EE 5301 – VLSI Design Automation I

Part VI: Routing

Kia Bazargan

University of Minnesota

References and Copyright

- Textbooks referred (none required)
  - [Mic94] G. De Micheli
    "Synthesis and Optimization of Digital Circuits"
    McGraw-Hill, 1994.
  - [CLR90] T. H. Cormen, C. E. Leiserson, R. L. Rivest
    "Introduction to Algorithms"
    MIT Press, 1990.
  - [Sar96] M. Sarrafzadeh, C. K. Wong
    "An Introduction to VLSI Physical Design"
    McGraw-Hill, 1996.
  - [She99] N. Sherwani
    "Algorithms For VLSI Physical Design Automation"
    Kluwer Academic Publishers, 3rd edition, 1999.

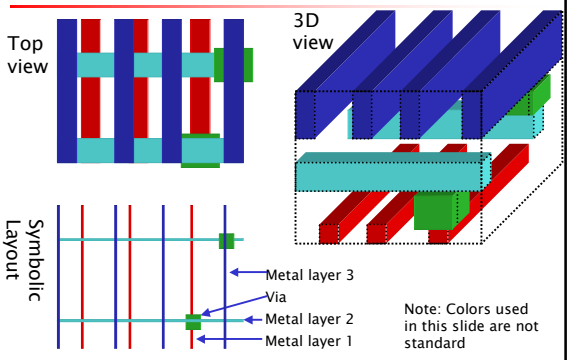
References and Copyright (cont.)

- Slides used: (*Modified by Kia when necessary*)
  - [©Sarrafzadeh] © Majid Sarrafzadeh, 2001;
    Department of Computer Science, UCLA
  - [©Sherwani] © Naveed A. Sherwani, 1992
    (companion slides to [She99])
  - [©Keutzer] © Kurt Keutzer, Dept. of EECS,
    UC-Berekeley
    http://www-cad.eecs.berkeley.edu/~niraj/ee244/index.htm
  - [©Gupta] © Rajesh Gupta
    UC-Irvine
    http://www.ics.uci.edu/~rgupta/ics280.html
  - [©Kang] © Steve Kang
    UIUC
    http://www.ece.uiuc.edu/ece482/

Routing

- Problem
  - Given a placement, and a fixed number of metal layers, find a valid pattern of horizontal and vertical wires that connect the terminals of the nets
  - Levels of abstraction:
    - Global routing
    - Detailed routing
- Objectives
  - Cost components:
    - Area (channel width) – min congestion in prev levels helped
    - Wire delays – timing minimization in previous levels
    - Number of layers (less layers → less expensive)
    - Additional cost components: number of bends, vias

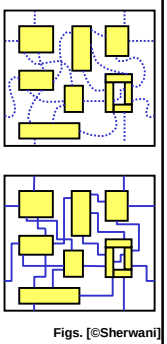
Routing Anatomy



Note: Colors used in this slide are not standard

Global vs. Detailed Routing

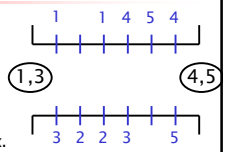
- Global routing
  - Input: detailed placement, with exact terminal locations
  - Determine "channel" (routing region) for each net
  - Objective: minimize area (congestion), and timing (approximate)
- Detailed routing
  - Input: channels and approximate routing from the global routing phase
  - Determine the exact route and layers for each net
  - Objective: valid routing, minimize area (congestion), meet timing constraints
  - Additional objectives: min via, power



Figs. [©Sherwani]

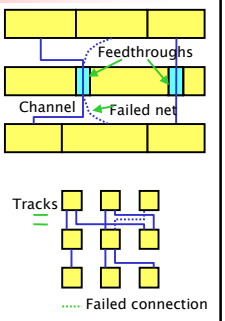
Routing Environment

- Routing regions
  - Channel
    - Fixed height ? (→ fixed number of tracks)
    - Fixed terminals on top and bottom
    - More constrained problem: switchbox. Terminals on four sides fixed
  - Area routing
    - Wires can pass through any region not occupied by cells (exception: over-the-cell routing)
- Routing layers
  - Could be pre-assigned (e.g., M1 horizontal, M2 vert.)
  - Different weights might be assigned to layers



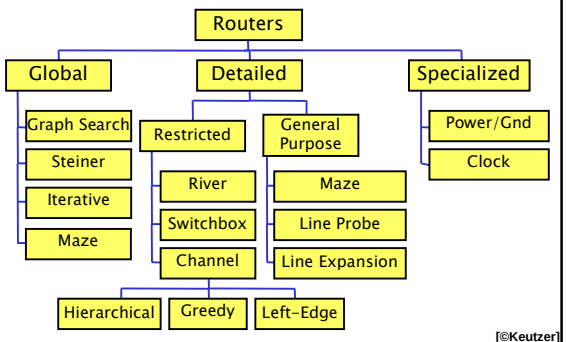
Routing Environment

- Chip architecture
  - Full-custom:
    - No constraint on routing regions
  - Standard cell:
    - Variable channel height?
    - Feed-through cells connect channels
  - FPGA:
    - Fixed channel height
    - Limited switchbox connections
    - Prefabricated wire segments have different weights



Figs. [©Sherwani]

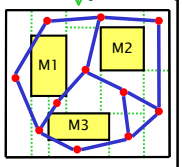
Taxonomy of VLSI Routers

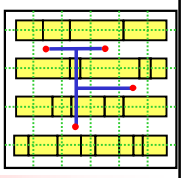


[©Keutzer]

Global Routing

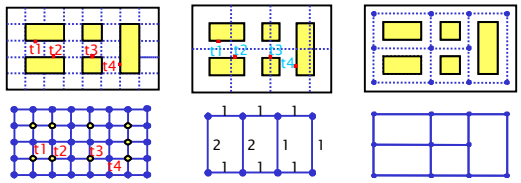
- Stages
  - Routing region definition
  - Routing region ordering
  - Steiner-tree / area routing
- Grid
  - Tiles super-imposed on placement
  - Regular or irregular
  - Smaller problem to solve, higher level of abstraction
  - Terminals at center of grid tiles
- Edge capacity
  - Number of nets that can pass a certain grid edge (aka congestion)
  - On edge E_{ij}, $Capacity(E_{ij}) \geq Congestion(E_{ij})$





Grid Graph

- Course or fine-grain
- Vertices: routing regions, edges: route exists?
- Weights on edges
  - How costly is to use that edge
  - Could vary during the routing (e.g., for congestion)
  - Horizontal / vertical might have different weights

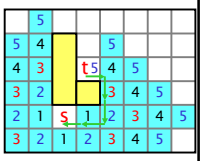


Global Routing – Graph Search

- Good for two-terminal nets
- Build grid graph (Coarse? Fine?)
- Use graph search algorithms, e.g., Dijkstra
- Iterative: route nets one by one
- How to handle:
  - Congestion?
  - Critical nets?
- Order of the nets to route?
  - Net criticality
  - Half-perimeter of the bounding box
  - Number of terminals

Global Routing – Maze Routing

- Similar to breadth-first search
  - Very simple algorithm
  - Works on grid graph
  - Time complexity: grid size (NxN)
- Algorithm
  - Propagate a "wave" from source until hit the sink (implemented using a queue)
  - Trace back to find the path
- Guaranteed to find the optimal solution
  - Usually multiple optimal solutions exist
- More than two terminals?
  - For the third terminal, use the path between the first two as the source of the wave



Maze Routing

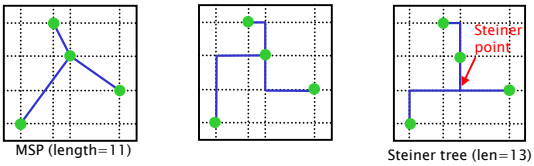
- Key to popularity:
  - Simplicity
  - Guaranteed to find the optimal solution
  - Can realize more complex cost functions too (e.g., number of bends in a path)
- Weakness:
  - Multiple terminals not handled efficiently
  - Dependent on grid, a two dimensional data structure
- Different variations exist
  - Soukup's alg:
    - First use DFS, when get to an obstacle, use BFS to get around
    - No guarantee to find the shortest path

Multiple Terminal Nets: Steiner Tree

- Steiner tree (aka Rectilinear Steiner Tree – RST):
  - A tree connecting multiple terminals
    - Original points: "demand points" – set D
    - Added points: "Steiner points" – set S
  - Edges horizontal or vertical only
- Steiner Minimum Tree (SMT)
  - Similar to minimum spanning tree (MST)
  - But finding SMT is NP-complete
  - Many good heuristics introduced to find SMT
- Algorithm
  - Find MST
  - Pass horizontal and vertical lines from each terminal to get the Hannan grid (optimal solution is on this grid)
  - Convert each edge of the MST to an L-shaped route on Hannan grid (add a Steiner point at the corner of L)

Steiner Tree

- Hannan grid reduces solution space (smaller grid)
  - For min length RST, Steiner points always on Hannan grid
- Convert MST to rectilinear paths
  - Length bounded by 1.5 times optimal SMT length
- Use alternate "L" routes to find the minimum tree



[©Sherwani]

Steiner Tree Routing

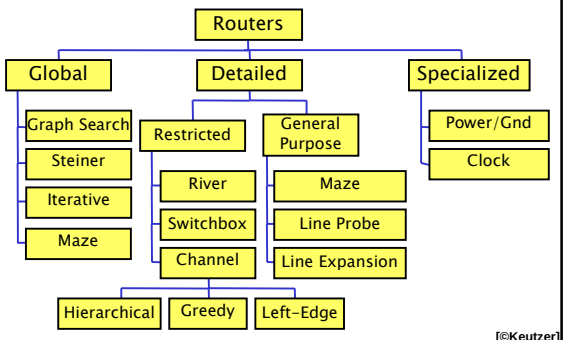
- Can apply different costs to different regions (or horizontal/vertical preference)
- Order of the nets
  - Sequential
    - Use # of terminals, criticality, etc. to determine order
  - Parallel
    - Divide the chip into large regions, perform the routing in parallel
- Key to popularity
  - Fast (not theoretically, but practically)
  - Bounded solution quality
- Shortcomings
  - Difficult to predict or avoid congestion

Global Routing Approaches

- A combination of different approaches might be used in chip-level routing
  - Route simple nets (2-3 pins in local area) directly (e.g., L-shaped or Z-shaped)
  - Use a "close to optimal" Steiner Tree algorithms to route nets of intermediate length
  - Route remaining "big" nets using a maze router
- Ordering
  - Some ordering is chosen, if can route all, then done, otherwise:
  - Rip-up and Re-route

[©Keutzer]

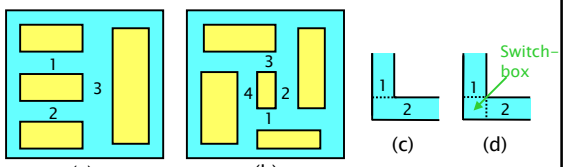
Taxonomy of VLSI Routers



[©Keutzer]

Detailed Routing: Channel vs. Switchbox

a) Channels have no conflicts
b) Conflicting channels
c) Conflict resolved using L-shaped channels
   - Order matters
d) Switchbox used to resolve the conflict
   - Order matters
   - Harder problem (compared to channel routing)



[©Sherwani]

Channel Routing Problem

- Input
  - Fixed terminal locations on the top and bottom
  - Possibly floating terminals on the left and right
  - Possibly fixed channel capacity constraint (capacity = max # of horizontal wires between the top and bottom boundaries of the channel)
  - Either gridless (aka area-based) or grid-based
- In the algorithms we consider
  - Constraints:
    - Grid structure
    - Two routing layers (one for H, another for V) Still relevant?
  - Minimize
    - # tracks (channel height)
    - Total wire length
    - # vias

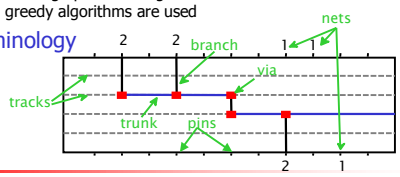

Grid-based


Gridless

[©Sherwani]
[©Keutzer]

Channel Routing Algorithms and Terminology

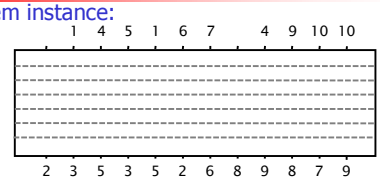
- General case is NP-Complete
- Algorithms
  - Simple case: left-edge algorithm (P)
  - General case: NP ➔ heuristics
    - Problem defined using horizontal constraints graph and vertical constraints graph
    - Either graph-based algorithms or greedy algorithms are used
- Terminology



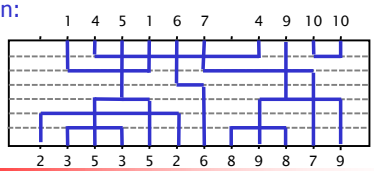
[©Keutzer]

Channel Routing Example

- Problem instance:



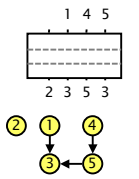
- Solution:



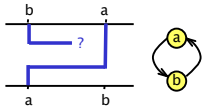
[©Keutzer]

Vertical Constraint Graph

- Represents the relative vertical positions of different net trunks (tracks)
  - Node: represents a net
  - Edge (x, y): if at the same column, x has a terminal on the upper edge and y has a terminal on the lower edge
  - (a,b) means that net "a" has to be above "b" – why?

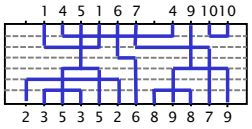


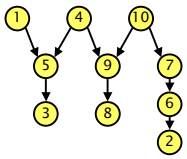
- Lower bound:
  - # tracks ≥ longest path in VCG
- VCG may have a cycle!
  - What shall we do??



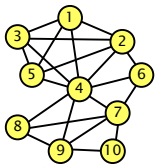
Horizontal / Vertical Constraint Graphs

- Channel routing problem can be completely characterized by VCG and HCG





Vertical constraint graph (VCG)

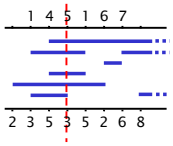


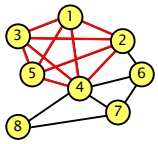
Horizontal constraint graph (HCG)

[©Keutzer]

Channel Density

- A net extends from its leftmost terminal to its rightmost one
- Local density at column C
  - ld (C) = # nets split by column C
- Channel density
  - d = max ld(c) over all C
- Relationship to HCG?
  - Local density ⇔ clique in HCG
  - d ⇔ size of maximum clique in HCG
- Lower bound:
  - # tracks ≥ d





[©Keutzer]

Left-edge Channel Routing Algorithm

- Used when no VCG edges
- Finds the optimal solution (# tracks = d)
- Nets are sorted according to their left endpoints

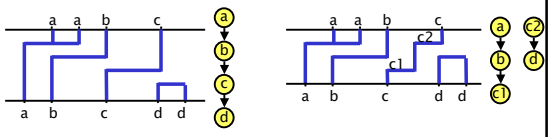
- Algorithm:

  Create an initial track t
  For all nets n_i in the order of their left endpoints
    if feasible to place the net on an existing track t_j,
      assign net n_i to track t_j.
    else create a new track t_{new} and assign n_i to it.

- Time complexity: O(n log n)

Doglegs

- Doglegs can reduce the longest path in VCG



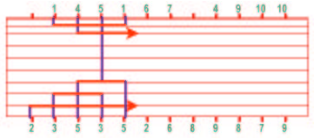
- Doglegs break cycles in VCG

[©Keutzer]

Greedy Channel Router

- Many greedy algorithms for channel routing exist
- Example: Rivest and Fiduccia DAC'82
  - Simple, linear algorithm
  - Guarantees routing of all nets
  - Uses doglegs (both restricted and unrestricted)
  - BUT may extend to right hand side of the channel
- Other techniques
  - Hierarchical: divide the channel into two smaller channels, route each small channel, merge
  - VCG reduction: nets that can be placed on the same track merged into one VCG node to reduce VCG size

Greedy Router: Rivest and Fiduccia

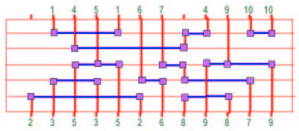


- Proceed column by column (left to right)
- Make connections to all pins in that column
- Free up tracks by collapsing as many tracks as possible to collapse nets
- Shrink range of rows occupied by a net by using doglegs
- If a pin cannot enter a channel, add a track
- O(pins) time

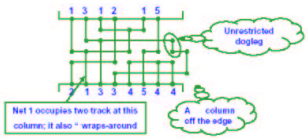
[©Keutzer]

Rivest and Fiduccia: Example

- Example output:



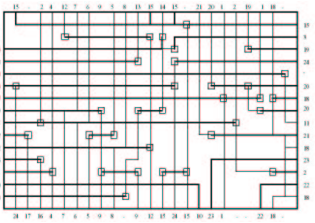
- Observations:
  - Always succeeds (even if cyclic conflict is present)
  - Allows unrestricted doglegs
  - Allows a net to occupy more than 1 track at a given column
  - May use a few columns off the edge



[©Keutzer]

Switchbox Routing

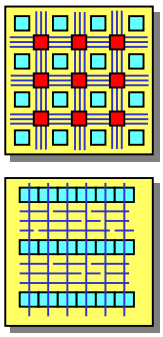
- Much harder problem than channel routing
  - Two-dimensional problem (channel routing was in a way one dimensional)
  - Need to solve in a hierarchical flow (split a channel into two, route one first, and route the second as a switchbox)
- A number of complex heuristic algorithms exist



[©Sarrafzadeh]

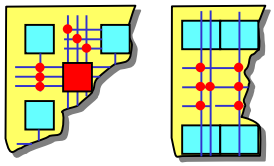
FPGA Architecture - Layout

- Island FPGAs
  - Array of functional units
  - Horizontal and vertical routing channels connecting the functional units
  - Versatile switch boxes
  - Example: Xilinx, Altera
- Row-based FPGAs
  - Like standard cell design
  - Rows of logic blocks
  - Routing channels (fixed width) between rows of logic
  - Example: Actel FPGAs

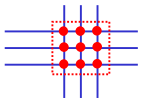


FPGA Programmable Switch Elements

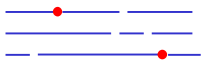
- Used in connecting:
  - The I/O of functional units to the wires



  - A horizontal wire to a vertical wire

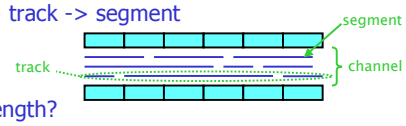


  - Two wire segments to form a longer wire segment

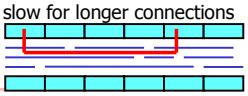


FPGA Routing Channels Architecture

- Note: fixed channel widths (tracks)
- Should "predict" all possible connectivity requirements when designing the FPGA chip
- Channel -> track -> segment

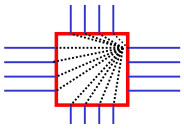


- Segment length?
  - Long: carry the signal longer, less "concatenation" switches, but might waste track
  - Short: local connections, slow for longer connections

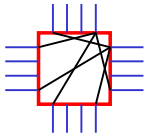


FPGA Switch Boxes

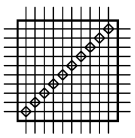
- Ideally, provide switches for all possible connections



- Trade-off:
  - Too many switches:
    - Large area
    - Complex to program
  - Too few switches:
    - Cannot route signals



One possible solution



Xilinx 4000

FPGA Routing

- Routing resources pre-fabricated
  - 100% routability using existing channels
  - If fail to route all nets, redo placement
- FPGA architectural issues
  - Careful balance between number of logic blocks and routing resources (100% logic area utilization?)
  - Designing flexible switchboxes and channels (conflicts with high clock speeds)
- FPGA routing algorithms
  - Graph search algorithms
    - Convert the wire segments to graph nodes, and switch elements to edges
  - Bin packing heuristics (nets as objects, tracks as bins)
  - Combination of maze routing and graph search algorithms