Today

• Cover and Place
  – Linear
    • GAMA
    • Optimal Tree-based
  – Area and Time
    • covering for
      • ... and linear placement
  • Two Dimensional Cover and Place
    – Lily

Covering Review

• Use dynamic programming to optimally cover trees
  – problem decomposable into subproblems
  – optimal solution to each are part of optimal
  – no interaction between subproblems
  – small number of distinct subproblems
    – single optimal solution to subproblem
  • Break DAG into trees then cover optimally

Covering Basics

Basic Idea:

• Assume have optimal solution to all subproblems smaller than current problem
• try all ways of implementing current root
  – each candidate solution is new gate + previously solve subtrees
  – pick best (smallest area, least delay, least power)

Placement

• How do we integrate placement into this covering process?

GaMa - Linear Placement

• Problem: cover and place datapaths in rows of FPGA-like cells to minimize area, delay
• Datapath width extends along one dimension (rows)
• Composition is 1D along other dimension (columns)
• Always covering SIMD row at a time
Basic Strategy
• Restrict each subtree to a contiguous set of rows
• Build up placement for subtree during cover
• When consider cover, also consider all sets of arrangements of subtrees
  – effectively expands library set

Simultaneous Placement Benefits
• Know real delay (including routing) during covering
  – make sure critical logic uses fastest inputs
  – …shortest paths
• Know adjacency
  – can use special resources requiring adjacent blocks

GaMa Properties
• Operates in time linear in graph size
  – $O(|rule\ set| \times |graph\ nodes|)$
• Finds area-optimum for restricted problem
  – trees with contiguous subtrees
• As is, may not find delay optimum

GaMa Delay Example

GaMa Delay Problem
• Area can affect delay
• Doesn’t know when to pick worse delay to reduce area
  – make non-critical path subtree slower/smaller
  – so overall critical path will be close later
• Only tracking single objective
• Fixable as next technique demonstrates

GaMa Results
• Comparable result quality (area, time) to running through Xilinx tools
• Placement done in seconds as opposed to minutes to hours for Xilinx
  – simulated annealing, etc.
  – not exploiting datapath regularity
Simultaneous Mapping and Linear Placement of Trees

- **Problem**: cover and place standard cell row minimizing area
- **Area**: cell width and cut width
- **Technique**: combine DP-covering with DP-tree layout

[Lou+Salek+Pedram/ICCAD’97]

Task

- Minimize:
  \[\text{Area} = \text{gate-width} \times (\text{gate-height} + c \times \text{wire-pitch})\]

Composition Challenge

- Minimum area solution to subproblems does not necessarily lead to minimum area solution:
  \[
  \begin{align*}
  \text{area}(s2) &= 3 \times (6+2) = 36 \\
  \text{area}(s1+s2) &= 7 \times (10) = 70 \\
  \text{area}(s3) &= 5 \times (5+2) = 35 \\
  \text{area}(s1+s2) &= 8 \times (7) = 63
  \end{align*}
  \]

Minimize Area

- Two components of area:
  - gate-area
  - cut-width
- Unclear during mapping when need
  - a smaller gate-area
  - vs. a smaller cut-width
  - at the expense of (local) cell area
  - (same problem as area vs. delay in GaMa)

Strategy

- Recognize that these are incomparable objectives
  - neither is strictly superior to other
  - keep all solutions
  - discard only inferior (dominated) solutions

Dominating/Inferior Solutions

- A solution is dominated if there is another solution strictly superior in all objectives
  - A=3, T=2 and A=2, T=3
    - neither dominates
  - A=3, T=3 and A=3, T=2
    - A=3, T=3 is inferior, being dominated by either of the other two solutions
Non-Inferior Curve

• Set of dominators defines a curve

This is a recurring theme—often prune work using dominator curve

Strategy

• Keep curve of non-inferior area-cut points
• During DP
  – build a new curve for each subtree
  – by looking at solution set intersections
    • cross product set of solutions from each subtrees feeding into this subtree

Consequences

• More work per graph point
  – keeping and intersecting many points
• Theory: points$^{(\text{fanin})} \times \text{gates}$
• Points $\leq$ range of solutions in smallest dimension
• e.g. points $\leq$ number of different cut-widths

Algorithm: Tree Cover+Place

• For each tree node from leafs
  – For each gate cover
    • For each non-inferior point in fanin-subtrees
      – compute optimal tree layout
      – keep non-inferior points (cutwidth, gate-area)
  • Optimal Tree Layout
    – Yannakakis/JACM v32n4p950, Oct. 1985

Time Notes

• Computing Optimal Tree layout: $O(N \log(N))$
• Per node: $O(\text{cutwidth}^{(\text{fanin})} \times N \log(N))$
• Loose bound
  – possible to tighten?
  – less points and smaller "N" in tree for earlier subproblems
  – higher fanin $\rightarrow$ less depth $\rightarrow$ more use of small "N" for linear layout problems

Empirical Results

• Claim: 20% area improvement
Covering for Area and Delay  
(no placement)

• Previously saw was hard to do DP to  
  – simultaneously optimize for area and delay  
  – properly generate area-time tradeoffs  
• **Problem:**  
  – whether or not needed a fast path  
  – not clear until saw speed of siblings  

[Chaudhary+Pedram/DAC’92]

Strategy

• Use same technique as just detailed for  
  – gate-area + cutwidth  
• *i.e.* -- at each tree cover  
  – keep all non-inferior points  
    • (effectively the full area-time curve)  
  – as cover, intersect area-time curves to  
    generate new area-time curve  
• When get to a node  
  – can pick smallest implementation for a  
    child node that does not increase critical  
    path

Strategy

• Repeat trick:  
  – keep non-inferior points in three-space  
    • \(<\text{cut-width}, \text{gate-area}, \text{delay}>\)  
  – Intersect spaces to compute new cover  
    spaces  
  – May really need to discretize points to limit  
    work

Covering and Linear Placement for Area and Delay

• Have both  
  – cut-width + gate-area affects  
  – delay tradeoff  
• Result  
  – have three objectives to minimize  
    • cut-width  
    • gate-area  
    • gate-delay

Points to Keep

• Usually small variance in times  
  – if use discrete model like LUT delays, only a small  
    number of different times  
  – if use continuous model, can get close to optimum  
    by discretizing and keeping a fixed set  
• Similarly, small total variance in area  
  – *e.g.* factor of 2-3  
  – discretizing, gets close w/out giving up much  
• Discretized: run in time linear in \(N\)  
  – assuming bounded fanin gates

GaMa -- Optimal Delay

• Use this technique in GaMa  
  – solve delay problem  
  – get good area-delay tradeoffs  
  – GARP has a discrete timing model  
    • so already have small spread  
  – for conventional FPGA  
    • will have to discretize

Points to Keep

• Usually small variance in times  
  – if use discrete model like LUT delays, only a small  
    number of different times  
  – if use continuous model, can get close to optimum  
    by discretizing and keeping a fixed set  
• Similarly, small total variance in area  
  – *e.g.* factor of 2-3  
  – discretizing, gets close w/out giving up much  
• Discretized: run in time linear in \(N\)  
  – assuming bounded fanin gates

GaMa -- Optimal Delay

• Use this technique in GaMa  
  – solve delay problem  
  – get good area-delay tradeoffs  
  – GARP has a discrete timing model  
    • so already have small spread  
  – for conventional FPGA  
    • will have to discretize
Note

- Delay calculation:
  - assumes delay in gates and fanout
  - fanout effect makes heuristic
    • maybe iterate/relax?
  - ignores distance
- "Optimal" tree layout algorithm being used
  - is optimal with respect to cut-width
  - not optimal with respect to critical path wire length

Empirical Results

- Mapping for delay:
  - 20% delay improvement
  - achieving effectively same area
    • (of alternative, not of self targeting area)

Two Dimensions?

- Both so far, one-dimensional
- One-dimensional
  - nice layout restrictions
  - simple metric for delay
  - simple metric for area
- How extend to two dimensions?

2D Cover and Place

- Problem: cover and place in 2D to minimize area (delay)
- Area: gate area + "wirelength" area
- Delay: gate delay + estimated wire delay

Example

- Covering wrt placement matters

Nor2(nand2(A,B),nand2(C,D))=AND(A,B,C,D)

Strategy

- Relax placement during covering
- Initially place unmapped using constructive placement (Day11)
- Cover via dynamic programming
- When cover a node,
  - fanins already visited
  - calculate new placement
    • Center of Mass…like Force Directed
- Periodically re-calculate placement
- Use estimated/refined placements to get area, delay
Incremental Placement

- Place newly covered nodes so as to minimize wire lengths (critical path delay?)

Empirical Results

- In 1μm
  - 5% area reduction
  - 8% delay reduction
- Not that inspiring
  - ...but this was in the micron era
  - probably have a bigger effect today

Summary

- Can consider placement effects while covering
- Many problems can't find optimum by minimizing single objective
  - delay (area effects)
  - area (cutwidth effects)
- Can adapt DP to solve
  - keep all non-inferior points
  - can keep polynomial time
    - if very careful, primarily increase constants

Big Ideas:

- Simultaneous optimization
- Multi-dimensional objectives
  - dominating points (inferior points)
  - use with dynamic programming
- Exploit stylized problems can solve optimally
- Phase Ordering: estimate/iterate