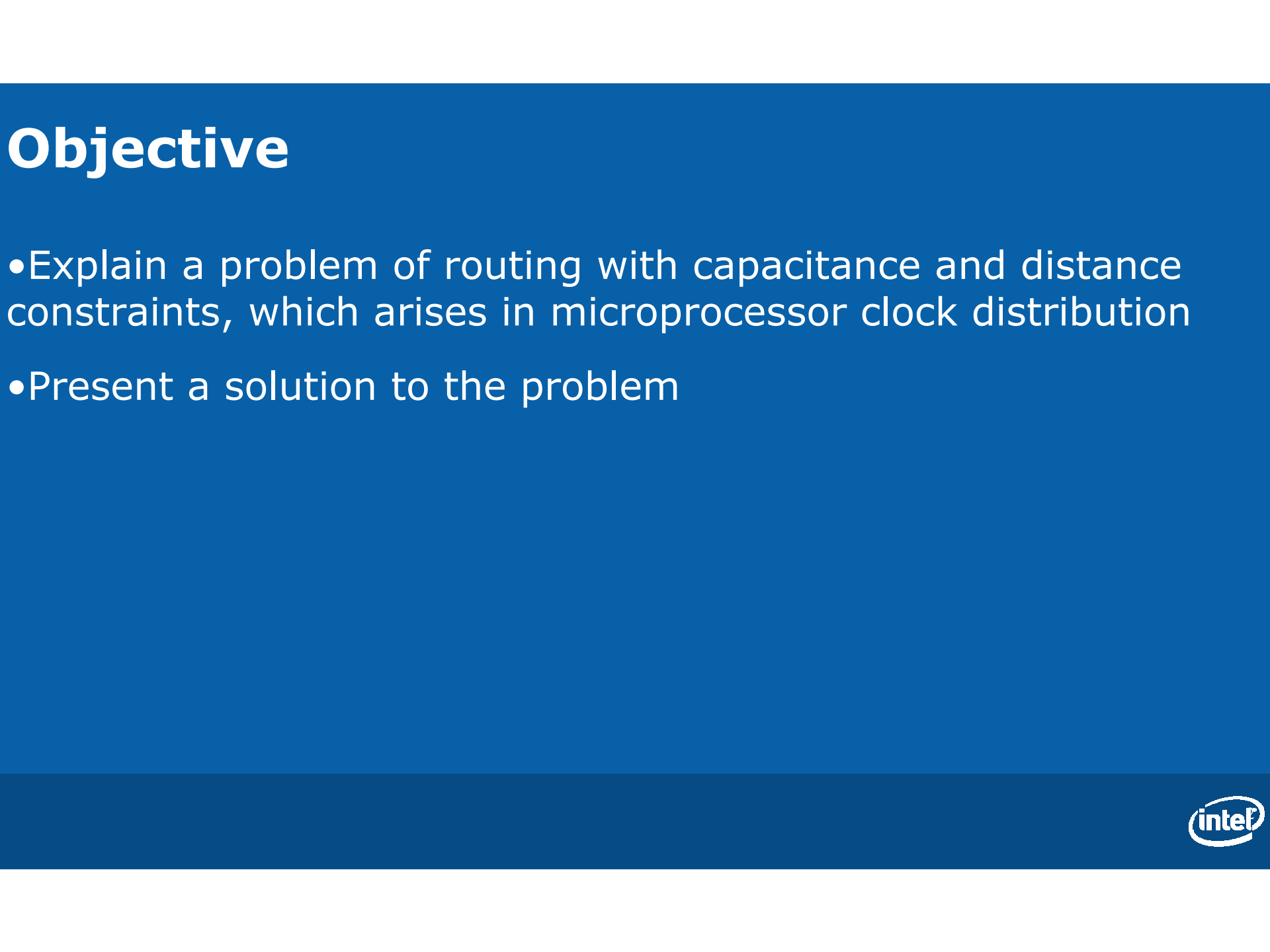
An Algorithm for Routing With Capacitance/Distance Constraints for Clock Distribution in Microprocessors

Rupesh S. Shelar
Technology & Manufacturing Group
Intel Corporation, Hillsboro, OR

March 31st 2009,
ISPD 2009, San Diego

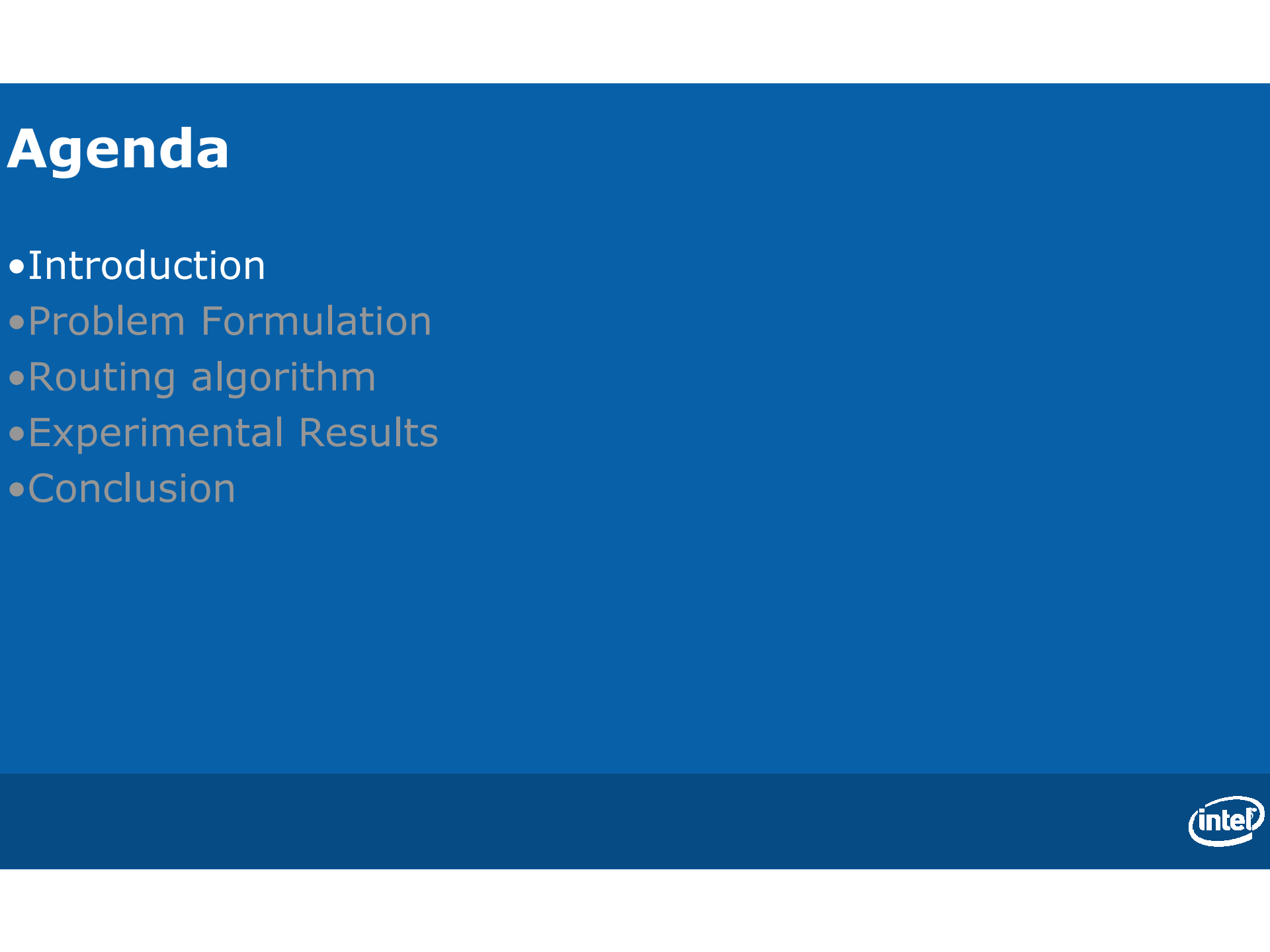
Objective

- Explain a problem of routing with capacitance and distance constraints, which arises in microprocessor clock distribution
- Present a solution to the problem



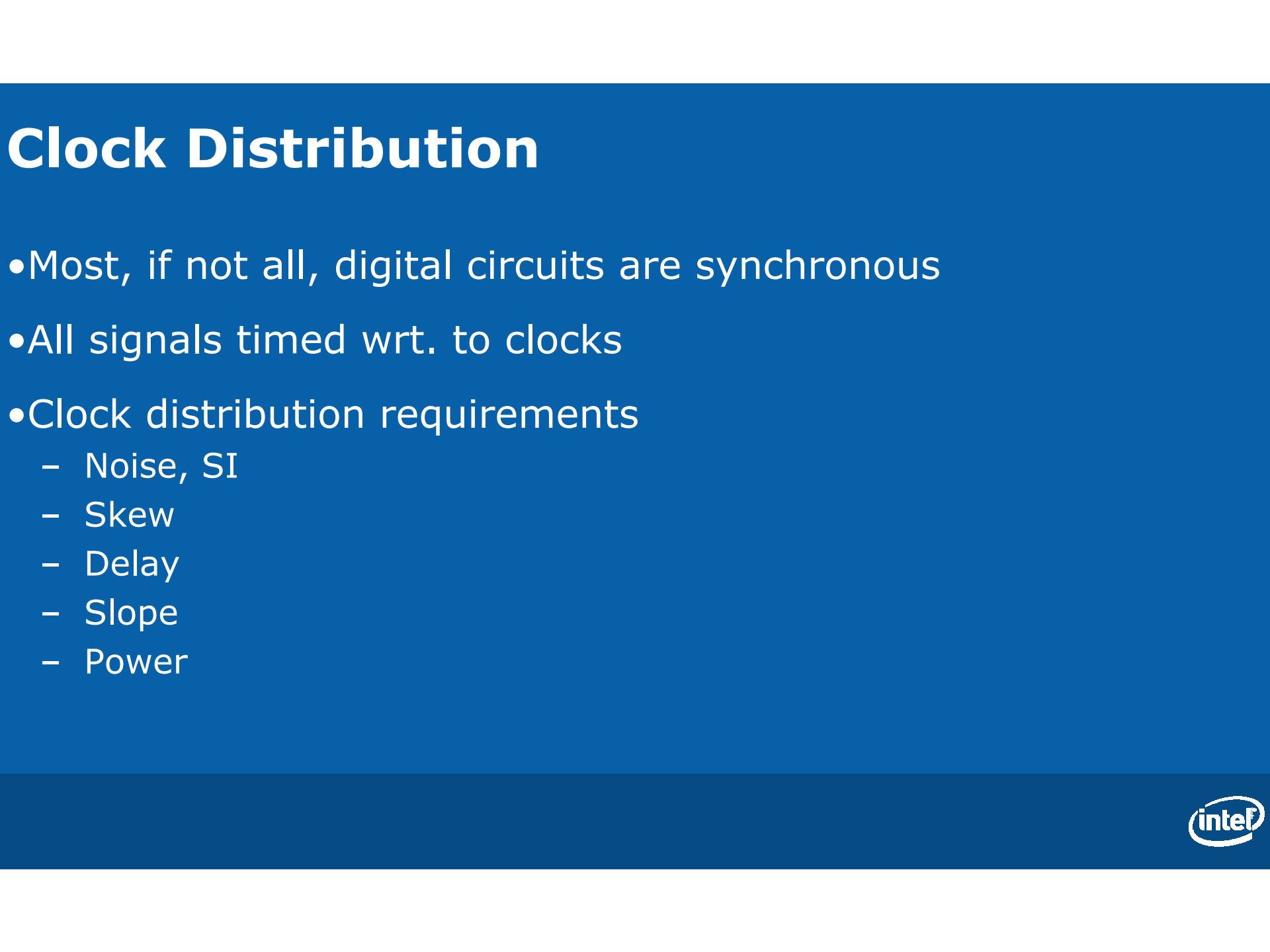
Agenda

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



Clock Distribution

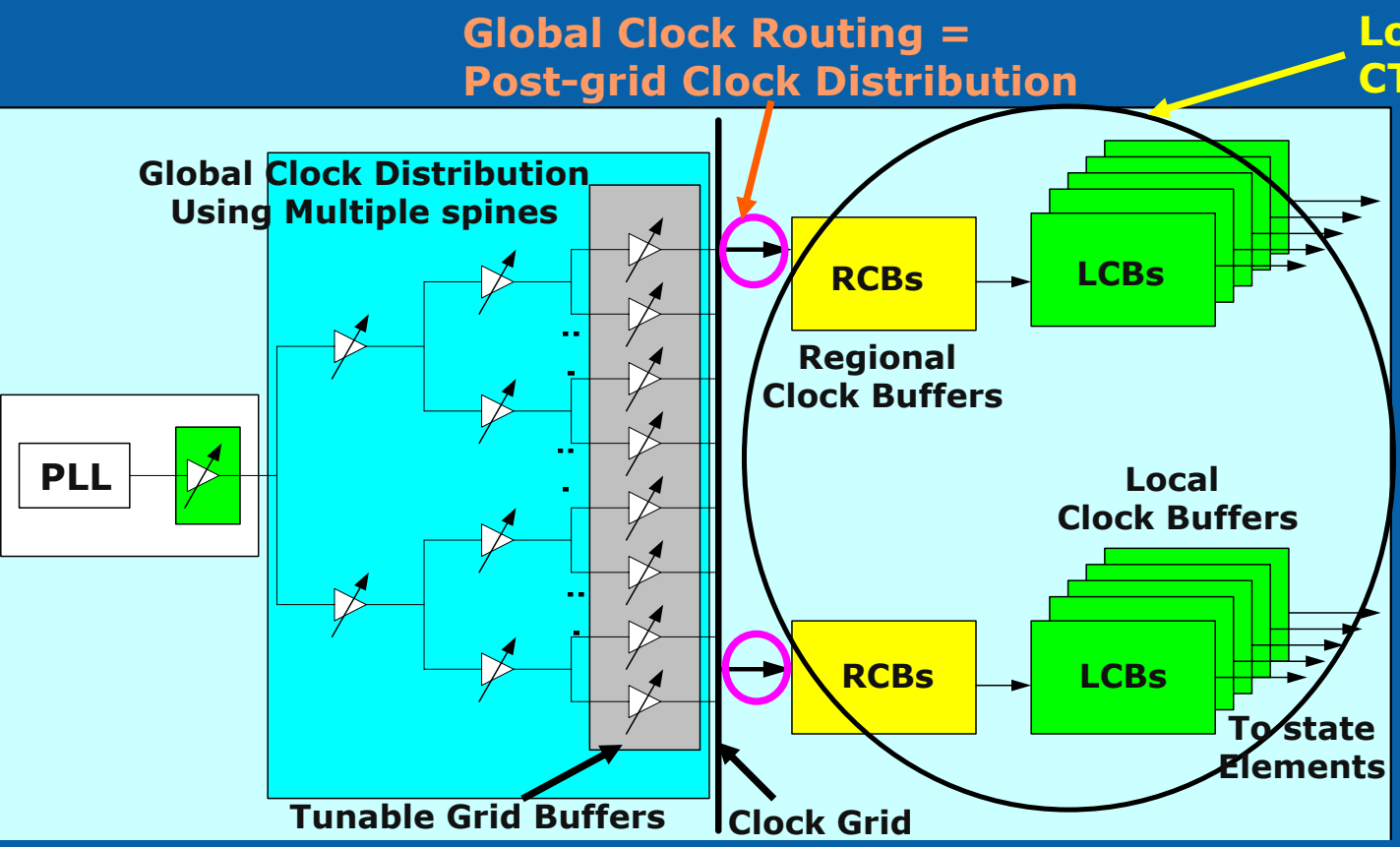
- Most, if not all, digital circuits are synchronous
- All signals timed wrt. to clocks
- Clock distribution requirements
  - Noise, SI
  - Skew
  - Delay
  - Slope
  - Power



Clock Network Classification

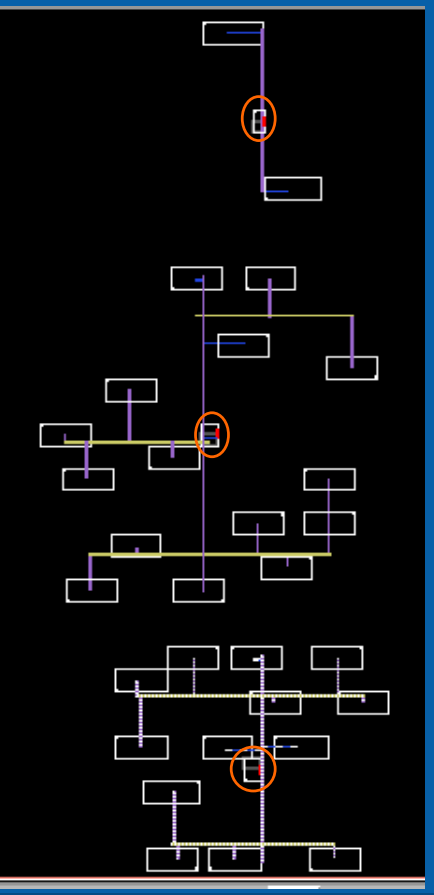
- Can be classified depending on underlying structure
  - Grid + buffered trees
    - High performance (GHz) processors; less skew possibly at the cost of power
    - Relatively less automation; most of the design is manual
    - See Bailey *et al.*, JSSC'98; Kurd *et al.* JSSC'01
  - Buffered trees
    - Most ASICs (~100s of MHz)
    - Supported by most modern physical design tools
    - See Vittal *et al.* DAC'95, and many others…, Mehta *et al.* ICCD'97, Shelar ISPD'07, Maβberg *et al.* JA'08
  - Unbuffered trees
    - Local distribution in ASICs/processors, using zero-skew routing, for example
    - See Tsay ICCAD'93; Edahiro DAC'94
  - Link-inserted (buffered) clock trees
    - See Rajaram *et al.* DAC'04 and many others…

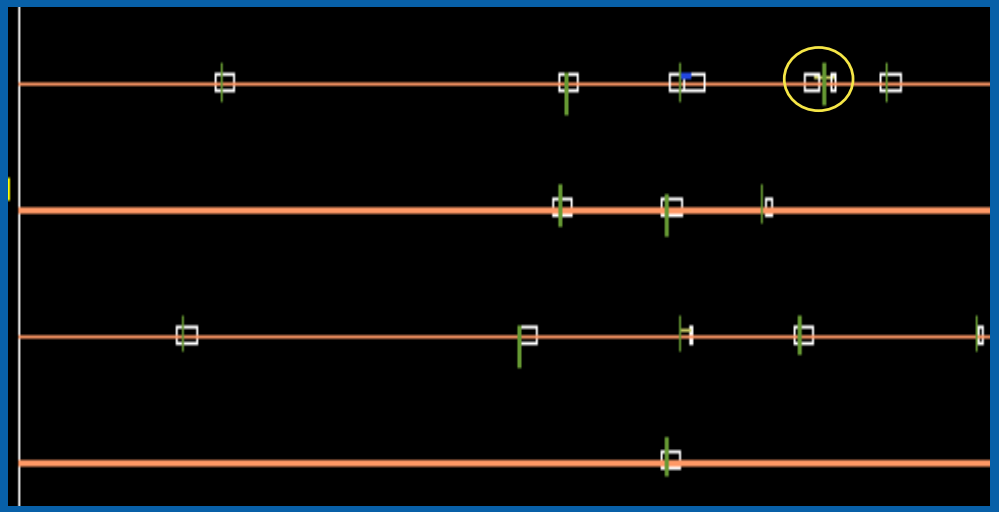
Microprocessor Clock Hierarchy

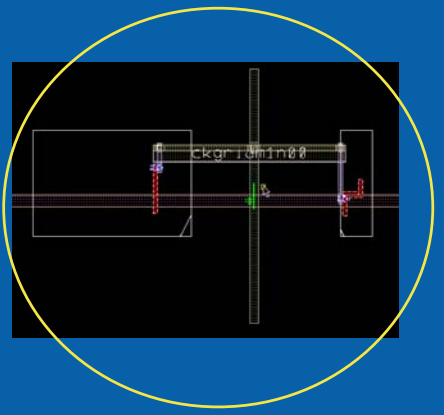


- Clock network in most high speed processors:
  - Distributed as a grid followed by trees

Block-level Clock Layout



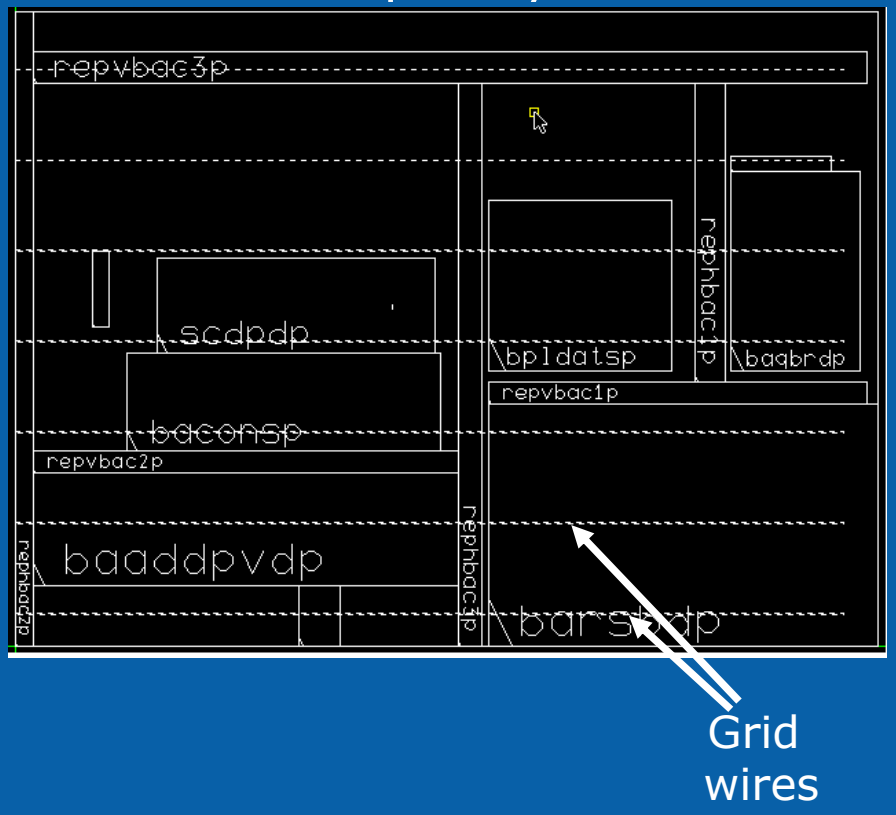




- Replicate, place, size clock cells and route clock wires with shielding/spacing
- Create block-level ports, aligned with tracks reserved for global clock distribution
- Capacitance/delay limits on ports

Microprocessor Layout Hierarchy

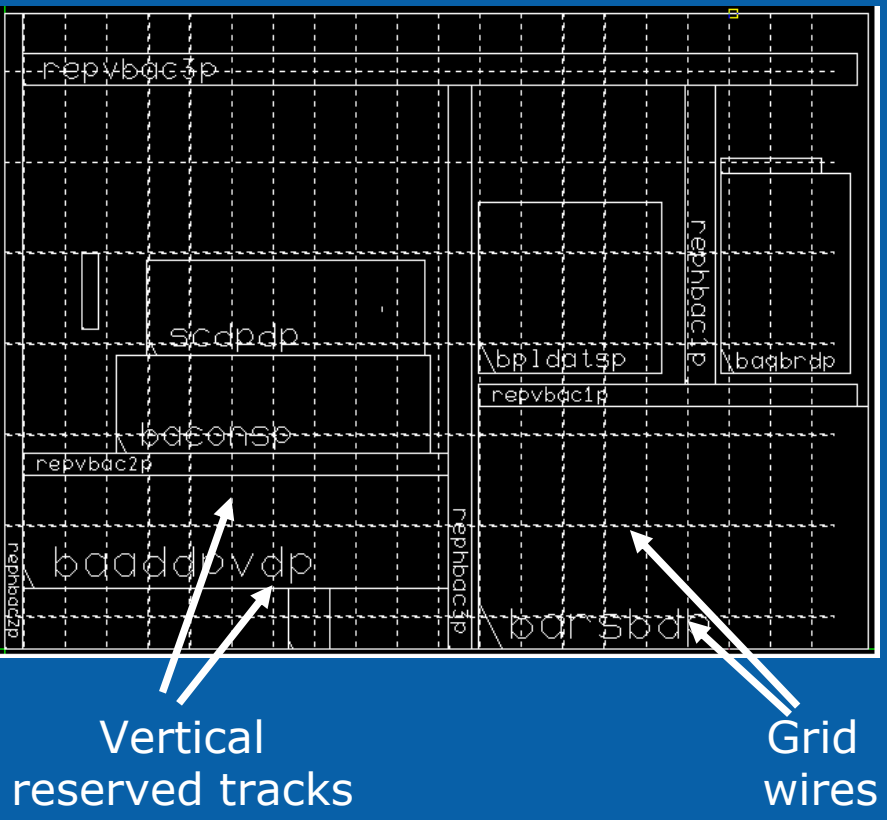
An example layout area



- Entire die divided into several layout areas
- Each layout area contains many blocks
- Each block contains std. cells or macros
- Each layout area contains
  - 100s to 1000s block-level clock ports
  - Grid-wires and tracks reserved for clock routes, typically in upper metal layers

Post-grid Global Clock Distribution

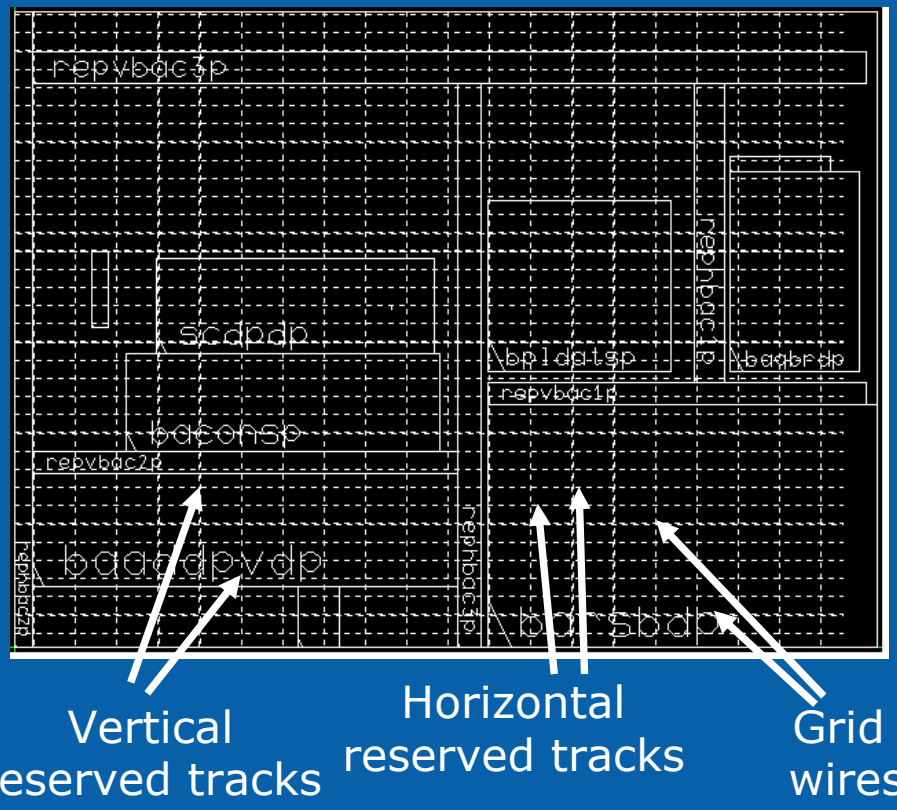
An example layout area



- Entire die divided into several layout areas
- Each layout area contains many blocks
- Each block contains std. cells or macros
- Each layout area contains
  - 100s to 1000s block-level clock ports
  - Grid-wires and tracks reserved for clo routes, typically in upper metal layers

Post-grid Global Clock Distribution

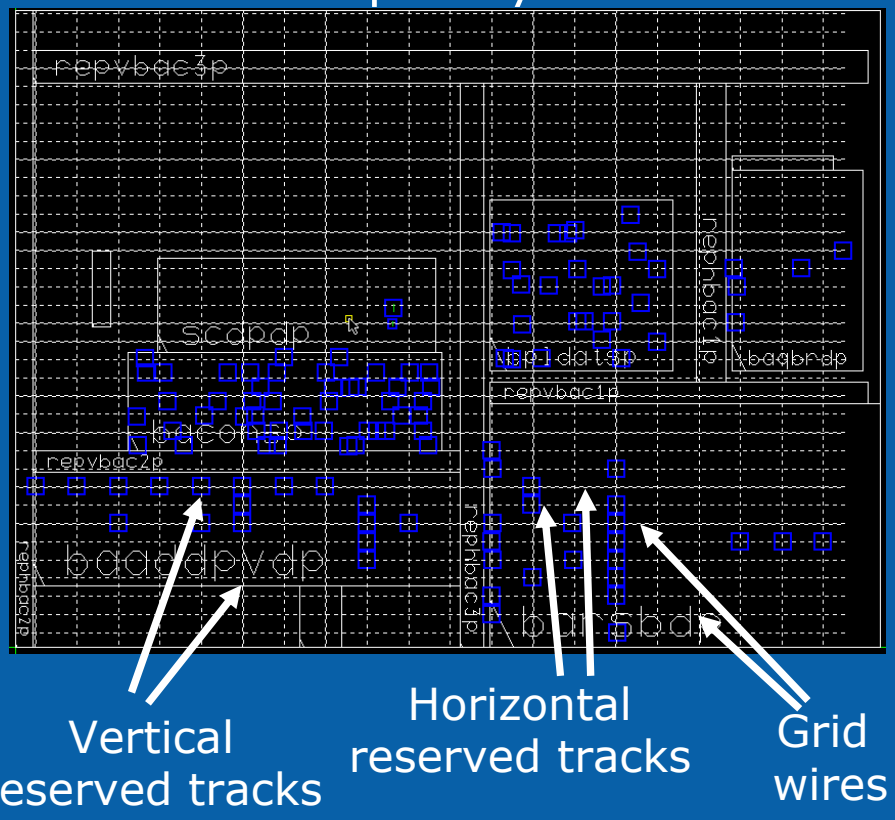
An example layout area



- Entire die divided into several layout areas
- Each layout area contains many blocks
- Each block contains std. cells or macros
- Each layout area contains
  - 100s to 1000s block-level clock ports
  - Grid-wires and tracks reserved for clo routes, typically in upper metal layers

Post-grid Global Clock Distribution

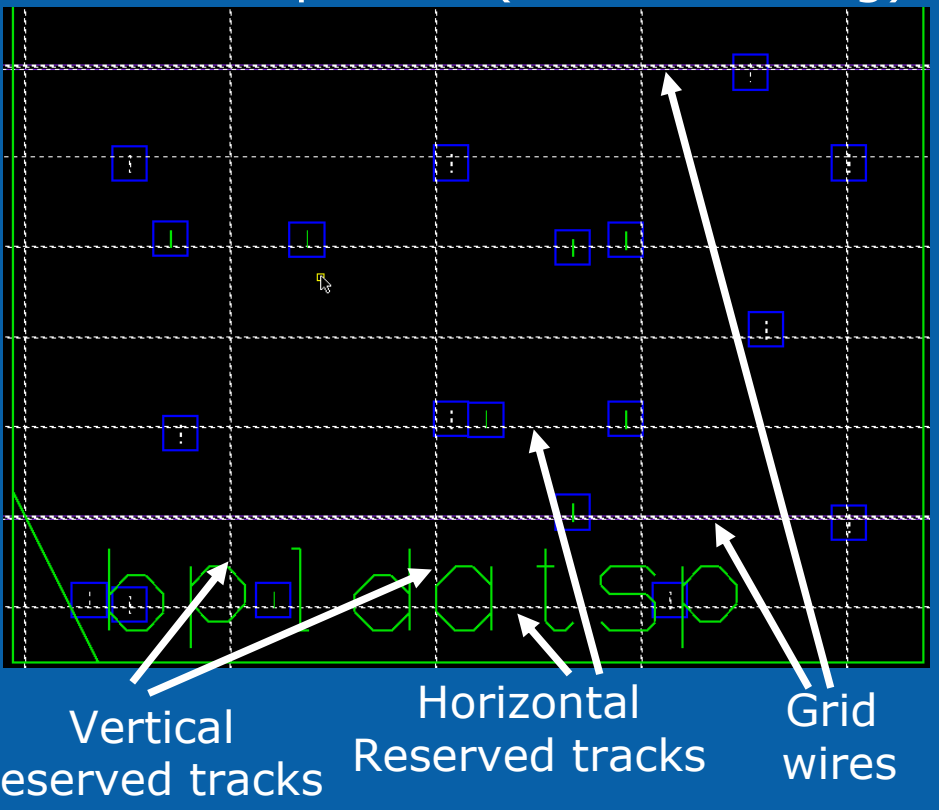
An example layout area



- Entire die divided into several layout areas
- Each layout area contains many blocks
- Each block contains std. cells or macros
- Each layout area contains
  - 100s to 1000s block-level clock ports
  - Grid-wires and tracks reserved for clock routes, typically in upper metal layers
  - Ports marked by blue squares

Post-grid Global Clock Distribution

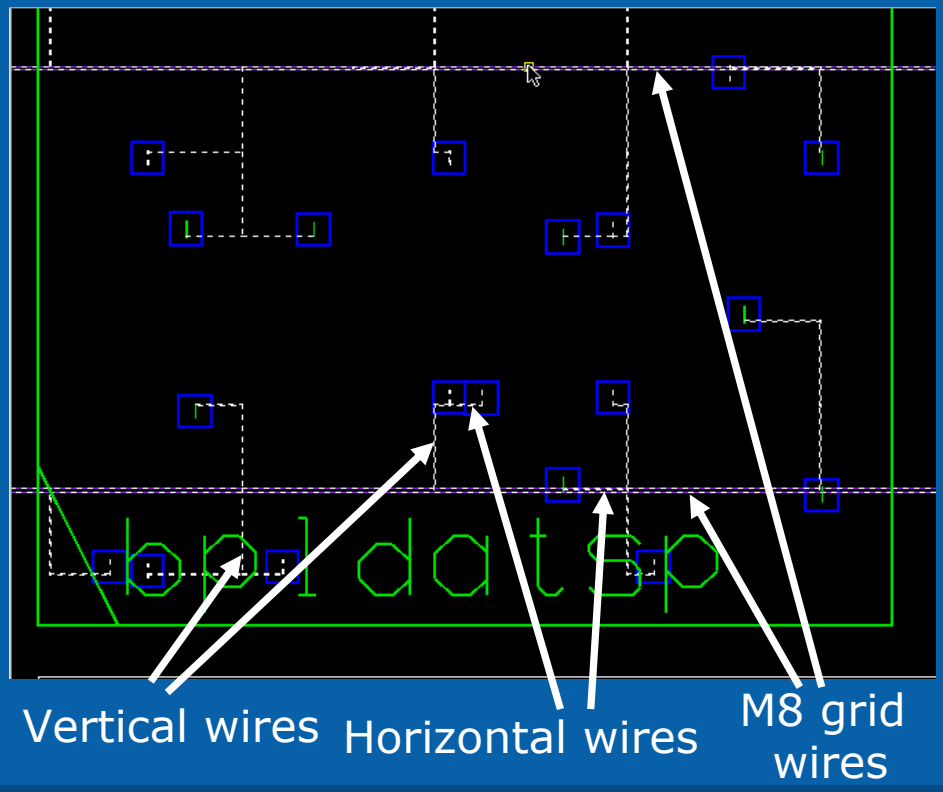
Zoomed in picture (before-routing)



- Entire die divided into several layou areas
- Each layout area contains many blocks
- Each block contains std. cells or macros
- Each layout area contains
  - 100s to 1000s block-level clock ports
  - Grid-wires and tracks reserved for clo routes, typically in upper metal layers
  - Ports marked by blue squares

Post-grid Clock Distribution

Zoomed in picture (post-routing)



- Entire die divided into several layou areas
- Each area contains many blocks
- Each block contains std. cells or macros
- Each layout area contains 100s to 1000s block-level clock ports
- Contains grid-wires and tracks reserved for clock routes, typically in upper metal layers
  - Ports marked by blue squares

Motivation for Fast Post-Grid Global Clock Distribution

- Global clock wires contribute significant load on the clock grid
- Without accurate (estimated/extracted) global clock wire load, simulations yield inaccurate arrival times at block-level clock pins
  - Affects timing convergence: path reordering due to actual arrival times
- Design space constrained by clock as well, and not just timing, area, power, …
  - If loads are too high, the clock may not toggle
  - Poor slopes at block-level ports may affect the chip-frequency
  - Block-level timing convergence does affect the load on grid
    - Sizing of sequentials, placing them in one area, splitting clock gating latches for timing
    - Difficult to capture using block-level metric, since the load depends on other blocks as well

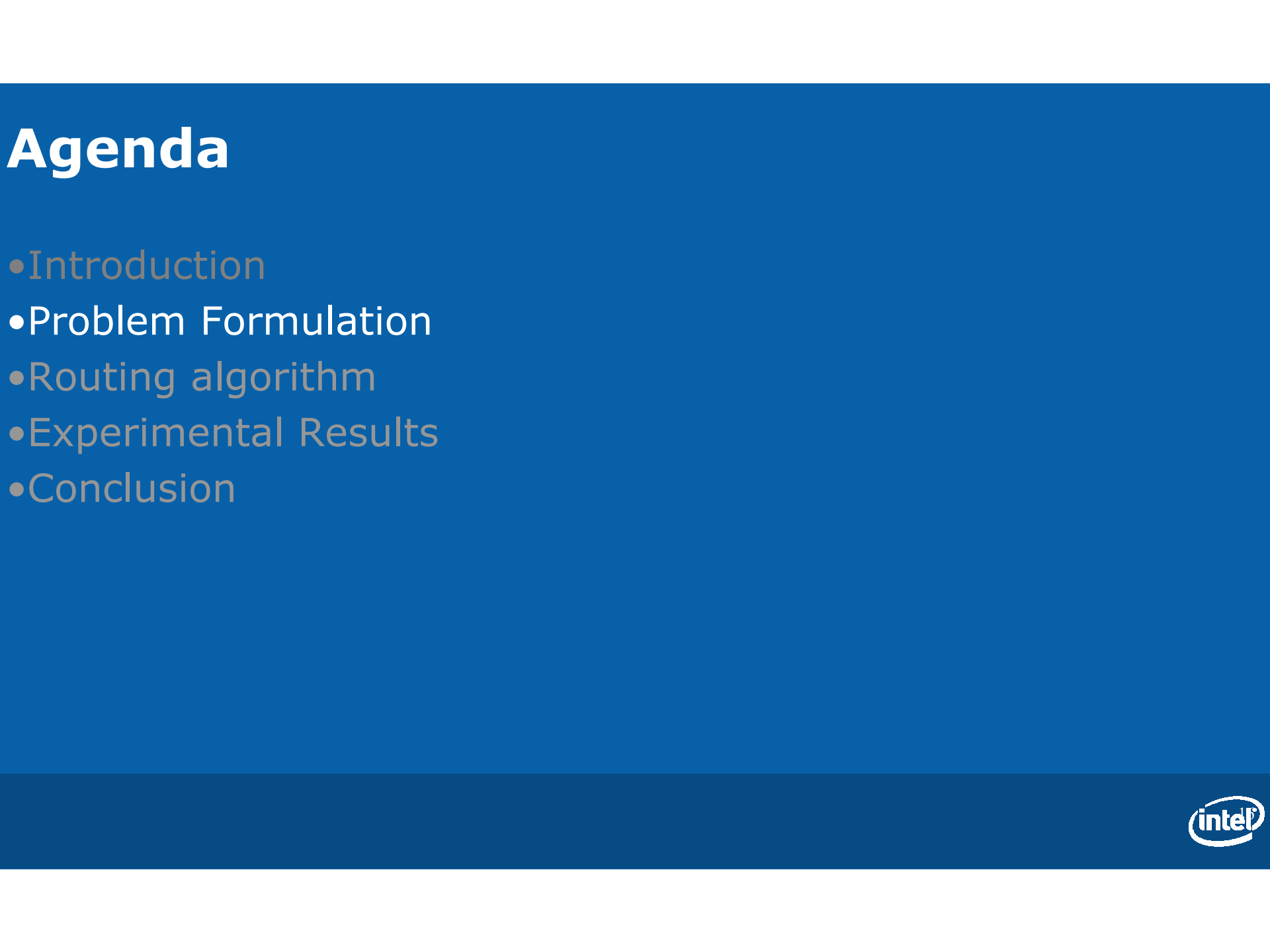
Previous work: Post-grid Clock Distribution

- No published work, possibly, since too specific a problem, limited to high-performance processors

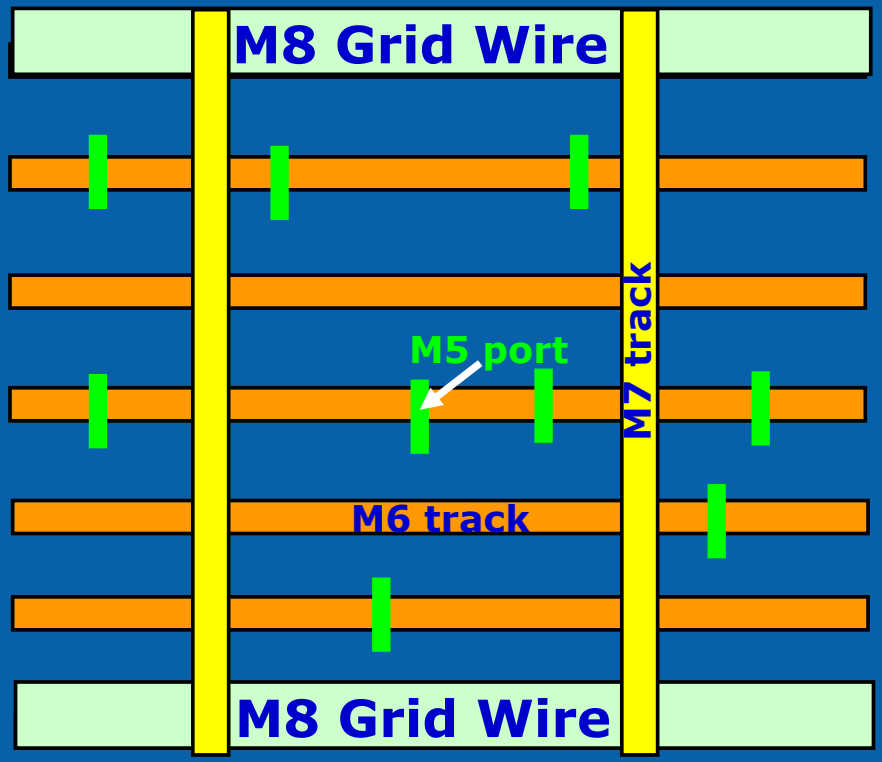
- In practice (it was/is), …
  - Mostly manual, using stable block-level data
    - Employing the nearest source heuristic
    - May not be the best even from total cap. perspective
    - May violate distance/capacitance constraints, leading to slope violations
    - May lead to violation of load limit on grid-wires
  - May have to be performed iteratively
    - Partly, since with the nearest source heuristic, capacitances are ignored
    - If there are slope violations on the receivers or grid-loading issues or block-level ECOs
    - Weeks of effort during critical tape-in period
    - Affects timing convergence and schedule/time to market

Agenda

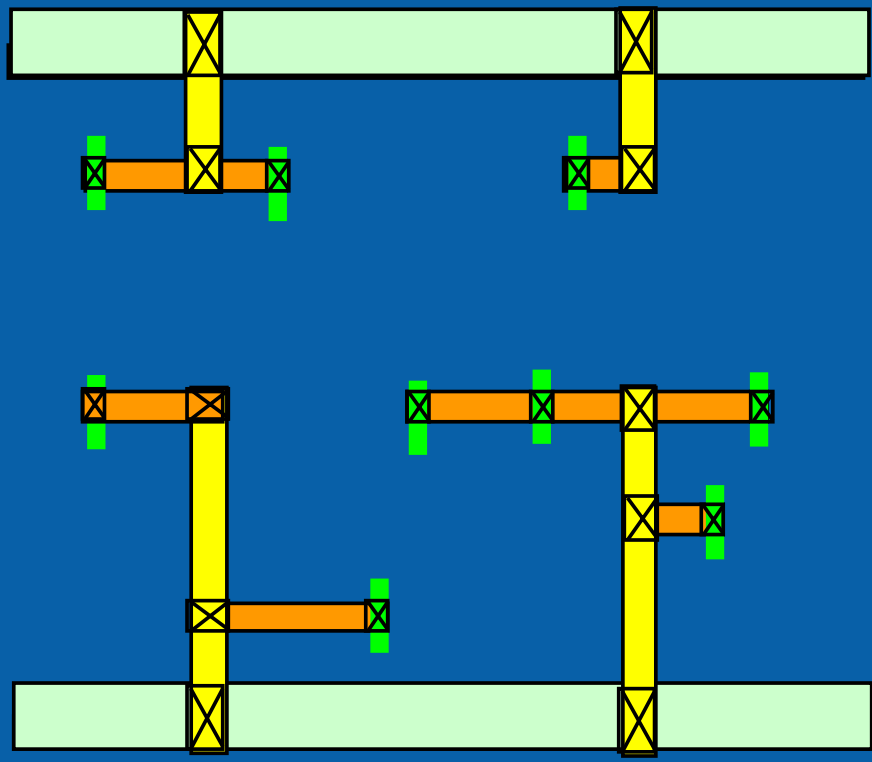
- Introduction
- Problem Formulation
- Routing algorithm
- Experimental Results
- Conclusion



Example

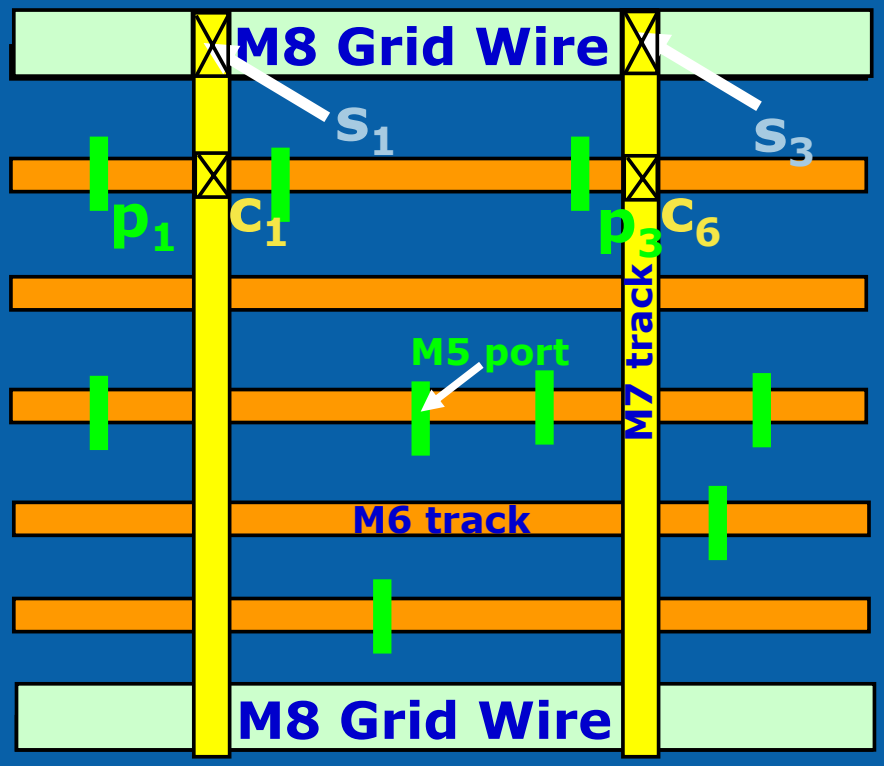


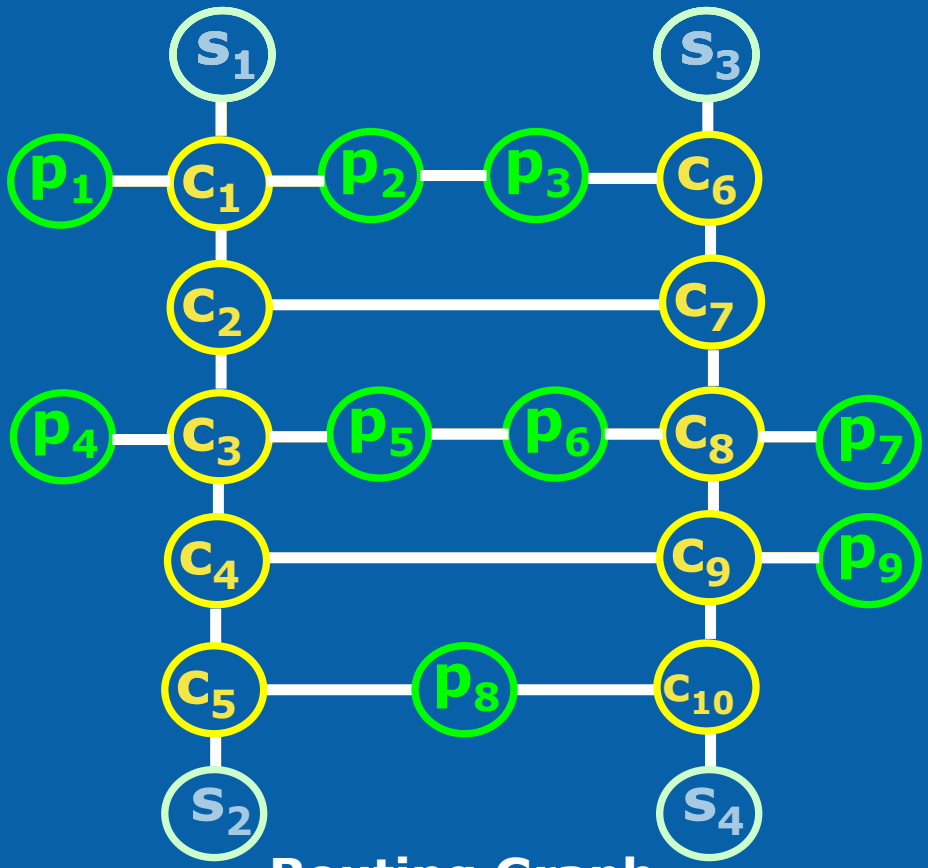
Global clock routing problem instance



A routing solution

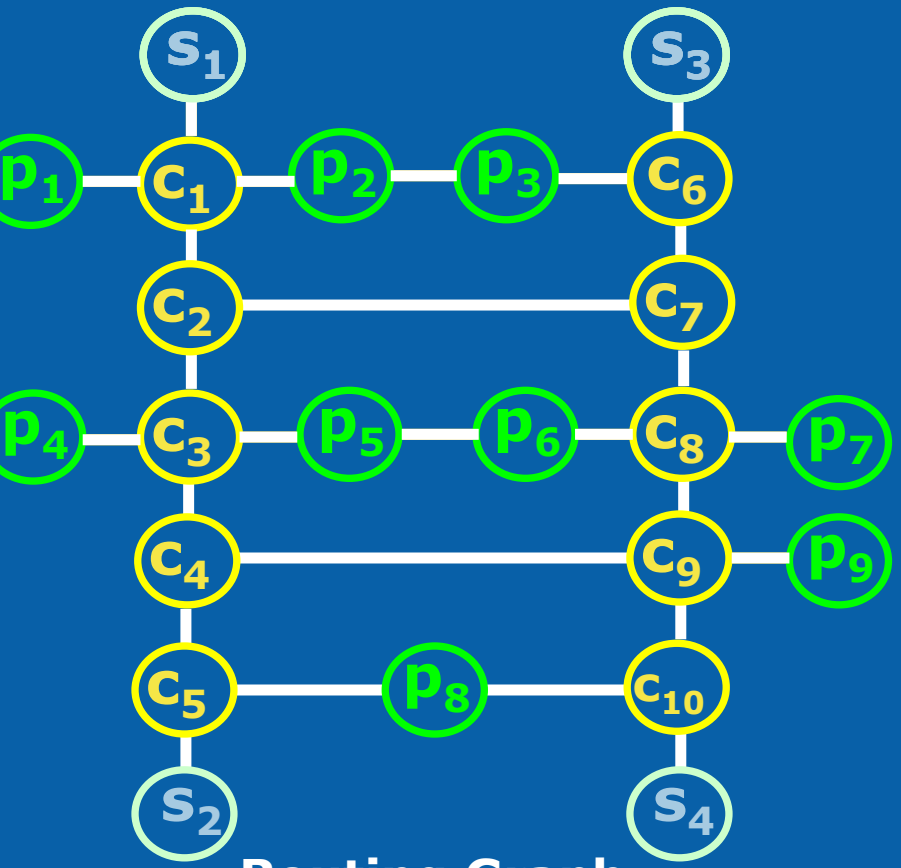
Problem Formulation: Graph Construction





Routing Graph

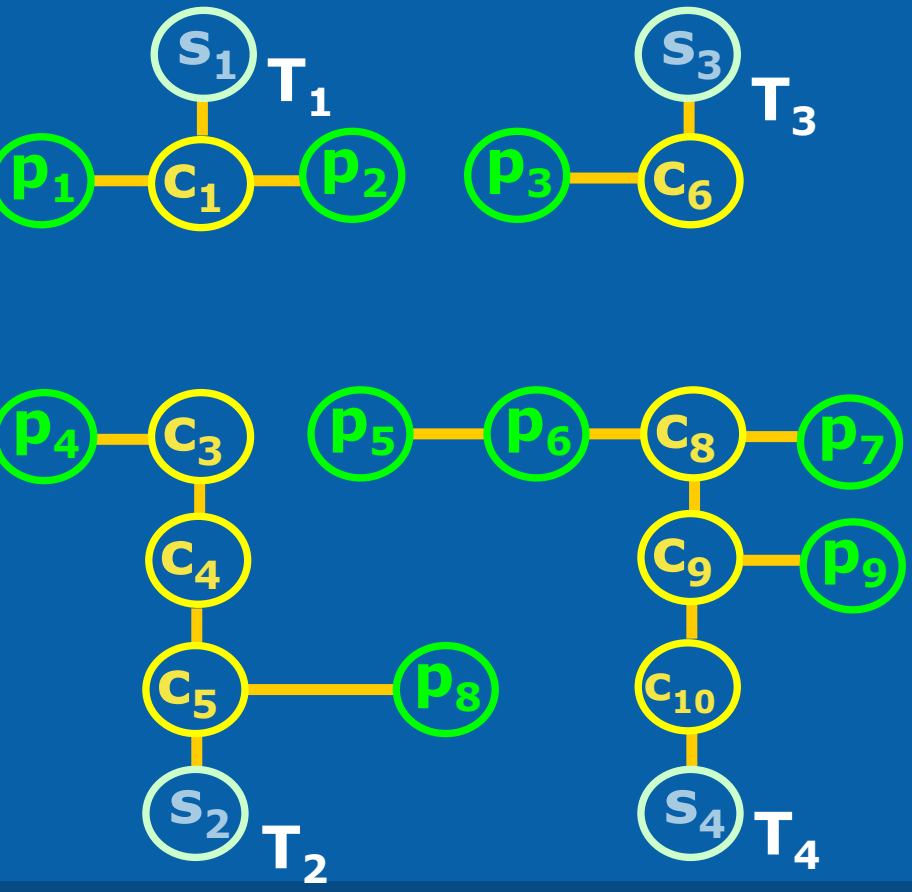
Problem Statement

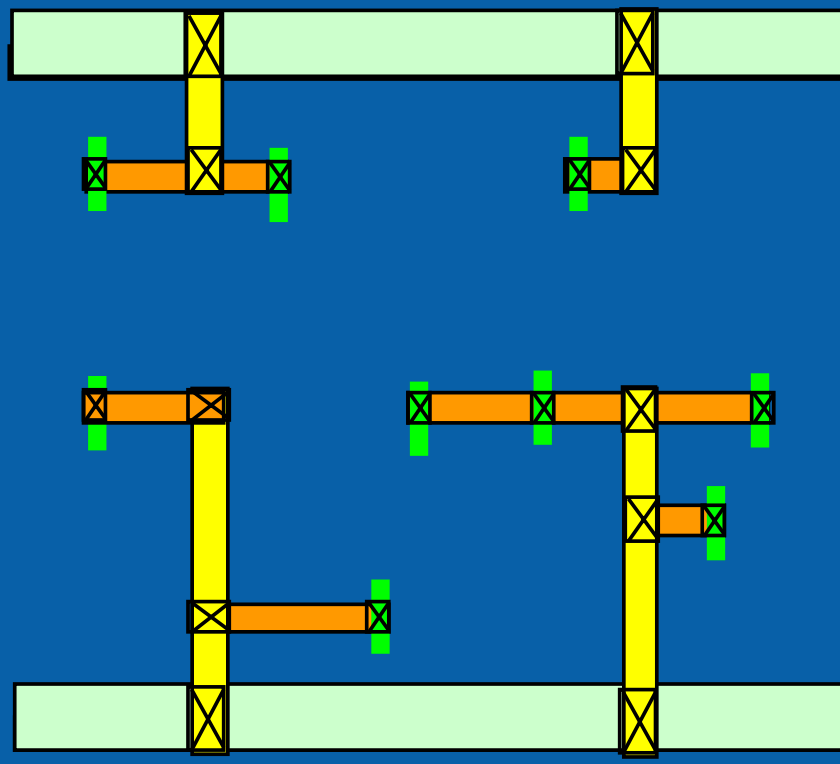


Routing Graph

- Find trees connecting s_* nodes to p_* nodes such that
  - For any tree T, distance from s_{T*} to $p_{T*} \leq \text{Distance}_{limit}$
  - For any tree T, $\Sigma \text{Cap}(p_{T*}) \leq \text{Cap}_{lim}$
  - $\Sigma_{\text{over all trees T*}}$ Wirelength(T*) is minimum
- One more constraint (ignored for now)
  - Grid loading constraint: Loads including that of interconnects on a grid wire is less than specified limit

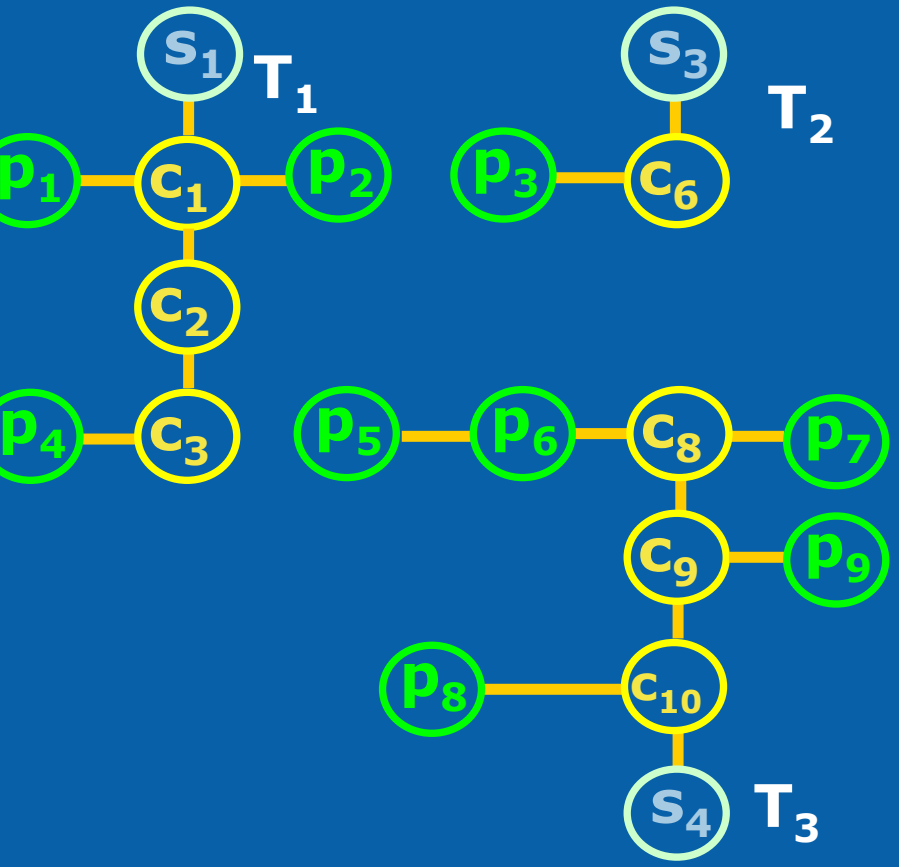
Trees to Global Clock Routes

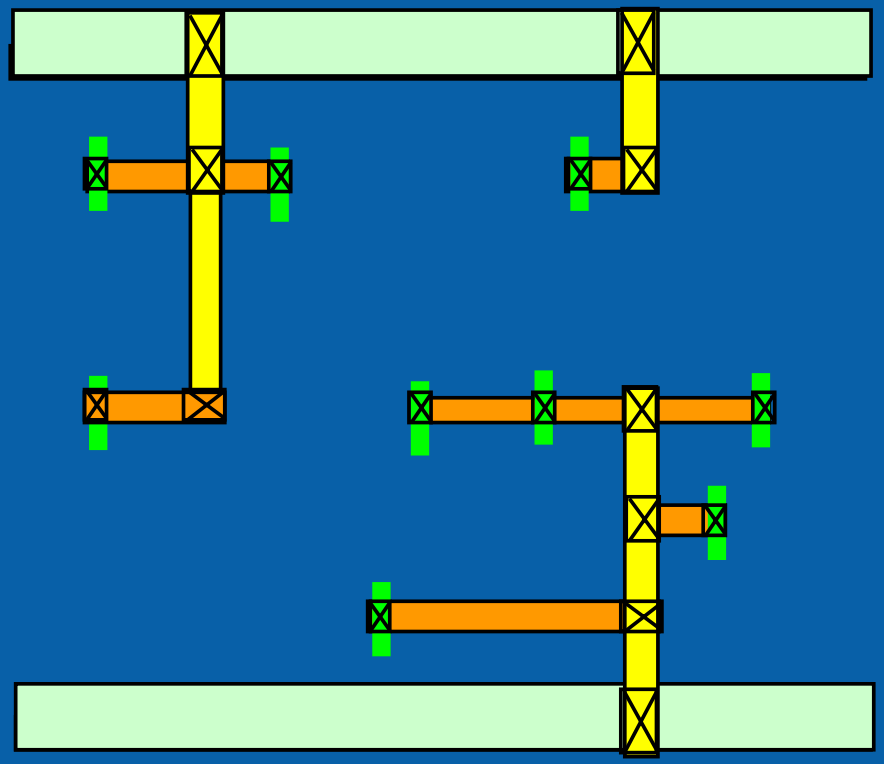




A routing solution

Tress to Global Clock Routes

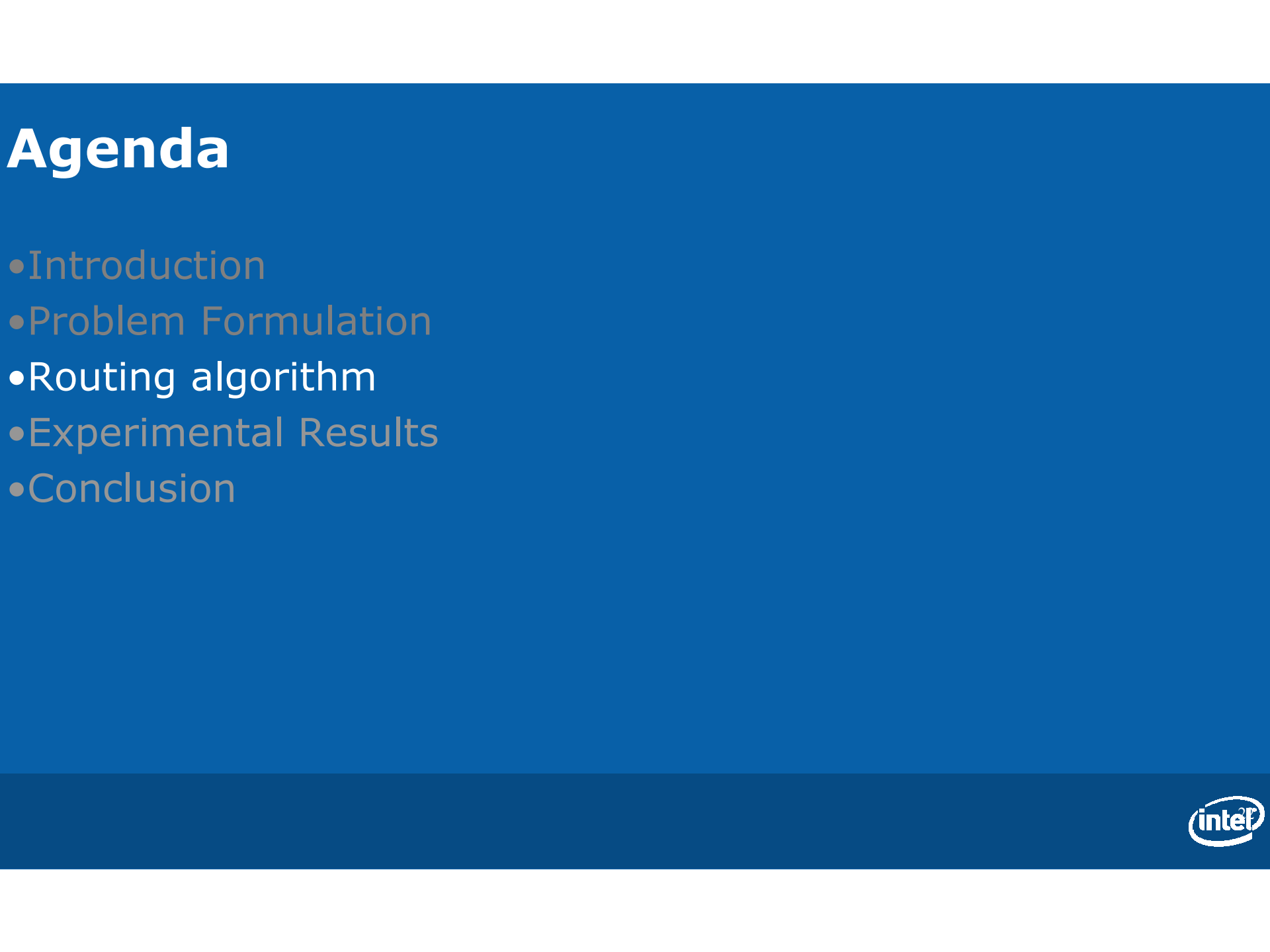




Another routing solution

Agenda

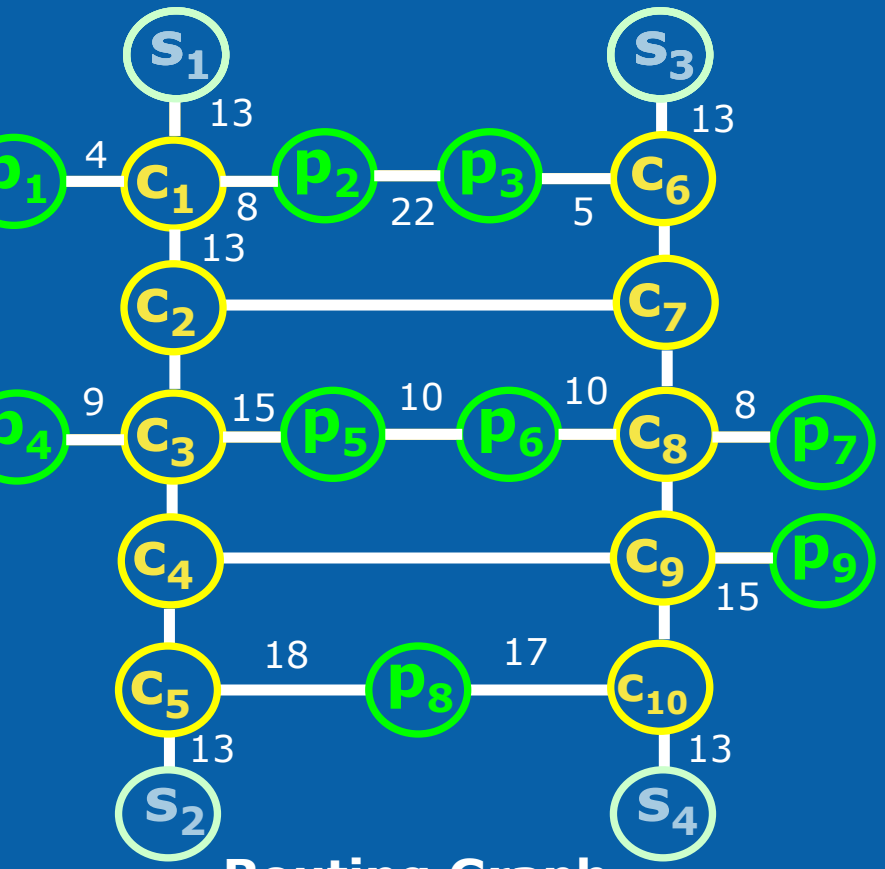
- Introduction
- Problem Formulation
- Routing algorithm
- Experimental Results
- Conclusion



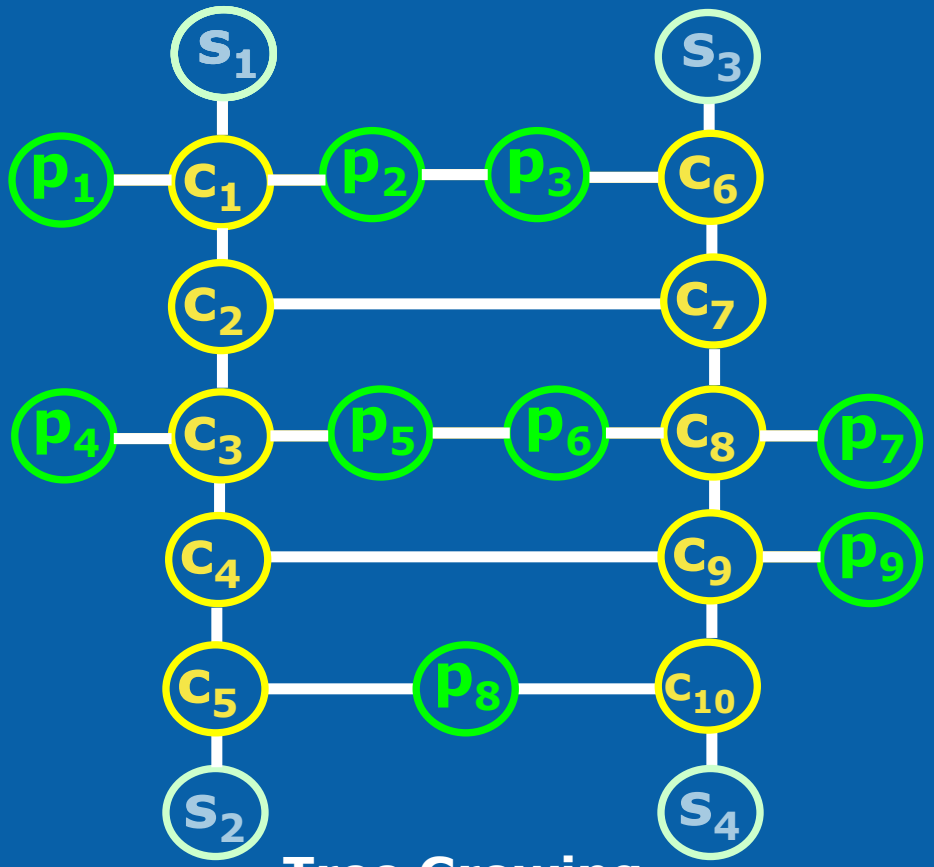
Algorithm for Routing with Capacitance and Distance constraint

- Multi-source/multi-destination routing problem with capacitance and distance constraint
  - Can be transformed to clustering with capacity constraints to minimize wirelength, which is NP-complete
- Efficient heuristics/approximation algorithms needed to solve the problem
- Tree Growing Heuristic:
  - Start growing trees from the source nodes
  - Grow trees by adding edges till all ports are connected
    - Edges are sorted in ascending order of the wire-cap to minimize total wire-cap due to global clock routes
  - Add edges iff:
    - Doing so does not violate capacitance/distance constraints
    - The node is already not connected to grid by some other route (tree)

Tree Growing Heuristic: Example

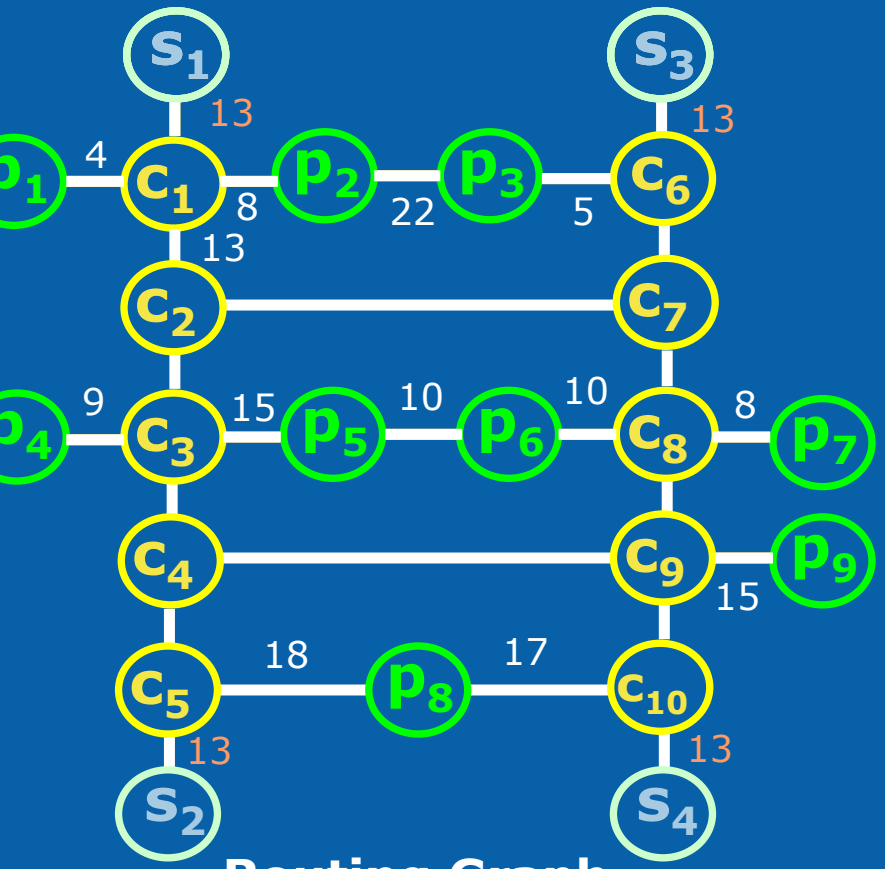


Routing Graph

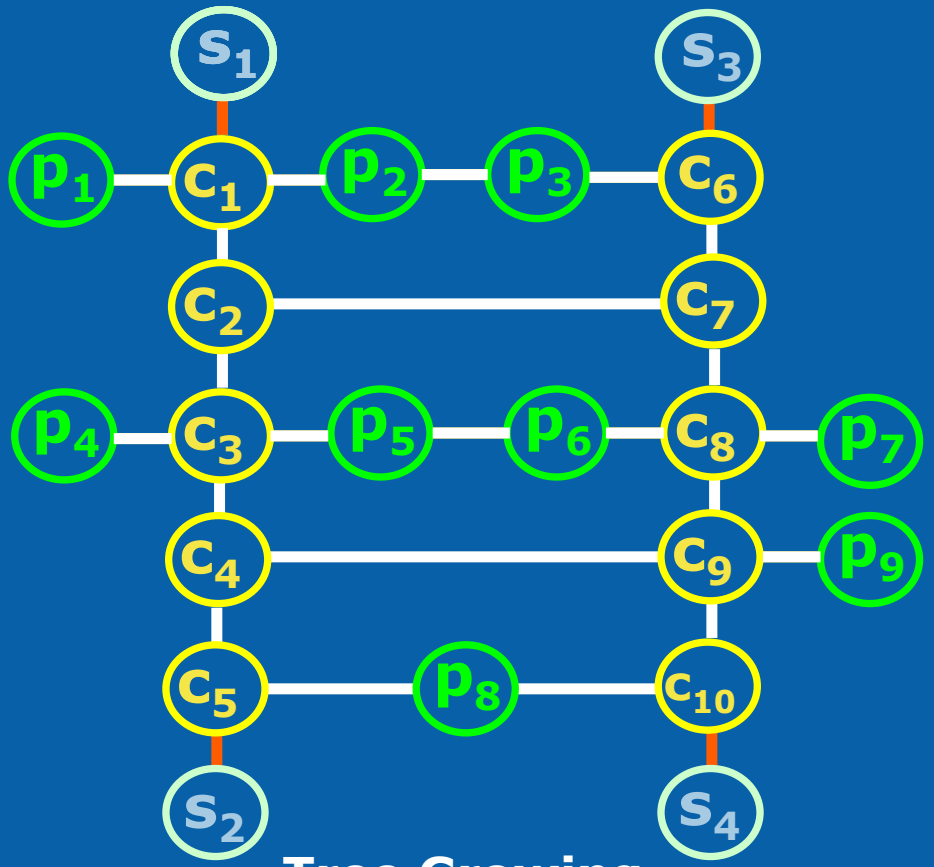


Tree Growing

Tree Growing Heuristic: Example

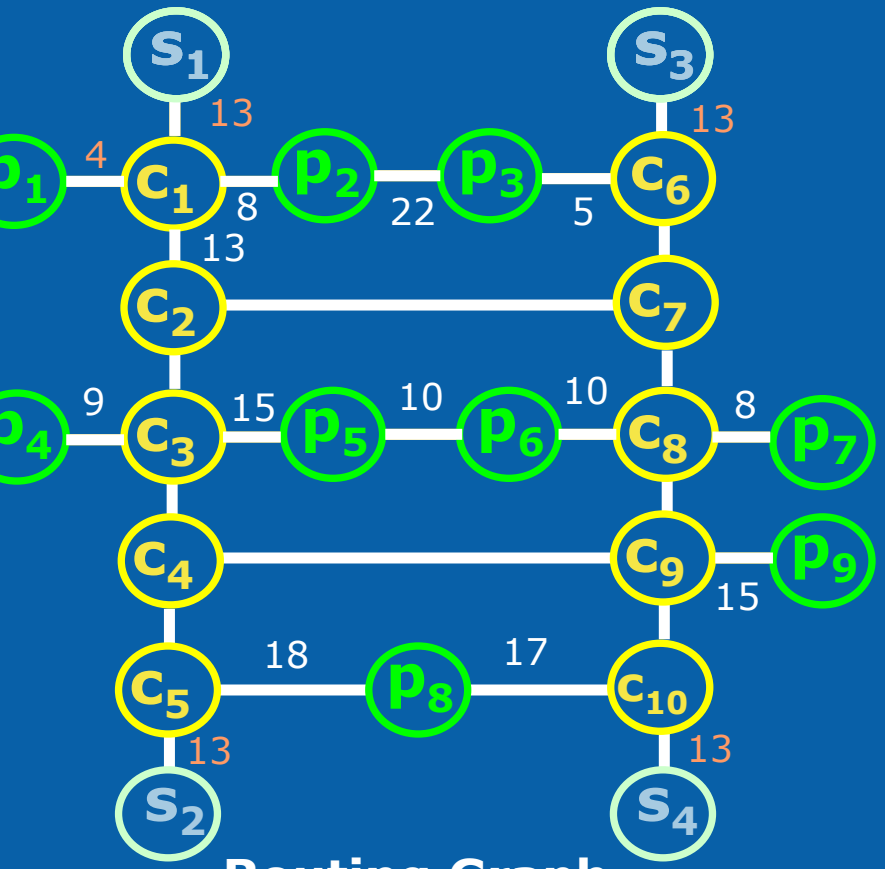


Routing Graph

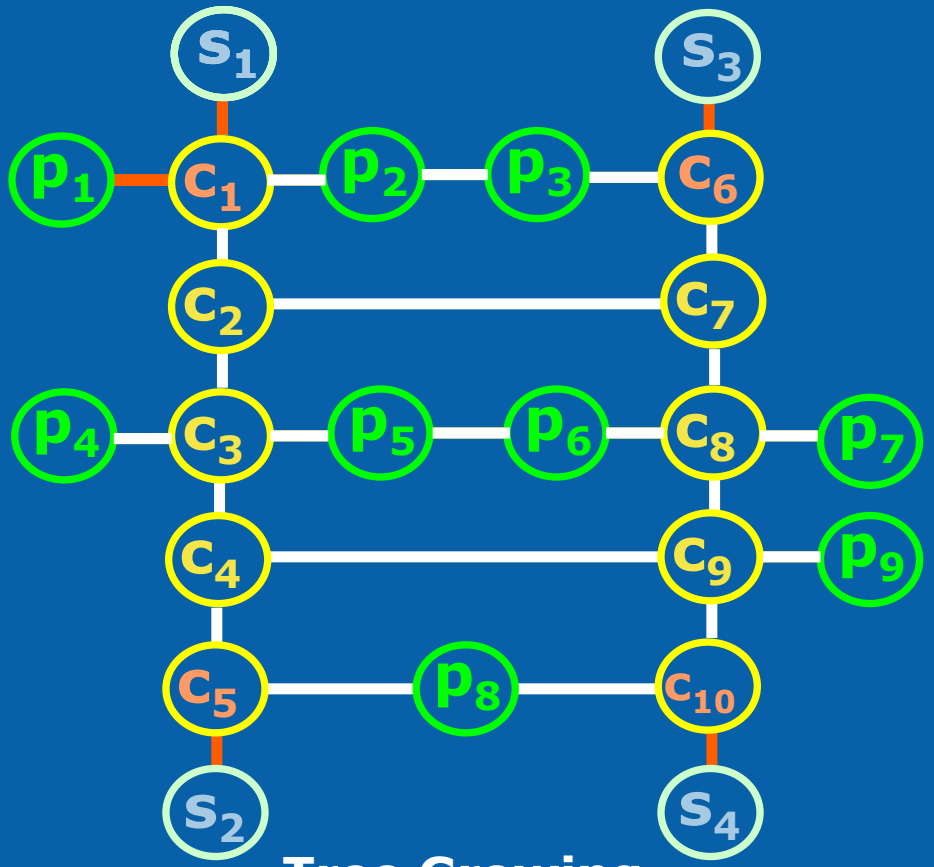


Tree Growing

Tree Growing Heuristic: Example

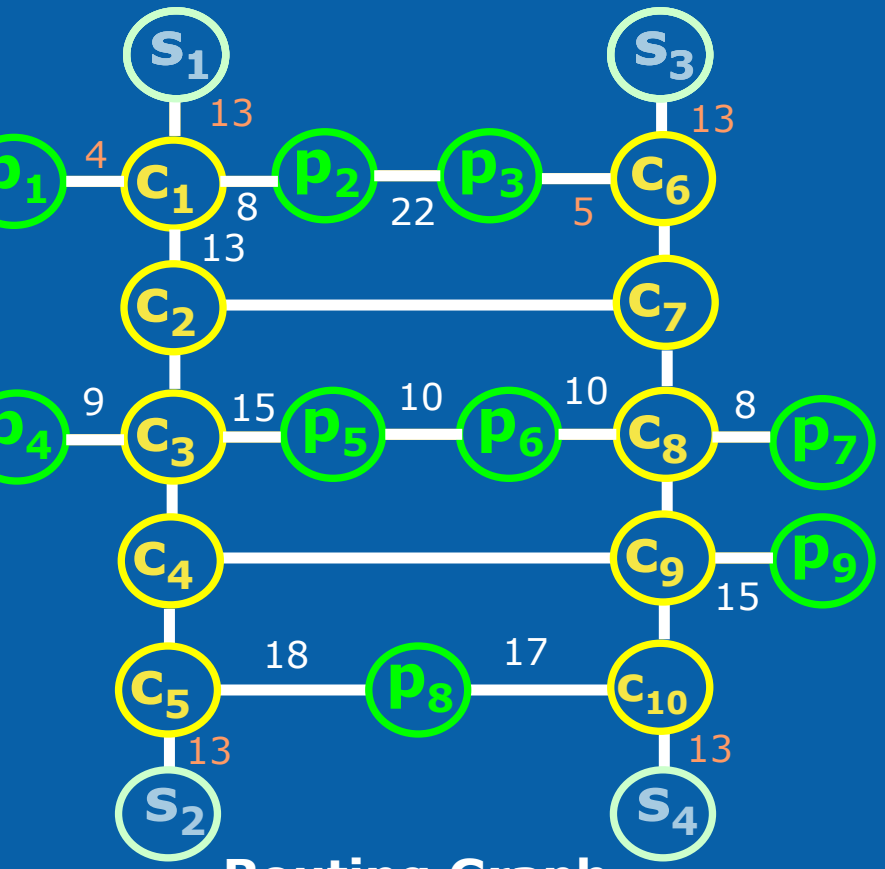


Routing Graph

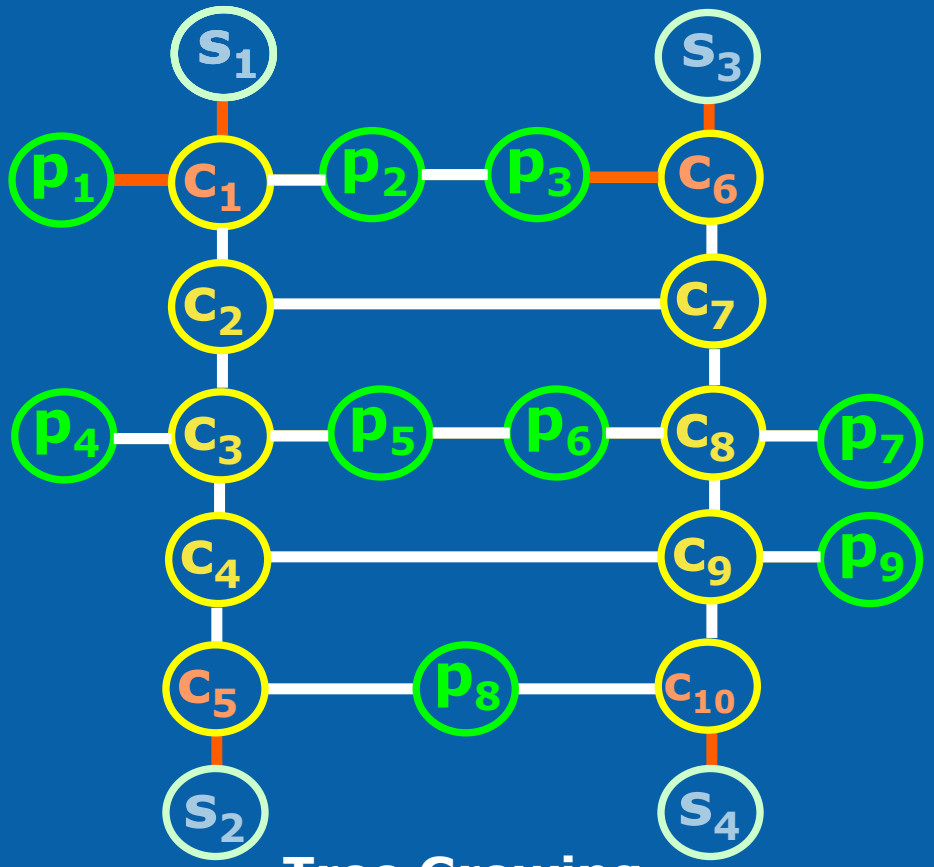


Tree Growing

Tree Growing Heuristic: Example

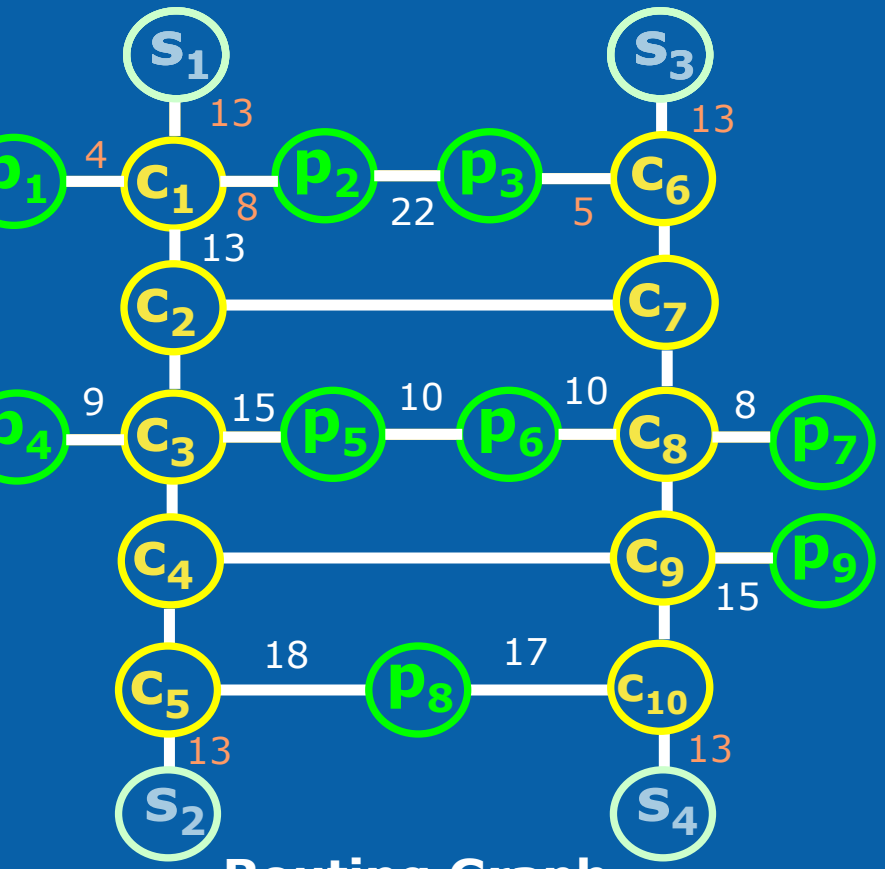


Routing Graph

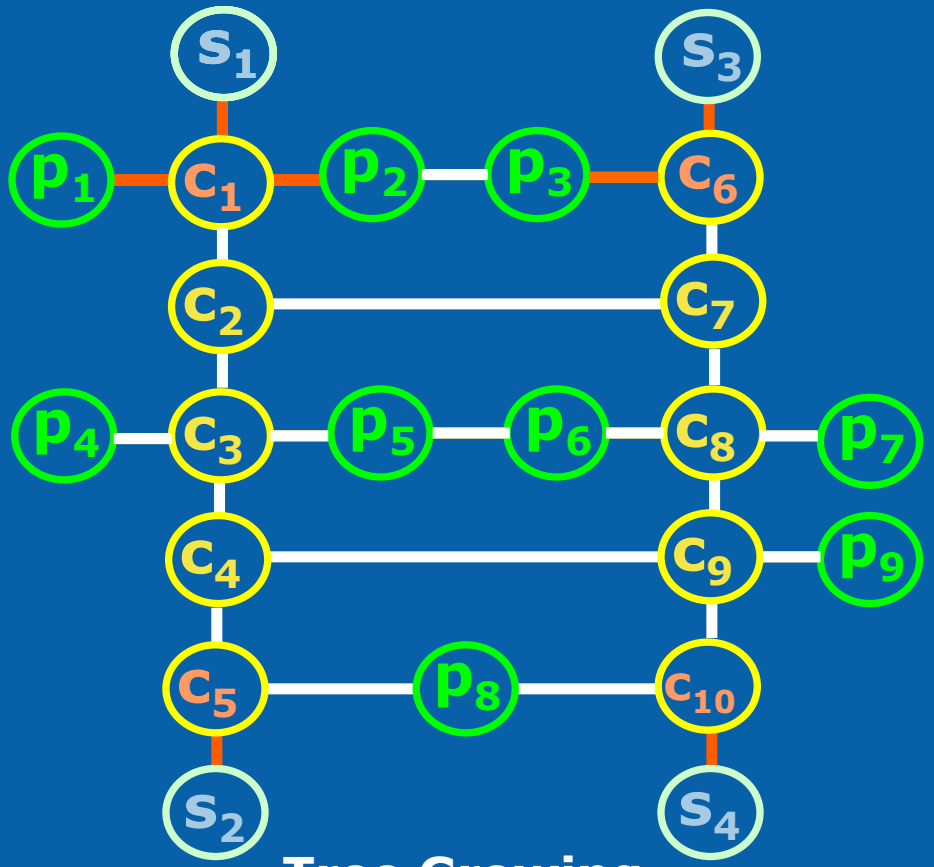


Tree Growing

Tree Growing Heuristic: Example

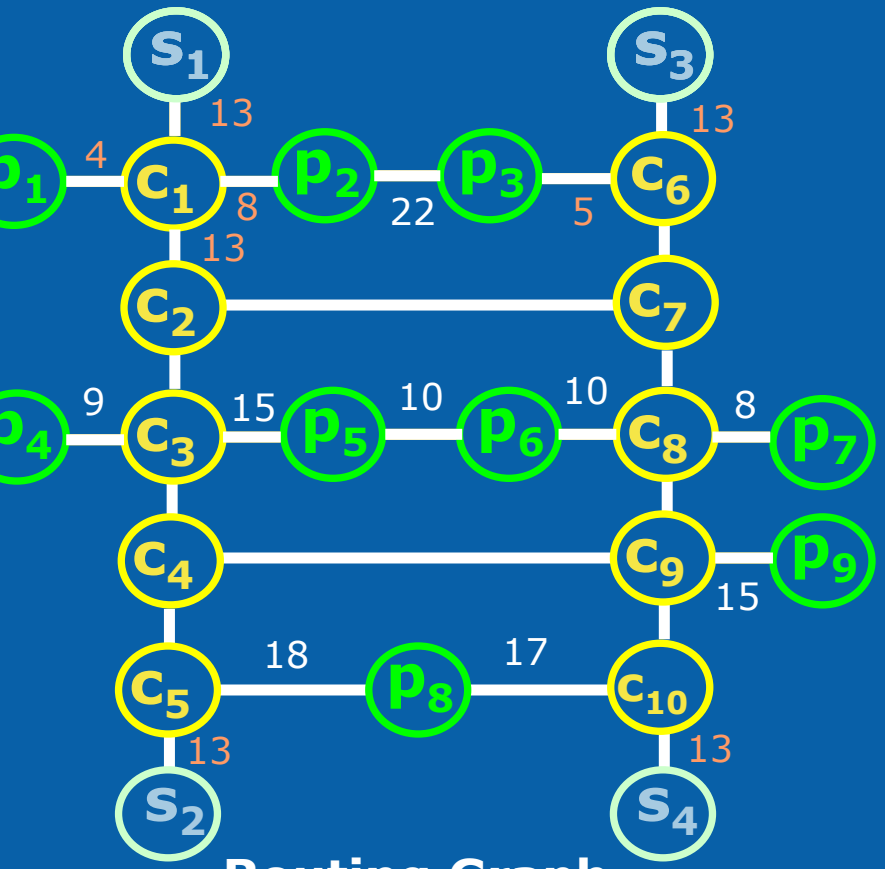


Routing Graph

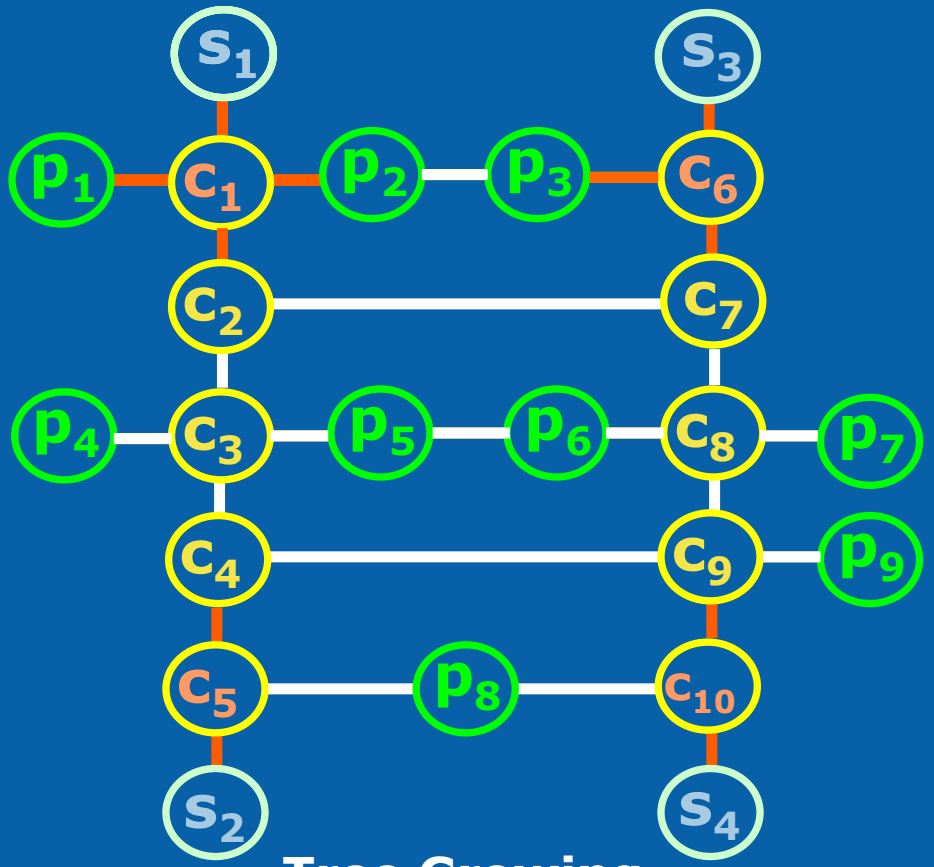


Tree Growing

Tree Growing Heuristic: Example

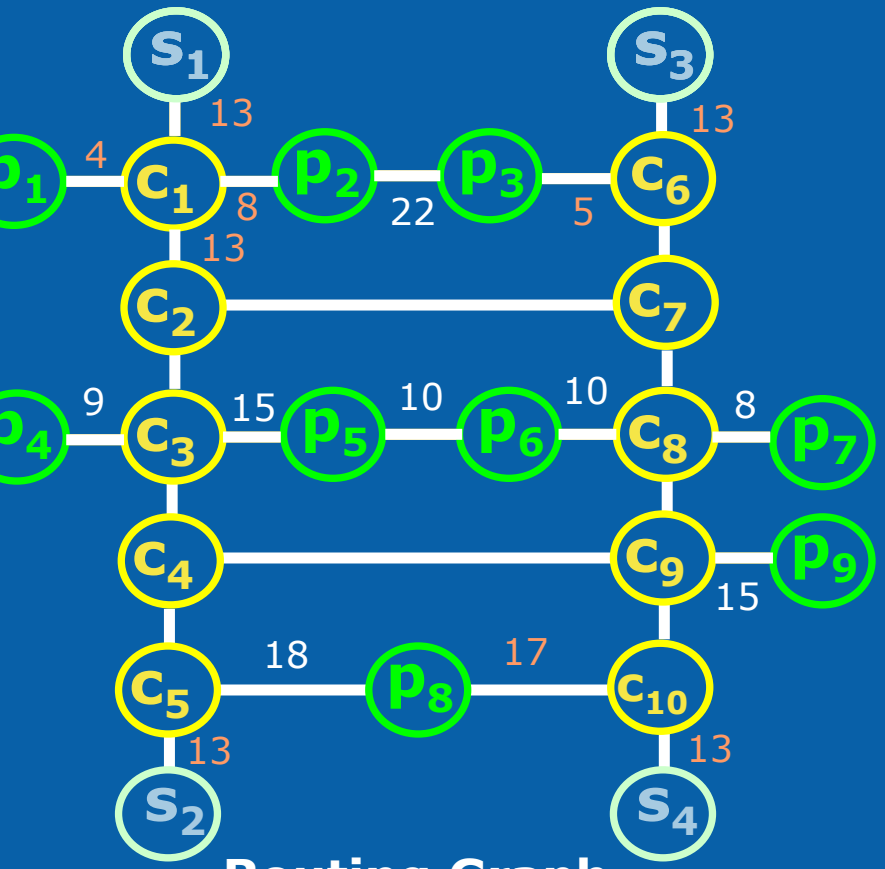


Routing Graph

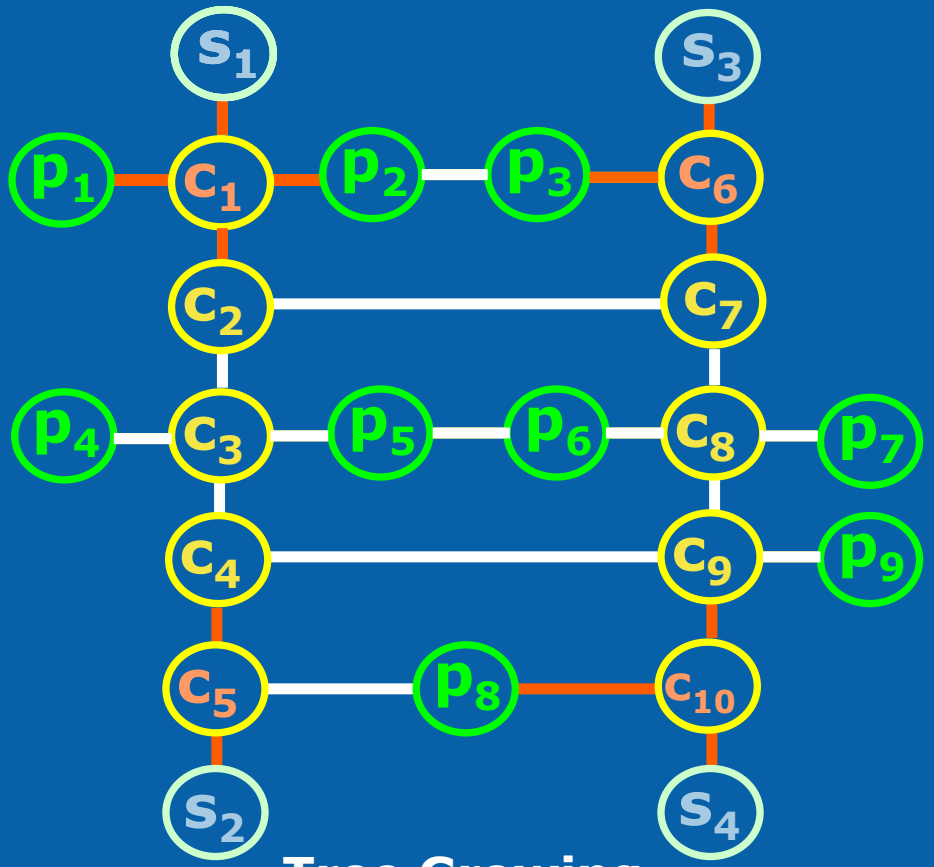


Tree Growing

Tree Growing Heuristic: Example

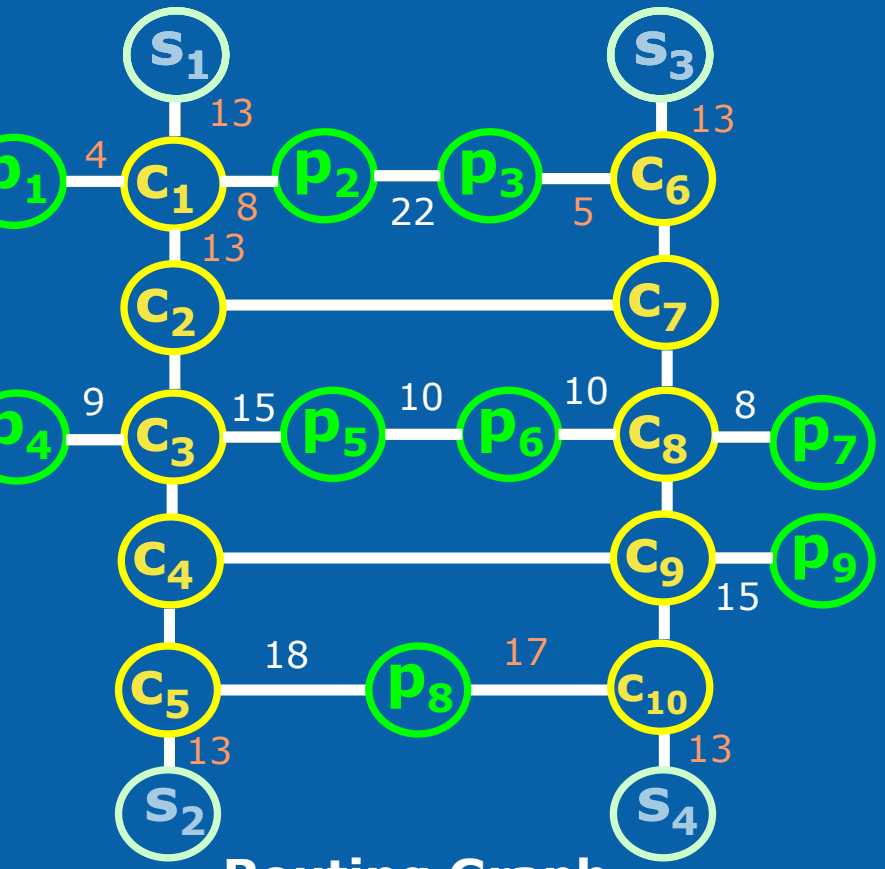


Routing Graph

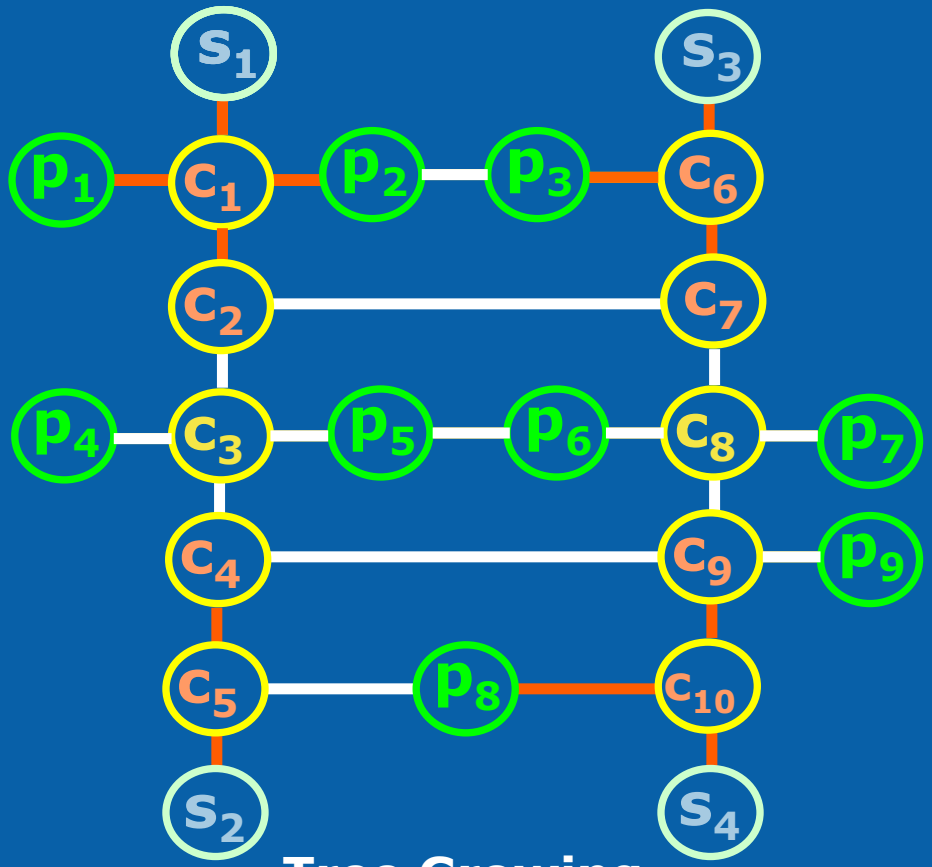


Tree Growing

Tree Growing Heuristic: Example

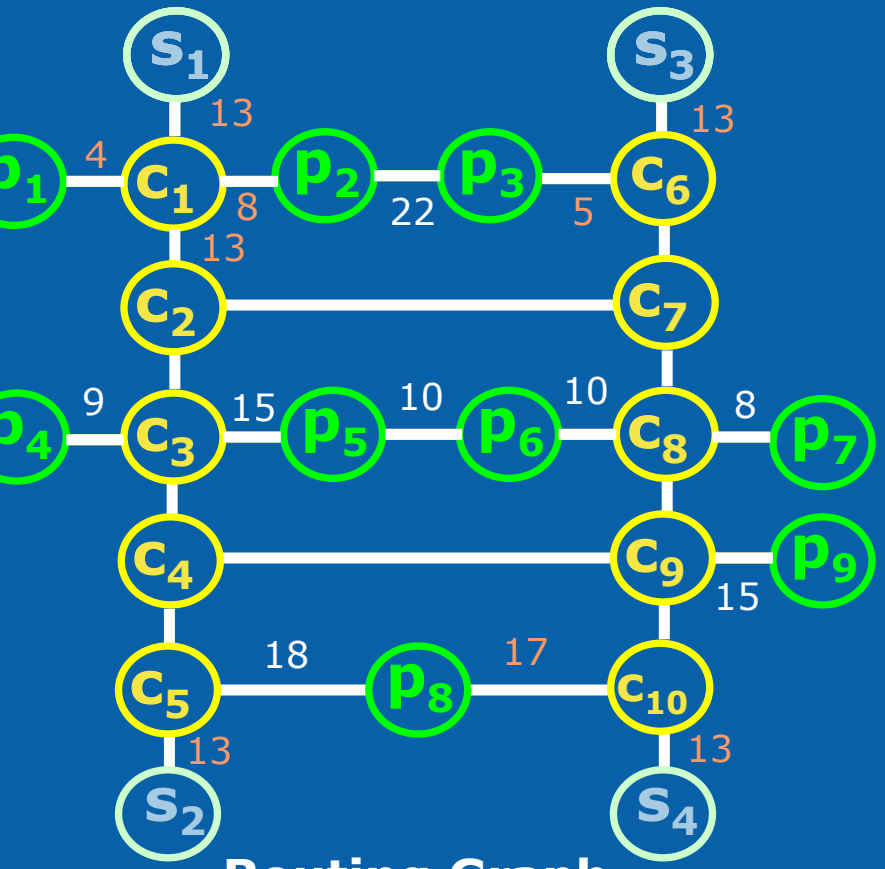


Routing Graph

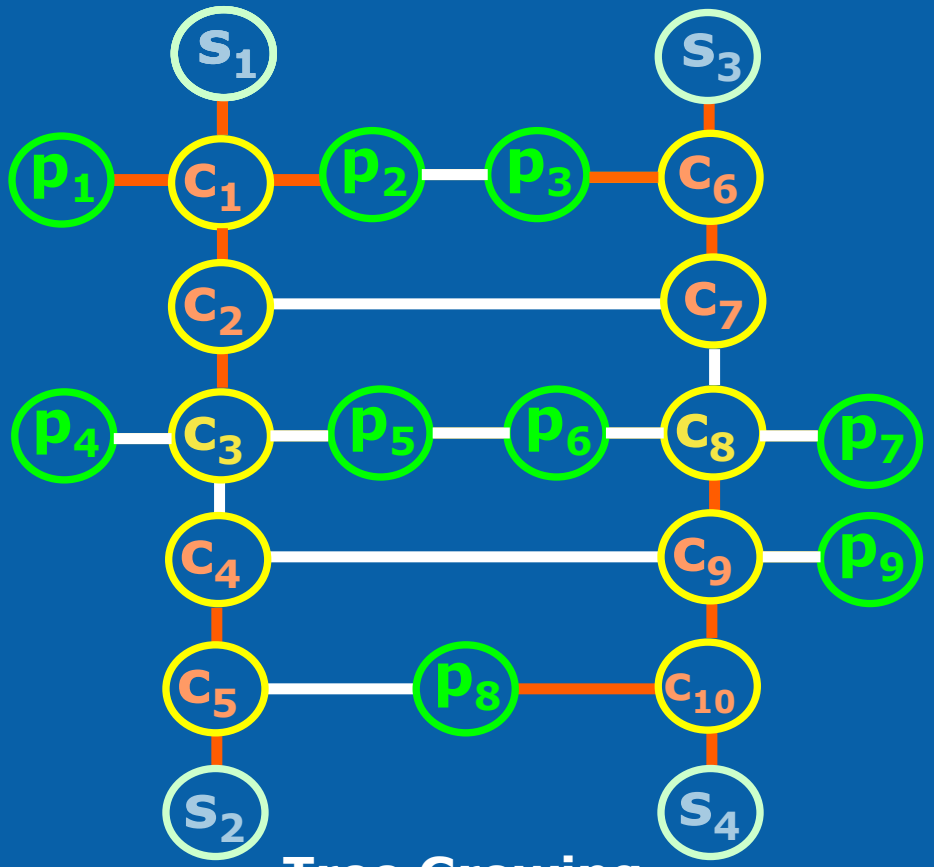


Tree Growing

Tree Growing Heuristic: Example

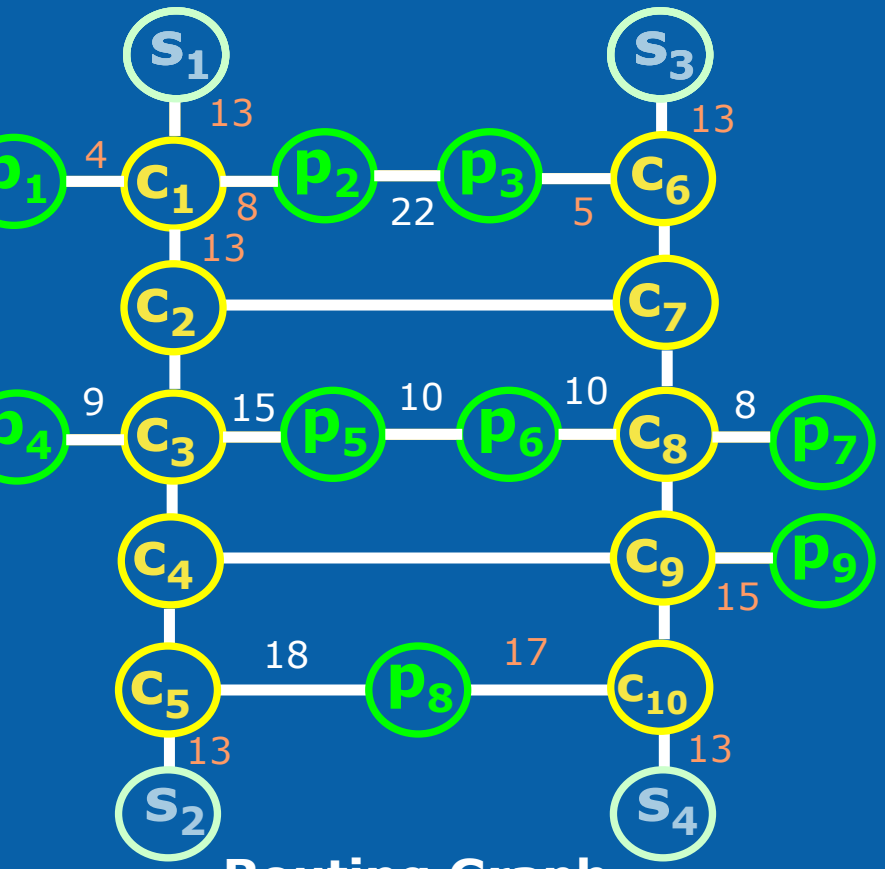


Routing Graph

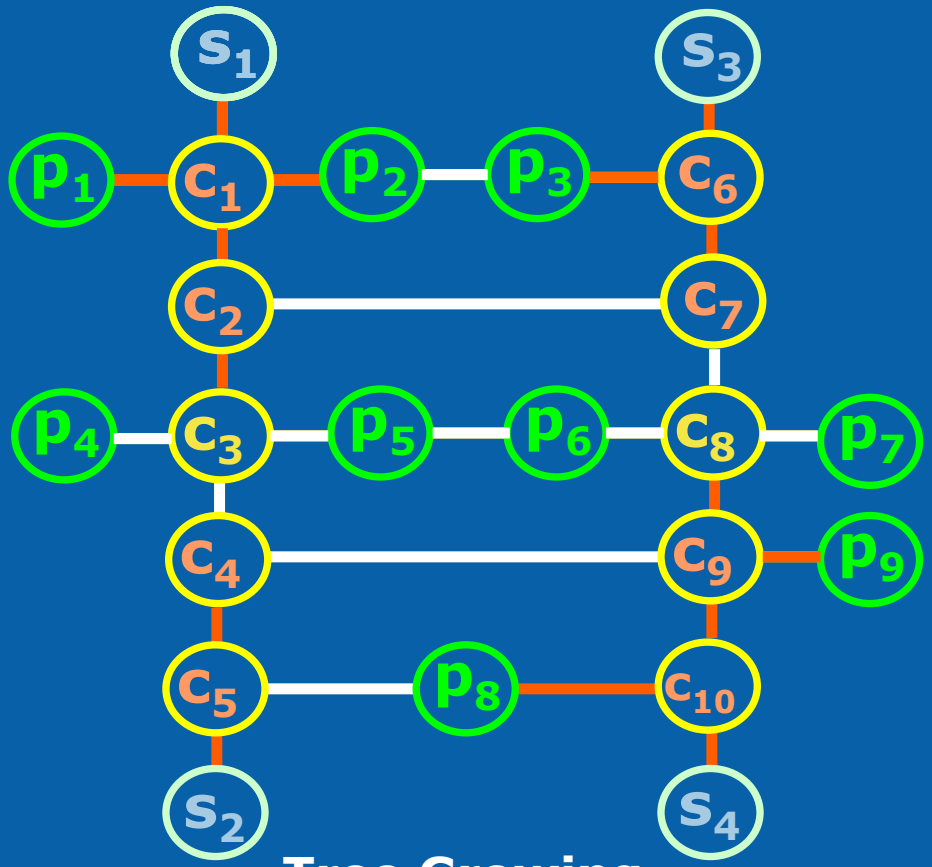


Tree Growing

Tree Growing Heuristic: Example

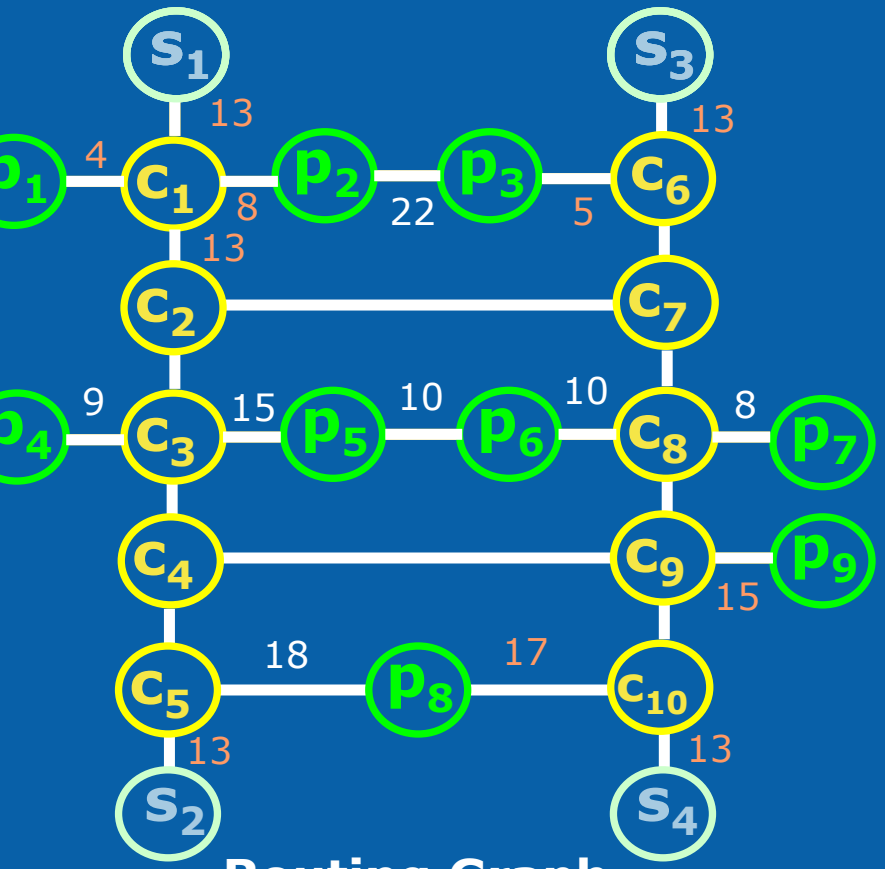


Routing Graph

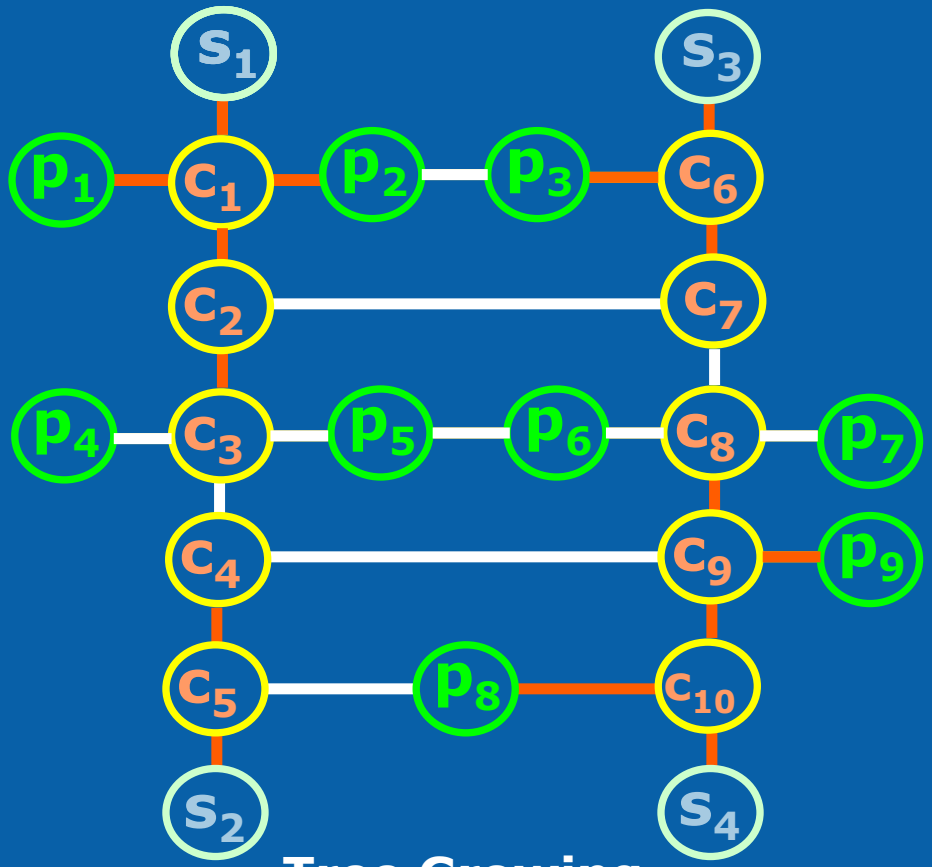


Tree Growing

Tree Growing Heuristic: Example

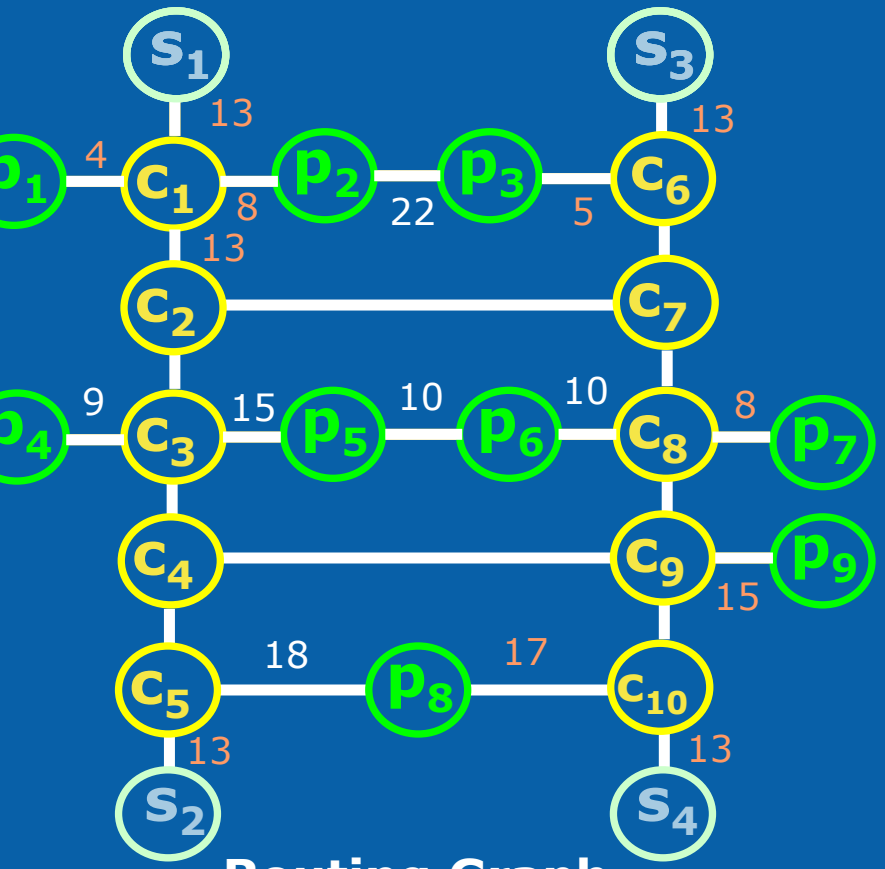


Routing Graph

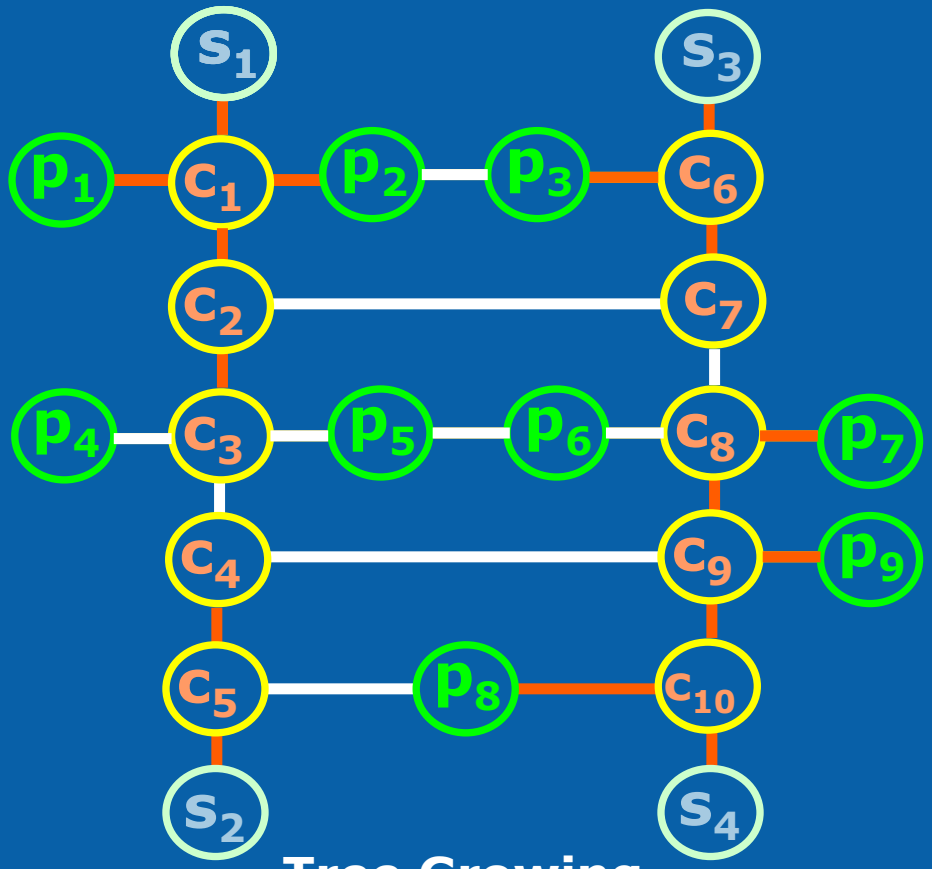


Tree Growing

Tree Growing Heuristic: Example

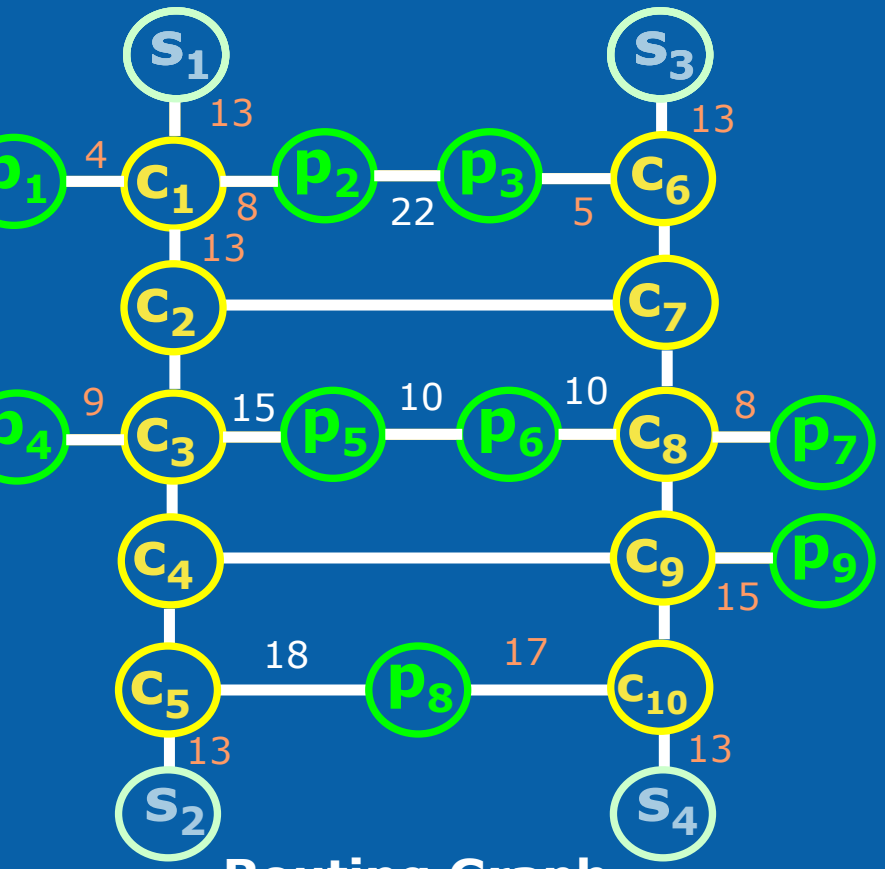


Routing Graph

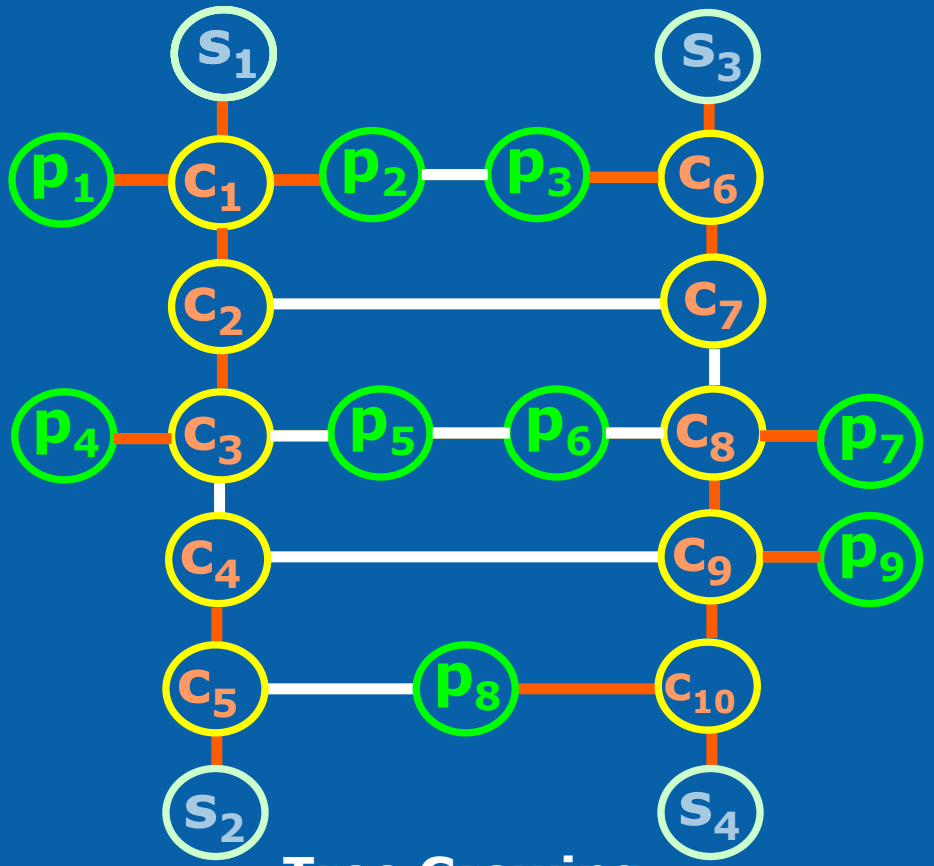


Tree Growing

Tree Growing Heuristic: Example

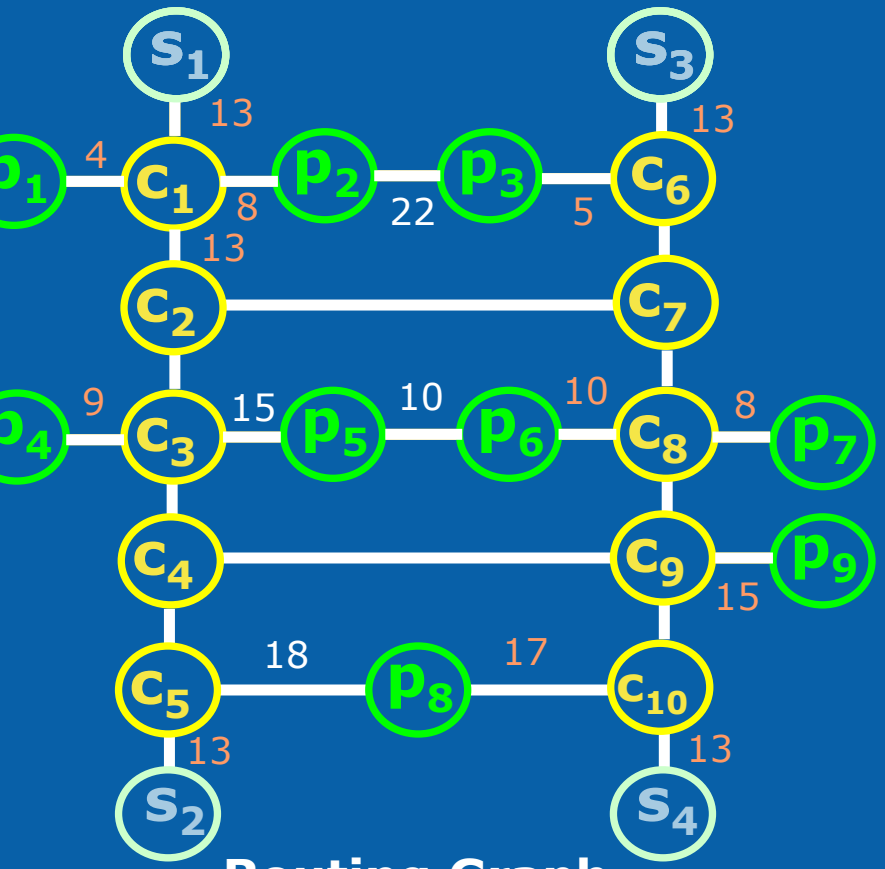


Routing Graph

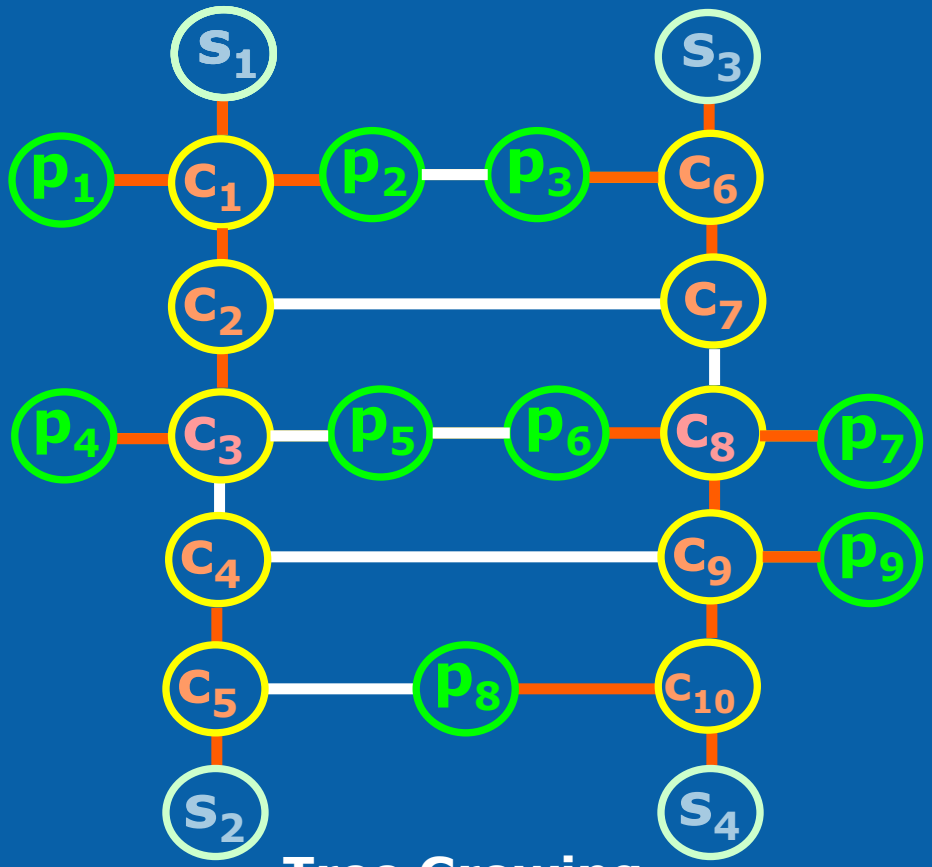


Tree Growing

Tree Growing Heuristic: Example

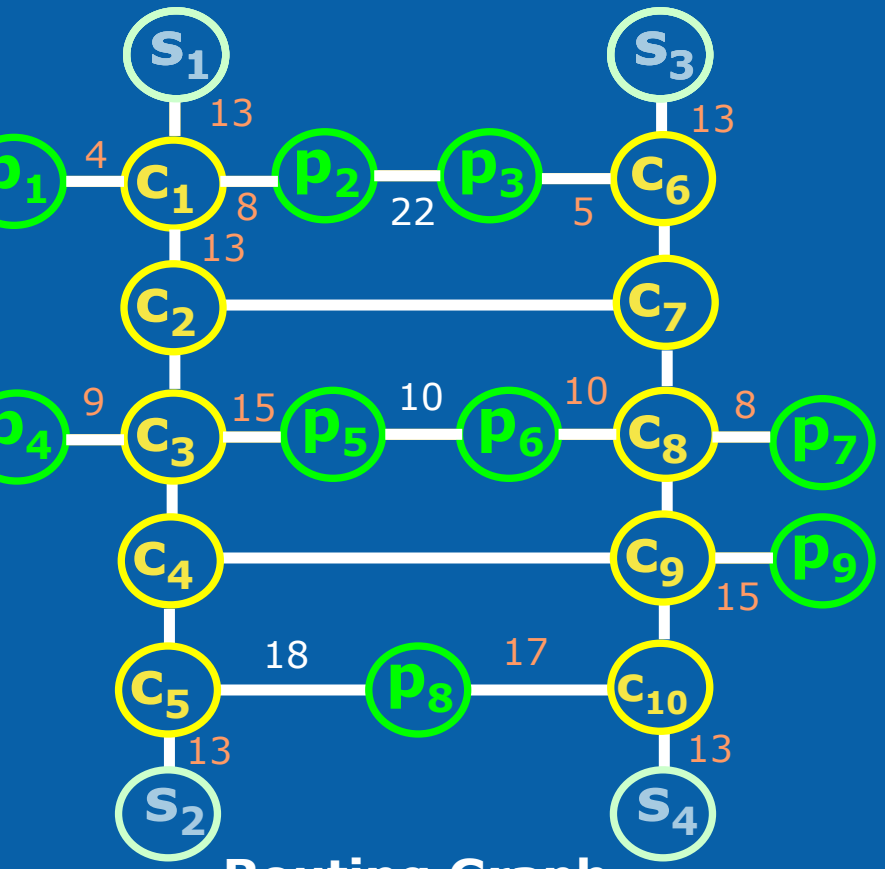


Routing Graph

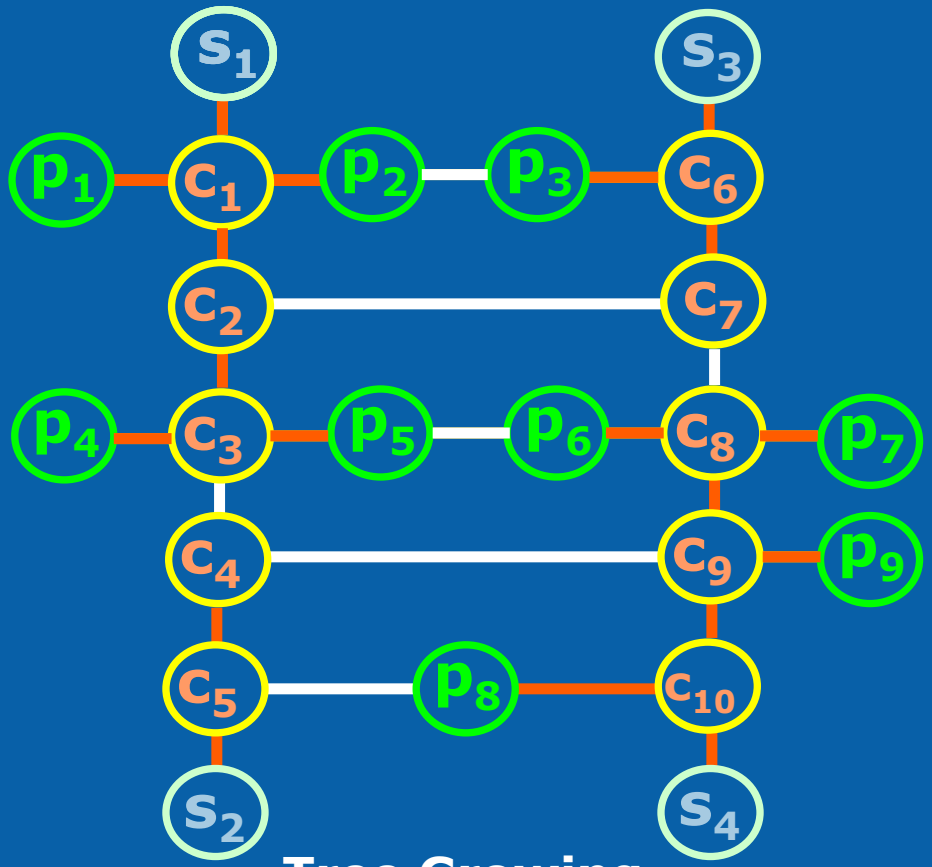


Tree Growing

Tree Growing Heuristic: Example

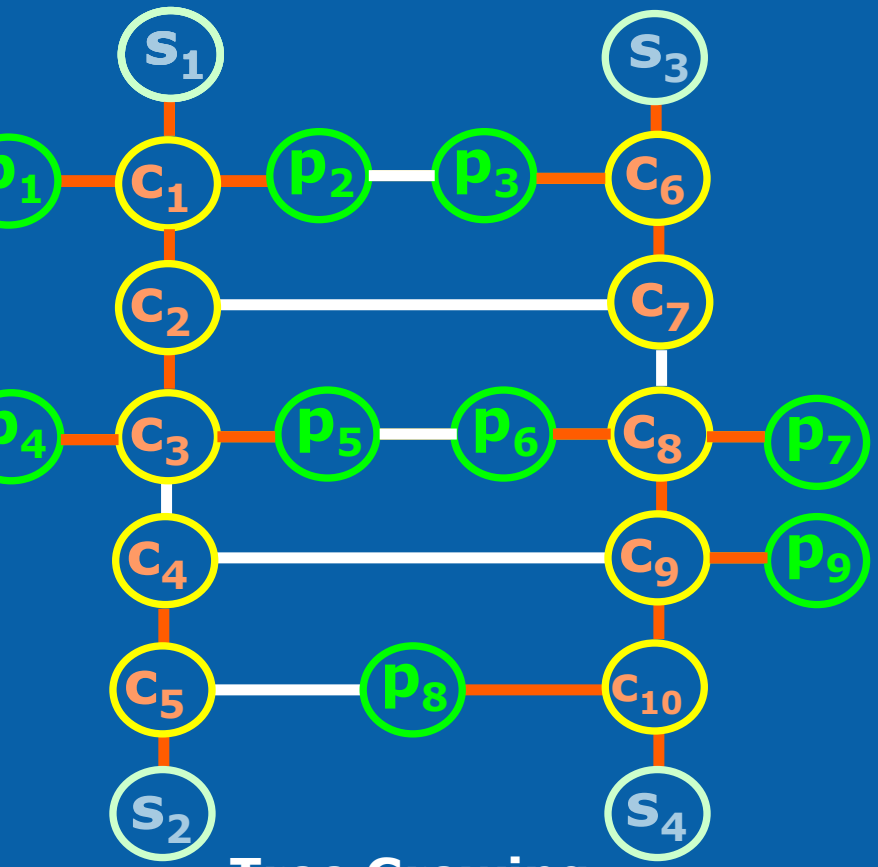


Routing Graph

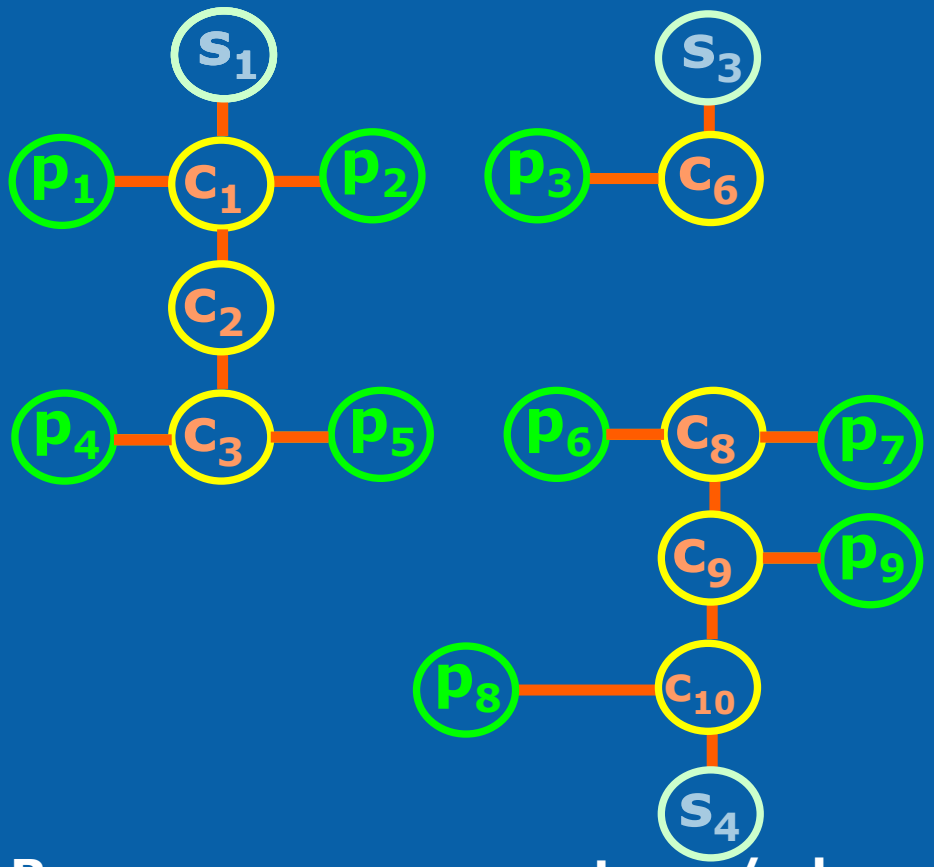


Tree Growing

Tree Growing Heuristic: Example

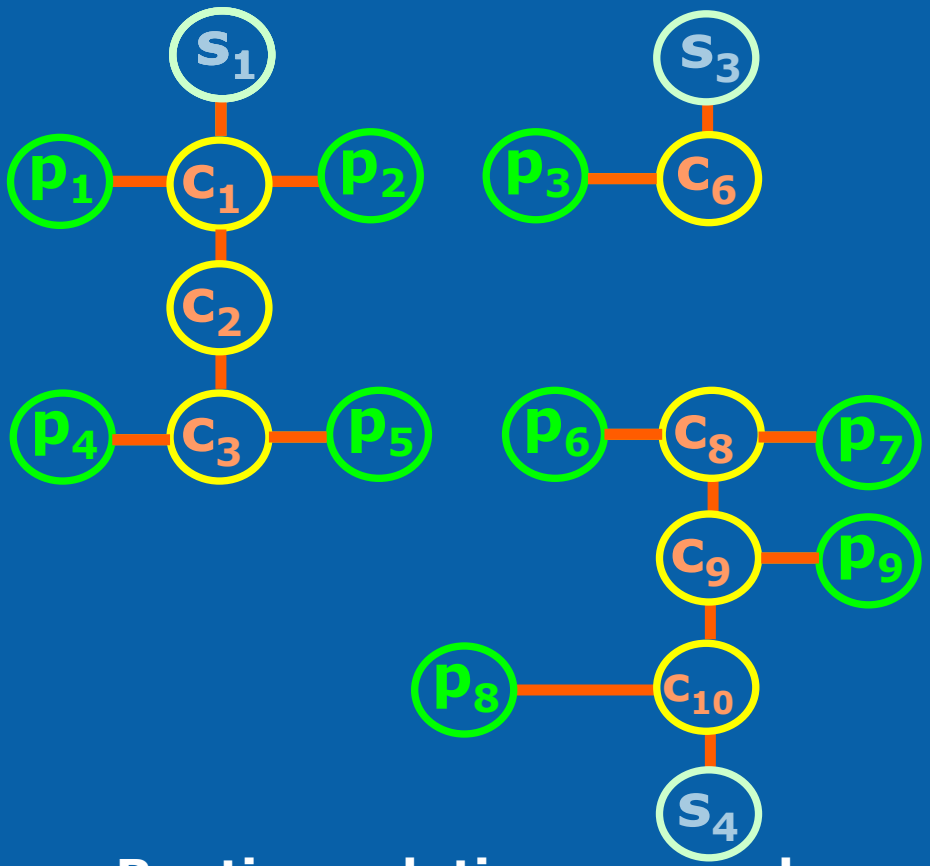


Tree Growing

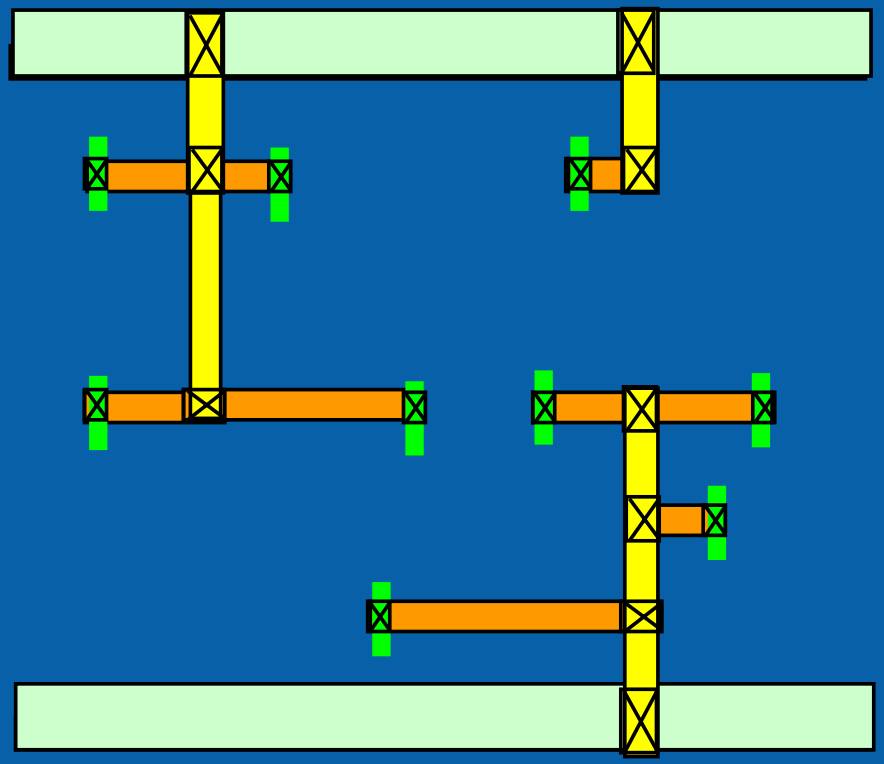


Remove unnecessary trees/edges

Tree Growing Heuristic: Example



Routing solution on graph



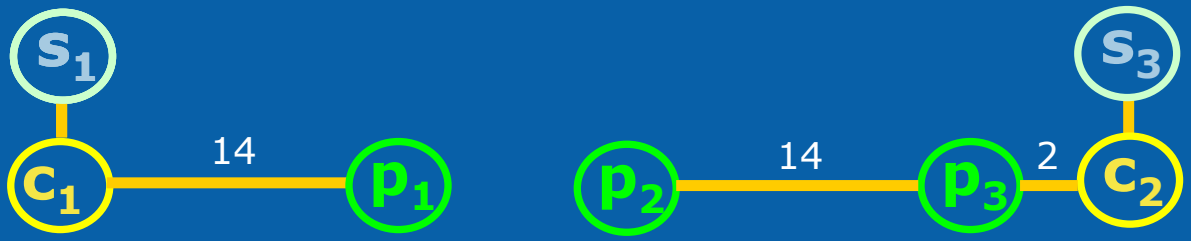
Routing solution

Time complexity and limitations

- Time complexity: $O(|E|^2 \text{Log}|E|)$, where E is the set of edges in the routing graph
  - $|E| \leq 2|P| + 4\,mn = O(|P|+mn)$, where P is the set of ports, m = number of horizontal tracks or grid-wires, n = number of vertical tracks
- Run-time, in practice, on microprocessor layout areas: up to 10 seconds
  - Typical problem size: 1000 ports; ~500 tracks each in horizontal/vertical direction; area 1000x1000 um^2
- Does not consider grid loading limits yet
  - The special case of underlying problem leads to a bin-packing problem

Comparison with Nearest Source Heuristic

Routing Graph



Solution due to Nearest Source Heuristic

Comparison with Nearest Source Heuristic

Routing Graph

Solution due to Nearest Source Heuristic

Tree Growing Heuristic

Comparison with Nearest Source Heuristic

Routing Graph

Solution due to Nearest Source Heuristic

Tree Growing Heuristic

Comparison with Nearest Source Heuristic

Routing Graph

Solution due to Nearest Source Heuristic

Tree Growing Heuristic

Comparison with Nearest Source Heuristic

Routing Graph

Solution due to Nearest Source Heuristic

Tree Growing Heuristic

Comparison with Nearest Source Heuristic

Routing Graph

Wirelength =30

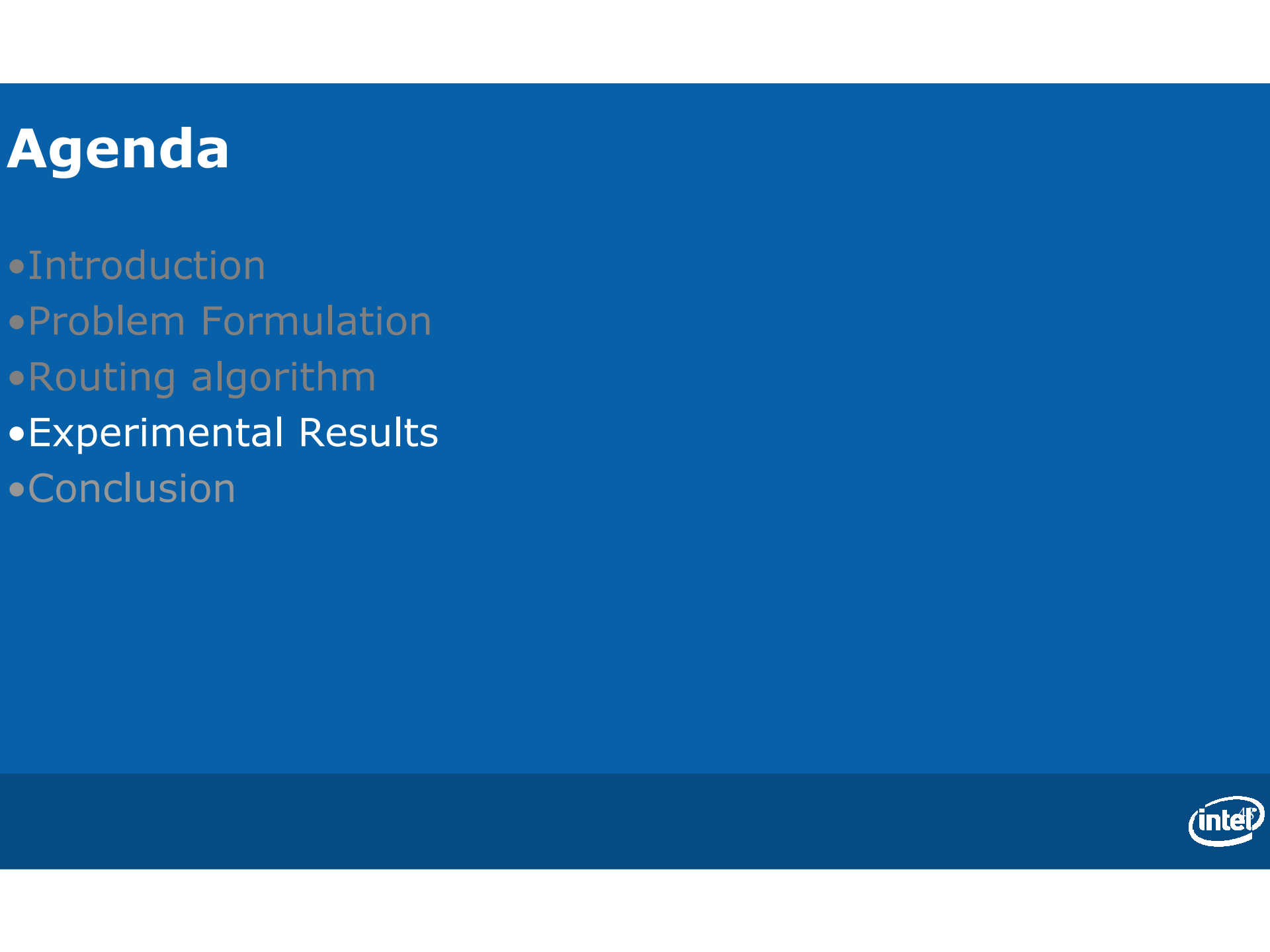
Solution due to Nearest Source Heuristic

Wirelength = 21

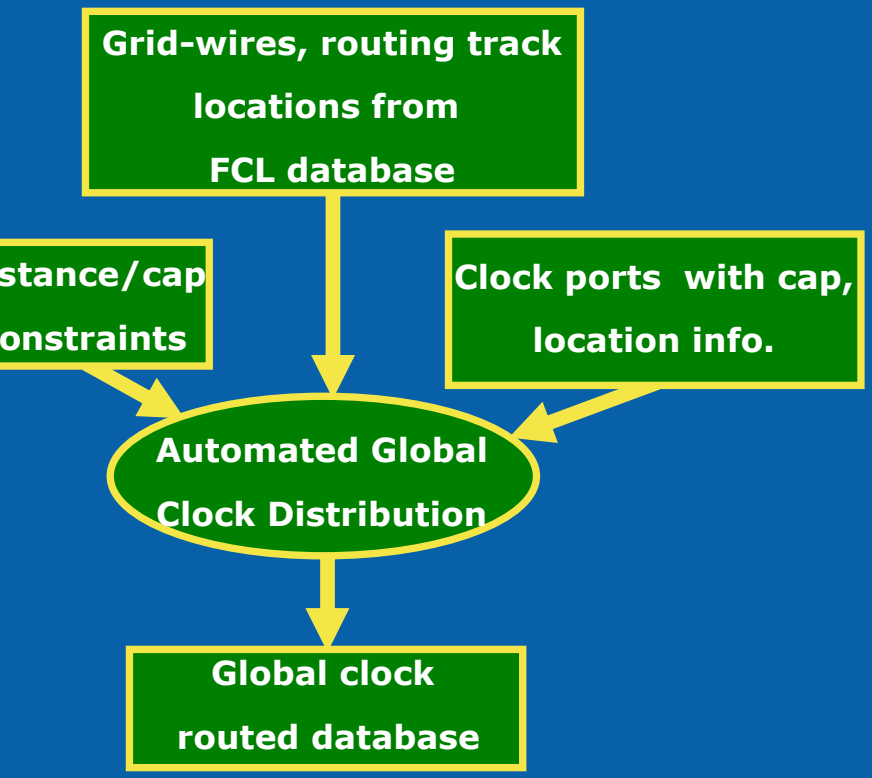
Solution due to Tree Growing Heuristic

Agenda

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

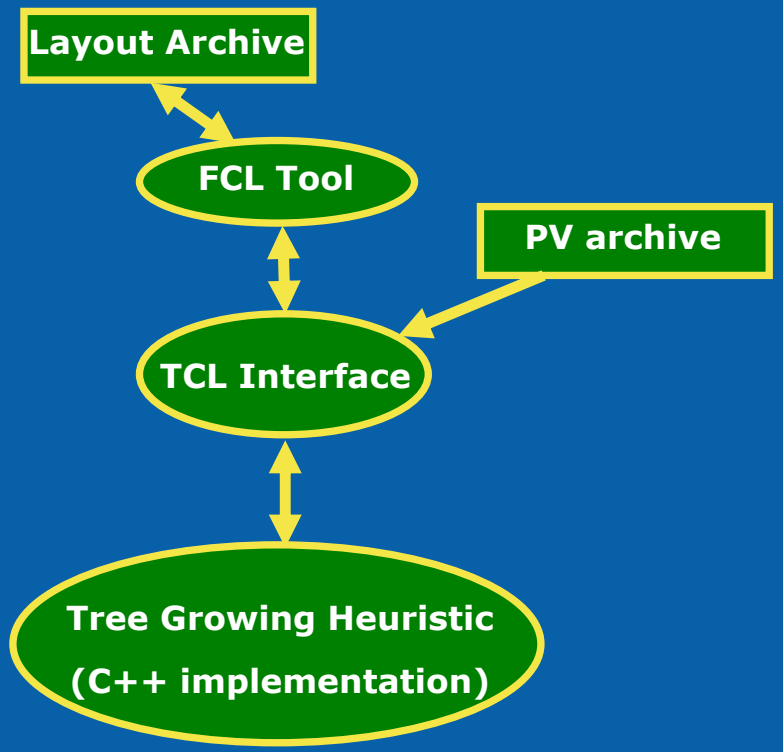


Global Clock Distribution Flow



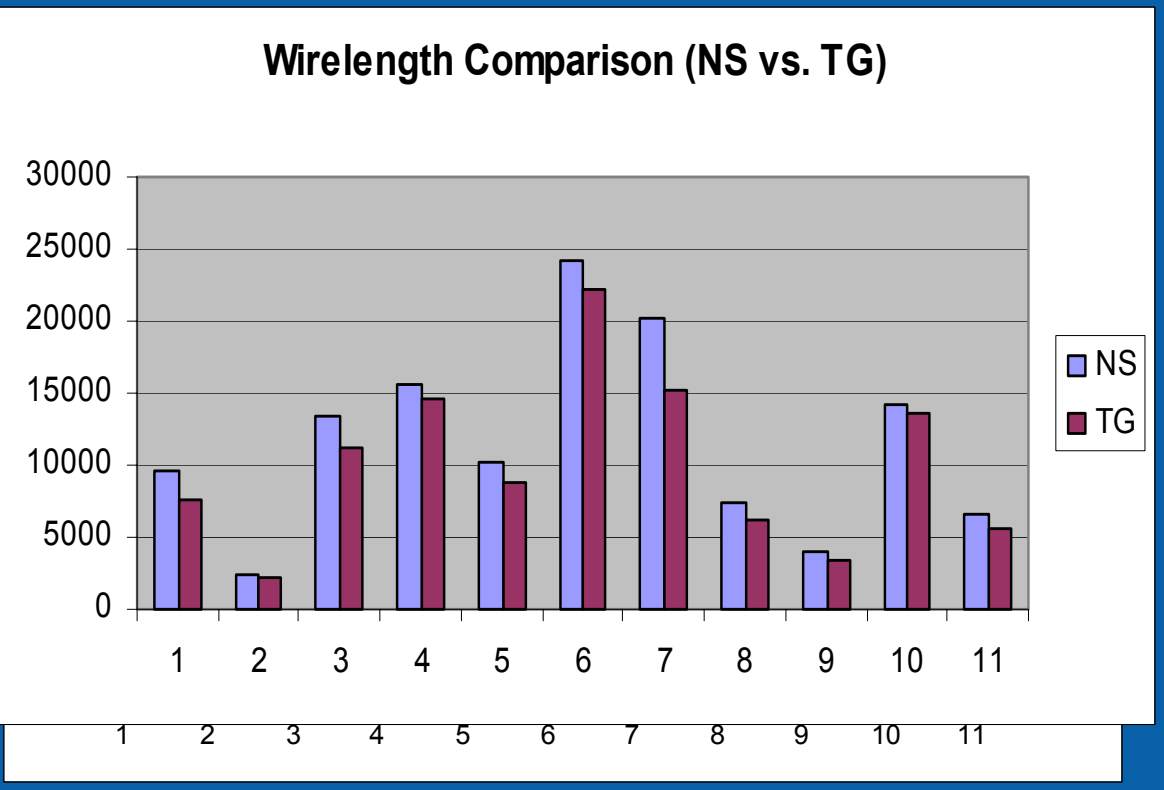
- Uses
  - grid-wires, M6/M7 track locations from full-chip layout
  - location and cap. on clock ports
- Considers distance and cap. constraints on routes from grid
  - Specified to ensure good slopes/delay/skew
- Generates routes obeying above constraints

Implementation Overview



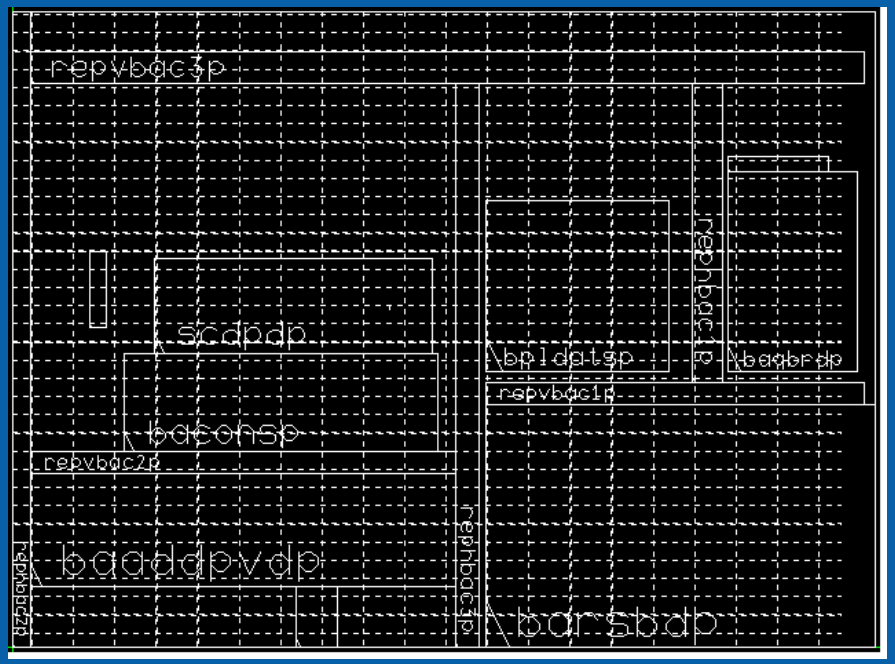
- Uses proprietary tool to read/update full-chip layout
- TCL interface to integrate the routing tool
- Tree Growing Heuristic: a solution for the problem of routing with capacitance and constraints

Wirelength/Power Comparison

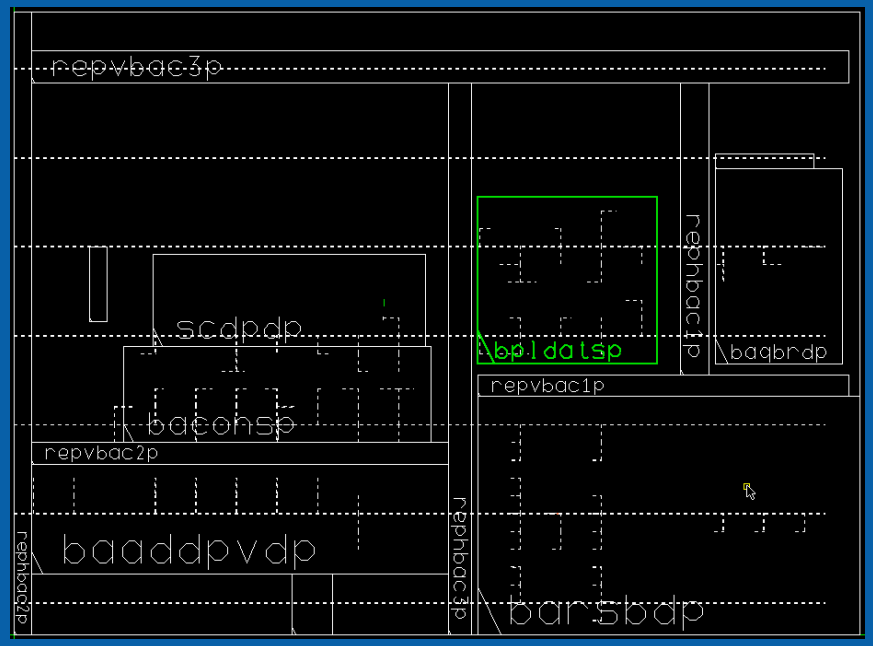


- Tree growing algorithm (TG) improves wirelength by 17%, on an average, over nearest source (NS) heuristic
- Practical run-times for both algorithms
- Deviations from ideal arrival time within ±4 ps
  - Mostly, due to the ports close to and far away from grid drivers

Global Clock Distribution: Layout Pictures

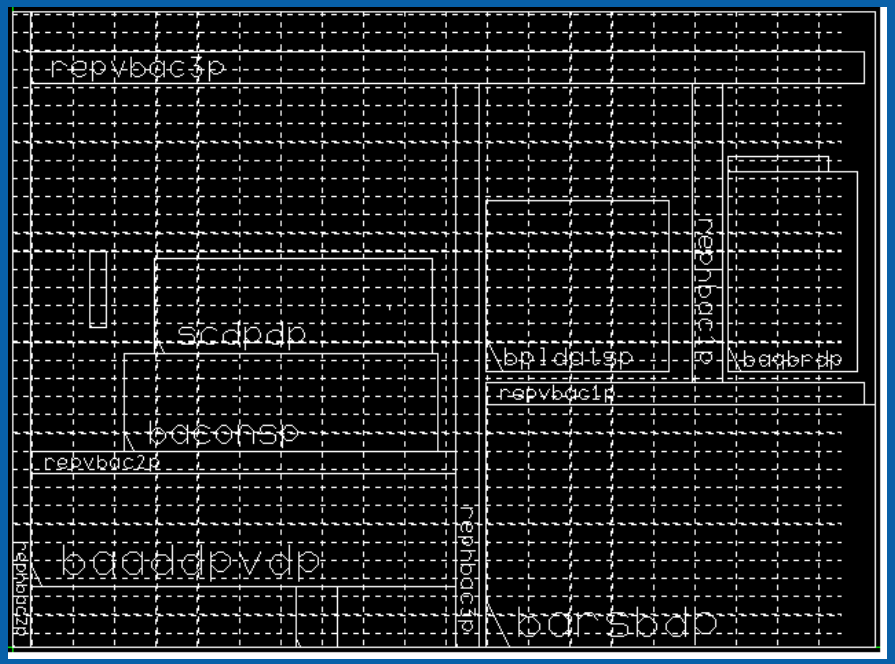


Layout area: with grid wires and tracks

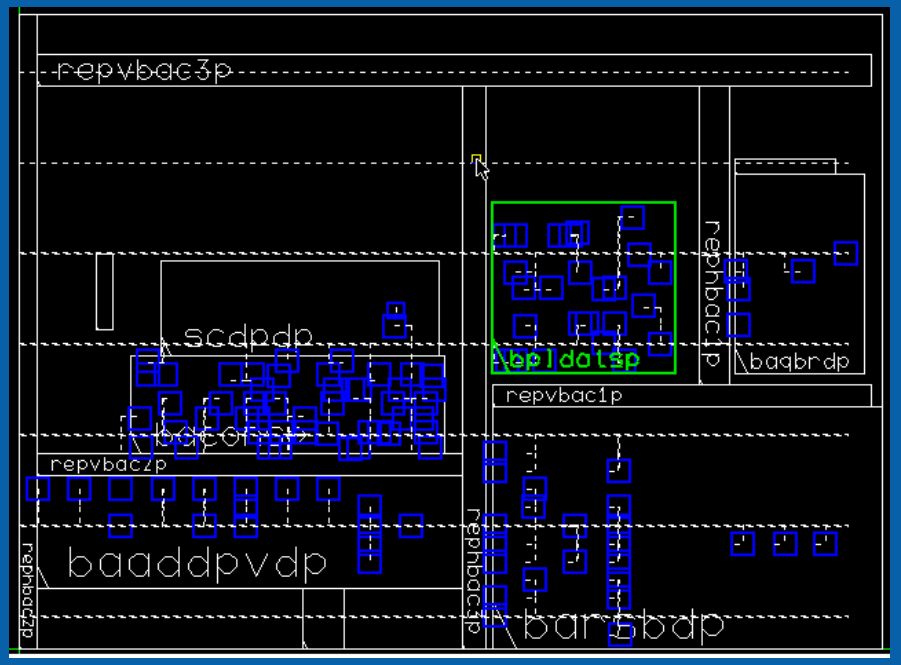


After global clock distribution

Global Clock Distribution: Layout Pictures

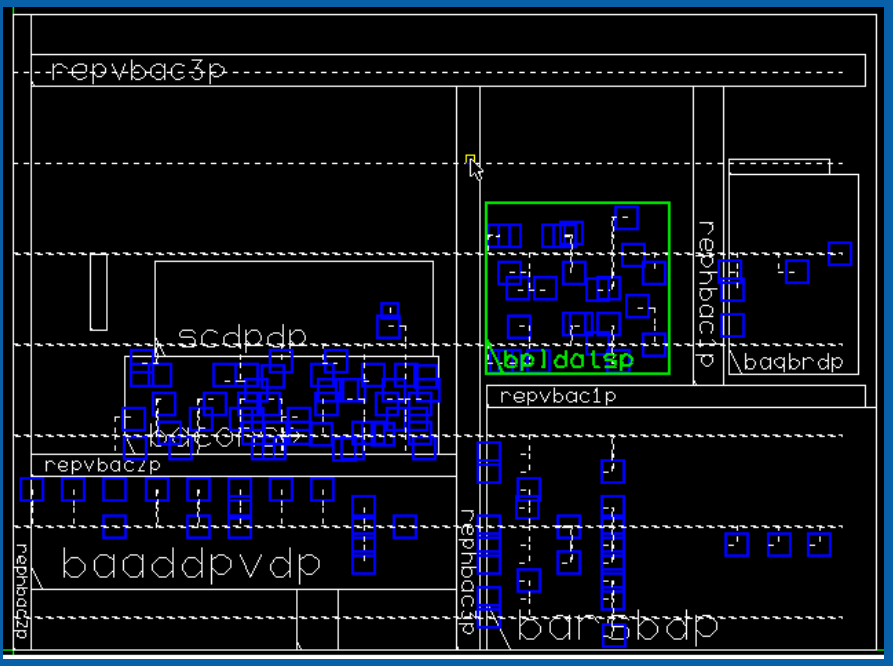


Layout area with grid wires and tracks



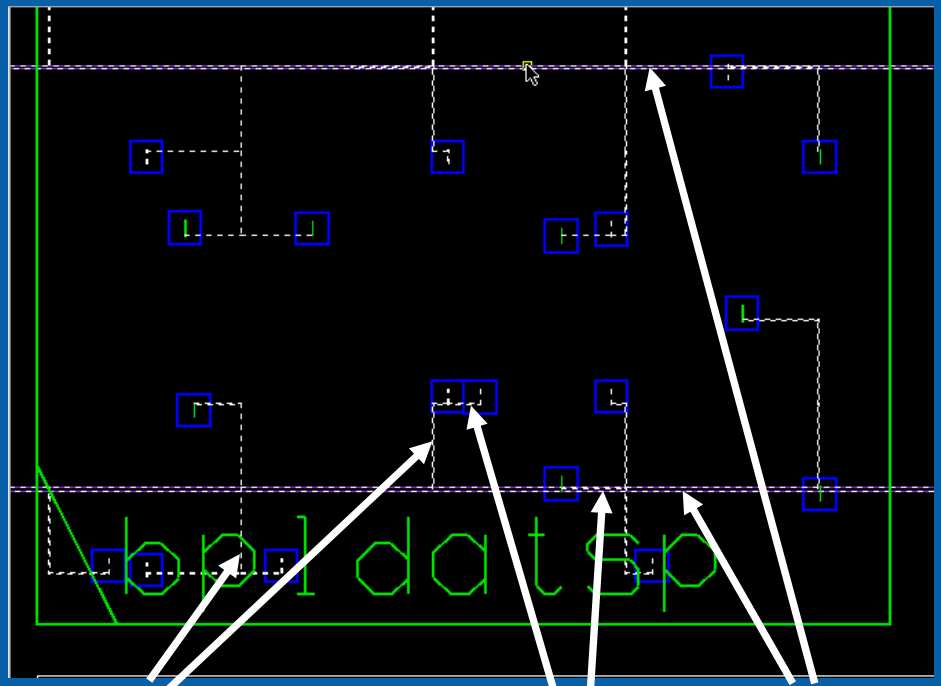
After global clock distribution
(with ports marked by blue squares)

Global Clock Distribution: Layout Pictures



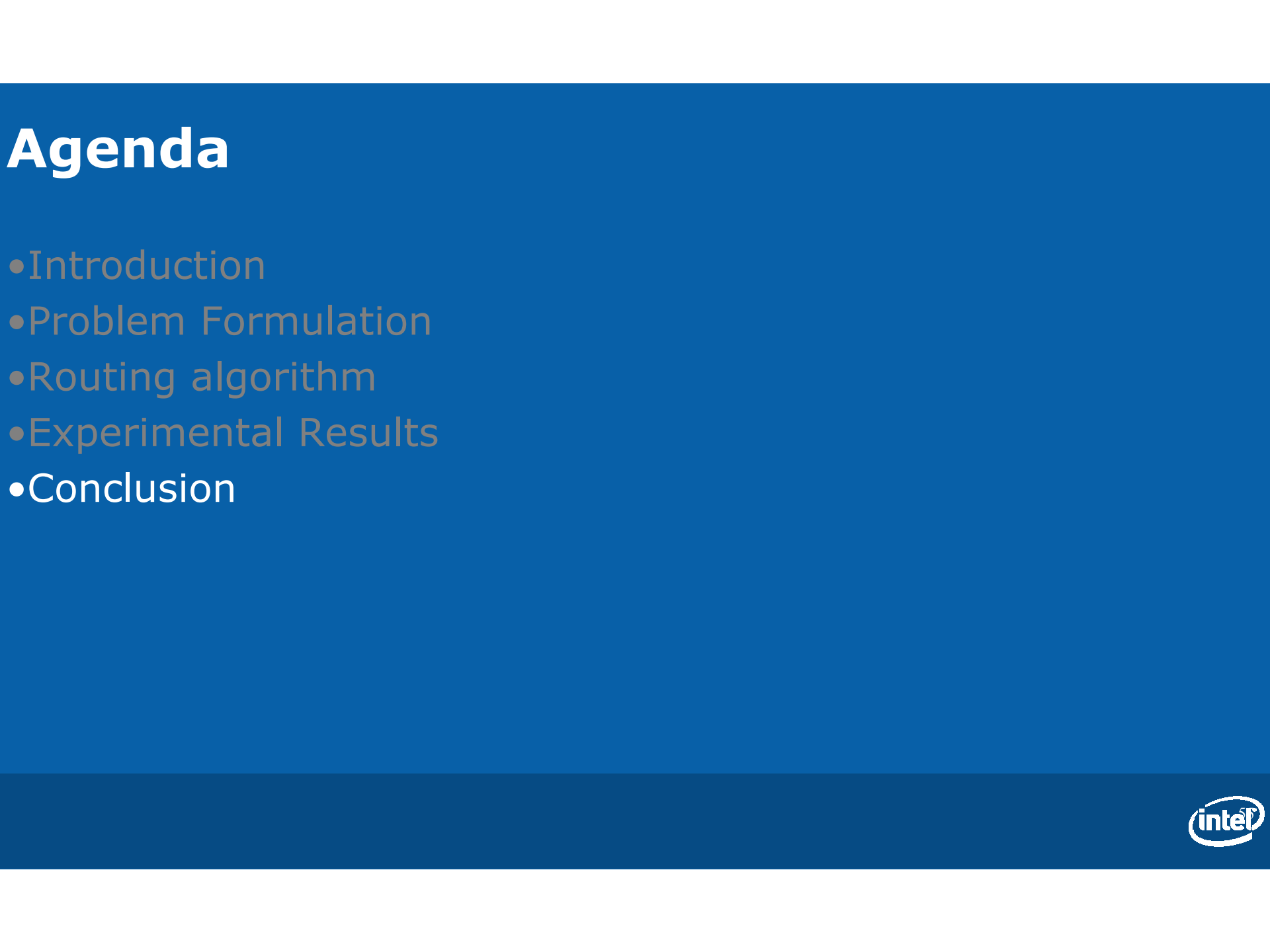
After global clock distribution
(with ports marked by blue squares)

Zoomed in picture (from slide 13)



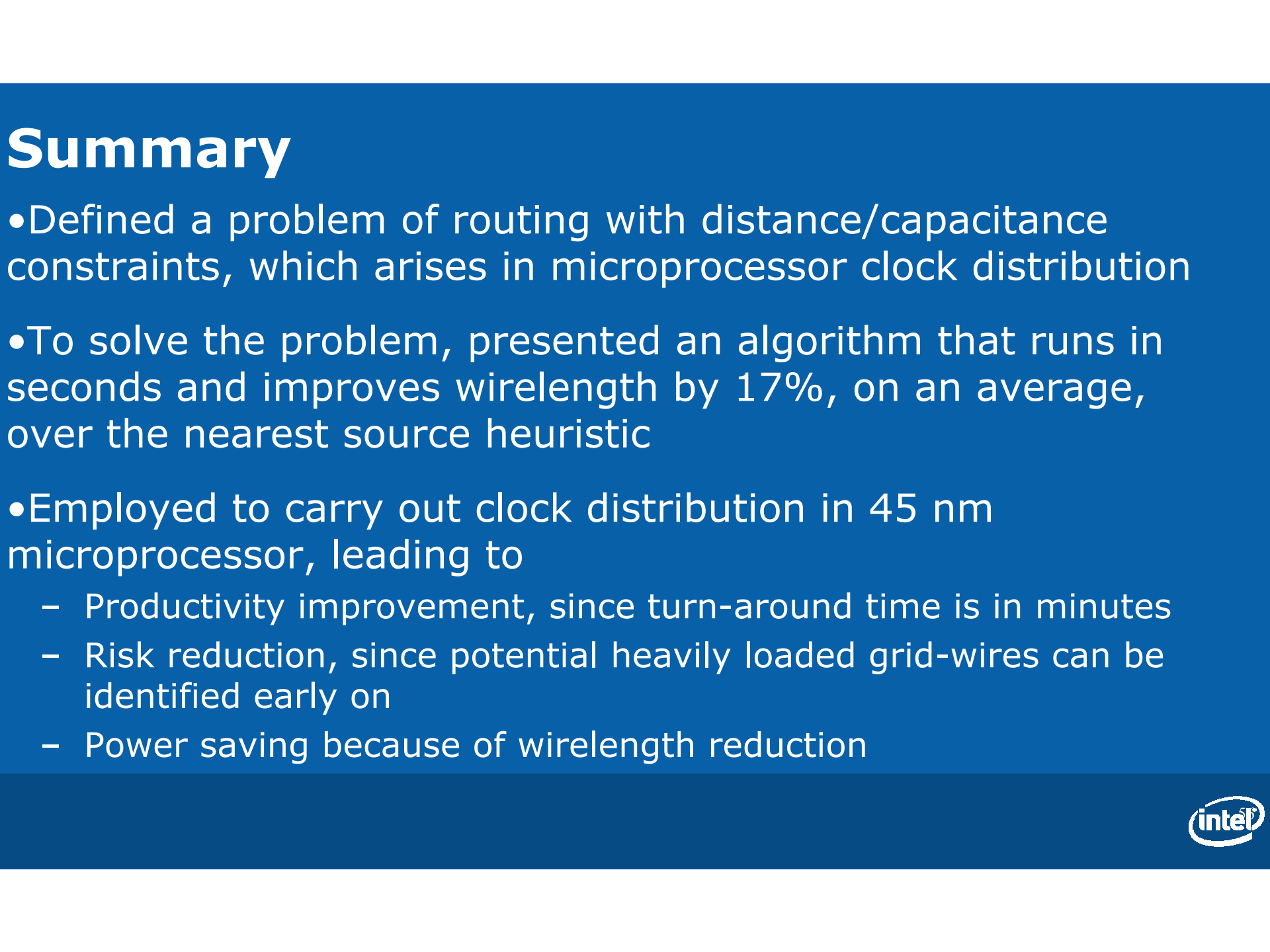
Agenda

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



Summary

- Defined a problem of routing with distance/capacitance constraints, which arises in microprocessor clock distribution
- To solve the problem, presented an algorithm that runs in seconds and improves wirelength by 17%, on an average, over the nearest source heuristic
- Employed to carry out clock distribution in 45 nm microprocessor, leading to
  - Productivity improvement, since turn-around time is in minutes
  - Risk reduction, since potential heavily loaded grid-wires can be identified early on
  - Power saving because of wirelength reduction



Future Directions

- Explore improvement possibilities
  - Better heuristics
  - Approximation algorithms
- Consider load limits on the grid wires
  - Leads to a problem similar to bin-packing

Q&A

Backup