Lecture 25: Placement and Routing

Deming Chen

Some slides courtesy of Prof. J. Cong and N. Carter
Outline

- Overview
  - The placement problem
  - Partitioning-based placement
  - The routing problem
  - Global routing and detailed routing

The Placement Problem

- *Placement* is to find locations for the cells in a netlist such that they
  - Fit within the available space
  - The router can make all of the necessary connections between cells

- Placement and routing are inherently connected
  - Standard CAD tools do them in separate passes to simplify the algorithms
  - Algorithms that do placement and routing simultaneously to improve performance are an active research area
What Makes a Placement Good?

- **Routability** -- if you can’t route the required connections, design won’t work
- **Performance** -- placement affects wire delays, which affects performance
- **Density** -- what percentage of the available area can the placer use while still maintaining routability
  - ~70% for current tools, depending on design
- **Physical Issues** -- May want to distribute power dissipation/heat generation/clock load across the chip
- Fortunately, designs that have good routability often have good performance, because placing connected modules close together improves both metrics
Placement Methods

- Constructive methods.
  - Cluster growth algorithm
  - Force-directed method
  - Algorithm by Goto
  - **Min-cut based method**

- II. Iterative improvement approaches
  - Pairwise exchange
  - Simulated annealing- Timberwolf
  - Genetic algorithm

- III. Analytical methods
Interconnection Cost

(a) Steiner Tree
Rectilinear Length = 14

(b) Steiner Tree with Trunk
Rectilinear Length = 15

(c) Minimum Spanning Tree
Rectilinear Length = 16

(d) Chain
Rectilinear Length = 17

(e) Complete Graph
Rectilinear Length = 42

Approximation: half perimeter of the bounding box
Measure of Congestion (Routing Area)

(a) Two Tracks Required. All connections routed

(b) Shorter Wire Length. Three Tracks Required. A failure occurs if only two tracks are available
Min-Cut Based Placement

“A procedure for placement of standard-call VLSI circuits”

A.E. Dunlop, B.W. Kernighan, IEEE Trans. on CAD, vol CAD-4 No.1, Jan 1985

Min-cut with terminal propagation
Min-Cut Based Placement (Cont’d)

This process continues until there are only a few cells in each group ($\approx 6$)

- Each group has $\leq 6$ cells
- Assign cells in each group close together in the same row or nearly in adjacent rows

Group: smallest partition
Terminal Propagation

We should use the fact that $s$ is in $L_1$!

Assuming located at center

Fictitious cell of net $s$

$p$ will stay in $R_1$ for the rest of partitioning
Terminal Propagation

When not to use $p$ to bias partitioning

In this case, $p$ should not be used to bias the solution in either direction.

In general

Do not use $p$

Net $s$ has cells in many groups:

Minimum cost rectilinear sterner tree

$P_2$ should be ignored! too close to the partition line
Terminal Propagation (Cont’d)

Terminal propagation reduce overall area by \( \sim 30\% \)

Creating Rows

Row 1 \{ C_1, C_2, C_3 \} \rightarrow \text{row 1}
Row 2 \{ \} \rightarrow \text{row 1}
Row 3 \{ \} \rightarrow \text{row 1}
Row 4 \{ \} \rightarrow \text{row 1}

Choose \( \alpha \) and \( \beta \) preferably to balance row length (During re-arrangement )
Creating Rows

Partitioning of circuit into 32 groups. Each group is either assigned to a single row or divided into 2 rows.
Experimental Results

CMOS Chip with 453 nets and 412 cells

Manual Solution:
track density = 147;

without terminal propagation: t.d. = 313;
(t.d. reduced to 235 by iterative interchanges)

with terminal propagation: t.d. = 186;
(t.d. reduced to 152 by iterative interchanges)

CPU time = 3230 secs       VAX 11/780
Iterative Interchange Refinement is helpful
Remarks on Min-Cut Based Placement

Also implemented F-M partitioning method. Much faster but solutions appeared to be not as good as K-L

1. Use S-A to do partitioning. Much slower. If restricted to a reasonable CPU time, solutions are of similar quality of those by F-M method. Easy to implement

2. Seeking an elegant way to force some cells to be in particular positions

3. Investigate other algorithms for terminal propagation. Terminal propagation is the bottleneck of CPU time
The Routing Problem

- Given a set of cells, a set of pin locations on each cell, and a set of nets connecting cells, connect the cell pins with wires to implement the nets
  - May also want to achieve other goals:
    - Minimize overall area
    - Minimize length of each wire
- This is a hard problem -- routing can consume more area than actual cells
- The routing problem on modern chips involves managing several metal layers and assigning wires to them
Global Routing Formulation

**Given**
(i) Placement of blocks/cells
(ii) channel capacities

**Determine**
Routing topology of each net

**Optimize**
(i) max # nets routed
(ii) min routing area
(iii) min total wirelength

In general cell design or standard cell design, we are able to move blocks or cell rows, so we can guarantee connections of all the nets.

In gate array design, exceeding channel capacity is not allowed.
Routing

- First, divide the chip into regions of the type that the router can handle
- Then, use a *global router* to assign each net to a set of regions
- Finally, use a *detailed router* to lay out wires within the regions assigned to each net
Many different types of routers have been developed.
Global Routing

- Can work on channel-graph
- Edges in the channel graph represent channels or subsections of channels
  - Weights on edges represent the capacity of the channel
- Vertices represent intersections between channels

- Goal of global router is to assign wires to channels without exceeding any capacities
  - Detailed router worries about wire position within each channel
One Example
Global Routing Algorithms

- The problem is to find a suitable path for each connection through the weighted graph.
- Ideally, we would like to find minimum-cost trees, but:
  - This is NP-complete
  - Does not model net interactions, required to control congestion.
- Instead, heuristic techniques are used, usually consisting of two phases:
  - An Initial Global Route Technique: use shortest path algorithms to find some routing (may be congested).
  - An Iterative Improvement Global Route Technique: select and re-route connections to reduce congestion.
Detailed Routers

- Once nets have been assigned to routing regions, use a detailed router to find exact wire locations.
- We’ll look at three detailed routers:
  - Maze routers
  - Line-probe routers
  - Channel Routers

- Maze and line-probe routers can handle arbitrary geometries.
- Channel routers can only handle rectangular channels.
Maze Routers

- Oldest kind of router, first used for printed circuit boards
- Operate on a grid graph, where the surface is represented as an array of cells
Maze Routers -- the Lee Algorithm

- Always finds a shortest path if one exits
- Starting at the source node, explore out in a breadth-first fashion, keeping a list of the leaf nodes of the exploration tree
- At each step, replace each leaf node with its neighbors and update their distance to the source node
- Stop when you see the target cell (vertex)
- Block cells used by this wire and go on to the next
Lee Algorithm Illustration
Issues with the Lee Algorithm

- Lee algorithm is serial
  - Only looks at one connection at a time
  - Early connections can block the path that a later connection needs
- Uses lots of memory (proportional to number of cells)
- Long search time
- Improvements made by others:
  - Akers developed a coding method to reduce memory use
  - Soukup developed modified version that is 10-50 times faster, at the cost of not always finding shortest paths
  - Many people have looked at parallel versions
- Because of the long run-times and memory requirements, maze routers are mainly used today to cleanup-route any wires that more specialized routers couldn’t route
Line-Probe Routers

- First proposed by Mikami/Tabuchi and Hightower (independently)
- Model routing surface as a list of lines
- Basic idea: project line-probes from starting point until you encounter an obstruction
Line-Probe Routers

- Specifically:
  - Project probes in both the horizontal and vertical directions.
  - If obstructed, choose *escape points* along the current probes.
  - Project new segments from the escape points.
- This continues until two probes originating from the root and the target intersect.
- The two algorithms (Mikami and Hightower) differ in the way they choose escape points:
  - Mikami generates additional escape points at every grid point.
    - It is breadth-first and guarantees a solution, if one exists
  - Hightower adds a single escape point, appropriately chosen.
    - It does not guarantee a solution, even if one exists.
Line- Probe Routers

- The added overhead of Mikami’s algorithm can be seen in this example:

Mikami Algorithm

Hightower Algorithm

- The main advantage of line-probe routers, over maze routers, is their reduced memory requirements.
Channel Routers

- By requiring that the routing region be a rectangle, we can reduce routing effort
- A channel is a rectangular routing region with no obstacles and with pins only on two parallel sides
- We’ll represent channels as follows:

![Channel Routers Diagram]

- top
- terminal
- trunk
- branch
- dogleg
- bottom
- terminal
- track
Channel Routers

- Within a channel, the routing problem now becomes simpler: ensure that the number of wires in the channel doesn’t exceed its capacity at any point.
- One simple algorithm: the left-edge algorithm
  1. Imposes simplifying restrictions
  2. Each net can only use one trunk section
  3. Trunks must be in one routing layer, branches in another
  4. With these constraints, the algorithm determines the minimum number of tracks required to route a set of wires.
Left-Edge Algorithm

1. Sort trunks by their left-edge coordinate

2. Place trunks in sequence, using free space before adding tracks

3. Add branches to the placed trunks
Left-Edge Algorithm

- Has problems if there are multiple terminals at one horizontal position
  - In PCB environment, could handle this by *jogging* one of the conflicting segments

- This doesn’t work in an IC environment, making the problem NP-complete
  - Practical Left-Edge algorithms use heuristics that don’t guarantee finding the absolute minimum number of channels
Comparing Routing Algorithms

- The Lee algorithm requires large memory and is slow, but guarantees connection.
- Line-probe techniques are quite fast for simple mazes and use much less memory, but do not guarantee connection.
- Both the above techniques solve the problem one net at a time.
- Channel routers consider multiple nets at a time. They are fast and use little memory, but are restricted.
Summary

- Both placement and routing are important for interconnect optimization
  - Placement, global routing, detailed routing
  - Placement has an important impact on routing

- Both placement and routing are NP-hard problems still with ongoing research

- Different algorithms offer different tradeoffs in terms of performance, runtime and memory usage.