CS137:
Electronic Design Automation

Day 22: December 2, 2005
Routing 2 (Pathfinder)

Today

• Routing
  – Pathfinder
  • graph based
  • global routing
  • simultaneous global/detail

Global Routing

• Problem: Find sequence of channels for all routes
  – minimizing channel sizes
  – minimize max channel size
  – meeting channel capacity limits

Global → Graph

• Graph Problem on routes through regions

Global/Detail

• With limited switching (e.g. FPGA)
  – can represent routing graph exactly

Routing in Graph

• Find (shortest/available) path between source and sink
  – search problem (e.g. BFS, Alpha-Beta)
Easy?

- Finding a path is moderately easy
- What’s hard?
  - Can I just iterate and pick paths?

Example

- All links capacity 1
- $s_1 \rightarrow d_1$

Challenge

- Satisfy all routes simultaneously
- Routes share potential resources
- Greedy/iterative
  - Not know who needs will need which resources
  - i.e. resource/path choice looks arbitrary
  - …but earlier decisions limit flexibility for later
    - Like scheduling
    - Order effect result

Negotiated Congestion

- Old idea
  - Try once
  - See where we run into problems
  - Undo problematic/blocking allocation
    - Rip up
    - Use that information to redirect/update costs on subsequent trials
    - Retry

Negotiated Congestion

- Here
  - Route signals
  - Allow overuse
  - Identify overuse and encourage signals to avoid
    - Reroute signals based on overuse/past congestion

Basic Algorithm

- Route signals along minimum cost path
- If congestion/overuse
  - Assign higher cost to congested resources
- Repeat until done
Key Idea

• Congested paths/resources become expensive
• When there is freedom
  – future routes, with freedom to avoid congestion will avoid it
• When there is less freedom
  – must take congested routes
• Routes which must use congested resources will, while others will chose uncongested paths

Cost Function (1)

• PathCost=Σ (link costs)
• LinkCost = base × f(#routes using, time)
• Base cost of resource
  – E.g. delay of resource
  – Encourage minimum resource usage
    • (minimum length path, if possible)
    – minimizing delay = minimizing resources
• Congestion
  – penalizes (over) sharing
  – increase sharing penalty over time

Example (first order congestion)

Reroute, avoid congestion.

Example (need for history)

Need to redirect uncongested paths; how encourage?
Example (need for history)

Local congestion alone won't drive in right directions.
Both paths equal cost ...
... neither resolves problem.
May ping-pong back and forth.
(can imagine longer chain like this)

Cannot route s3→d3

Cost Function (2)

• Cost = (base + history)*f(#resources, time)

• History
  – avoid resources with history of congestion

Example (need for history)

S3→d3 and s4→d4 initially ping-pong

Builds up congestion history on path 3 and 4

Eventually makes path 3 and 4 more expensive than path 1;
... resolves conflict...

Adaptive cost scheme.

What about delay?

• Existing formulation uses delay to
reduces resources, but doesn’t directly treat

• Want:
  – prioritize critical path elements for shorter delay
  – allow nodes with slack to take longer paths

Cost Function (Delay)

• Cost=
  – W(edge)*delay + (1-W(edge))*congest
  – congest as before
    • (base+history)*f(#signals, time)
• W(edge) = D(edge)/Dmax
  – 1 for edge on critical path critical path
  – <1 for paths with slack
• Use W(edge) to order routes
• Update critical path and W each round

Convergence

• Chan+Schlag [FPGA’2000]
  – cases where doesn’t converge
  – special case of bipartite graphs
    • converge if incremental
    • or if prefer uncongested to least history cost
• theory (continuous)
  – only reroute overflow
  – converge in O(|E|) reroutes
  – But then have fractional routes...
Rerouting

• Default: reroute everything
• Can get away rerouting only congested nodes
  – if keep routes in place
  – history force into new tracks
    • causing greedy/uncongested routes to be rerouted

Rerouting

• Effect of only reroute congested?
  – maybe more iterations
    • (not reroute a signal until congested)
  – less time
  – ? Better convergence
  – ? Hurt quality?
    • (not see strong case for)
  – …but might hurt delay quality
    • Maybe followup rerouting everything once clear up congestion?

Run Time?

• Route |E| edges
• Each path search O(|E_graph|) worst case
  – …generally less
• Iterations?

Quality and Runtime Experiment

• For Synthetic netlists on HSRA
  – Expect to be worst-case problems
• Number of individual route trials limited (measured) as multiple of nets in design
  – (not measuring work per route trial)

Quality: fixed runtime

Quality Target
Quality vs. Time

Conclusions?
- Iterations increases with N
- Quality degrade as we scale?

Search Ordering
- Default: breadth first search for shortest
  - $O(\text{total-paths})$
  - $O(N^p)$ for HSRA
- Alternately: use $A^*$:
  - estimated costs/path length, prune candidates earlier
  - can be more depth first
    - (search promising paths as long as know can’t be worse)

Search: one-side vs. two-sides

Search: Oblivious vs. Directed
(BFS vs. $A^*$)

Single-side, Directed ($A^*$)
- Only expand search windows as prove necessary to have longer route.
Searching

- In general:
  - greedy/depth first searching
    - find a path faster
    - may be more expensive
      - (not least delay, congest cost)
  - tradeoff by weighting
    - estimated delay on remaining path vs. cost to this point
    - control greediness of router
  - More greedy is faster at cost of less optimal paths (wider channels)
    - 40% W → 10x time reduction [Tessier/thesis’98]

Domain Negotiation

- For Conventional FPGAs (and many networks)
  - path freedom
    - bushy in middle
    - low on endpoints

Multistage/Benes

Switches in all paths 0000 to 1111

Mesh Expand

Conventional FPGA Domains

Called: subset disjoint
Conventional FPGA Domains

Called: subset disjoint

Domain Routing

- No point in searching along an entire path from source
- Just to find it’s heavily congested at sink

HSRA Domains

Domain Negotiation

- Path bottlenecks exist at both endpoints
- Most critical place for congestion
- Most efficient: work search from both ends
  - more limiting in A*/Alpha-Beta search
  - focus on paths with least (no) congestion on endpoints first
  - FPGAs -- picking “domain” first
  - otherwise paths may look equally good up to end (little pruning)

Summary

- Finding short path easy/well known
- Complication: need to route set of signals
  - who gets which path?
  - Arbitrary decisions earlier limit options later
- Idea: iterate/relax using congestion history
  - update path costs based on congestion
  - Cost adaptive to route
  - reroute with new costs
- Accommodate delay and congestion

Next Term

- Next term
  - Project
    - Bias: Graph Machine focused
    - Attack EDA problems with Graph Machine
  - Go through all phases
    - Select problem
    - Formulate
    - Literature/technique review
    - Propose solution
    - Initial implementation
    - Final implementation
    - Select lectures
    - Driven by projects, interests
Admin

• This was last lecture
• Final Assignment
  – Due 12/7
• EAS Course Questionnaires

Big Ideas

• Exploit freedom
• Technique:
  – Graph algorithms (BFS, DFS)
  – Search techniques (A*, Alpha-Beta)
  – Iterative improvement/relaxation