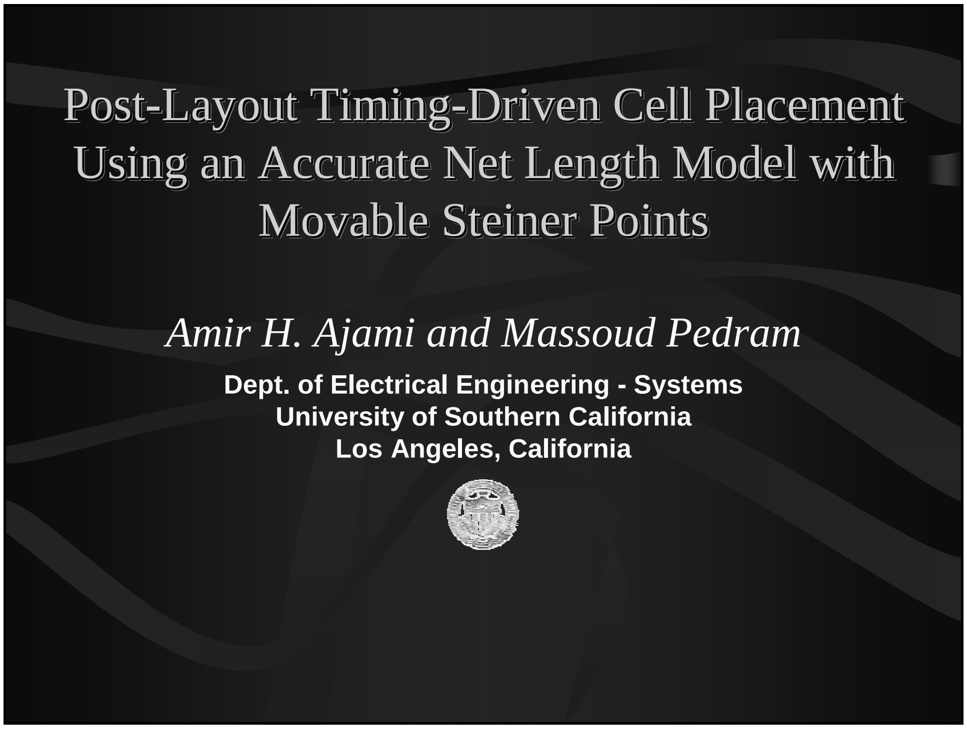
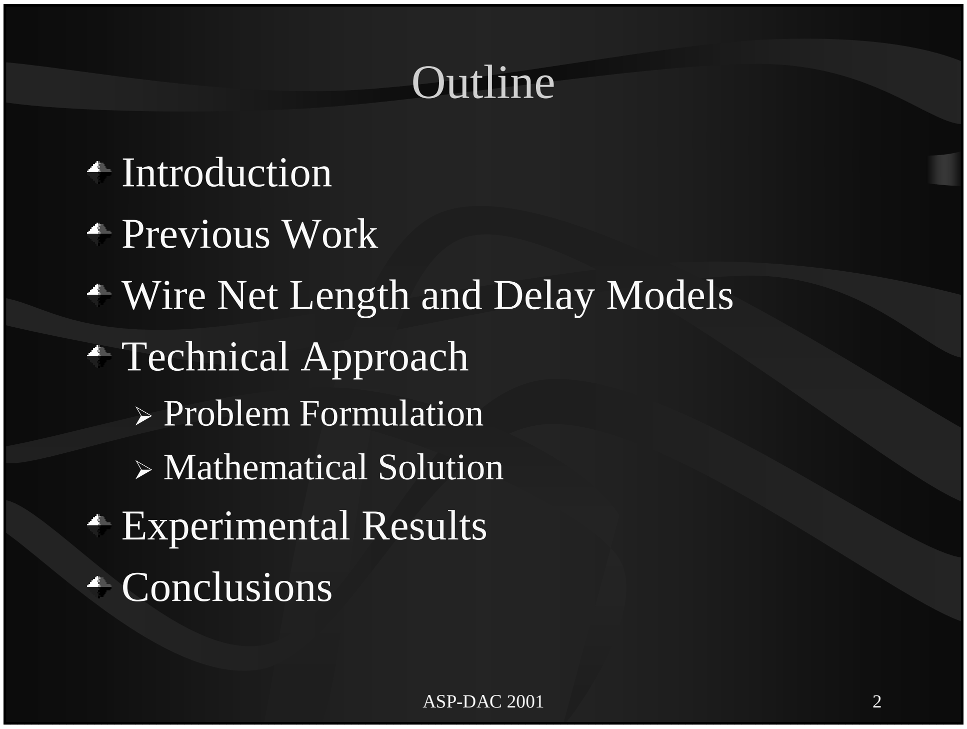


Post-Layout Timing-Driven Cell Placement Using an Accurate Net Length Model with Movable Steiner Points

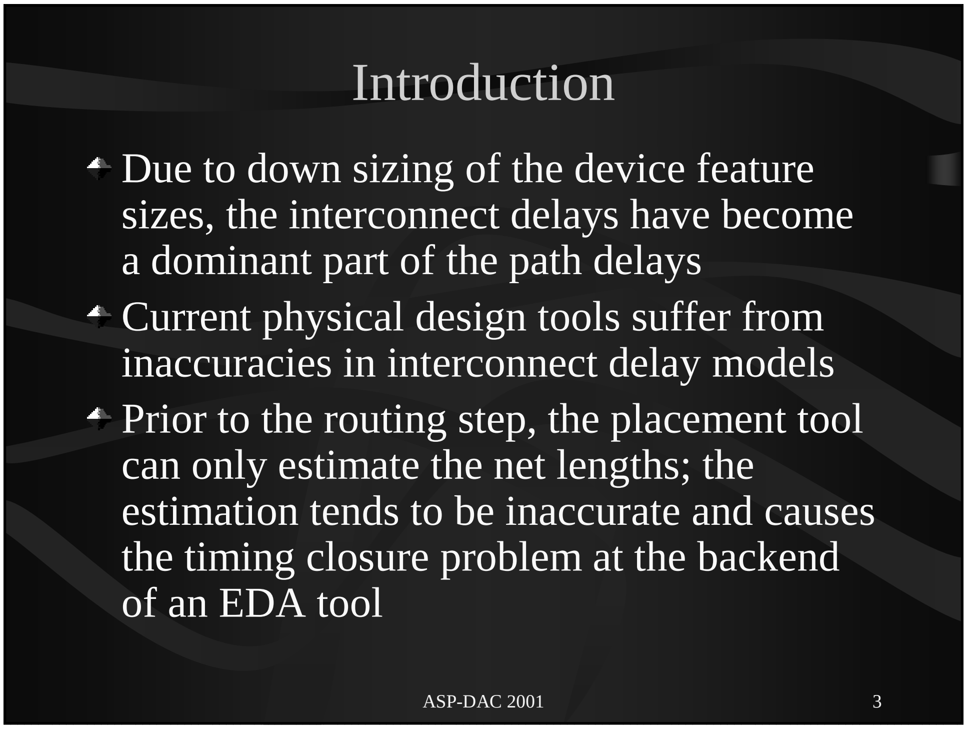
Amir H. Ajami and Massoud Pedram

Dept. of Electrical Engineering - Systems
University of Southern California
Los Angeles, California



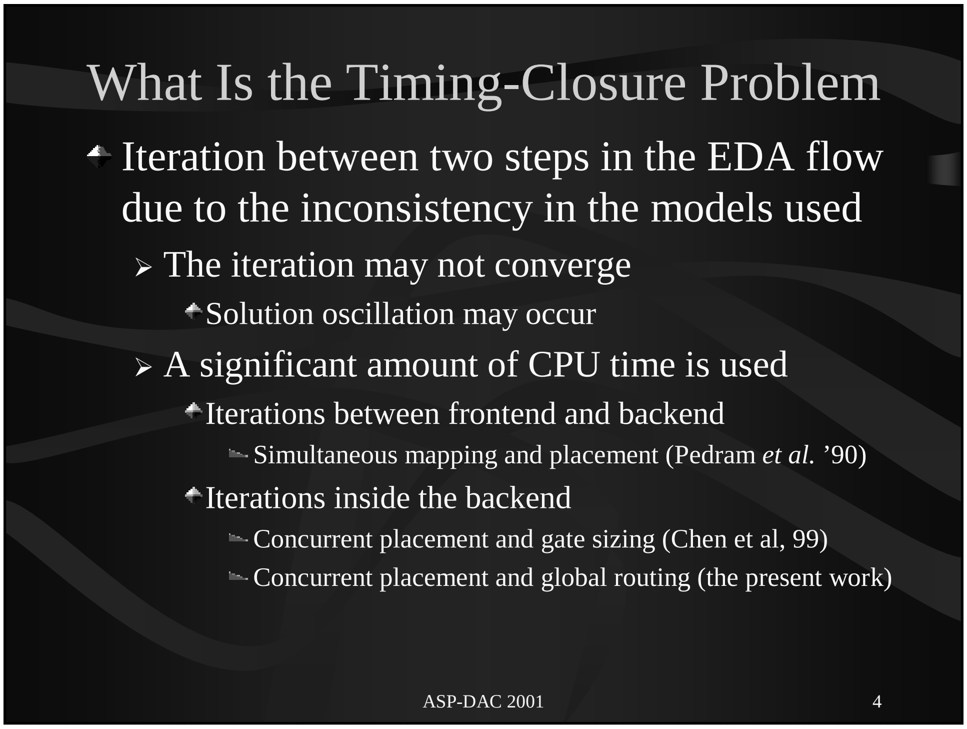
Outline

- Introduction
- Previous Work
- Wire Net Length and Delay Models
- Technical Approach
  - Problem Formulation
  - Mathematical Solution
- Experimental Results
- Conclusions



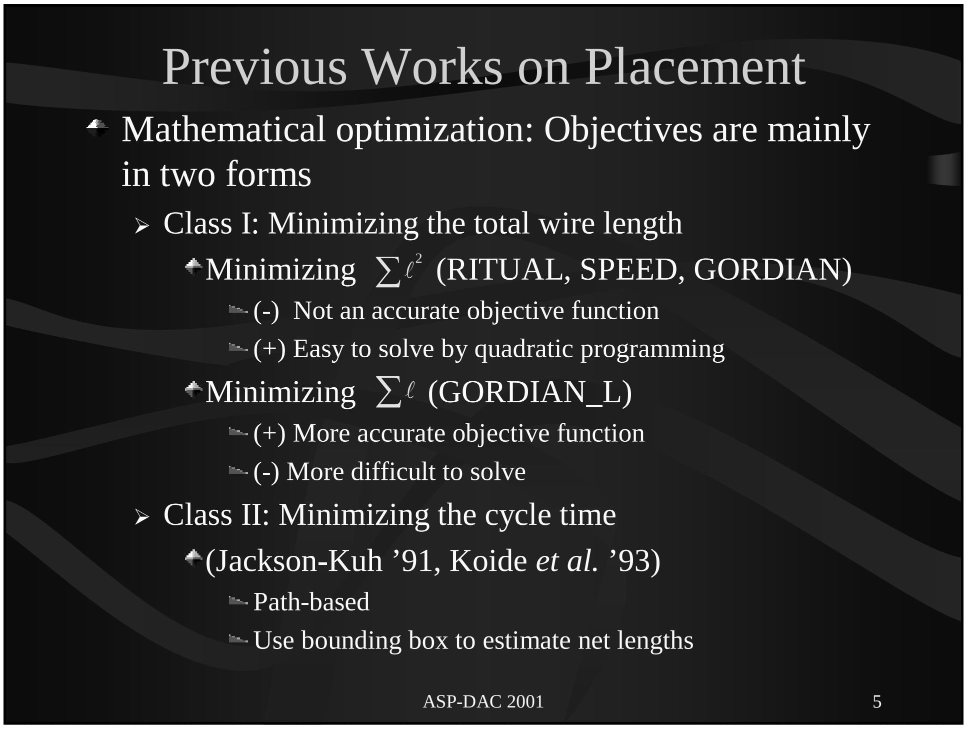
Introduction

- Due to down sizing of the device feature sizes, the interconnect delays have become a dominant part of the path delays
- Current physical design tools suffer from inaccuracies in interconnect delay models
- Prior to the routing step, the placement tool can only estimate the net lengths; the estimation tends to be inaccurate and causes the timing closure problem at the backend of an EDA tool



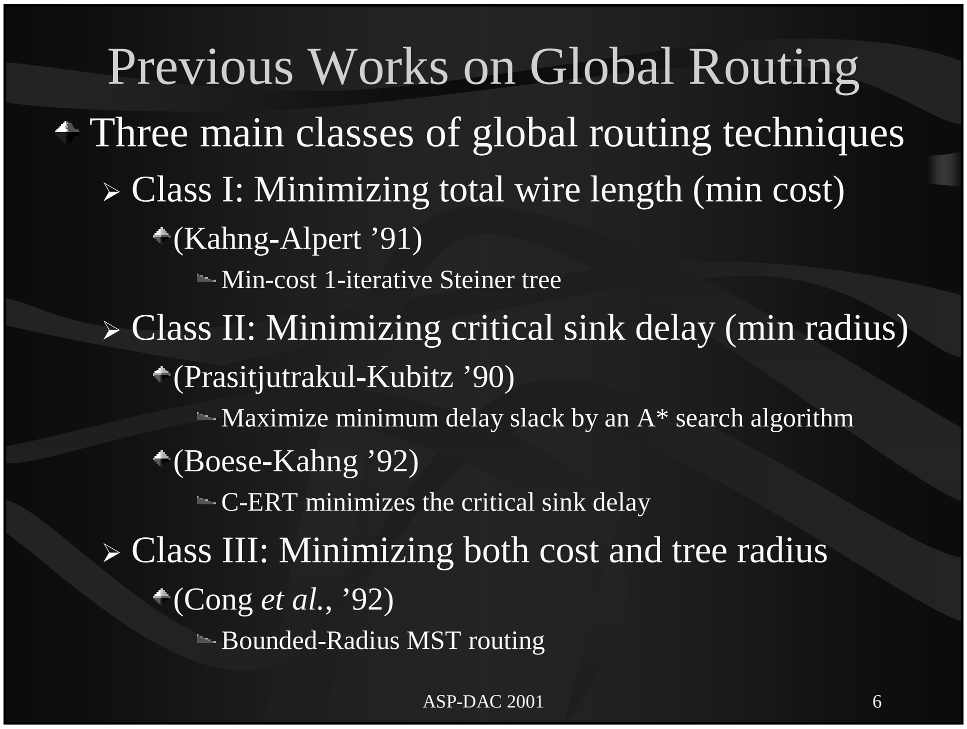
What Is the Timing-Closure Problem

- Iteration between two steps in the EDA flow due to the inconsistency in the models used
  - The iteration may not converge
    - Solution oscillation may occur
  - A significant amount of CPU time is used
    - Iterations between frontend and backend
      - Simultaneous mapping and placement (Pedram *et al.* ’90)
    - Iterations inside the backend
      - Concurrent placement and gate sizing (Chen et al, 99)
      - Concurrent placement and global routing (the present work)



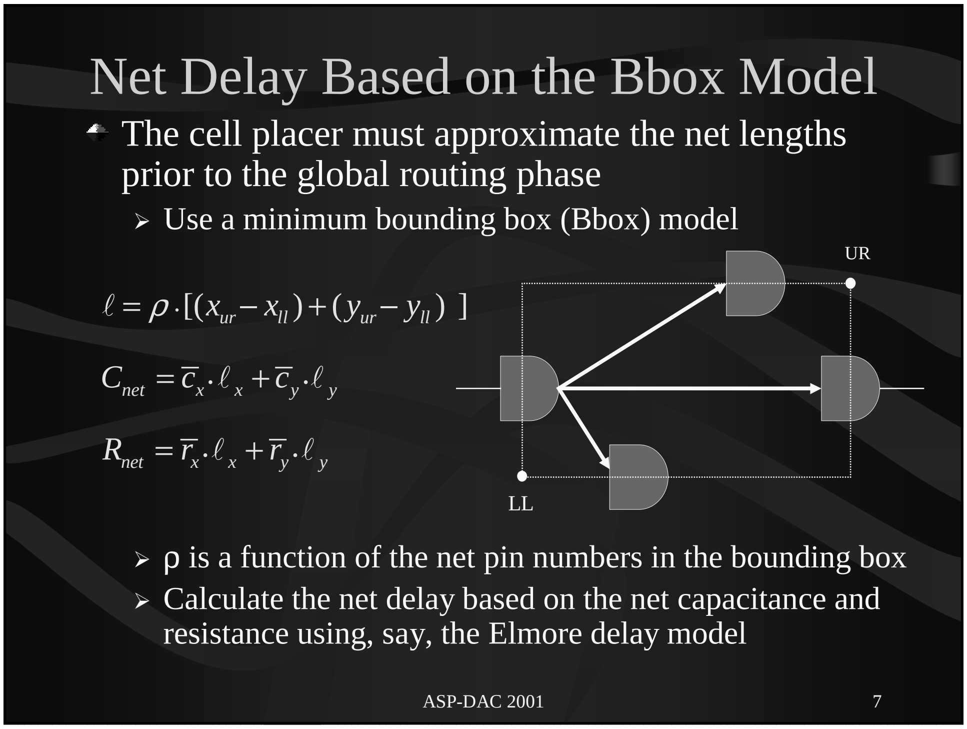
Previous Works on Placement

- Mathematical optimization: Objectives are mainly in two forms
  - Class I: Minimizing the total wire length
    - Minimizing $\sum \ell^2$ (RITUAL, SPEED, GORDIAN)
      - (-) Not an accurate objective function
      - (+) Easy to solve by quadratic programming
    - Minimizing $\sum \ell$ (GORDIAN_L)
      - (+) More accurate objective function
      - (-) More difficult to solve
  - Class II: Minimizing the cycle time
    - (Jackson-Kuh '91, Koide *et al.* '93)
      - Path-based
      - Use bounding box to estimate net lengths



Previous Works on Global Routing

- Three main classes of global routing techniques
  - Class I: Minimizing total wire length (min cost)
    - (Kahng-Alpert '91)
      - Min-cost 1-iterative Steiner tree
  - Class II: Minimizing critical sink delay (min radius)
    - (Prasitjutrakul-Kubitz '90)
      - Maximize minimum delay slack by an A* search algorithm
    - (Boese-Kahng '92)
      - C-ERT minimizes the critical sink delay
  - Class III: Minimizing both cost and tree radius
    - (Cong *et al.*, '92)
      - Bounded-Radius MST routing



Net Delay Based on the Bbox Model

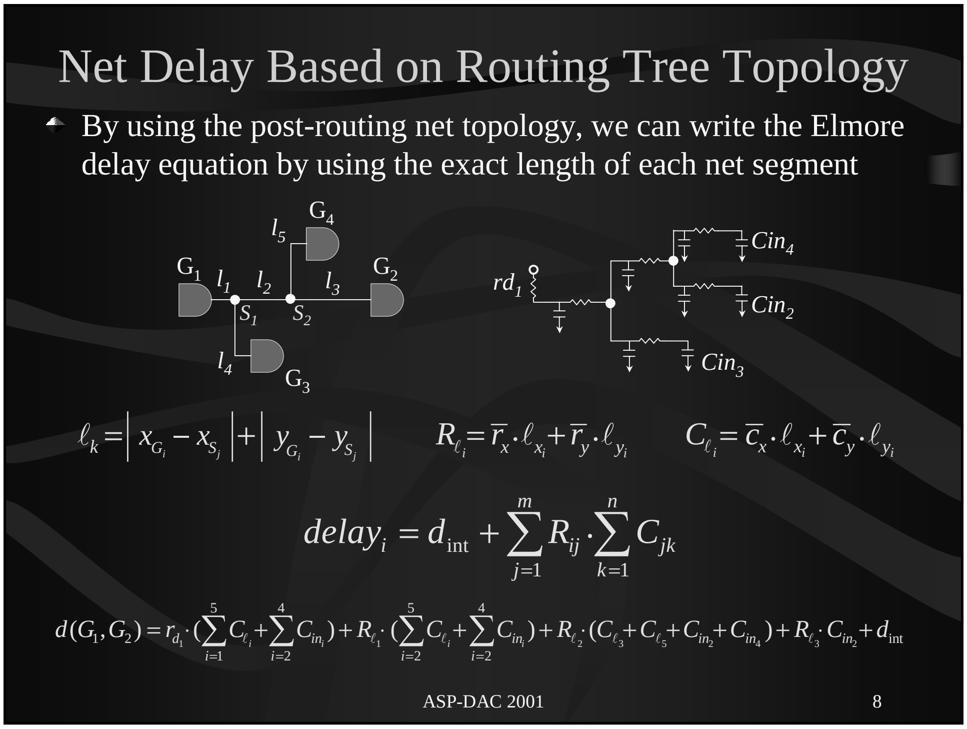
- The cell placer must approximate the net lengths prior to the global routing phase
  - Use a minimum bounding box (Bbox) model

$$\ell = \rho \cdot [(x_{ur} - x_{ll}) + (y_{ur} - y_{ll})\ ]$$

$$C_{net} = \overline{c}_x \cdot \ell_x + \overline{c}_y \cdot \ell_y$$

$$R_{net} = \overline{r}_x \cdot \ell_x + \overline{r}_y \cdot \ell_y$$

  - ρ is a function of the net pin numbers in the bounding box
  - Calculate the net delay based on the net capacitance and resistance using, say, the Elmore delay model



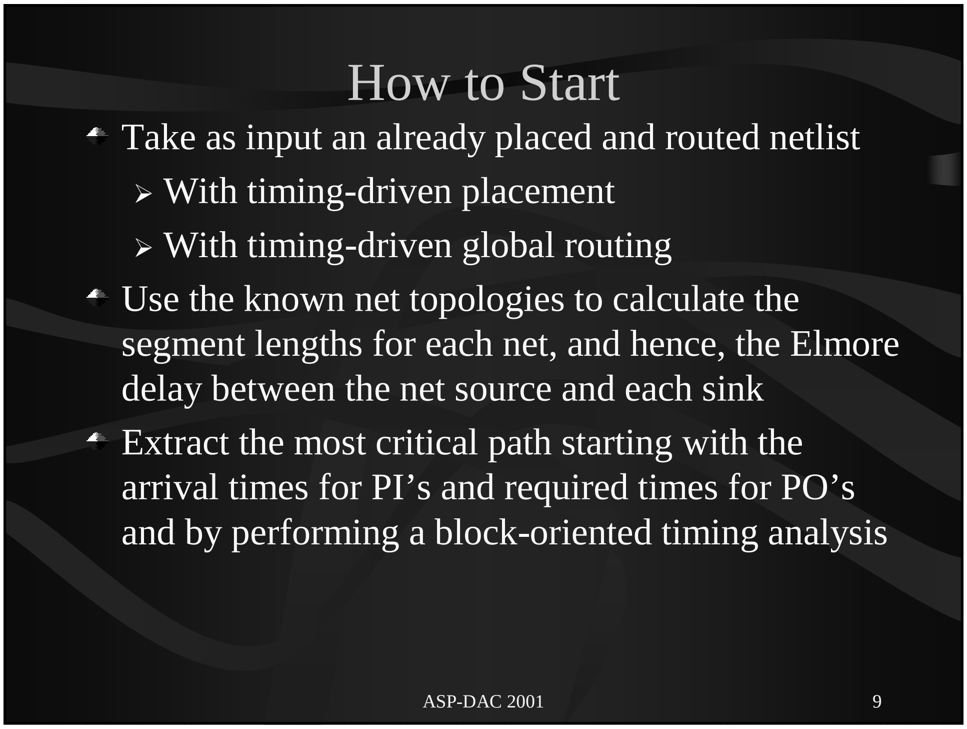
Net Delay Based on Routing Tree Topology

- By using the post-routing net topology, we can write the Elmore delay equation by using the exact length of each net segment

$$\ell_k = \left| x_{G_i} - x_{S_j} \right| + \left| y_{G_i} - y_{S_j} \right| \qquad R_{\ell_i} = \overline{r}_x \cdot \ell_{x_i} + \overline{r}_y \cdot \ell_{y_i} \qquad C_{\ell_i} = \overline{c}_x \cdot \ell_{x_i} + \overline{c}_y \cdot \ell_{y_i}$$

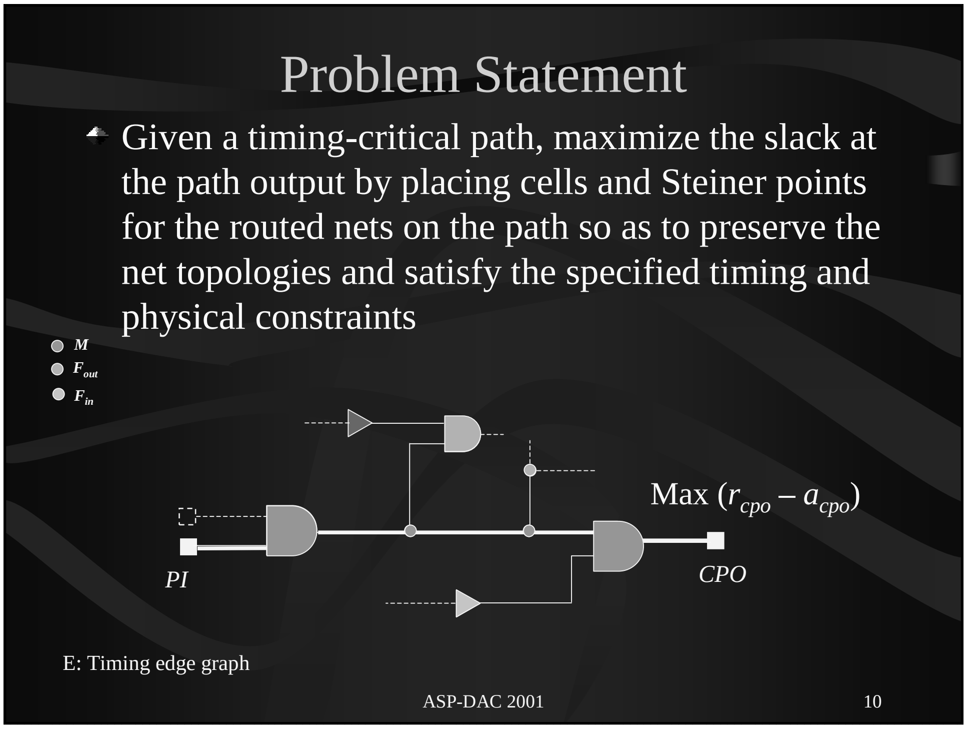
$$delay_i = d_{\text{int}} + \sum_{j=1}^{m} R_{ij} \cdot \sum_{k=1}^{n} C_{jk}$$

$$d(G_1, G_2) = r_{d_1} \cdot (\sum_{i=1}^{5} C_{\ell_i} + \sum_{i=2}^{4} C_{in_i}) + R_{\ell_1} \cdot (\sum_{i=2}^{5} C_{\ell_i} + \sum_{i=2}^{4} C_{in_i}) + R_{\ell_2} \cdot (C_{\ell_3} + C_{\ell_5} + C_{in_2} + C_{in_4}) + R_{\ell_3} \cdot C_{in_2} + d_{\text{int}}$$



How to Start

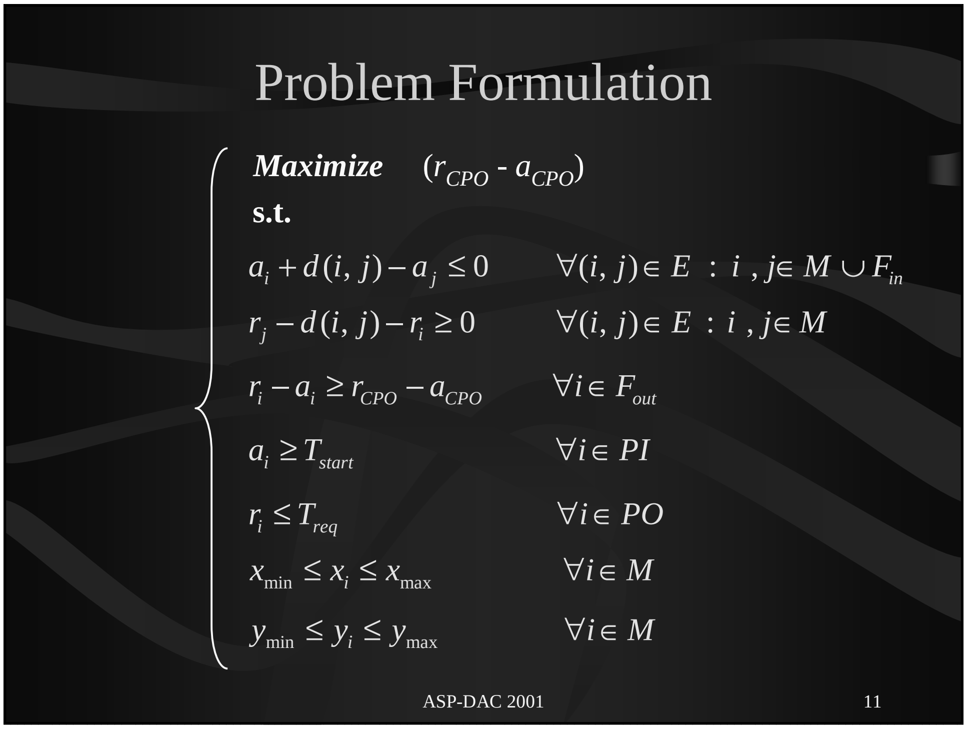
- Take as input an already placed and routed netlist
  - With timing-driven placement
  - With timing-driven global routing
- Use the known net topologies to calculate the segment lengths for each net, and hence, the Elmore delay between the net source and each sink
- Extract the most critical path starting with the arrival times for PI's and required times for PO's and by performing a block-oriented timing analysis



Problem Statement

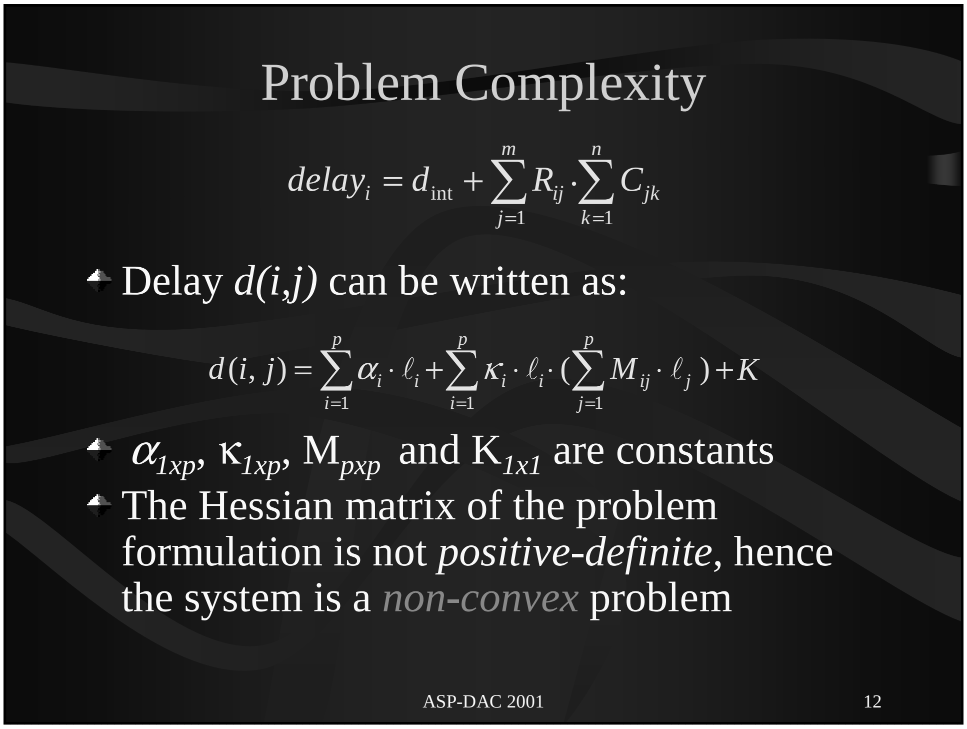
- Given a timing-critical path, maximize the slack at the path output by placing cells and Steiner points for the routed nets on the path so as to preserve the net topologies and satisfy the specified timing and physical constraints

E: Timing edge graph



Problem Formulation

$$\begin{cases} \textbf{\textit{Maximize}} \quad (r_{CPO} - a_{CPO}) & \\ \textbf{s.t.} & \\ a_i + d(i,j) - a_j \le 0 & \forall (i,j) \in E \ : \ i\ , j \in M \cup F_{in} \\ r_j - d(i,j) - r_i \ge 0 & \forall (i,j) \in E \ : \ i\ , j \in M \\ r_i - a_i \ge r_{CPO} - a_{CPO} & \forall i \in F_{out} \\ a_i \ge T_{start} & \forall i \in PI \\ r_i \le T_{req} & \forall i \in PO \\ x_{\min} \le x_i \le x_{\max} & \forall i \in M \\ y_{\min} \le y_i \le y_{\max} & \forall i \in M \end{cases}$$



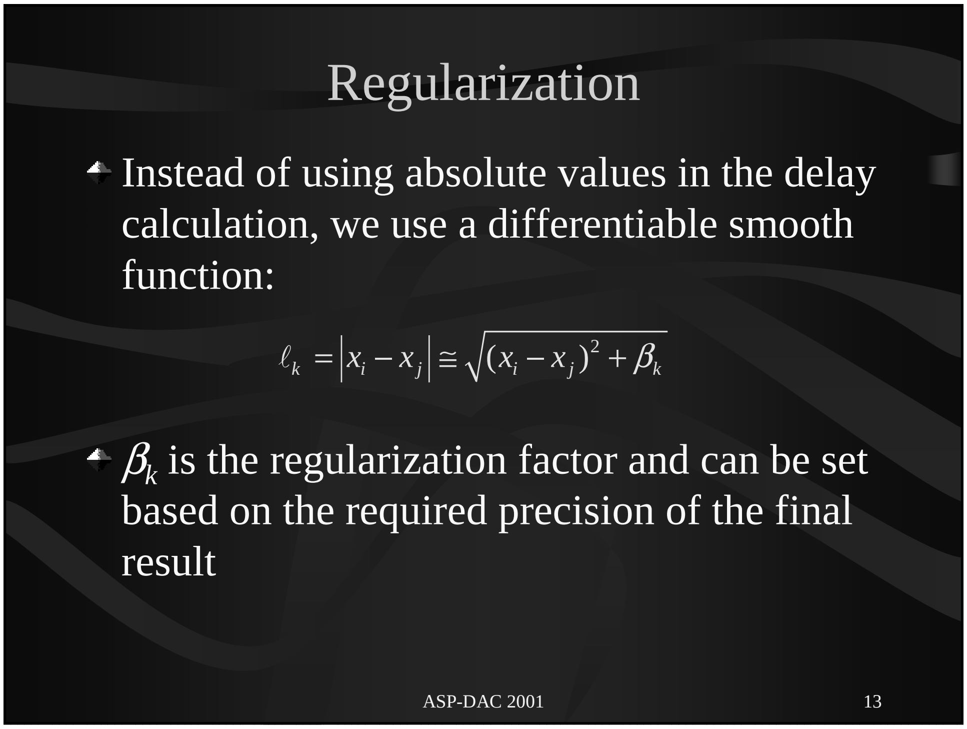
Problem Complexity

$$delay_i = d_{\text{int}} + \sum_{j=1}^{m} R_{ij} \cdot \sum_{k=1}^{n} C_{jk}$$

- Delay *d(i,j)* can be written as:

$$d(i,j) = \sum_{i=1}^{p} \alpha_i \cdot \ell_i + \sum_{i=1}^{p} \kappa_i \cdot \ell_i \cdot (\sum_{j=1}^{p} M_{ij} \cdot \ell_j) + K$$

- α_{1xp}, κ_{1xp}, M_{pxp} and K_{1x1} are constants
- The Hessian matrix of the problem formulation is not *positive-definite*, hence the system is a *non-convex* problem

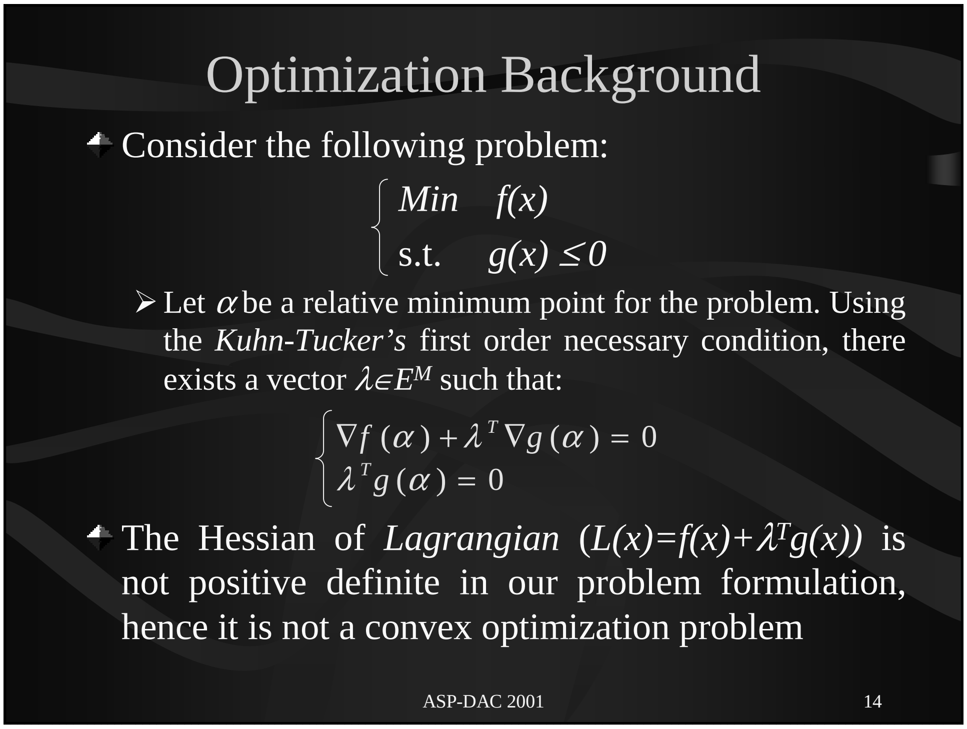


Regularization

- Instead of using absolute values in the delay calculation, we use a differentiable smooth function:

$$\ell_k = \left| x_i - x_j \right| \cong \sqrt{(x_i - x_j)^2 + \beta_k}$$

- β_k is the regularization factor and can be set based on the required precision of the final result



Optimization Background

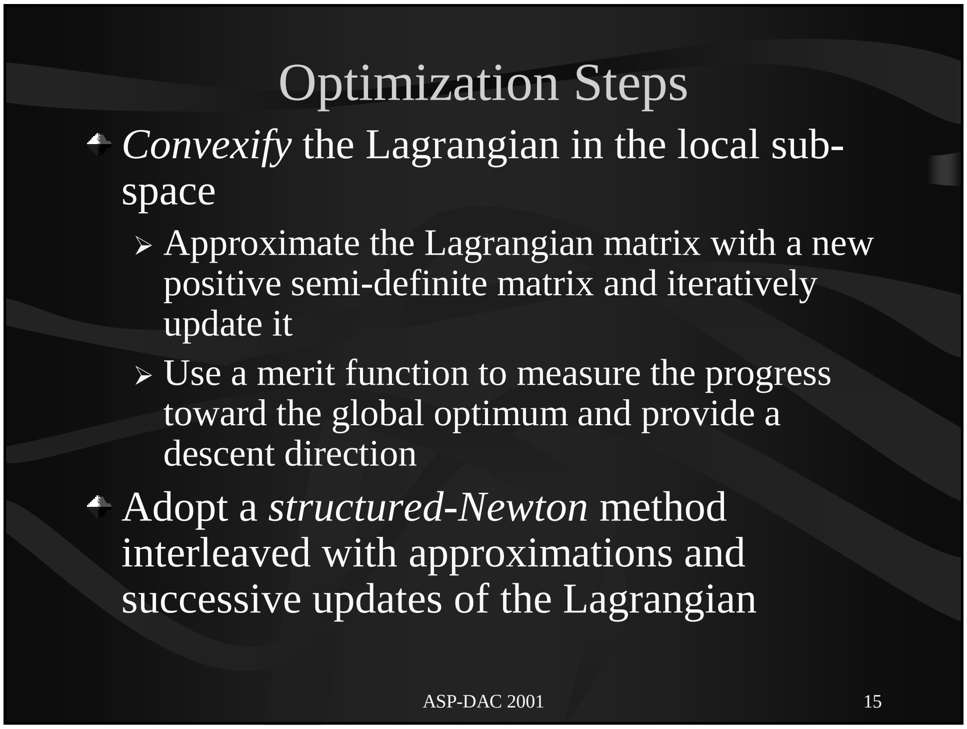
- Consider the following problem:

$$\begin{cases} Min \quad f(x) \\ \text{s.t.} \quad g(x) \le 0 \end{cases}$$

  - Let α be a relative minimum point for the problem. Using the *Kuhn-Tucker's* first order necessary condition, there exists a vector $\lambda \in E^M$ such that:

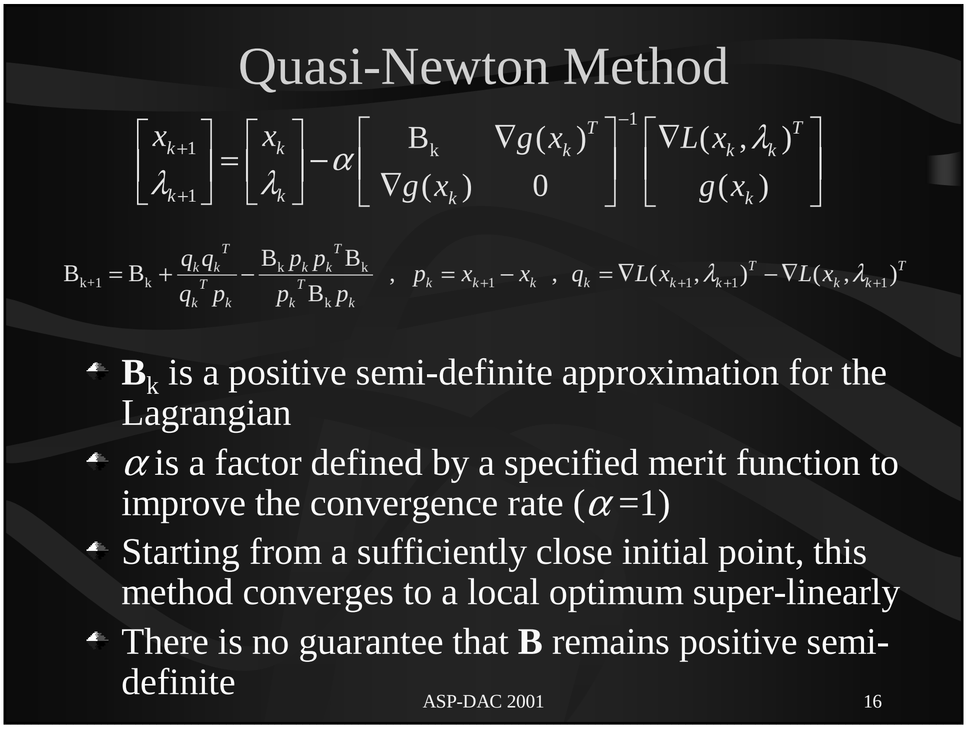
$$\begin{cases} \nabla f(\alpha) + \lambda^T \nabla g(\alpha) = 0 \\ \lambda^T g(\alpha) = 0 \end{cases}$$

- The Hessian of *Lagrangian* ($L(x)=f(x)+\lambda^T g(x)$) is not positive definite in our problem formulation, hence it is not a convex optimization problem



Optimization Steps

- *Convexify* the Lagrangian in the local sub-space
  - Approximate the Lagrangian matrix with a new positive semi-definite matrix and iteratively update it
  - Use a merit function to measure the progress toward the global optimum and provide a descent direction
- Adopt a *structured-Newton* method interleaved with approximations and successive updates of the Lagrangian

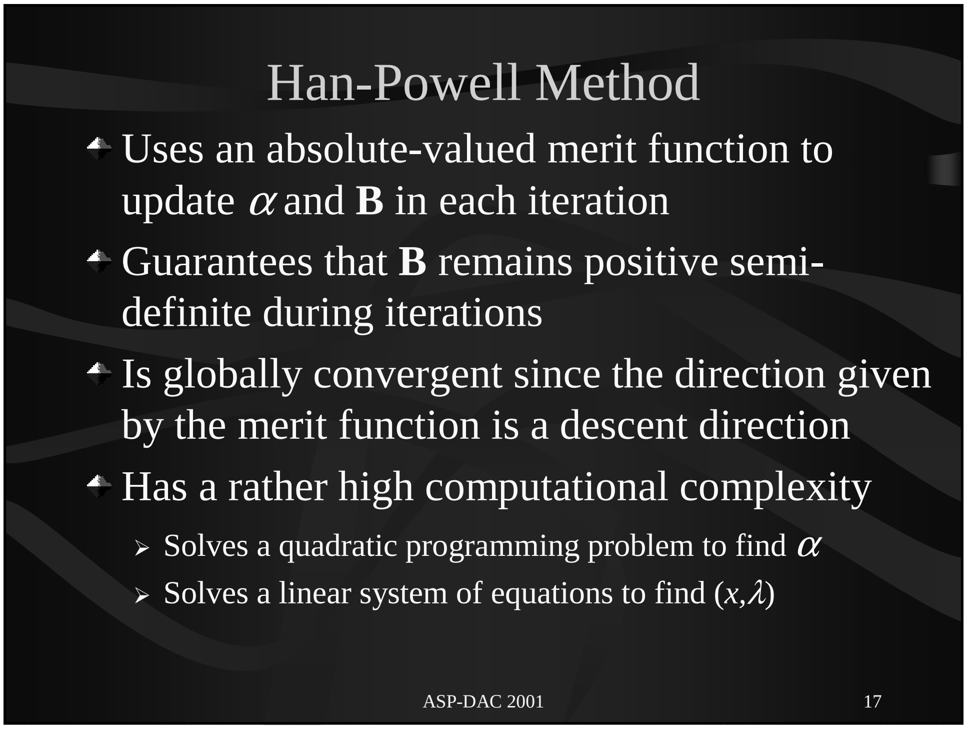


Quasi-Newton Method

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} - \alpha \begin{bmatrix} \mathrm{B_k} & \nabla g(x_k)^T \\ \nabla g(x_k) & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla L(x_k, \lambda_k)^T \\ g(x_k) \end{bmatrix}$$

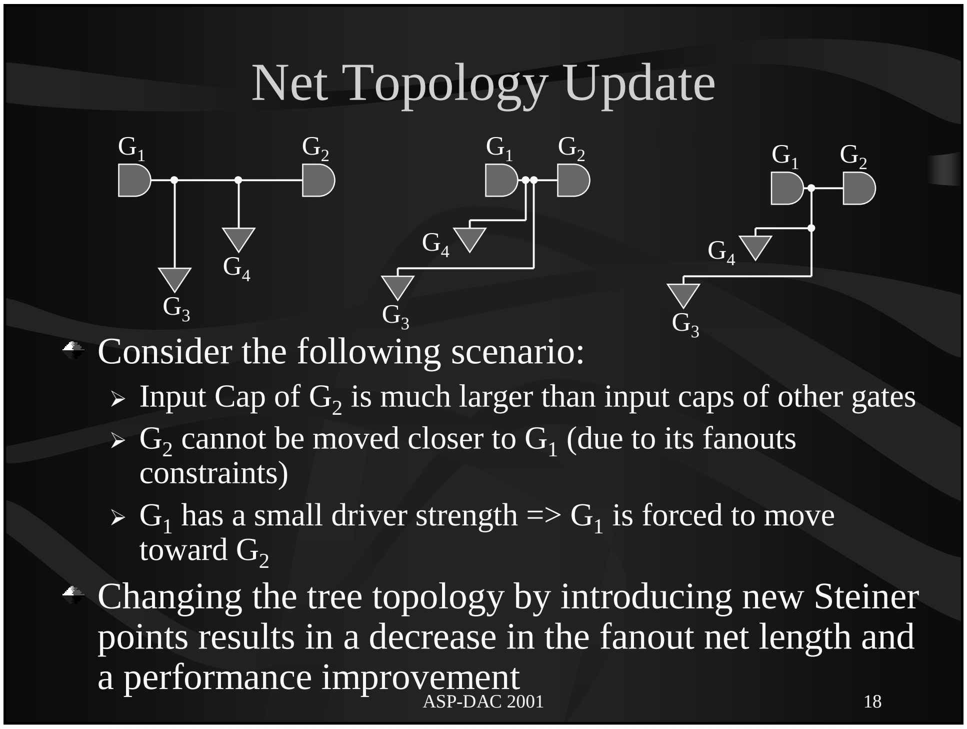
$$\mathrm{B_{k+1}} = \mathrm{B_k} + \frac{q_k q_k{}^T}{q_k{}^T p_k} - \frac{\mathrm{B_k} p_k p_k{}^T \mathrm{B_k}}{p_k{}^T \mathrm{B_k} p_k} \;,\; p_k = x_{k+1} - x_k \;,\; q_k = \nabla L(x_{k+1}, \lambda_{k+1})^T - \nabla L(x_k, \lambda_{k+1})^T$$

- $\mathbf{B}_\mathrm{k}$ is a positive semi-definite approximation for the Lagrangian
- α is a factor defined by a specified merit function to improve the convergence rate (α=1)
- Starting from a sufficiently close initial point, this method converges to a local optimum super-linearly
- There is no guarantee that **B** remains positive semi-definite



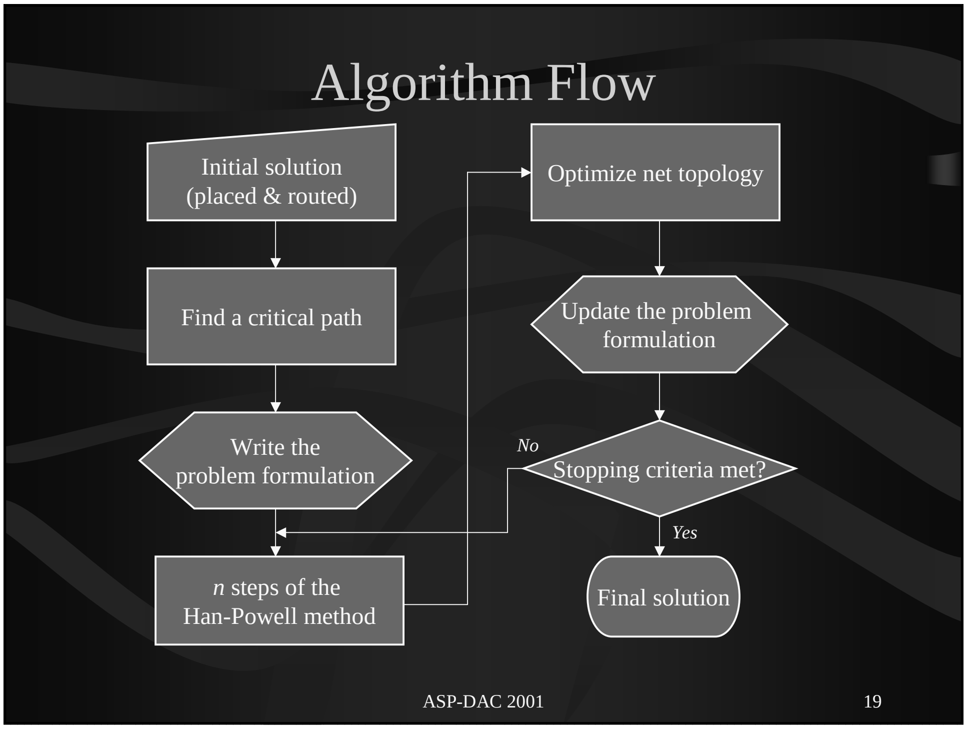
Han-Powell Method

- Uses an absolute-valued merit function to update α and **B** in each iteration
- Guarantees that **B** remains positive semi-definite during iterations
- Is globally convergent since the direction given by the merit function is a descent direction
- Has a rather high computational complexity
  - Solves a quadratic programming problem to find α
  - Solves a linear system of equations to find (x,λ)



Net Topology Update

- Consider the following scenario:
  - Input Cap of G_2 is much larger than input caps of other gates
  - G_2 cannot be moved closer to G_1 (due to its fanouts constraints)
  - G_1 has a small driver strength => G_1 is forced to move toward G_2
- Changing the tree topology by introducing new Steiner points results in a decrease in the fanout net length and a performance improvement



Algorithm Flow

Initial solution (placed & routed)

Find a critical path

Write the problem formulation

n steps of the Han-Powell method

Optimize net topology

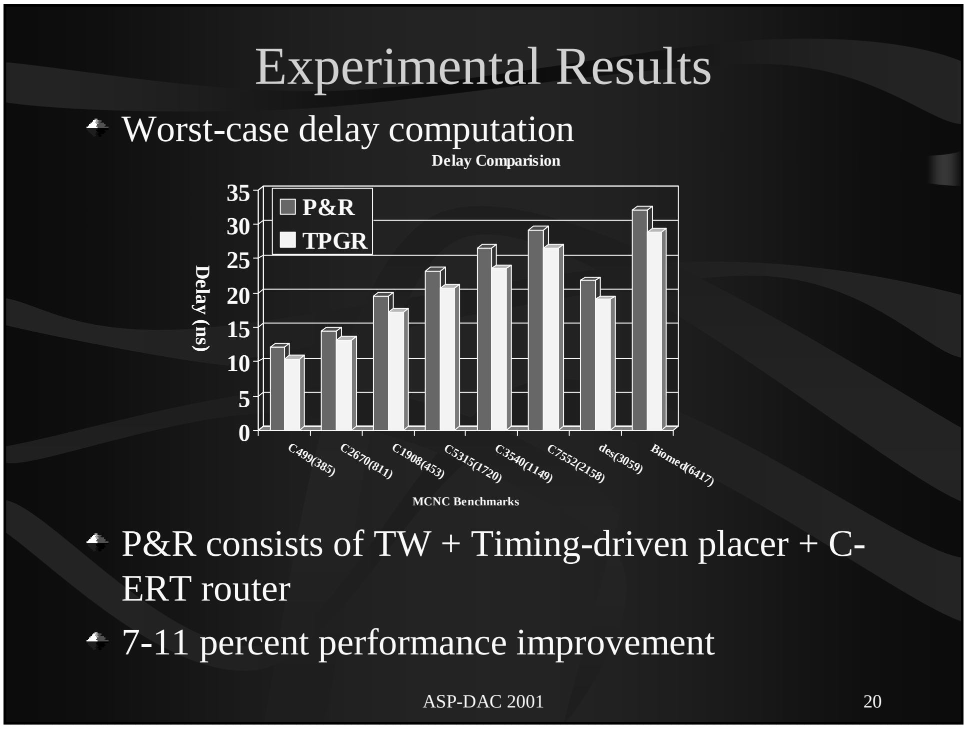
Update the problem formulation

Stopping criteria met?

No

Yes

Final solution

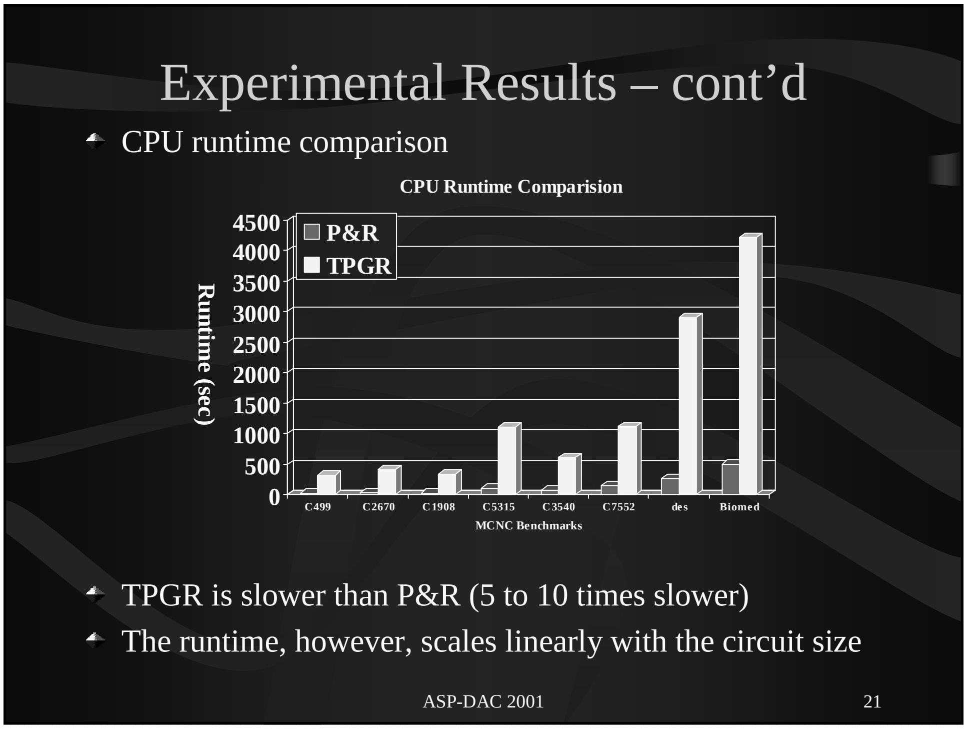


Experimental Results

- Worst-case delay computation

Delay Comparision

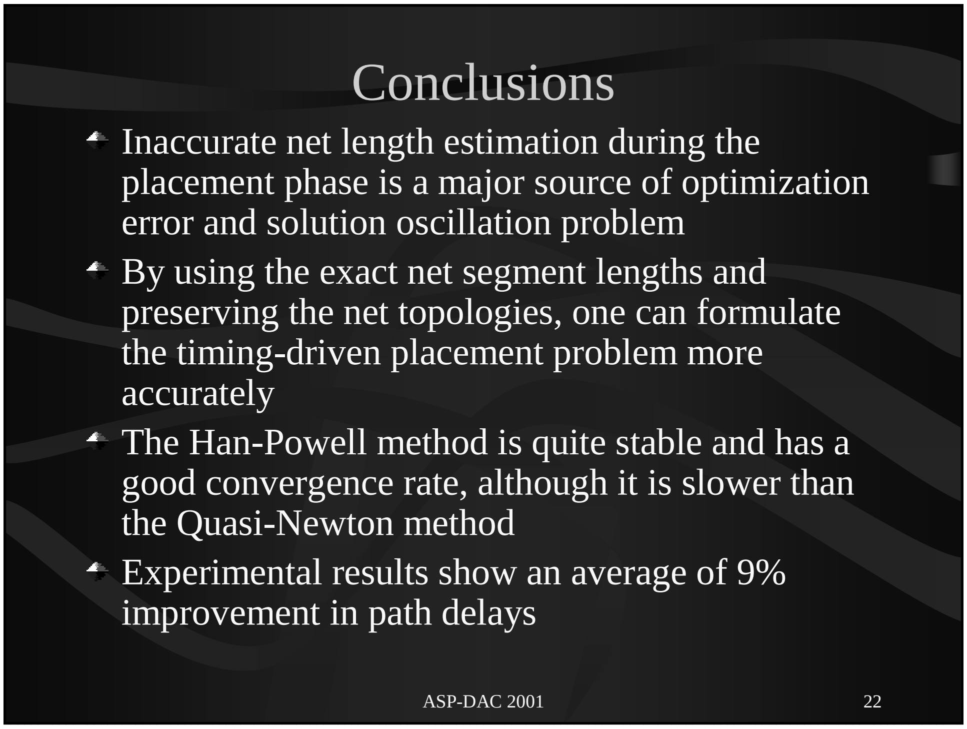
- P&R consists of TW + Timing-driven placer + C-ERT router
- 7-11 percent performance improvement



Experimental Results – cont’d

- CPU runtime comparison

- TPGR is slower than P&R (5 to 10 times slower)
- The runtime, however, scales linearly with the circuit size



Conclusions

- Inaccurate net length estimation during the placement phase is a major source of optimization error and solution oscillation problem
- By using the exact net segment lengths and preserving the net topologies, one can formulate the timing-driven placement problem more accurately
- The Han-Powell method is quite stable and has a good convergence rate, although it is slower than the Quasi-Newton method
- Experimental results show an average of 9% improvement in path delays