Unit 8: Channel, Clock, and Power/Ground Routing

- Course contents
  - Channel routing
  - Clock routing
  - Power/ground routing

- Readings
  - Chapters 9.3 and 9.4

Order of Routing Regions and L-Channels

(a) No conflicts in case of routing in the order of 1, 2, and 3.
(b) No ordering is possible to avoid conflicts.
(c) The situation of (b) can be resolved by using L-channels.
(d) An L-channel can be decomposed into a channel and a switchbox.
Routing Considerations

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

Routing Models

- **Grid-based model:**
  - A grid is super-imposed on the routing region.
  - Wires follow paths along the grid lines.
  - Pitch: distance between two grid lines.
- **Gridless model:**
  - 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).
  - Two-layer: HV (horizontal-Vertical), VH
  - Three-layer: HVH, VHV

3 types of 3-layer models

Terminology for Channel Routing

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

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

Horizontal Constraint Graph (HCG)

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

A routing problem and its HCG.
**Vertical Constraint Graph (VCG)**

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

---

**2-L Channel Routing: Basic Left-Edge Algorithm**

- **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\((U, \text{track}(j))\)

1. \(U\): set of unassigned intervals (nets) \(I_1, \ldots, I_n\);
2. \(I_j = [s_j, e_j]\): interval \(j\) with left-end \(x\)-coordinate \(s_j\) and right-end \(e_j\);
3. \(\text{track}(j)\): track to which net \(j\) is assigned.

begin
\(U \leftarrow \{I_1, I_2, \ldots, I_n\}\);
\(t \leftarrow 0\);
while \((U \neq \emptyset)\) do
\(t \leftarrow t + 1\);
watermark \(\leftarrow 0\);
while (there is an \(I_j \in U\) s.t. \(s_j > \text{watermark}\)) do
Pick the interval \(I_j \in U\) with \(s_j > \text{watermark}\), nearest \(\text{watermark}\);
\(\text{track}(j) \leftarrow t\);
watermark \(\leftarrow e_j\);
\(U \leftarrow U - \{I_j\}\);
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 \(I_4\)

![Diagram of Basic Left-Edge Example](image)
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.

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

1 begin
2 U ← { I_1, I_2, ..., I_n};
3 t ← 0;
4 while (U ≠ ∅) do
5     t ← t + 1;
6     watermark ← 0;
7     while (there is an unconstrained I_j ∈ U s.t. s_j > watermark) do
8         Pick the interval I_j ∈ U that is unconstrained,
         with s_j > watermark, nearest watermark;
9         track[j] ← t;
10        watermark ← e_j;
11    U ← U - {I_j};
12 end
```
Constrained Left-Edge Example

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

Dogleg Channel Router

- Drawback of Left-Edge: cannot handle the cases with constraint cycles.
  - Doglegs are used to resolve constraint cycle.

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

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

Restricted vs. Unrestricted Doglegging

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

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

Robust Channel Router

- Select the set of nets for the current side by solving the maximum weighted independent set problem for interval graphs.
  - NP-complete for general graphs, but can be solved efficiently for interval graphs using dynamic programming.
- Main ideas:
  - The interval for net $i$ is denoted by $[x_{i_{\text{min}}}, x_{i_{\text{max}}}]$; its weight is $w_i$.
  - Process channel from left to right column; the optimal cost for position $c$ is denoted by total[$c$];
  - A net $n$ with a rightmost terminal at position $c$ is taken into the solution if total[$c - 1$] < $w_n$ + total[$x_{n_{\text{min}}} - 1$].
- Can apply maze routers to fix local congestion or to post-process the results. (Why not apply maze routers to channel routing directly??)
Interval Graphs

- There is a vertex for each interval.
- Vertices corresponding to overlapping intervals are connected by an edge.
- Solving the track assignment problem is equivalent to finding a **minimal vertex coloring** of the graph.

Weight Computation

1. favor nets that contribute to the channel density: add a large $B$ to $w_i$.
2. favor nets with current side terminals at column $x$: add $d(x)$ to $w_i$.
3. penalize nets whose routing at the current side would cause vertical constraint violations: subtract $Kd(x)$ from $w_i$, $K = 5 \sim 10$.

- Assume $B = 1000$ and $K = 5$ in the 1st iteration (top side):
  - $w_i = (0) + (1) + (-5 \times 2) = -9$
  - Net 1 does not contribute to the channel density
  - One net 1 terminal on the top
  - Routing net 1 causes a vertical constraint from net 2 at column 2 whose density is 2.

- $d(1) = 1$
- $d(2) = 2$
- $d(3) = 2$
- $d(4) = 3$ (nets 2, 3, 4)
- $d(5) = 2$
Weight Computation (cont’d)

Computation of the weight $w_i$ for net $i$:
1. favor nets that contribute to the channel density: add a large $B$ to $w_i$.
2. favor nets with current side terminals at column $x$: add $d(x)$ to $w_i$.
3. penalize nets whose routing at the current side would cause vertical constraint violations: subtract $Kd(x)$ from $w_i$, $K = 5 \sim 10$.

Assume $B = 1000$ and $K = 5$ in the 1st iteration (top side):
- $w_1 = (0) + (1) + (-5 \times 2) = -9$
- $w_2 = (1000) + (2) + (-5 \times 3) = 987$
- $w_3 = (1000) + (2+2) + (0) = 1004$
- $w_4 = (1000) + (3) + (-5 \times 2) = 993$

Top-Row Net Selection

A net $n$ with a rightmost terminal at position $c$ is taken into the solution if: $\text{total}[c-1] < w_n + \text{total}[x_{\text{min}}] - 1$.

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

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

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

  \[
  \begin{array}{|c|c|}
  \hline
  \text{total}[1] & 0 & \text{selected}_{\text{net}}[1] = 0 \\
  \text{total}[2] & \max(0, 0+1002) = 1002 & \text{selected}_{\text{net}}[2] = 1 \\
  \text{total}[3] & 1002 & \text{selected}_{\text{net}}[3] = 0 \\
  \text{total}[4] & \max(1002, 0+992) = 1002 & \text{selected}_{\text{net}}[4] = 0 \\
  \text{total}[5] & \max(1002, 1002+991) = 1993 & \text{selected}_{\text{net}}[5] = 4 \\
  \hline
  \end{array}
  \]

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

### Maze Routing + Rip-up & Re-route

- **3rd iteration**
  - Routing net 2 in the middle row leads to an infeasible solution.
  - Apply maze routing and rip-up and re-route nets 2 and 4 to fix the solution.
Robust Channel Router

```c
struct channel {
    int width;
    int length;
};

struct router {
    int x;
    int y;
    int z;
};

void route(int N, int M) {
    int i, j;
    for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
            if (route(i, j)) {
                printf("Route found: (%d, %d)!
", i, j);
            }
        }
    }
}
```

The Clock Routing Problem (CRP)

- **Digital systems**
  - **Synchronous systems**: Highly precised clock achieves communication and timing.
  - **Asynchronous systems**: Handshake protocol achieves the timing requirements of the system.

- **Clock skew** is defined as the difference in the minimum and the maximum arrival time of the clock.

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

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

- Pathlength-based approaches

- RC-delay based approaches:
  1. Exact zero skew: Tasy, ICCAD-91.
  2. Lagrangian relaxation: Chen, Chang, Wong, DAC-96.

---

**H-Tree Based Algorithm**

The Geometric Matching Algorithm

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

![Diagram of clock matching algorithm](image)

Elmore Delay: Nonlinear Delay Model

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

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

- Delay grows as \textbf{square} of wire length.
Wire Models

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

\[ \begin{align*}
\pi \text{-model: If no capacitive loads for } C \text{ and } D, \\
A \text{ to } B: & \quad \delta_{AB} = r_1 \left( \frac{c_1}{2} + c_2 + c_3 \right) \\
B \text{ to } C: & \quad \delta_{BC} = r_2 \left( \frac{c_2}{2} \right) \\
B \text{ to } D: & \quad \delta_{BD} = r_3 \left( \frac{c_3}{2} \right).
\end{align*} \]

Example Elmore Delay Computation

- 0.18 \( \mu \)m technology.: unit resistance \( \tilde{r} = 0.075 \Omega/ \mu \)m; unit capacitance \( \tilde{c} = 0.118 \, \text{fF}/\mu \)m.
  - Assume \( C_C = 2 \, \text{fF} \), \( C_D = 4 \, \text{fF} \).
  - \( \delta_{BC} = r_{BC} \left( \frac{c_{BC}}{2} + C_C \right) = 0.075 \times 150 \left( 17.7/2 + 2 \right) = 120 \, \text{fs} \)
  - \( \delta_{BD} = r_{BD} \left( \frac{c_{BD}}{2} + C_D \right) = 0.075 \times 200 \left( 23.6/2 + 4 \right) = 240 \, \text{fs} \)
  - \( \delta_{AB} = r_{AB} \left( \frac{c_{AB}}{2} + C_B \right) = 0.075 \times 100 \left( 11.8/2 + 17.7 + 2 + 23.6 + 4 \right) = 400 \, \text{fs} \)
  - Critical path delay: \( \delta_{AB} + \delta_{BD} = 640 \, \text{fs} \).
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 $t_{ij} = \sum_{j \in \delta(i, j)} r_j C_j$.

$t_{03} = r_0 (c_1 + c_2 + c_3 + c_4 + c_5^s + c_6^s) + r_2 (c_2/2 + c_3 + c_4 + c_6^s + c_3^s) + r_4 (c_4/2 + c_3^s)$.

Exact Zero Skew Algorithm


To ensure the delay from the tapping point to leaf nodes of subtrees $T_1$ and $T_2$ being equal, it requires that $r_1 (c_1 + c_2) + t_1 = r_2 (c_2/2 + c_3) + t_2$.

Solving the above equation, we have $z = \frac{(t_2 - t_1) + \alpha \left( c_2 + \frac{\beta l}{2} \right)}{\alpha l (c_1 + c_2 + \frac{\beta l}{2})}$, where $\alpha$ and $\beta$ 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** \( \dot{\alpha} = \frac{(t_2 - t_1) + \alpha (c_2 + \alpha/2)}{\alpha (C_1 + C_2)}, \) \( \dot{\beta} : \) per unit values of resistance (capacitance); \( l: \) length of the wire;

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

- If \( x \not\in [0, 1], \) we need **snaking** to find the tapping point.

- **Exp:** \( \alpha = 0.1 \Omega/\text{unit}, \beta = 0.2 \text{ F/\text{unit}.} \) (Find tapping points \( E \) for \( A \) and \( B, \) \( F \) for \( C \) and \( D, \) and \( G \) for \( E \) and \( F. \))

Delay Computation for Buffered Wires

- **Wire:** \( \alpha = 0.068 \Omega/\mu\text{m}, \beta = 0.118 \text{ fF/\mu\text{m}^2}; \) **buffer:** \( \alpha' = 180 \Omega/\text{unit size}, \beta' = 23.4 \text{ fF/unit size}; \) driver resistance \( R_d = 180 \Omega; \) unit-sized wire, buffer.
IR (Voltage) Drop

- Power consumption and rail parasitics cause actual supply voltage to be lower than ideal
  - Metal width tends to decrease with length increasing in nanometer design
- Effects of IR drop
  - Reducing voltage supply reduces circuit speed (5% IR drop => 15% delay increase)
  - Reduced noise margin may cause functional failures

Power/Ground (P/G) Routing

- 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 (IR) drop small, widen wires for more power-consuming modules and higher density current (1.5 mA per \( \mu \)m width for Al). (So area metric?)
P/G Network Design Process

- Construct an initial P/G network
  - Consider the voltage level at each node from left to right with $V_{S1}$ (and then from right to left with $V_{S2}$)
  - Partition the nodes with the voltage levels greater than $V_{min}/2$ into two sets $A$ and $B$.
  - If $A \neq B = c$, add a power strap in the middle.

P/G Network Optimization

- If the voltages of adjacent nodes are greater than $V_{min}$, decrease the width of the corresponding power strap or even remove the strap.