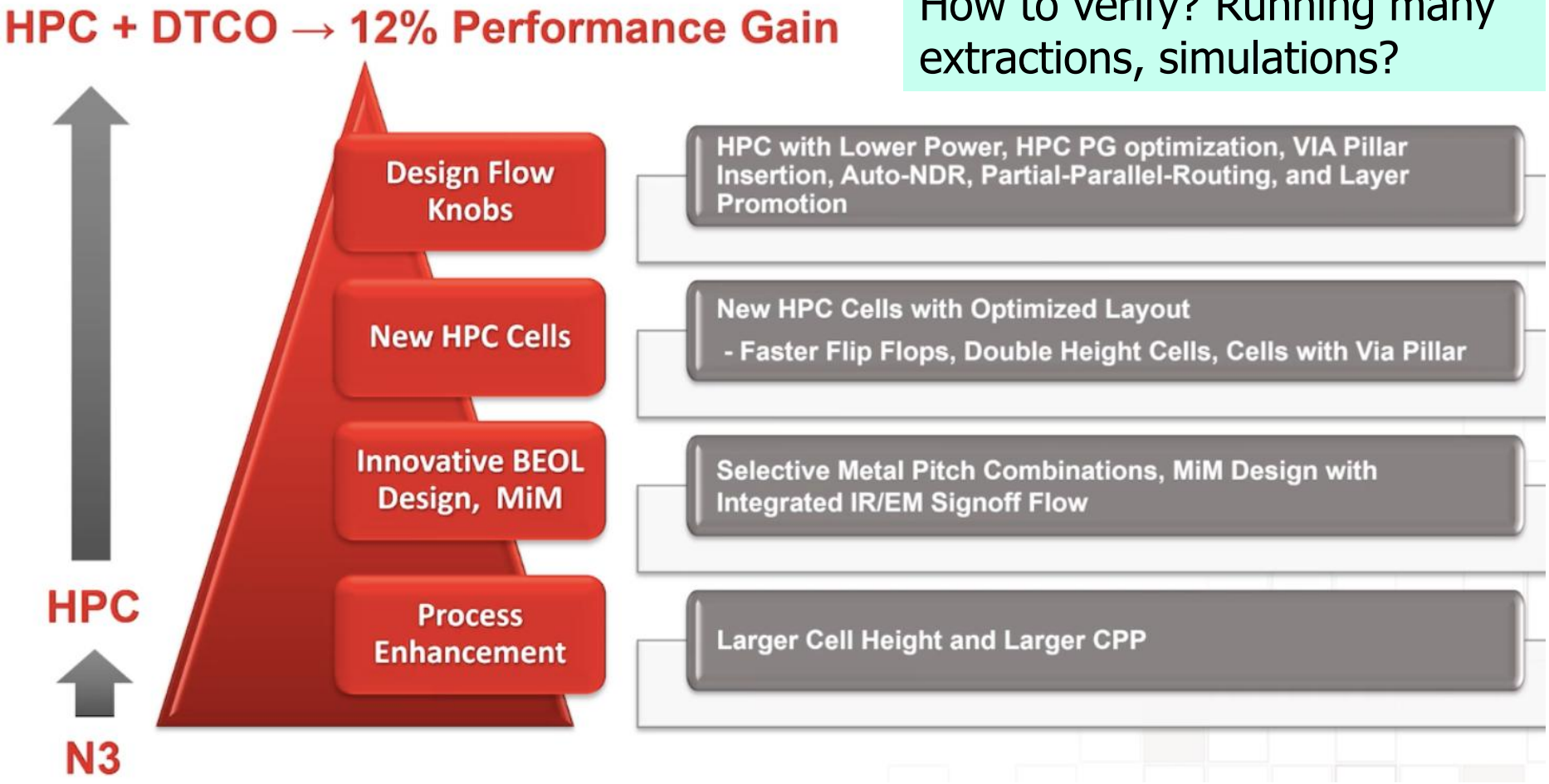


AutoNDR Routing

Enlarge metal width and spacing of timing critical nets (non-default routing, NDR) for RC reduction



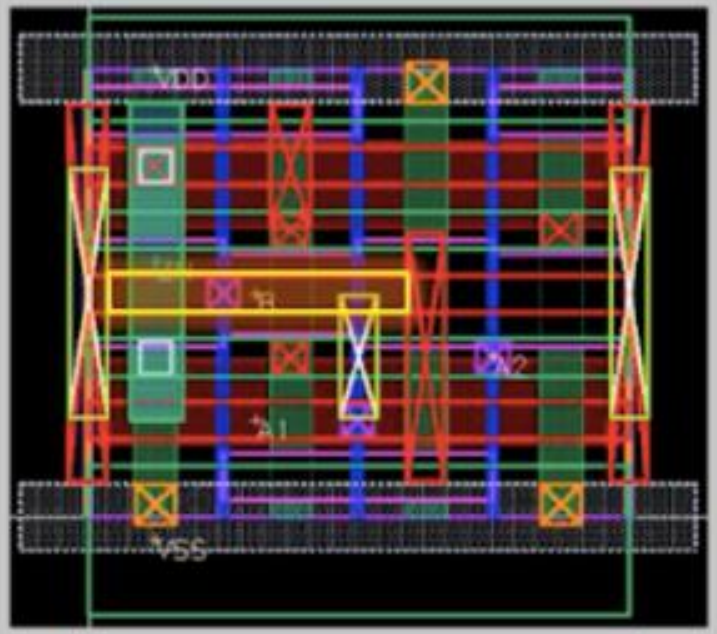
How to verify? Running many extractions, simulations?



https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/oip-2021-n3

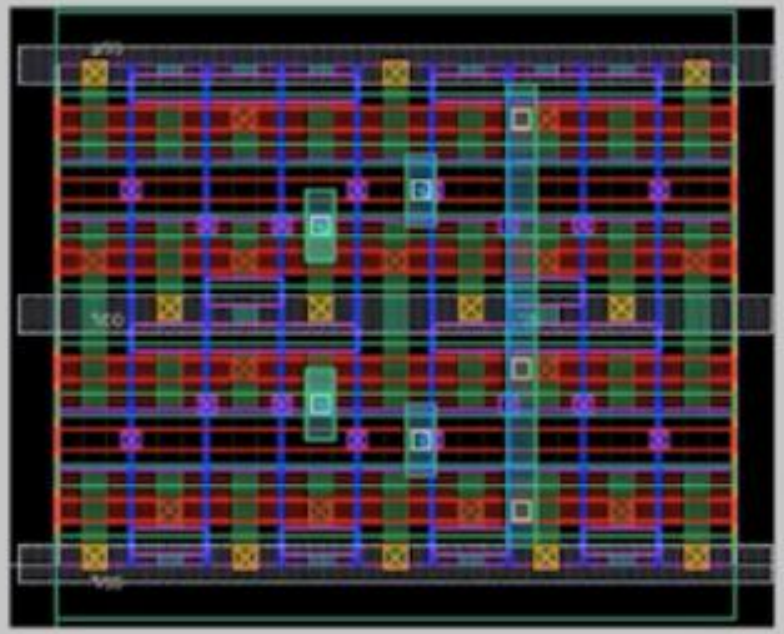
M0 Optimization

Cell M0 optimization for lower cap and higher speed



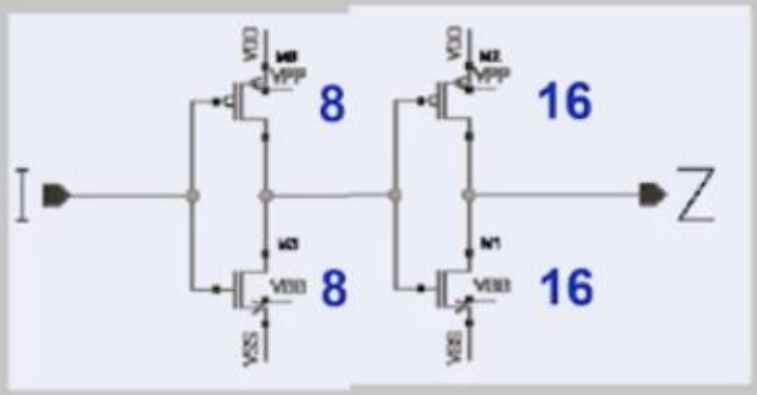
Double Height Cells

For better form factor for high-driving cells



Stage Ratio Optimization

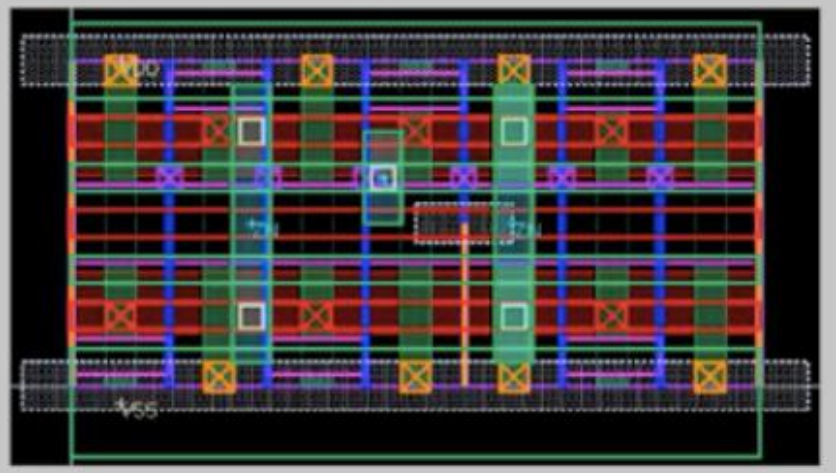
Optimized stage ratio and speed for multi-stage cells

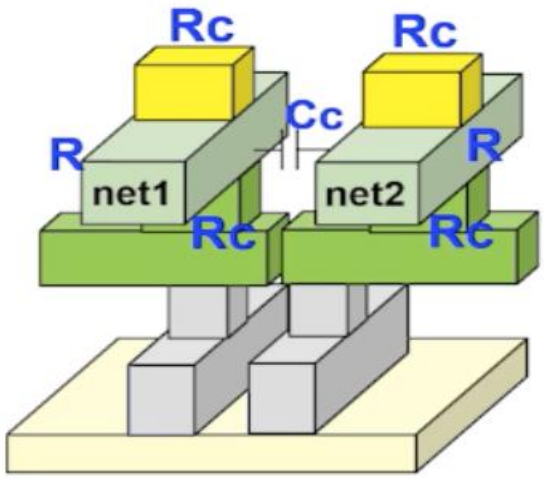


BUFFD16

Skew Cell Optimization

Skew cells for better path timing with optimized resistance and pin cap





Wide Metal Pitch with larger VIA for HPC Designs

HPC applications often call for a larger metal pitch (lower RC) and a larger via (lower resistance).

TSMC created special metal pitch combinations and design rules to have a good tradeoff for PPA. The result is a 2-4% gain in performance.

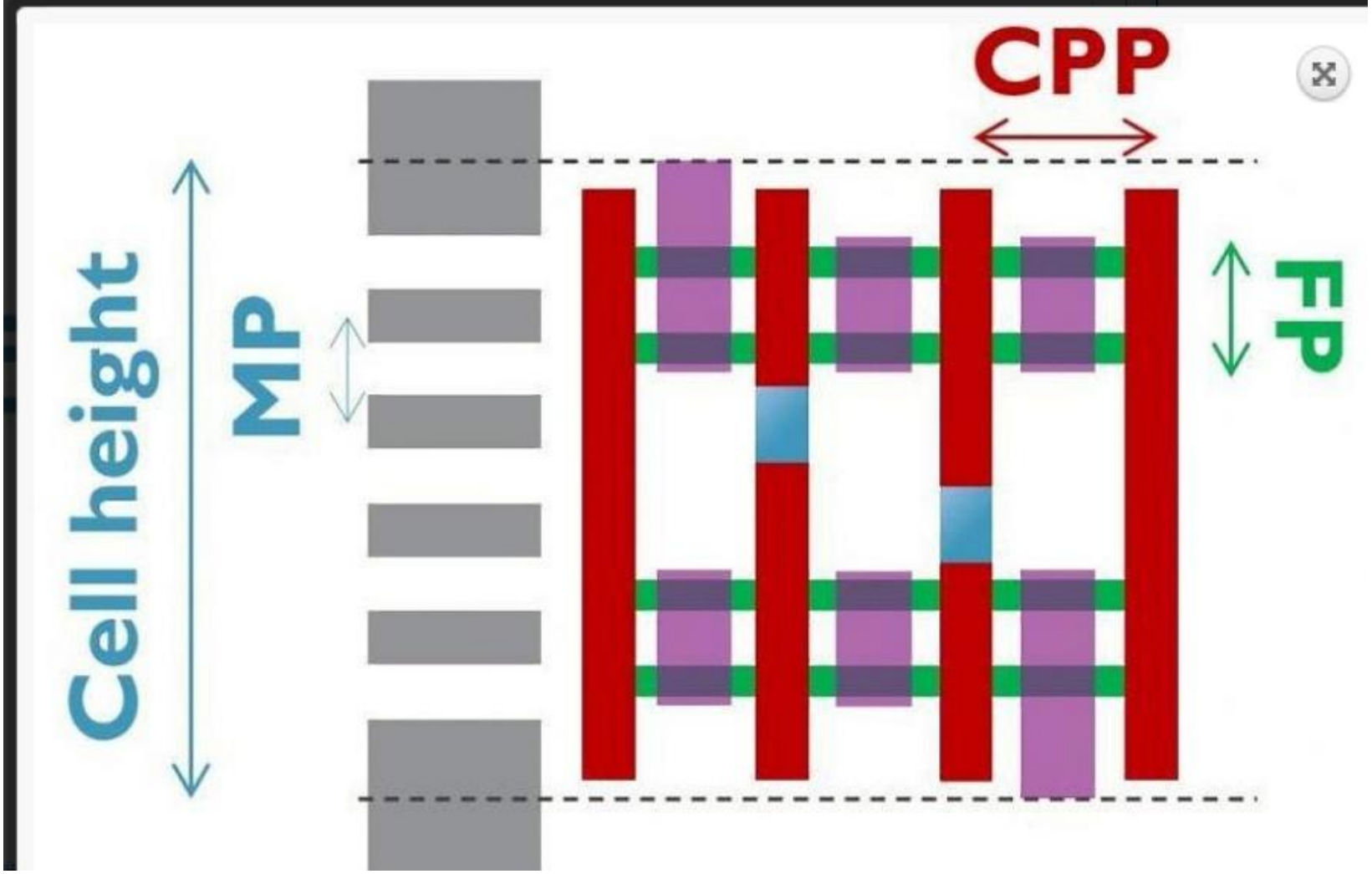
How to verify 2-4%?

Solution	Features
Model	Larger CPP device
STD Cell Library	New HPC library with larger cell height
	New HPC cells and lower-power cells
Design Flow	HPC lower-Power solution
	HPC PG optimization
	VIA pillar insertion, auto-NDR, partial-parallel routing, and layer promotion
High Density MiM Flow	MiM implementation flow and kits
	IR/EM signoff flow for MiM
DRM/DRC	Larger CPP rule enablement
	High density MiM rule enablement
	Wide metal pitch with larger VIA rule enablement

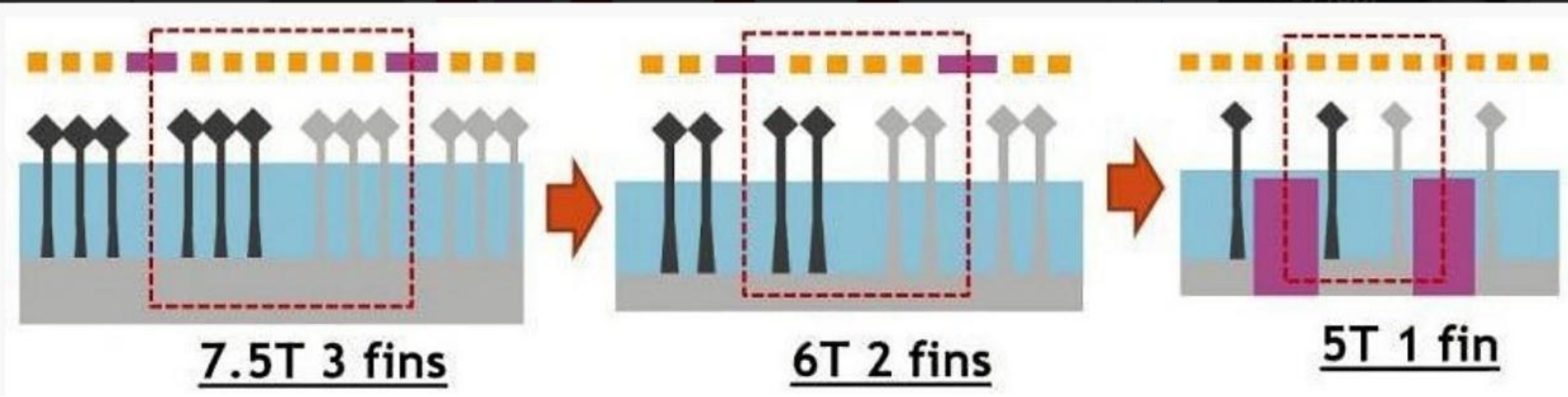
https://www.ctimes.com.tw/DispArt-tw.asp?O=HK64L9RT9A6ARASTD0

評比奈米片、叉型片與CFET架構

圖一：邏輯標準單元佈局的示意圖：接觸式多晶矽閘極間距（contacted poly pitch；CPP）、鰭片間距（fin pitch；FP）、金屬層間距（metal pitch；MP），以及標準單元高度（cell height）。



圖二：為了進一步微縮標準單元，FinFET架構必須減少鰭片數量，新一代設計的鰭片構形會更長、更薄且更緊密，驅動電流會隨之降低，變異性也會增加。

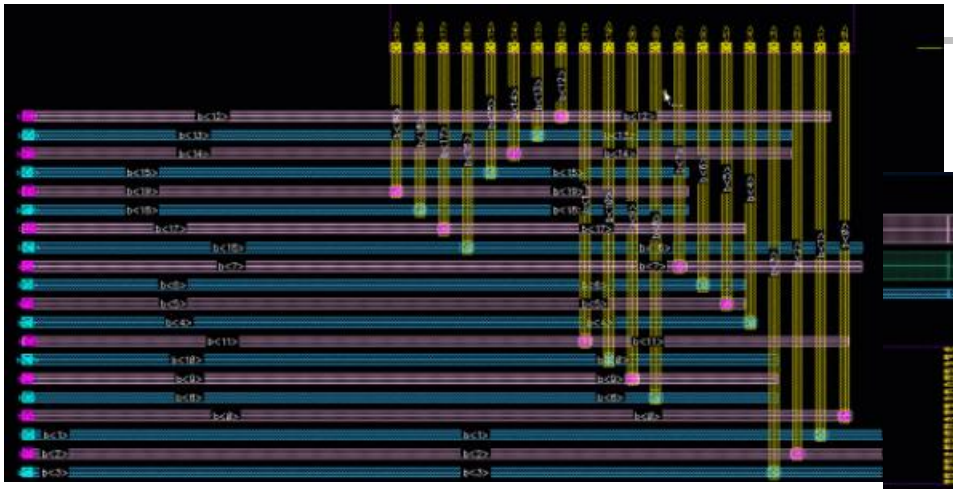


Standard Cell Libraries

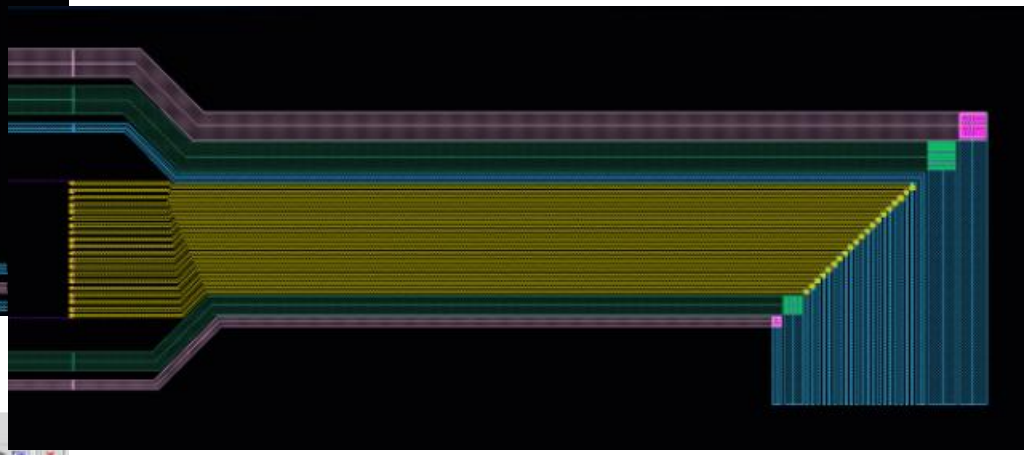
https://www.eng.biu.ac.il/temanad/files/2017/02/Lecture-4-Standard-Cell-Libraries.pdf
This pdf contains many good info
Lecture-4-Standard-Cell-Libraries.pdf

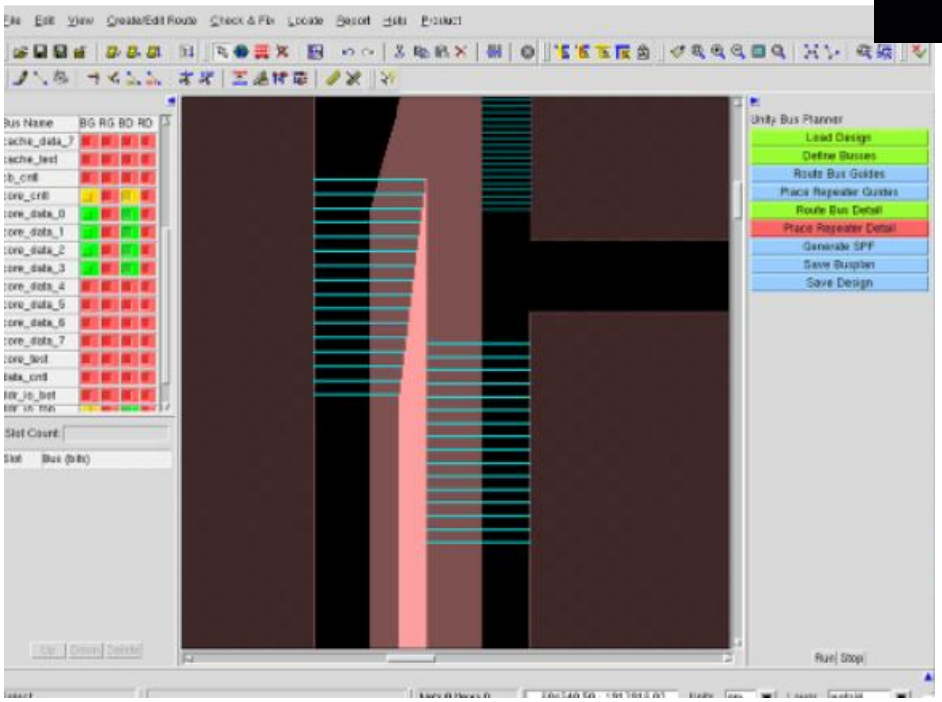
Special Routing

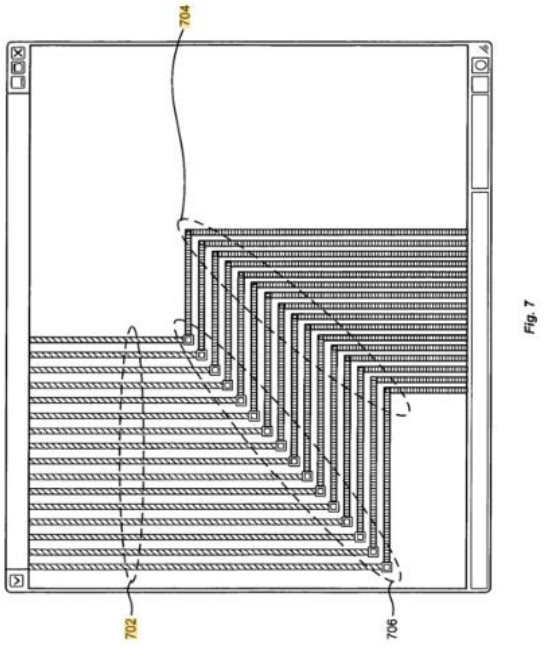
Bus Routing



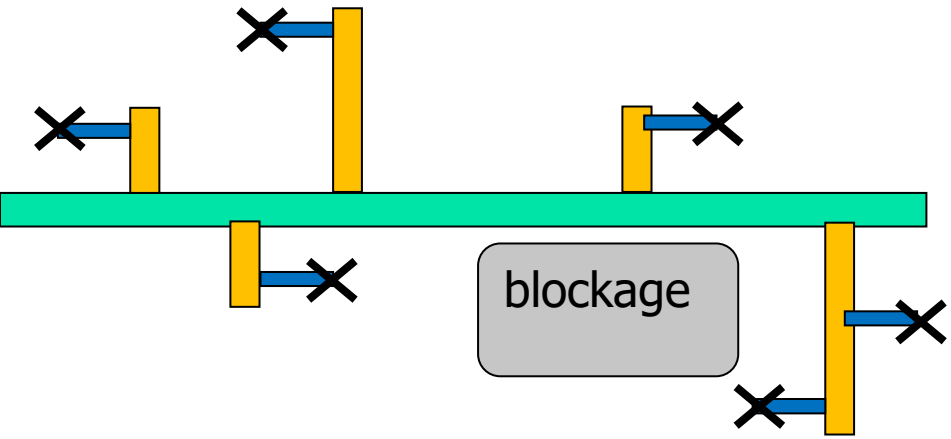
https://www.ema-eda.com/products/skillcad/ic-layout-automation-system







Spine Routing (Fishbone Routing)



- Where to put down the trunks?
  - >1 trunk?
  - Shall we allow a "break"?
- Where to assign the 2nd branches?
  - How many pins to be grouped
- Pin-to-trunk
- Connecting to pins is not easy

H-tree
Clock routing
Non-default routing rules
More requests for Custom Routing

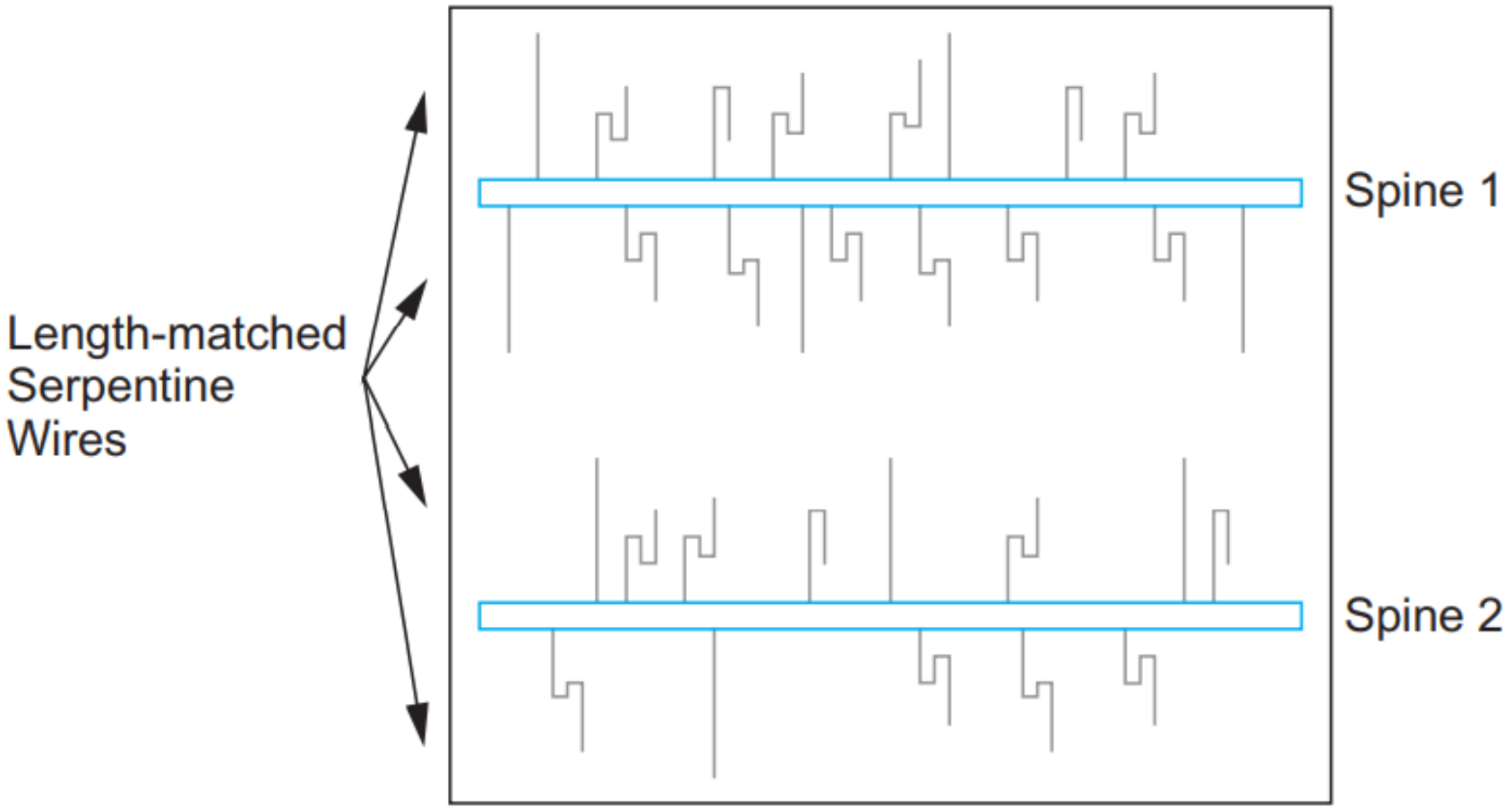


FIGURE 13.27 Clock spines with serpentine routing

Fishbone: A Block-Level Placement and Routing Scheme

Fan Mo and Robert K. Brayton
EECS, UC Berkeley

Outline

- The block level placement and routing problem
  - Routability, predictability
- Fishbone scheme
  - Spine net topology
  - Base/Virtual pin pair
  - Row/column routing with left-edge algorithm
- Integrated placement and routing
- Experimental results
- Discussion

Spine Topology

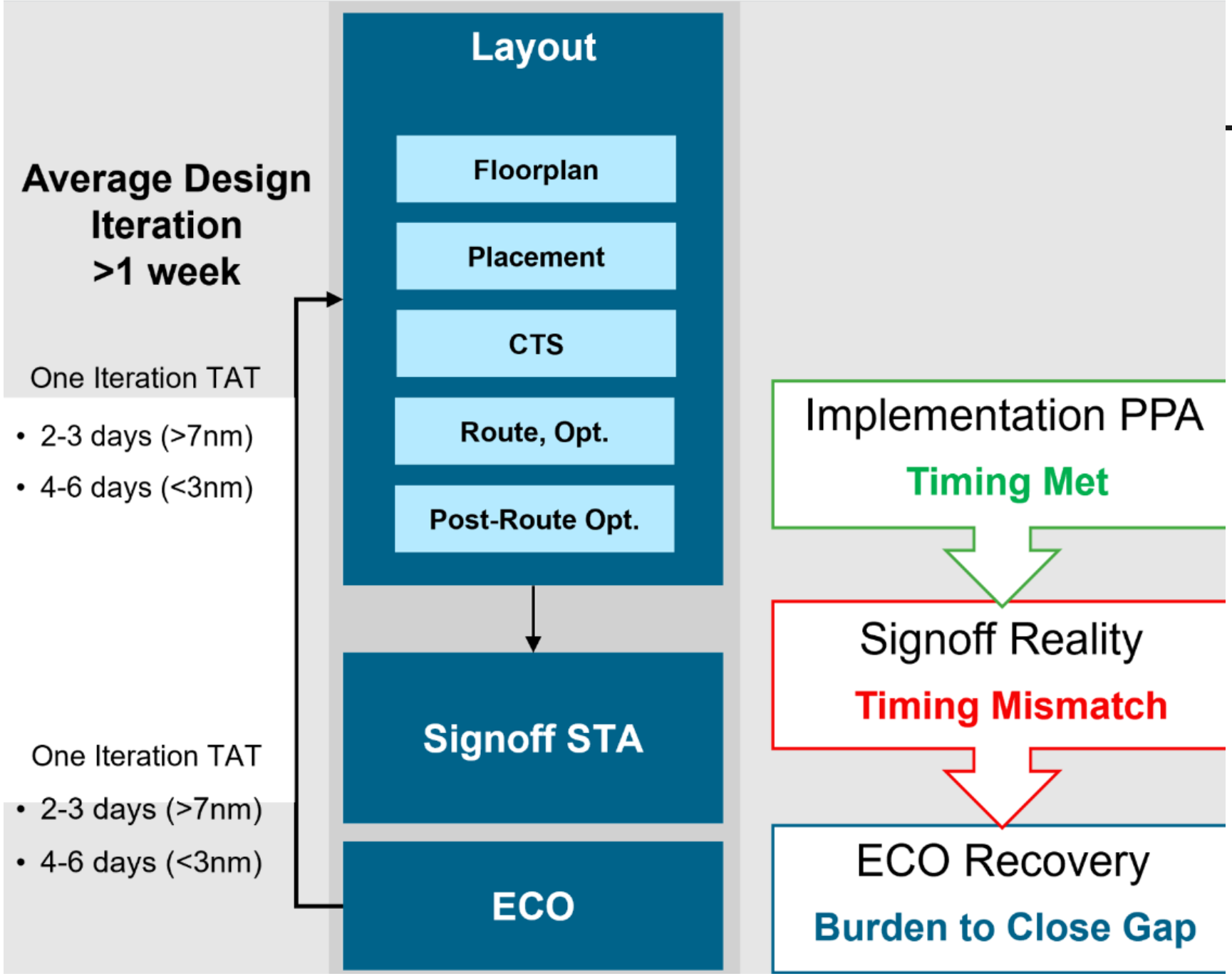
- The output pin of a net is on a vertical wire called "trunk"; and all the input pins connect to the spine by horizontal "branches".
  - Given pin positions, the net shape is fully determined.
  - Pin-pin distance is Manhattan.
  - Routability is easy to detect, given all pin positions.

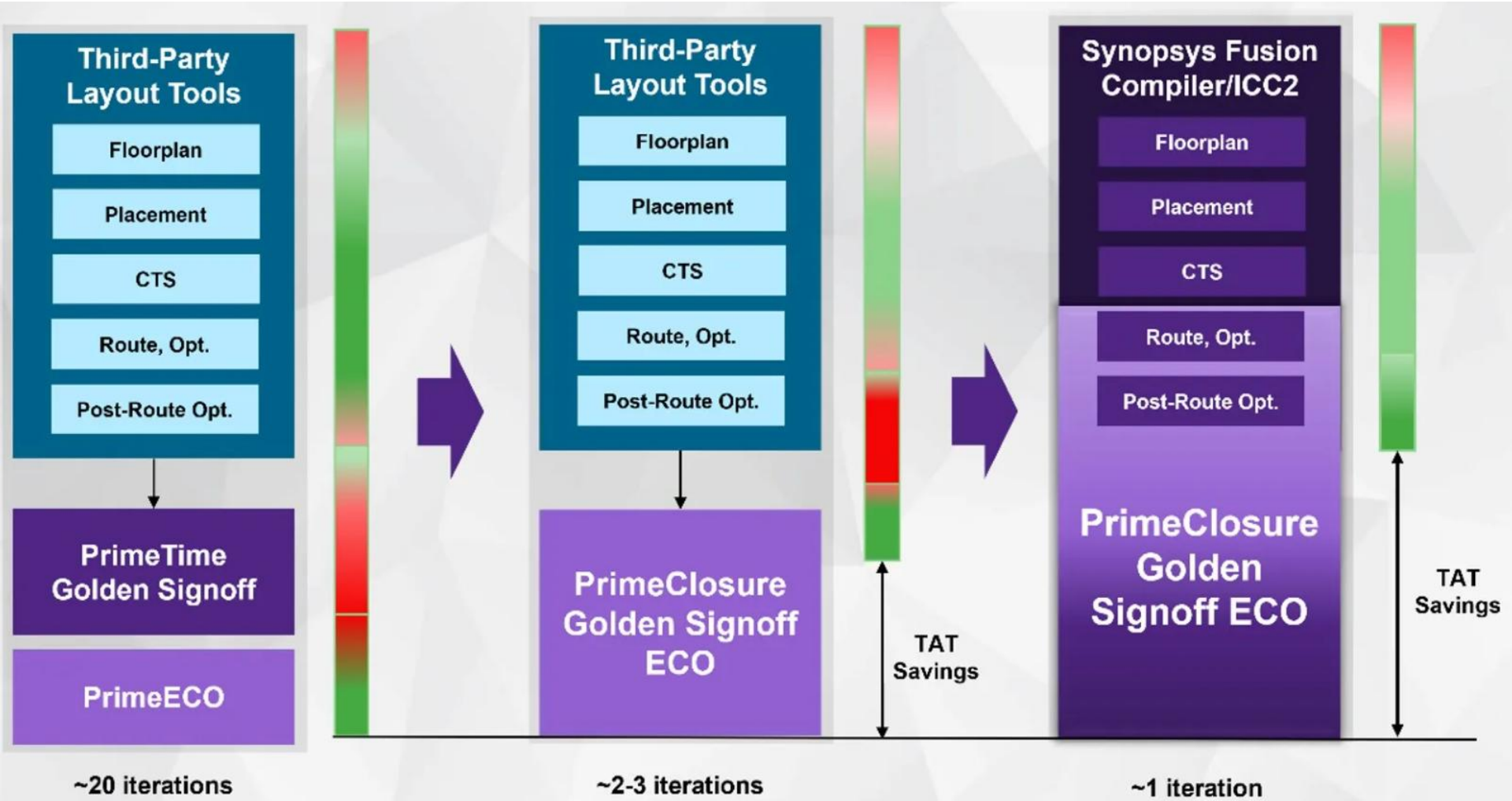
https://slideplayer.com/slide/1508976/

Engineering Change Order (ECO)

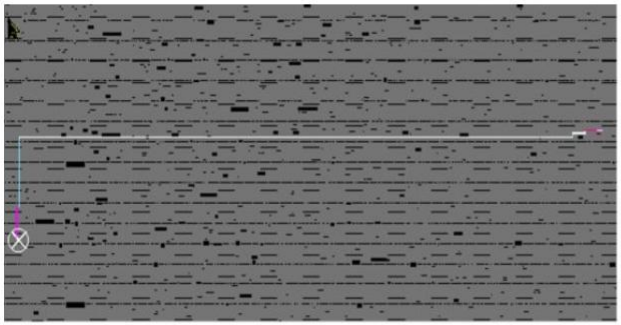
- After routing, we will extract RC and go through static timing analysis to fix timing errors or improve timing by
  - Cell sizing
  - Buffer insertion
  - Placement is impacted → ECO placement → minimize displacement
  - Routing is impacted → need to re-route
    - A question is: do we need to call global route for this ECO?
  - Base layers and metal layers are changed

- Other alternatives(?)
  - Metal ECO (https://blogs.sw.siemens.com/aprisa/2022/04/26/a-cure-for-eco-headaches-aprisa-automates-metal-eco/
    - Can we only change metal layers and keep base layers intact? Save time
    - Can new routing be completed in time?
  -

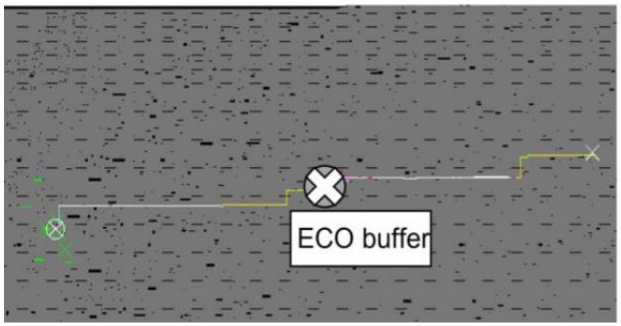

Layout
Floorplan
Placement
CTS
Route, Opt.
Post-Route Opt.
Signoff STA
ECO
Average Design Iteration >1 week
One Iteration TAT
• 2-3 days (>7nm)
• 4-6 days (<3nm)
One Iteration TAT
• 2-3 days (>7nm)
• 4-6 days (<3nm)
Implementation PPA
Timing Met
Signoff Reality
Timing Mismatch
ECO Recovery
Burden to Close Gap


Third-Party Layout Tools
Floorplan
Placement
CTS
Route, Opt.
Post-Route Opt.
PrimeTime Golden Signoff
PrimeECO
~20 iterations
Third-Party Layout Tools
Floorplan
Placement
CTS
Route, Opt.
Post-Route Opt.
PrimeClosure Golden Signoff ECO
~2-3 iterations
TAT Savings
Synopsys Fusion Compiler/ICC2
Floorplan
Placement
CTS
Route, Opt.
Post-Route Opt.
PrimeClosure Golden Signoff ECO
~1 iteration
TAT Savings

Metal Only ECO



Before ECO

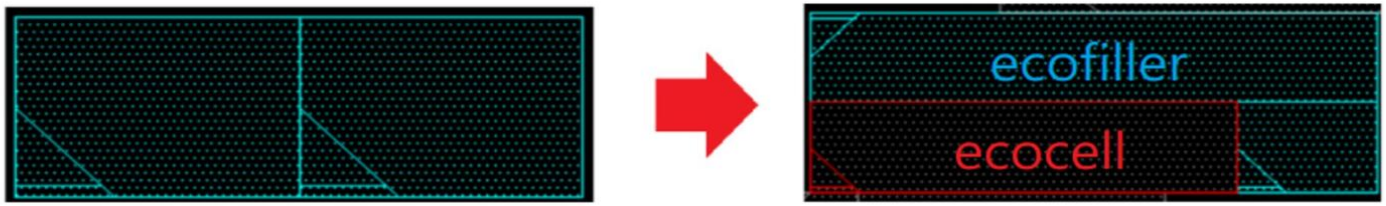


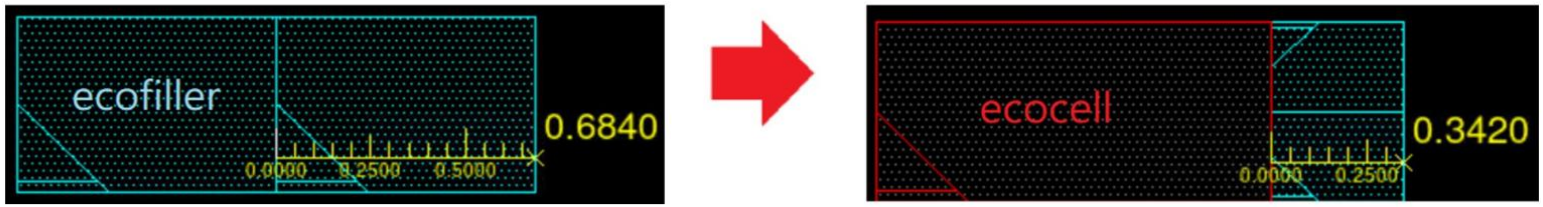
After ECO

- Using gate-array cells
  - GA filler/decap cell can be converted to any functional cells
  - ECO functional cells

Before ECO

After ECO





PCB Routing

https://resources.altium.com/p/right-auto-routing-algorithms-can-make-or-break-your-next-pcb

Auto-Interactive Routing: The Best of Both Worlds

The interactive portion of this type of tool allows the user to select specific routing points between the source and destination of an interconnect or signal net. The autorouter portion then routes traces automatically between these points. This allows the designer to inject their experience into the routing process while exploiting the primary time-saving feature of a typical autorouter.

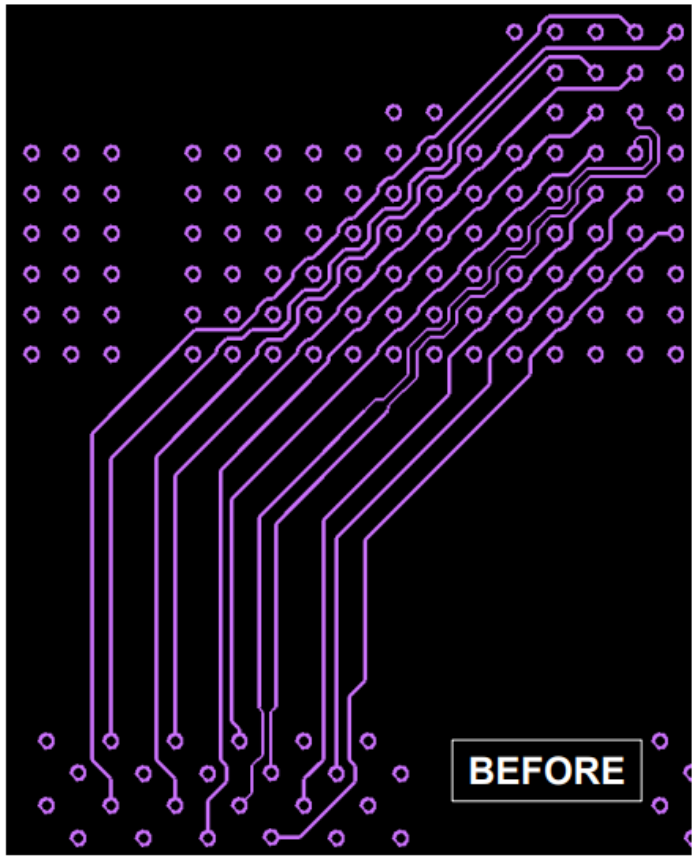
Routing DDR4 Interfaces

https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/pcb-design-analysis/pcb-west-2016-47-rte-ddr4-interfaces-cp.pdf

Delay Matching

Timing Challenge

Route tuning examples—Excess length added during matching

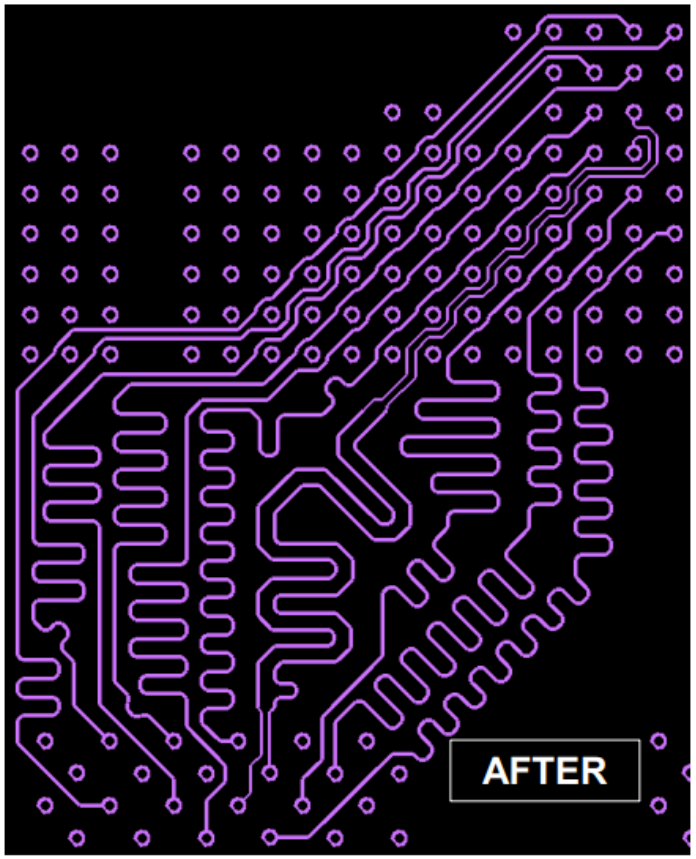


Match group
Tuned to *5mils* of each other

Overall length
1435mils

Board area
0.3128 sq. in

Blocked channels after tuning
~10 route paths



PCB West 2016 — Routing DDR4 Interfaces Quickly and Efficiently

cadence

Course Ordering Of The Major Topics

- Digital design flow
- CMOS logic gates and Boolean equations
- Algorithms and complexity
- Compaction
- Partitioning
- Floorplanning
- Placement
- Basic logic synthesis, technology mapping
- Routing
- Clock/PG routing & Elmore delay
- Simulation
- High level synthesis (next)