Partitioning

Readings: Chapter 5

Circuits can exceed chip capacity

Split circuits into chip-sized subcircuits
  - Meet capacity constraints
  - Reduce interconnect demand
  - Meet performance requirements

Partitioning

Floorplanning
Placement
Global Routing
Detailed Routing
Compaction

Partitioning Levels

Inter-board Partitioning
  - Reduce interconnect between different system board, smaller backplane

Inter-chip Partitioning
  - Reduce # of board levels (interconnect)
Intra-chip partitioning (quick placement)
Circuit Terminology

We represent circuits as graphs:

- Nodes/Cells/Vertices – logic elements
- Nets/Edges – wires in the design
- Terminals – connections between nodes and nets

Partitioning Terminology

- Cutset - set of nets connected to nodes in different partitions
- Cutsize - number of nets in the cutset
- Gain - improvement in cutsize if a given node is moved
Max-flow/Min-cut [Ford-Fulkerson]

Pick nodes S and T, and find optimum split of network such that S is in partition 1 and T is in partition 2.

Algorithm:
Assume each edge can carry at most 1 unit of “flow” in either direction
While we can find a path from S to T with remaining capacity
  Increment flow along path
Flow from S is cutsize, partition 1 is all nodes still reachable from S.

Limitations of Ford-Fulkerson

What are the expected results on a real circuit?
How does this differ from what we want?
Size-constrained Min-cut

Partition the circuit into two parts such that:
1.) Each partition has at least X% of the circuit
   45%-55% is typical bound.
2.) The cutset is minimized

Note 1: Size-constrained bipartitioning is NP-Complete

Note 2: Multi-way partitioning often extension of bipartitioning, or recursive
   bipartitioning.

Group Migration & Greedy Algorithm

Group migration - given an initial partitioning, move elements between partitions
Greedy algorithm: Move nodes, in order of gain, until no more good moves left.
   Make no move that violates the size constraint.
Fiduccia-Mattheyses

Size constraint: 4-6 nodes per partition

Fiduccia-Mattheyses Algorithm

Create initial partitioning;
While cutsize is reduced {
    While valid moves exist {
        Find unlocked node in each partition that most improves/least degrades cutsize when moved to other partition;
        Move whichever of the two nodes most improves/least degrades cutsize while not exceeding partition size bounds;
        Lock moved node;
        Update nets connected to moved nodes, and nodes connected to these nets;
    } endwhile;
    Backtrack to the point with minimum cutsize in move series just completed;
    Unlock all nodes;
} endwhile;
### Performance of FM

Create initial partitioning;
For # of iterations { /* Typically 4-5 iterations */
    While valid moves {
        Find best node from each partition;
        Pick best of two nodes
        Lock moved node;
        Update nets connected to moved nodes, and nodes
        connected to these nets;
    } endwhile;
    Backtrack;
    Unlock all nodes;
} endwhile;

Performance (ignoring red):
Performance of red depends on bucket data structure

### Finding Best Node: Bucket Data Structure

Array of lists, organized by gain
Max_gain points to highest occupied list

Find best node:
While max_gain entry is empty
max_gain--
Stop if max_gain drops off bottom

Update nodes:
Remove moved node
Foreach net attached to that node
adjust node gains for nodes attached to that net
If any node gain > max_gain
change max_gain

Array length = (Max # of node terminals) * 2 + 1
**Bucket Data Structure Delay**

Update nodes:
- Foreach attached net
  - adjust node gains

Operations = (# of times each net could change gain) * (nets * terminals/net)
  = (# of times each net could change gain) * (terminals)

Find best node:
- While max_gain array is empty max_gain--

Operations = Array length + moves upwards (in update nodes)
  = ((Max # of terminals*2)+1) + (times each net could change gain*terminals)

---

**Bucket Data Structure Delay (cont.)**

Net States:
- Uncut - can be cut by 1 move
- Frozen - can’t be cut by 1 move
- Cut - can be uncut by 1 move
- Stuck - can’t be uncut by 1 move
Final Performance of FM

Create initial partitioning;
For # of iterations { /* Typically 4-5 iterations */
    While valid moves {
        Find best node from each partition;
        Pick best of two nodes
        Lock moved node;
        Update nets connected to moved nodes, and nodes
        connected to these nets;
    } endwhile;
    Backtrack;
    Unlock all nodes;
} endwhile;

Non-red = iterations * nodes
Find best node = iterations *
   ((Max_terminals*2+1) + 4*terminals)
Update nodes = iterations * 4 * terminals

Overall =

Clustering

Group nodes together into clusters
Clusters are nodes that probably should be in same partition
Example: recursively cluster with neighbor to whom it shares the most edges

Partition with clusters, then remove clustering & repeat
Multiple Runs

Use multiple runs to improve expected value, reduce variance
Start from multiple, random initializations

![Bar Chart]