Detailed and Specialized Routing

- Course contents:
  - Introduction
  - Channel routing
  - Grid-based and gridless DR
  - Multilevel full-chip routing
  - Specialized routing
    - Clock net routing
    - P/G routing

Routing Introduction

placement

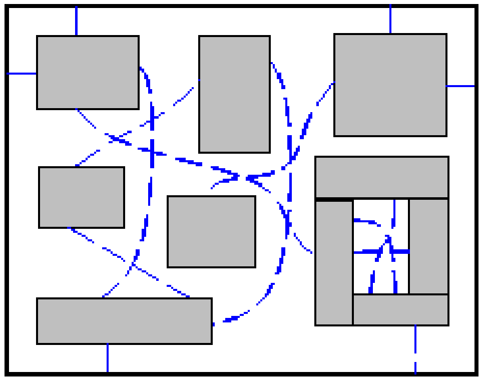
- Generates a "loose" route for each net.
- Assigns a list of routing regions to each net without specifying the actual layout of wires.

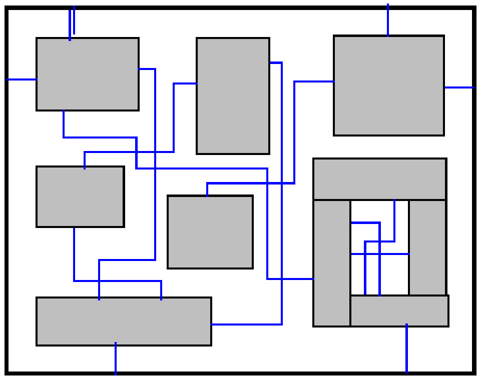
global routing

detailed routing

- Finds the actual geometric layout of each net within the assigned routing regions.

compaction


Global routing


Detailed routing

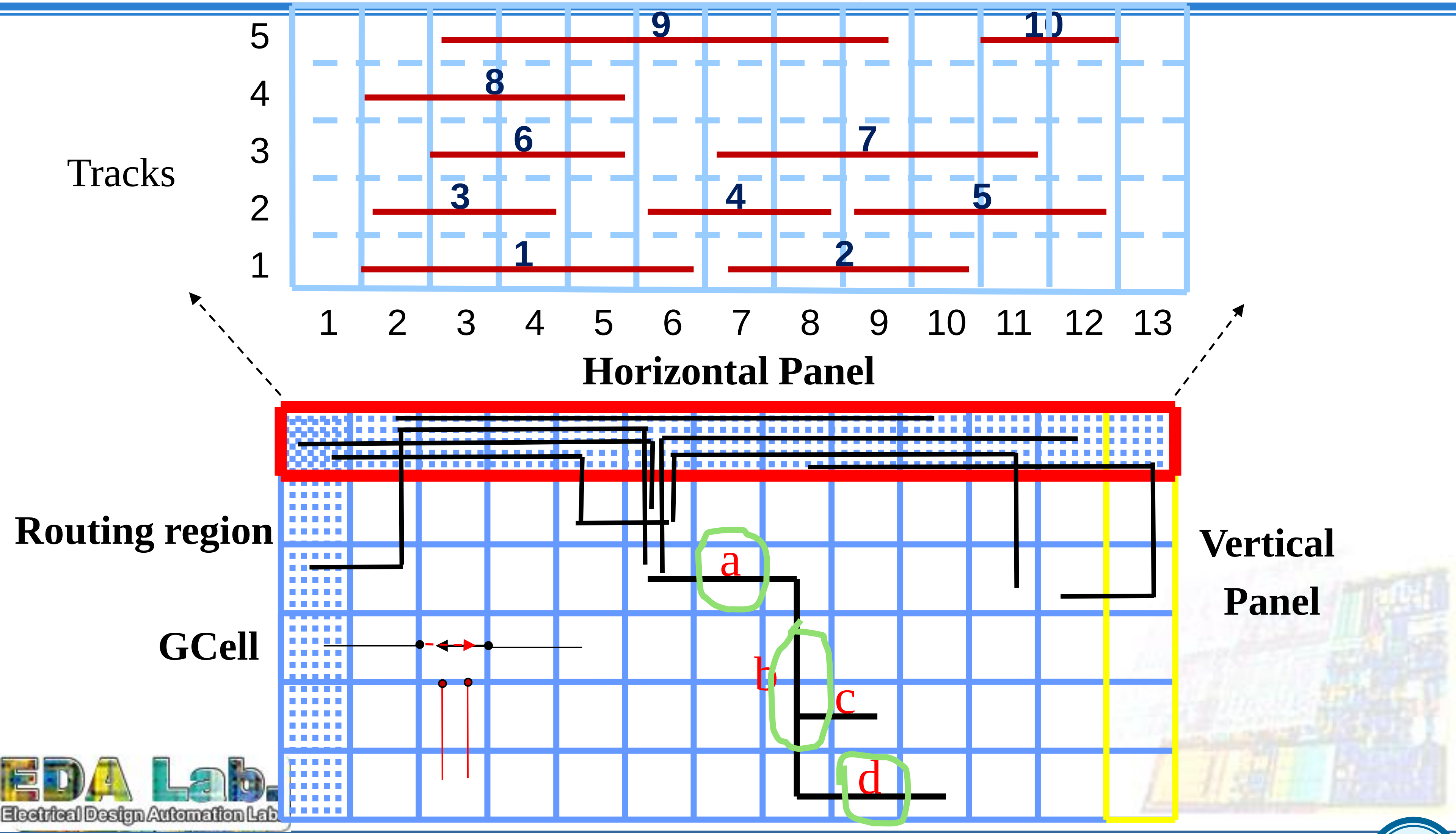
Track Assignment

- Track Assignment (TA) was proposed to help detailed routing reduce computation load and promote routing quality.
  - ✓ Speedup for the entire routing flow.
  - ✓ Better routing quality – straight paths.
- An objective for track assignment is that it should be fast and yet provide a good starting point for the detail router to complete with relative ease.

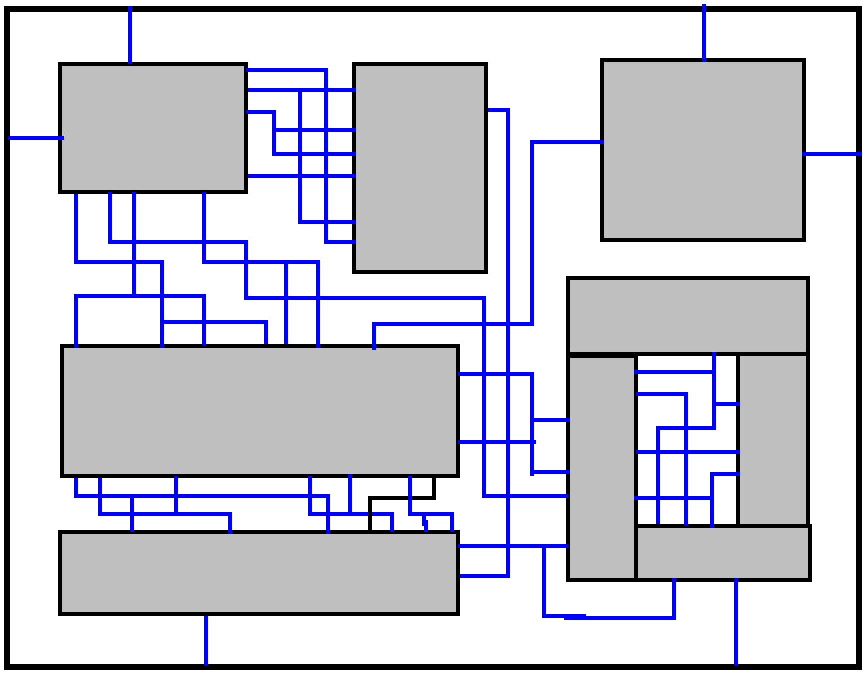
Track Assignment

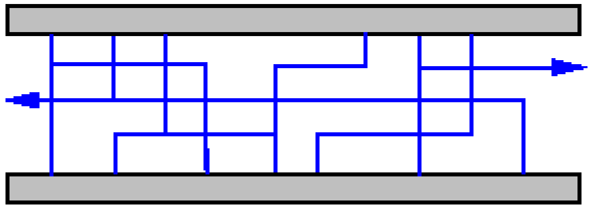
- *GCells* : global cells, a tile-structured region that is produced by partitioning original routing area before global routing.
- *Tracks* : a horizontal/vertical empty space on which a routed net can be placed.
- *IRoute* : a net segment that passes through at least one *GCell*.
- *Panel* : it is composed of a series of *GCells* in a row or a column.

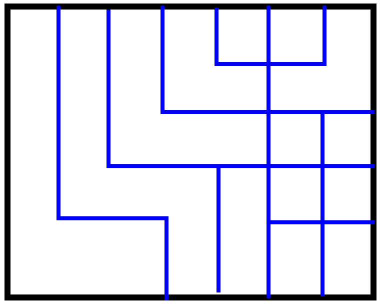
Track Assignment



Channel/Switchbox Routing


Detailed routing


channel routing


switchbox routing

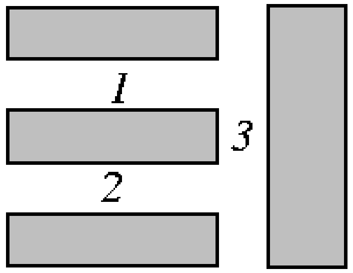
Order of Routing Regions and L-Channels

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

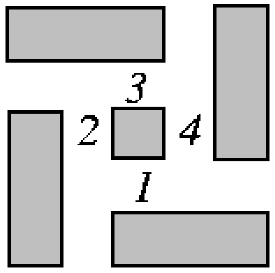
(b) No ordering is possible to avoid conflicts.

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

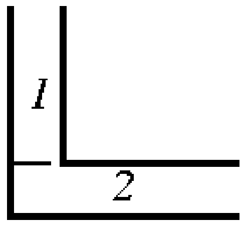
(d) An L-channel can be decomposed into two channels and a switchbox.



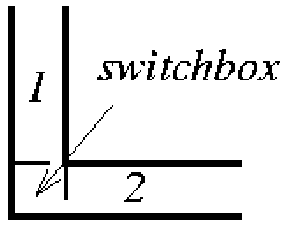
(a)



(b)



(c)



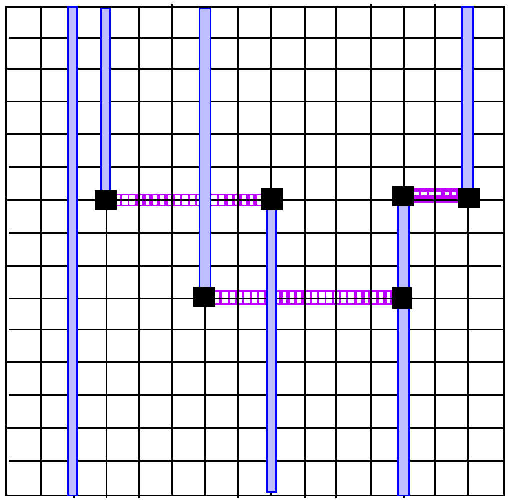
(d)

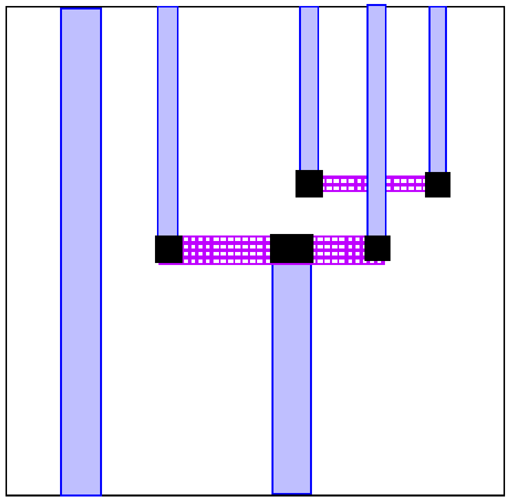
Routing Considerations

- Number of terminals (two-terminal vs. multi-terminal nets)
- Net widths (power and ground vs. signal nets)
- Via restrictions (stacked vs. conventional vias)
- Boundary types (regular vs. irregular)
- Number of layers (two vs. three, more layers?)
- Net types (critical vs. non-critical nets)

Detailed Routing Models

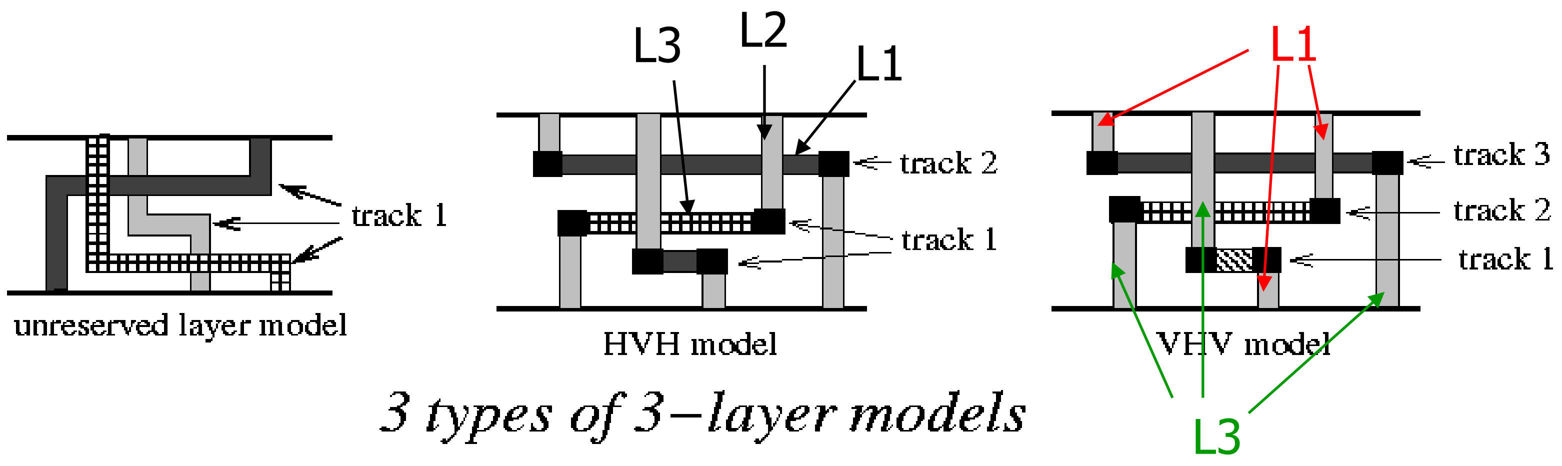
- **Grid-based model:**
  - A grid is super-imposed on the routing region.
  - Wires follow paths along the grid lines.
  - **Pitch:** distance between two gridded lines
- **Gridless model:**
  - Any model that does not follow this "gridded" approach.


grid–based


gridless

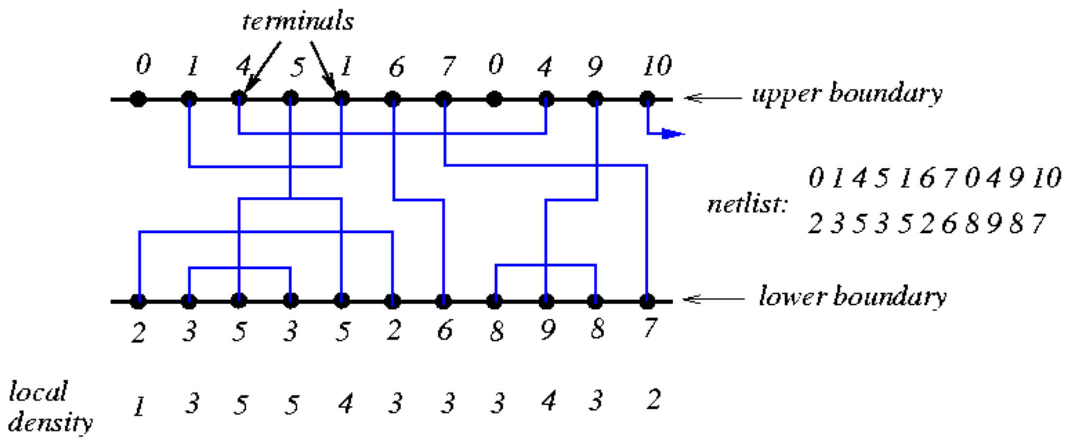
Models for Multi-Layer Routing

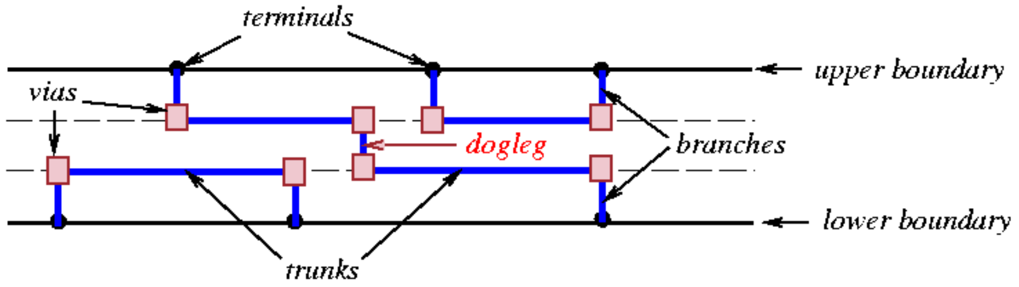
- **Unreserved layer model:** Any net segment is allowed to be placed in any layer.
- **Reserved layer model:** Certain type of segments are restricted to particular layer(s).
  - Two-layer: HV (horizontal-Vertical), VH
  - Three-layer: HVH, VHV (HVH preferred: min channel height)



3 types of 3−layer models

Terminology for Channel Routing Problems





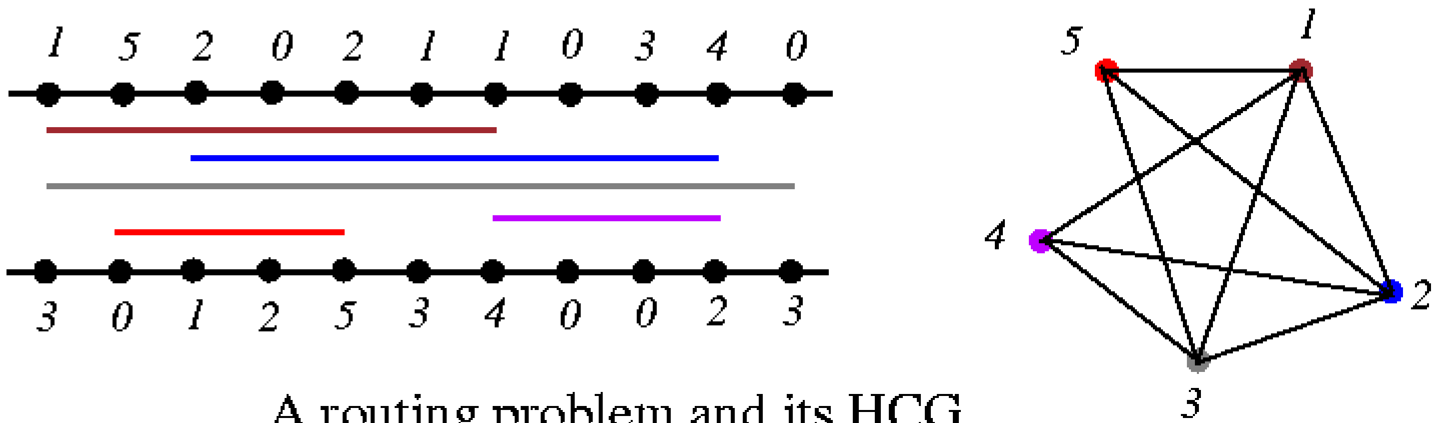
- Local density at column *i, d(i)*: total # of nets that crosses column *i*.
- Channel density: maximum local density
  - # of horizontal tracks required ≥ channel density.

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. (HCG)
  - 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 the other net. (VCG)
- Objective: Channel height is minimized (i.e., channel area is minimized).

Horizontal Constraint Graph (HCG)

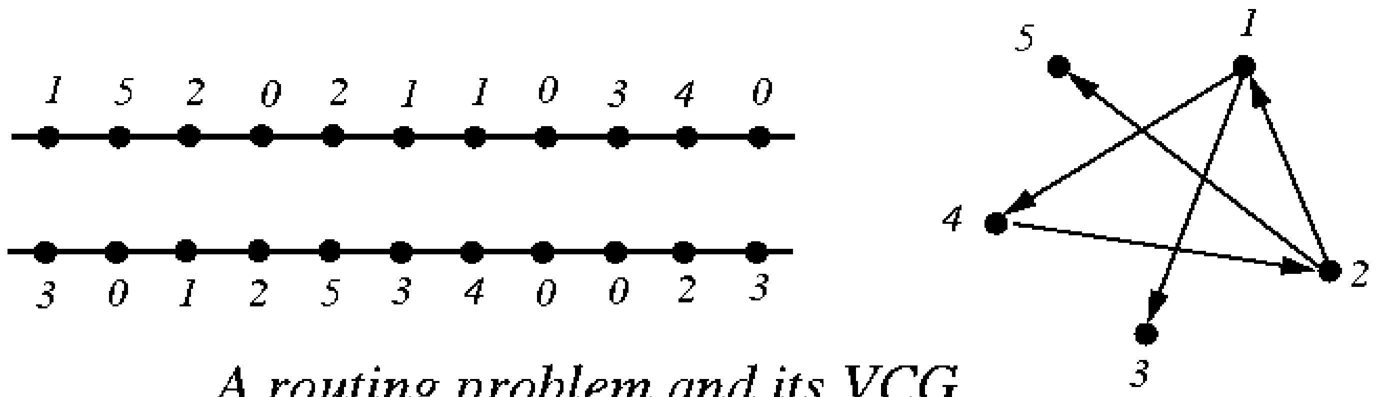
- HCG $G = (V, E)$ is **undirected** graph where
  - $V = \{ v_i \mid v_i$ represents a net $n_i\}$
  - $E = \{(v_i, v_j) \mid$ a horizontal constraint exists between n_i and $n_j\}$.
- For graph G: vertices ⇔ nets; edge (i, j) ⇔ net i overlaps net j.



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-71.
- **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).

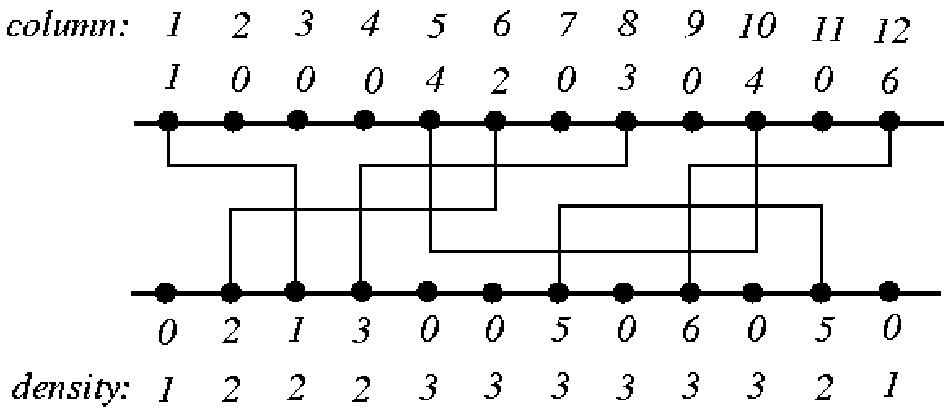
Basic Left-Edge Algorithm

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.

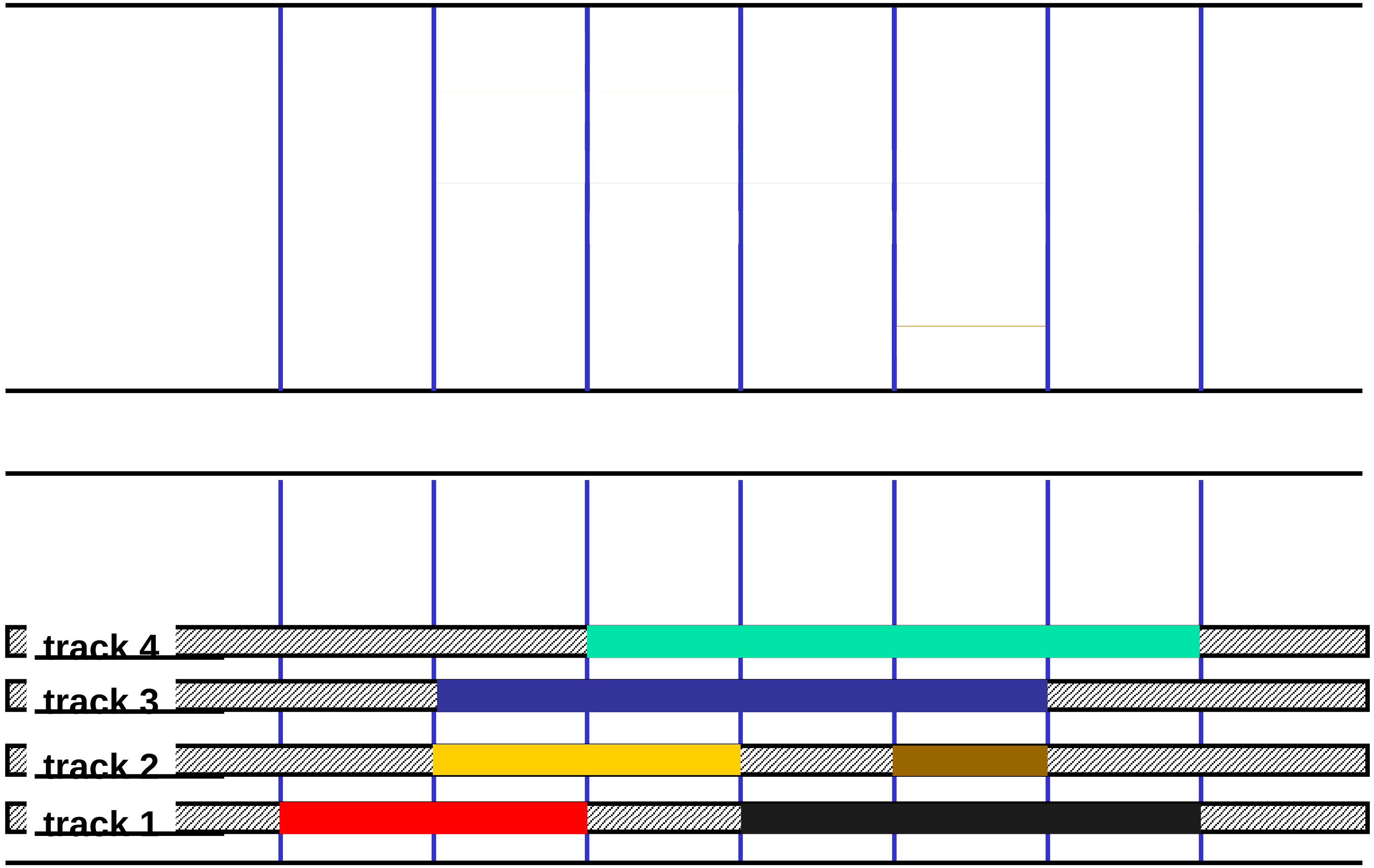
```
1 begin
2 $U \leftarrow \{I_1, I_2, \ldots, I_n\}$;
3 $t \leftarrow 0$;                // t: track #
4 while ($U \neq \emptyset$) do
5     $t \leftarrow t + 1$;
6     $watermark \leftarrow 0$;
7     while (there is an $I_j \in U$ s.t. $s_j > watermark$) do
8         Pick the interval $I_j \in U$ with $s_j > watermark$,
          nearest $watermark$;
9         $track[j] \leftarrow t$;
10      $watermark \leftarrow e_j$;
11      $U \leftarrow U - \{I_j\}$;
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

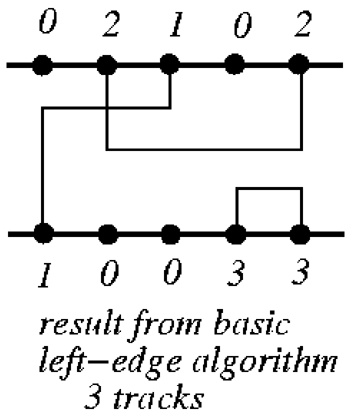

column: 1 2 3 4 5 6 7 8 9 10 11 12
1 0 0 0 4 2 0 3 0 4 0 6
0 2 1 3 0 0 5 0 6 0 5 0
density: 1 2 2 2 3 3 3 3 3 3 2 1

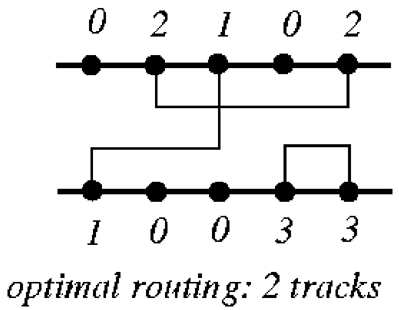
Basic Left-Edge Example

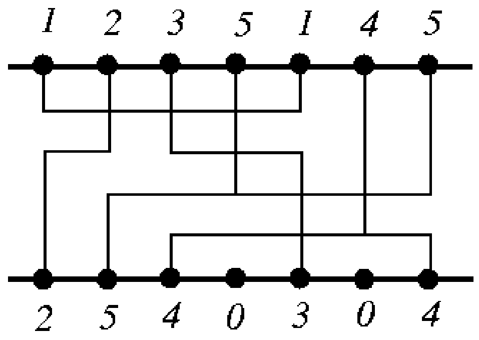


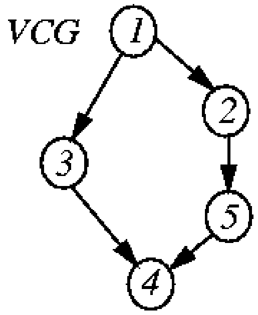
Problems in Basic Left-Edge Algorithm

- 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.









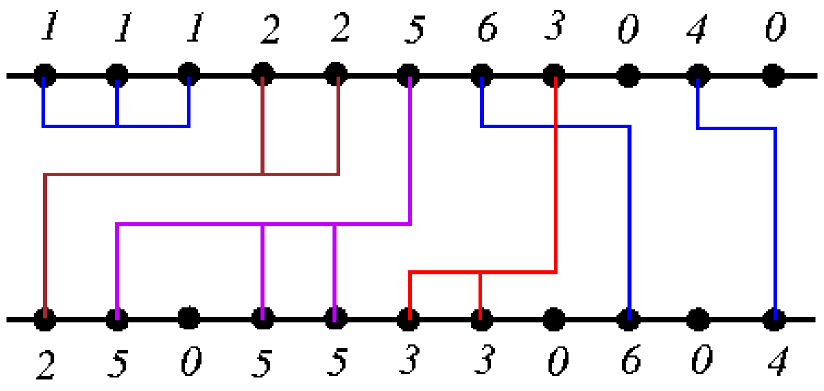
Constrained Left-Edge Algorithm (Persky 1976)

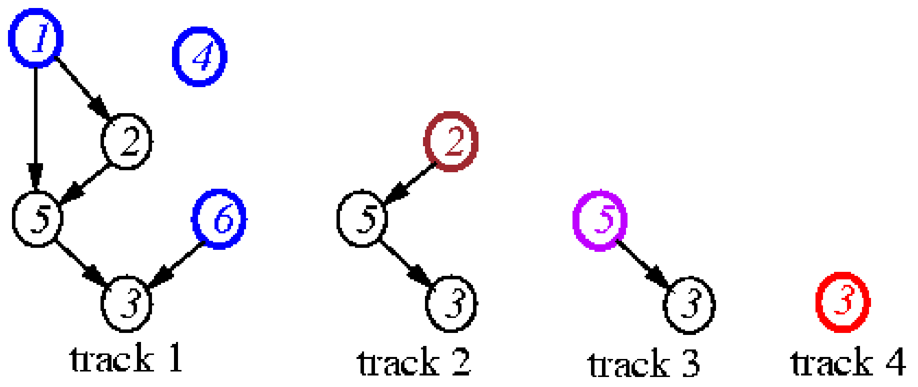
Algorithm: Constrained_Left-Edge(U, $track[j]$**)**
U: set of unassigned intervals (nets) $I_1, \ldots, I_n$;
$I_j=[s_j, e_j]$: interval j with left-end x-coordinate s_j and right-end e_j;
$track[j]$: track to which net j is assigned.

```
1 begin
2 $U \leftarrow \{ I_1, I_2, \ldots, I_n\}$;
3 $t \leftarrow 0$;
4 while ($U \neq \emptyset$) do
5     $t \leftarrow t + 1$;
6     $watermark \leftarrow 0$;
7     while (there is an unconstrained $I_j \in U$ s.t. $s_j > watermark$) do
8         Pick the interval $I_j \in U$ that is unconstrained,
          with $s_j > watermark$, nearest $watermark$;
9         $track[j] \leftarrow t$;
10        $watermark \leftarrow e_j$;
11        $U \leftarrow U - \{I_j\}$;
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.

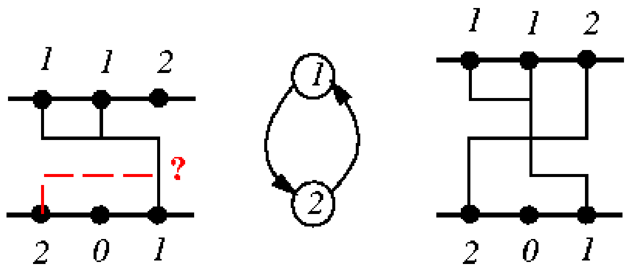




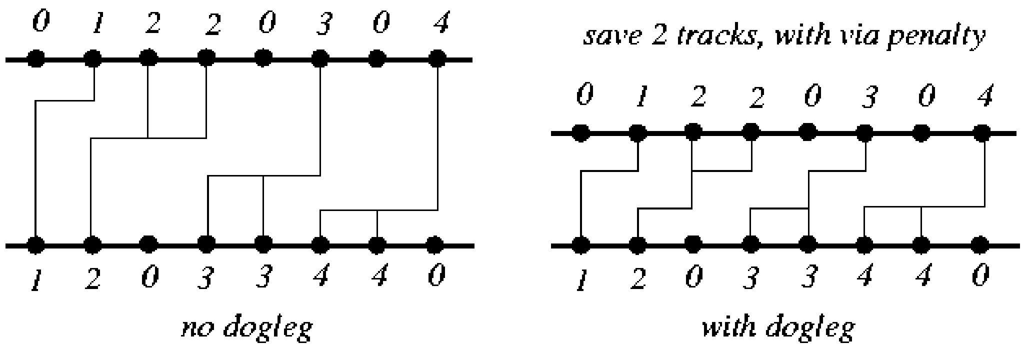
Note: In text, it is run backward (start from the bottom)

Dogleg Channel Router (1/3)

- Deutsch, “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. (splitting of horizontal segments)



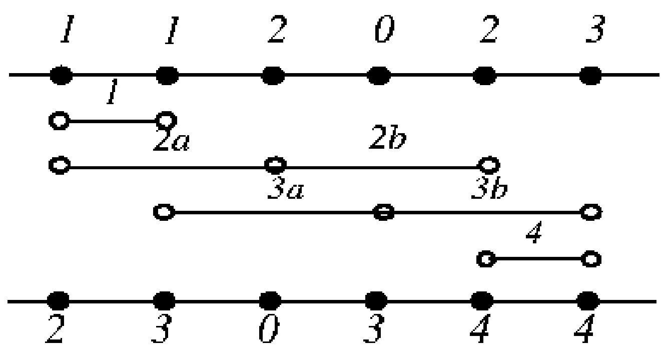
- **Drawback of Left-Edge: the entire net is on a single track.**
  - **Doglegs** are used to place parts of a net on different tracks to minimize channel height.
  - Might incur penalty for additional vias.

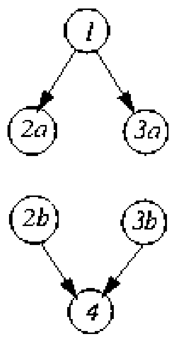


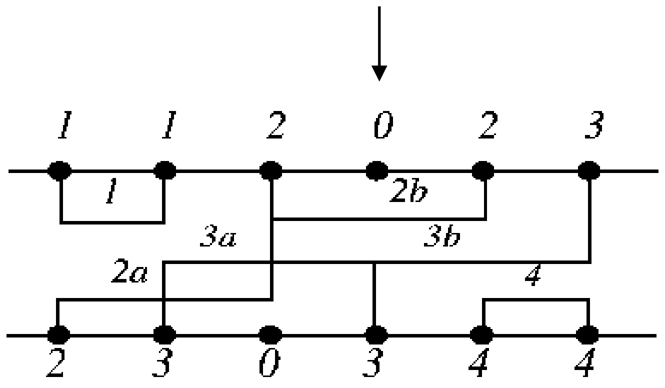
Dogleg Channel Router (2/3)

- Each multi-terminal net is broken into a set of 2-terminal nets.
- Modified Left-Edge Algorithm is applied to each subnet.

Selecting a horizontal segment that has no descendants for the placement in the bottom tracks, and segments that have no ancestors for placement in the top tracks

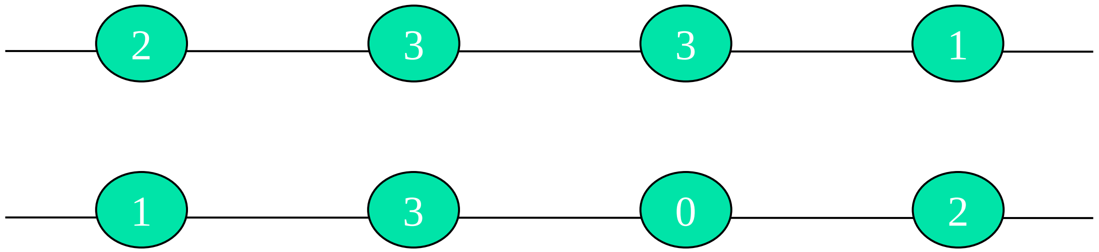






Dogleg Channel Router (3/3)

- Drawback - cannot solve general cyclic VCG
  - There is no intermediate terminal to break the h-segment

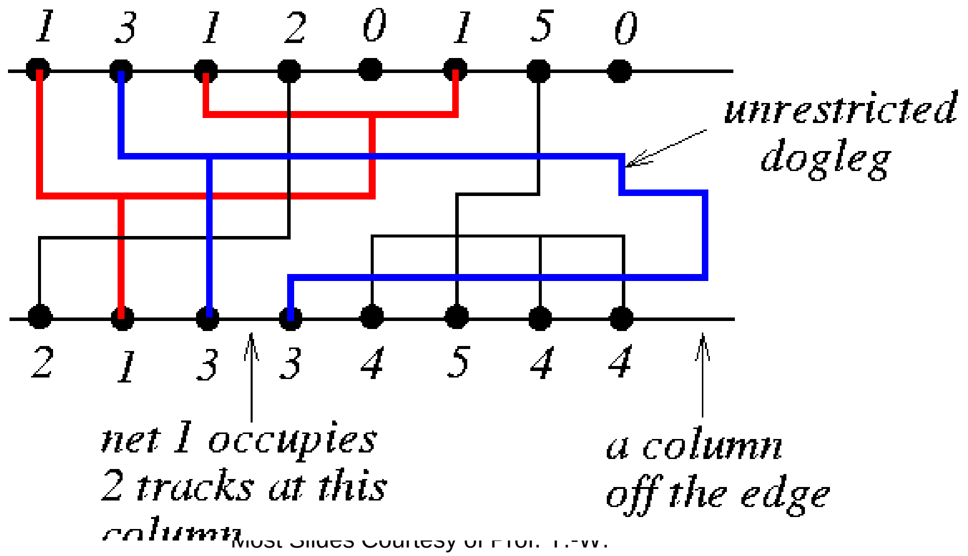


Yoshimura-Kuh (YK) Algorithm

- Yoshimura & Kuh, "Efficient algorithms for channel routing," IEEE TCAD, Jan. 1982.
- YK algorithm considers both HCG and VCG.
  - Reports better results than the dogleg algorithm
- Nets are assigned to minimize the effect of vertical constraint chains in VCG.
- Does not allow **"unrestricted"** doglegs and cannot handle vertical constraint cycles.
- Algorithm consists of:
  - Zone representation of horizontal segments. (from HCG)
  - Merging of nets. (minimizing channel density)
    - Will not bring cyclic VCG when original VCG has no cycles
  - Apply left-edge algorithm and assign horizontal tracks
- Same idea can be extended to three-layer channel routing (Chen & Liu, IEEE TCAD, 1984)

Greedy Channel Router

- Rivest & Fiduccia, “A greedy channel router,” *DAC-82*, (*IEEE TCAD*, May 1983).
- Always succeed (even if cyclic conflict is present)
- Allows unrestricted dogleg
- Allows a net to occupy more than 1 track at a given column.
- May use a few columns off the edge.

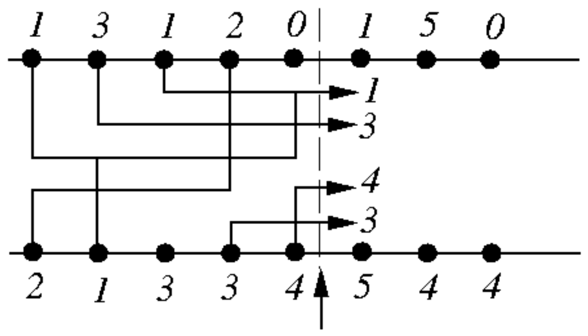


Overview of Greedy Router

- Left-to-right, column-by-column scan.

```
1 begin
2 $c \leftarrow 0$;
3 while (not done) do
4     $c \leftarrow c + 1$;
5     Complete wiring at column $c$;
6 end
```

- In general, a net may be
  1. empty (net 5)
  2. unsplit (nets 1, 4)
  3. split (net 3)
  4. completed (net 2)

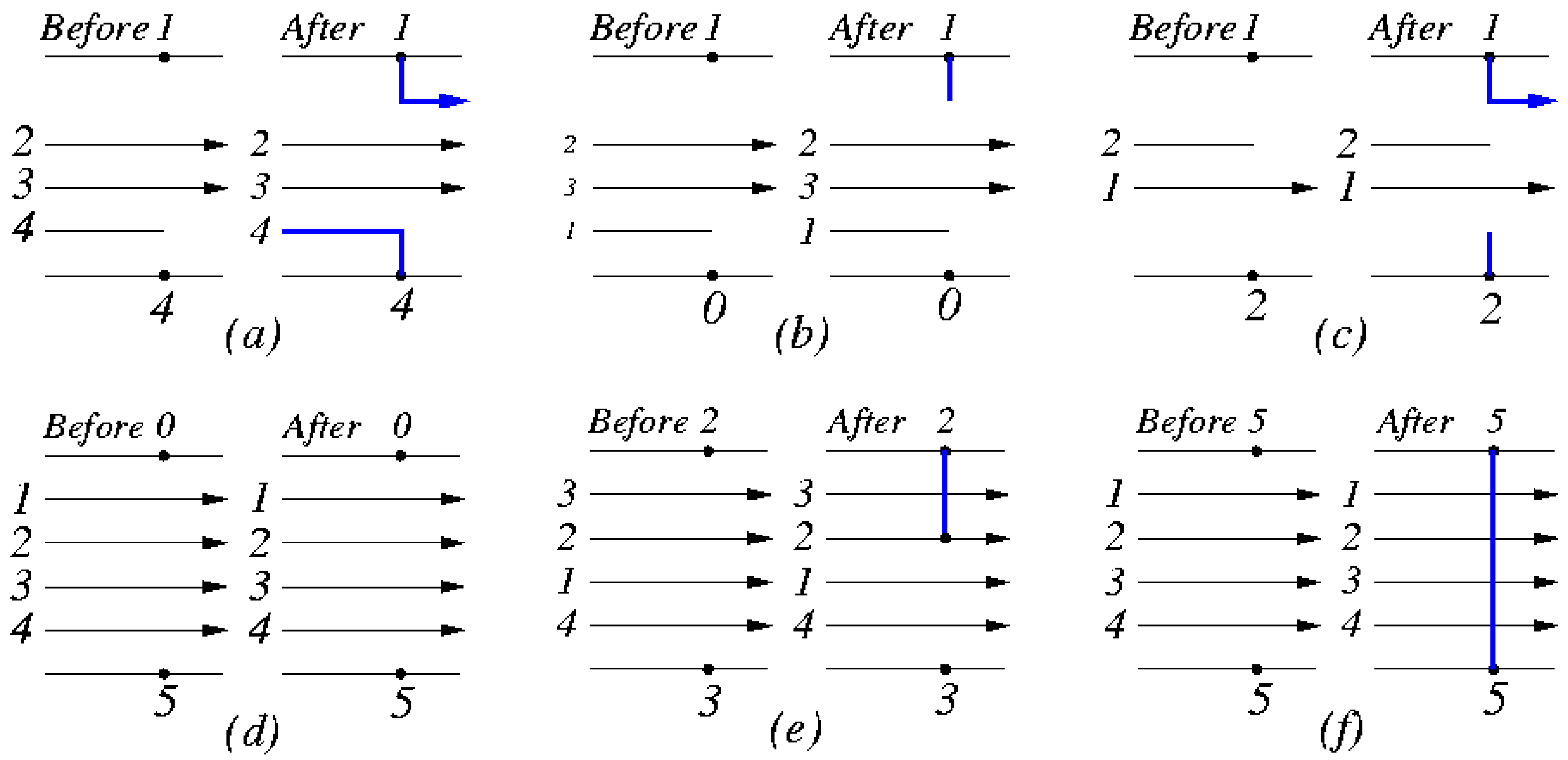


Greedy Heuristics

- At each column, the greedy router tries to maximize the utility of the wiring produced:
  - A: Make minimal feasible top/bottom connections;
  - B: Collapse split nets;
  - C: Move split nets closer to one another;
  - D: Raise rising nets/Lower falling nets;
  - E: Widen channel when necessary;
  - F: Extend to next column.

A: Make Minimal Feasible Top/Bottom Connections

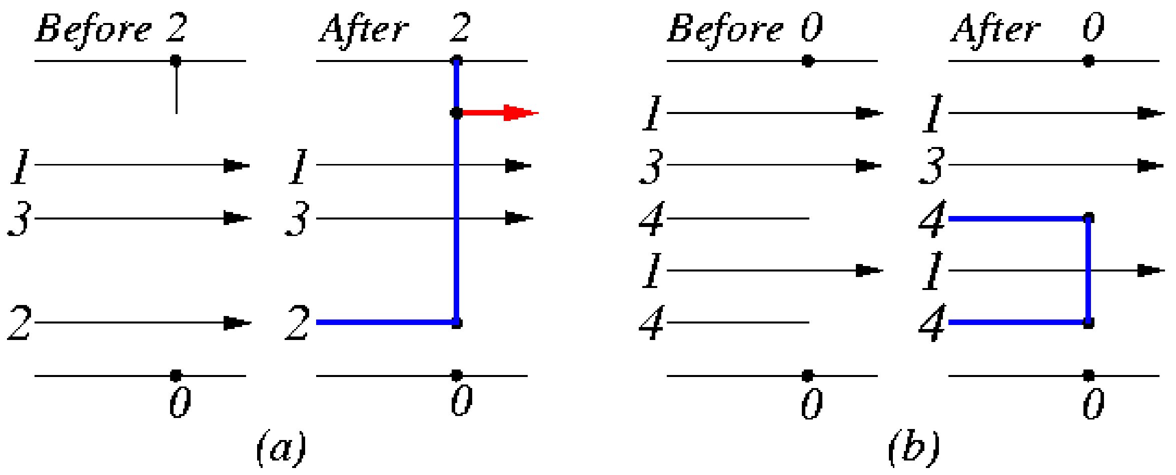
Brought to empty tracks



Deferred untill step E

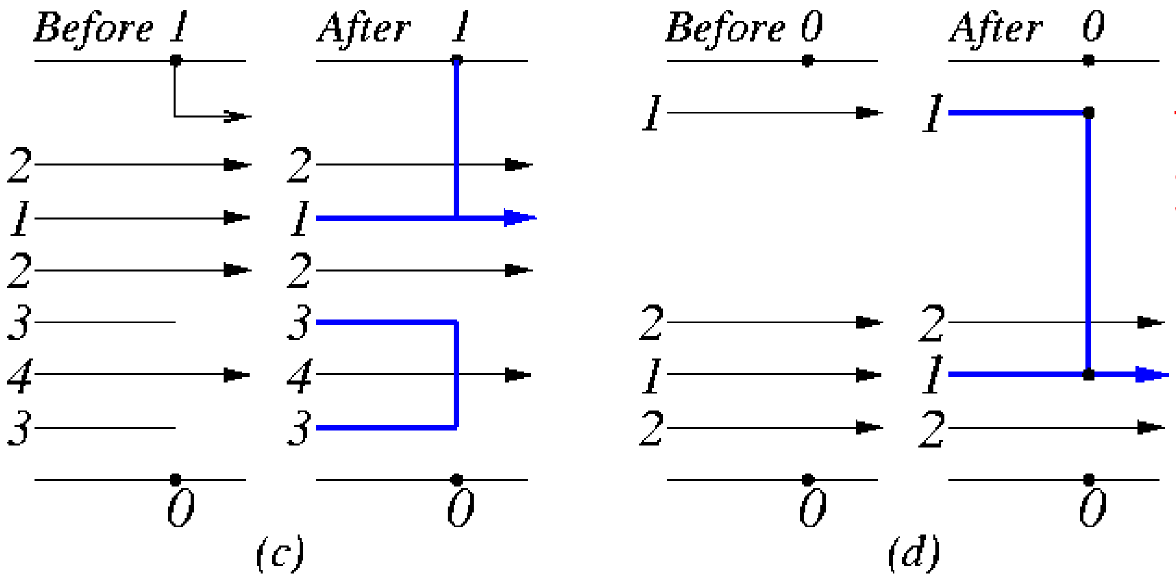
B: Collapse Split Nets

- To free up the most tracks



when there are ties

Keep free area as close to the sides as possible

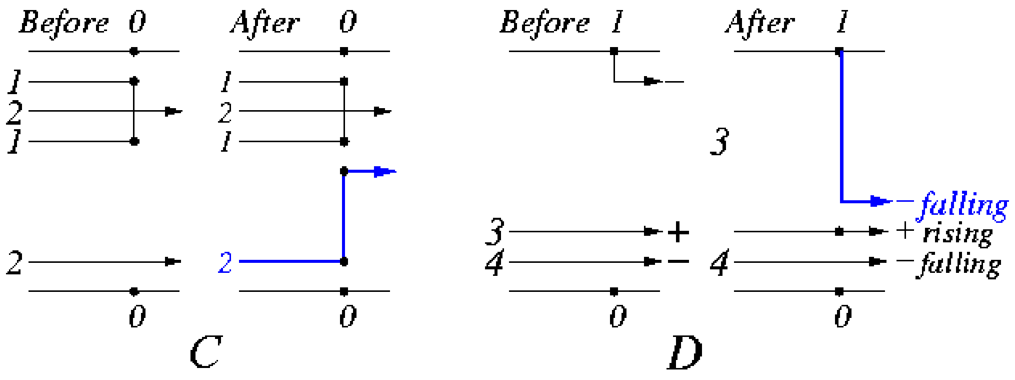


Try to maximize amount of vertical wire (that is to free max horizontal tracks)

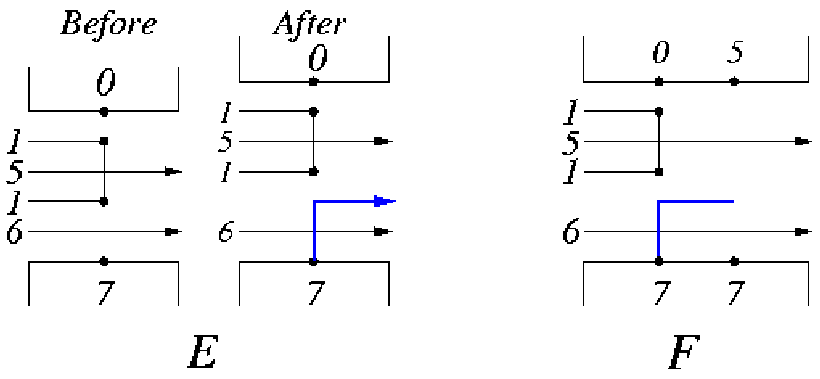
Heuristics: C, D, E, and F

- C: Move split nets closer to one another;
- D: Raise rising nets/Lower falling nets;
- E: Widen channel when necessary;
- F: Extend to next column.

Add jogs



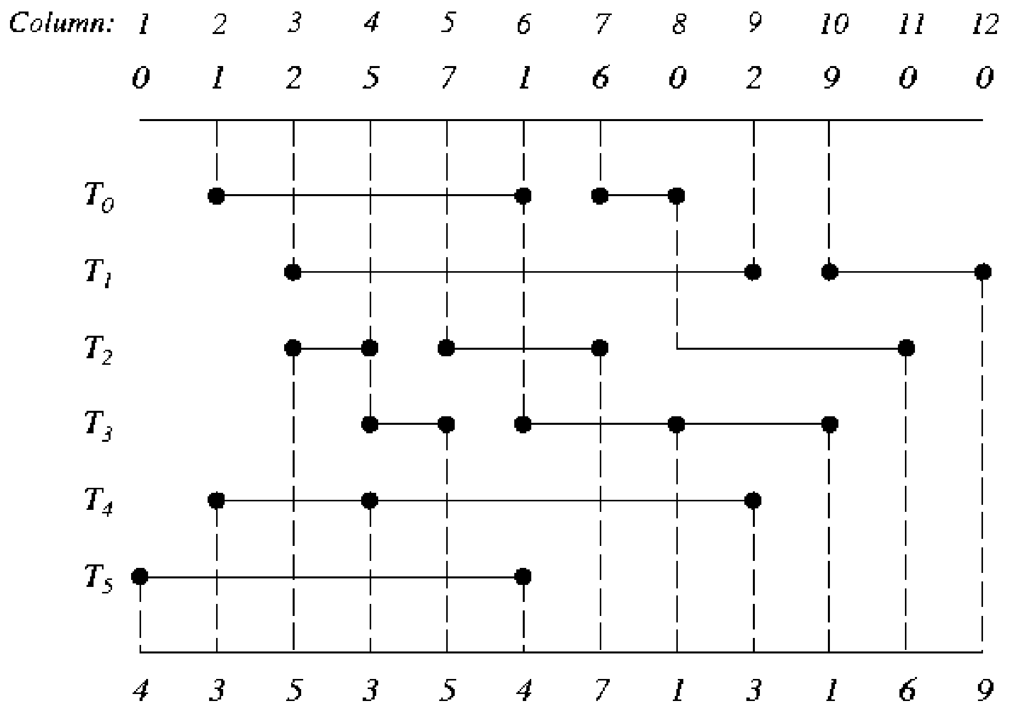
New tracks placed at the center of the channel



split net

unfinished net

Greedy Routing Example



- C_3: Connect pin 5 to $T_3 \rightarrow$ Jog net 5 from T_3 to T_2 (since net 5 is rising).
- C_4: Connect pin 5 to $T_2 \rightarrow$ Jog net 5 from T_2 to T_3 (since net 5 is falling).
- C_6: Connect pin 1 to $T_0 \rightarrow$ Jog net 1 from T_0 to T_3 (since net 1 is falling).
- C_7: Connect pin 7 to $T_5 \rightarrow$ Merge tracks T_2 and T_5 (last pin 7).
- C_8: Connect pin 1 to $T_5 \rightarrow$ Jog net 6 from T_0 to T_2 and net 1 from T_5 to T_3.
- … … … … … …

Comparison of Two-Layer Channel Routers

	Left-edge	Dogleg	YK	Greedy	**Robust**
Layer assignment	reserved	reserved	reserved	reserved	reserved
Dogleg	not allowed	allowed	allowed (restricted)	allowed	allowed
Vertical constraint	not allowed	allowed	allowed	allowed	allowed
Cyclic constraint	not allowed	not allowed	not allowed	allowed	allowed

Robust: Yoeli, "A robust channel router," IEEE TCAD, 1991.

Comparison of Two-Layer Channel Routers

- Comparison using the benchmark example: **Deutsch's "difficult example."**
- Channel density: 19

Routers	Tracks	Vias	wire length
Left-edge	31	290	6526
Dogleg	21	346	5331
YK	20	403	5381
Greedy	20	329	5078
Hierarchical	19	336	5023
YACR2	19	287	5020
Robust	19	319	4961

YACR2: J. Reed et.al., "A new symbolic channel router: YACR2," TCAD 1985

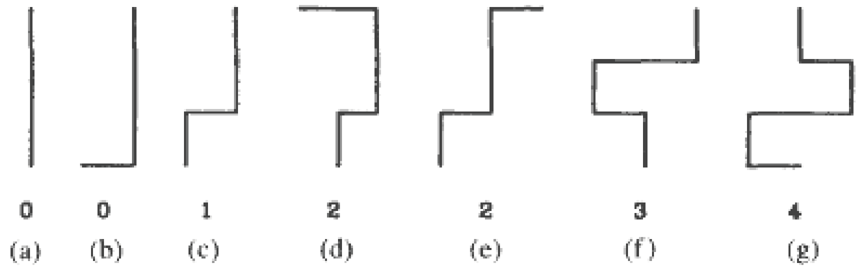
Grid Based Detailed Routing - Mighty

- Deterministic scheme: MIGHTY
  - Minimum cost path finder
    - Find a path to connect any two components of a net and push the on-the-fly path to the heap
  - Path Conformer
    - Realize, if feasible, the path popped from the heap
    - If not feasible, find a new path considering existing obstacles
  - Modification – if the path found by path conformer is not good
    - Weak modification
      - Push existing paths
    - Strong modification
      - Rip up existing paths

®H. Shin, Alberto Sangiovanni-vincentelli, "A Detailed Router Based on Incremental Routing Modifications: Mighty" IEEE Trans. CAD, 1987

Grid Based Detailed Routing - Mighty

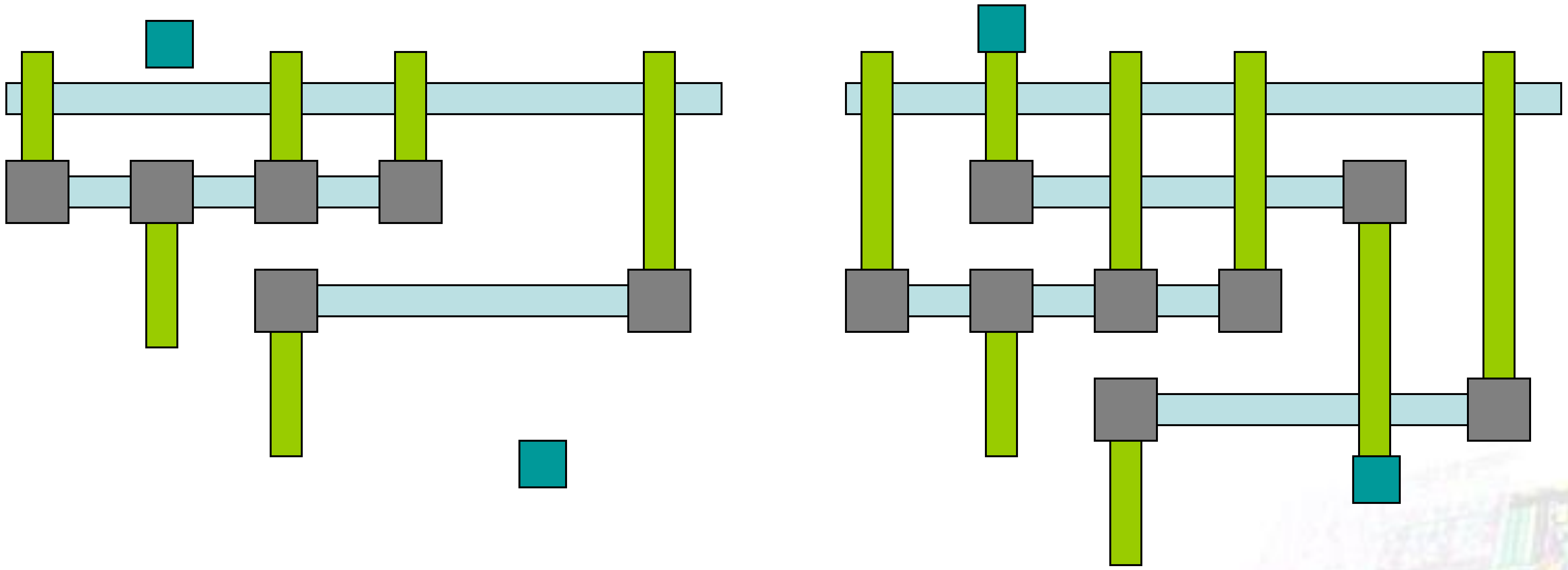
- Whether a found path is good is judged by the number of bends.
  - Floating segment – a segment that is not connected to a pin or a pseudopin.
  - Good path – a path which has less than n floating segments. (3 ≤ n ≤5)



The number of floating segments of several cases. With $n = 3$, (a) to (e) are good paths, while (f) to (g) are bad paths.

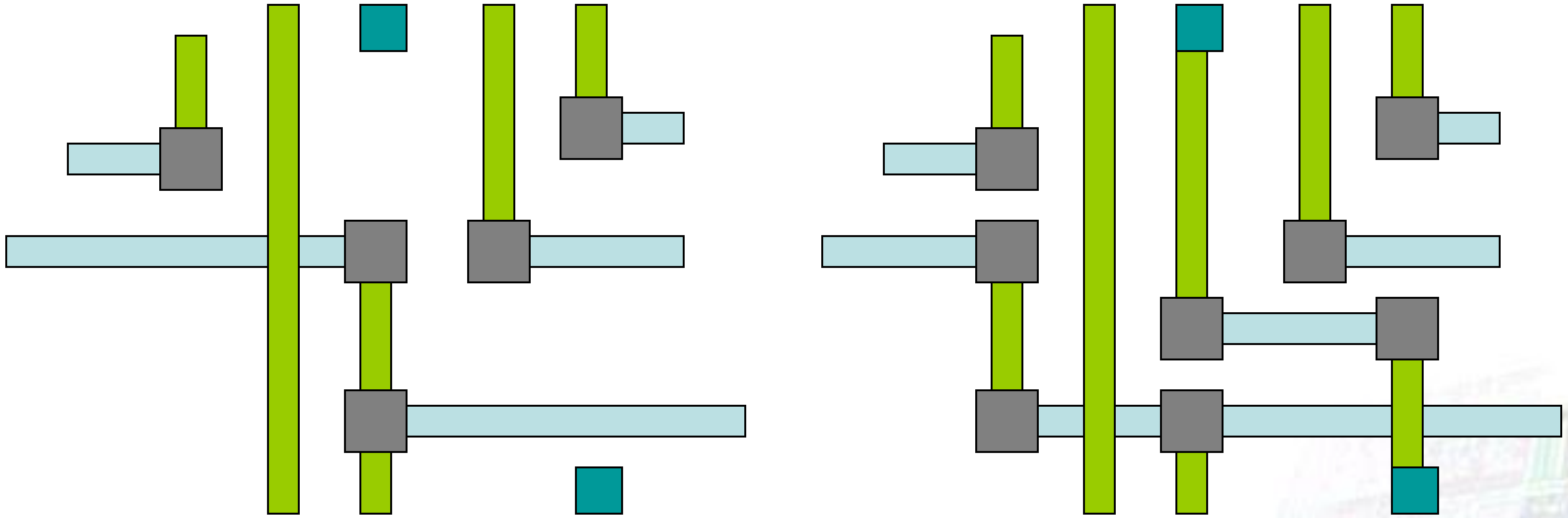
Grid Based Detailed Routing - Mighty

- Weak modification (I)



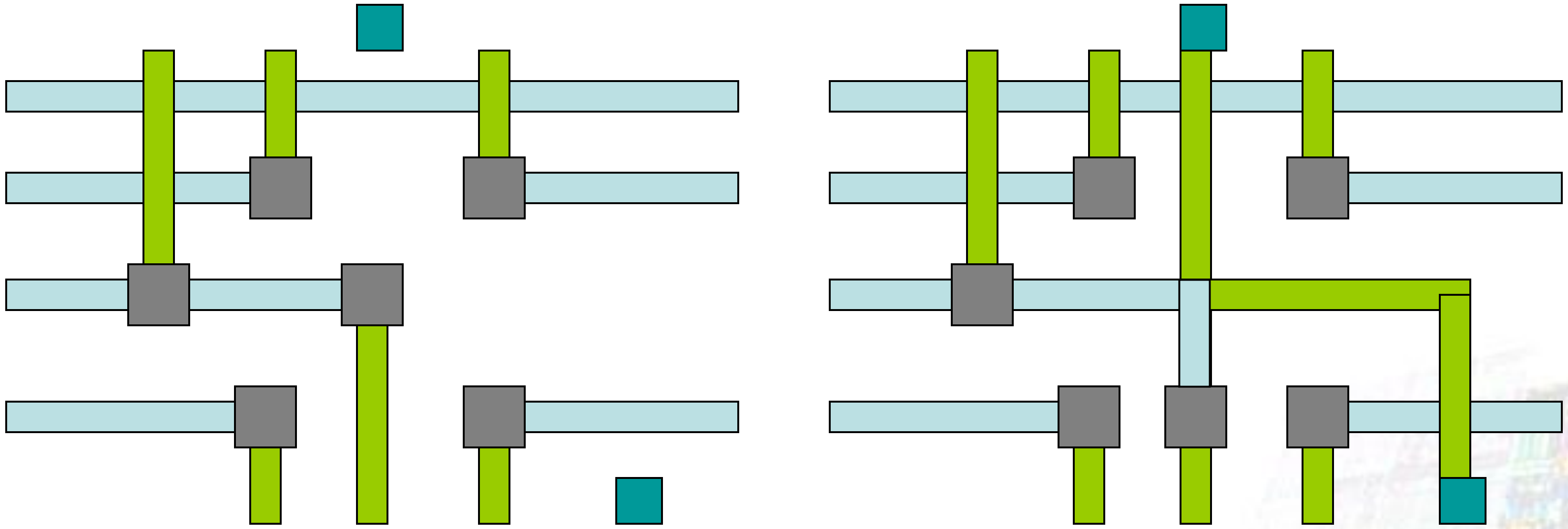
Grid Based Detailed Routing - Mighty

- Weak modification (II)



Grid Based Detailed Routing - Mighty

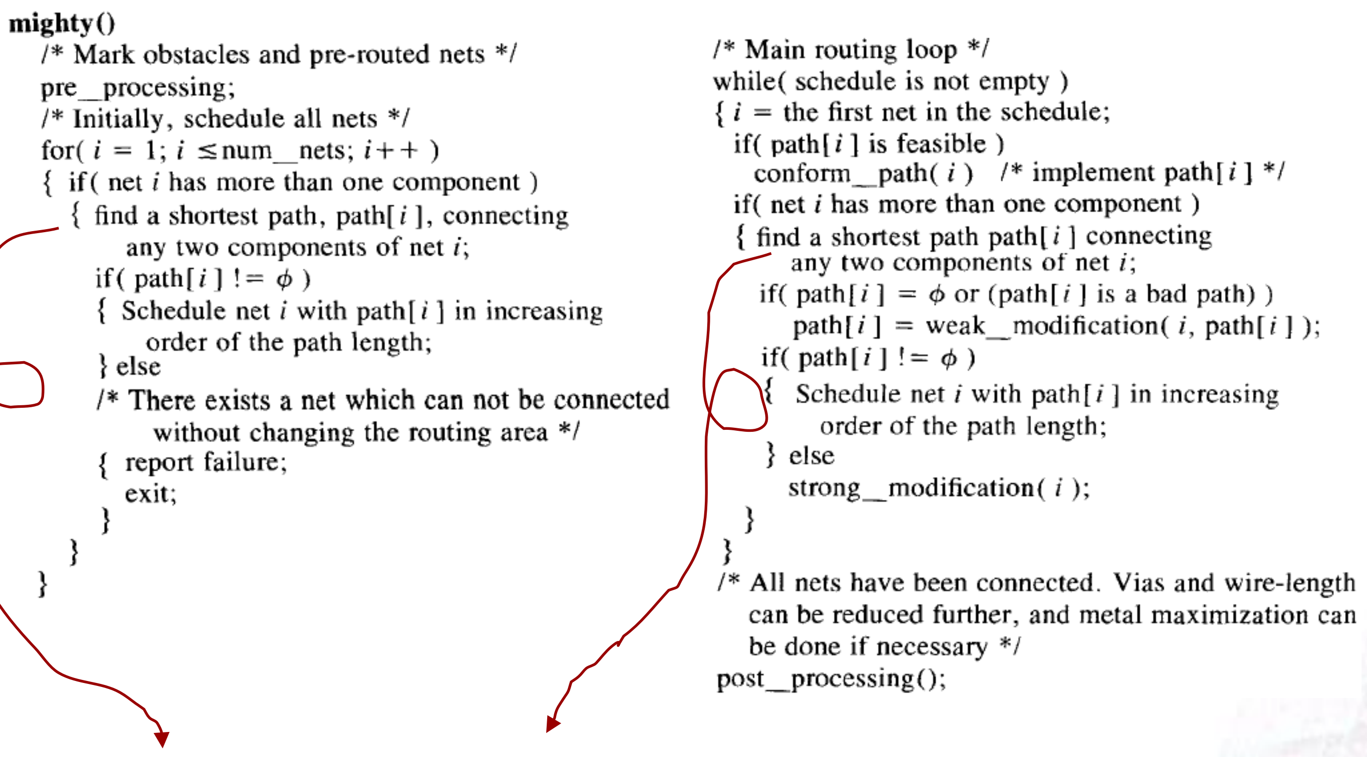
- Weak modification (III)



Grid Based Detailed Routing - Mighty

- Strong modification for net i
  - Find two closest pins of blocked net i
  - Find a minimum-cost rip-up path connecting two pins
  - Rip up all connections of the nets
  - Re-schedule the ripped-up nets
  - Conform the found path and iterate routing and rip-up (if necessary) to complete the routing of net i

Grid Based Detailed Routing - Mighty



```
mighty()
   /* Mark obstacles and pre-routed nets */
   pre_processing;
   /* Initially, schedule all nets */
   for( $i$ = 1; $i$ ≤num_nets; $i$++ )
   {  if( net $i$ has more than one component )
      {  find a shortest path, path[$i$], connecting
            any two components of net $i$;
         if( path[$i$] != $\phi$ )
         {  Schedule net $i$ with path[$i$] in increasing
               order of the path length;
         } else
         /* There exists a net which can not be connected
               without changing the routing area */
         {  report failure;
            exit;
         }
      }
   }

/* Main routing loop */
while( schedule is not empty )
{ $i$ = the first net in the schedule;
  if( path[$i$] is feasible )
     conform_path( $i$ )   /* implement path[$i$] */
  if( net $i$ has more than one component )
  { find a shortest path path[$i$] connecting
        any two components of net $i$;
     if( path[$i$] = $\phi$ or (path[$i$] is a bad path) )
        path[$i$] = weak_modification( $i$, path[$i$] );
     if( path[$i$] != $\phi$ )
     {  Schedule net $i$ with path[$i$] in increasing
           order of the path length;
     } else
        strong_modification( $i$ );
  }
}
/* All nets have been connected. Vias and wire-length
   can be reduced further, and metal maximization can
   be done if necessary */
post_processing();
```

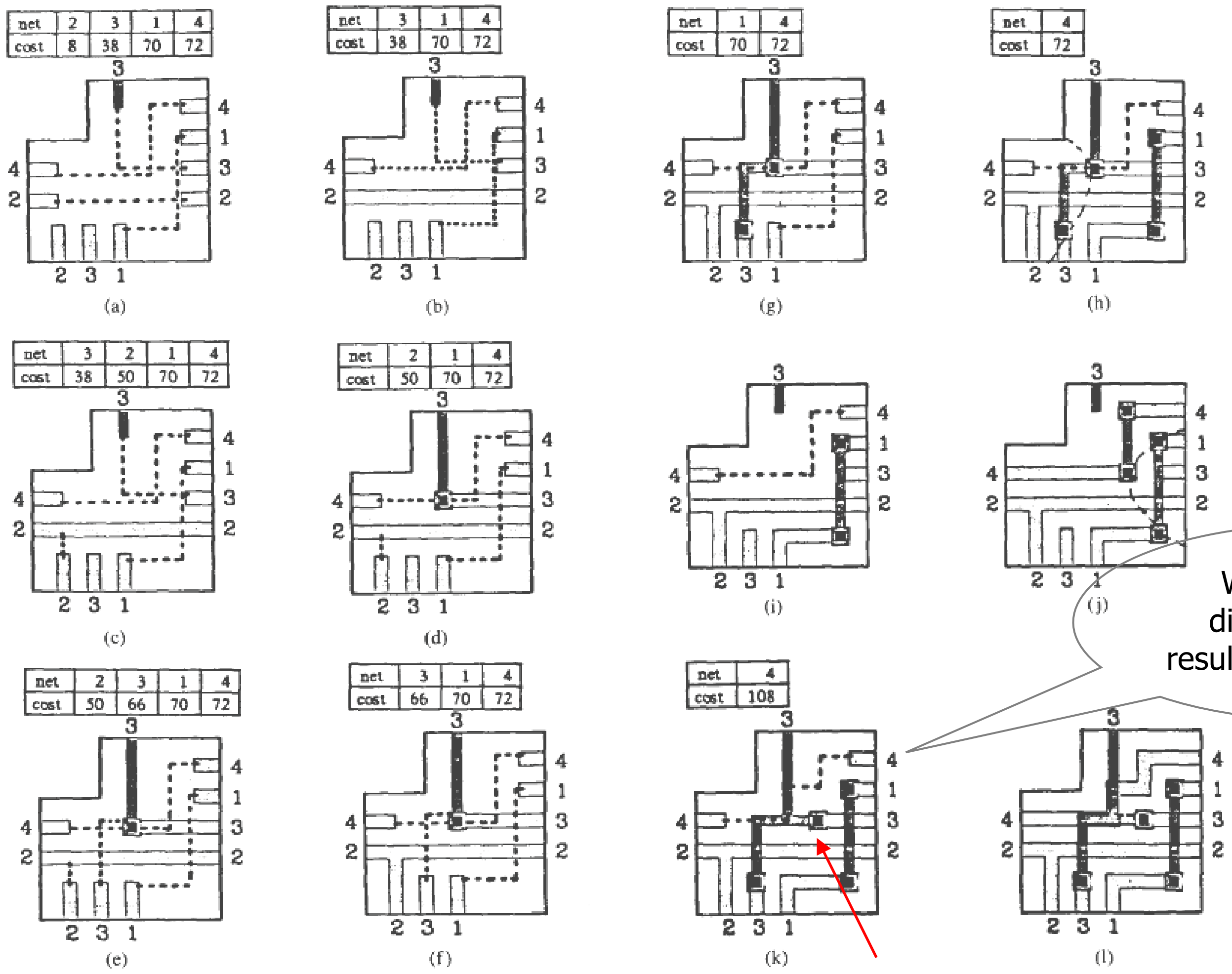
Only consider existing blockages

Grid Based Detailed Routing - Mighty

```
strong_modification( i )
    /* find a pair of blocked pins of net i */
    find two closest pins p1 and p2 of net i
        which are not in the same component;
    find a minimum-cost rip-up path connecting p1 and p2;
    /* Limit_cost is a user defined parameter */
    if( rip-up cost ≥ Limit_cost )
    {  /* rip-up cost is too large */
       report_failure();
       exit();
    } else
    {  /* remove all the connections in rip-up-path.
         reschedule all the affected nets. */
       for( each net k in the rip-up path )
       {
          difficulty[k] = difficulty[k] + Delta;
          remove all the connections of net k;
          /* reschedule with zero cost and nil path */
          reschedule( k );
          clear_history( k );
       }
    while( net i has more than one component )
    {
       find a shortest path[ i ] connecting any two
          components of net i;
       if( path[ i ] != φ )
          conform_path( i );
       else
          strong_modification( i );
    }
    /* reset the ith row and ith column of history
      which is used in weak modification */
    clear_history( i );
    }
```

Grid Based Detailed Routing - Mighty

cost
Via: 30
Prefer-dir:2
Anti-dir: 50


net 2 3 1 4 / cost 8 38 70 72
(a)
net 3 1 4 / cost 38 70 72
(b)
net 1 4 / cost 70 72
(g)
net 4 / cost 72
(h)
net 3 2 1 4 / cost 38 50 70 72
(c)
net 2 1 4 / cost 50 70 72
(d)
(i)
(j)
net 2 3 1 4 / cost 50 66 70 72
(e)
net 3 1 4 / cost 66 70 72
(f)
net 4 / cost 108
(k)
(l)
4
1
3
2
2 3 1

Why net 3 has different routing result – minimum-cost rip-up path

Rerouting net 3 uses black wire here to yield least-cost rip-up path

Another Channel Route Result from Mighty

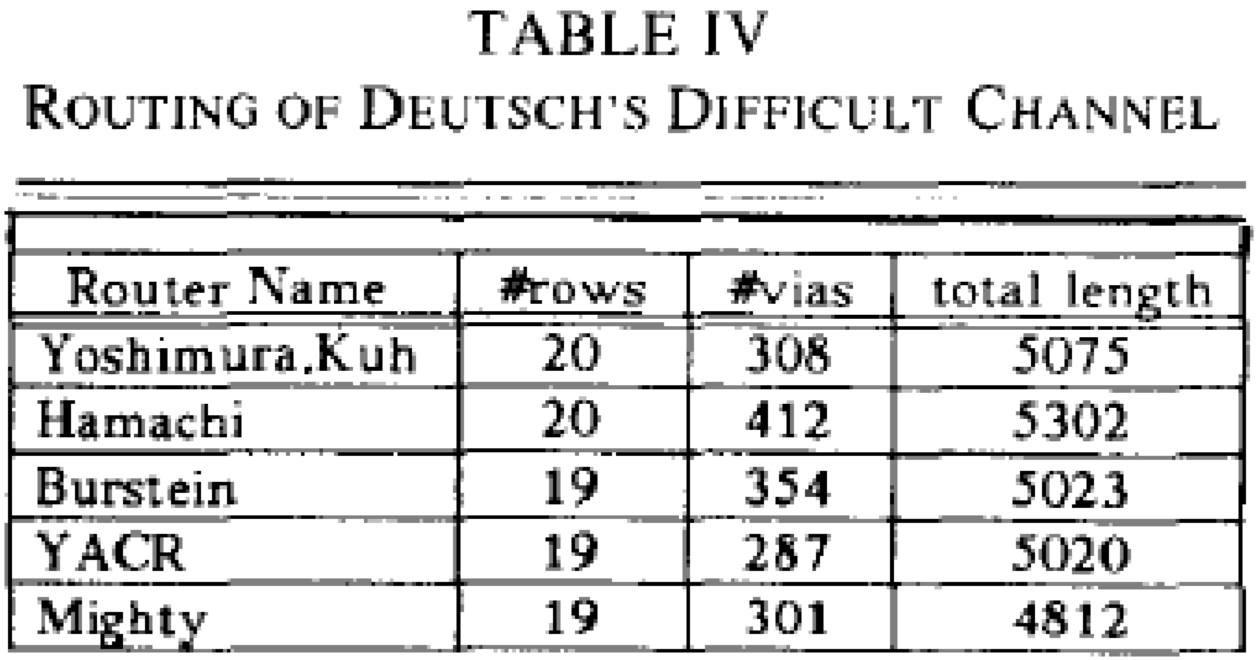


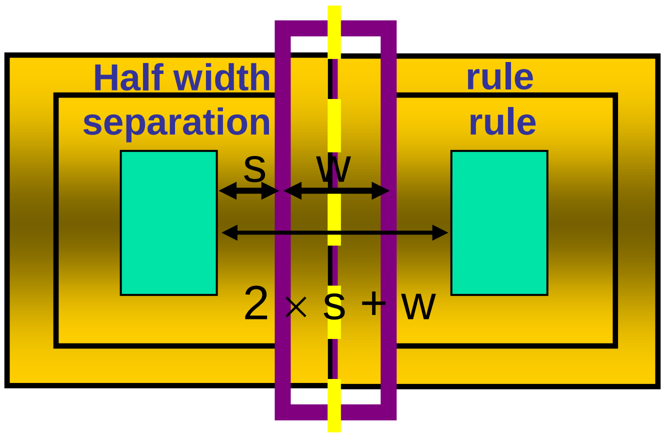
TABLE IV
ROUTING OF DEUTSCH'S DIFFICULT CHANNEL

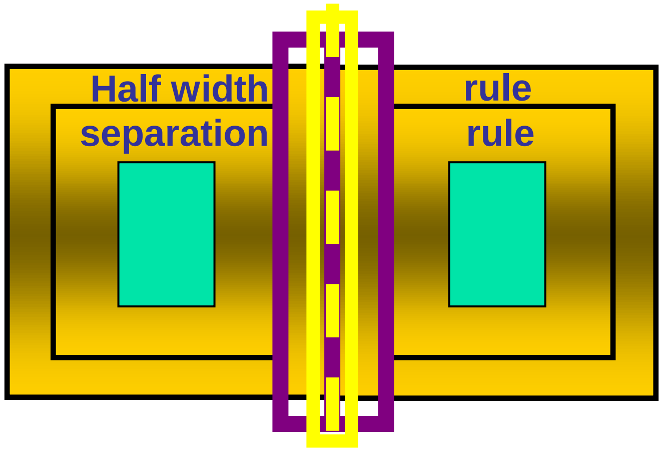
Router Name	#rows	#vias	total length
Yoshimura,Kuh	20	308	5075
Hamachi	20	412	5302
Burstein	19	354	5023
YACR	19	287	5020
Mighty	19	301	4812

Why Gridless Routing ?

- Variable-width and variable-space routing becomes inevitable for modern designs
  - Wide space and fat wire for crosstalk and delay optimization
- Gridless routers are more flexible for variable-rule routing than grid-based routers

Basic Concept





Zero-width Model

Implicit connection graph-based router

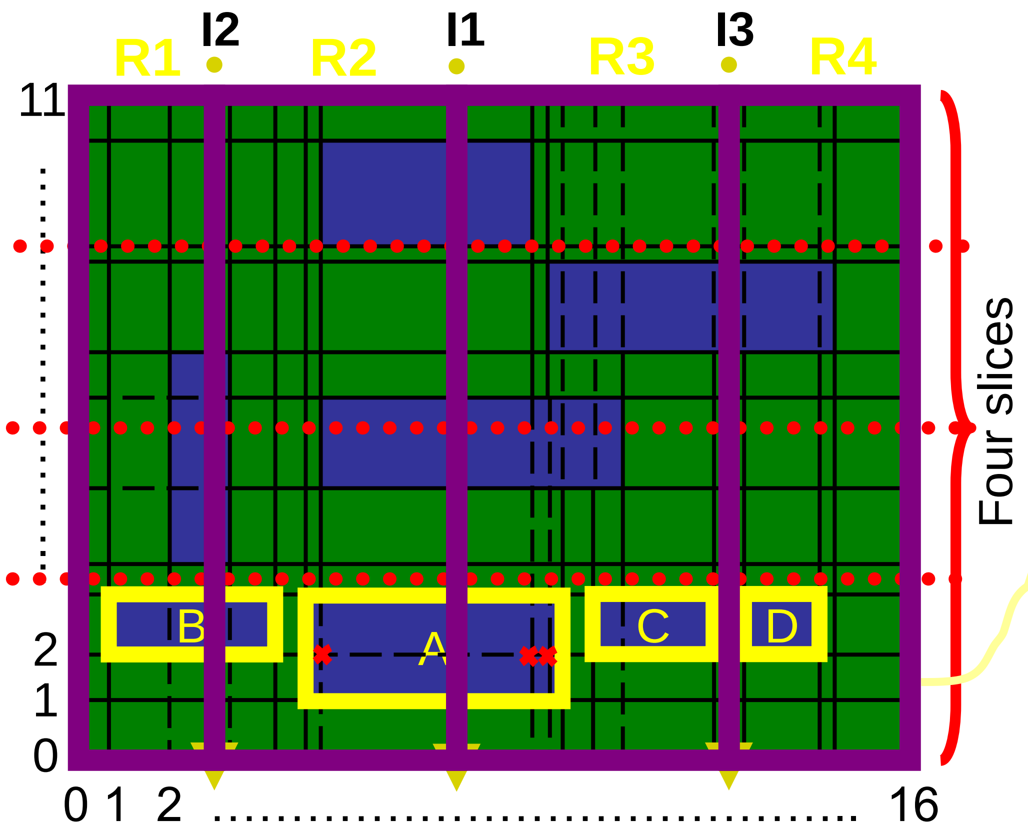
Nonzero-width Model

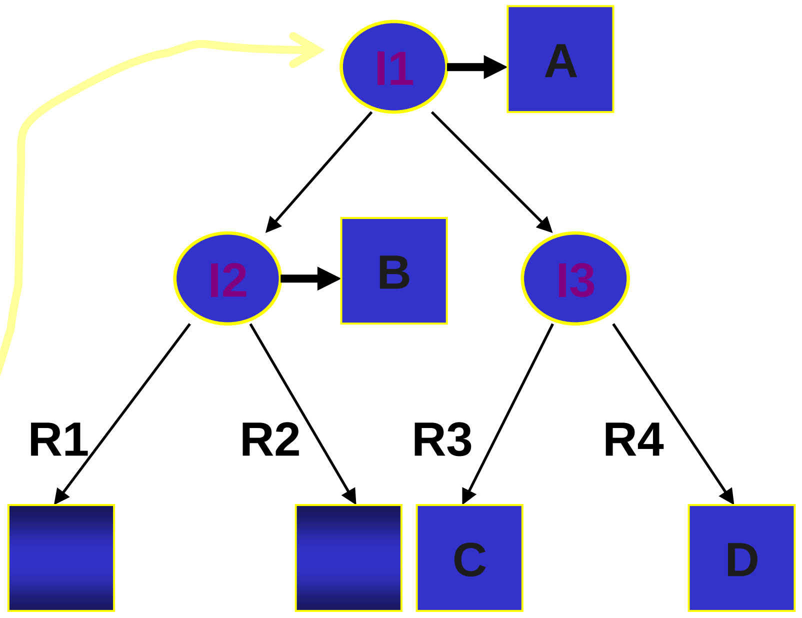
Tile-based router

Implicit Connection Graph-Based Router

- Routing flow
  - Construct a slit and interval tree using the raw wires
  - Construct implicit connection graph using (a) 2-D point array(s) by extending the borders of expanded wires
    - Single routing plane
    - Multiple routing planes with each plane integrating all wires of two adjacent layers
  - Grid maze over the implicit connection graph

Implicit Connection Graph-Based Router



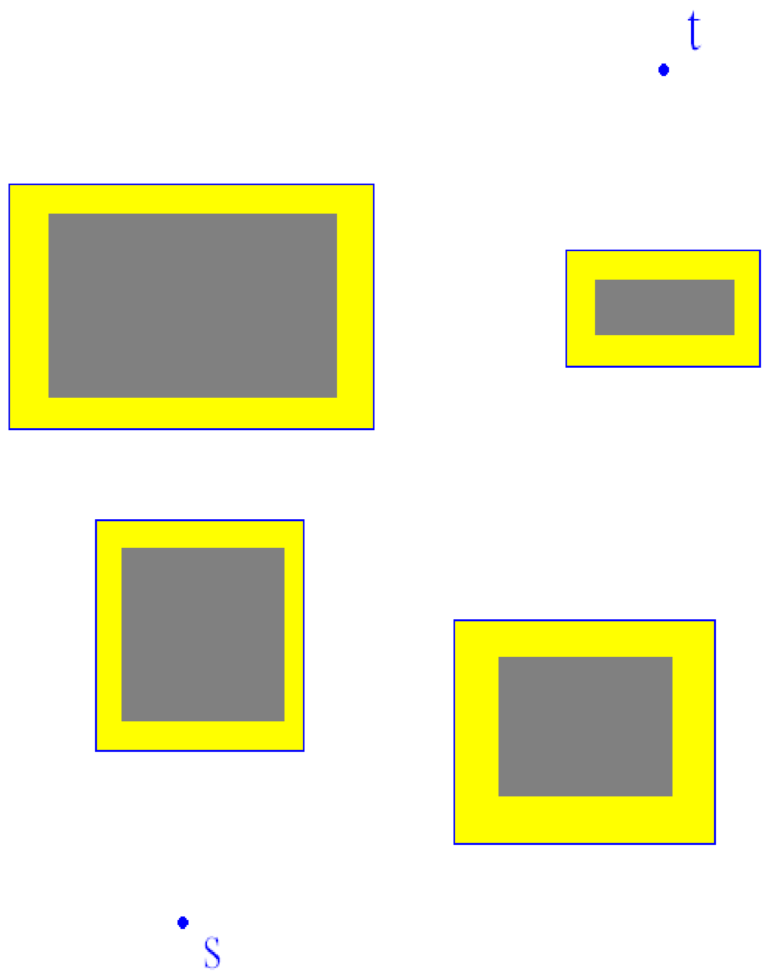


Fast routing graph construction

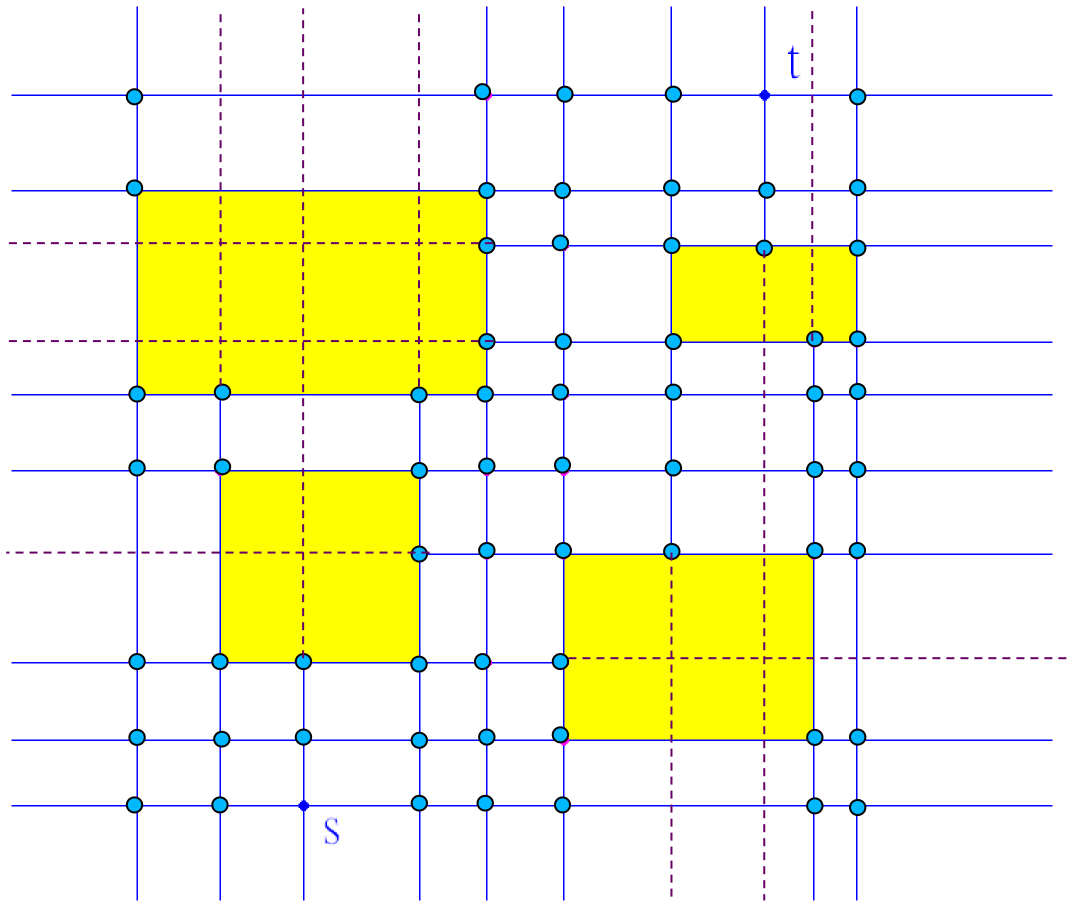
- **J. Cong et al., "An Implicit connection graph maze Routing Algorithm for ECO routing," in ISPD99**

Efficient query data structure (Slit tree + Interval Tree)

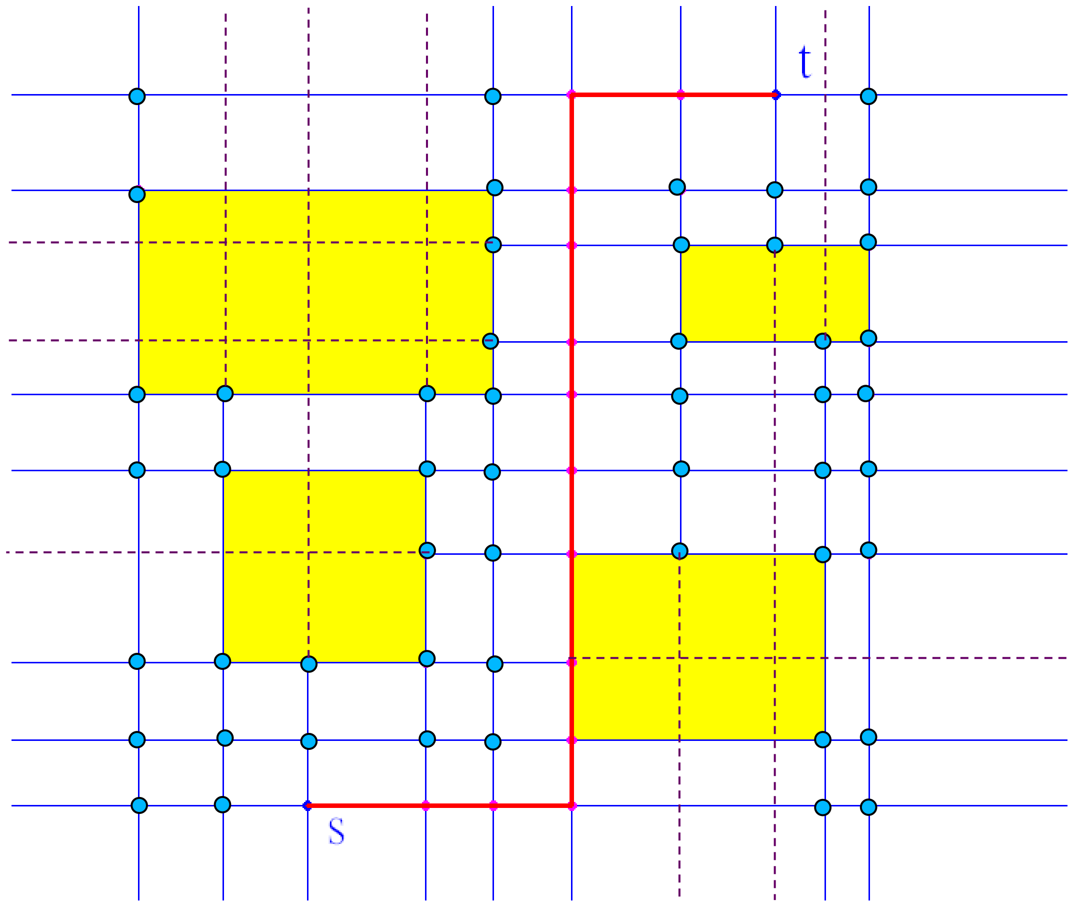
Implicit Connection Graph Example



Implicit Connection Graph Example – cont.



Implicit Connection Graph Example – cont.



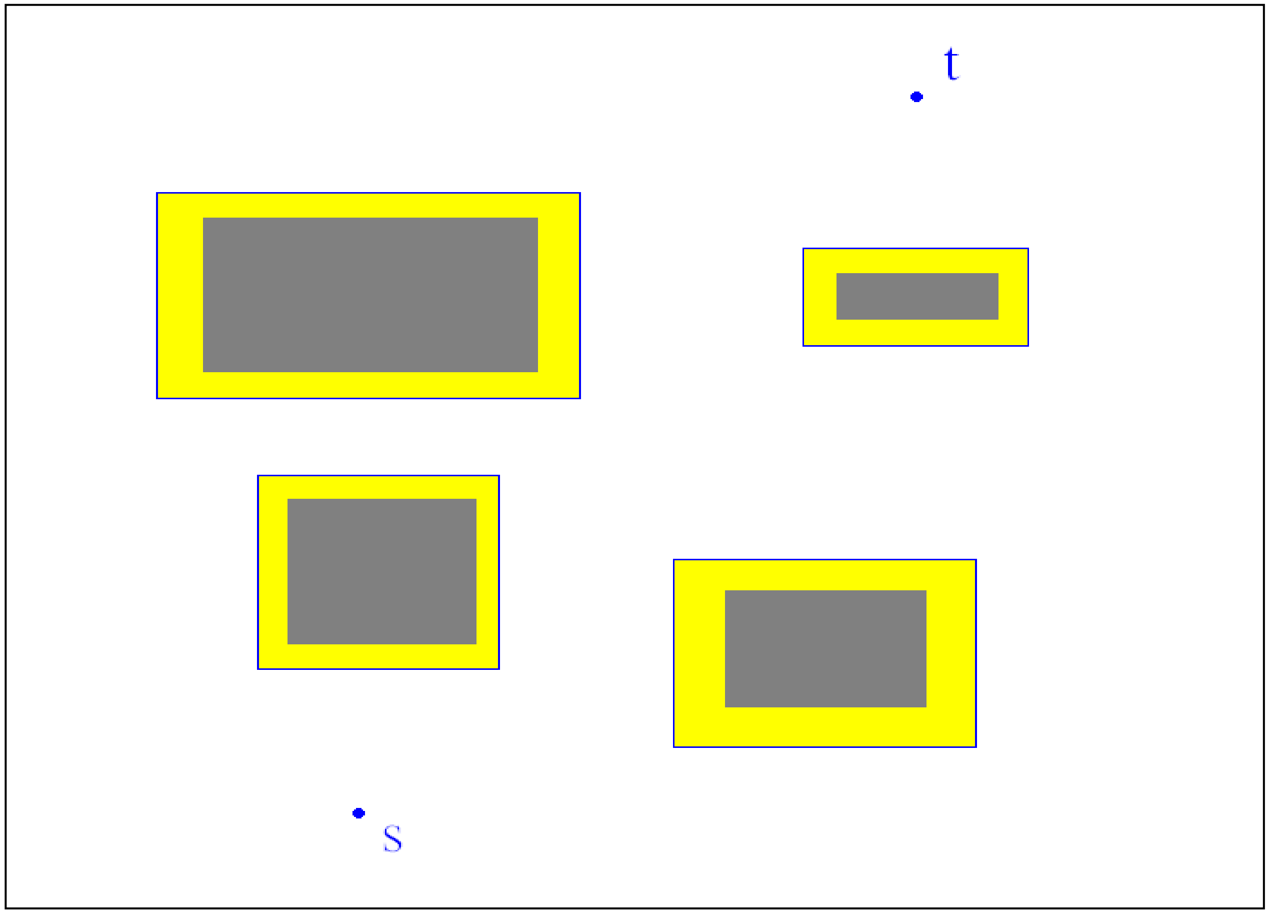
Tile-Based Router

- Routing flow
  - Add contours to existing wires
  - Construct corner-stitching tile plane by extending the borders of the expanded wires
  - Extract starting segments
  - Propagate routing paths over the tile plane
  - Construct final path from a series of connected tiles (abut or overlap with neighboring tiles)

TCAD Feb 2007: An Efficient Tile-Based ECO Router Using Routing Graph Reduction and Enhanced Global Routing Flow

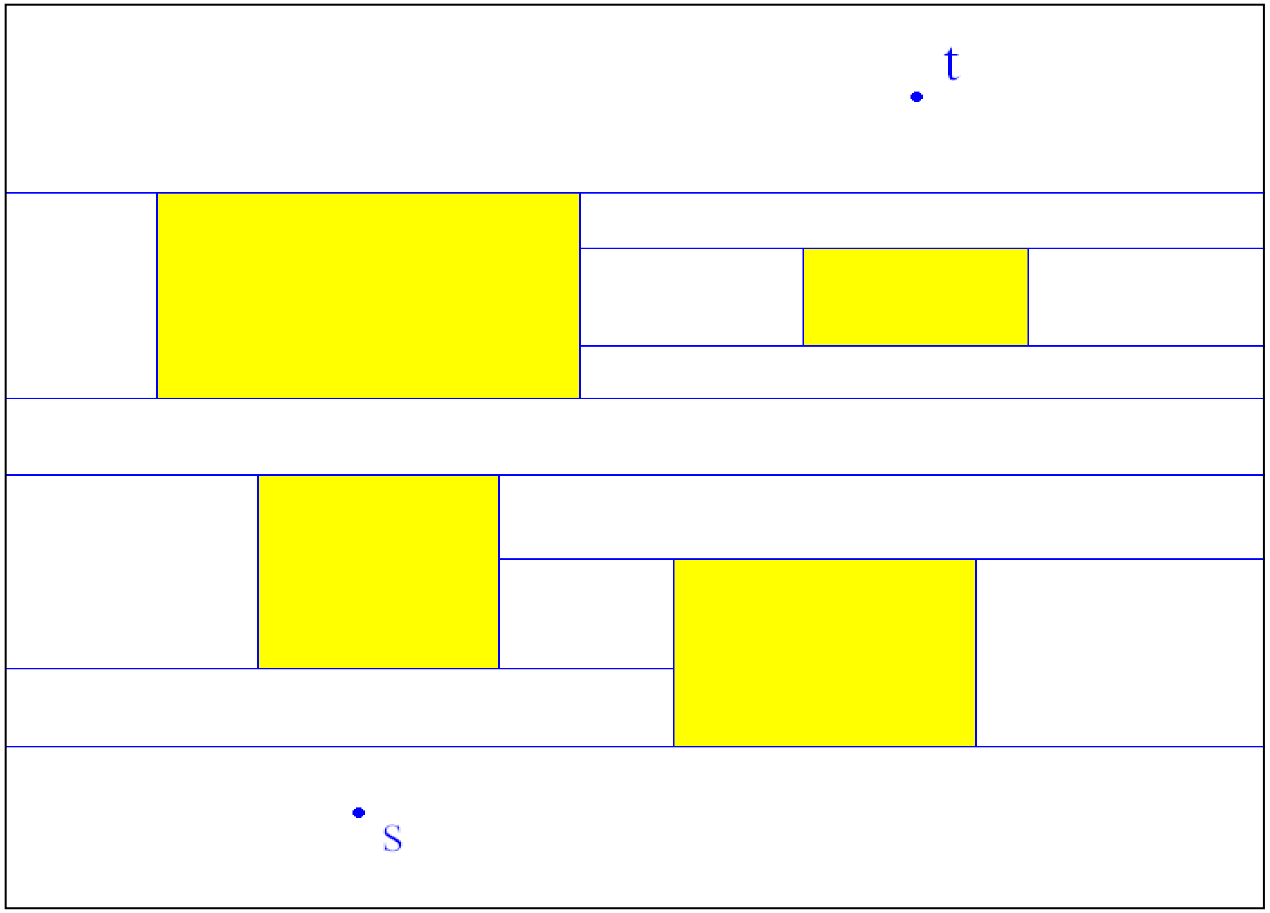
Tile-Based Router Example

- Contour Insertion



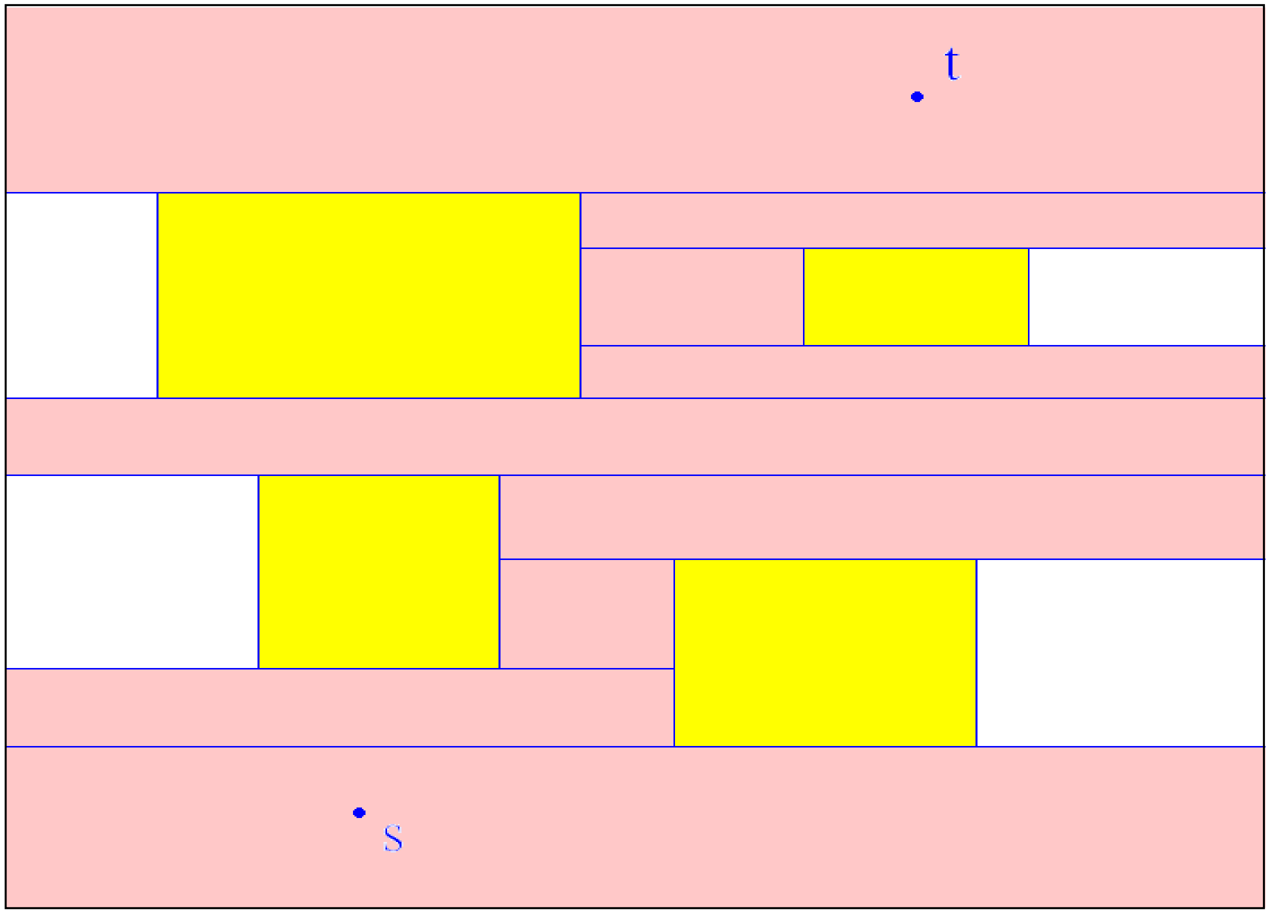
Tile-Based Router Example

- Corner Stitching Tile Plane Creation



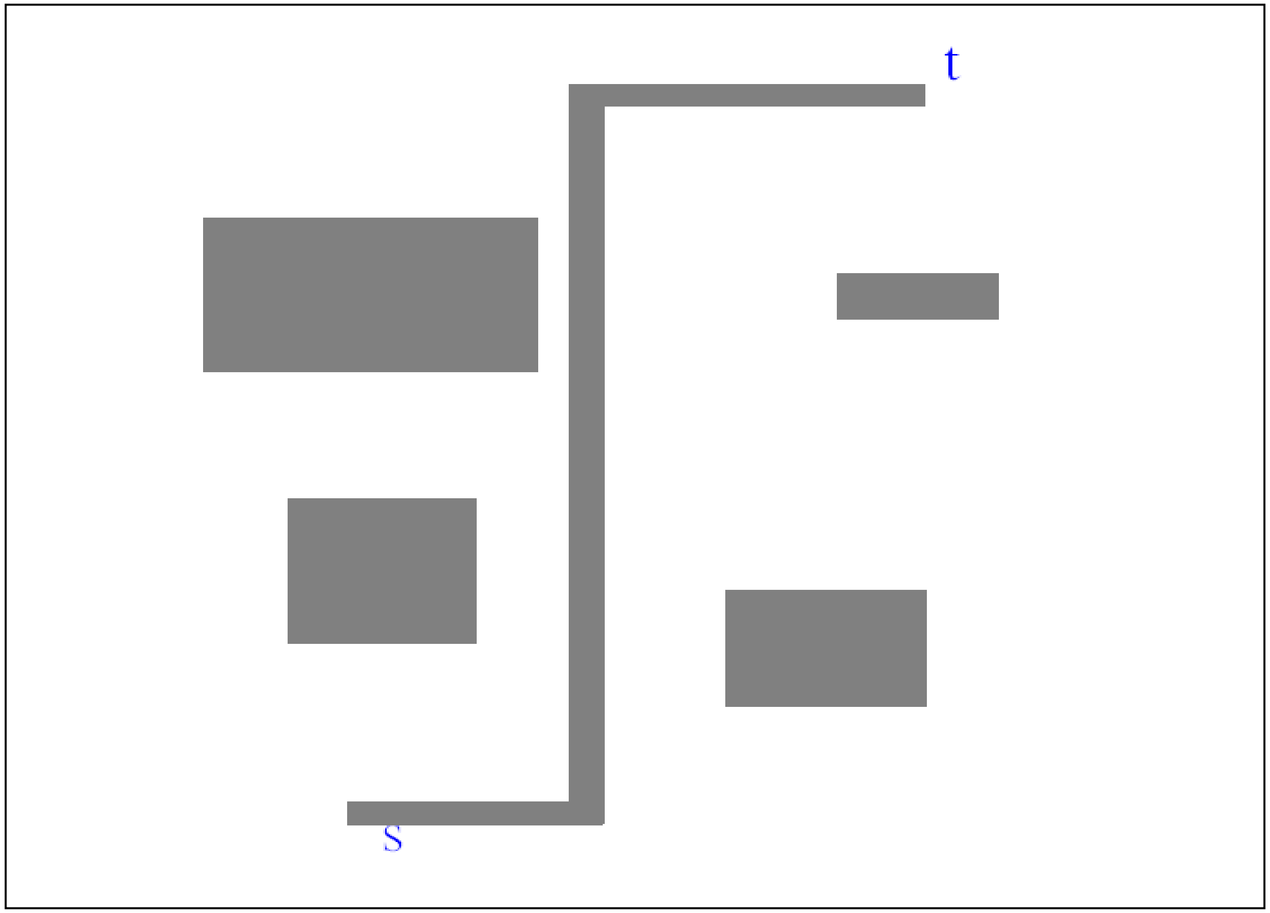
Tile-Based Router Example

- Tile Propagation



Tile-Based Router Example

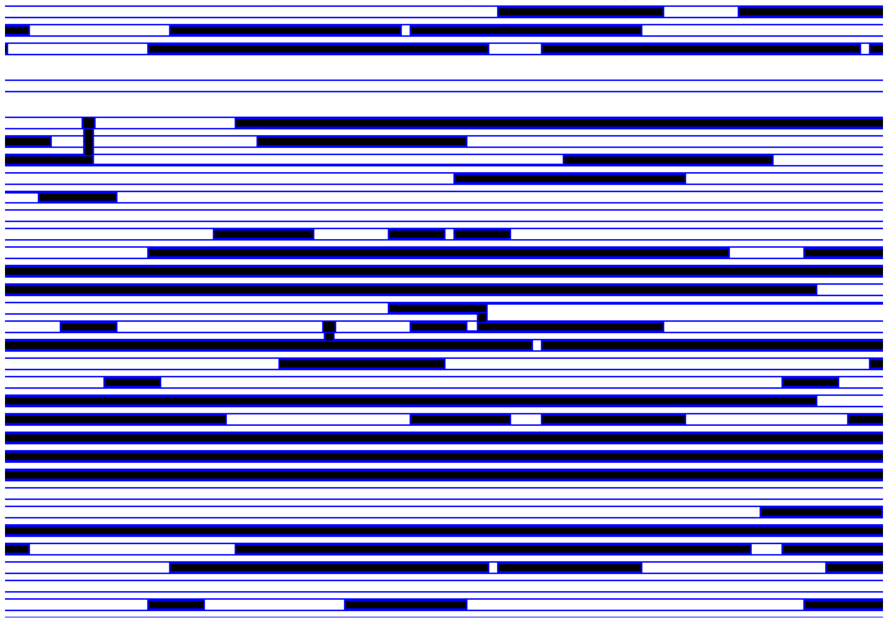
- Path construction



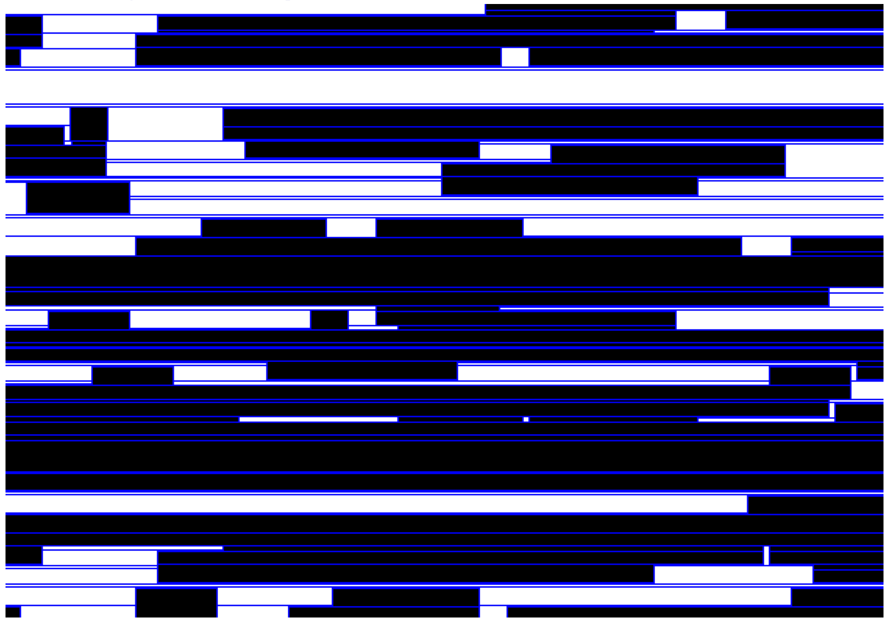
Fragmented Tile Planes

- Too many slim tiles
- Reducing the no. of tiles can accelerate path searching
- Routing graph reduction was proposed to simplify tile plane and then to improve tile propagation

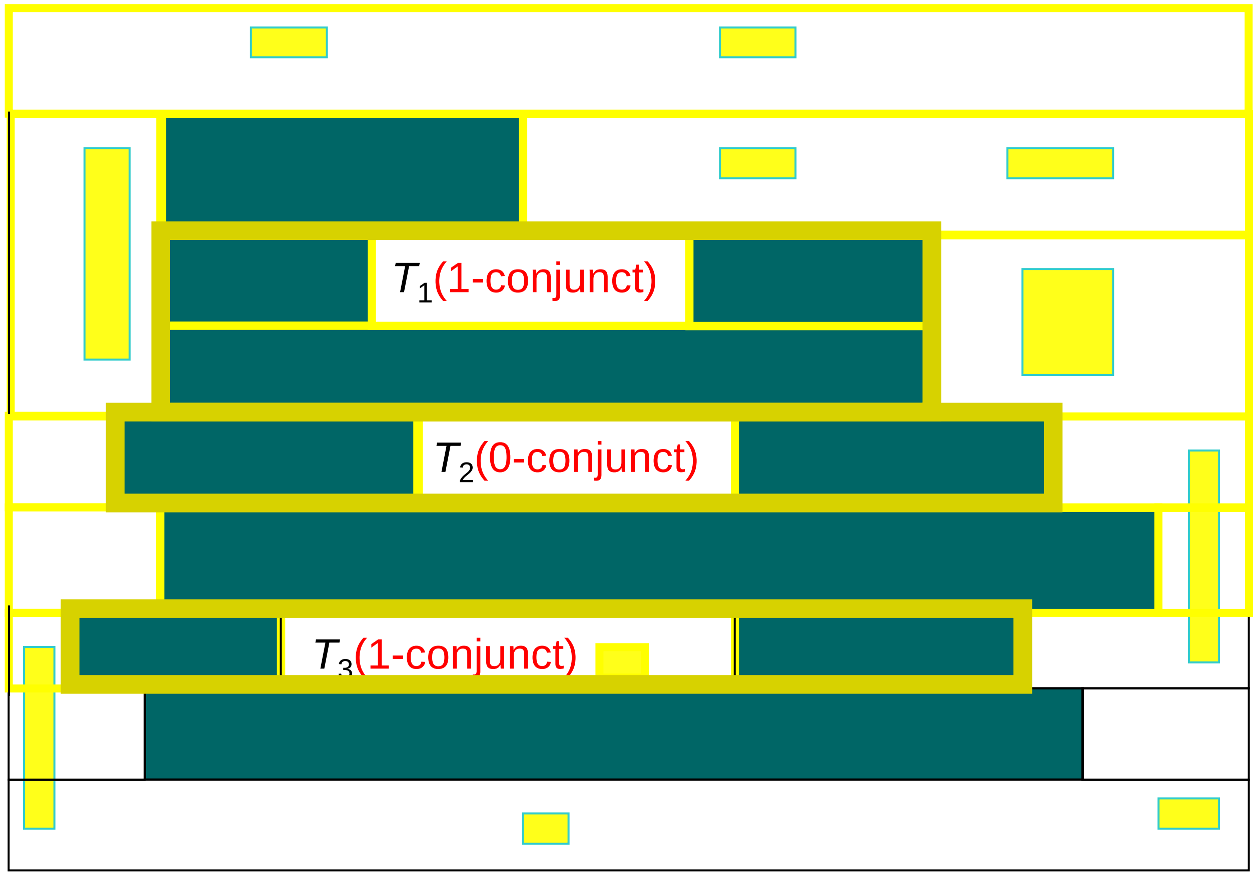
An original H-layer tile plane



A H-layer tile plane after contour insertion



Redundant Tiles Removal

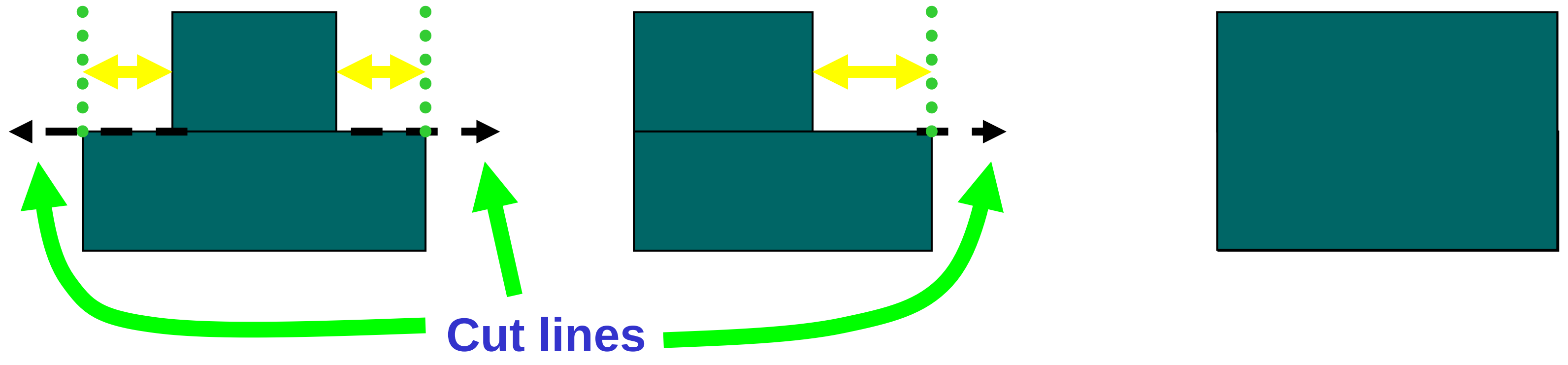


🟩 Block Tile ⬜ Space Tile 🟨 Via region (the region that can accommodate new via)

Block Tile : **10→6** Space Tile: **16→13**

Neighboring Tiles Alignment

- We can adjust and align the left and right sides of two adjacent block tiles to merge them as a block tile.
  - Adjusting border is to enlarge block tiles and to shrink space tiles.

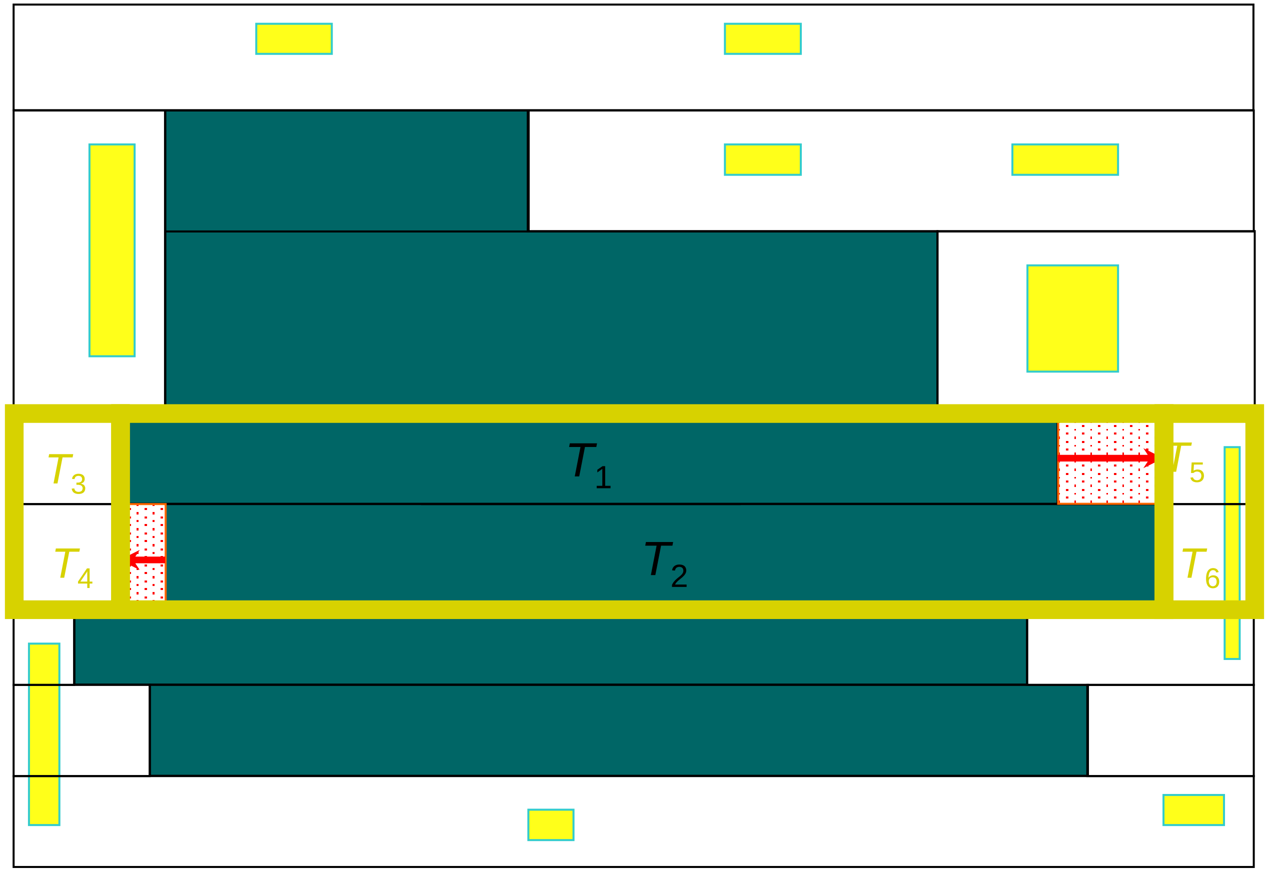


Neighboring Tiles Alignment

(T_1,T_2)

(T_3,T_4)

(T_5,T_6)



Comparison of Implicit Connection Graph-Based Router and Tile-Based Router

Time Complexity	Point locating	Neighbor finding	Legality check	Single object insertion	Query for layer switching
Implicit connection graph-based router	***Log*(n_i)** n_i: the # of intervals (leaf nodes) in the interval tree	Constant time	**n_o*Log*(n_i)** n_o: the # of objects attached to a node	***n*** *n*: the # of objects	Constant time
Tile-based router	$\sqrt{n_t}$ n_t: the # of tile on a plane	Constant time	Constant time	$\sqrt{n_t} + m^2$ overlapping the inserted tile	$\sqrt{n_t}$

Space Complexity	**Implicit connection graph**	**Tile plane**
# of instance	**536,214,892**	**3,173,533**

NEMO: A New Implicit-Connection-Graph-Based Gridless Router With Multilayer Planes and Pseudo Tile Propagation

TCAD 2007

Decompose multi-terminal net routing into multiple 2-pin net routing

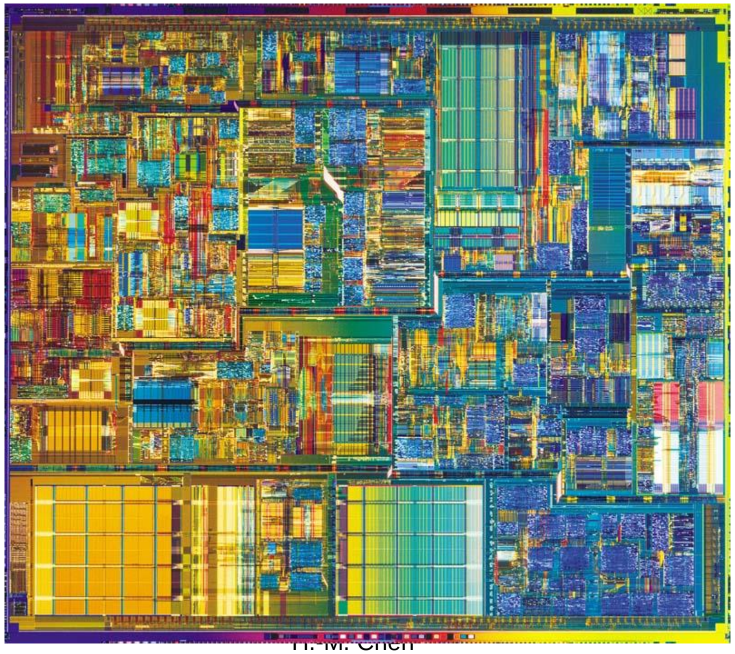
Perform congestion-driven global routing

Rip up and Rerouting

Complete point-to-point detailed routing

Routing Trends

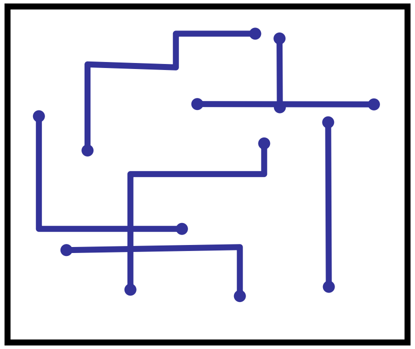
- Billions of transistors may be fabricated in a single chip for nanometer technology.
- Need tools for very large-scale designs.
- Framework evolution for CAD tools
  - Flat ➔ Hierarchical ➔ Multilevel

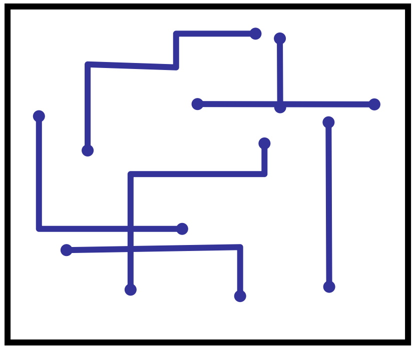


Pentium 4
42 M
Transistors
(Y2000)

Flat Routing Framework

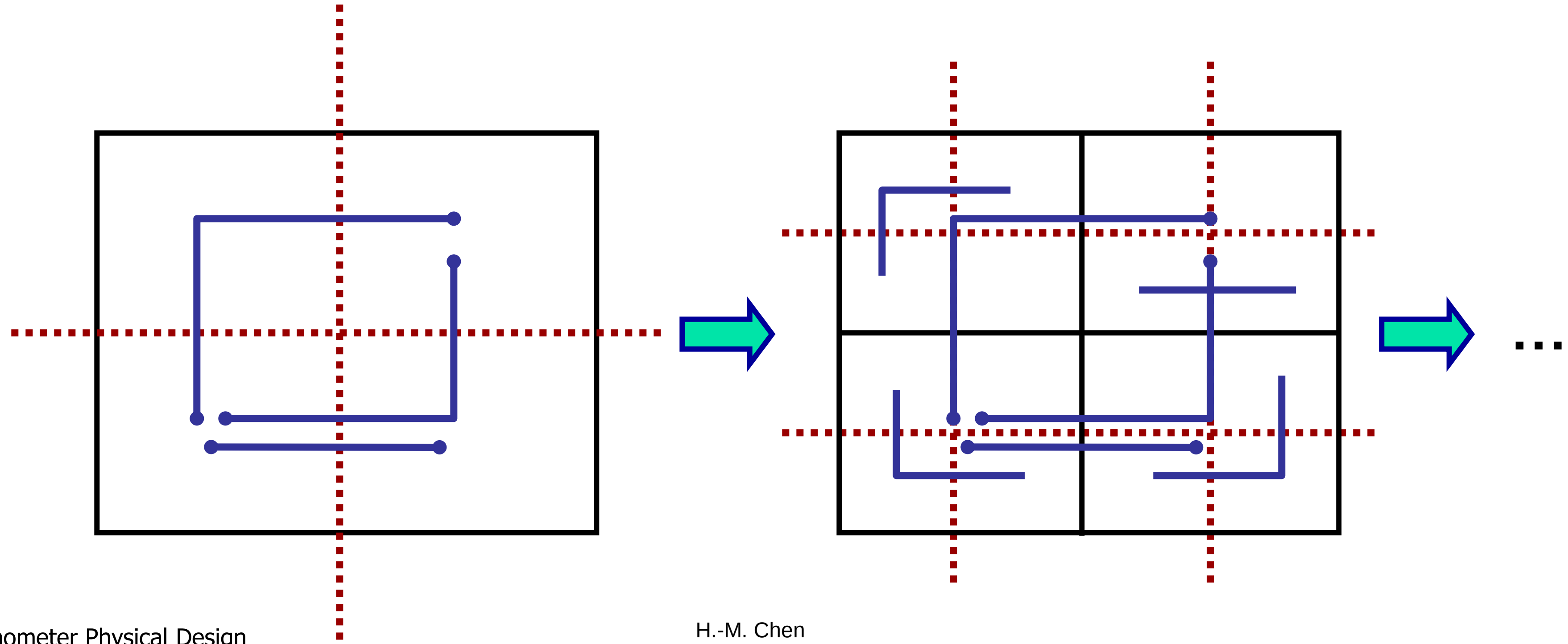
- Sequential approaches
  - Maze searching
  - Line searching
- Concurrent approaches
  - Network-flow based algorithms
  - Linear assignment formulation
- Drawback: hard to handle larger problems


Sequential


Concurrent

Hierarchical Routing Framework

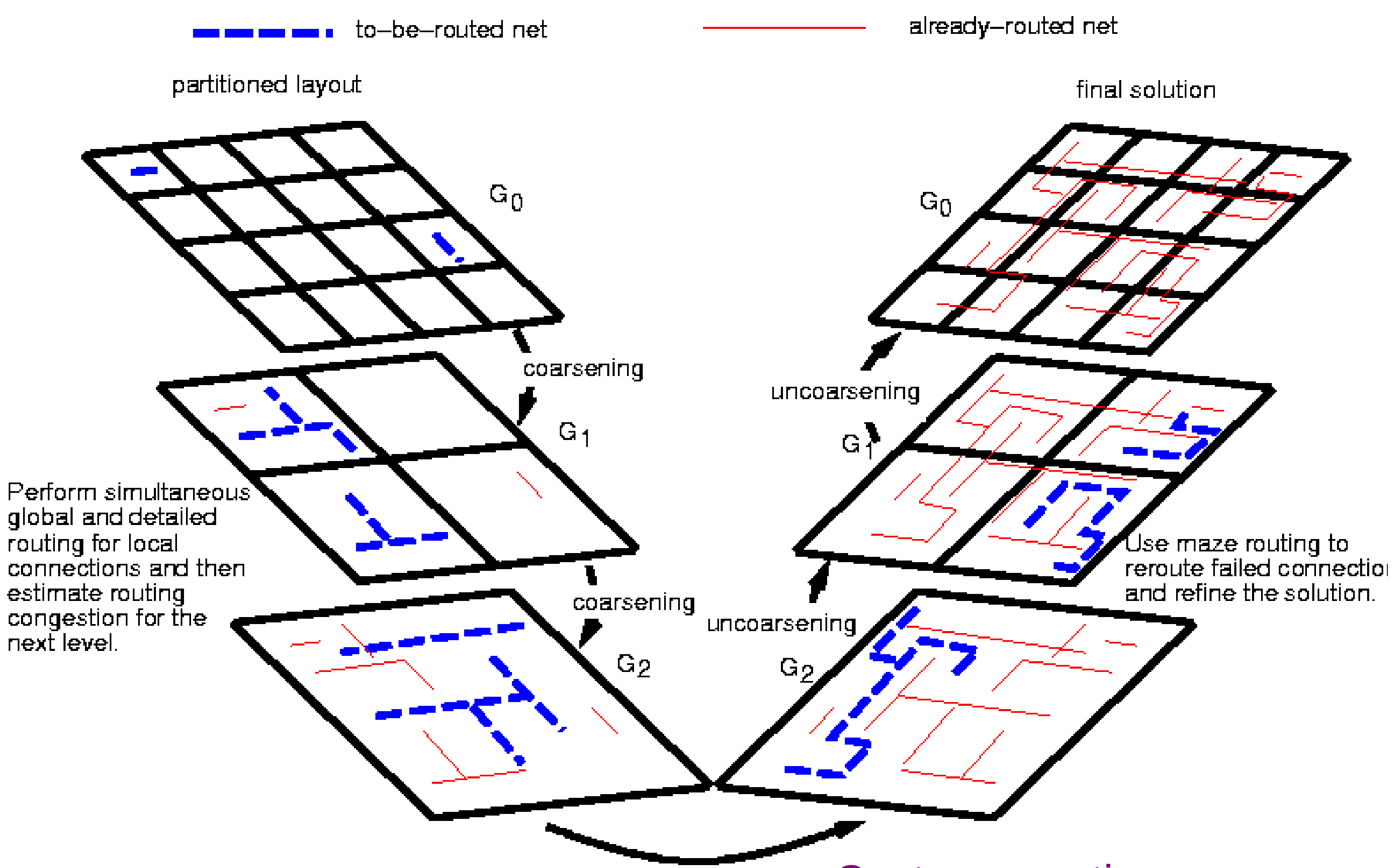
- The hierarchical approach recursively divides a routing region into a set of subregions and solve those subproblems independently.
- Drawbacks: lack the global information for the interaction among subregions.



Multilevel Routing Framework

- Lin and Chang, “A novel framework for multilevel routing considering routability and performance,” ICCAD-2002 (TCAD, 2003).
- Multilevel framework: coarsening followed by uncoarsening.
- Coarsening (bottom-up) stage:
  - Constructs the net topology based on the minimum spanning tree.
  - Processes routing tiles one by one at each level, and only local nets (connections) are routed.
  - Applies two-stage routing of global routing followed by detailed routing.
  - Uses the L-shaped & Z-shaped pattern routing.
  - Performs resource estimation after detailed routing to guide the routing at the next level.
- Uncoarsening (top-down) stage
  - Completes the failed nets (connections) from the coarsening stage.
  - Uses a global and a detailed maze routers to refine the solution.

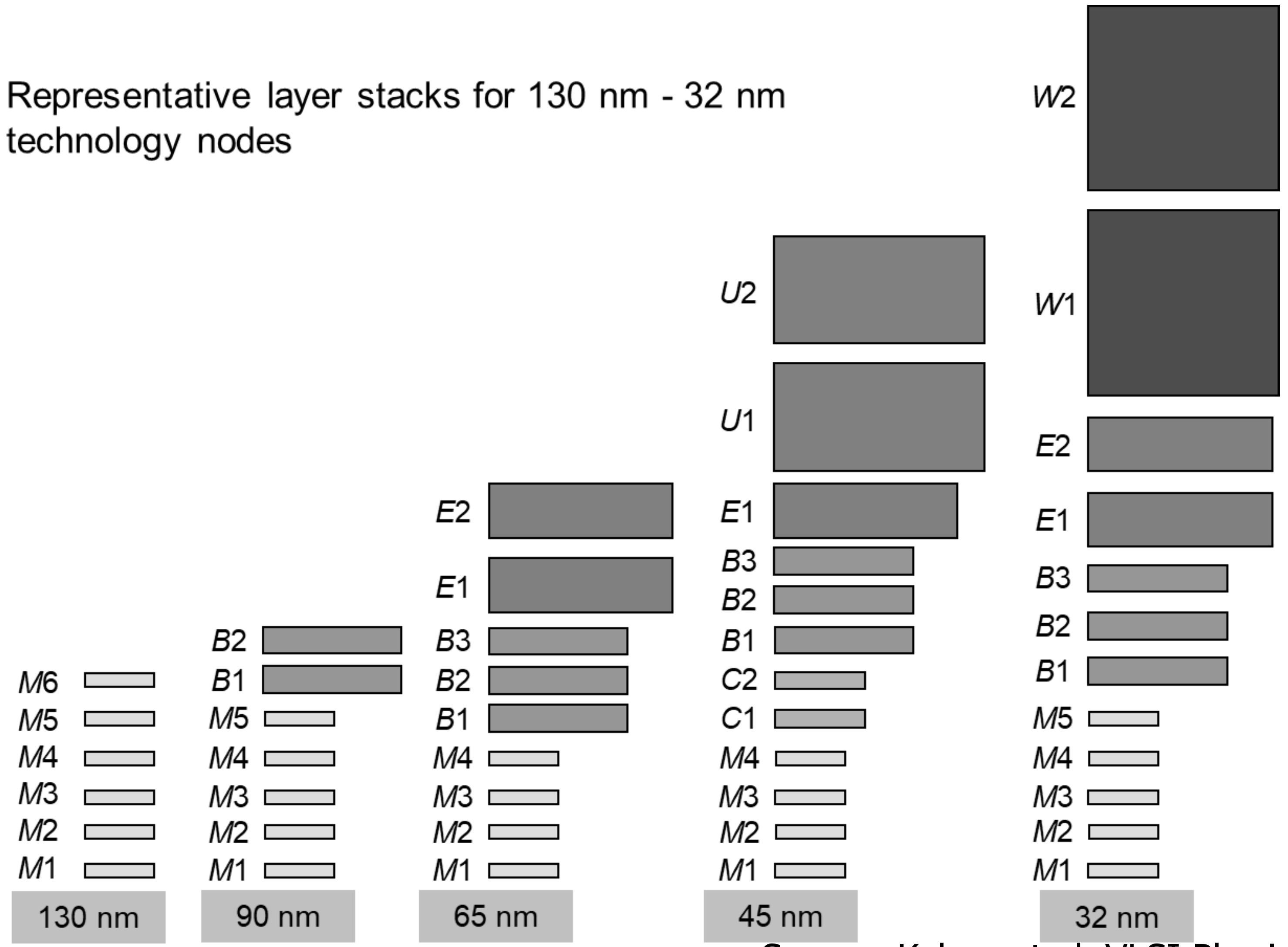
A Multilevel Routing Framework


to–be–routed net
already–routed net
partitioned layout
final solution
G0
G0
coarsening
uncoarsening
G1
G1
coarsening
uncoarsening
G2
G2
Perform simultaneous global and detailed routing for local connections and then estimate routing congestion for the next level.
Use maze routing to reroute failed connectio
and refine the solution.

Cost: congestion

Cost: congestion
+ net length

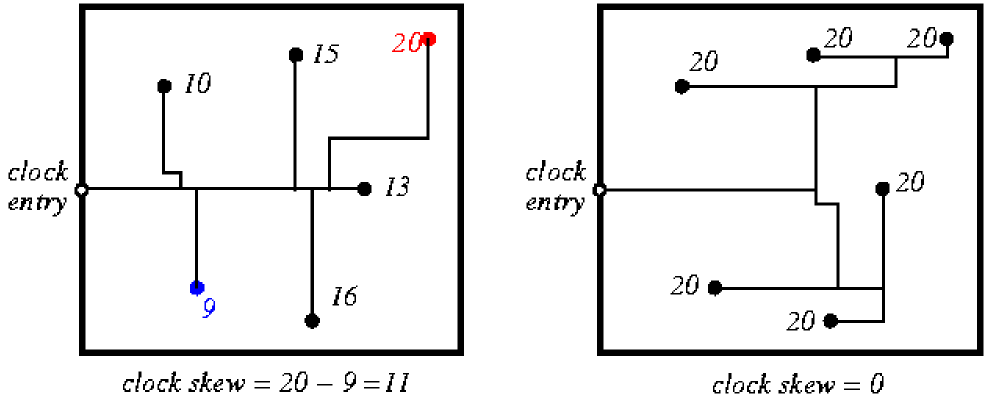
Technology Evolution Impacting Routing



Source: Kahng et al, VLSI Physical Design, Springer 2011

Clock Routing

- Digital systems
  - **Synchronous systems:** Highly precise 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.



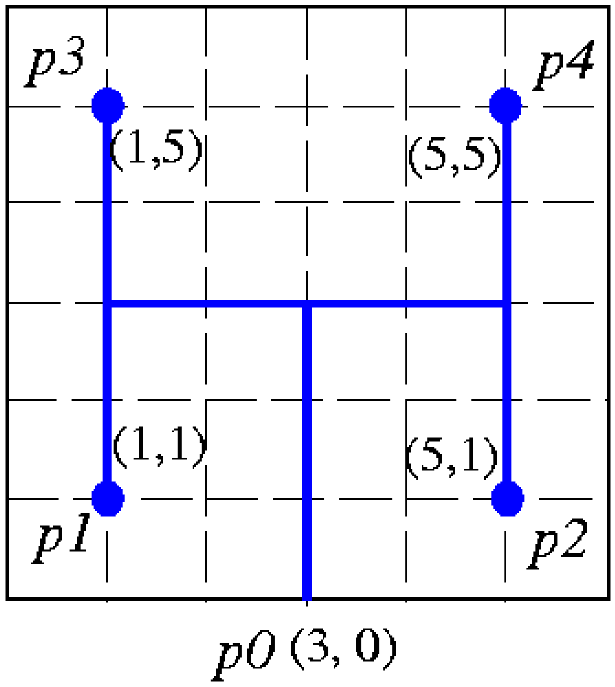
- **Clock Routing Problem:** Routing clock nets such that
  1. clock signals arrive simultaneously
  2. clock delay is minimized
     - Other issues: total wirelength, power consumption

Clock Routing Problem (CRP)

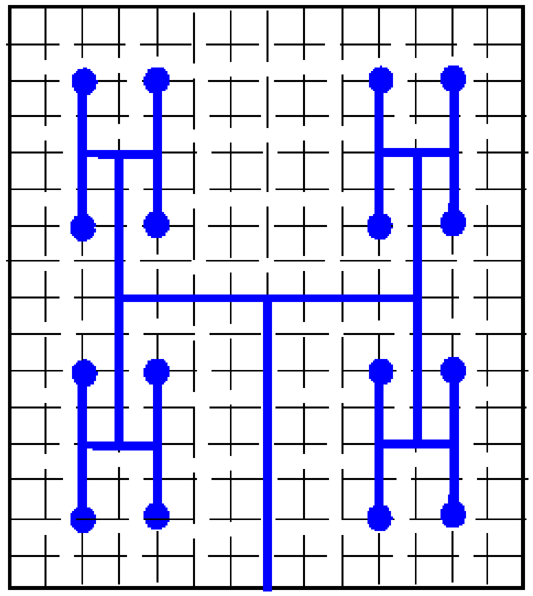
- Given the routing plane and a set of points $P = \{p_1, p_2, \ldots, p_n\}$ within the plane and clock entry point p_0 on the boundary of the plane, the **Clock Routing Problem (CRP)** is to interconnect each $p_i \in P$ such that $\max_{i, j \in P}|t(0, i) - t(0, j)|$ and $\max_{i \in P} t(0, i)$ are both minimized.
- Pathlength-based approaches
  1. *H*-tree: Dhar, Franklin, Wang, ICCAD-84; Fisher & Kung, 1982.
  2. Methods of means & medians (MMM): Jackson, Srinivasan, Kuh, DAC-90.
  3. Geometric matching: Cong, Kahng, Robins, DAC-91.
- RC-delay based approaches:
  1. Exact zero skew: Tasy, ICCAD-91.
  2. Lagrangian relaxation: Chen, Chang, Wong, DAC-96.

H-Tree Based Algorithm

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



H–tree over 4 points



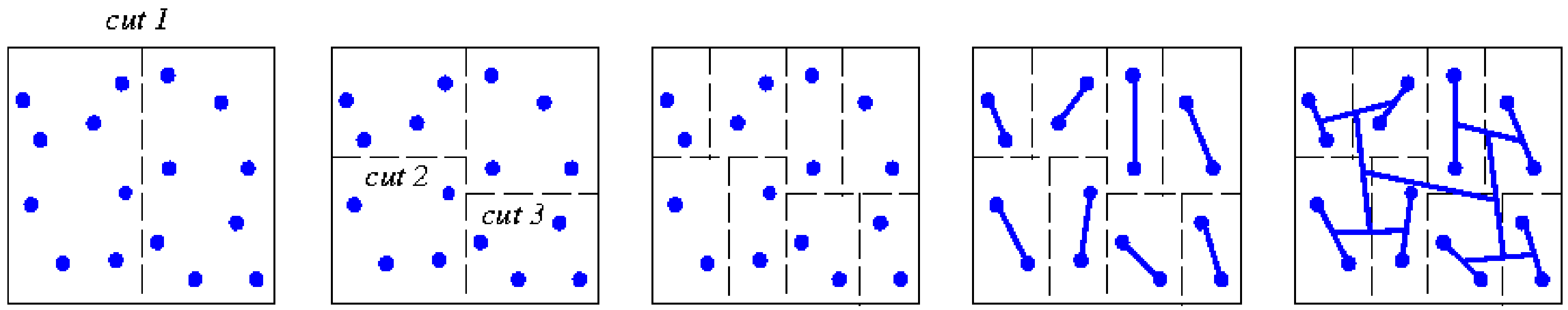
H–tree over 16 points

The MMM Algorithm

- Jackson, Sirinivasan, Kuh, “Clock routing for high-performance ICs,” DAC-1990.

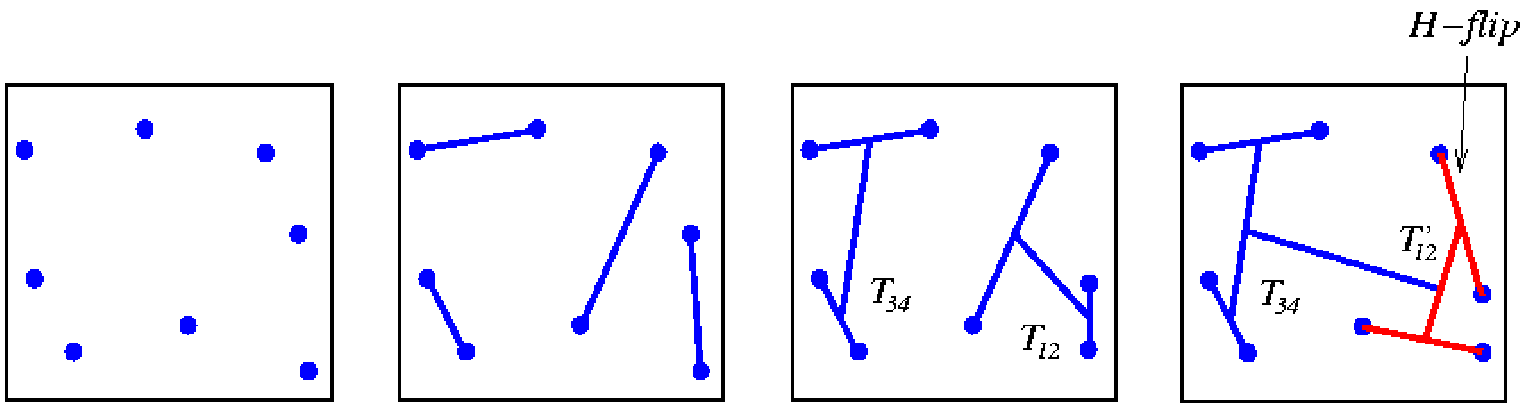
1. Each block pin is represented as a point in the region, *S*.
2. The region is partitioned into two subregions, S_L and S_R.
3. The center of mass is computed for each subregion.
4. The center of mass of the region *S* is connected to each of the centers of mass of subregion S_L and S_R.
5. The subregions S_L and S_R are then recursively split in *Y*-direction.

- Steps 2--5 are repeated with alternate splitting in *X*- and *Y*-direction.
- Time complexity: $O(n \log n)$.



The Geometric Matching Algorithm

- Cong, Kahng, Robins, “Matching based models for high-performance clock routing,” IEEE TCAD, 1993.
- Clock pins are represented as n nodes in the clock tree ($n = 2^k$).
- 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. (bottom up approach)
- Apply H-flipping to further reduce clock skew (and to handle edges intersection).
- Time complexity: O(n^2 log n).

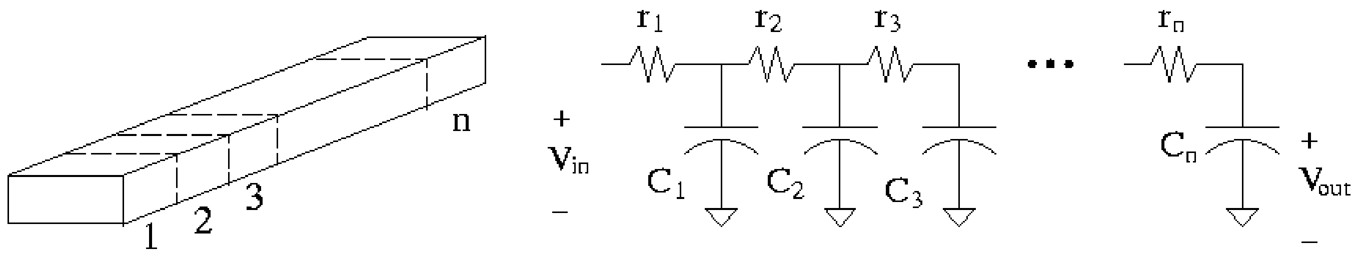


Most Slides Courtesy of Prof. T. W.

Elmore Delay: Nonlinear Delay Model

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

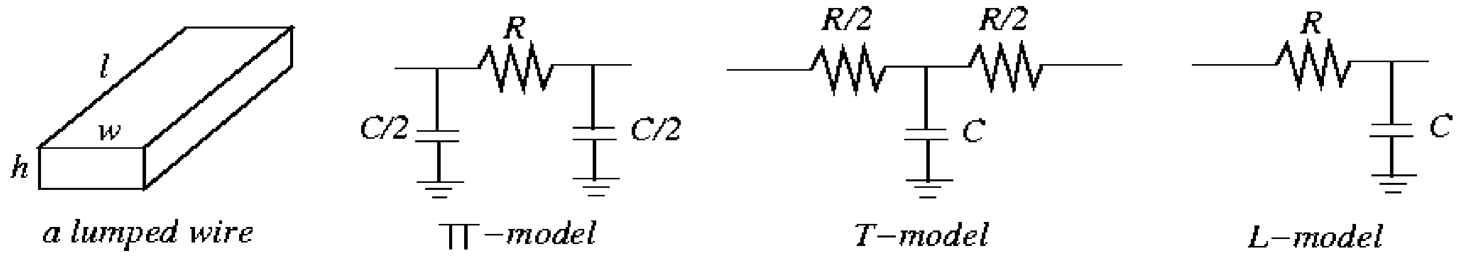
$$\delta = \sum_{i=1}^{n}\left(r_i \sum_{k=i}^{n} c_k\right) = \sum_{i=1}^{n} r(n-i+1)c = \frac{n(n+1)}{2}rc.$$



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

Wire Models

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

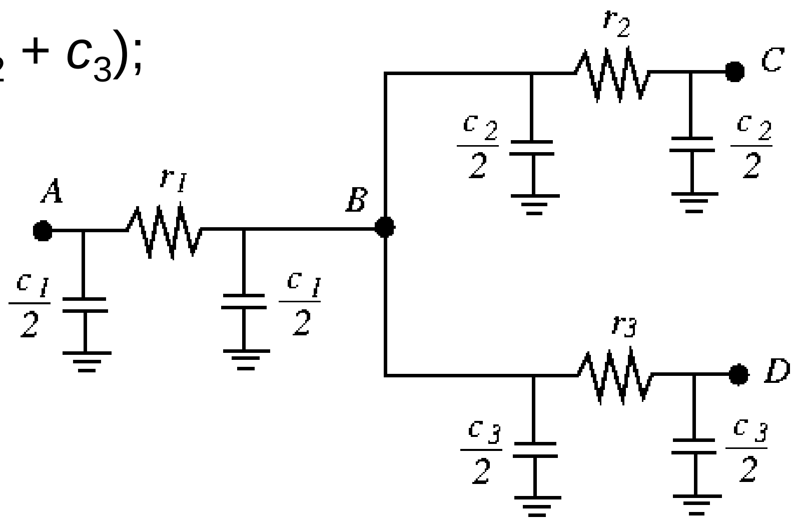


- π-model: If no capacitive loads for *C* and *D*,

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

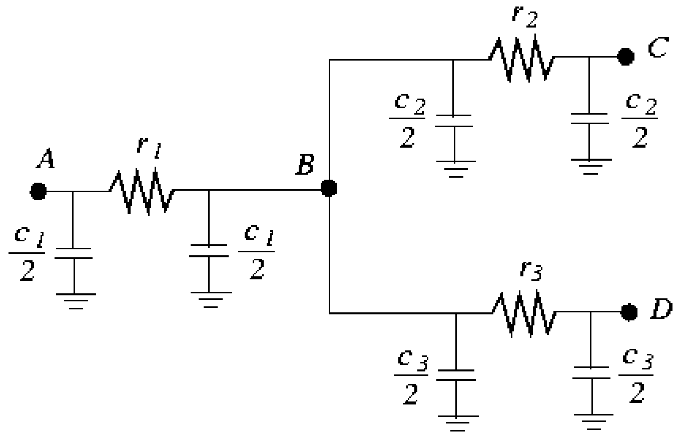
B to *C*: $\delta_{BC} = r_2 (c_2/2)$;

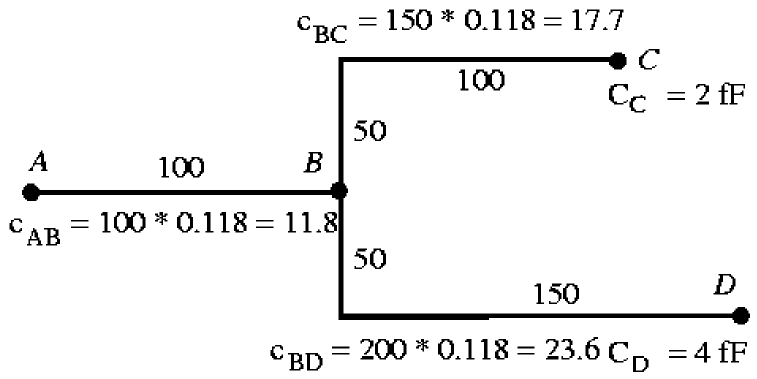
B to *D*: $\delta_{BD} = r_3 (c_3/2)$.



Example Elmore Delay Computation

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

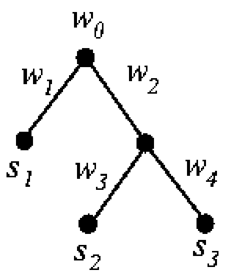




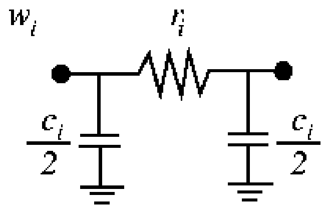
$c_{BC} = 150 * 0.118 = 17.7$
C
100
$C_C = 2$ fF
50
A
100
B
$c_{AB} = 100 * 0.118 = 11.8$
50
150
D
$c_{BD} = 200 * 0.118 = 23.6$ $C_D = 4$ fF

Delay Calculation for a Clock Tree

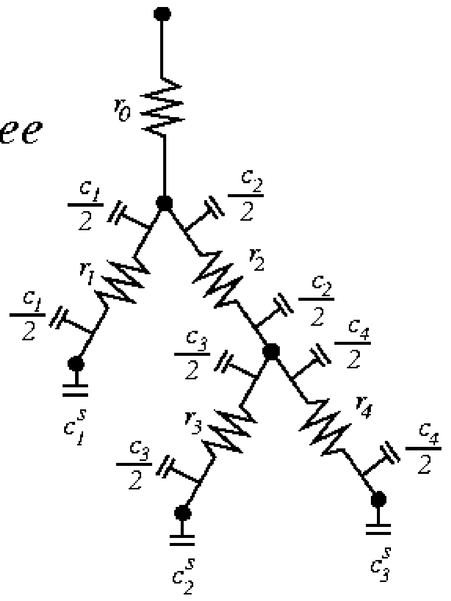
- Let T be an RC tree with points $P = \{p_1, p_2, \ldots, p_n\}$, c_i the capacitance of p_i, r_i the resistance of the edge between p_i and its immediate predecessor.
- The subtree capacitance at node i is given as $C_i = c_i + \sum_{j \in S_i} C_j$, where S_i is the set of all the immediate successors 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

RC tree


Exact Zero Skew Algorithm

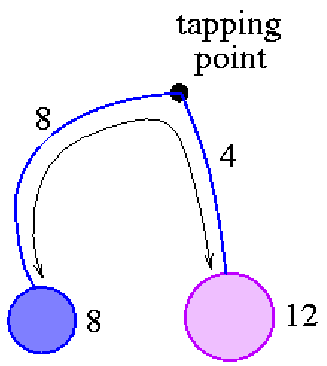
- Tasy, “Exact zero skew algorithm,” ICCAD-91.
- To ensure the delay from the **tapping point** to leaf nodes of subtrees T_1 and T_2 being equal, it requires that

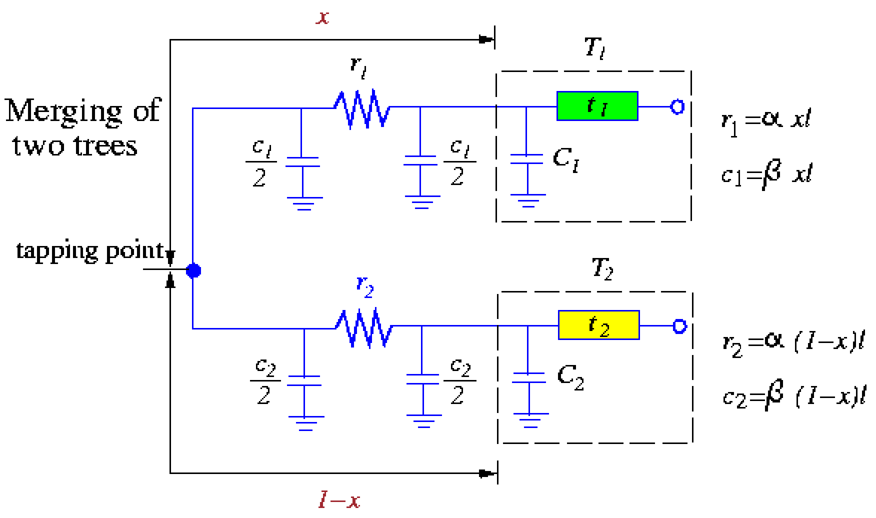
$$r_1\,(c_1/2 + C_1) + t_1 = r_2\,(c_2/2 + C_2) + t_2.$$

- Solving the above equation, we have

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

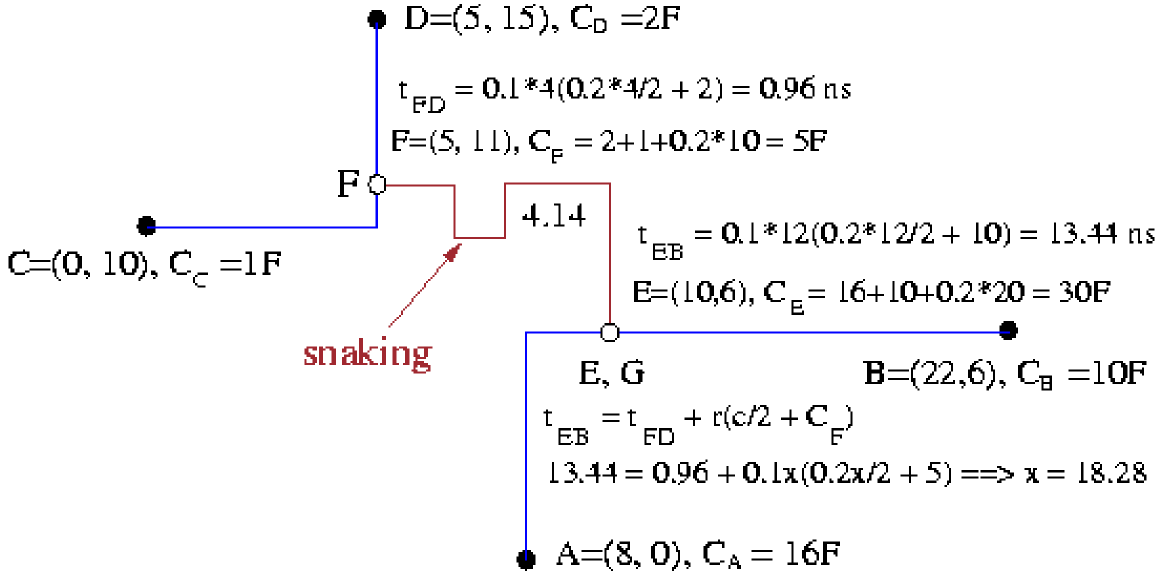
where α and β are the per unit values of resistance and capacitance, l the length of the interconnecting wire, $r_1 = \alpha x l$, $c_1 = \beta x l$, $r_2 = \alpha(1 - x)\,l$, $c_2 = \beta(1 - x)l$.





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; $r_1 = \alpha x l$, $c_1 = \beta x\, l$; $r_2 = \alpha(1 - x)\, l$, $c_2 = \beta\,(1 - x)\, l$.
- If $x \notin [0, 1]$, we need **snaking** to find the tapping point.
- Exp: $\alpha = 0.1\ \Omega$ /unit, $\beta = 0.2$ *F/unit* (tapping points: *E*, *F*, *G)*



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

Deferred-Merge Embedding (DME) for Zero Skew Clock Routing

- Chao et.al., “Zero skew clock routing with minimum wirelength,” IEEE TCAS-92.
- Deferred Merge Embedding (DME) is a liner time algorithm which optimally embeds any given topology in the Manhattan plane (with exact zero skew and minimum total wirelength).
- Two phases: bottom up and top down processes.

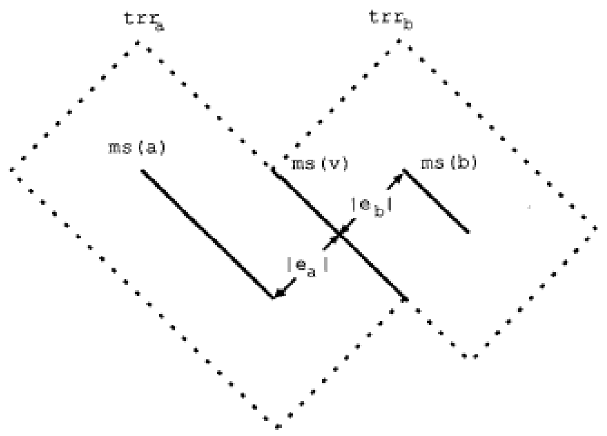
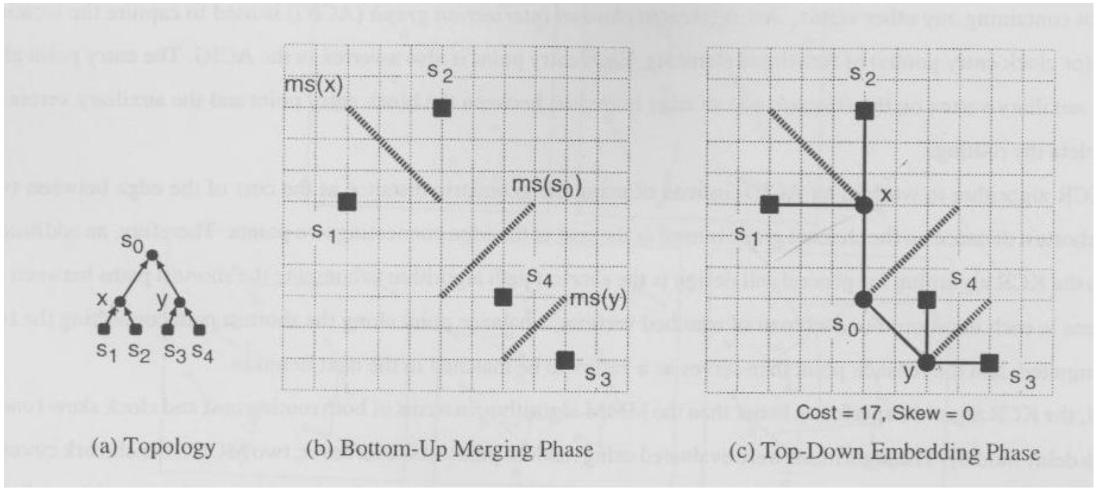


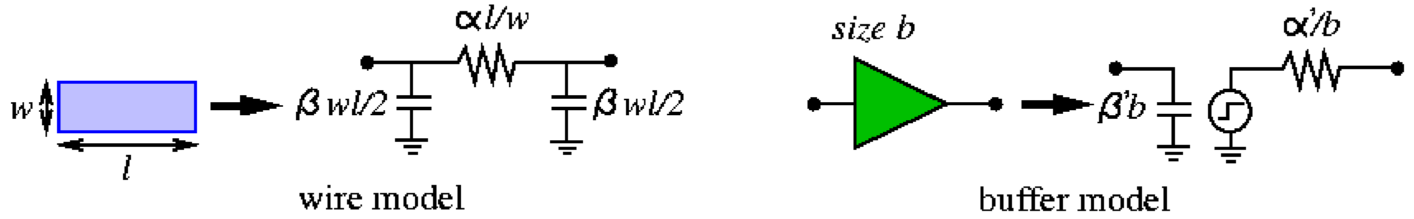
Fig. 2. Construction of merging segment $ms(v)$ when the merging cost equals κ.

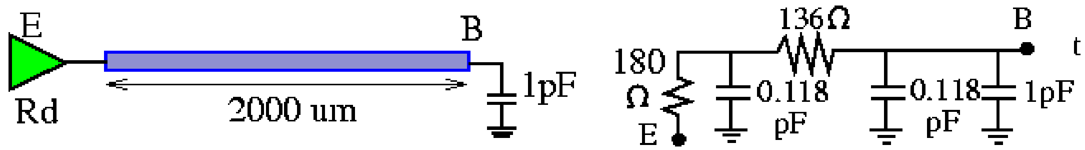


(a) Topology (b) Bottom-Up Merging Phase (c) Top-Down Embedding Phase

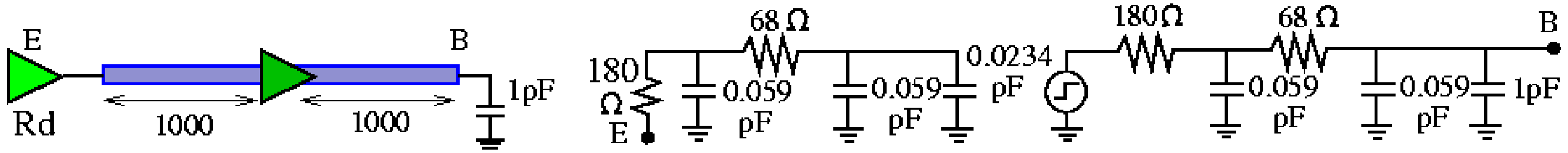
Delay Computation for Buffered Wires

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

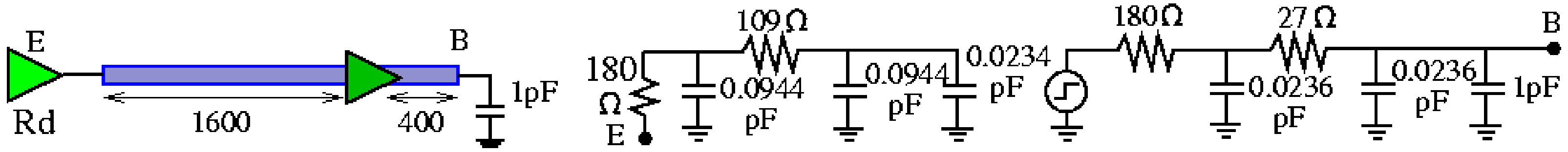




t_{EB} = 180(0.118+0.118+1)+136(0.118+1)
= 375 psec



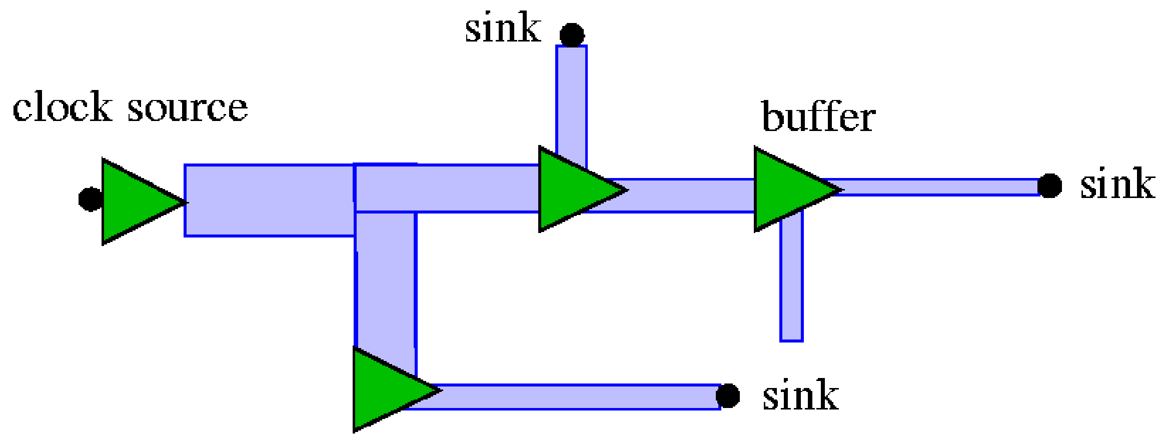
t_{EB} = 180(0.059+0.059+0.0234)+68(0.059+0.0234)
+ 180(0.059+0.059+1)+68(0.059+1) = 304 psec



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

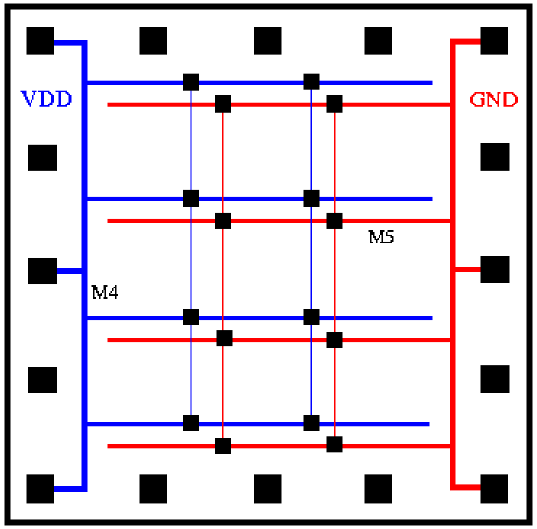
Buffering and Wire Sizing for Skew Minimization

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

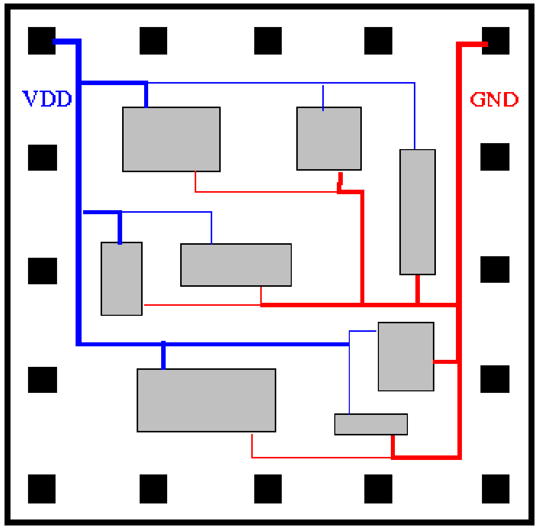


Power/Ground Network Generation

- Are usually laid out entirely on metal layers for smaller parasitics.
- Two steps:
  1. **Construction of interconnection topology:** non-crossing power, ground trees.
  2. **Determination of wire widths:** prevent metal migration, keep voltage 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



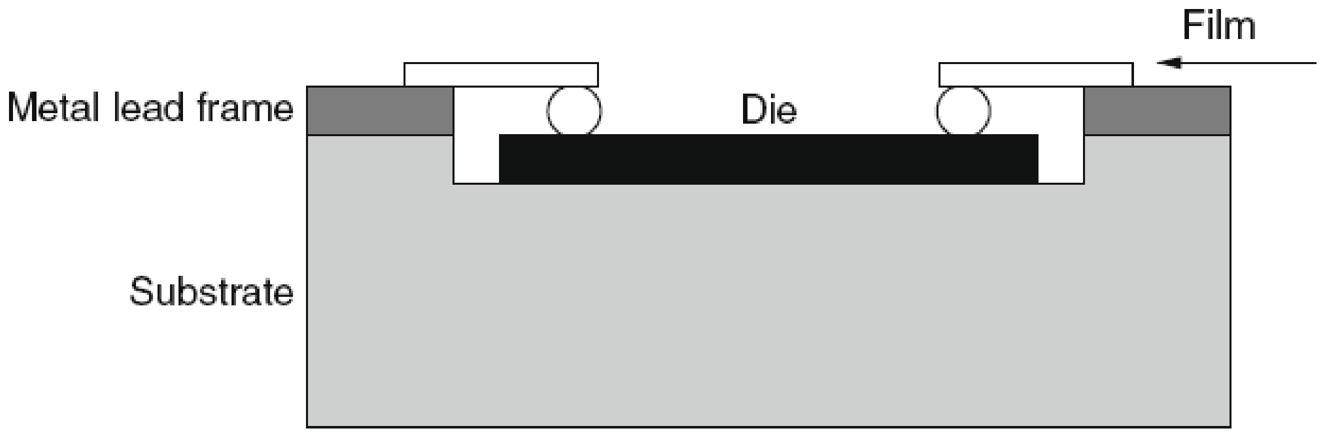
interdigitated trees

Power/Ground Topologies

- As in the case of clock network design, it is common to see a combination of these various topologies in a single P/G network.
- Because of its robustness, a mesh structure typically sits at the topmost level in the hierarchy of a P/G network.
- Comblike structures and tree topologies, with wire width tapering, are usually used for the local distribution of power supply.

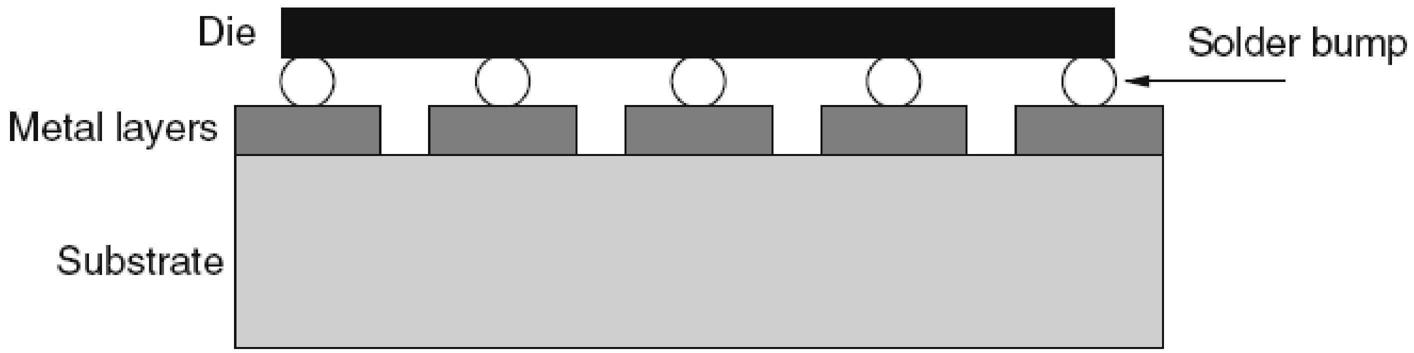
Power/Ground Topologies

- Packaging technologies also play a significant role in enhancing the robustness of power supply.
- One of the most common interfaces for external power being supplied to an IC is along the periphery of the die.



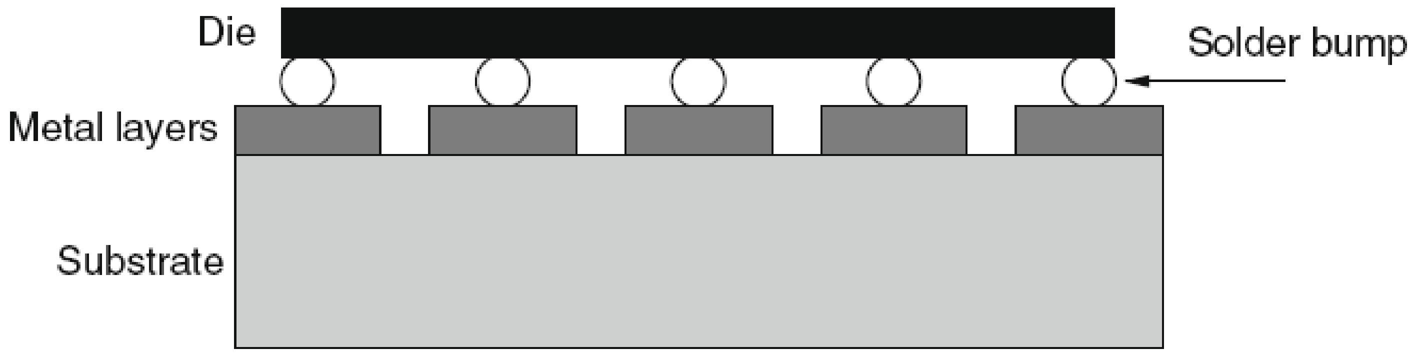
Power/Ground Topologies

- Flip-chip packaging makes it possible to supply external power into the interior of the die area directly.
- For flip-chip mounting, VDD and GND pads are distributed across the topmost layer. The die is flipped upside down and connected to the substrate of the package with solder bumps.



Power/Ground Topologies

- Flipchip packaging provides two benefits: the power supply is available at any position on the chip and the parasitic inductances and capacitances of such packages are lower.
- Used in conjunction with a power mesh, the VDD (or GND) pads usually reside on the grid points of the mesh.



Power/Ground Network Synthesis

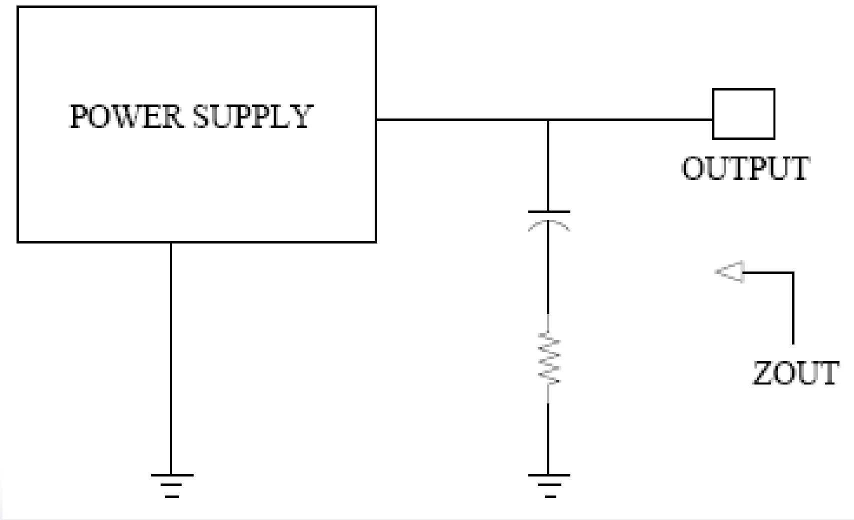
- Problems
  - Determine topology of P/G network
  - Place power pads
  - Insert decoupling capacitances
- Constraints
  - Maximum current density for each wire
  - Maximum voltage drop at each node
- Minimize wiring resource consumption

Topology Optimization

- Recursively partition the chip
- Assign coarse grid
  - Wide wires
  - Large Pitch
- Recursively biparition the grid
  - Replace wire with narrower wires
- Stop when we satisfy the two constraints
- Manageable computation due to use of locality
  - Solve small grid in each iteration

Decoupling Capacitance

- Decoupling capacitor placed next to load
  - Reduces size of current loop
  - Reduces current induced noise
    - IR, Ldi/dt
- 10% of chip area needed for decoupling capacitors



Decoupling Capacitance Placement

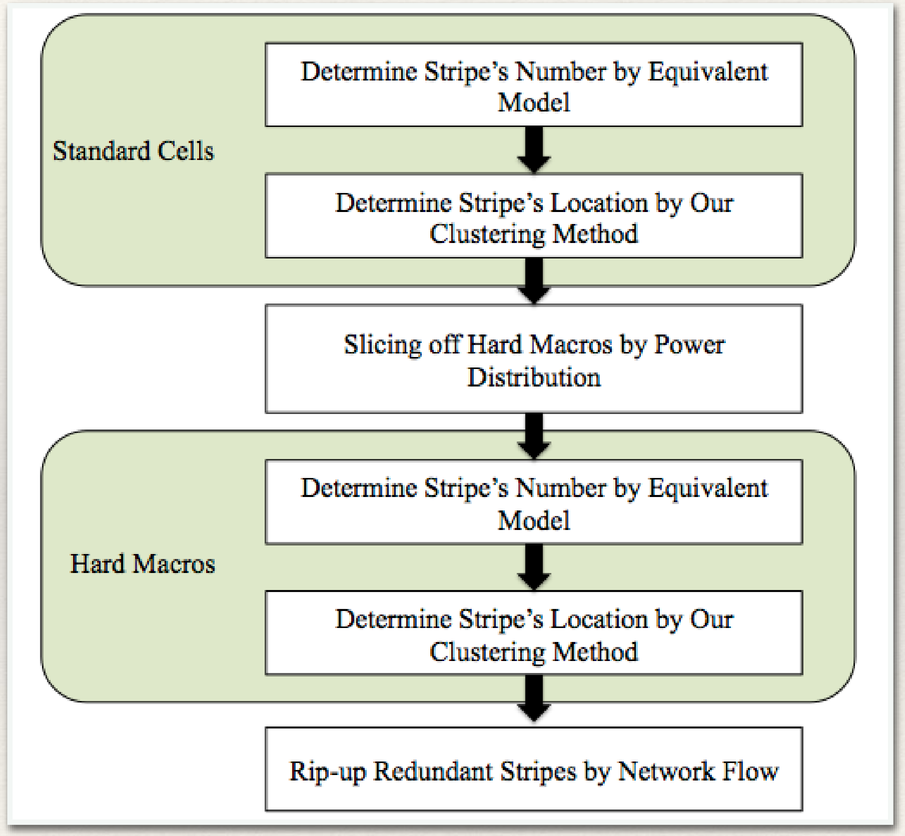
- Greedy Scheme
- Allocate to each module

$$C = \frac{\textit{Maximum total charge}}{\textit{Maximum allowed noise}}$$

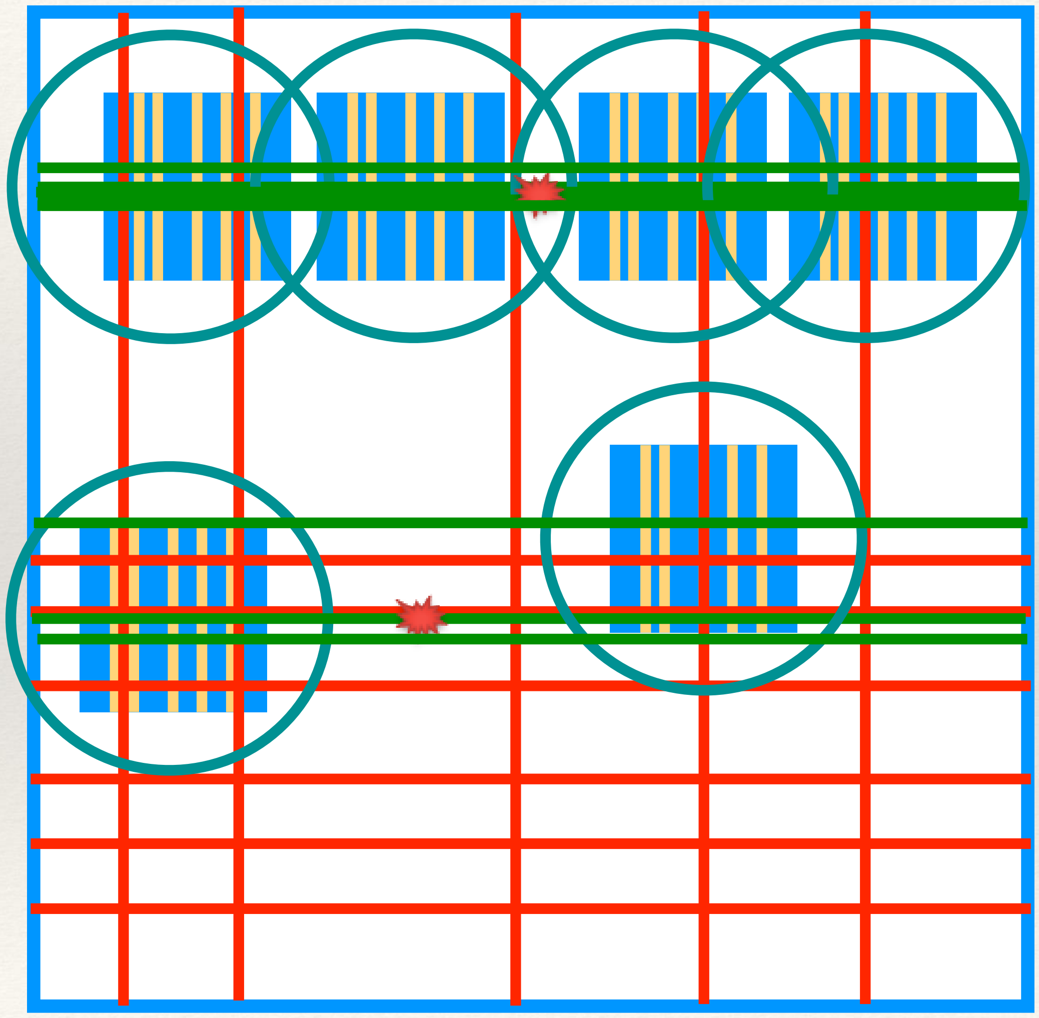
- Results in over allocation of capacitance
  - Sharing between modules not take into account
- Iterative scheme
  - Recalculate allowable noise for each module once decoupling capacitance has been allocated to any module

Prototyping PDN Design Flow

- The prototyping stage is divided into two stage, one is for standard cells and the other stage is for hard macros.
- Considering the dimension of hard macros, we slice of hard macros into several blocks and allocate its expected power consumption to those blocks.


Standard Cells
Determine Stripe's Number by Equivalent Model
Determine Stripe's Location by Our Clustering Method
Slicing off Hard Macros by Power Distribution
Hard Macros
Determine Stripe's Number by Equivalent Model
Determine Stripe's Location by Our Clustering Method
Rip-up Redundant Stripes by Network Flow

The Process of Prototyping PDN



Synthesizing PDN of Standard Cells

Synthesizing PDN of Each Macro

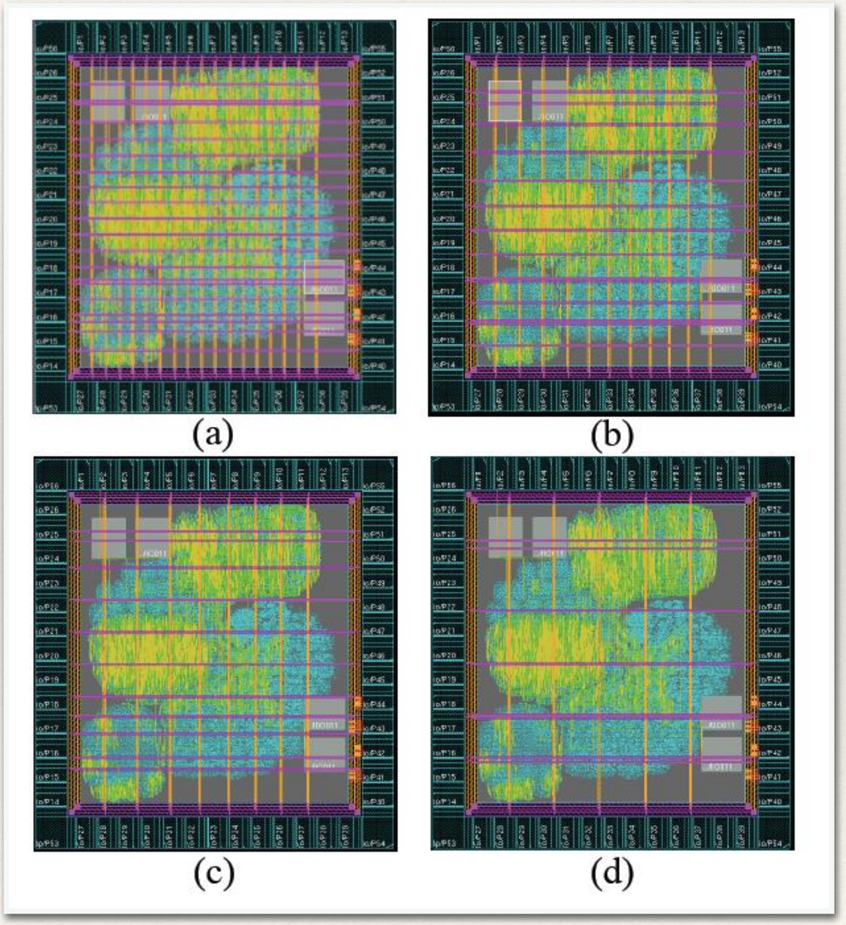
Rip Up
Redundant Nets

Check Spacing of Each Stripe

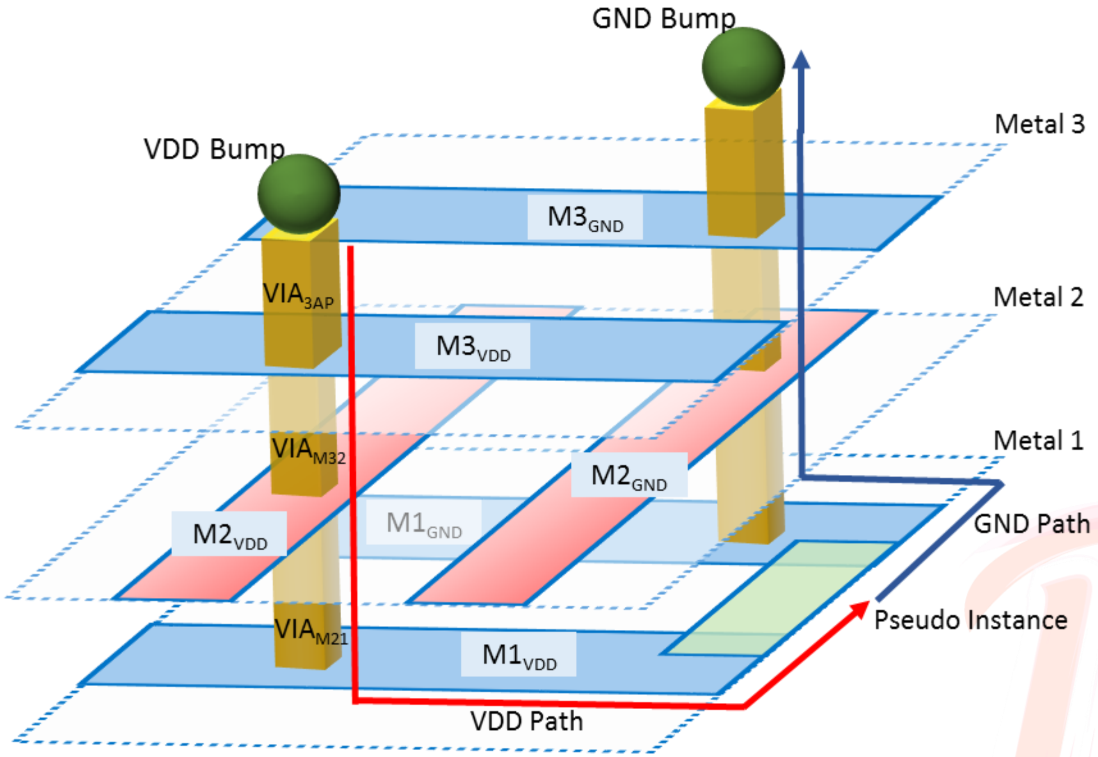
Evenly Distribute the Violation Stripe

Experimental Result on Our PNS

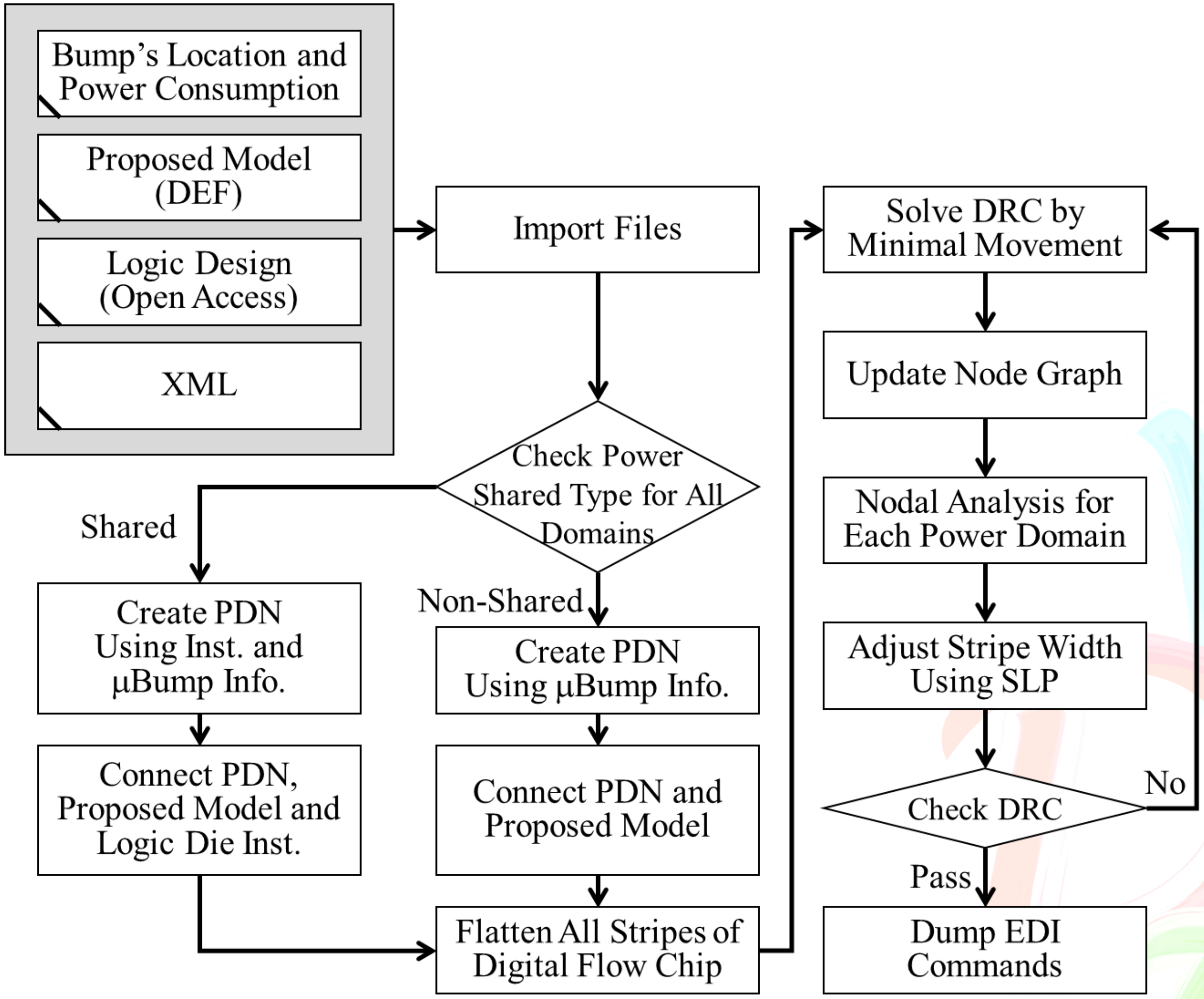
- The topology of our framework with different IR-Drop's constraints, (a)3%,(b)4.5%, (c)6% and (d)10%.



Proposed Equivalent Model on Heterogeneous Chip Power Delivery Modeling and Co-synthesis (ASP-DAC17)



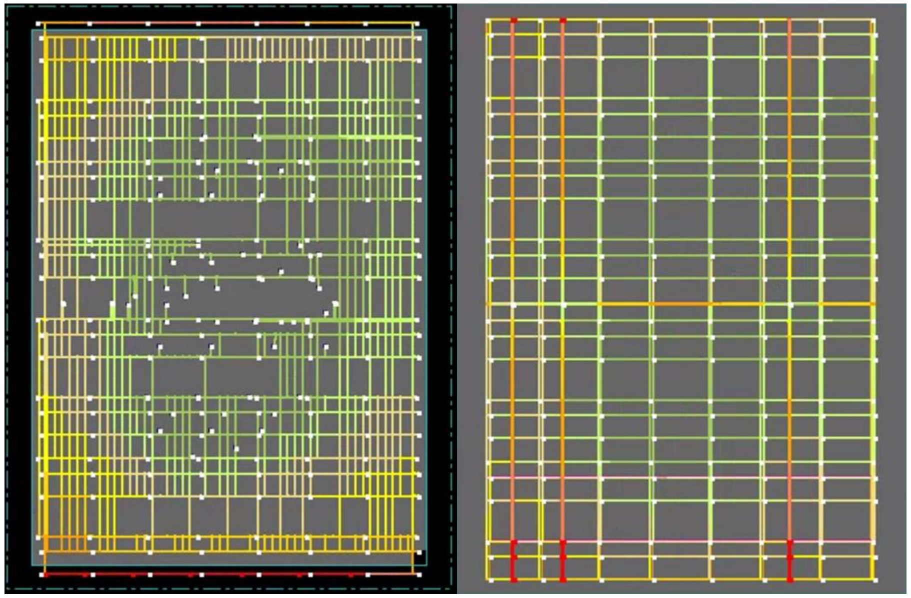
Co-synthesis Flow



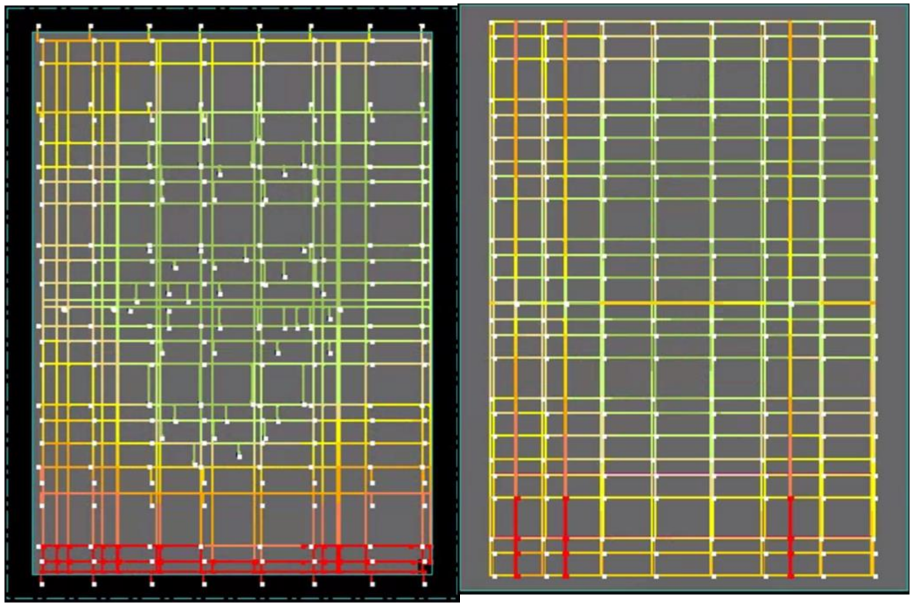
Bump's Location and Power Consumption
Proposed Model (DEF)
Logic Design (Open Access)
XML
Import Files
Check Power Shared Type for All Domains
Shared
Non-Shared
Create PDN Using Inst. and μBump Info.
Connect PDN, Proposed Model and Logic Die Inst.
Create PDN Using μBump Info.
Connect PDN and Proposed Model
Flatten All Stripes of Digital Flow Chip
Solve DRC by Minimal Movement
Update Node Graph
Nodal Analysis for Each Power Domain
Adjust Stripe Width Using SLP
Check DRC
No
Pass
Dump EDI Commands

Effectiveness of our proposed model

- PDN of STACK1 is synthesized without using proposed model
- PDN of STACK1 is co-synthesized with proposed model


STACK1
Worst vol.:1.15257v

STACK2
Worst vol.:1.11365v


STACK1
Worst vol.:1.14123v

STACK2
Worst vol.:1.0961v

- Result of STACK2 is using proposed equivalent model

Summary: Global Routing

- A preparatory step to detailed routing
- Formulated differently for different design styles
  - Gate-array: fixed capacity horizontal and vertical channels
  - Standard-cell: minimize channel congestion and overall connection length
  - Building-block: minimize the required routing space and overall interconnection length
- Maze routing is still considered a good tool in global routing stage
  - Can be modified to fit into other global routing constraints or other interconnect models
  - Modified maze routing for OPC: Wu, Tsai, and Wang (ASPDAC-05)

Summary: Detailed Routing

- Channel routing is considered a well solved problem.
- Gridless routing is another important topic due to more flexible usage in routing resource
- Hierarchical and multilevel are keys to handle large-scale routing problems.
- Routing considerations for nanometer technology
  - Noise (crosstalk) constraints
  - Buffer insertion for timing optimization
  - Additional design rules: antenna rules, OPC (optical proximity correction) rules
  - Electromigration constraints
  - Process variation considerations

Summary: Specialized Routing

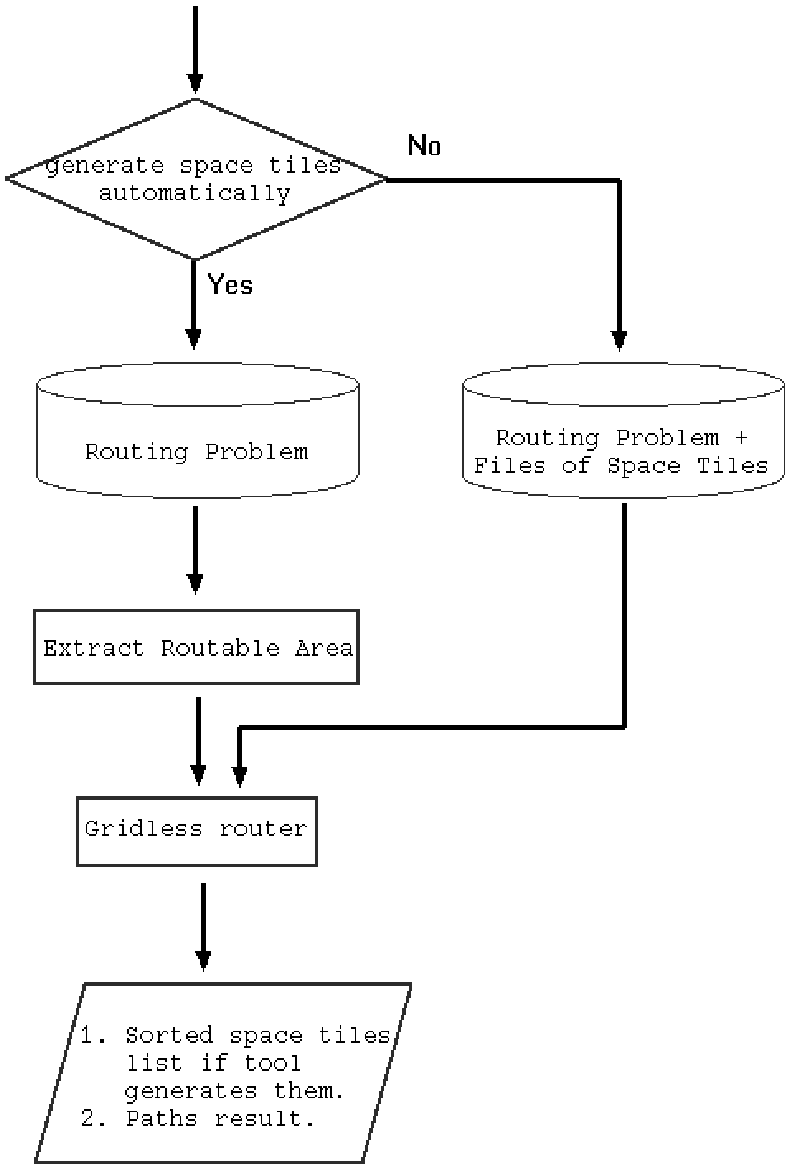
- Clock net routing
  - Traditional issues focus on zero skew or skew minimization
  - Low power CTS is a popular topic
- Power/ground routing
  - Use the minimum amount of chip area for wiring P/G networks while avoiding potential reliability failures due to electromigration and excessive IR drops
  - Power integrity driven design methodology: Wu and Chang (DAC-04)
- Scan chain reordering and routing
  - Low power DFT
  - Routing resource estimation for testing circuits

Supplemental: ISPD22 Routing Survey

- Complex design rules
  - ISPD 2018/2019 DR contests
- GPU accelerated routing
- Timing driven GR
- PCB routing
- Analog routing

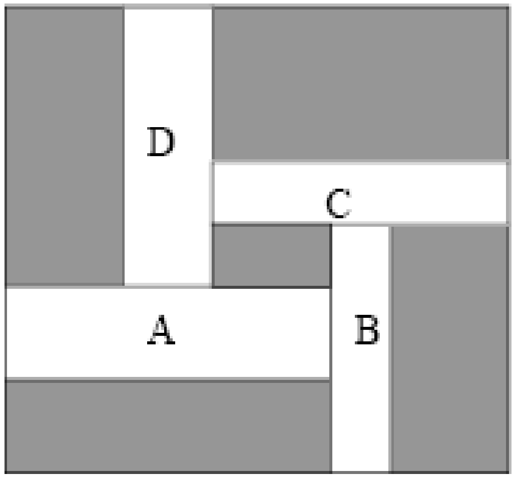
2001 MOE IC/CAD Contest Problem 4

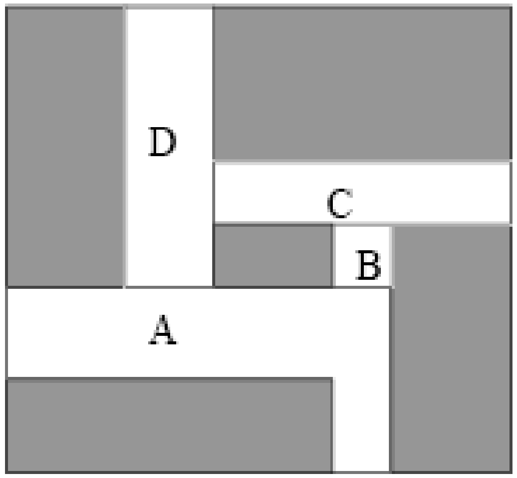
- Gridless router
  - Use corner stitching data structure


generate space tiles automatically
No
Yes
Routing Problem
Routing Problem + Files of Space Tiles
Extract Routable Area
Gridless router
1. Sorted space tiles list if tool generates them.
2. Paths result.

2002 MOE IC/CAD Contest Problem 6

- Generalized L-shaped channel router
  - To get rid of cyclic channel routing precedence constraints
  - Can be routed in the following order: B, C, D, A





2004 MOE IC/CAD Contest Problem 2

- Low power clock tree synthesis (also in 2001 P6 (original) and 2003 P5 (with useful skew))

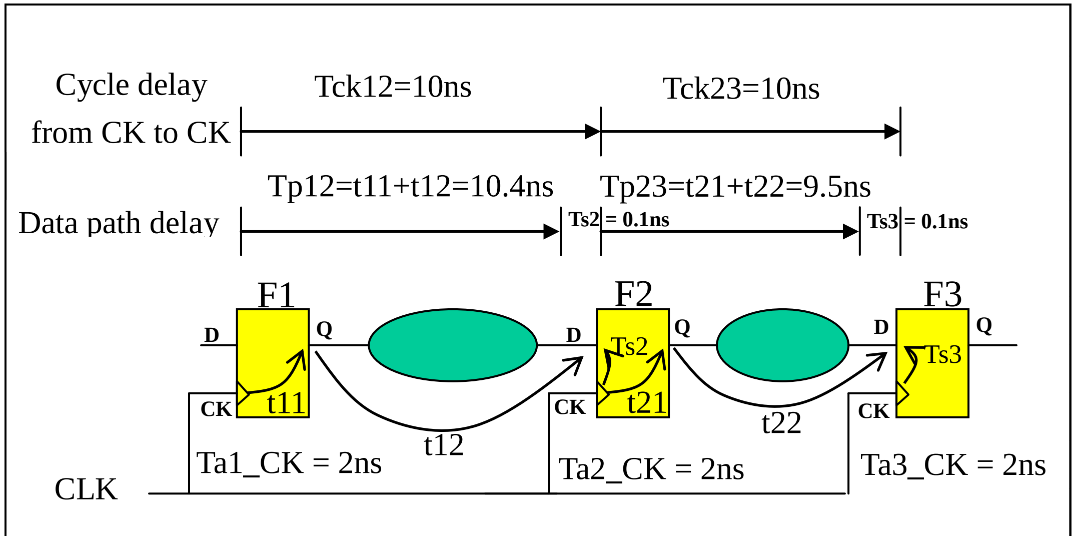


Fig 1. A design with timing violations

2003 MOE IC/CAD Contest Problem 5

- How to use skew to fix timing violation

t11: cell delay from F1.CK to F1.Q	**t12**: path delay from F1.Q to F2.D
Tp12: path delay from F1.CK to F2.D (t11+t12)	**Ta1_CK**: arrival time of F1.CK
Tck12: the cycle delay from F1.CK to F2.CK	**Ts2**: setup time of F2.D w.r.t. F2.CK
Tr2_D: the required time of F2.D	**Ta2_D**: the arrival time of F2.D

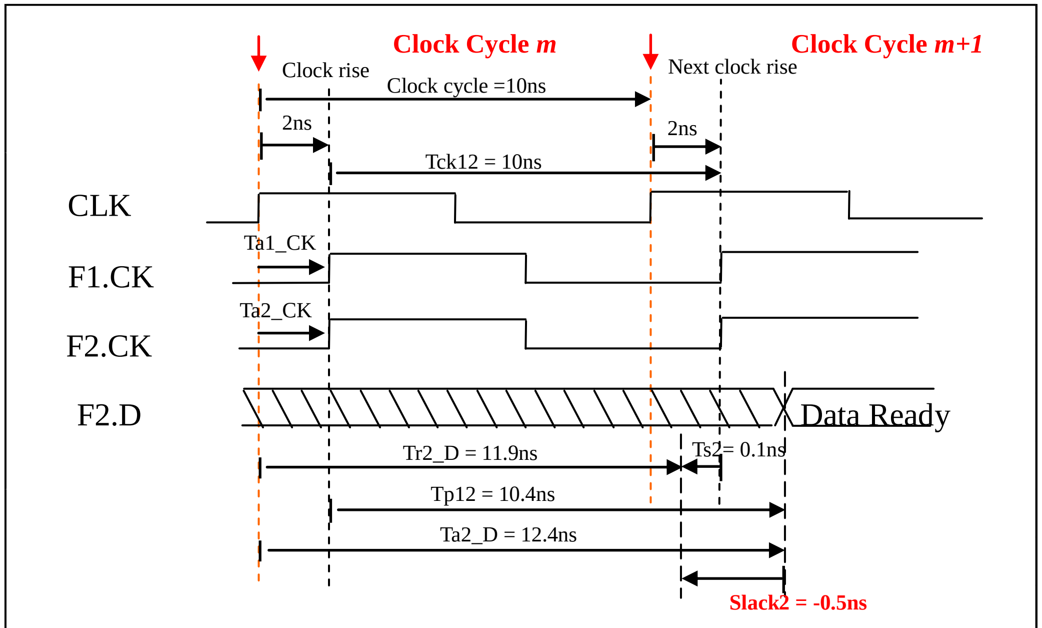


Fig 2. Timing diagram of Fig. 1

2003 MOE IC/CAD Contest Problem 5

In Fig 1, the clock latency and clock skew of CLK are 2ns and 0ns, respectively. Assume that the clock cycle time is 10ns, both of Ts2 and Ts3 are 0.1ns, and the data path delays of Tp12 and Tp23 are 10.4ns and 9.5ns, respectively. By referring to the timing diagram shown in Fig. 2, the data launched at F1 (F2) at clock cycle *m* is to be captured at F2 (F3) at clock cycle *m+1*. Thus, the data required time of F2.D and F3.D are as follows.

Tr2_D = (Clock cycle time) + (F2.CK clock latency) – (F2.D setup time)
= 10ns + 2ns – 0.1ns = 11.9ns

Tr3_D = (Clock cycle time) + (F3.CK clock latency) – (F3.D setup time)
= 10ns + 2ns – 0.1ns = 11.9ns

And, the data arrival times are

Ta2_D = (F1.CK clock latency) + (F1.CK to F2.D path delay)
= 2ns + 10.4ns = 12.4ns

Ta3_D = (F2.CK clock latency) + (F2.CK to F3.D path delay)
= 2ns + 9.5ns = 11.5ns

Thus, we can find the timing slacks are

Slack2 = Tr2_D – Ta2_D = -0.5ns ←--------- Negative timing slack (setup time)

Slack3 = Tr3_D – Ta3_D = 0.4ns ←--------- Meet timing requirement.

There is 0.5ns negative timing slack in the critical path from F1.CK to F2.D.

2003 MOE IC/CAD Contest Problem 5

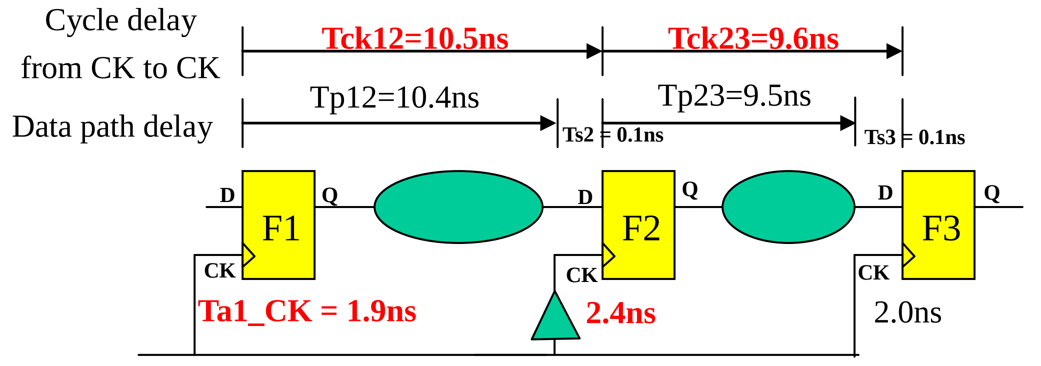


Fig 3. A design without timing violation

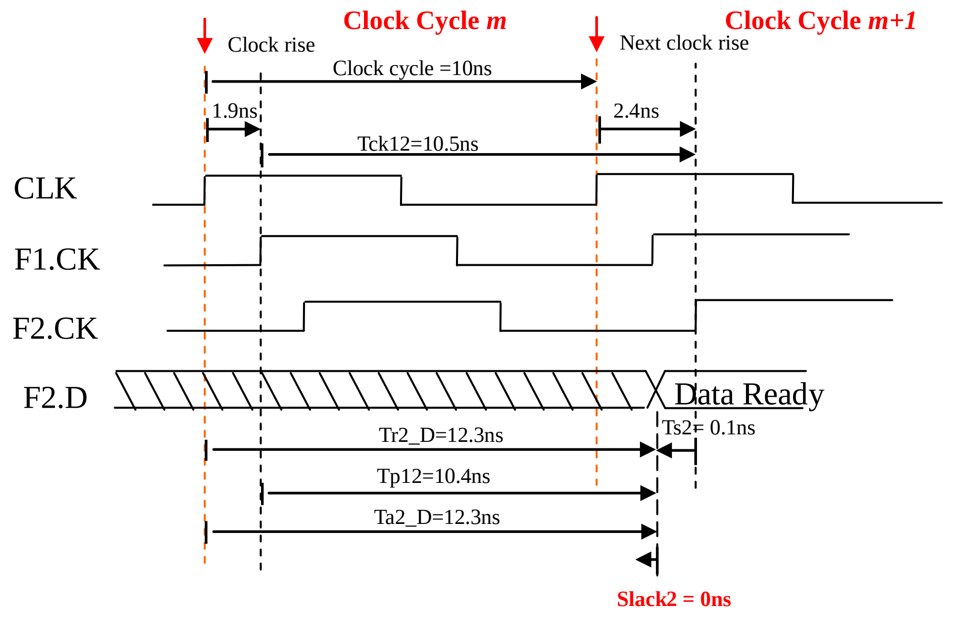


Fig 4. Timing diagram of Fig. 3

2003 MOE IC/CAD Contest Problem 5

To resolve the timing violation above, one can adjust the clock latencies of F1 and F2, and let F1 launches data earlier and F2 captures data later to compensate the negative timing slack. Fig 3 demonstrates an example of fixing the timing violation, in which the data required times of F2.D and F3.D are

Tr2_D = (Clock cycle time) + (F2.CK clock latency) – (F2.D setup time)
= 10ns + 2.4ns – 0.1ns = 12.3ns

Tr3_D = (Clock cycle time) + (F3.CK clock latency) – (F3.D setup time)
= 10ns + 2ns – 0.1ns = 11.9ns

And, the data arrival times are

Ta2_D = (F1.CK clock latency) + (F1.CK to F2.D path delay)
= 1.9ns + 10.4ns = 12.3ns

Ta3_D = (F2.CK clock latency) + (F2.CK to F3.D path delay)
= 2.4ns + 9.5ns = 11.9ns

Now there is no more setup time violations as follows.

Slack2 = Tr2_D – Ta2_D = 0.0ns ←--------- Meet timing requirement.

Slack3 = Tr3_D – Ta3_D = 0.0ns ←--------- Meet timing requirement.