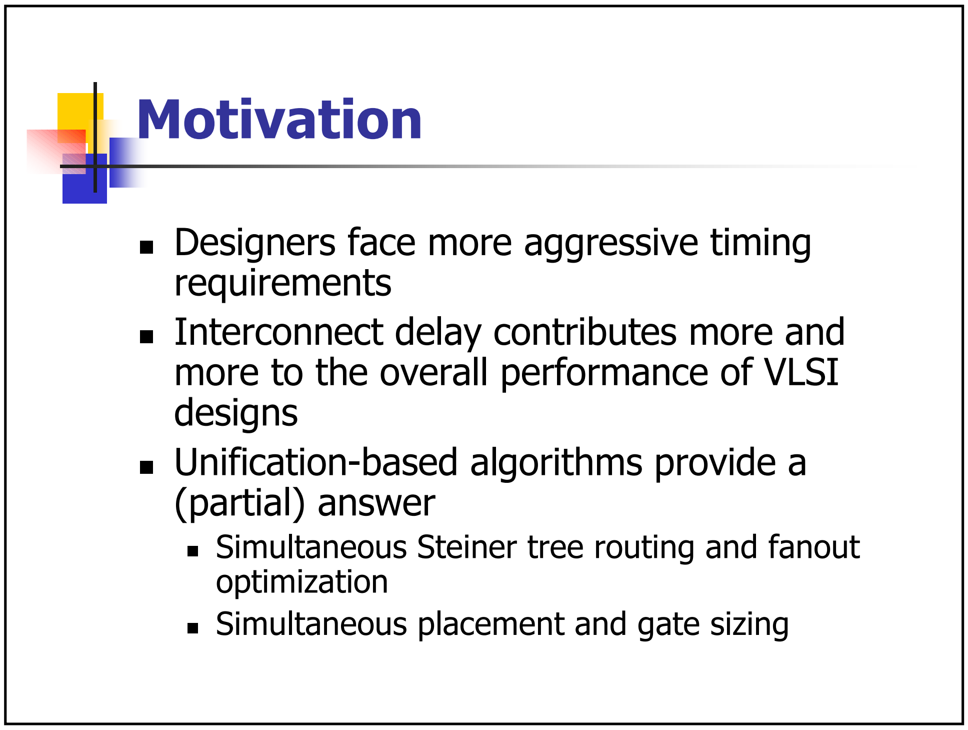
# PILOT: Placement with Integrated Logic Optimization for Timing

**Jinan Lou, Wei Chen, M. Pedram**

Department of EE - Systems
University of Southern California
Los Angeles, CA 90089

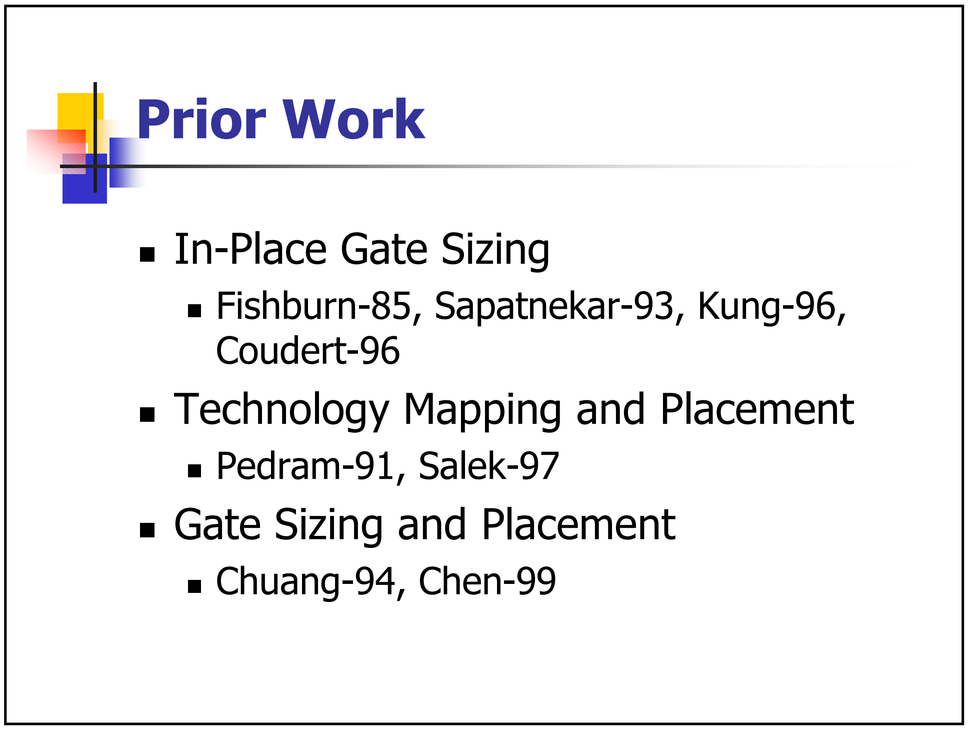
## Outline

- Introduction
- Algorithm
- Experimental Results
- Conclusions



## Motivation

- Designers face more aggressive timing requirements
- Interconnect delay contributes more and more to the overall performance of VLSI designs
- Unification-based algorithms provide a (partial) answer
  - Simultaneous Steiner tree routing and fanout optimization
  - Simultaneous placement and gate sizing



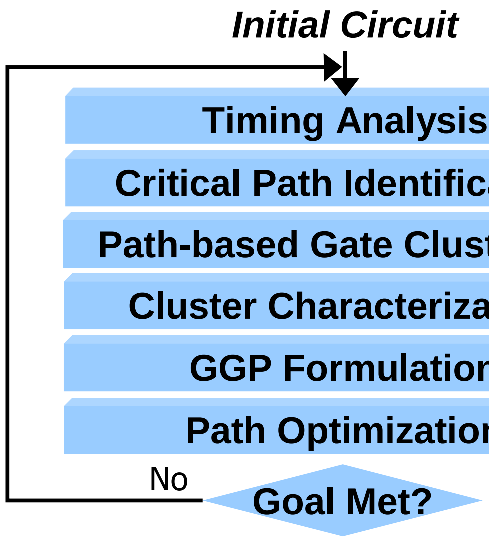
## Prior Work

- In-Place Gate Sizing
  - Fishburn-85, Sapatnekar-93, Kung-96, Coudert-96
- Technology Mapping and Placement
  - Pedram-91, Salek-97
- Gate Sizing and Placement
  - Chuang-94, Chen-99

## Problem Definition

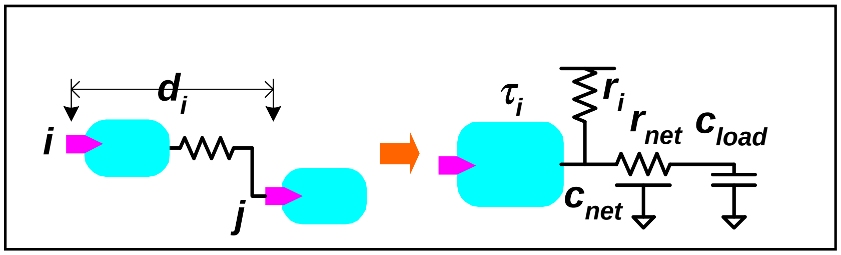
- Given a mapped and placed circuit, perform simultaneous logic restructuring and re-placement of newly-created gates in the circuit so as to minimize the circuit delay

## PILOT Flow



## Delay Calculation

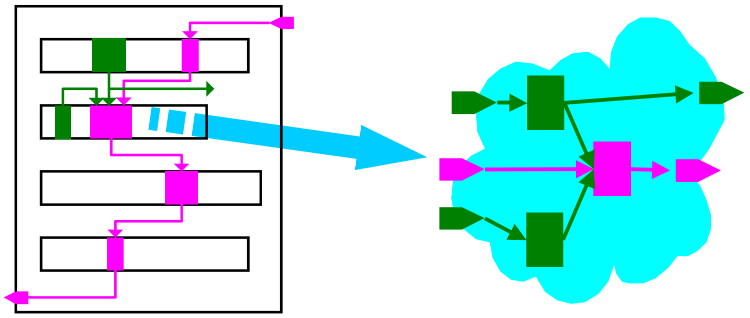
***A pin-to-pin delay model:***



$$d_{ij} = \underbrace{\tau_i + r_i \times (c_{net} + c_{load})}_{\text{super-cell delay}} + \underbrace{r_{net} \times c_{load}}_{\text{wire delay}}$$

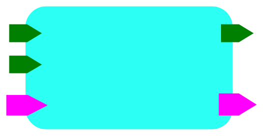
## Super-cell Definition

- A super-cell implements a multi-input, multi-output Boolean function
- A level-*L* super-cell is identified by a root and a set of gates, which are its level-*L* fanins



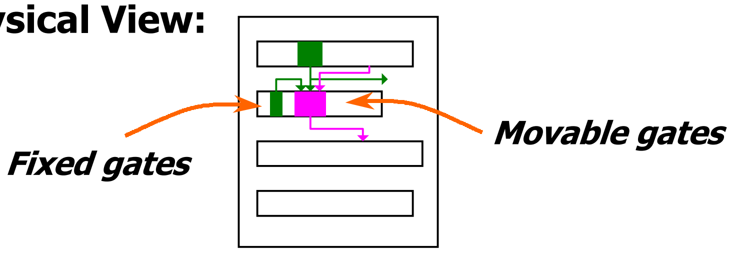
## Two Views of a Super-cell

**Logical View:**



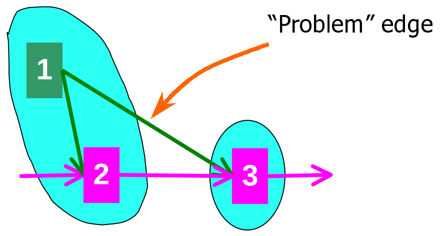
**Boolean function**

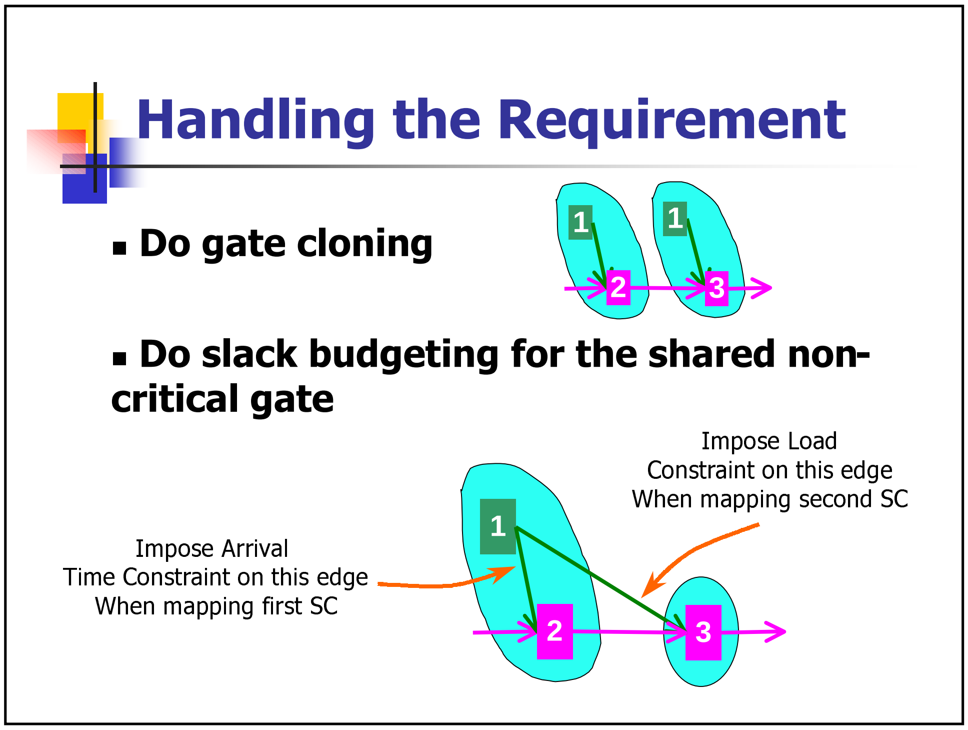
**Physical View:**



## Basic Requirement on Super-cell Connections

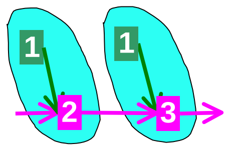
- **Super-cells should be directly connected through the target critical path only**



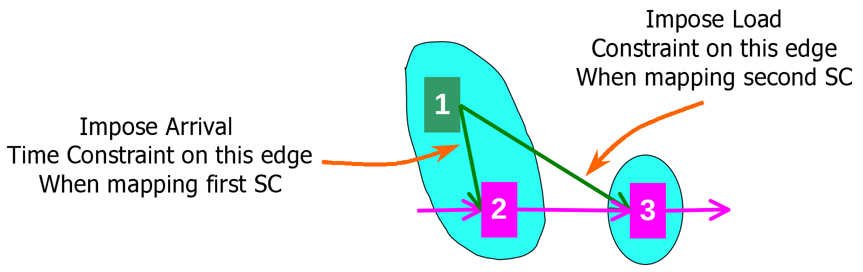


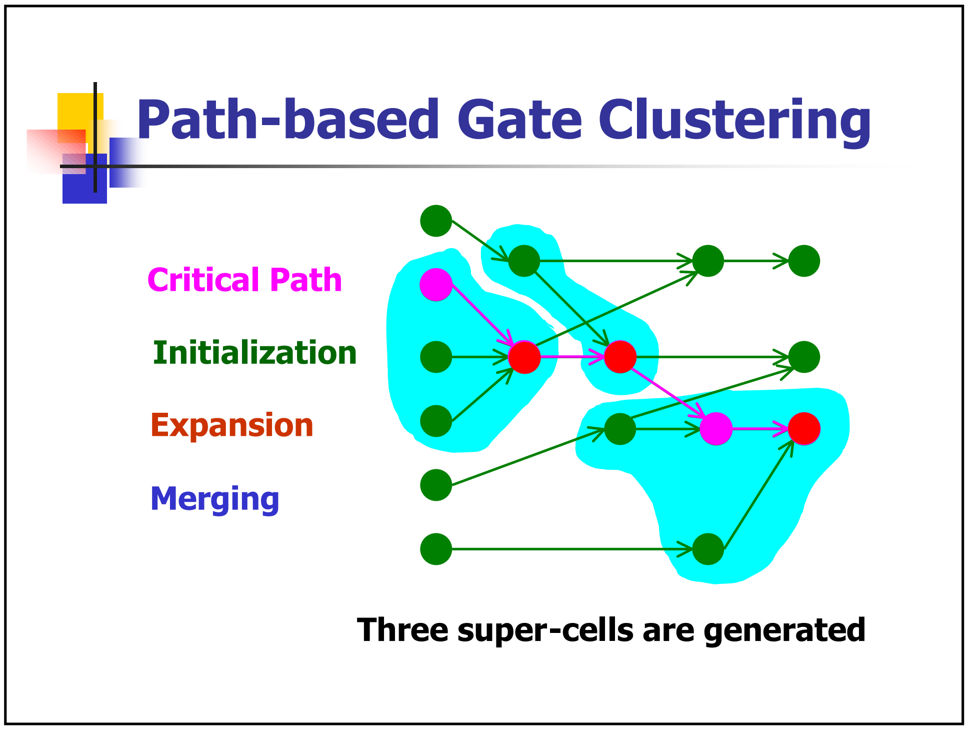
## Handling the Requirement

- Do gate cloning



- Do slack budgeting for the shared non-critical gate


Impose Load Constraint on this edge When mapping second SC
Impose Arrival Time Constraint on this edge When mapping first SC
1
2
3



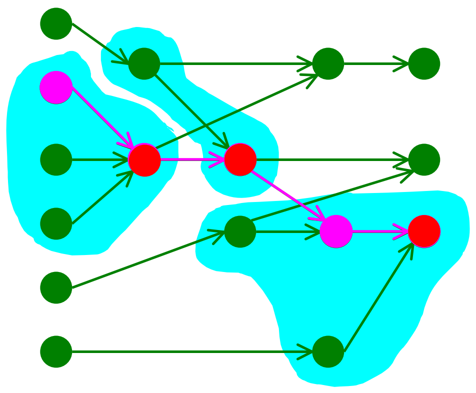
## Path-based Gate Clustering

Critical Path

Initialization

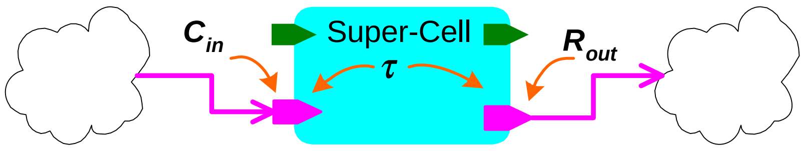
Expansion

Merging



Three super-cells are generated

## Super-Cell Characterization

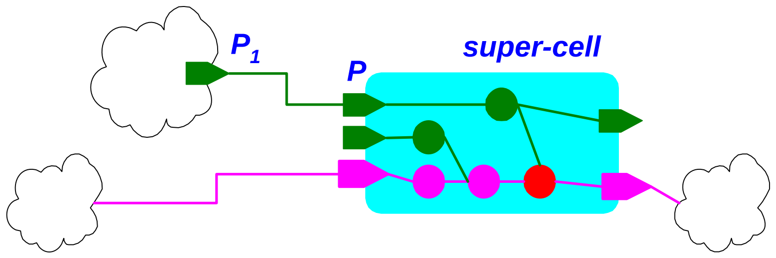


***Characterization function:***

$$\tau = F\left(C_{in},\ R_{out}\right)$$

## Timing Constraints I

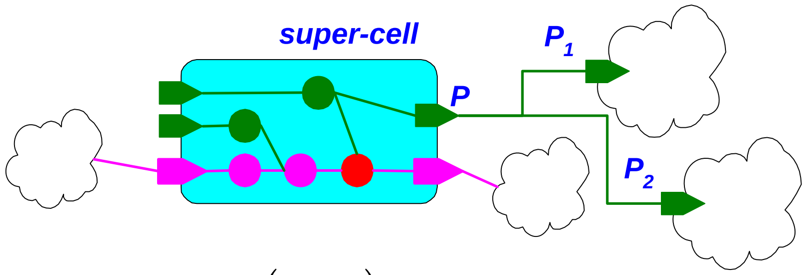
**Fanin-induced timing constraints**



$$C_{in,P} \le \alpha_1 \times C_{\max,p_1}$$

## Timing Constraints II

**Fanout-induced timing constraints**



$$a_P \leq \alpha_2 \times \mathrm{MIN}\left(req_{P_i}\right) \qquad \forall fanout\ P_i\ of\ P$$

**Choose $\alpha_1$ and $\alpha_2$ to distribute slack between the mapping and the placement**

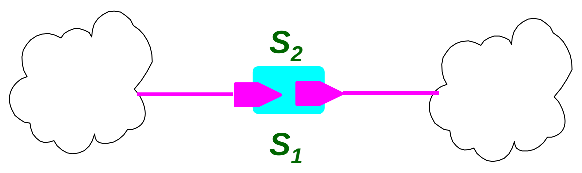
## Timing-Infeasible Solutions

- **Any mapping solution that violates the timing constraints is deemed timing-infeasible**
- **We exclude timing-infeasible solutions from further consideration**

## Inferior Solutions

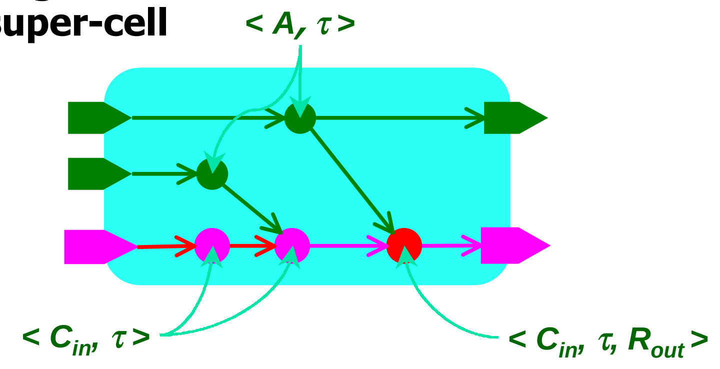
- $S_1$ is said to be inferior to $S_2$ iff:

$$C_{in,1} \geq C_{in,2},\ \tau_1 \geq \tau_2,\ R_{out,1} \geq R_{out,2}$$

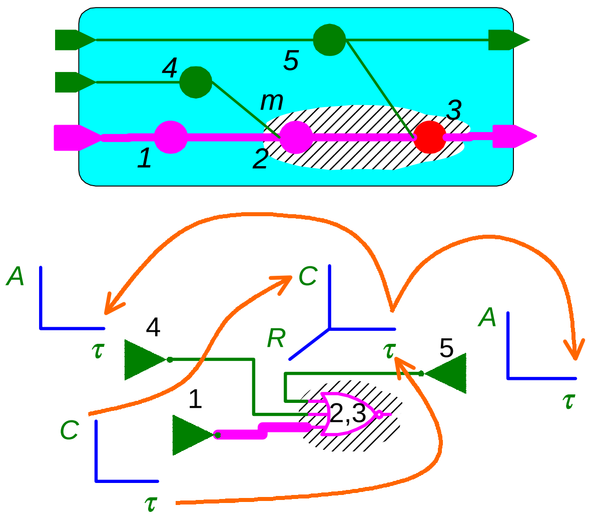


## Dynamic Programming

- Three types of solution curves are used during the dynamic programming step to generate the solution set for each super-cell

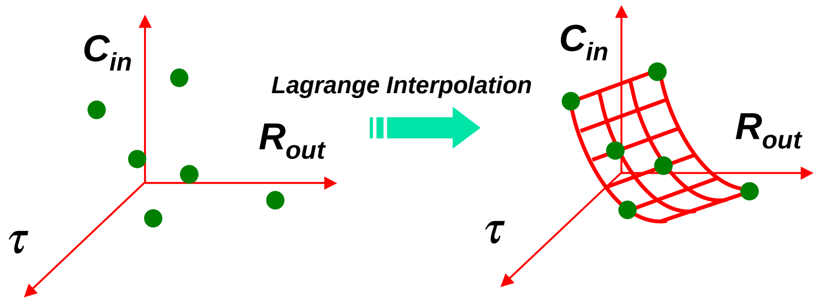


## Solution Curve Generation



## Solution-Set Interpolation

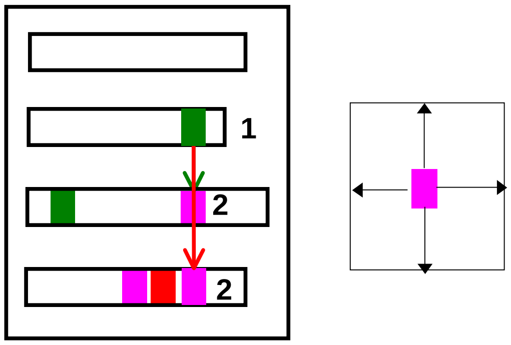
- **The discrete solution set is interpolated to generate the characterization function**



- **The resulting solution curve captures all fanin & fanout-induced timing constraints**

## Repositioning Constraints

- **For every solution of a super-cell, a repositioning constraint is generated to exploit the remaining slack on side fanins and fanouts**



## Generalized Geometric Programming Problem

minimize $t_{cycle}$

s.t.

$$a_j \geq a_i + d_{ij}, \quad \forall SC_i \text{ and its } i^{th} \text{ fanin}$$

$$a_j \leq t_{cycle}, \quad \forall \text{primary outputs}$$

$$a_j \geq 0, \quad \forall \text{primary inputs}$$

$$|x_i - \hat{x}_i| \leq \Delta_{ix}, \quad \forall SC_i$$

$$|y_i - \hat{y}_i| \leq \Delta_{iy}, \quad \forall SC_i$$

where

$$d_{ij} = \tau_i + R_i \times (c_{net} + c_{load}) + r_{net} \times c_{load} \text{ for gates}$$

$$d_{ij} = F(C_i, R_i) + R_i \times (c_{net} + c_{load}) + r_{net} \times c_{load} \text{ for SCs}$$

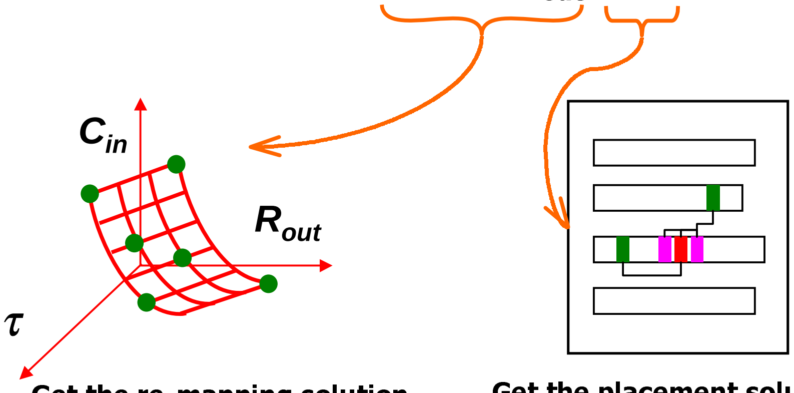
$\Delta_{ix}$ and $\Delta_{iy}$ are repositioning constraints

## GGP Solution

- Solving the GGP problem:
  - Transform the original GGP problem into a sequence of (convex) GP problems
  - The sequence of optimal solutions to the GP sequence converges to a point satisfying the Kuhn-Tucker necessary conditions for the optimality of GGP

## Solution Update

**GGP solver returns:** $\underbrace{t, C_{in}, R_{out}}, \underbrace{x, y}$



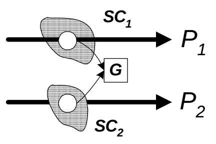
**Get the re-mapping solution**

**Get the placement solution**

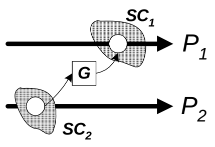
## Extensions to Multiple Critical Paths

- PILOT can handle up to *k* critical paths at the same time
- Consider two cases:
  - Non-intersecting critical paths
  - Intersecting critical paths

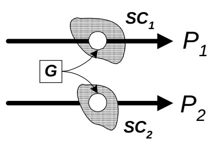
## Non-intersecting Critical Paths



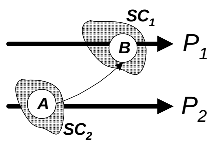
Enforce fanout-induced timing constraints



Do slack sharing for arrival-time and load of G



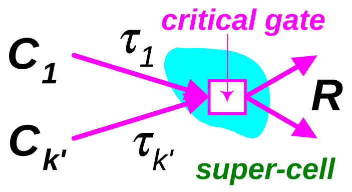
Do slack sharing between the two loads



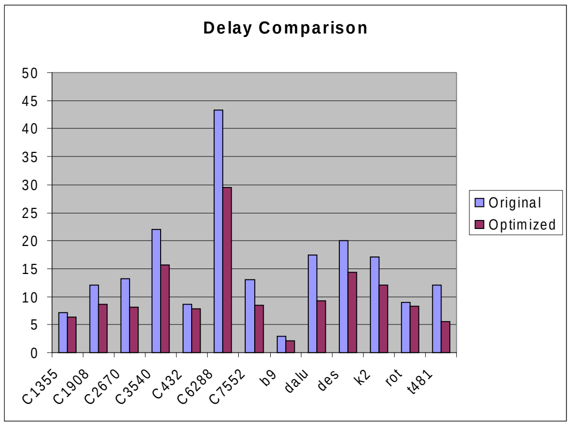
Do slack sharing for arrival-time and load between the two super-cells

## Intersecting Critical Paths

- We have to use a $2k'+1$ dimensional solution curve to model the super-cell if $k'$ critical paths intersect at this super-cell

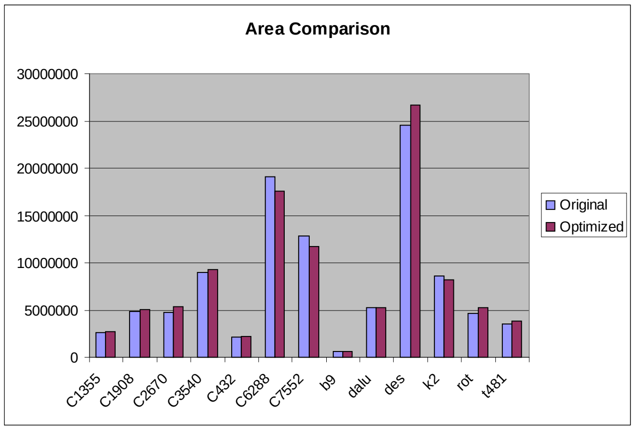


## Experimental Results



Average 29% improvement in terms of delays

## Experimental Results



**Average 4% penalty in terms of areas**

## Conclusions

- **PILOT improves timing by balancing the path delays, i.e. longer delay paths get shorter at the expense of shorter delay paths getting longer**
- **Typical 30% improvement in delay compared to the sequential map-place flow**
- **Future work will focus on doing simultaneous placement, mapping and buffer insertion**