

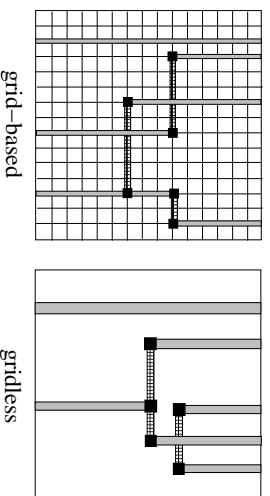
Order of Routing Regions and L-Channels

- (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.
-

201

Routing Models

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



203

Routing Considerations

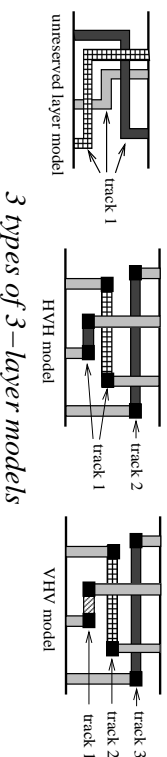
- Number of terminals (two-terminal vs. multi-terminal nets)
- Net widths (power and ground vs. signal nets)
- Via restrictions (stacked vs. conventional vias)
- Boundary types (regular vs. irregular)
- Number of layers (two vs. three, more layers?)
- Net types (critical vs. non-critical nets)

202

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).

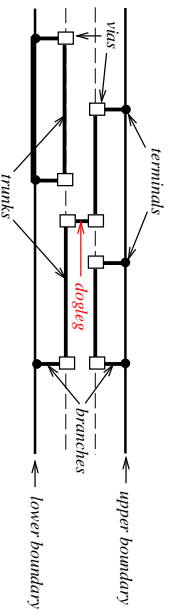
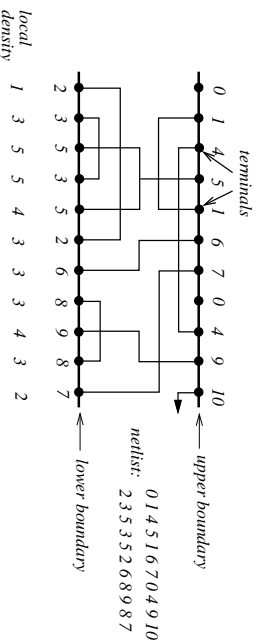
- Two-layer: HV (horizontal-Vertical), VH
- Three-layer: HVH, VHV



3 types of 3-layer models

204

Terminology for Channel Routing Problems

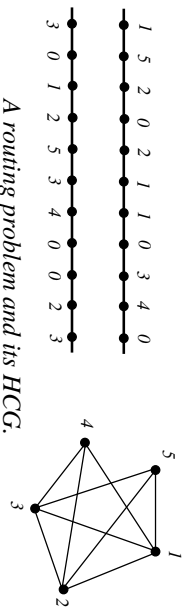


- Local density at column i : total # of nets that crosses column i .
- Channel density: maximum local density; # of horizontal tracks required \geq channel density.

205

Horizontal Constraint Graph (HCG)

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



A routing problem and its HCG.

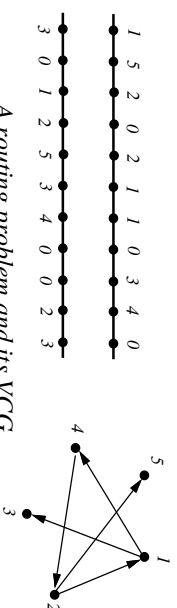
Channel Routing Problem

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

206

Vertical Constraint Graph (VCG)

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



A routing problem and its VCG.

207

208

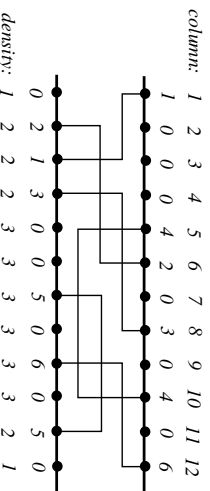
2-L Channel Routing: Basic Left-Edge Algorithm

- 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).

209

Basic Left-Edge Example

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



211

Basic Left-Edge Algorithm

```

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

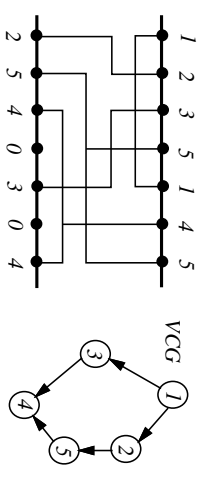
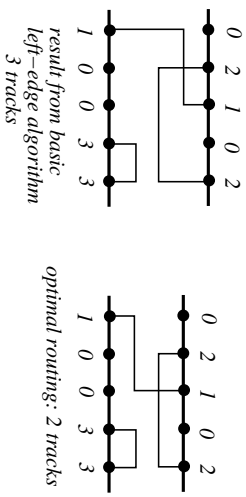
1 begin
2  $U \leftarrow \{I_1, I_2, \dots, I_n\}$ ;
3  $t \leftarrow 0$ ;
4 while ( $U \neq \emptyset$ ) do
5    $t \leftarrow t + 1$ ;
6   watermark  $\leftarrow 0$ ;
7   while (there is an  $I_j \in U$  s.t.  $s_j > watermark$ ) do
8     Pick the interval  $I_j \in U$  with  $s_j > watermark$ ,
 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

210

Basic Left-Edge Algorithm

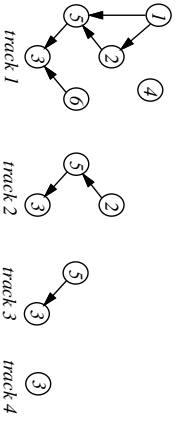
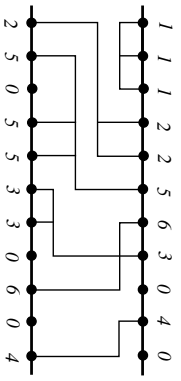
- If there is no vertical constraint, the basic left-edge algorithm is optimal.
- If there is any vertical constraint, the algorithm no longer guarantees optimal solution.

212



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 .



Constrained Left-Edge Algorithm

Algorithm: Constrained Left-Edge($U, track[j]$)
 U : set of unassigned intervals (nets) I_1, \dots, 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.

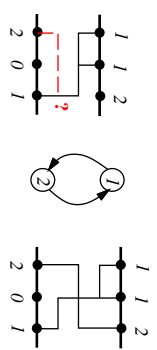
```

1 begin
2  $U \leftarrow \{I_1, I_2, \dots, I_n\}$ ;
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

```

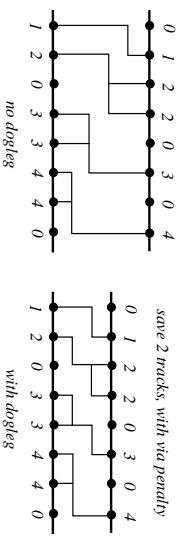
Dogleg Channel Router

- 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.



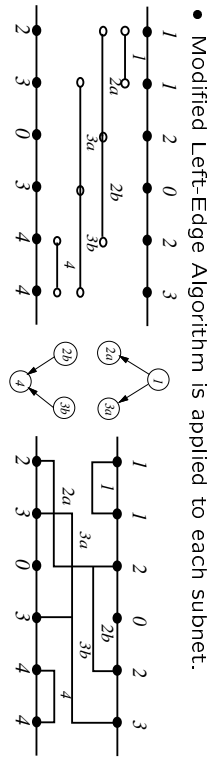
Yoshimura-Kuh (YK) Algorithm

- Yoshimura & Kuh, "Efficient algorithms for channel routing," IEEE TCAD, Jan. 1982.
- YK algorithm considers both HCG and VCG.
- 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 two major steps:
 - Zone representation of horizontal segments.
 - Merging of nets.

- The two steps are carried out to minimize vertical constraints and track assignment.
- Same idea can be extended to three-layer channel routing (Chen & Liu, IEEE TCAD, 1984)

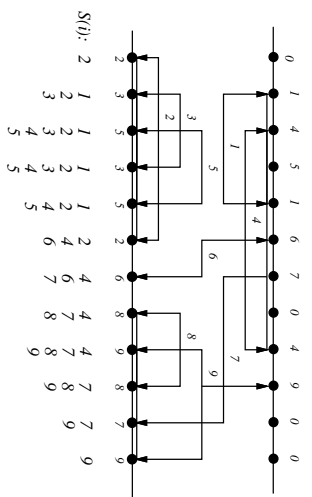
Dogleg Channel Router

- 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.



Zone Representation of Horizontal Segments

- Zones are maximal clique in the interval graph of horizontal net segments.
- $S(i)$: set of nets whose horizontal segments intersect Column i .
- Zone numbers are assigned to the columns at which $S(i)$ is maximum.



$S(i):$

2	1	1	1	1	1	2	4	4	4	4	7	7	9
2	2	2	2	2	4	6	7	7	8	9	9	9	9
3	3	3	3	4	4	6	7	8	8	9	9	9	9
4	4	4	4	5	5	7	8	8	9	9	9	9	9
5	5	5	5	5	5	7	8	8	9	9	9	9	9

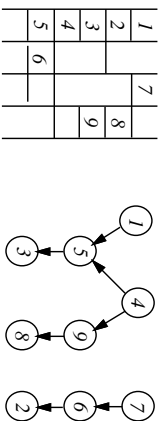
zone representation

Z1	Z2	Z3	Z4
1	1	1	7
2	2	2	8
3	3	3	9
4	4	4	9
5	5	5	9

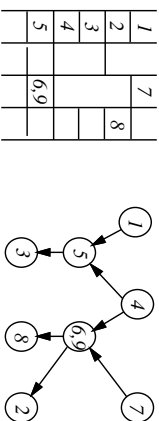
Merging of Nets

- Two nets n_i and n_j can be merged if
 - there is no edge between v_i and v_j in HCG;
 - no directed path exists between v_i and v_j in VCG.

nets 6 and 9 can be merged:

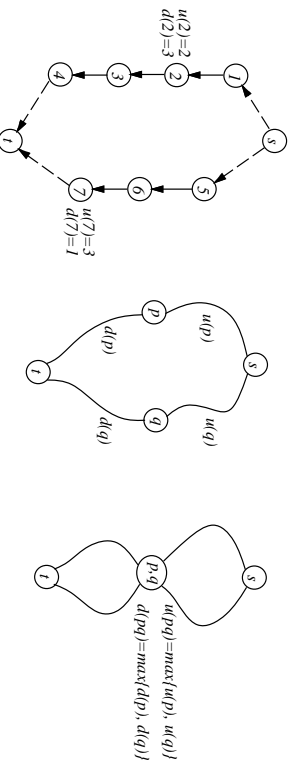


updated graph and zone representation:

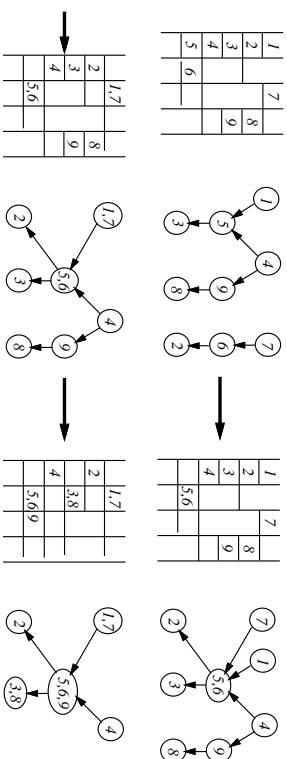


Minimizing the Longest Path

- Merge 2 nodes so as to minimize the increase of the longest path length in the VCG.
- Heuristic rule to select nets to merge sequentially.



Zone Processing



- Track 1 (Track 2): net (1, 7) or net 4; track 3: net (5, 6, 9); track 4 (track 5): net 2 or net (3, 8)

Algorithm for Merging Nets

Algorithm: Net_Merging(z_s, z_t)

z_s : Leftmost zone;
 z_t : rightmost zone;
 1 begin
 2 $L \leftarrow \{\}$;
 3 $z_s \leftarrow$ leftmost zone; $z_t \leftarrow$ rightmost zone;
 4 for $z \leftarrow z_s$ to z_t do
 5 $L \leftarrow L +$ {nets which terminate at or before zone z };
 6 $R \leftarrow$ {nets which begin at zone $z + 1$ };
 7 Merge L and R so as to minimize the increase of the longest path in the vertical constraint graph;
 8 $L \leftarrow L - \{n_1, n_2, \dots, n_j\}$; where $\{n_1, n_2, \dots, n_j\}$ are the nets merged at Step 7;
 9 end

Implementation

- Procedure to select 2 nodes for merging
 - For $m \in LEFT$ (set of nodes on the left path)

$$f(m) = C_{\infty} \times \{u(m) + d(m)\} + \max\{u(m), d(m)\}, C_{\infty} \gg 1$$
 - For $n \in RIGHT, m \in LEFT$

$$g(n, m) = C_{\infty} \times h(n, m) - \{\sqrt{u(m) \times u(n)} + \sqrt{d(m) \times d(n)}\}, \text{ where } h(n, m) = \max\{u(n), u(m)\} + \max\{d(n), d(m)\} - \max\{u(n) + d(n), u(m) + d(m)\}.$$

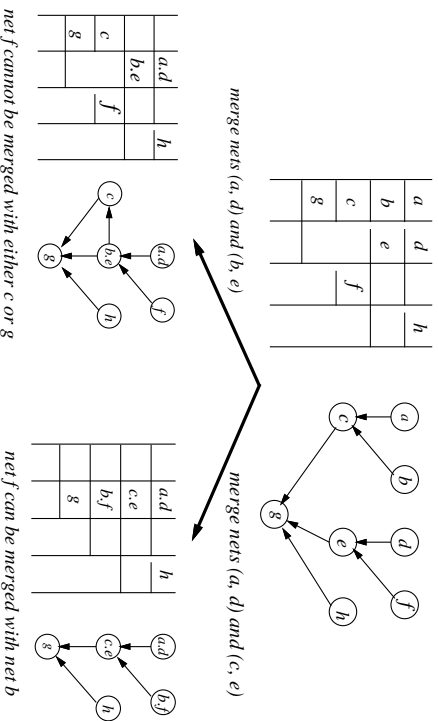
- Two steps
 - Find $m^* \in LEFT$ which maximizes $f(m)$
 - Find $n^* \in RIGHT$ which minimizes $g(n, m^*)$

$$f(m^*) = C_{\infty} \times \underbrace{\{u(m^*) + d(m^*)\}}_{\text{ties on a longest path}} + \underbrace{\max\{u(m^*), d(m^*)\}}_{\text{farthest away from } s \text{ or } t}$$

$$g(n^*, m^*) = \underbrace{C_{\infty} \times h(n^*, m^*)}_{\text{min increase in longest path length}} - \underbrace{\{\sqrt{u(m^*) \times u(n^*)} + \sqrt{d(m^*) \times d(n^*)}\}}_{\substack{\text{sum of } \sqrt{u(s)} \\ \text{and } \sqrt{d(s)}}}$$

Merging Considerations

- A merging of 2 nodes may block subsequent mergings.



Algorithm for the Implementation

Algorithm: Merge_Implementation(LEFT, RIGHT)

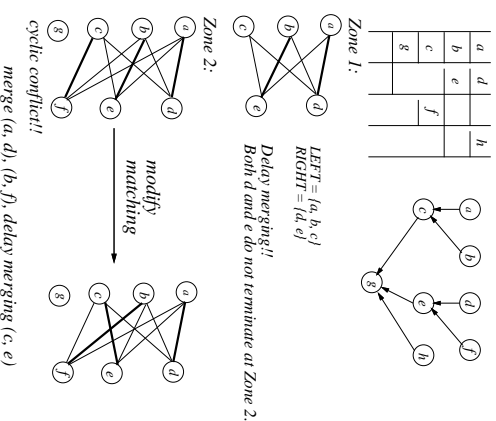
```

LEFT: left s-to-t path;
RIGHT: right s-to-t path;

1 begin
2 while LEFT ≠ ∅ do
3   Among LEFT, find m* which maximizes f(m);
4   Among RIGHT, find n* which minimizes g(n, m*), and which
   is neither ancestor nor successor of m*;
5   Merge n* and m*;
6   Remove m* and n* from LEFT and RIGHT, respectively;
7 end
    
```

Second Approach Based on Matching

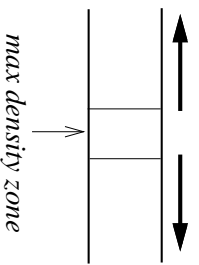
- Global merging of nets using algorithms for maximum cardinality matching.
- Delay net merging.



Comments on the YK Algorithm

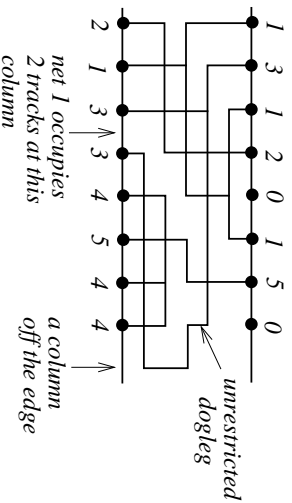
- Use a process "merging of subnets" to avoid unnecessary introduction of dogleg.
 - Subnet i and subnet j can be merged only if merging subnet i and subnet j will not increase the longest path length passing through node i and node j on VCG.
- Need not start at Zone 1. Can obtain better results by starting at the maximum density zone.

– Chan, "A new channel routing algorithm," *CIT VLSI Design Conf.*, 1983.



Greedy Channel Router

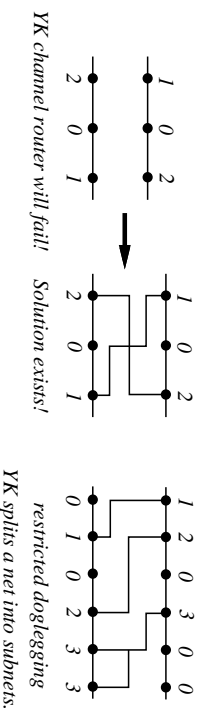
- Rivest & Fiduccia, "A greedy channel router," *DAC-82, (IEEE TCAD, May 1983)*.
- Always succeed (even if cyclic conflict is present)
- Allows unrestricted dogleg
- Allows a net to occupy more than 1 track at a given column.
- May use a few columns off the edge.



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 YK channel router does not allow unrestricted doglegging.



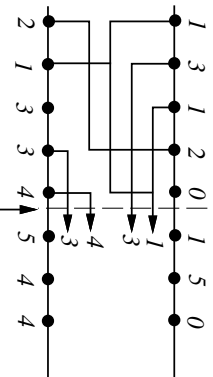
Overview of Greedy Router

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

```

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

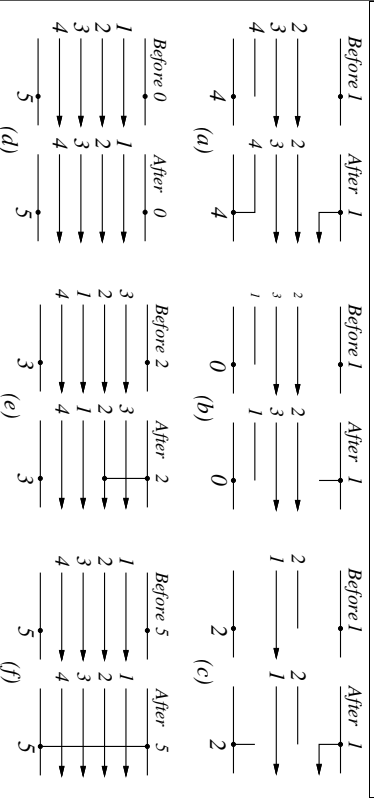
- In general, a net may be
 - (1) empty (net 5)
 - (2) unsplit (nets 1, 4)
 - (3) split (net 3)
 - (4) completed (net 2)



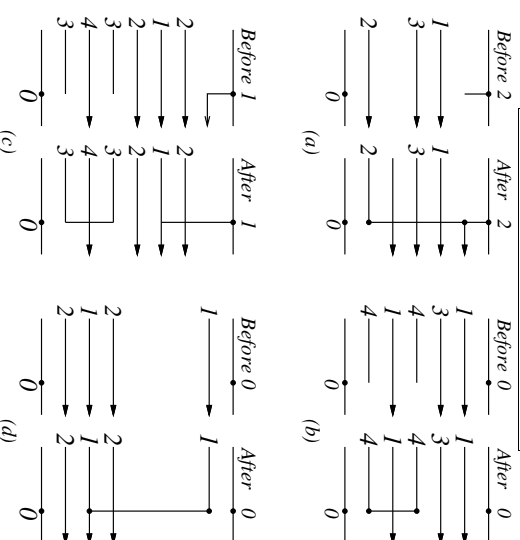
Greedy Heuristics

- 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

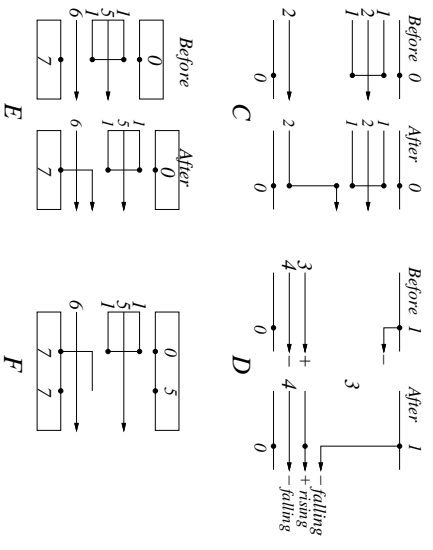


B: Collapse Split Nets

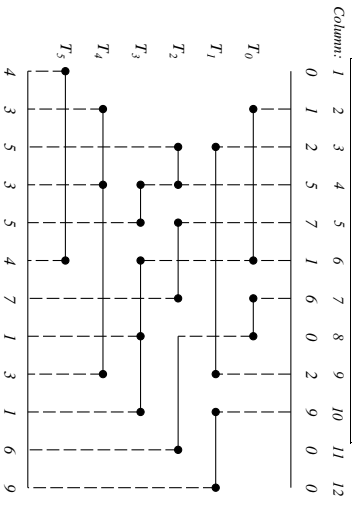


Heuristics: C, D, E, and F

- 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.



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 .

Parameters to Greedy Router

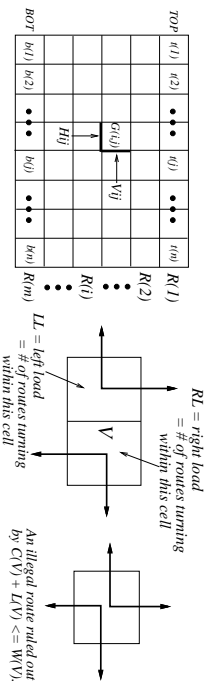
- Initial-channel-width: icw
- Minimum-jog-length: mjl
- Steady-net-constant: snc (window size in terms of # of columns; determines # of times a multipin net changes tracks)
- Usually start icw as d , the density.
- mjl controls the number of vias, use a large mjl for fewer vias.
- snc also controls # of vias. Typically, $snc = 10$.

Hierarchical Channel Router

- Burstein & Pelavin, "Hierarchical channel router," *INTEGRATION: the VLSI Journal*, 1 (1983) (Also "Hierarchical wire routing," *DAC-83, IEEE TCAD*, Oct. 1983)
- Uses a divide-and-conquer approach.
- A routing problem in $m \times n$ grid is reduced to $2 \times n$ grid.
- Each column in these subgrids is considered as a supercell.
- Capacity of each vertical boundary is the sum of corresponding boundary capacities.
- Grid is partitioned into a $2 \times n$ grid.
- Nets are routed one at a time in the $2 \times n$ grid.
- Terminal positions for the new $2 \times n$ are defined by the routing in previous hierarchy.

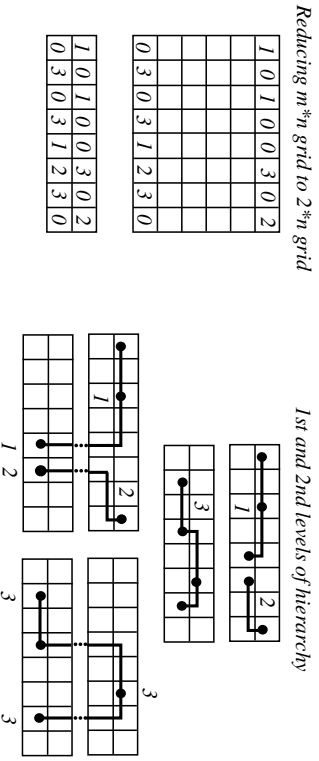
Routing Capacity Considerations

- Rectilinear channel grid: $G(i, j), i = 1, 2, \dots, m; j = 1, 2, \dots, n$; route: $R(i), i = 1, 2, \dots, m$.
- Weighted boundaries: for each horizontal boundary $H, 0 \leq W(H) \leq 1$; for a top or bottom vertical boundary $V_{ij}, W(V_{ij}) = 0, i = 1, m; j = 1, 2, \dots, n$.
- For any boundary B , let $C(B)$ be the total # of routes crossing B .
- Routing $R(1), R(2), \dots, R(m)$ is legal iff (1) $C(B) \leq W(B)$, \forall boundaries; (2) $C(V) + L(V) \leq W(V)$, \forall vertical boundaries, where $L(V) = \max\{L, RL\}$.
- All boundary capacities = 1 (except the vertical boundaries of the top and bottom rows) \Rightarrow traditional channel routing.



Example of Hierarchical Channel Router

- Define (weight) cost functions for horizontal and vertical boundaries (and boundaries outside the channel) in terms of routing congestions (functions of $W(B), C(B), L(B)$, etc).
- Find a minimal cost tree interconnecting a set of terminals located on a $2 \times n$ grid.



Comparison of Two-Layer Channel Routers

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

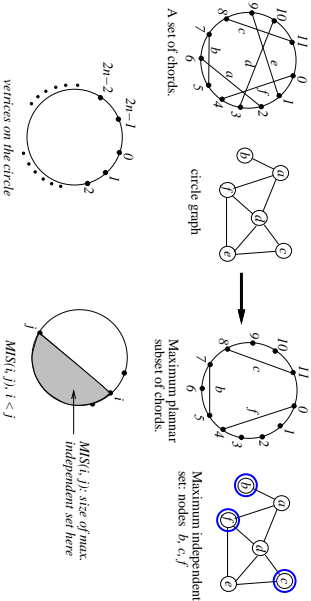
Comparison of Two-Layer Channel Routers

- 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	287	5020

Supowit's Algorithm

- Supowit, "Finding a maximum planner subset of a set of nets in a channel," IEEE TCAD, 1987.
- Problem: Given a set of chords, find a maximum planner subset of chords.
 - Label the vertices on the circle 0 to $2n - 1$.
 - Compute $MIS(i, j)$: size of maximum independent set between vertices i and j , $i < j$.
 - Answer = $MIS(0, 2n - 1)$.

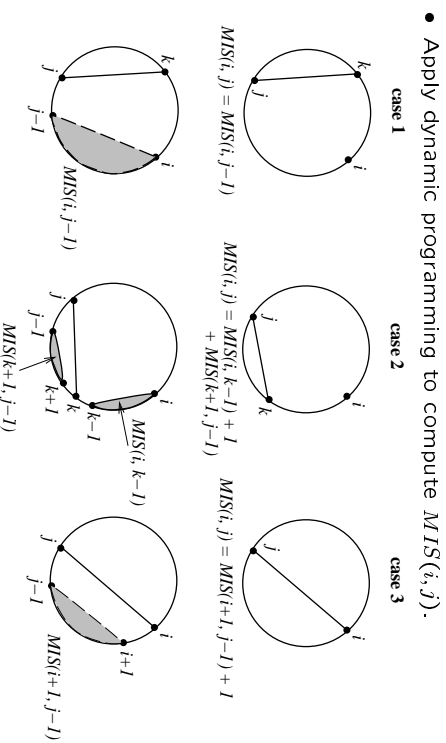


250

Final Project

- Options:
 - Algorithm implementation
 - Paper survey: stress on comparison, extensions, and future work
 - Research-oriented project (a default problem on Crosstalk provided; may choose/formulate a problem by yourself)
- Individual (preferred) or 2-person project.
- Hand-in: (1) a written report and/or (2) programs of implementation.
- Due date: 3pm for demo & report, Jan. 12, 1999 (Tuesday).
- **No late project accepted!**

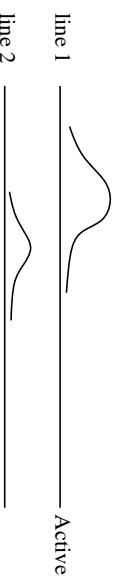
Dynamic Programming in Supowit's Algorithm



251

Crosstalk

- Effects of smaller feature size
 - Reduced switching time for interconnects and devices.
 - Lower signal transition time.
- Effect on signals
 - Increasing capacitive and inductive coupling, leading to (1) increased signal delays (2) decreased signal integrity.
- This coupling effect is called **crosstalk**.
 - Crosstalk may cause unexpected circuit switching or other undesirable behavior.
 - Crosstalk proportional to overlapping segment length and inversely proportional to distance between segments.



252

253

Crosstalk Problem

- Some references
 - Chen & Wong, "Wiring and crosstalk avoidance in multi-chip module design," *IEEE 1992 Custom Integrated Circuits Conference*.
 - Gao & Liu, "Minimum crosstalk channel routing," *ICCAD-93* (switchbox: *ICCAD-94*).
 - Thakur, Chao, Wong, "An optimal layer assignment algorithm for minimizing crosstalk for three layer VHV channel routing," *JSCAS-95*.
 - Zhou & Wong, "An optimal algorithm for river routing with crosstalk constraints," *ICCAD-96*.
 - Xue, Kun, Wang, "Post global routing crosstalk risk estimation and reduction," *ICCAD-96*.
 - Zhou & Wong, "Global routing with crosstalk constraints," *DAC-98*.
 - Tseng, Scheffer, Sechen, "Trimming and crosstalk driven area routing," *DAC-98*.
- Issues needed to be considered: **(1) routing models; (2) crosstalk computation; (3) design/optimization objectives**
- Routing resources
 - Global or detailed routing
 - Gridded or gridless channel routing: two-layer or multi-layer (VHV, HVH)

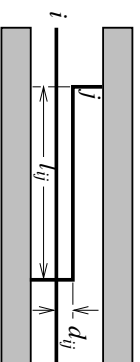
254

Problem Consideration: Objectives

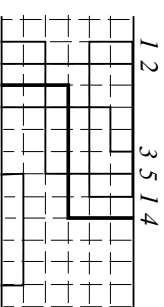
- Design channel routing algorithms or other detailed routing algorithms
- At least two objectives
 - Given an upper bound on crosstalk of each net, minimize # of tracks.
 - Given an upper bounds on the # of tracks used and on the crosstalk of a net, minimize total crosstalk $\sum_i C_i$ or maximize the minimum crosstalk slack.
- Design global routing algorithms with consideration of crosstalk avoidance
 - Given an upper bound on crosstalk of each net, minimize channel density.
 - Given an upper bound on channel density, minimize total crosstalk $\sum_i C_i$.
 - Crosstalk estimation? Cost function?

Problem Consideration: Crosstalk Computation

- Simplification in gridded channel routing
 - Adjacent tracks or columns only
 - Horizontal or vertical segments overlapping only
 - Linear distance model ($k = 1$)
- Example: $k = 1$, $C_{ij} = l_{ij}$ (adjacent tracks/columns only)
 - Net 4: Consider both vertical and horizontal segments $\Rightarrow C_{4,j} = 3 + 3 + 4 + 5 + 2 = 17$
 - Net 4: Consider horizontal segments only $\Rightarrow C_{4,j} = 4 + 5 = 9$



$$C_{ij} = a \frac{l_{ij}}{(d_{ij})^k}$$



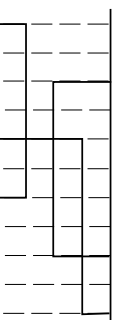
$$C_{4,j} = 3 + 3 + 4 + 5 + 2 = 17$$

255

An Example

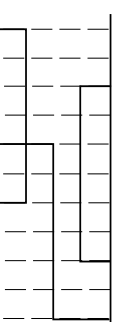
- Density is 3 for both, but configuration (b) has smaller crosstalk.
- Trade-offs
 - Model accuracy and solution quality
 - Project is graded based on model **and** algorithm quality
 - May explore formulation by considering more issues
- Test data will be available on-line.

- Display the results of routing (global routing?)



$$C = 4 + 4 + 4 + 4 = 16, D = 3$$

(a)



$$C = 4 + 4 + 4 + 2 + 2 = 12, D = 3$$

(b)

256

257

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

