Unit 8: Special Net Routing & Performance Optimization

- **Course contents:**
  - Clock net routing
  - Power/ground routing
  - Performance optimization

- **Readings**
  - W&C&C: Chapter 13
  - S&Y: Chapter 7

---

The Clock Routing Problem

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

- **Clock skew:** the difference in the minimum and the maximum arrival times of the clock.

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

• 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 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.

Clock Routing Algorithms

• Pathlength-based Clock-Tree Synthesis (CTS)
  1. \( H \)-tree: Dhar, Franklin, Wang, ICCD-84; Fisher & Kung, 1982.

• RC-delay based CTS
  1. Exact zero skew: Tsay, ICCAD-91.
  2. Deferred-merge embedding (DME) algorithm: Boese & Kahng, ASICON-92; Chao & Hsu & Ho, DAC-92; Edahiro, NEC R&D, 1991.

• Simulation-based CTS
  - ISPD-09 CTS contest (ASP-DAC-10, DATE-10)

• Timing-model independent CTS
  - Shih & Chang, DAC-10; Shih et al., ICCAD-10.

• Mesh-based & tree-link-based clock routing
**H-Tree Based Algorithm**


![H-tree diagram]

*H*-tree over 4 points

*H*-tree over 16 points

*Similar topology: X-tree*

**The MMM Algorithm**

- Each block pin is represented as a point in the region, S.
- The region is partitioned into two subregions, $S_L$ and $S_R$.
- The center of mass is computed for each subregion.
- The center of mass of the region $S$ is connected to each of the centers of mass of subregion $S_L$ and $S_R$.
- 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

- 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: \( O(n^2 \log n) \).

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 \( \times \) downstream capacitance.

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

- 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: \( \pi \)-model (most popular), \( T \)-model, \( L \)-model.

- \( \pi \)-model: If no capacitive loads for \( C \) and \( D \),
  - \( A \) to \( B \): \( \delta_{AB} = r_1 \left( \frac{c_1}{2} + c_2 + c_3 \right) \);
  - \( B \) to \( C \): \( \delta_{BC} = r_2 \left( \frac{c_2}{2} \right) \);
  - \( B \) to \( D \): \( \delta_{BD} = r_3 \left( \frac{c_3}{2} \right) \).

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: \( \delta_{AB} + \delta_{BD} = 640 \text{fs} \).
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
  \[ r_1 \left( \frac{c_1}{2} + C_1 \right) + t_1 = r_2 \left( \frac{c_2}{2} + C_2 \right) + t_2 \]
- Solving the above equation, we have
  \[ x = \frac{(t_2 - t_1) + \alpha \left( C_2 + \frac{\beta l}{2} \right)}{\alpha (\beta l + C_1 + C_2)} \]

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**: $r_1 \left( \frac{c_1}{2} + C_1 \right) + t_1 = r_2 \left( \frac{c_2}{2} + C_2 \right) + t_2$.
- **Compute tapping points**: $x = \frac{(t_2 - t_1) + \alpha \left( C_2 + \frac{\beta l}{2} \right)}{\alpha (\beta l + C_1 + C_2)}$; $\alpha$ ($\beta$): 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 $x \not\in [0, 1]$, we need **snaking** to find the tapping point.
- **Exp**: $\alpha = 0.1 \, \Omega \text{ /unit}$, $\beta = 0.2 \, F \text{ /unit}$ (tapping points: $E$, $F$, $G$)
Deferred Merge Embedding (DME)

- Boese & Kahng, ASICON-92; Chao & Hsu & Ho, DAC-92; Edahiro, NEC R&D, 1991
- Consists of two stages: bottom-up + top-down
- Bottom-up: Build the potential embedding locations of clock sinks (i.e., a segment for potential tapping points)
- Top-down: Determine exact locations for the embedding

Delay Computation for Buffered Wires

- Wire: $\alpha = 0.068 \ \Omega / \mu m, \ \beta = 0.118 \ fF/ \mu m^2$; buffer: $\alpha' = 180 \ \Omega / \text{unit size}, \ \beta' = 23.4 \ fF/\text{unit size};$ driver resistance $R_d = 180 \ \Omega$; unit-sized wire, buffer.
Buffering and Wire Sizing for Skew Minimization

- Discrete wire/buffer sizes: dynamic programming
- Continuous wire/buffer sizes: mathematical programming (e.g., Lagrangian relaxation)
  - Considers clock skew, area, delay, power, clock-skew sensitivity simultaneously.

Clock Meshes

- More alternative paths to clock sinks
  - Good for high-performance circuits with stringent skew and variation constraints
- Drive mesh from the boundary or from grid points
- H-tree is a good candidate to drive mesh
Power Integrity: 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 mA / \( \mu m^2 \) at 25 °C for 0.18 \( \mu m \) technology). (So area metric?)
Power/Ground Network Optimization

- Use the minimum amount of chip area for wiring P/G networks while avoiding potential reliability failures due to electromigration and excessive IR drops.
  - Build the equivalent models for series resistors and apply a sequence of the linear programming (SLP) method to solve the problem.
  - Size wire segments assuming the topologies of P/G networks to be fixed.
- Wu and Chang, “Efficient power/ground network analysis for power integrity driven design methodology,” DAC-2004.
- Liu and Chang, “Floorplan and power/ground co-synthesis for fast design convergence,” ISPD-06 (TCAD-07).

Problem Formulation

- Let $G = (N, B)$ be a P/G network with $n$ nodes $N = \{1, \ldots, n\}$ and $b$ branches $B = \{1, \ldots, b\}$; branch $i$ connects two nodes: $i_1$ and $i_2$ with current flowing from $i_1$ to $i_2$.
- Let $l_i$ and $w_i$ be the length and width of branch $i$, respectively. Let $\rho$ be the sheet resistivity. Then the resistance $r_i$ of branch $i$ is $r_i = \frac{V_{i_2} - V_{i_1}}{I} = \rho \frac{l_i}{w_i}$.
- Total P/G routing area is as follows:

$$f(V, I) = \sum_{i \in B} I_i w_i = \sum_{i \in B} \frac{\rho l_i^2}{V_{i_1} - V_{i_2}}.$$

- P/G network optimization is to minimize $f(V, I)$ subject to the constraints listed in the next slide.
- Relax the nonlinear objective function and then translate the constrained nonlinear programming problem into a SLP problem.
Constraints

- The voltage IR drop constraints.
  - $V_i \geq V_{i\_min}$ for power networks.
  - $V_i \leq V_{i\_max}$ for ground networks.
- The minimum width constraints: $w_i = \rho \frac{I_i}{V_{i\_in} - V_{i\_out}} \geq w_{i\_min}$
- The electro-migration constraints: $I_i/w_i \leq \sigma \implies |V_{i\_in} - V_{i\_out}| \leq \rho \sigma$
  - $\sigma$ is a constant for a particular routing layer with a fixed thickness.
- Equal width constraints: $w_i = w_j$ or $\frac{V_{i\_in} - V_{i\_out}}{I_i} = \frac{V_{j\_in} - V_{j\_out}}{I_j}$
- Kirchoff’s current law (KCL): $\sum_{i(B(j))} l_i = 0$
  - For each node $j = \{1, \ldots, n\}$, $B(j)$ is the set of indices of branches connecting to node $j$.

Reducing the Problem Size with Equivalent Circuits

- Consider a series resistor chain commonly seen in the P/G network below.
- The equivalent resistor $R_e$ is just the sum of all the resistors in series: $R_e = \sum_{i=1}^{n} R_i$.
- By superposition, the equivalent currents $I_{e\_1}$ and $I_{e\_n}$ can be computed as follows:
  $$I_{e\_1} = \sum_{i=1}^{n-2} \frac{\sum_{j=i+1}^{n-1} R_j}{R_i} I_{e\_1}$$
  $$I_{e\_n} = \sum_{i=1}^{n-2} \frac{\sum_{j=i+1}^{n-1} R_j}{R_i} I_{e\_i}$$
The voltages at the intermediate nodes are calculated based on superposition as follows:

\[ V_{i+1} = V_i - \frac{R_i}{R_s} V_s - R_i I_{i+1} \]

\[ I_{i+1} = I_i - I_s \]
Design Methodology Evolution

- IR-drop aware design methodology for faster design convergence

- IR-drop Analysis
  - Floorplanning
  - P&R
  - RC Extraction
  - Simulation
  - SI Analysis

- SI Analysis
  - yes
  - no

- OK

- iterative loop

Traditional flow

DAC-04 flow
(Wu & Chang)

ISPD-06 (TCAD-07) flow
(Liu & Chang)

Ideal Scaling of MOS Transistors

- Feature size scales down by $S$ times:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scaling factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions ($W, L, t_{ox}$, junction depth $X_j$)</td>
<td>$1/S$</td>
</tr>
<tr>
<td>Area per device ($A = WL$)</td>
<td>$1/S^2$</td>
</tr>
<tr>
<td>Substrate doping ($N_{SUB}$)</td>
<td>$S$</td>
</tr>
<tr>
<td>Voltages ($V_{DD}, V_t$)</td>
<td>$1/S$</td>
</tr>
<tr>
<td>Current per device ($I_{ds} \propto \frac{W_{eq} L}{L_{eq}}$)</td>
<td>$1/S$</td>
</tr>
<tr>
<td>Gate capacitance ($C_g = \frac{C_{ox}WL}{L_{ox}}$)</td>
<td>$1/S$</td>
</tr>
<tr>
<td>Transistor on-resistance ($R_o \propto \frac{V_{DD}}{I_{ds}}$)</td>
<td>$1$</td>
</tr>
</tbody>
</table>

- Intrinsic gate delay ($\tau = R_o C_g$) $1/S$
- Power dissipation per gate ($P \propto IV$) $1/S^2$
- Power-dissipation density ($P/A$) $1$
### Ideal Scaling of Interconnections

- Feature size scales down by $S$ times:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scaling factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross sectional dimensions ($W_s, H_s, t_{ox}$)</td>
<td>1/2</td>
</tr>
<tr>
<td>Resistance per unit length ($r_0 = \rho / W_s H_s$)</td>
<td>$S^2$</td>
</tr>
<tr>
<td>Capacitance per unit length ($C_0 = W_s t_{ox} / t_{ox}$)</td>
<td>1/2</td>
</tr>
<tr>
<td>RC constant per unit length ($r_0 C_0$)</td>
<td>$S^2$</td>
</tr>
<tr>
<td>Local interconnection length ($l_i$)</td>
<td>1/2</td>
</tr>
<tr>
<td>Local interconnection RC delay ($r_0 C_0 l_i^2$)</td>
<td>$S^2$</td>
</tr>
<tr>
<td>Die size ($D_s$)</td>
<td>$S_0$</td>
</tr>
<tr>
<td>Global interconnection length ($l_a$)</td>
<td>$S_0$</td>
</tr>
</tbody>
</table>

![Diagram showing cross sectional dimensions and scaling factors](image)

### Techniques for Higher Performance

- In very deep submicron technology, interconnect delay dominates circuit performance.
- Techniques for higher performance
  - **SOI:** lower gate delay.
  - **Copper interconnect:** lower resistance.
  - **Dielectric with lower permittivity:** lower capacitance.
  - **Buffering:** Insert (and size) buffers to “break” a long interconnection into shorter ones.
  - **Wire sizing:** Widen wires to reduce resistance (careful for capacitance increase).
  - **Shielding:** Add/order wires to reduce capacitive and inductive coupling.
  - **Spacing:** Widen wire spacing to reduce coupling.
  - **Others:** padding, track permutation, net ordering, etc.
Interconnect Dominates Circuit Performance!!

Worst-case interconnect delay due to crosstalk

Interconnect delay
Gate delay

Delay (ps)

650 500 350 250 150 100 70 (nm)

In ≤ 0.18μm wire-to-wire capacitance dominates ($C_W$>$C_g$)

Optimal Buffer Sizing w/o Considering Interconnects

- Delay through each stage is $\alpha t_{min}$, where $t_{min}$ is the average delay through any inverter driving an identically sized inverter.
- $\alpha^n = C_L/C_g \Rightarrow n = \ln (C_L/C_g)/\ln \alpha$, where $C_L$ is the capacitive load and $C_g$ the capacitance of the minimum size inverter.
- Total delay: $T_{tot} = n\alpha t_{min} = \frac{\alpha}{\ln \alpha} t_{min} \ln \frac{C_L}{C_g}$.
- Optimal stage ratio: $\frac{\alpha_{tot}}{\alpha_{do}} = 0 \Rightarrow \alpha = e$.
- Optimal delay: $T_{opt} = \alpha t_{min} \ln \frac{C_L}{C_g}$.
- Buffer sizes are exponentially tapered ($\alpha = e$).
Wire Sizing

- Wire length is determined by layout architecture, but we can choose wire width to minimize delay.
- Wire width can vary with distance from driver to adjust the resistance which drives downstream capacitance.
- Wire with minimum delay has an exponential taper.
- Can approximate optimal tapering with segments of a few widths.
- Recent research claims that buffering is more effective than wire sizing for optimizing delay, and two wire widths are sufficient for area/delay trade-off.

Optimal Wire-Sizing Function

- Suppose a wire of length \( L \) is partitioned into \( n \) equal-length wire segments, each of length \( \Delta x = L/n \); unit resistance and capacitance: \( \frac{R}{\Delta x} \), \( \frac{C}{\Delta x} \).
- The respective resistance and capacitance of \( i \)-th wire segment can be approximated by \( \frac{R}{\Delta x} f(x_i) \) and \( \frac{C}{\Delta x} f(x_i) \), where \( f(x) \) is the width at position \( x \).
- Elmore delay: \( D_n = R_d \left( C_L + \sum_{i=1}^{n} \frac{\Delta x}{f(x_i)} \right) + \sum_{i=1}^{n} C_L \left( \sum_{j=i}^{n} \frac{\Delta x}{f(x_j)} \right) \)
- As \( n \to \infty \), \( D_n \to D \): \( D = R_d \left( C_L + \int_0^L C_L \frac{f(\xi)}{f(x)} d\xi \right) + \int_0^L \frac{\eta}{f(x)} \left( \int_0^L \frac{f(\xi)}{f(x)} d\xi + C_L \right) d\xi \)
- Optimal wire sizing function \( f(x) = ae^{-bx} \), where
  \[
  a = \frac{\alpha}{b R_d} \left( \frac{R_d C_L}{\alpha} - e^{-b L/2} \right) = 0.
  \]
Simultaneous Wire & Buffer Sizing

- **Input:** Wire length $L$, driver resistance $R_d$, load capacitance $C_L$, unit wire area capacitance $c_0$, unit wire fringing capacitance $c_f$, unit-sized wire resistance $r_0$, unit-size capacitance of a buffer $c_b$, unit-size buffer resistance $r_b$, intrinsic buffer delay $T_{in}$, and the number of buffers $N$.

- **Objective:** Determine the stage ratio $\beta$ for buffer sizes and the stage ratio $\omega$ for wire widths such that the wire delay is minimized.

Wire/Buffer Size Ratios for Delay Optimization

- Chang, Chang, Jiang, ISQED-2002.

$$D_N(\beta, \omega) = R_d(c_0 h + c_f l) + \beta R_d c_b + N T_{in} + \beta (N - 1) r_b c_b + \frac{r_d c_b L}{h \omega N} + \frac{r_b}{\beta N}$$

$$+ \frac{1}{2}(N + 1) \frac{r_0 c_f l^2}{h} + \sum_{i=1}^{N} \beta^i (c_0 w^{i} h + c_f l) + \sum_{i=1}^{N} \omega^i \frac{r_b c_b}{w^{i} h}.$$

- In practice, the delay of a wire $D_N(\beta, \omega)$ is a convex function of the stage ratio $\beta$ for practical buffer sizes and the stage ratio $\omega$ for practical wire widths.

- Can apply efficient search techniques (e.g., binary search) to find the optimum ratios.
Performance Optimization: A Sizing Problem

- Minimize the maximum delay $D_{\text{max}}$ by changing $w_1, \ldots, w_n$

$$\begin{align*}
\text{Minimize} & \quad D_{\text{max}} \\
\text{subject to} & \quad D_i(W) \leq D_{\text{max}}, \quad i = 1..m \\
& \quad L \leq w_i \leq U, \quad i = 1..n
\end{align*}$$

Popular Sizing Works

- Algorithmic approaches: faster, non-optimal for general problems
  - TILOS (Fishburn, Dunlop, ICCAD-85)
  - Weighted Delay Optimization (Cong et al., ICCAD-95)
- Traditional mathematical programming: often slower, optimal
  - Geometric Programming (TILOS)
  - Augmented Lagrangian (Marple et al., 86)
  - Sequential Linear Programming (Sapatnekar et al.)
  - Interior Point Method (Sapatnekar et al., TCAD-93)
  - Sequential Quadratic Programming (Menezes et al., DAC-95)
  - Augmented Lagrangian + Adjoin Sensitivity (Visweswariah, et al., ICCAD-96, ICCAD-97)
- Lagrangian relaxation based mathematical programming: (Chen, Chang, Wong, DAC-96; Jiang, Chang, Jou, DAC-99 [TCAD, Sept. 2000]; and many more)
  - Fast and optimal
**TILOS: Heuristic Approach**

- Finds sensitivities associated with each gate
- Up-sizes the gate with the maximum sensitivity
- Minimizes the objective function

Minimize $D_{max}$

```
Minimize  D_{max}  \\
\begin{align*}
D_1 &< D_{max} \\
D_2 &< D_{max}
\end{align*}
```

**Weighted Delay Optimization**

- Cong, et. al., ICCAD-95
- Sizes one wire at a time in the DFS order
- Minimize the weighted delay
- Best weights?

Minimize $\lambda_1 D_1 + \lambda_2 D_2$
From Mathematical Prog. to Lagrangian Relaxation

Mathematical formulation

\[ \min cx \]
\[ \text{st} \ A x \leq b \]
\[ x \in X \]

Posynomial forms

Positive coefficient polynomials

Lagrange multipliers \( \lambda \)

\[ \min L(\lambda) = cx + \lambda(Ax - b) \]
\[ \text{st} \ x \in X \]

Mathematical Programming

- **Formulation:** Minimize \( f(x) \)
  subject to \( g_i(x) \leq 0, \ i = 1..m \)

- **Lagrangian:** \( L(\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x), \text{ where } \lambda_i \geq 0 \)

- **Optimality (Necessary) Condition (Kuhn-Tucker theorem):**
  \[ \frac{\partial L(\lambda)}{\partial x_i} = 0 \Rightarrow \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) = 0 \]
  \[ \lambda_i g_i(x) = 0 \text{ (Complementary Condition)} \]
  \[ g_i(x) \leq 0, \lambda_i \geq 0 \text{ (Feasibility Condition)} \]
Lagrangian Relaxation

\[
\begin{align*}
\text{Minimize } & \quad f(x) \\
\text{subject to } & \quad g_i(x) \leq 0, \quad i = 1..n \\
& \quad g_i(x) \leq 0, \quad i = n + 1..m
\end{align*}
\]

LRS (Lagrangian Relaxation Subproblem)

- There exist Lagrangian multipliers \( \lambda \) that lead LRS to the optimal solution for convex programming
  - When \( f(x) \), \( g(x) \)'s are all positive polynomials (posynomials)
- The optimal solution for any LRS is a lower bound of the original problem

\[\text{Minimize } f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \]

\[
\begin{align*}
\text{subject to } & \quad g_i(x) \leq 0, \quad i = n + 1..m
\end{align*}
\]

Lagrangian Relaxation

\[
\begin{align*}
\text{Minimize } & \quad D_{\max} \\
\text{subject to } & \quad D_i(W) - D_{\max} \leq 0, \quad i = 1..m \\
& \quad L \leq w_i \leq U, \quad i = 1..n
\end{align*}
\]

\[
\begin{align*}
\text{Minimize } & \quad D_{\max} + \sum_{i=1}^{m} \lambda_i (D_i(W) - D_{\max}) \\
\text{subject to } & \quad L \leq w_i \leq U, \quad i = 1..n
\end{align*}
\]

By \( \frac{\partial L_{\lambda}}{\partial D_{\max}} = 0 \), we have \( \sum_{i=1}^{m} \lambda_i = 1 \)

Minimize \( \sum_{i=1}^{m} \lambda_i D_i(W) \)

subject to \( L \leq w_i \leq U, \quad i = 1..n \)
Lagrangian Relaxation

Lagrangian Relaxation

Augmented Lagrangian

Weighted Lagrangian

TILOS

Algorithmic approaches

SLP

Mathematical Programming

Sink Weights = Multipliers

Lagrangian Relaxation Framework

Update Multipliers

Weighted Delay Optimization

Converge?

Yes
done

No
Lagrangian Relaxation Framework

More Critical -> More Resource -> Larger Weight

Weighted Minimization

- Traverse the circuit in the topological order
- Resize each component to minimize Lagrangian during visit

Minimize $\lambda_1 D_1 + \lambda_2 D_2$
Multiplier Adjustment: A Subgradient Approach

Step 1: \( \lambda_i^{new} = \lambda_i^{old} + \theta_k (D_i - D_{\max}) \),

where \( \lim_{k \to \infty} \theta_k \to 0, \sum_{k=1}^{\infty} \theta_k \to \infty \)

Step 2: Project \( \lambda \) to the nearest feasible solution

- Subgradient: An extension definition of gradient for non-smooth functions.
- Experience: Simple heuristic implementation can achieve a very good convergence rate.

Convergence Sequence

Minimize \( \sum_{i=1}^{m} \lambda_i D_i(w) \)

subject to \( L \leq w_i \leq U, i = 1..n \)

Any Feasible Maximum Delay = Upper Bound

Lagrangian = Lower Bound
Weighted Delay <= Maximum Delay

# Iterations
Path Delay Formulation

\[ A_a + d_1 + d_2 \leq D_1 \]
\[ A_b + d_1 + d_2 \leq D_1 \]
\[ A_b + d_1 + d_3 \leq D_2 \]
\[ A_c + d_3 \leq D_2 \]

- Exponential growth
- More accurate
- Can exclude false paths

Stage Delay Formulation

\[ A_a + d_1 \leq A_{e} \]
\[ A_b + d_1 \leq A_{e} \]
\[ A_e + d_2 \leq D_1 \]
\[ A_e + d_3 \leq D_2 \]
\[ A_c + d_3 \leq D_2 \]

- Polynomial size
- Less accurate
- Contains false paths
Both Multipliers Satisfy KCL (Flow Conservation)

Stage Based

Path Based

\[ \lambda_{43} + \lambda_{53} = \lambda_{31} + \lambda_{32} \]

\[ \lambda_{51} = \lambda_{3,in} = \lambda_{3, out} \]

\[ \sum_{j \text{input}(i)} \lambda_{ji} = \sum_{k \text{output}(i)} \lambda_{ik} \quad \forall i \]

Appendix A:

Shih and Chang
“Fast timing-model independent clock-tree synthesis”
DAC-10, TCAD-12
Introduction

- Skew-minimized buffered clock-tree synthesis plays an important role in VLSI designs for synchronous circuits.
- Due to the insufficient accuracy of timing models, embedding simulation into synthesis becomes inevitable.
- Runtime becomes prohibitively huge as design complexity grows.

Symmetrical Structure

- Skew is minimized by structural optimization.
- Buffering and wiring of all paths are almost the same:
  - Is timing-model independent.
  - Do not need simulation information.

0ps skew (Elmore delay)
0.123ps skew (simulation)
Problem Formulation

- **Problem**: Buffered Clock-Tree Synthesis (BCTS)

- **Instance**
  - Given a set of clock sinks, a slew-rate constraint, and a library of buffers

- **Question**
  - Construct a buffered clock tree to minimize its *skew*, subject to no slew-rate violation

Symmetrical Clock Tree Synthesis

- **Specification**
  - *Number of branches, wirelength* and *inserted buffers* are the same at each level

- **Flow**
  - Assign specific branch numbers to each tree level
  - Cluster sub-trees level by level bottom-up
  - Lengthen shorter connection by snaking
  - Insert identical buffers along trees
Branch-Number Planning

- **Observation**
  - Total branch number of some level equals the number of preceding level times its branch number
  - The multiplication sequence forms a *factorization*

\[ n = f_1 \times f_2 \times \ldots \times f_q, \quad f_i \leq f_{i+1}, \forall i < q \]

- **Planning**
  - Branch-Number Plan (BNP) is arranged in non-increasing order

\[ B(n) = \langle b_1, b_2, ..., b_m \rangle \rightarrow \langle f_q, f_{q-1}, ..., f_1 \rangle \]

\[ \text{level-1 branch number} \]

- Factorization may result in a big branch number, implying a large fan-out size that could not be driven
- **Pseudo sinks** are added to increase the total sink number until all branch numbers are feasible

\[
\begin{align*}
\text{BNP} &= B(216) = \langle 3, 3, 3, 2, 2, 2 \rangle \\
\text{BNP} &= B(212) = \langle 53, 2, 2 \rangle \\
\text{BNP} &= B(212+1) = \langle 71, 3 \rangle \\
\text{BNP} &= B(212+2) = \langle 107, 2 \rangle \\
\text{BNP} &= B(212+3) = \langle 43, 5 \rangle
\end{align*}
\]
Tree Construction

- Achieve identical wirelength in this stage
  - Cluster sub-trees level by level bottom-up
  - Lengthen shorter connection by snaking

Flow

- Partitioning
- Embedding-Region Construction
  - Divide sub-trees into desired clusters
  - Divide sub-trees into desired clusters
  - Apply a common connection length to each cluster, and locate potential embedding positions to which snaked wires can reach
  - Repeat the two stages till the embedding region of the root is built
  - Find exact physical locations for nodes and route wires top-down

Node Embedding

Tilted Rectangular Region (TRR)

- Represents potential embedding positions (embedding region)
- Is a 45- or 135-degree rectangular region
  - core: a 45- or 135-degree line segment
  - radius: the Manhattan distances from the core to the region boundaries

Configuration

Operation

Definition

TRR

Manhattan distance

Configuration

Operation

Definition

TRR
Partitioning

- The objective is to minimize cluster diameter
  - Cluster diameter: the maximum distance among sub-trees within the same cluster
  - Maximum cluster diameter is the upper bound of the common connection length
- Sub-trees are divided recursively along the BNP in a top-down manner
- Non-binary tree can also be handled by this technique

Dividing: Cake Cutting

- Borrow the idea of cake cutting, i.e., slicing a cake into pieces from the center of the cake
- Sort the polar angles of sub-trees relative to the geometric center of the cluster
- Apply dynamic programming to find the minimum cluster diameter by restricting the dividing on this sorted order
Recursive Dividing

- For \( i \)-th level partitioning along the given BNP \(<b_1, b_2, \ldots, b_q>\), dividing is performed recursively until \( b_1 \times b_2 \times \ldots \times b_{i-1} \) clusters are derived.
- Desired cluster diameter could be obtained since global sub-tree distribution is considered throughout the whole process.

<table>
<thead>
<tr>
<th>Recursive Dividing</th>
<th>Final Divided Result</th>
<th>Corresponding Clusters</th>
</tr>
</thead>
</table>

Embedding-Region Construction

- Assign the common connection length (CCL) as the half length of the maximum cluster diameter.
- Extend the TRRs of children nodes and make intersection to construct the embedding region of their parents.

<table>
<thead>
<tr>
<th>Given Divided Result</th>
<th>Region Extension/Intersection</th>
<th>Resulting Regions</th>
</tr>
</thead>
</table>
Node Embedding

- Set the tree root as the closest position of the embedding region w.r.t. the clock source
- Propagate embedding information level by level top-down
- Perform snaking to meet the uniform length, if necessary

Pseudo-Sink Handling

- For partitioning
  - Relax the sizes of clusters in a partition which can differ by at most one for the first recursion
- For embedding-region construction
  - Construct no embedding regions for pseudo sinks to reserve the flexibility of snaking
- For node embedding
  - Let the embedding regions of pseudo sinks cover entire chip
  - Dangling wires can be identified and attached to proper sub-trees successfully
Buffer Insertion

- Align buffer distribution on the symmetrical tree topology
- Insert identical buffers level by level top-down

First-Time Insertion  Second-Time Insertion  Third-Time Insertion

Experimental Results on IBM Benchmarks

- Our approach can obtain much smaller skews in much shorter runtime than the state of the art, with marginal overheads of snaking for symmetry

<table>
<thead>
<tr>
<th></th>
<th>Shih et al. [ASPDAC’10]</th>
<th>Shih et al. [ASPDAC’10]</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o simulation</td>
<td>w/ simulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>skew usage runtime</td>
<td>skew usage runtime</td>
<td>skew usage runtime</td>
</tr>
<tr>
<td></td>
<td>(ps) (fs) (s)</td>
<td>(ps) (fs) (s)</td>
<td>(ps) (fs) (s)</td>
</tr>
<tr>
<td>r1</td>
<td>267</td>
<td>14.005 14001 2</td>
<td>5.012 15229 5126</td>
</tr>
<tr>
<td>r2</td>
<td>598</td>
<td>16.012 28011 11</td>
<td>6.421 29234 7374</td>
</tr>
<tr>
<td>r3</td>
<td>862</td>
<td>16.532 39123 26</td>
<td>5.611 41431 12739</td>
</tr>
<tr>
<td>r4</td>
<td>1903</td>
<td>17.792 89312 165</td>
<td>5.418 91015 1781</td>
</tr>
<tr>
<td>r5</td>
<td>3101</td>
<td>21.557 149875 498</td>
<td>7.028 156854 26045</td>
</tr>
<tr>
<td>avg.</td>
<td>7.93</td>
<td>0.92 46.29</td>
<td>2.77 0.96 24343.13</td>
</tr>
</tbody>
</table>

More than 80000X faster than the ISPD-09 contest winners (simulation-based methods)
Resulting Clock Tree: *ispd09f22*

Appendix B:

Liu and Chang

“Floorplan and power/ground network co-synthesis for fast design convergence”

ISPD-06 (TCAD-07)
**Floorplan & P/G Network Co-Synthesis**

- Liu and Chang, “Floorplan and power/ground network co-synthesis for fast design convergence,” ISPD-06 (TCAD-07).
- Apply the B*-tree floorplan representation and simulated annealing (SA).
- Analyze the P/G network (typical flow)
  - Circuit modeling
  - Global P/G network construction
  - P/G network modeling/reduction
  - P/G network evaluation (IR-drop computation)
- Reduce floorplan solution space

---

**Implementation of the Design Flow**

**Data preparation**
- Power profile
  - Power consumption data of the modules generated by PrimePower
- Hierarchical circuit partition
  - Organize the design into hard modules and soft modules according to the hierarchy

**Post-layout verification**
- AstroRail
  - Static cell-level P/G analysis
Simulated Annealing Process

- Non-zero probability for up-hill climbing:
  \[ p = \min\left(1, e^{-\frac{\Delta \Psi}{T}}\right) \]

- Perturbations (neighboring solutions)
  - Op1: Rotate a block
  - Op2: Move a node/block to another place
  - Op3: Swap two nodes/blocks
  - Op4: Resize a soft block

- The cost function \( \Psi \) is based on the floorplan cost and P/G network cost

- \( T \) is decreased every \( n \) cycles, where \( n \) is proportional to the number of blocks

Cost Function

- Cost function:
  \[ \Psi = \alpha \cdot W + \beta \cdot A + \gamma \cdot \Phi + \omega \cdot \frac{A}{D_{\text{pitch}}^2} \]

- \( W \): Wirelength
- \( A \): Area
- \( \Phi \): P/G network cost (penalty of power integrity violation)
- \( D_{\text{pitch}} \): pitch of P/G network
  - Increasing power mesh density (reducing \( D_{\text{pitch}} \)) reduces \( \Phi \)
  - Update \( D_{\text{pitch}} \) by multiplying \( \hat{\Phi} / \Phi_{\text{avg}} \)
  - \( \Phi_{\text{avg}} \): Average P/G network cost at a temperature
  - \( \hat{\Phi} \): 0 < \( \hat{\Phi} < 1 \), a factor for adjusting the density of P/G networks
    - Smaller \( \hat{\Phi} \) for higher P/G density and larger one for lower P/G density
Pitch Updating: An Example

- At the beginning of SA, \( D_{\text{pitch}} = 2 \) and \( \Phi = 0.02 \)
- During SA process, \( D_{\text{pitch}} \approx \frac{\Phi}{\Phi_{\text{avg}}} \times D_{\text{pitch}} \)

\[
\frac{\Phi}{\Phi_{\text{avg}}} \text{ converges to 1 while temperature cools down}
\]

P/G Network Cost

\[ \Phi: \text{P/G network cost} \]

\[ \Phi = \theta \cdot \frac{|B_{\text{em}}|}{|B|} + (1 - \theta) \sum_{p_{vi} \in P} \frac{v_{pvi}}{V_{\text{lim,pi}}} \]

- \( B_{\text{em}} \): set of branches violating electromigration constraints
- \( B \): total branches of the P/G mesh
- \( v_{pvi} \): amount of the violation at the pin \( p_{vi} \)
- \( P \): set of all P/G pins
- \( P_v \): set of violating P/G pins
- \( V_{\text{lim,pi}} \): IR-drop constraint of the P/G pin \( p_i \)
P/G Network Construction

- For each floorplan, we construct a uniform global P/G network according to $D_{pitch}$.
- The number of trunks is defined by $\text{round}[\text{width}/D_{pitch}] + 1 \& \text{round}[\text{height}/D_{pitch}] + 1$

2X4 uniform P/G network is constructed

Floorplan

Width

Calculate the P/G network dimension

P/G Network Modeling

Apply static analysis for fast P/G network evaluation
- Use resistive P/G Model
- Model P/G pins by current sources
  - Current value: maximum current drawn from P/G pins
- Reduce circuit size
  - Connect current sources to nearest global trunk nodes
P/G Network Modeling

Apply static analysis for fast P/G network evaluation

- Use resistive P/G Model
- Model P/G pins by current sources
  - Current value: maximum current drawn from P/G pins
- Reduce circuit size
  - Connect current sources to nearest global trunk nodes

---

Macro Current Modeling

- Divide the floorplan into regions
- For hard macros
  - Connect P/G pins to the nearest global trunk nodes
- For soft macros (worst-case scenario)
  - Collect the largest current drawn by standard cells in the overlapping area of the region and the soft macro
Macro Current Modeling

- Divide the floorplan into regions
- For hard macros
  - Connect P/G pins to the nearest global trunk nodes
- For soft macros (worst-case scenario)
  - Collect the largest current drawn by standard cells in the overlapping area of the region and the soft macro

![Diagram of Macro Current Modeling]

Soft Macro Modeling

- Derive the largest current drawn by standard cells of the overlapping area
  - Maximize the current of the overlapping area
  - Constraint: total standard cell area < the overlapping area
  - The problem is known as 0-1 Knapsack Problem (NP-complete)
- Approximate it by Fractional Knapsack Algorithm
  - Assume standard cells can be broken into arbitrary smaller pieces
  - Rank cells by current to area ratio
  - Apply a greedy algorithm (complexity \(O(n \lg n)\))
Evaluation of P/G Network

- The static analysis of a P/G network is formulated as the following modified nodal analysis (MNA) formula:

\[ Gx = i \]

- \( G \): conductance matrix (sparse positive definite matrix)
- \( x \): vector of node voltages
- \( i \): vector of current loads and voltage sources
- Dimensions of \( G \), \( i \) and \( x \) are equal to the number of nodes in the P/G network

- Solve the linear equation
  - Apply Preconditioned Conjugated Gradient (PCG) method
  - The time complexity is linear

Idea of Solution Space Reduction

- The IR-drop of a P/G pin is proportional to the effective resistance between the P/G pin and the power pad
  - The closer the P/G pin is placed to the power pad, the smaller the IR-drop

- A technique to reduce solution space
  - Place the modules consuming larger current (power-hungry modules) near the boundary of the floorplan
  - Place power pads close to them
### B*-tree Boundary Properties

- **Bottom boundary modules**: the leftmost branch
- **Left-boundary condition**
  - Left boundary modules: the rightmost branch
- **Right-boundary condition**
  - Right boundary modules: the bottom-left branch
- **Top-boundary condition**
  - Top boundary modules: bottom-right branch

### Power-Hungry Modules Handling

- **Power-Hungry Modules**
  - Are clustered and restricted to satisfy the boundary property during B*-tree perturbation
  - P/G pads are placed near these modules
**Results on OpenRISC1200**

- Improve on runtime and max IR-drop with little overheads on delay & wirelength (UMC 0.18 um technology)

<table>
<thead>
<tr>
<th></th>
<th>OpenRISC1200</th>
<th>*Astro Flow</th>
<th>*Astro w/ IR-drop Driven Placement</th>
<th>Our Flow</th>
<th>Our Improv. vs. Astro w/ IR-drop</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Die Area (mm²)</strong></td>
<td>3.86</td>
<td>3.86</td>
<td>3.33</td>
<td>15.9%</td>
<td></td>
</tr>
<tr>
<td><strong>Utilization (%)</strong></td>
<td>62</td>
<td>62</td>
<td>72</td>
<td>13.9%</td>
<td></td>
</tr>
<tr>
<td><strong>Wirelength (µm)</strong></td>
<td>1655463</td>
<td>1539125</td>
<td>1540172</td>
<td>-0.1%</td>
<td></td>
</tr>
<tr>
<td><strong>Avg. Delay (ns)</strong></td>
<td>8.62</td>
<td>8.54</td>
<td>8.55</td>
<td>-0.1%</td>
<td></td>
</tr>
<tr>
<td><strong>Max IR-drop (mv)</strong></td>
<td>80.18</td>
<td>78.20</td>
<td>55.14</td>
<td>41.8%</td>
<td></td>
</tr>
<tr>
<td><strong>CPU Runtime (s)</strong></td>
<td>505</td>
<td>346</td>
<td>135</td>
<td>2.56X</td>
<td></td>
</tr>
<tr>
<td><strong>Iterations</strong></td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

*Need iterative and manual P/G network fix

---

**Resulting Voltage Map**

- **Astro design flow**
  - Power-hungry blocks (register files A&B) are placed far away from the power pad

- **Our design flow**
  - Power-hungry blocks are placed beside the power pad