CS137: Electronic Design Automation

Day 15: March 8, 2002

Routing 1

Today

- Custom/Semi-custom Routing
- Slicing
- Channel Routing
- Over-the-Cell/Multilayer

Routing Problem

- Where to wires run?
- Once know where blocks live,
  - where do the wires go?
  - In such a way as to:
    - Fit in fixed resources
    - Minimize resource requirements
      - (channel width → area)

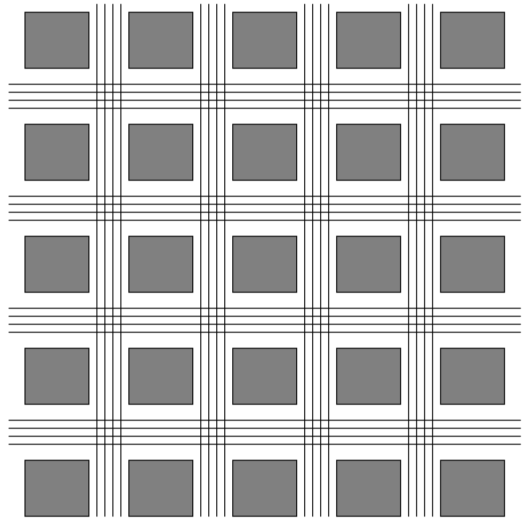
Variants

- Gate-Array
- Standard-Cell
- Full Custom

CALTECH CS137 Winter2004 -- DeHon

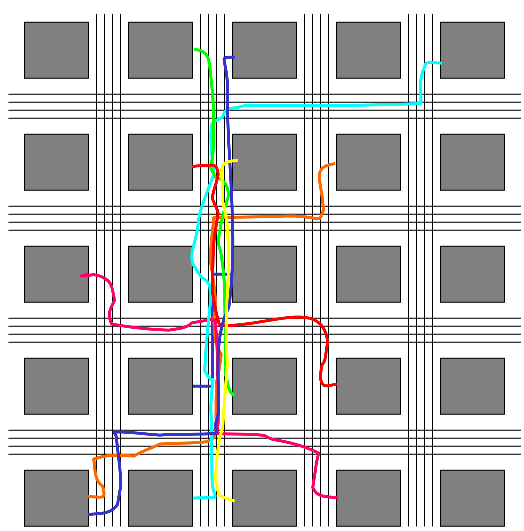
Gate Array

- Fixed Grid
- Fixed row and column width
- Must fit into array channel capacity



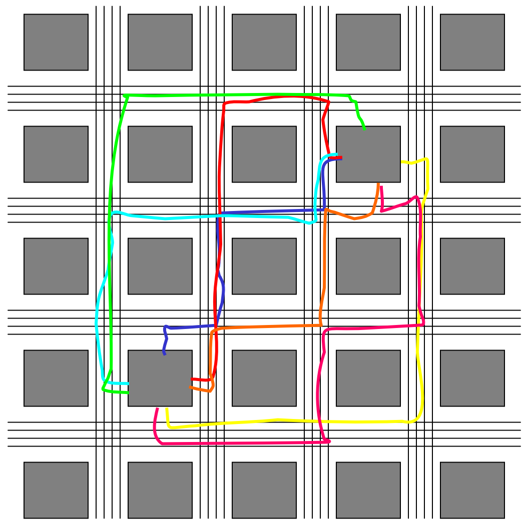
Gate Array

- Challenge
  - Everyone can't use same channel



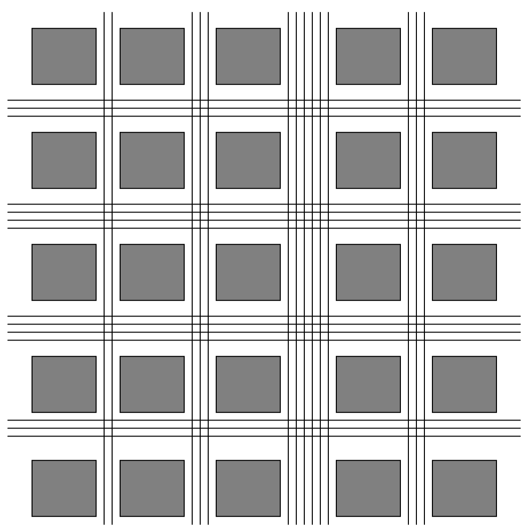
Gate Array

- Opportunities
  - Choice in paths
  - How exploit freedom to:
    - Meet channel limits
    - Minimize channel width



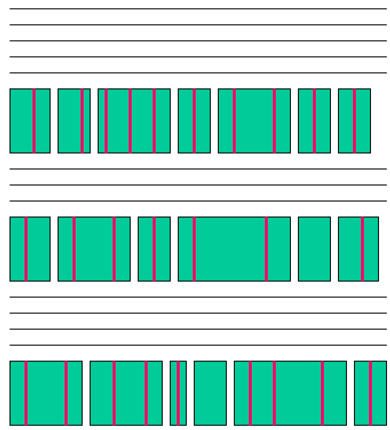
Semicustom Array

- Float Channel widths as needed
- Becomes a questions of minimizing total channel widths



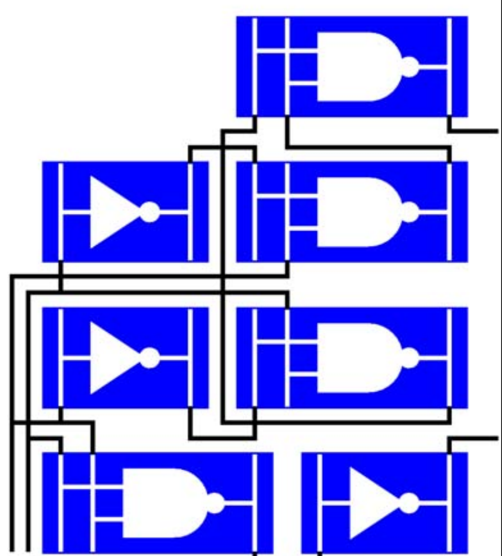
Row-based Standard Cell

- Variable size
  - Cells
  - Channels
- Primary route within row
- Vertical feed throughs



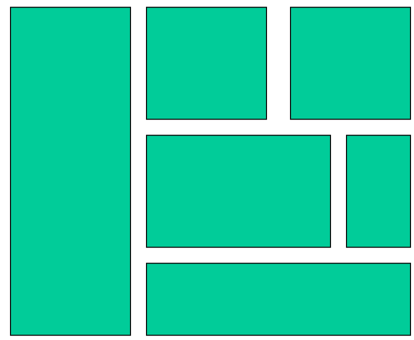
Standard Cell Gates

- IOs on one or both sides
- Design in Feed-thru



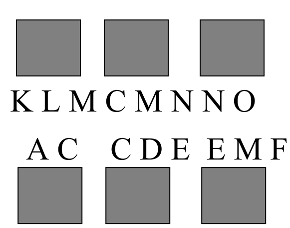
Full Custom / Macroblock

- Allow arbitrary geometry
  - Place larger cells
    - E.g. memory
  - Datapath blocks



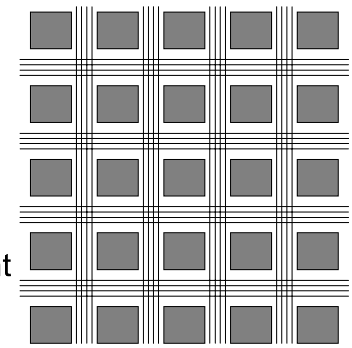
Channel Routing

- Key subproblem in all variants
- Psuedo 1D problem
- **Given**: set of terminals on one or both sides of channel
- Assign to tracks to minimize channel width



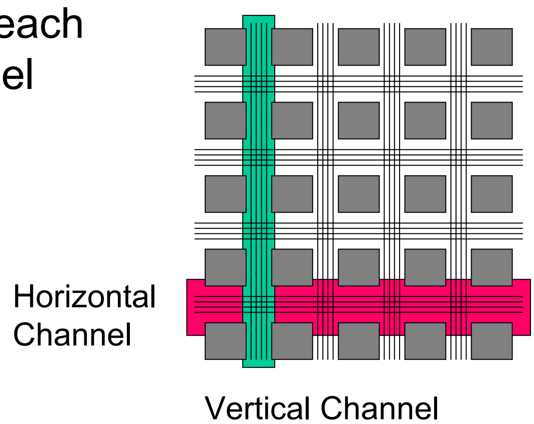
Gate Array → Channel

- Global route first
  - Decide which path each signal takes
  - Sequence of channels
  - Minimize congestion
    - Wires per channel segment



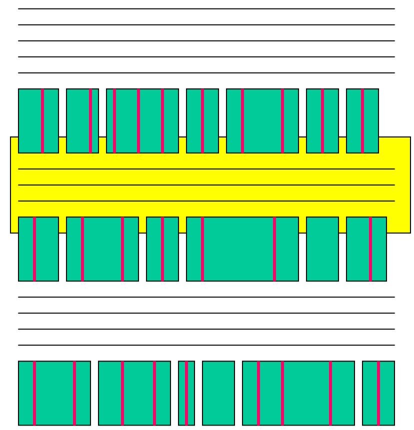
Gate Array → Channel

- Channel route each resulting channel



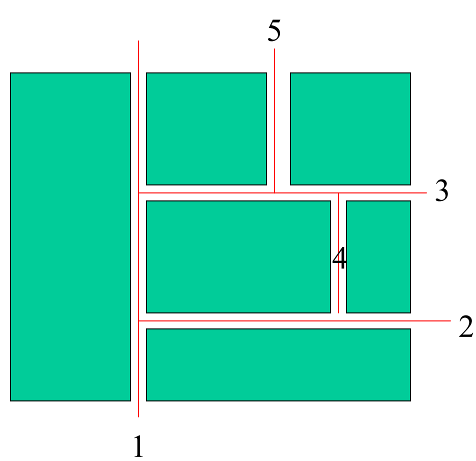
Std.Cell → Channel Route

- Plan feed through
- Channel route each row



Macroblock→Channel Route

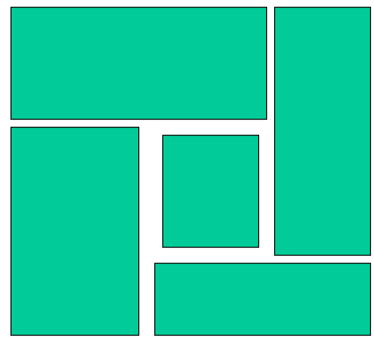
- **Slice** into pieces
- Route each as channel
- Work inside out
- Expand channels as needed
- Complete in one pass



CALTECH CS137 Winter2004 -- DeHon

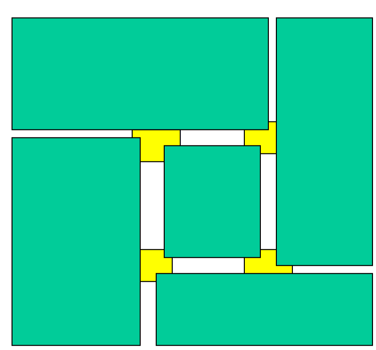
Not all Assemblies Sliceable

- No horizontal or vertical slice will separate
- Prevents ordering so can route in one pass



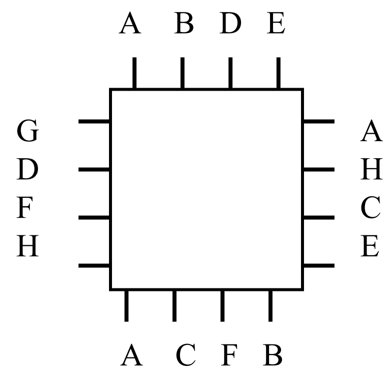
Switchbox Routing

- Box with 3 or 4 sides fixed
- Try to route signals with
- Identify in macroblock…



Switchbox Route

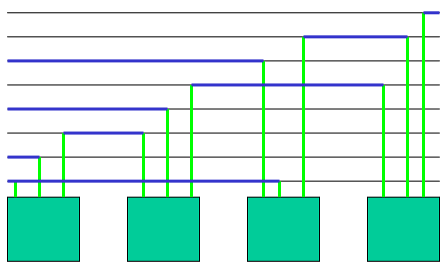
- Terminals on 4 sides
- Link up terminal



Channel Routing

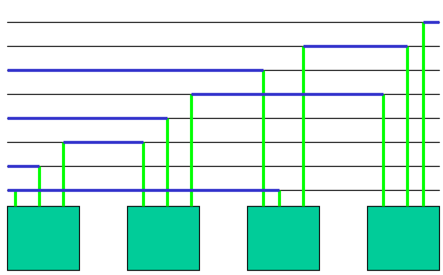
Trivial Channel Routing

- Assign every net its own track



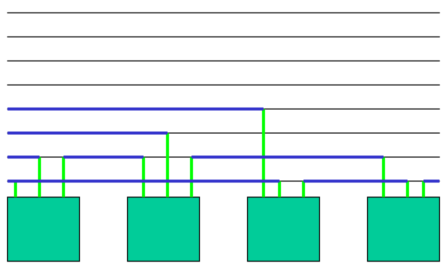
Trivial Channel Routing

- Assign every net its own track
  - Channel width > N (single output functions)
  - Chip bisection $\propto$N → chip area N^2



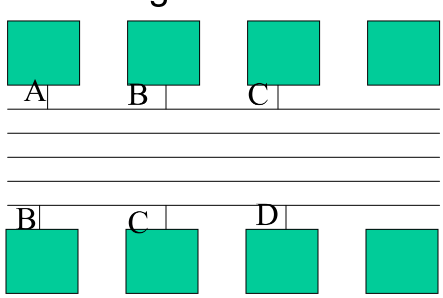
Sharing Tracks

- Want to Minimize tracks used
- Trick is to share tracks



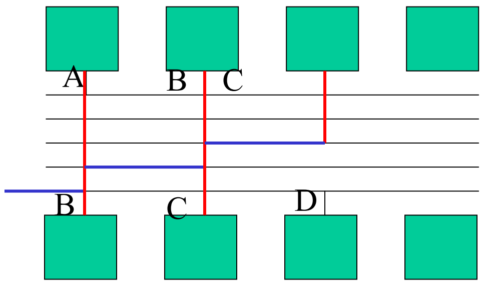
Not that Easy

- With Two sides
  - Even assigning one track/signal may not be enough



Not that Easy

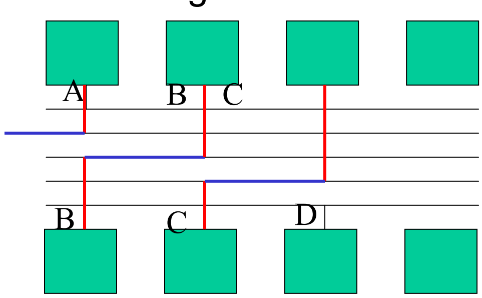
- With Two sides
  - Even assigning one track/signal may not be enough



Bad assignment
Overlap:
A,B
B,C

Not that Easy

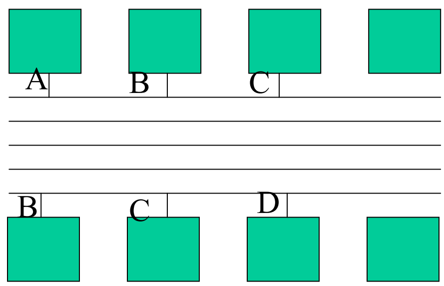
- With Two sides
  - Even assigning one track/signal may not be enough



Valid assignment avoids overlap

Not that Easy

- With Two sides
  - Even assigning one track/signal may not be enough



There are vertical constraints on ordering

Vertical Constraints

- For vertically aligned pins:
  - With single "vertical" routing layer
  - Cannot have distinct top pins on a lower track than bottom pins
    - Leads to vertical overlap
  - Produces constraint that top wire be higher track than lower
  - Combine across all top/bottom pairs
    - Leads to a Vertical Constraint Graph (VCG)

Channel Routing Complexity

- With Vertical Constraints
  - Problem becomes NP-complete
- Without Vertical Constraints
  - Can be solved optimally
  - Tracks = maximum channel density
  - Greedy algorithm

No Vertical Constraints

- Single-sided channel
  - (no top and bottom pins)
- Three layers for routing
  - Two vertical channels allow top and bottom pins to cross
  - May not be best way to use 3 layers…

Left-Edge Algorithm

1. Sort nets on leftmost end position
2. Start next lowest track; end=0
3. While there are unrouted nets with lowest left position > end of this track
   - Select unrouted net with lowest left position > end
   - Place selected net on this track
   - Update end position on this track to be end position of selected net
4. If nets remain, return to step 2

Greedy, optimal.

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Nets:
  - 0:1—5
  - 1:2—4
  - 2:5—6
  - 3:2—6
  - 4:4—7
  - 5:3—7
  - 6:1—3

FUTURE:
rename nets
to alpha
Note: show up
as numbers in
conv. Channel
routing file
formats.

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Nets:
  - 0:1—5
  - 1:2—4
  - 2:5—6
  - 3:2—6
  - 4:4—7
  - 5:3—7
  - 6:1—3
- Sort Left Edge:
  - 0:1—5
  - 6:1—3
  - 1:2—4
  - 3:2—6
  - 5:3—7
  - 4:4—7
  - 2:5—6

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Sort Left Edge:
  - 0:1—5
  - 6:1—3
  - 1:2—4
  - 3:2—6
  - 5:3—7
  - 4:4—7
  - 2:5—6
  - Track 0:
    - End 0
    - Add 0:1—5
    - End 5

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Sort Left Edge:
  - 6:1—3
  - 1:2—4
  - 3:2—6
  - 5:3—7
  - 4:4—7
  - 2:5—6

- Track 0: 0:1—5
- Track 1:
  - End 0
  - 6:1—3
  - End 3
  - 4: 4—7
  - End 7

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Sort Left Edge:
  - 1:2—4
  - 3:2—6
  - 5:3—7
  - 2:5—6

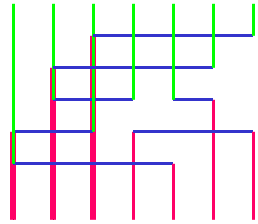
- Track 0: 0:1—5
- Track 1: 6:1—3, 4:4—7
- Track 2:
  - End 0
  - 1:2—4
  - End 4
  - 2:5—6
  - End 6

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Sort Left Edge:
  - 3:2—6
  - 5:3—7
  - Track 0: 0:1—5
  - Track 1: 6:1—3, 4:4—7
  - Track 2: 1:2—4, 2:5—6
  - Track 3: 3:2—6
  - Track 4: 5:3—7

Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Track 0: 0:1—5
- Track 1: 6:1—3, 4:4—7
- Track 2: 1:2—4, 2:5—6
- Track 3: 3:2—6
- Track 4: 5:3—7

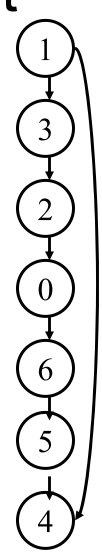


Constrained Left-Edge

1. Construct VCG
2. Sort nets on leftmost end position
3. Start new track; end=0
4. While there are nets that have
   - ✓ No descendents in VCG
   - ✓ And left edge > end
     1. Place net on track and update end
     2. Delete net from list, VCG
5. If there are still nets left to route, return to 2

Example: Constrained Left-Edge

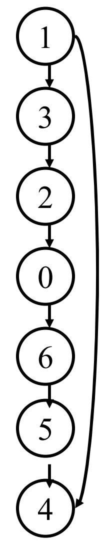
- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Nets:
  - 0:1—5
  - 1:2—4
  - 3:2—6
  - 3:5—6
  - 4:4—7
  - 5:3—7
  - 6:1—3
- Vertical Constraints
  - 0→6
  - 1→3
  - 6→5
  - 1→4
  - 2→0
  - 3→2
  - 5→4



Example: …

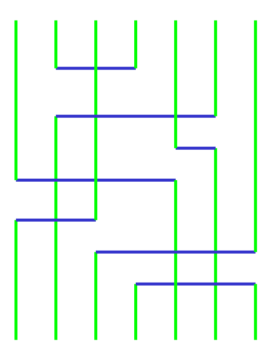
- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- Sort Left Edge:
  - 0:1—5
  - 6:1—3
  - 1:2—4
  - 3:2—6
  - 5:3—7
  - 4:4—7
  - 2:5—6

- Track 0:
  - 4:4—7
- Track 1:
  - 5:3—7
- Track 2:
  - 6:1—3
- Track 3:
  - 0:1—5
- Track 4:
  - 2:5—6
- Track 5:
  - 3:2—6
- Track 6:
  - 1:2--4



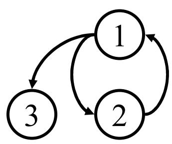
Example: Left-Edge

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4
- 4:4—7
- 5:3—7
- 6:1—3
- 0:1—5
- 2:5—6
- 3:2—6
- 1:2--4



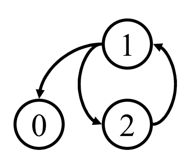
VCG Cycles

- Top: 1 1 2
- Bottom: 2 3 1
- VCG:



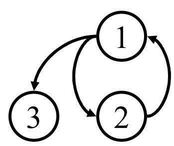
VCG Cycles

- No channel ordering satisfies VCG
- Must relax **artificial** constraint of single horizontal track per signal
- **Dogleg**: split horizontal run into multiple track segments
- In general, can reduce track requirements

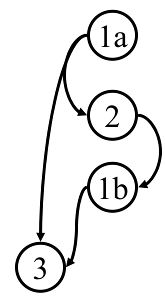


Dogleg Cycle Elimination

- Top: 1 1 2
- Bottom: 2 3 1
- VCG:

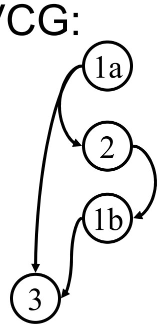


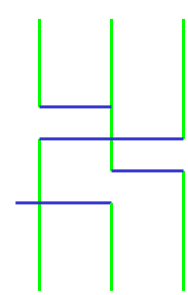
- Top: 1a 1a/1b 2
- Bottom: 2 3 1b
- VCG:



Dogleg Cycle Elimination

- Top: 1a 1a/1b 2
- Bottom: 2 3 1b
- VCG:



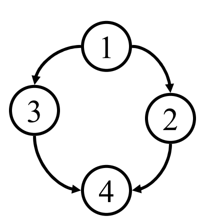


Dogleg Algorithm

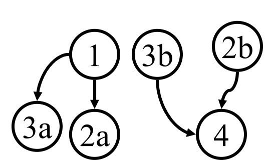
1. Break net into segments at pin positions
2. Build VCG based on segments
3. Run constrained on segments rather than full wires

Dogleg Example

- Top: 1 1 2 - 2 3
- Bottom: 2 3 - 3 4 4

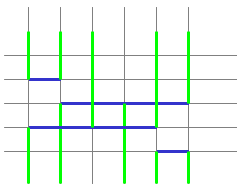


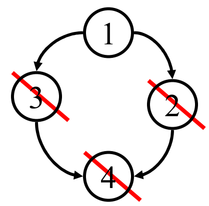
1 1 2a/2b – 2b 3b
2a 3a – 3a/b 4 4



No Dogleg

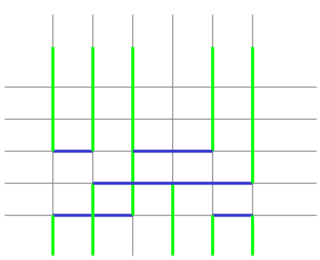
- Top: 1 1 2 - 2 3
- Bottom: 2 3 - 3 4 4

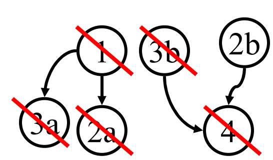




With Dogleg

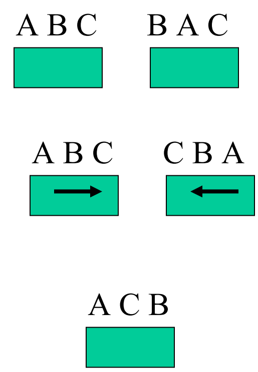
- Top: 1 1 2a/2b – 2b 3b
- Bottom: 2a 3a – 3a/b 4 4





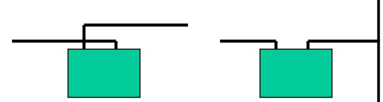
Other Freedoms

- Swap equivalent pins
  - *E.g.* nand inputs equivalent
- Mirror cells
  - if allowed electrically
- Choose among cell instances
  - Permute pins

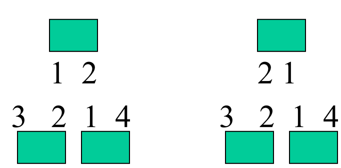


Exploit Freedom To

- Reduce channel density



- Reduce/Eliminate vertical constraints
  - Cycles
  - VCG height

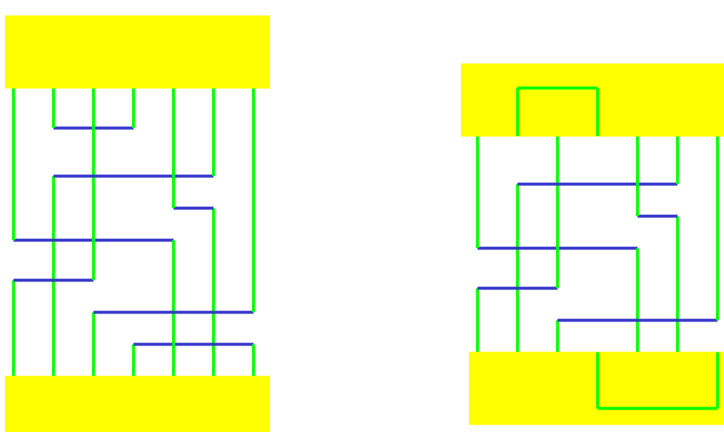


Over The Cell

- Limit cell to lower metal
  - Maybe only up to M1
- Can route over with higher metal

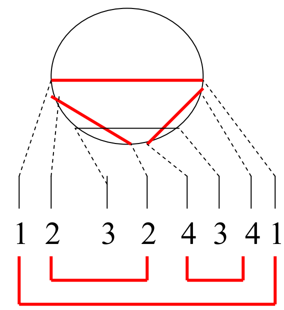
Example: OTC

- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4



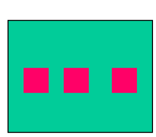
Over The Cell

- Compute maximal independent set
  - To find nets can be routed in 1 layer (planar) over cell
  - MIS can be computed in $O(n^2)$ time with dynamic programming
- Then route residual connections in channel
- Works on 2-metal if only M1 in cell
  - Feedthrus in M1

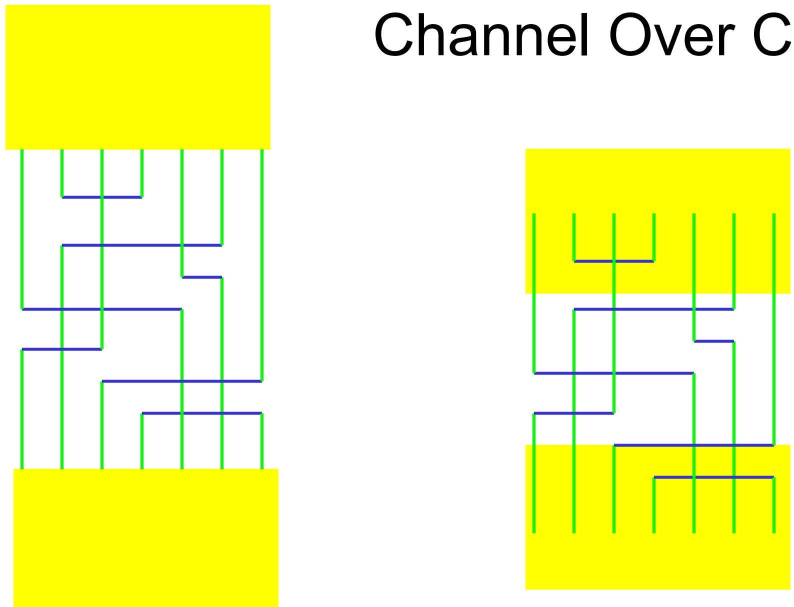


Multilayer

- With 3 layer
  - Can run channel over cells
  - Put Terminals in center of cell



Channel Over Cell



Route Over Cells

- If channel width < cell height
  - Routing completely on top of cells
- If channel width > cell height
  - Cell area completely hidden under routing channel
  - More typical case
    - Especially for large rows

Summary

- Decompose Routing
- Channel Routing
- Left-Edge
- Vertical Constraints
- Exploiting Freedom
  - Dogleg, pin swapping
- Routing over logic

Reminder

- Wednesday last lecture
  - EAS course feedback forms
- Read: Pathfinder for Wednesday
  - online

Big Ideas

- Decompose Problem
  - Divide and conquer
- Interrelation of components
- Structure: special case can solve optimally
- Technique: Greedy algorithm
- Use greedy as starting point for more general algorithm