Detailed Routing

Placement

Global Routing

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

Detailed Routing

- Finds actual geometric layout of each net within assigned routing regions.
- No layouts of two different nets intersect on the same layer.
- Problem is solved incrementally, one region at a time in a predefined order.
A Routing Example

Global Routing (a)

Detailed Routing (b)
Channels and Switchboxes

Routing regions divided into channels and switchboxes.
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) L-channels can be decomposed into channels and a switchbox.
Routing Considerations

- Number of terminals (i.e. two terminal vs. multiterminal nets)
- Net width (i.e. power and ground vs. signal nets)
- Via restrictions (i.e. stacked vs. conventional vias)
- Boundary type (i.e. regular vs. irregular)
- Number of layers (i.e. two vs. three layer model)
- Net types (i.e. critical vs. non-critical nets)
Routing Models

(a) Grid-based   (b) Gridless

- Grid-based model
  1. A grid is super-imposed on the routing region.
  2. Wires follow paths along the grid lines.

- Gridless model
  1. Any model that does not follow this ‘gridded’ approach.
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).
  1. Two-Layer
     - (a) HV (Horizontal-Vertical)
     - (b) VH
  2. Three-Layer
     - (a) VHV
     - (b) HVH
Detailed Routing

Terminology for Channel Routing Problems

Net list:

```
1 4 2 0 2 1 1 0 3 4 0
3 0 1 2 0 3 4 0 0 2 3
```

Channel:

- **Terminal**
- **Upper boundary**
- **Lower Boundary**
- **Upper boundary**
- **Terminals**
- **Via**
- **Tracks**
- **Dogleg**
- **Branches**
- **Trunks**
- **Lower boundary**
Channel Routing Problem Formulation

- Assignments of horizontal segments of nets to tracks.
- Assignments of vertical segments to connect
- Horizontal segments of the same net in different tracks, and
- Horizontal and vertical constraints must not be violated.

1. Horizontal span of two nets overlaps each other.
2. 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.

Channel height is minimized.
Detailed Routing

Horizontal Constraint Graph (HCG)

- HCG $G = (V, E)$ is undirected graph where $V = \{v_i | v_i$ represents a net $n_i \}$ and $E = \{(v_i, v_j) |$ a horizontal constrain exists between $n_i$ and $n_j \}$.

![Diagram of HCG](image)

A routing problem and its HCG
Vertical constraint Graph (VGC)

- VCG $G = (V, E)$ is directed graph where $V = \{v_i | v_i$ represents a net $n_i\}$ and $E = \{(v_i, v_j) |$ a vertical constrain exists between $n_i$ and $n_j\}$.

A routing problem and its VCG
Detailed Routing

Single-Layer Routing Problems

A routable single-layer routing problem

An unroutable single-layer routing problem
Single-Layer Routing Algorithms

1. General River Routing Algorithm
2. Single Row Routing Algorithm
   (a) Basic Single Row Routing Algorithm
   (b) Algorithm for Street Congestion Minimization
   (c) Algorithm for Minimizing Doglegs
**General River Routing Problem**

- A special case of single-layer routing problem.
- All terminals lie on the boundary of the region.
- All nets are two-terminal nets.
- There are no blocks in the region.
- Nets must be planar.

Example
General River Routing Algorithm

- Routing in arbitrary shaped rectilinear routing regions
- Gridless-based
- Nets are routed one-by-one.
- Each net is routed as close to the boundary as possible in counterclockwise direction.
- Guarantees to find a solution if one exists.
- The algorithm consists of following phases:
  1. Starting Terminal Assignment
  2. Net Ordering
  3. Path Searching
  4. Corner Minimization
Starting Terminal Assignment

- Each net is routed in the counter-clock direction.
- A starting terminal is assigned to each net such that the length of its layout along boundary is shorter.

T1 is starting terminal since P1 is shorter than P2.
Net Ordering

- Nets are routed in an order such that all nets whose layouts are 'contained' in the layout of the net to be routed have been routed.

**Algorithm** NET-ORDERING

begin
  for $i = 1$ to $2n$ do
    if END-TERMINAL($T_i$) then MATCHED($T_i$) = 0;
    else MARKED($T_i$) = 0;
    stack = $\phi$; $i = 1$;
  $T =$ any terminal in the circular list;
  while $i \leq n$ do
    if START-TERMINAL($T$) and MATCHED($T$) = 0
      then PUSH($T$, stack);
      MARKED($T$) = 1;
    else if END-TERMINAL($T$) and MATCHED($T$) = 0
      then $T_1 =$ POP(stack);
        if $T = T_1$ then
          MATCHED($T$) = 1; ASSIGN-NUMBER($i$, NET($T$)); $i = i + 1$;
        else exit;
    $T =$ next terminal in the circular list;
  end.
Path Searching

- Based on the net order, each net is routed as close to the pseudo-boundary as possible.
- Pseudo-boundary is formed by the boundary and the paths of nets that have been routed.
Corner Minimization

- After routing, corners are a lot more denser than the center.

- Length of nets can be minimized by flipping corners toward the inside of the routing region.
Terminology for Single Row Routing Problems (SRRP)
Basic Single Row Routing Algorithm

- Nets are represented by intervals.
- Nets or segments of nets above reference lines are mapped onto paths in the upper street while those below reference lines are mapped onto paths in the lower street.

(a) Reference lines
(b) Nets N_1, N_2, N_3, N_4, N_5
(c) Reference lines and nets
Street Congestion Minimization

- Let $q_{max}$ and $q_{min}$ be the maximum and minimum over the cut numbers of all nets respectively.
- In the example, $q_{max} = 4$ and $q_{min} = 2$.
- The problem of finding the layout with minimum congestion is NP-Hard.
- Let $Q_0$ be the maximum over the upper and lower street congestions, then $Q_0 \geq \max\{q_{min}, \left\lceil \frac{q_{max}}{2} \right\rceil \}$. 
Algorithm for Street Congestion Minimization

- Cliques are found in the interval graph.
- If the clique intersection of two neighboring (pseudo-) cliques is high, a pseudo-clique is formed by combining them.
- Maximum pseudo-clique is routed first. Other pseudo-cliques are then routed.
- Time complexity is $O(N^2 \log n)$ where $n$ is the number of intervals.

**Algorithm** SRRP-ROUTE ( )

begin
  FIND-CLIOQUES ($L, C$)
  COMBINE-CLIQUS ($L, C, D$)
  (* $D$ contains super-cliques $SC_j, j = 1, \ldots, r$ *)
  MAX-PSEUDO-CLIQUE ($D, SC_k$)
  SOLVE ($SC_k, M$)
  for $j = 1$ to $k - 1$
    INSERT ($SC_j$)
  for $j = k + 1$ to $r$
    INSERT ($SC_j$)
end.

Sherwani, Deogun, and Roy *Journal of Circuits, Systems and Computers* June 1993

Algorithms for VLSI Physical Design Automation
Detailed Routing

Dogleg Minimization

- An SRRP can be routed without doglegs. \(\iff\) Overlap graph is bipartite.

- An SRRP can be routed with at most 1 dogleg. \(\iff\) Containment graph is null.

A line is cut at most once.

At most one dogleg is used.

(a) (b) (c)
Features of Algorithm for Minimizing Doglegs

- The interval graph is decomposed into \( k \) independent sets.
- First two independent sets are routed using upper and lower streets.
- Remaining independent sets are inserted using maximum \( O(k) \) doglegs per net.
- Time complexity is \( O(n \log n) \) where \( n \) is the number of nets.

Wu, Sarrafzadeh, and Sherwani *Transactions of Circuits and Systems* May 1992

*Algorithms for VLSI Physical Design Automation*
Algorithm for Minimizing Doglegs

Algorithm K-DOGLEG-I()
begin

Phase 1:
(* Use Left-edge algorithm to decompose \( \mathcal{N} \) into \( k \) independent net lists *)
(* lists \( \mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k \) of \( \mathcal{N} \). *)
\( \mathcal{N}_i \) = LEDGE(\( \mathcal{N} \)); (\( i = 1, \ldots, k \)).

for ( \( N_i \in \mathcal{N}_1 \) i = 1, \ldots, \( m_1 \) ) do (* Assign \( \mathcal{N}_1 \) to the upper street. *)
\( T^U_i = T^U_i \cup N_i; \)
(* Assign \( \mathcal{N}_2 \) to the lower street. *)

for ( \( N_i \in \mathcal{N}_2 \) i = 1, \ldots, \( m_2 \) ) do
\( T^B_i = T^B_i \cup N_i; \)

Phase 2:
(* Insert the remaining independent net lists.*)
\( t = U; u = 1; l = 1; \)

for ( \( G^i \) i = 3, \ldots, \( k \) ) do

for ( \( N_j \in \mathcal{N}_i \) (\( j = 1, \ldots, m_i \) ) ) do

\( k = \min \{ q | 1 \leq q \leq p, N_j \in T^i_q \}; \) (* Find the smallest track which contains \( N_j \). *)

if (\( N_j \) contained by previously routed net at \( T^i_k \))
then

\( \text{INSERT}(N_j, T^i_k); \) (* Insert \( N_j \) under \( T^i_k \). *)

else

\( T^t_p = T^t_p \cup N_j; \) (* Assign the new net to the outer track. *)

if (\( t = U \)) then (* Switch street. *) \( t = B; l = l + 1; p = l; \)
else \( t = U; u = u + 1; p = u; \)

end.
Two-Layer Channel Routing Algorithms

- Left-Edge Algorithms (LEA)
  1. Basic Left-Edge Algorithm
  2. Dogleg Router
  3. Symbolic Channel Router: YACR2

- Constraint-Graph Based Routing Algorithms
  1. Net Merge Channel Router
  2. Glitter: A Gridless Channel Router

- Greedy Channel Router
- Hierarchical Channel Router
Features of Basic Left-Edge Algorithm

- Only two-terminal nets in the problem.
- No Vertical constraint in the problem.
- HV layer model is used.
- Doglegs are not allowed.
- Nets are sorted according to the x-coordinate of the leftmost terminal of the net.
- 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.
- It produces a routing solution with minimum number of tracks.
**Basic Left-Edge Algorithm**

**Algorithm** LEA \((N, I)\)

begin

FORM-INTERVAL\((N, I)\);
FORM-HCG\((I, HCG)\);
\(d = DENSITY(HCG)\);

let \(T = \{T_1 T_2, \ldots, T_d\}\) denote the set of routing tracks from top to bottom;

SORT-INTERVAL\((I)\);

for \(i = 1\) to \(n\) do

  for \(j = 1\) to \(d\) do

    if DOES-NOT-OVERLAP\((I_i, T_j)\) then

      assign interval \(I_i\) to \(T_j\);

  for \(i = 1\) to \(n\) do

    (* connect the vertical segments of net \(N_i\) to its *)

    (* horizontal segment *)

    VERTICAL-SEGMENT\(left(I_i), left(N_i)\);

    VERTICAL-SEGMENT\(right(I_i), right(N_i)\);

dend.
Detailed Routing

Example of LEA

(a)
**Detailed Routing**

**Dogleg Router**

- Drawback of LEA: the entire net is on a single track.
- Doglegs are used to place parts of a net on different tracks, thereby minimizing channel height.

![Diagram of dogleg router](image)

Using a dogleg to reduce channel height
**Dogleg Router**

- Each Multi-terminal net is broken into a set of two-terminal nets.
- Two parameters are used to control routing:
  1. range: Determine the number of consecutive two-terminal subnets of the same net that can be placed on the same track.
  2. routing sequence: Specifies the starting position and the direction of routing along the channel.
- Modified LEA is applied to each subnet.

![Diagram](image)

**Example of Dogleg Router**
Vertical Constraint Violations

- Drawback of LEA: Vertical constraints are not allowed. Vertical constraints may be violated if LEA is used.
- Vertical constraint violations are localized problem and may be resolved by
  1. local rip-up and reroute
  2. localized maze routing
Features of YACR2

- YACR2 uses two steps:
  1. Horizontal segments are assigned to tracks such that $vof(c_i)$ is minimized for each column $c_i$.
  2. For each column $c_i$:
     - (a) If $vof(c_i) = 0$, appropriate vertical segments are placed in $c_i$;
     - (b) If $vof(c_i) > 0$: localized maze routing techniques (Maze1, Maze2) are used to resolve the vertical constraint violations in $c_i$.

\[ vof(c_i) = q-p+1 = 4-1+1 = 4 \]
**Maze1 Routing of YACR2**

- **Via**
- Metal 1 (vertical segments)
- Metal 2 (horizontal segments)
- Area where metal 1 passes horizontally under metal 2

**Detailed Routing**

**Metal 1 (vertical segments)**

**Metal 2 (horizontal segments)**

**Area where metal 1 passes horizontally under metal 2**
Detailed Routing

Maze2 Routing of YACR2

- **Via**
- **Metal 1 (vertical segments)**
- **Metal 2 (horizontal segments)**
**Detailed Routing**

**Net Merge Channel Router (YK Algorithm)**

- YK algorithm considers both HCG and VCG.
- Nets are assigned to minimize the effect of vertical constraint chains in VCG.
- Does not allow doglegs and cannot handle vertical constraint cycles.
- Algorithm consists of two major steps:
  1. Zone representation of Horizontal segments
  2. Merging of Nets
- These two steps are carried out so at minimize vertical constraints and track assignment.

Steps of YK Algorithm

1. 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.

2. Merging of Nets:
   Two nets, $N_i$ and $N_j$ will 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.

Same idea has been extended by Chen and Liu for three layers.

Net Merge Channel Router

Effect of net merging on channel height
Example of Merging of Nets
Algorithm for Merging Nets

Algorithm NET-MERGE
begin
    $L = \emptyset$;
    for $z = (z_1 \text{ to } z_{t-1})$ do
        $L = L + \{z_i - (z_i \cap z_{i+1})\}$;
        $R = \{z_{i+1} - (z_i \cap z_{i+1})\}$;
        $L' = \text{MERGE}(L, R)$;
        $L = L - L'$;
    end.

Function MERGE merges two list $L$ and $R$ so as to minimize the increase in the longest path length in VCG.
Illustration of Algorithm NetMerge

(a)  
(b)  
(c)  
(d)  
(e)
Glitter: A Gridless Channel Router

- First gridless, variable width channel router.
- Supports multiple layer technology and design rules.
- Terminals can be located at arbitrary positions
- No columns are track used for routing
- Nets are allowed to have different widths to satisfy design needs.
- It is a reserved-layer model routing algorithm.

Features of Glitter

- Construct a weighted constraint graph using HCG & VCG.
- The weight of the edge between two nodes is the minimum vertical distance required between the two corresponding nets.
- The idea is to assign directions to each undirected edge such that:
  1. No cycles are generated
  2. Total weight of maximum weighted directed path is minimized.
- It can be extended to accommodate irregularly-shaped channels.
Glitter: A Gridless Channel Router

![Diagram of Glitter: A Gridless Channel Router]
Detailed Routing

Greedy Channel Router

- Starting from the leftmost column, place all net segments column by column.
- Connect any terminal to the trunk segment of corresponding net.
- Collapse any split nets using a vertical segment.
- Try to reduce the range or distance between two tracks of same net.
- Try to move the nets closer to the boundary which contains the next terminal of that net.
- Add additional tracks if no tracks are available

Examples of Greedy Channel Router

(a) A split net (b) The collapsed split net

Reducing the distance between split nets
Greedy Channel Router

Algorithm GREDY-CHANNEL-ROUTER ($\mathcal{N}$)
begin
  $d = DENSITY(\mathcal{N}$);
  (* calculate the lower bound of channel density *)
  insert $d$ tracks to channel;
  for $i = 1$ to $m$ do
    $T1 = GET-EMPTY-TRACK$;
    if $T1 = 0$ then
      ADD-TRACK($T1$);
      ADD-TRACK($T2$);
    else
      $T2 = GET-EMPTY-TRACK$;
      if $T2 = 0$ then
        ADD-TRACK($T2$);
        CONNECT($T_i$, $T1$);
        CONNECT($B_i$, $T2$);
Greedy Channel Router

join split nets as much as possible;
bring split nets closer by jogging;
bring nets closer to either top or bottom boundary;

while split nets exists do
increase number of column by 1;
join split nets as much as possible;

end.

Channel routed using a greedy router
Hierarchical Channel Router

- 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.
- Nets are routed one at a time in the $2 \times n$ grid.
- Each $2 \times n$, grid is partitioned into a $2 \times n$ grid.
- Terminal positions for the new $2 \times n$ are defined by the routing in previous hierarchy.

Example of Hierarchical channel router

Reducing \((m \times n)\) grid to \((2 \times n)\) grid

First and second levels of hierarchy
Recursively Computing $T^i(6)$ from $T^i(5), i = 1, 2, 3, 4$
## Comparison of Two-Layer Channel Routers

<table>
<thead>
<tr>
<th>Features</th>
<th>LEA</th>
<th>Dogleg</th>
<th>Y-K</th>
<th>Greedy</th>
<th>YACR2</th>
<th>Hierarchical</th>
<th>Glitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>grid-based</td>
<td>grid-based</td>
<td>grid-based</td>
<td>grid-based</td>
<td>grid-based</td>
<td>grid-based</td>
<td>grid-less</td>
</tr>
<tr>
<td>Dogleg</td>
<td>not allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>not allowed</td>
</tr>
<tr>
<td>Layer assignment</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved*</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>vertical constraints</td>
<td>not allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>allowed</td>
</tr>
<tr>
<td>cyclic constraints</td>
<td>not allowed</td>
<td>not allowed</td>
<td>not allowed</td>
<td>allowed</td>
<td>allowed</td>
<td>not allowed</td>
<td>allowed</td>
</tr>
</tbody>
</table>
### Comparison of Two-Layer Channel Routers

<table>
<thead>
<tr>
<th>ROUTER</th>
<th>Tracks</th>
<th>Vias</th>
<th>Wire length</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEA</td>
<td>31</td>
<td>290</td>
<td>6526</td>
</tr>
<tr>
<td>Dogleg router</td>
<td>21</td>
<td>346</td>
<td>5331</td>
</tr>
<tr>
<td>Y-K router</td>
<td>20</td>
<td>403</td>
<td>5381</td>
</tr>
<tr>
<td>Greedy router</td>
<td>20</td>
<td>329</td>
<td>5078</td>
</tr>
<tr>
<td>Hierarchical router</td>
<td>19</td>
<td>336</td>
<td>5023</td>
</tr>
<tr>
<td>YACR2</td>
<td>19</td>
<td>287</td>
<td>5020</td>
</tr>
</tbody>
</table>
Three-Layer Channel Routing Algorithms

- Extended Net Merge Channel Router
- HVH Routing from HV Solution
- Hybrid HVH-VHV Router
Detailed Routing

HVH Routing from HV Solution

• Similar to YK algorithm.
• Composite nets in YK algorithm are combined to form supercomposite nets.
• Objective is to reduce the number of supercomposite nets.
• Two composite nets in a supercomposite net can be assigned to different layers on the same track.
• A track ordering graph is used.

An example of HVH Routing from HV Solution

(a) 

(b) 

(c) 

<table>
<thead>
<tr>
<th>Time</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( t_1 )</td>
<td>( t_2 )</td>
</tr>
<tr>
<td>2</td>
<td>( t_3 )</td>
<td>( t_4 )</td>
</tr>
<tr>
<td>3</td>
<td>( t_5 )</td>
<td></td>
</tr>
</tbody>
</table>

(d)
Limitations of HVH or VHV Router

Partitioning for hybrid routing
Hybrid HVH-VHV router

- Uses both HVH and VHV routing schemes.
- Pure HVH and VHV are special cases of Hybrid router.
- It partitions the channel into two portions - not necessarily of the same size.
- One portion is used for HVH and the other for VHV.
- One track is required for interconnection between the two portions.

Example of Hybrid Routing

(a) Metal 1
(b) Metal 2
(c) Metal 3

Track 1
Track 2
Track 3
Track 4
Transition Track (3)
HVH
VHV
Metal 3
Metal 2
Metal 1
Switchbox Routing Algorithms

- Greedy Router
- Rip-up and Re-route Based Router: MIGHTY
- Computational Geometry Based Router: BEAVER
**Detailed Routing**

**Greedy Switchbox Router**

- Terminals on left boundary are brought to first column as horizontal tracks.
- Nets are jogged to target rows in addition to jogging to next top or bottom terminal.
- Nets are jogged based on the following priority scheme:
  1. A net whose right side of the target row and vertical track between the net and target row is empty.
  2. A net whose right side of the target row is empty and priority is based on how close a net can be jogged to target row.
  3. A net that can be brought closer to its target row.

Routing Schemes

Routing schemes: (a) scheme 1, (b) scheme 2, (c) scheme 3

Routing scheme 4 (a) $JOG_r$ for net $N_i$, (b) $JOG_{t/b}$ for net $N_i$
### Algorithm GREEDY-SB-ROUTER

**Algorithm GREEDY-SB-ROUTER**

**begin**

Determine the scan direction;
Bring left terminals into column 1;
**for** i = 1 to M **do**
  **if** no empty track exists **then**
    increase number of tracks;
  bring \( T(i) \) and \( B(i) \) to empty tracks;
  join split nets as much as possible;
  **for** each net with no right terminals **do**
    bring split nets closer by jogging;
  **for** each net with a right terminal **do**
    use scheme 4;
  **if** close to right edge **then**
    fanout to all target rows;
  **while** split net exist **do**
    join split nets as much as possible;

**end.**
Algorithm MIGHTY

Algorithm MIGHTY
begin
1. Extend all pins on the boundaries of the region inside by one unit;
   \( L \rightarrow \phi; \) (* Initialize list *)
2. (* path finder *)
   for each net do
     MAZE-ROUTE( net, L );
3. sort \( L \) in increasing value of costs;
4. while \( L \neq \phi \) do
   Get next path \( p \) from \( L \);
   if no grid cell in \( p \) is occupied then
     Implement \( p \); goto step 5;
   else invoke the path-finder to find a new feasible
     minimum path connecting two unconnected
     subnets of the net;
     Let \( \delta \) be the increase in cost for the new path \( p' \);
**Algorithm MIGHTY**

if $\delta < MAXINCREASE$ then  
Implement $p'$; goto step 5;  
(* weak modification *)  
Push implemented nets around to  
obtain a ‘good’ connection for the given net;  
if weak modification fails then  
(* strong modification *)  
Remove an existing connection and  
try to obtain ‘good’ connection;  

5. Remove $p$ from $L$;  
end.
Detailed Routing

**Computational Geometry based router**

- Based on a delayed layering scheme with computational geometry techniques.
- Main objectives are via and wire length minimization.
- It's an unreserved layer model routing algorithm.
- Uses a priority queue to determine the order in which nets are interconnected.
- Uses 3 methods to find interconnections for nets:
  1. Corner router
  2. Line sweep router
  3. Thread router

Algorithm BEAVER

Algorithm BEAVER
begin
    Initialize control information;
    Initialize corner-priority-queue;
corner route;
    if there are nets to be routed then
        Initialize line-sweep-priority-queue;
        Line sweep route;
    if there are nets to be routed then
        Relax control constraints;
        Reinitialize line-sweep-priority-queue;
        Line sweep route;
    if there are nets to be routed then
        Initialize thread-priority-queue;
        Thread route;
    Perform layer assignment;
end.
Example of Overlap Cycles and Their Solutions

Prototype Linesweep Connections
Summary

1. The detailed routing problem is solved by routing the channels and switchboxes.
2. Routing results may differ based on the selection of routing models. A routing model can be grid-based or based on the layer assignments of different net segments.
3. The objectives for routing a channel is to minimize channel density, the length of routing nets, and the number of vias.
4. The main objective of (channel) routing is to minimize the total routing area.
5. The objective of switchbox routing is to determine the routability.