CS137:
Electronic Design Automation

Day 11: February 18, 2004
Placement
(Intro, Constructive)

Today

- Placement Problem
- Partitioning→Placement
- Quadrisection
- Refinement

Placement

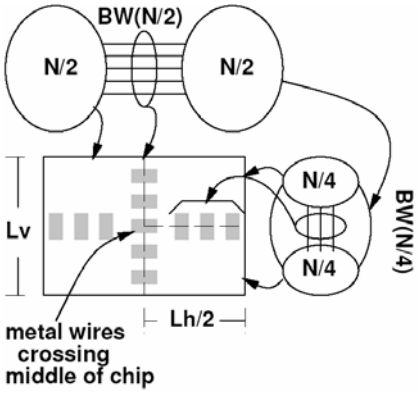
- **Problem**: Pick locations for all building blocks
  - minimizing energy, delay, area
  - really:
    - minimize wire length
    - minimize channel density

Bad Placement

- How bad can it be?
  - Area
  - Delay
  - Energy

Bad: Area

- All wires cross bisection
- $O(N^2)$ area
- good: O(N)

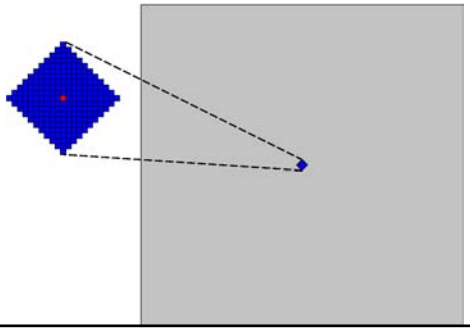


Bad: Delay

- All critical path wires cross chip
- Delay =$O(|PATH|*2*L_{side})$
  - [and L_{side} as O(N)]
- good: $O(|PATH|* L_{cell})$
- compare 50ps gates to many nanoseconds to cross chip

Clock Cycle Radius

- Radius of logic can reach in one cycle (45 nm)
  - Radius 10
    - Few hundred PEs
  - Chip side 600-700 PE
    - 400-500 thousand PEs
  - 100s of cycles to cross



Bad: Energy

- All wires cross chip:

  $O(L_{side})$ long $\rightarrow$ $O(L_{side})$ capacitance per wire
  - Recall Area$\rightarrow O(N^2)$
  - So $L_{side} \rightarrow O(N)$

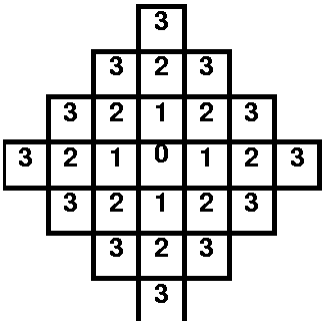
  $\times O(N)$ wires $\rightarrow O(N^2)$ capacitance

- Good:

  O(1) long wires $\rightarrow$ O(N) capacitance

Distance

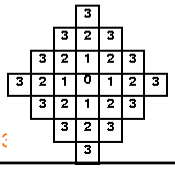
- Can we place everything close?

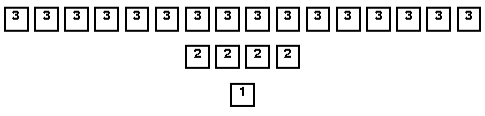


"Closeness"

- Try placing "everything" close

Manhattan Distance	Places	Transitive Fanin
1	4	4
2	8	16
3	12	64
⋮	⋮	⋮
n	$4n$	4^n





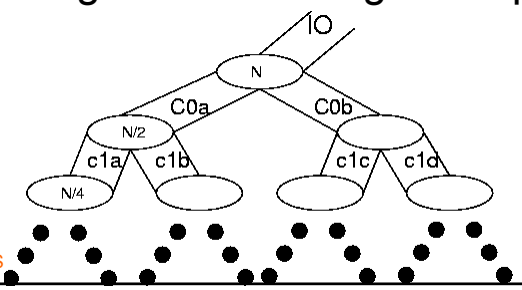
Problem Characteristics

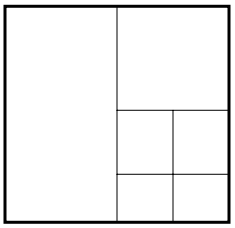
- Familiar
  - NP Complete
  - local, greedy not work
  - greedy gets stuck in local minima

Constructive Placement

Basic Idea

- Partition (bisect) to define halves of chip
  - minimize wire crossing
- Recurse to refine
- When get down to single component, done





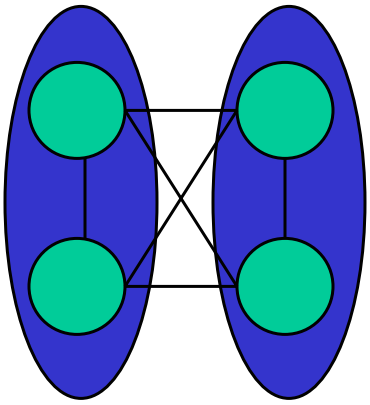
Adequate?

- Does recursive bisection capture the primary constraints of two-dimensional placement?

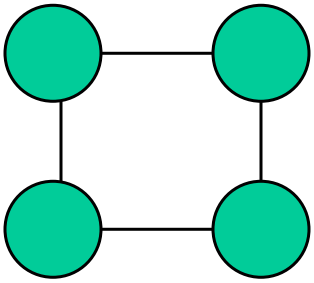
Problems

- Greedy, top-down cuts
  - maybe better pay cost early?
- Two-dimensional problem
  - (often) no real cost difference between H and V cuts
- Interaction between subtrees
  - not modeled by recursive bisect

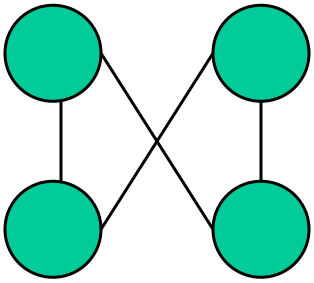
Interaction



Example

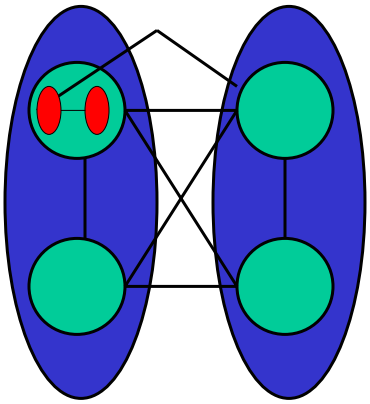


Ideal split
(not typical)



"Equivalent" split
ignoring external constraints
Practically -- makes all H
cuts also be V cuts

Interaction



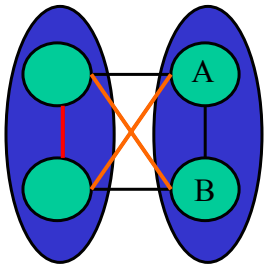
Problem

- Need to keep track of where things are
  - outside of current partition
  - include costs induced by above
- Don't necessarily know where things are
  - still solving problem

Improvement: Ordered

- Order operations
- Keep track of existing solution
- Use to constrain or pass costs to next subproblem

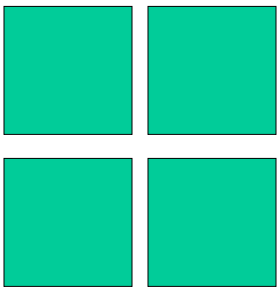
- Flow cut
  - use existing in src/sink
  - A nets = src, B nets = sink

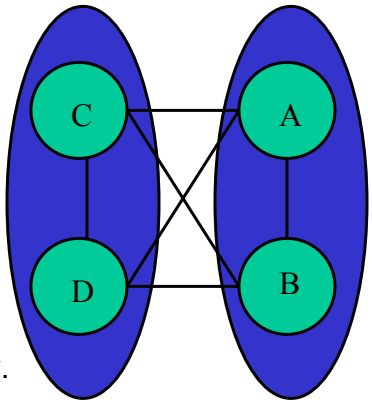


Improvement: Constrain

- Partition once
- Constrain movement within existing partitions
- Account for both H and V crossings
- Partition next
  - (simultaneously work parallel problems)
  - easy modification to FM

Constrain Partition





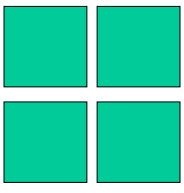
Solve AB and CD concurrently.

Improvement: Quadrisect

- Solve more of problem at once
- Quadrisection:
  - partition into 4 bins simultaneously
  - keep track of costs all around

Quadrisect

- Modify FM to work on multiple buckets
- k-way has:
  - k(k-1) buckets
  - |from|×|to|
  - quad→ 12
- reformulate gains
- update still O(1)

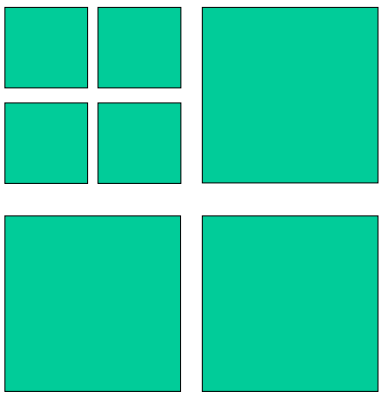


Quadrisect

- Cases (15):
  - (1 partition) x 4
  - (2 part) x 6 = (4 choose 2)
  - (3 part) x 4 = (4 choose 3)
  - (4 part) x 1

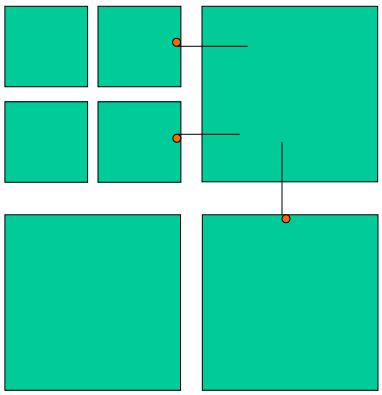
Recurse

- Keep outside constraints
  - (cost effects)
- Don't know detail place
- Model as at center of unrefined region



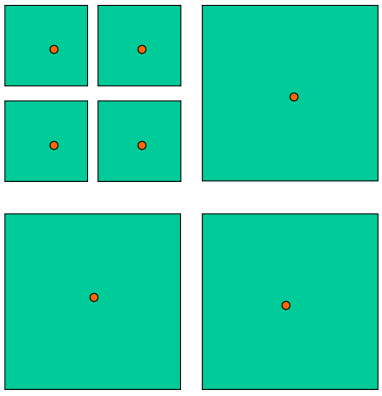
Option: Terminal Propagation

- Abstract inputs as terminals
- Partition based upon
- Represent cost effects on placement/refinement decisions

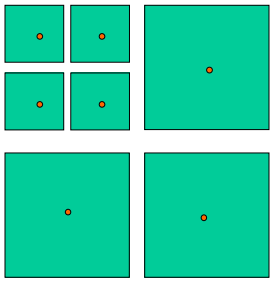


Option: Refine

- Keep refined placement
- Use in cost estimates



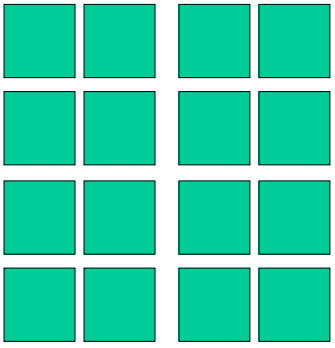
Problem



- Still have ordering problem
- Earlier subproblems solved with weak constraints from later
  - (cruder placement estimates)
- Solved previous case by flattening
  - …but in extreme give up divide and conquer

Iterate

- After solve later problems
- Relax solution
- Solve earlier problems again with refined placements (cost estimates)
- Repeat until converge

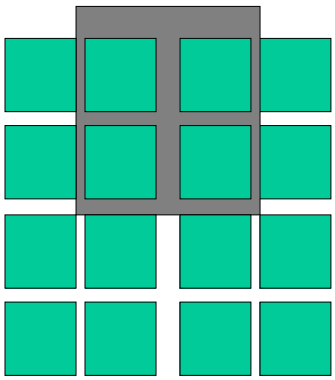


Iteration/Cycling

- General technique to deal with phase-ordering problem
  - what order do we perform transformations, make decisions?
  - How get accurate information to everyone
- Still basically greedy

Refinement

- Relax using overlapping windows
- Deal with edging effects
- Khang etc. claim 10-15% improve
  - cycle
  - overlap



Possible Refinement

- Allow unbalanced cuts
  - most things still work
  - just distort refinement groups
  - allowing unbalance using FM quadrisection looks a bit tricky
  - gives another 5-10% improvement

Runtime

- Each gain update still O(1)
  - (bigger constants)
  - so, FM partition pass still O(N)
- O(1) iterations expected
- assume O(1) overlaps exploited
- O(log(N)) levels

- Total: O(N log(N))
  - very fast compared to typical annealing
    - (annealing next time)

Uses

- Good by self
- Starting point for simulated annealing
  - speed convergence
- With synthesis (both high level and logic)
  - get a quick estimate of physical effects
  - (play role in estimation/refinement at larger level)
- Early/fast placement
  - before willing to spend time looking for best
- For fast placement where time matters
  - FPGAs, online placement?

Summary

- Partition to minimize cut size
- Additional constraints to do well
  - Improving constant factors
- Quadrisection
- Keep track of estimated placement
- Relax/iterate/Refine

Admin

- Homework #3 due Friday
- No class Monday
- Lecture on Wednesday by Wrighton

Big Ideas:

- Divide-and-conquer
- Successive Refinement
- Phase ordering: estimate/relax/iterate