Today

- How do we map to LUTs?
- What happens when
  - IO dominates
  - Delay dominates?
- Lessons…
  - for non-LUTs
  - for delay-oriented partitioning

LUT Mapping

- **Problem**: Map logic netlist to LUTs
  - minimizing area
  - minimizing delay

- Old problem?
  - Technology mapping? (Day 3)
  - How big is the library for K-input LUT?
    - $2^K$ gates in library

Simplifying Structure

- K-LUT can implement any K-input function

Preclass: **Cover in 4-LUT?**

Preclass: **Cover in 4-LUT?**
Preclass: Cover in 4-LUT?

Cost Function

- **Delay**: number of LUTs in critical path
  - doesn’t say delay in LUTs or in wires
  - does assume uniform interconnect delay

- **Area**: number of LUTs
  - Assumes adequate interconnect to use LUTs

LUT Mapping

- NP-Hard in general
- Fanout-free -- can solve optimally *given* decomposition
  - (but which one?)
- Delay optimal mapping achievable in Polynomial time
- Area w/ fanout NP-complete

Preliminaries

- What matters/makes this interesting?
  - Area / Delay target
  - Decomposition
  - Fanout
    - replication
    - reconvergent
Costs: Area vs. Delay

Decomposition

Fanout: Replication

Fanout: Reconvergence
Fanout: Reconvergence

What makes it hard?
- Cost does not monotonically increase as cover more of graph.
- Not clear when to stop?
- We say cost does not have a monotone property.

Preclass Revisited

Definition
- **Cone**: set of nodes in the recursive fanin of a node

Example Cones

Delay
Delay of Preclass Circuit?

- Poll: Delay of preclass circuit

Dynamic Programming

- Optimal covering of a logic cone is:
  - Minimum cost (all possible coverings)
- Evaluate costs of each node based on:
  - cover node
  - cones covering each fanin to node cover
- Evaluate node costs in topological order
- **Key**: are calculating optimal solutions to subproblems
  - only have to evaluate covering options at each node

Flowmap

- **Key Idea**:
  - LUT holds anything with K inputs
  - Use network flow to find cuts
    - logic can pack into LUT including reconvergence
    - ...allows replication
  - Optimal depth arise from optimal depth solution to subproblems

Max-Flow / Min-Cut

- The maximum flow in a network is equal to the minimum cut
  - ...the bottleneck
- We can find the mincut by computing the maxflow.
- Conceptually, how would we determine the maximum flow?
MaxFlow

- Set all edge flows to zero
  - $F[u,v]=0$
- While there is a path from $s,t$
  - (breadth-first-search)
  - for each edge in path $f[u,v]=f[u,v]+1$
  - $f[v,u]=-f[u,v]$
  - When $c[v,u]=f[v,u]$ remove edge from search
- $O(|E|^*\text{cutsize})$

Example Flow cut
• Delay objective:
  – minimum height, K-feasible cut
  – i.e. cut no more than K edges
  – start by bounding fanin ≤ K
• Height of node will be:
  – height of predecessors or
  – one greater than height of predecessors
• Check shorter first

Examples are K=4

• Construct flow problem
  – sink ← target node being mapped
  – source ← start set (primary inputs)
  – flow infinite into start set
  – flow of one on each link
  – to see if height same as predecessors
    • collapse all predecessors of maximum height
      into sink (single node, cut must be above)
    • height +1 case is trivially true
Augmenting Flows

Collapse at max height: works for K=4
Collapse not work (K still 4) (different/larger graph)

Forced to label height+1

Reconvergent fanout (yet different graph)

Can label at height

Flowmap

- Max-flow Min-cut algorithm to find cut
- Use augmenting paths until discover max flow > K
- O(K|E|) time to discover K-feasible cut (or that does not exist)
- Depth identification: O(KN|E|)

Min-cut may not be unique

To minimize area achieving delay optimum

- find max volume min-cut
  - Compute max flow => find min cut
  - remove edges consumed by max flow
  - DFS from source
  - Compliment set is max volume set

Collapse at max height: works for K=4
BFS from Source

Collapsed Node

Max-Volume Mincut

Flowmap

- Covering from labeling is straightforward
  - process in reverse topological order
  - allocate identified K-feasible cut to LUT
  - remove node
  - postprocess to minimize LUT count

- Notes:
  - replication implicit (covered multiple places)
  - nodes purely internal to one or more covers may not get their own LUTs
Flowmap Roundup

- Label
  - Work from inputs to outputs
  - Find max label of predecessors
  - Collapse new node with all predecessors at this label
  - Can find flow cut $\leq K$?
    - Yes: mark with label (find max-volume cut extent)
    - No: mark with label+1
- Cover
  - Work from outputs to inputs
  - Allocate LUT for identified cluster/cover
  - Recurse covering selection on inputs to identified LUT

Area

Changing Cost Functions Now
(previous was delay)

DF-Map

- Duplication Free Mapping
  - can find optimal area under this constraint
  - (but optimal area may not be duplication free)

[Cong+Ding, IEEE TR VLSI Sys. V2n2p137]

Maximum Fanout Free Cones

MFFC: bit more general than trees

MFFC

- Follow cone backward
- end at node that fans out (has output) outside the cone

MFFC example

Identify FFC
MFFC example

DF-Map
- Partition into graph into MFFCs
- Optimally map each MFFC
- In dynamic programming
  - for each node
    - examine each K-feasible cut
      - note: this is very different than flowmap where only had to examine a single cut
      - Example to follow
    - pick cut to minimize cost
      - $1 + \sum$ cones for fanins

DF-Map Example

Cones?
Start mapping cone
Similar to previous
DF-Map Example

1 1 1 1 3

3

DF-Map Example

1 1 1 1 1

DF-Map Example

2

1 1 1

2

DF-Map Example

3

1 1 1 1 1

2
DF-Map Example

DF-Map Example

DF-Map Example

DF-Map Example

DF-Map Example

DF-Map Example
Composing

- Don’t need minimum delay off the critical path
- Don’t always want/need minimum delay
- Composite:
  - map with flowmap
  - Greedy decomposition of “most promising” non-critical nodes
  - DF-map these nodes
Variations on a Theme

Applicability to Non-LUTs?
- *E.g.* LUT Cascade
  - can handle some functions of K inputs
- How apply?

Adaptable to Non-LUTs
- Sketch:
  - Initial decomposition to nodes that will fit
  - Find max volume, min-height K-feasible cut
  - ask if logic block will cover
    - yes $\Rightarrow$ done
    - no $\Rightarrow$ exclude one (or more) nodes from block and repeat
      - exclude $==\$ collapse into start set nodes
  - this makes heuristic

Partitioning?
- Effectively partitioning logic into clusters
  - LUT cluster
    - unlimited internal "gate" capacity
    - limited I/O (K)
    - simple delay cost model
      - 1 cross between clusters
      - 0 inside cluster

Partitioning
- Clustering
  - if strongly I/O limited, same basic idea
  - typically: partitioning onto multiple FPGAs
  - assumption: inter-FPGA delay $>>$ intra-FPGA delay
  - w/ area constraints
    - similar to non-LUT case
      - make min-cut
      - will it fit?
      - Exclude some LUTs and repeat

Clustering for Delay
- W/ no IO constraint
- area is monotone property
- DP-label forward with delays
  - grab up largest labels (greatest delays)
  - until fill cluster size
- Work backward from outputs creating clusters as needed
Area and IO?

- Real problem:
  - FPGA/chip partitioning
- Doing both optimally is NP-hard
- Heuristic around IO cut first should do well
  - (e.g., non-LUT slide)
  - [Yang and Wong, FPGA'94]

Partitioning

- To date:
  - primarily used for 2-level hierarchy
  - i.e., intra-FPGA, inter-FPGA
- Open/promising
  - adapt to multi-level for delay-optimized partitioning/placement on fixed-wire schedule
  - localize critical paths to smallest subtree possible?

Summary

- Optimal LUT mapping NP-hard in general
  - fanout, replication, ….
- K-LUTs makes delay optimal feasible
  - single constraint: IO capacity
  - technique: max-flow/min-cut
- Heuristic adaptations of basic idea to capacity constrained problem
  - promising area for interconnect delay optimization

Today’s Big Ideas:

- IO may be a dominant cost
  - limiting capacity, delay
- Exploit structure: K-LUTs
- Mixing dominant modes
  - multiple objectives
- Define optimally solvable subproblem
  - duplication free mapping

Admin

- Reading Wednesday on web
- Assignment 3 due Thursday