Grid-to-Ports Clock Routing for High Performance Microprocessor Designs

Haitong Tian[#], Wai-Chung Tang[#], Evangeline F.Y. Young[#] and C.N. Sze[*]

[#] Department of Computer Science and Engineering
The Chinese University of Hong Kong

[*] IBM Austin Research Laboratory

ISPD '11, Santa Barbara , USA
March 28, 2011

Outline

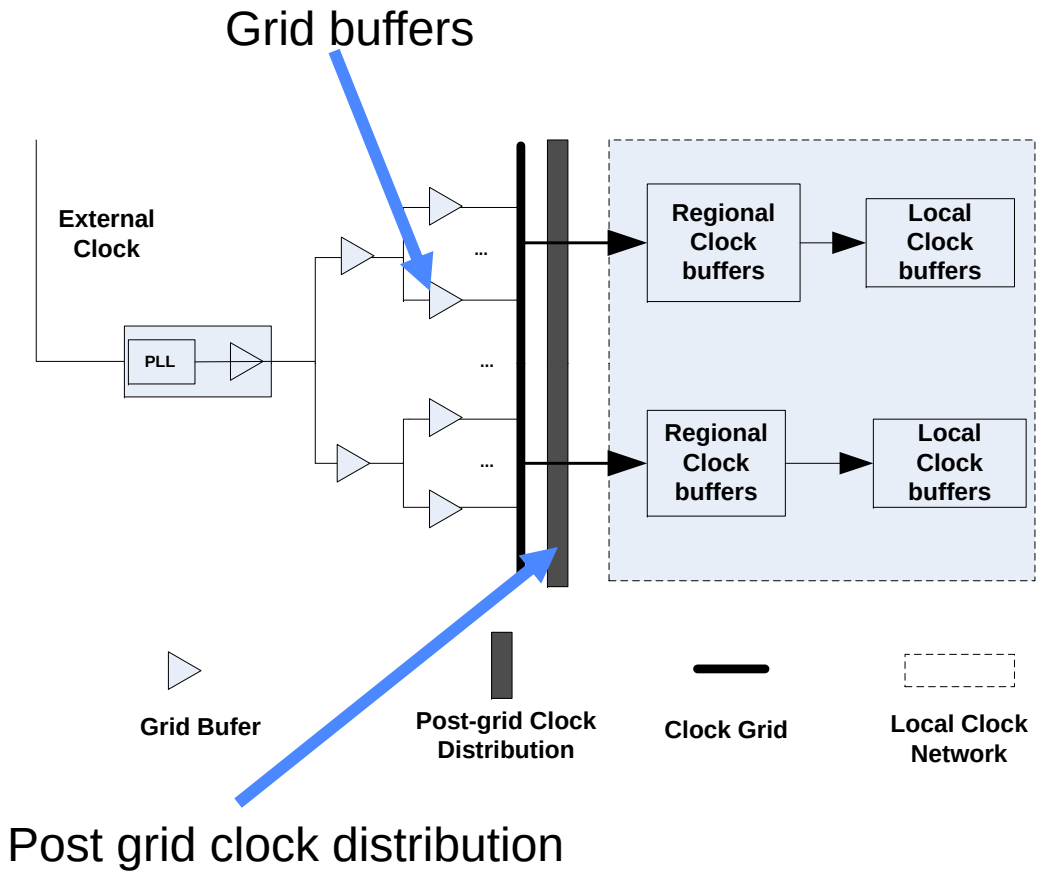
- Introduction
- Problem Formulation
- Routing Algorithm
- Experimental Results
- Conclusion

Clock Distribution Categories

- Clock distribution is an very important issue
  - Buffered and unbuffered trees
    - Used in various ASICs
    - Supported by many physical design tools
    - See Tsay TCAD'93, Xi DAC'95
  - Non-tree structure with crosslinks
    - Intended for reducing clock skews
    - See Rajaram DAC'04, TCAD'06
  - Grid and buffered trees
    - High performance processors
    - Sometimes manually design the clock structures
    - See Shelar ISPD'09, TCAD'10, Guru VLSI Circuits'10

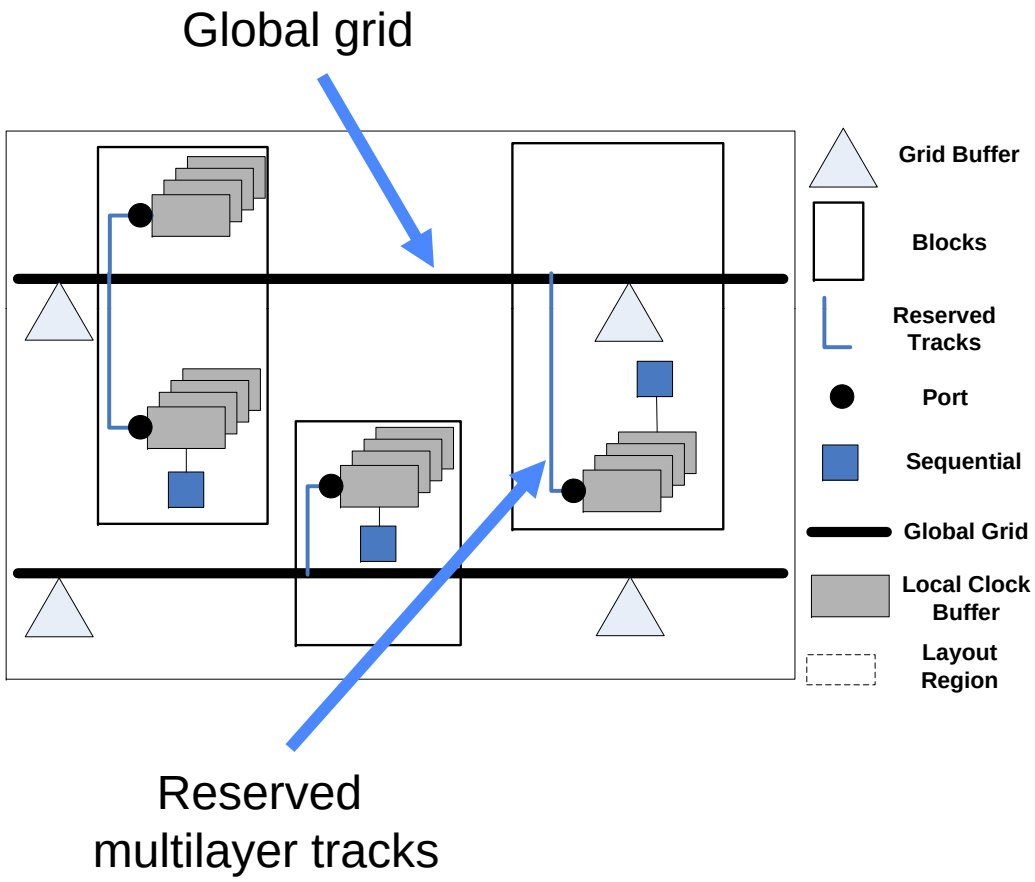
High Performance Clock Distribution

- Clock network in high performance microprocessors
  - Distributed as global grid followed by buffered trees
  - See Shelar ISPD'09, TCAD'10, Guru VLSI Circuits'10
  - This paper focuses on the post-grid clock distribution area



Post-grid Clock Distribution

- In our modeling
  - Entire chip divided into several layout areas
  - Each layout area contains many blocks
  - Each block contains standard cells and/or macros
  - Each layout area contains
    - 100s-1000s clock ports
    - Grid wires reserved for clock routing
    - Typically upper mental layers



Motivations

- Clock distribution of microprocessor:
  - Crucial importance
  - Major source of power dissipation
- High capacitance usage
  - 18.1% of total clock capacitance [1]
  - See Pham Solid State Circuits'06
- Manually design in practice
  - Hard to satisfy delay/slew constraints
  - Time to market
  - See Shelar ISPD'09, TCAD'10

[1]: D. Pham, T. Aipperspach, D. Boerstler, M. Bolliger, R. Chaudhry, D. Cox, P. Harvey, P. Harvey, H. Hofstee, C. Johns, et al. Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor. IEEE Journal of Solid-State Circuits, 41(1):179–196, Jan. 2006.

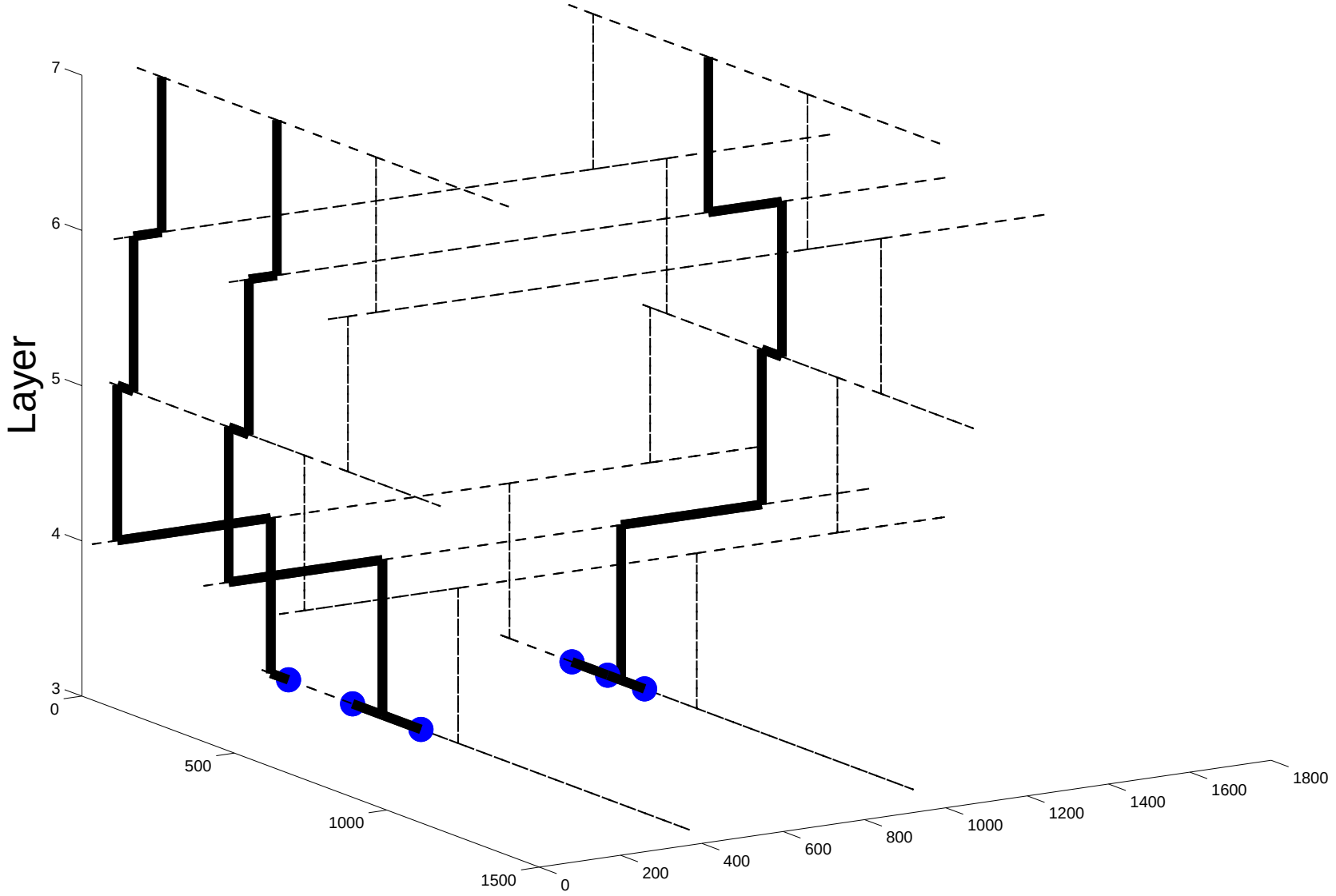
Outline

- Introduction
- Problem Formulation
- Routing Algorithm
- Experimental Results
- Conclusion

Problem Formulation

- Input
  - A set of reserved tracks
  - Locations and capacitances of ports P
  - Different types of wires on each metal layer
  - Delay limit D. Slew limit S
- Output
  - A clock network (may be non-tree structures)
- Objective
  - Connecting every port to the source
  - Satisfying delay and slew constraints
  - Minimizing capacitance usage

Post-grid Clock Routing

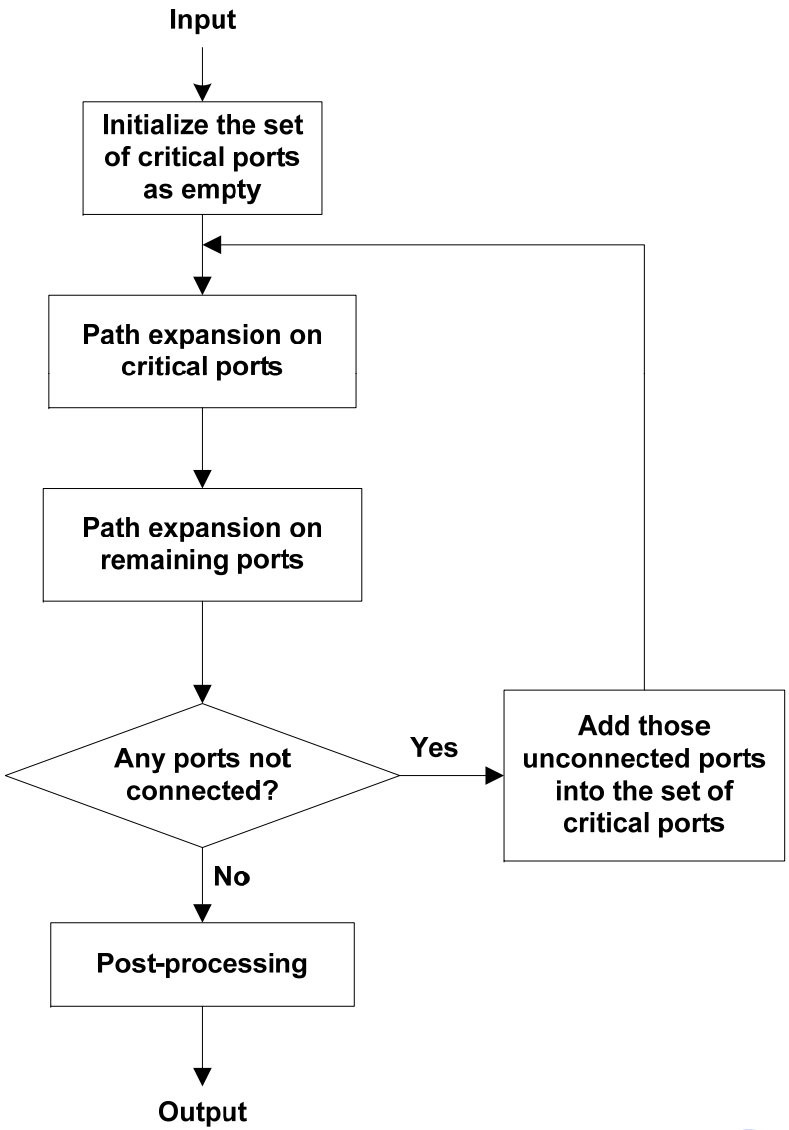


Outline

- Introduction
- Problem Formulation
- Routing Algorithm
- Experimental Results
- Conclusion

Overall Algorithm

- Critical ports
  - Ports with large capacitance or far away from the source
- Path expansion algorithm
  - Elmore-delay driven
  - Expanding in some selected directions
- Post-processing
  - Wire replacement
  - Topology refinement
- Iterations
  - The overall algorithm is repeatedly invoked
  - May fail when number of iterations > **K** (user specified)

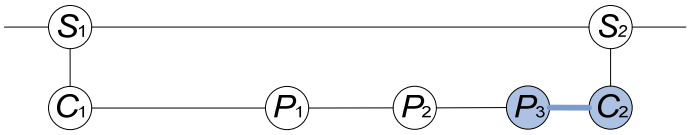


Delay-driven Path Expansion Algorithm

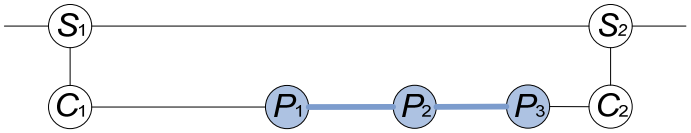
- Basic steps
  - Simultaneously expand from all ports
  - Select the path with the minimum Elmore delay to further expand
  - Connect the ports to the source once the path reaches the source grid
  - Check delay/slew constraints

A Routing Example

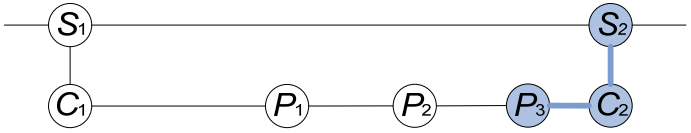
- Initially, the heap is empty
- First iteration (simultaneously expand from all ports)
  - Heap={(P_1,P_2);(P_1,C_1);(P_2,P_3);(P_2,P_1);(P_3,P_2);(P_3,C_2)}



- Second iteration (P_1,P_2)
  - Heap={(P_3,C_2);(P_1,P_2,P_3);(P_1,C_1);(P_2,P_3);(P_2,P_1);(P_3,P_2)}

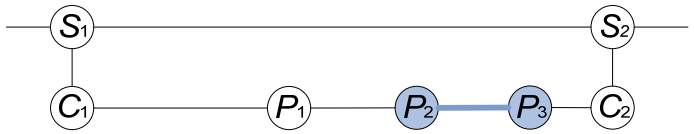


- Third iteration (P_3,C_2)
  - Heap ={(P_3,C_2,S_2);(P_1,P_2,P_3);(P_1,C_1);(P_2,P_3);(P_2,P_1);(P_3,P_2)}

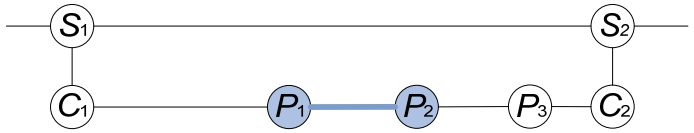


A Routing Example

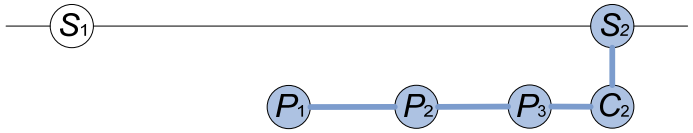
- Fourth iteration (identify chain paths)
  - Heap ={(P_1,P_2,P_3);(P_1,C_1);(P_2,P_3);(P_2,P_1)}
  - Chain path={(P_1,P_2,P_3);(P_2,P_3)}



- Fifth iteration (P_2,P_3)
  - Heap={(P_1,P_2,P_3);(P_1,C_1)}
  - Chain path={(P_1,P_2,P_3);(P_1,P_2)}



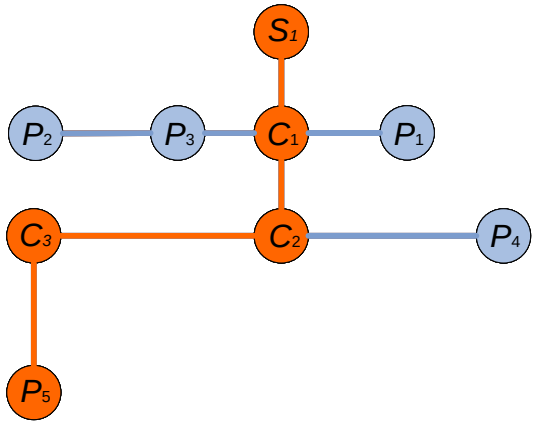
- Sixth iteration (P_1,P_2)
  - Heap={}, chain path={}
  - Final result



Post-processing Techniques

- Wire replacement
  - Two types of wires
    - capacitance/resistance tradeoff
  - Procedures
    - Identify port P_l with the largest Elmore delay
    - Replace wires in a bottom-up style
    - Check delay/slew constrains

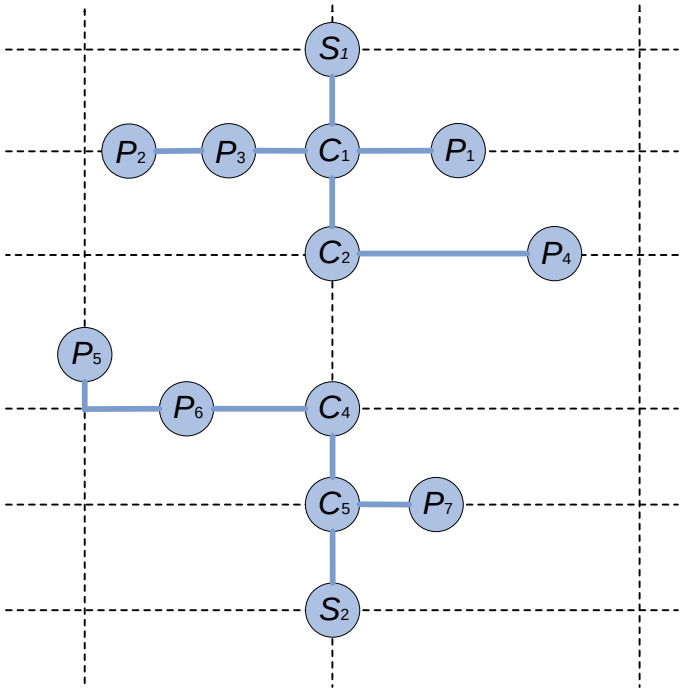
- Wire replacement
  - Port with largest delay: P_5
  - Replace edge P_1C_1
  - Replace edge P_4C_2
  - Replace edge P_2P_3, P_3C_1
  - Replace P_5C_3, C_3C_2, C_2C_1, C_1S_1



Post-processing Techniques

- Topology refinement
  - Procedures
    - Disconnect a port P
    - Expand P towards all directions
    - Select paths with smaller capacitance
    - Check delay/slew constraints

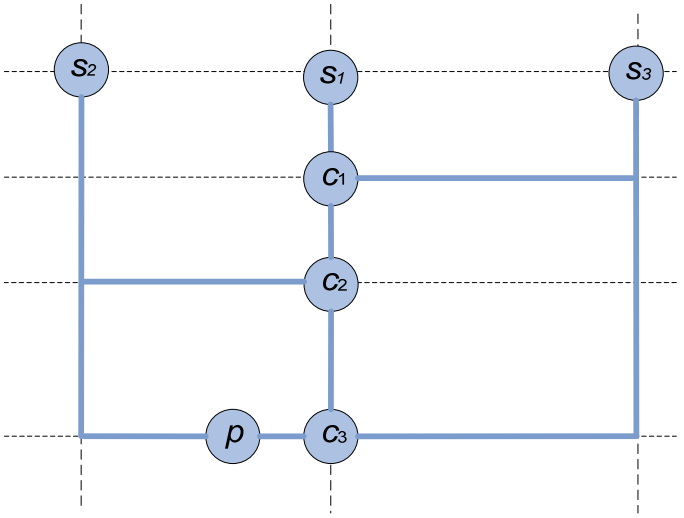
- Topology refinement
  - Elmore delay:
    - $P_5 > P_4 > P_6 > P_2 > P_1 > P_3 > P_7$
  - Sequentially process all the ports



Non-tree Extensions

- A small number of ports have exceptionally large capacitances
  - The delay of its shortest path exceeds the delay limit ***D***
  - Procedures
    - Establish a shortest path for p
    - Find a second shortest path
    - Target delay not met? Add all useful corsslinks
    - Target delay not met? Do the same thing for parent node of p

- Non-tree extensions
  - Connect p to S_1
  - Find a second source S_2
  - Add crosslinks
  - Find a third source S_3
  - Add crosslinks



Outline

- Introduction
- Problem Formulation
- Routing Algorithm
- Experimental Results
- Conclusion

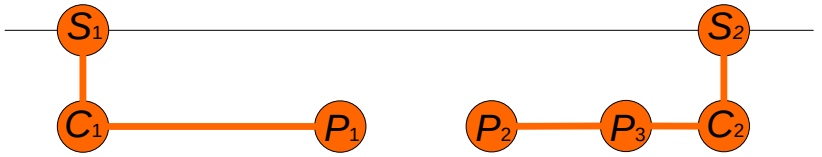
Experiment Setup

- Environment
  - Implemented in C++
  - Run on Linux server
    - Intel Pentium 4 3.2GHz
    - 2GB RAM
  - Delay setup: 5ps
  - Slew setup: input: 10ps; output: 15 ps
- Benchmarks
  - 3 test cases are provided by industry
  - 11 test cases are from ISPD 2010 Clock Network Synthesis Contest
- Comparisons
  - Compared with TG, which was proposed by Shelar in ISPD'09, TCAD'10

Tree Growing Algorithm

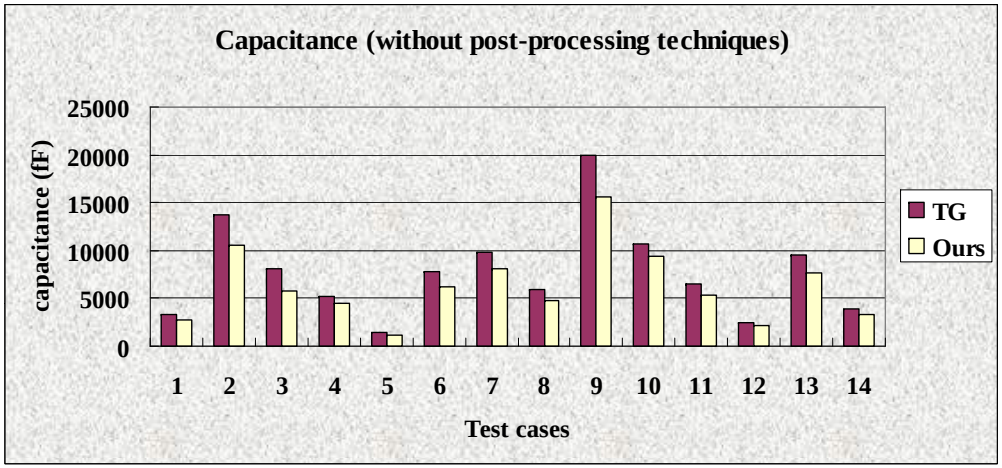
- Proposed in R. Shelar ISPD'09, TCAD'10
  - Delay/Slew constraints
  - Greedy expansion from the source
  - Edges with the smallest capacitance will be added into the network

- Tree Growing Algorithm
  - Expand from the source
  - Add S_1C_1, S_2C_2
  - Add C_2P_3
  - Add C_1P_1
  - Add P_3P_2

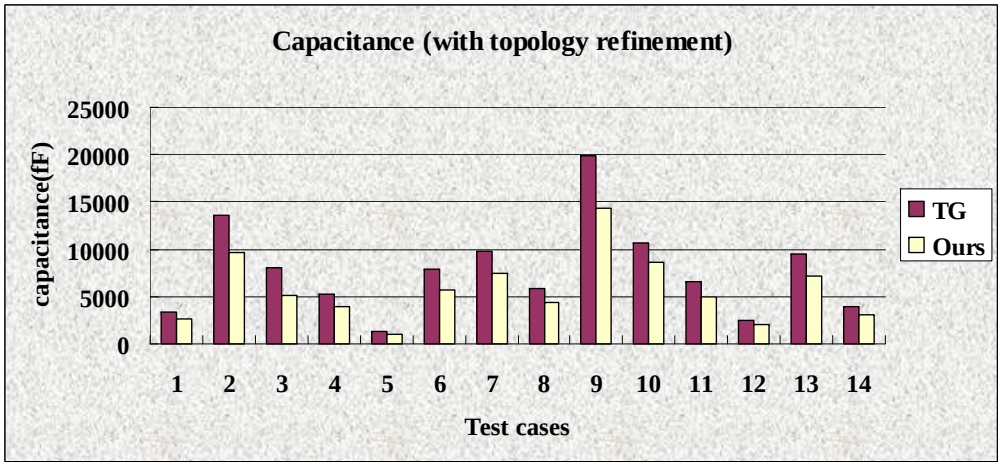


Comparisons: capacitance

- Without post-processing: 18.3% improvement

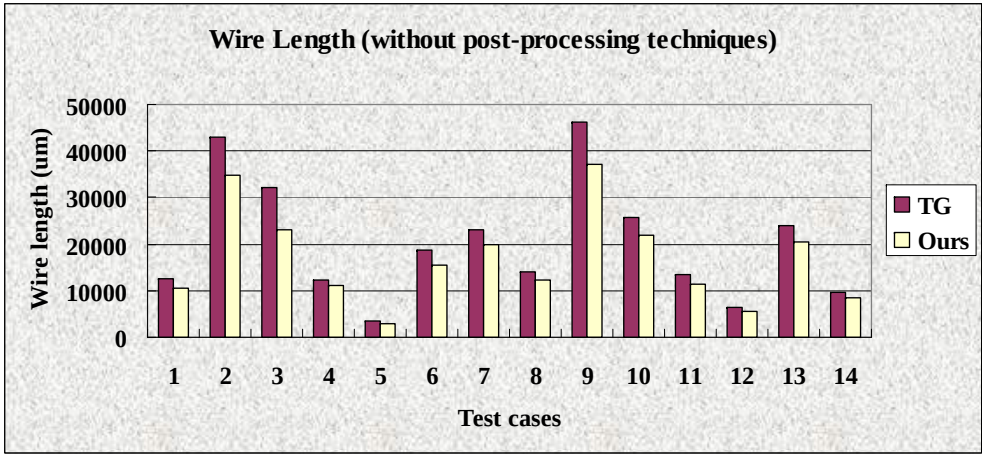


- With topology refinement: 24.6% improvement

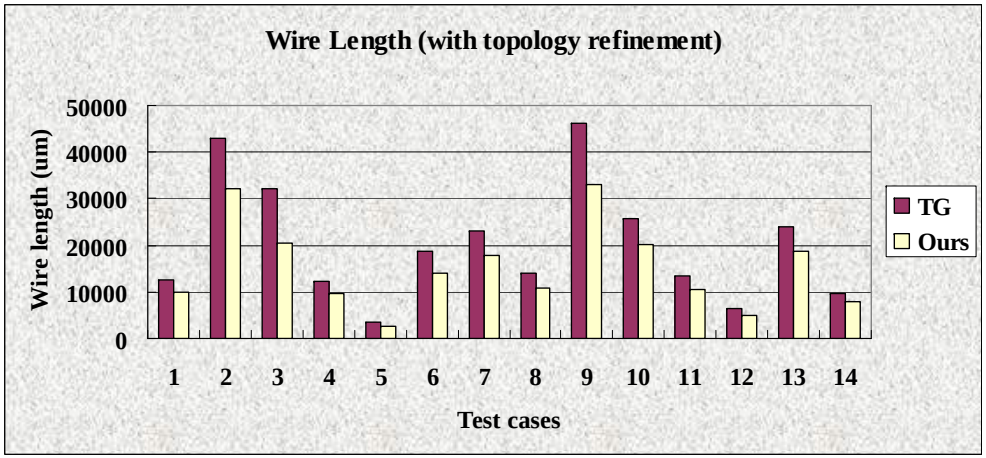


Comparisons: wire length

- Without post-processing: 16.8% improvement

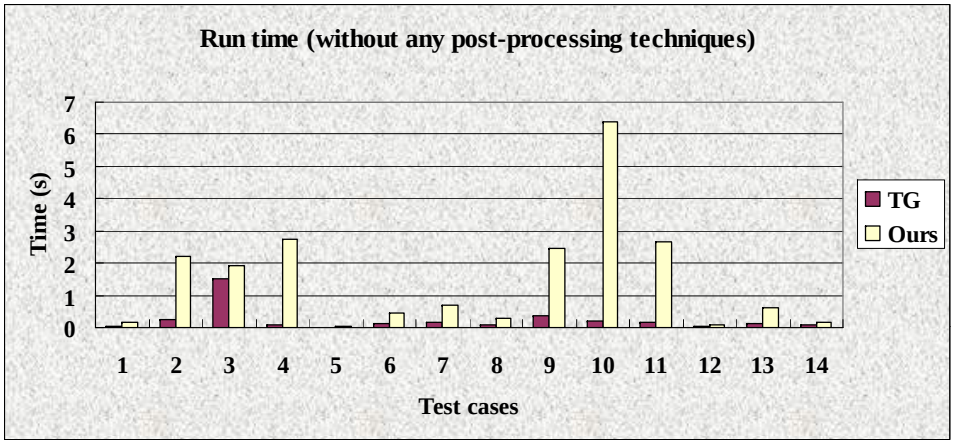


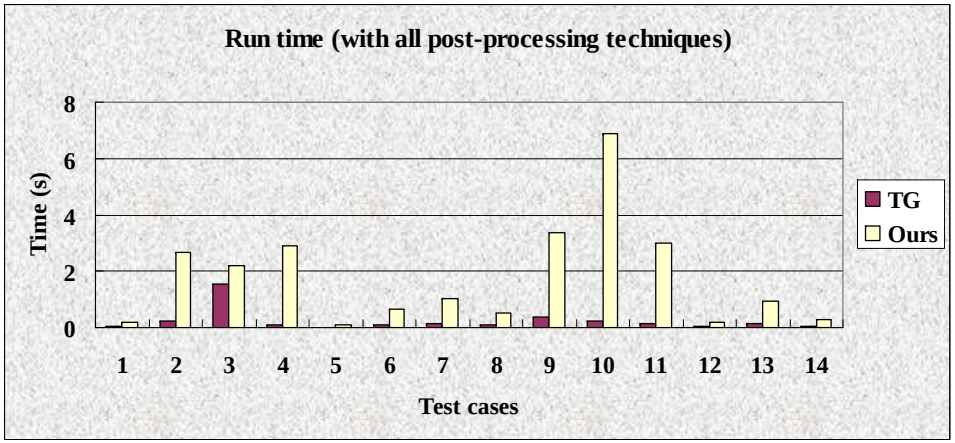
- With topology refinement: 23.6% improvement



Comparisons: run time

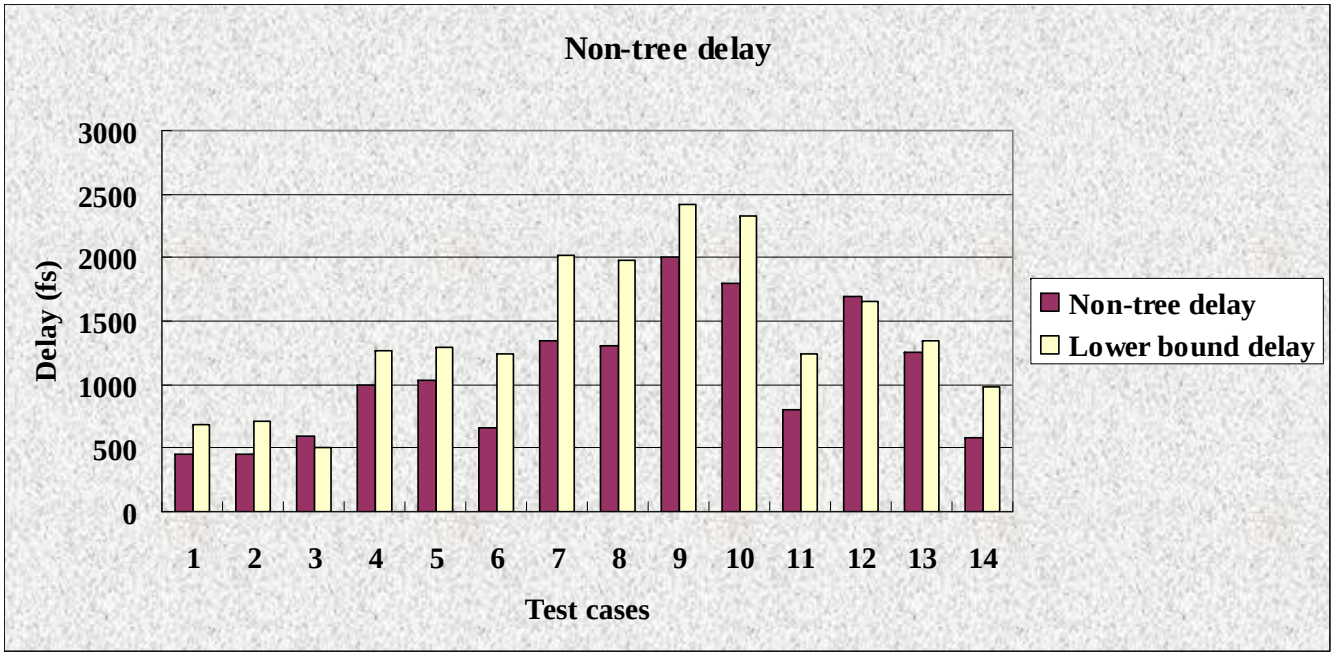
- TG
  - Average run time 0.14s
- Ours
  - Without any post-processing techniques: 1.49s on average
  - With all techniques: 1.78s on average
  - Practical runtime





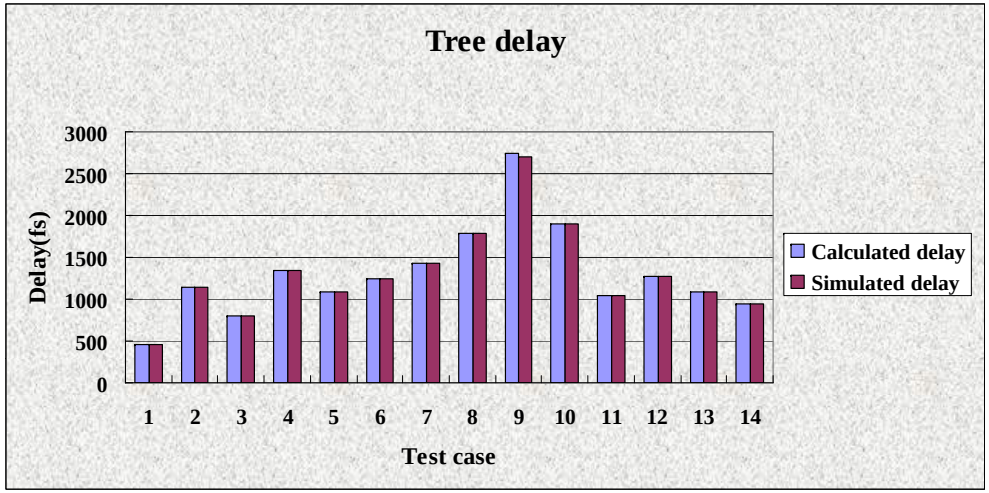
Non-tree extension

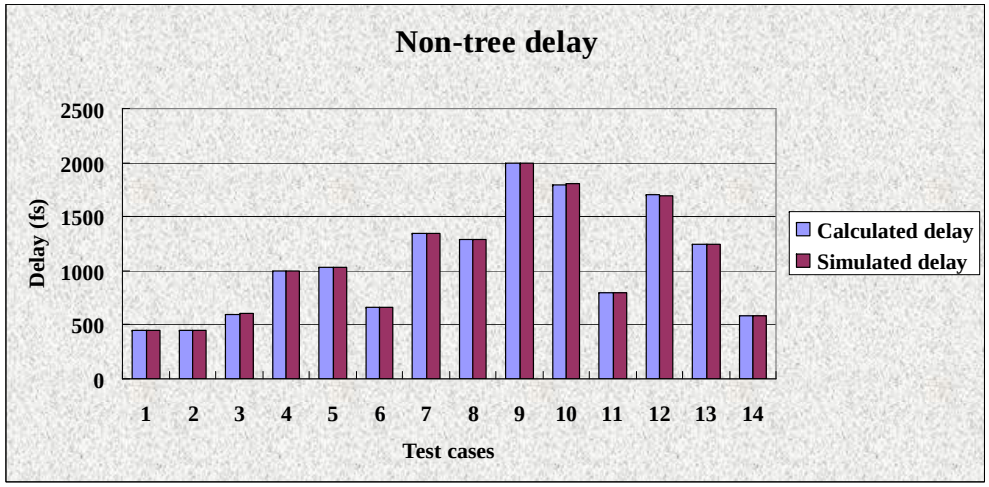
- We also did some experiments to see the results of our non-tree extension
  - A 23.4% improvement on the minimum port delay
  - Minimum port delay is the lower bound one can achieve using tree structures



Simulation Results

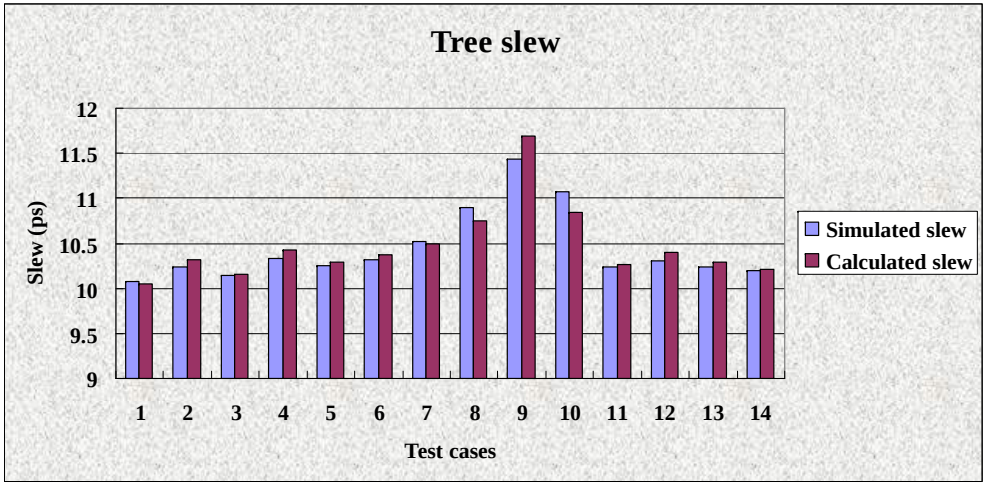
- Simulation tool: Hspice
- Delay
  - Correlation coefficient
    - Tree: 99%
    - Non-tree: 99%

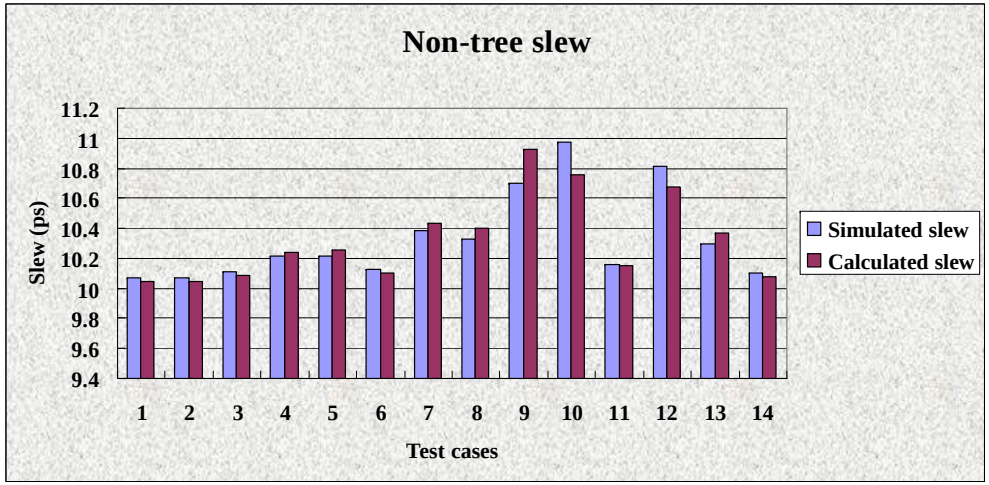




Simulation Results

- Slew
  - Correlation coefficient
    - Tree: 96%
    - Non-tree: 94%





Outline

- Introduction
- Problem Formulation
- Routing Algorithm
- Experimental Results
- Conclusion

Conclusion

- Proposed an efficient algorithm to construct a post-grid clock network on reserved multi-layer metal tracks
- Extended the algorithm to allow non-tree structures to further brings down the delay
- Verified our results using Hspice simulation
- Expected to reduce energy consumption, improve grid to port delay in real post-grid clock networks

Thank You!