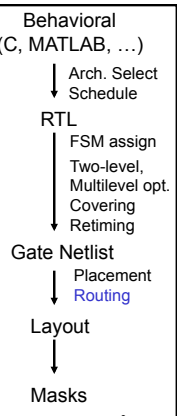
# ESE535: Electronic Design Automation

Day 13: March 3, 2015
Routing 1



## Today

- Routing Cases
- Routing Problem Decomposition
- Channel Routing
- Variations
  - Over-the-cell






## Routing Problem

Once know where blocks live (placement),

- How do we connect them up?
  - *i.e.* where do the wires go?
- In such a way as to:
  - Fit in fixed resources
  - Minimize resource requirements
    - (channel width → area)



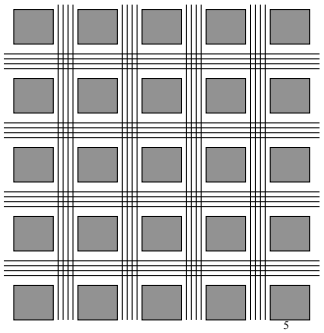
## Routing Cases

Gate Array
Standard Cell
Full Custom



## Gate Array

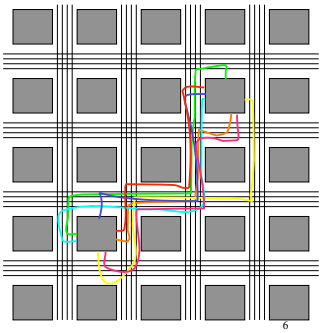
- Fixed Grid
- Fixed row and column width
- Must fit into prefab channel capacity
- Resource-constrainted routing





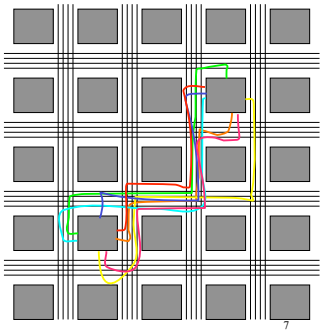
## Gate Array

- What freedom can we exploit in routing?



## Gate Array

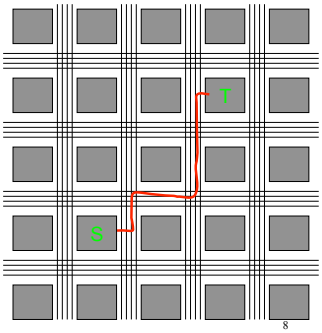
- Opportunities
  - Choice in paths
  - Exploit freedom to:
    - Meet channel limits
    - Minimize channel width





## Gate Array

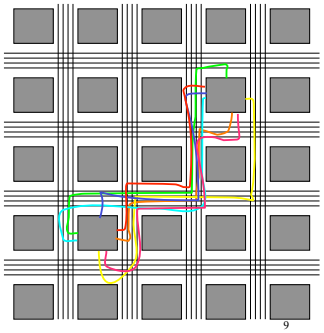
- What other paths could the red wire take?






## Gate Array

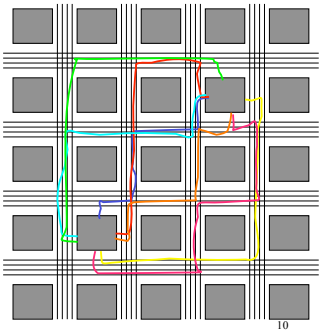
- Opportunities
  - Choice in paths
  - Exploit freedom to:
    - Meet channel limits
    - Minimize channel width





## Gate Array

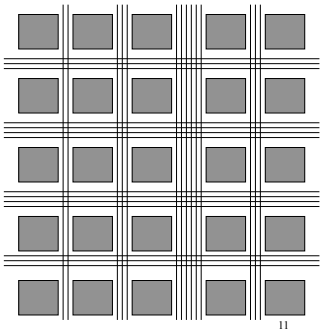
- Opportunities
  - Choice in paths
  - Exploit freedom to:
    - Meet channel limits
    - Minimize channel width





## Semicustom Array

- Float channel widths as needed
- How do we optimize area in this case?





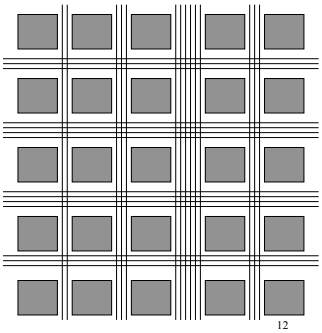
## Semicustom Array

- Float Channel widths as needed
- Area
  - minimize total channel widths

$A=H*V$

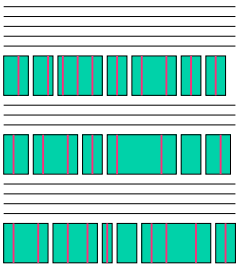
$H=\Sigma H_i$

$V=\Sigma V_i$



## Row-based Standard Cell

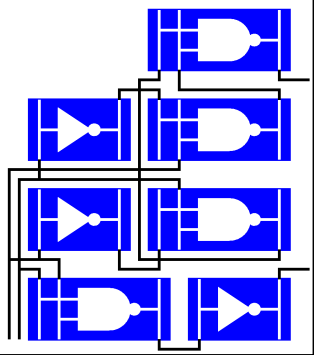
- Variable size
  - Cells
  - Channels
- Primary route within row
  - Minimize tracks in channel
- Vertical feed throughs





## Standard Cell Gates

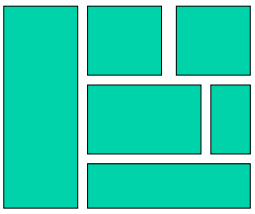
- IOs on one or both sides
- Design in Feed-thru





## Full Custom / Macroblock

- Allow arbitrary geometry
  - Place larger cells
    - E.g. memory
  - Datapath blocks
- Less regular, but still have channels…





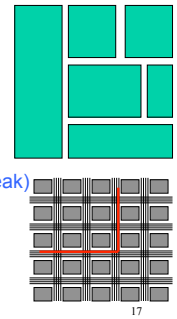
## Routing Decomposed



## Phased Routing

After placement…

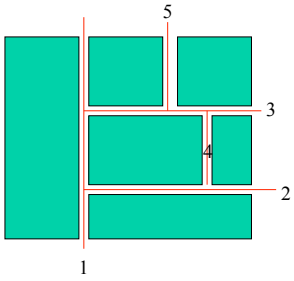
1. Slice (macroblock case)
   - And order channels
2. Global Route
   - Which channels to use
   - (suitable approach Monday after break)
3. Channel Route
   - Today
4. Switchbox Route





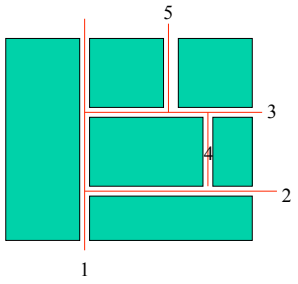
## Macroblock→Channel Route

- **Slice** into pieces
- Route each as channel
- Significance of numbers?



## Macroblock→Channel Route

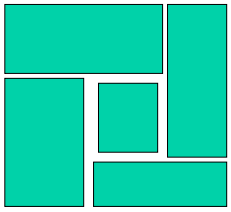
- **Slice** into pieces
- Route each as channel
- If work inside out
  - Can expand channels as needed
  - Complete in one pass






## Not all Assemblies Sliceable

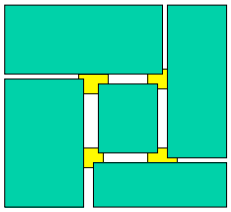
- No horizontal or vertical slice will separate
- Prevents ordering that allows us to route in one pass





## Switchbox Routing

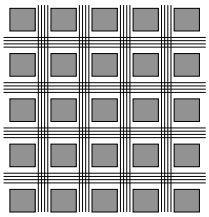
- Box with 3 or 4 sides fixed
- Contrast channel routing with only 2 sides fixed





## Gate Array → Channel

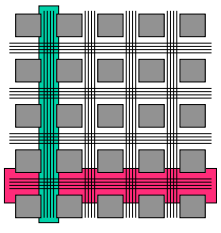
- Global route first
  - Decide which path each signal takes
  - Sequence of channels
  - Minimize congestion
    - Wires per channel segment





## Gate Array → Channel

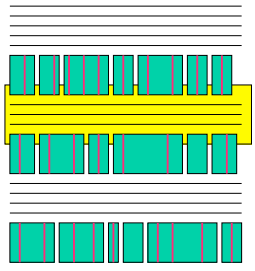
- Then Channel route each resulting channel





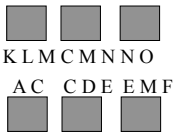

## Std.Cell → Channel Route

- Plan feed through
- Channel route each row



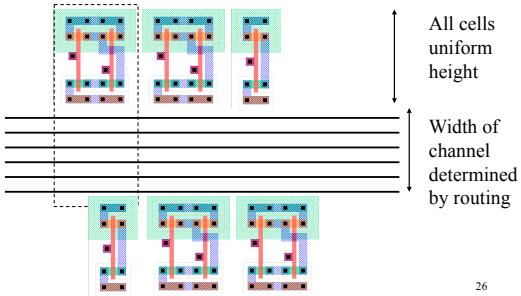
## Channel Routing

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



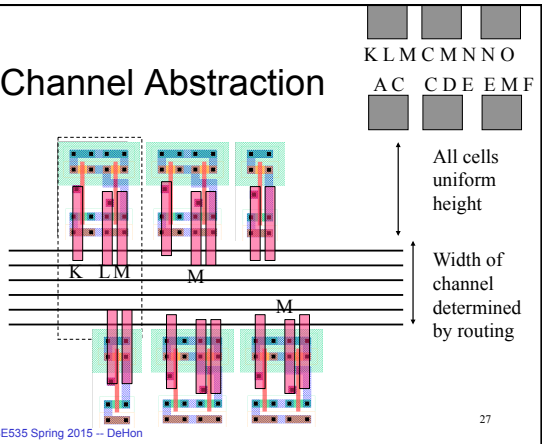



## Standard Cell Area






## Channel Abstraction

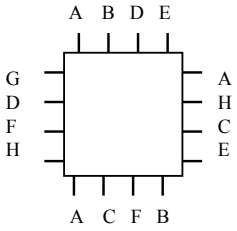





[since don't go any further with this, not clear worth showing]

## Switchbox Route

- Terminals on 4 sides
- Link up terminal



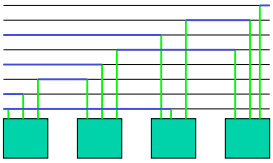



## Channel Routing



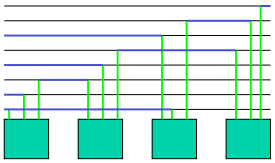
## Trivial Channel Routing

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



## Trivial Channel Routing

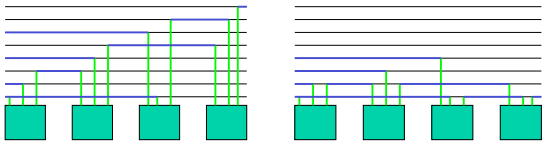
- How can we do better?
  - What do we want to exploit?





## Sharing Tracks

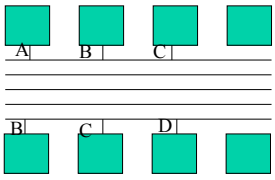
- Want to Minimize tracks used
- Trick is to share tracks





## Not that Easy

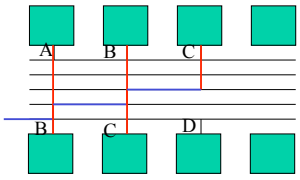
- With Two sides
  - Even assigning one track/signal may not be sufficient






## Not that Easy

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

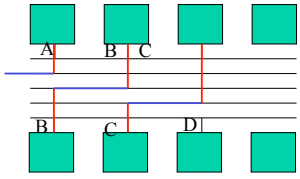



Bad assignment
Overlap:
A,B
B,C



## Not that Easy

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

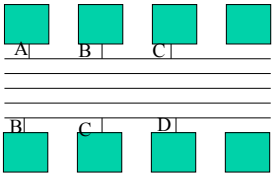



Valid assignment avoids overlap



## Not that Easy

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



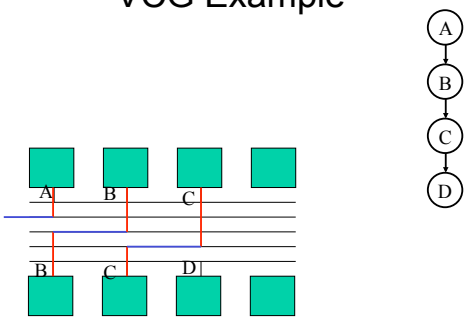

*i.e.* there are vertical constraints on ordering

## 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)



## VCG Example






## 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

Good for:

- 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

Note: nets (shown as letters here) show up as numbers in conv. channel routing file formats.

- Top: a b g b c d f
- Bottom: g d f e a c e
- Nets:
  - a:1—5
  - b:2—4
  - c:5—6
  - d:2—6
  - e:4—7
  - f:3—7
  - g:1—3

## Example: Left-Edge

- Top: a b g b c d f
- Bottom: g d f e a c e

Nets:
- a:1—5
- b:2—4
- c:5—6
- d:2—6
- e:4—7
- f:3—7
- g:1—3

- Sort Left Edge:
  - a:1—5
  - g:1—3
  - b:2—4
  - d:2—6
  - f:3—7
  - e:4—7
  - c:5—6



## Example: Left-Edge

- Top: a b g b c d f
- Bottom: g d f e a c e
- Sort Left Edge:
  - a:1—5
  - g:1—3
  - b:2—4
  - d:2—6
  - f:3—7
  - e:4—7
  - c:5—6

- Track 0:
  - End 0
  - Add a:1—5
  - End 5



## Example: Left-Edge

- Top: a b g b c d f
- Bottom: g d f e a c e

- Sort Left Edge:
  - g:1—3
  - b:2—4
  - d:2—6
  - f:3—7
  - e:4—7
  - c:5—6

- Track 0: a:1—5
- Track 1:
  - End 0
  - g:1—3
  - End 3
  - e: 4—7
  - End 7



## Example: Left-Edge

- Top: a b g b c d f
- Bottom: g d f e a c e

- Sort Left Edge:
  - b:2—4
  - d:2—6
  - f:3—7
  - c:5—6

- Track 0: a:1—5
- Track 1: g:1—3, e:4—7
- Track 2:
  - End 0
  - b:2—4
  - End 4
  - c:5—6
  - End 6



## Example: Left-Edge

- Top: a b g b c d f
- Bottom: g d f e a c e

- Sort Left Edge:
  - d:2—6
  - f:3—7

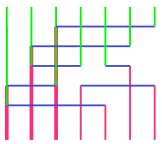
- Track 0: a:1—5
- Track 1: g:1—3, 4:e—7
- Track 2: b:2—4, c:5—6
- Track 3: d:2—6
- Track 4: f:3—7



## Example: Left-Edge

- Top: a b g b c d f
- Bottom: g d f e a c e

- Track 0: a:1—5
- Track 1: g:1—3, e:4—7
- Track 2: b:2—4, c:5—6
- Track 3: d:2—6
- Track 4: f:3—7



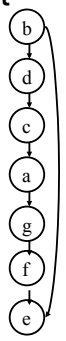
## Constrained Left-Edge

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: a b g b c d f
- Bottom: g d f e a c e
- Nets:
  - a:1—5
  - b:2—4
  - c:5—6
  - d:2—6
  - e:4—7
  - f:3—7
  - g:1—3
- Vertical Constraints [draw board]
  - a→g
  - b→d
  - g→f
  - b→e
  - c→a
  - d→c
  - f→e

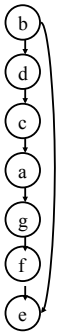





## Example: …

- Top: a b g b c d f
- Bottom: g d f e a c e
- Sort Left Edge:
  - a:1—5
  - g:1—3
  - b:2—4
  - d:2—6
  - f:3—7
  - e:4—7
  - c:5—6

- Track 0:
  - e:4—7
- Track 1:
  - f:3—7
- Track 2:
  - g:1—3
- Track 3:
  - a:1—5
- Track 4:
  - c:5—6
- Track 5:
  - d:2—6
- Track 6:
  - b:2—4





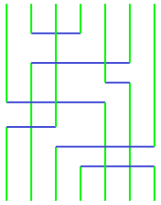

## Vertical Constraints

- Also give a lower bound on routed channel width
  - Channel width >= channel density
  - Channel width >= height of VCG graph



## Example: Left-Edge

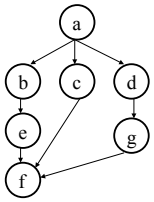
- Top: a b g b c d f
- Bottom: g d f e a c e
- Nets:
  - a:1—5
  - b:2—4
  - c:5—6
  - d:2—6
  - e:4—7
  - f:3—7
  - g:1—3



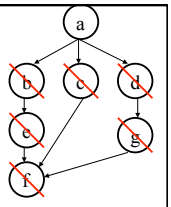


## Example 2: …

- Top: a a a b e d g c
- Bottom: b c d e f g f f
- Sort Left Edge:
  - b:1—4
  - a:1—3
  - c:2—8
  - d:3—6
  - e:4—5
  - f:5—8
  - c:6—7



## Example 2: …



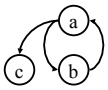

- Top: a a a b e d g c
- Bottom: b c d e f g f f
- Sort Left Edge:
  - ~~b:1—4~~
  - a:1—3
  - ~~c:2—8~~
  - ~~d:3—6~~
  - ~~e:4—5~~
  - ~~f:5—8~~
  - ~~g:6—7~~

Track 0: f
Track 1: c
Track 2: e, g
Track 3: b
Track 4: d
Track 5: a



## VCG Cycles

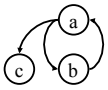
- Top: a a b
- Bottom: b c a
- VCG:






## VCG Cycles

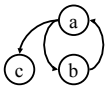
- No channel ordering satisfies VCG
- Must relax **artificial** constraint of single horizontal track per signal
- **Dogleg**: split horizontal run into multiple track segments
- In general, can reduce track requirements



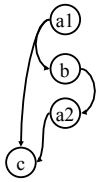



## Dogleg Cycle Elimination

- Top: a a b
- Bottom: b c a
- VCG:



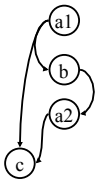

- Top: a1 a1/a2 b
- Bottom: b c a2
- VCG:

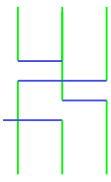





## Dogleg Cycle Elimination

- Top: a1 a1/a2 b
- Bottom: b c a2
- VCG:



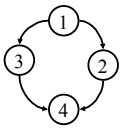


## Dogleg Algorithm

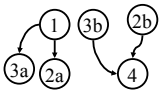
1. Break net into segments at pin positions
2. Build VCG based on segments
3. Run constrained on segments rather than full wires

## Dogleg Example (no cycle)

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




1 1 2a/2b – 2b 3b
2a 3a – 3a/b 4 4

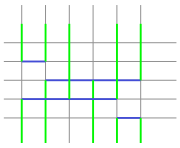


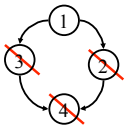

[note: switch to numbers for terminals]



## No Dogleg

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

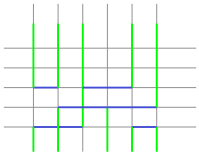


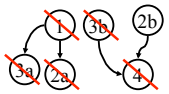





## With Dogleg

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





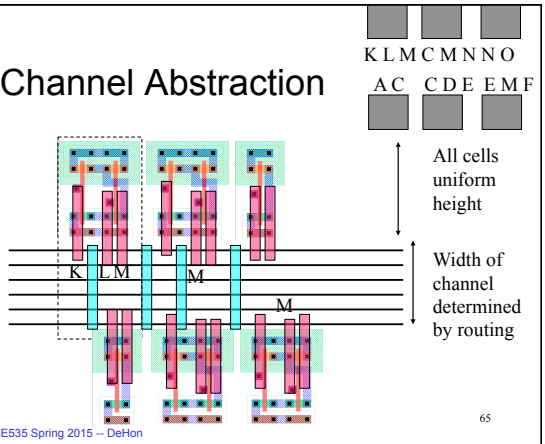



## Doglegs

- Exploiting dogleg
  - Introduces more freedom
  - Can reduce track requirements
    - Reduces height of VCG
- In general, any unused vertical track could support some dogleg



## Channel Abstraction



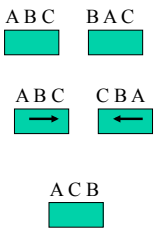



## Doglegs

- Exploiting dogleg
  - Introduces more freedom
  - Can reduce track requirements
- In general, any unused vertical track could support some dogleg
  - How select which signal uses track for dogleg?
    - Creates a larger optimization problem
  - Might support multiple?

## Other Freedoms

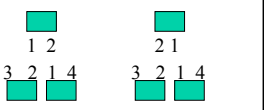
- Swap equivalent pins
  - *E.g.* nand inputs equivalent
- Mirror cells
  - if allowed electrically
- Choose among cell instances
  - Permute pins






## Exploit Freedom To

- Reduce channel density
- Reduce/Eliminate vertical constraints
  - Cycles
  - VCG height





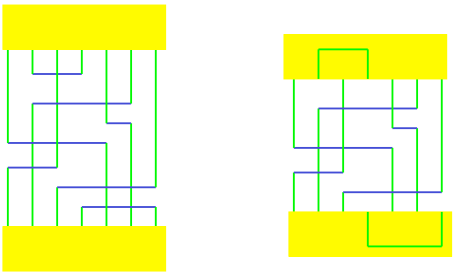

## Over The Cell

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



## Example: OTC

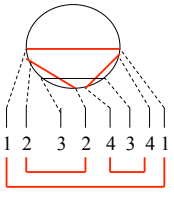
- Top: 0 1 6 1 2 3 5
- Bottom: 6 3 5 4 0 2 4





## 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

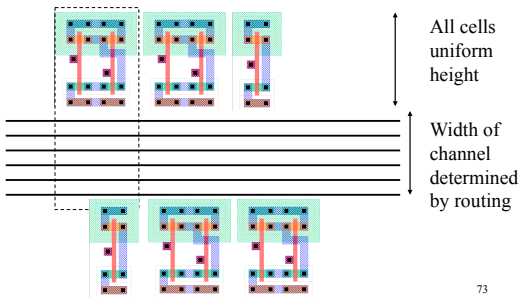





## Multilayer

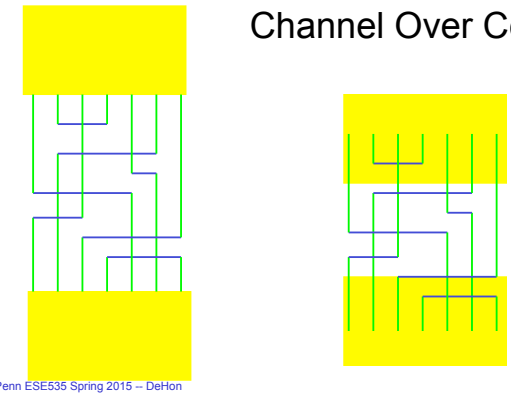
- With 3 layer
  - Can run channel over cells
  - Put Terminals in center of cell

## Standard Cell Area






## 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



## 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



## Admin

- Next week Spring Break
- Reading for Monday after break online
- Andre out week after break
  - Monday: Guest lecture
  - Wednesday: No class (no midterm)
- Assign 7, 8 on routing out