

## EDA (CS286.5b)

Day 3  
Clustering  
(LUT Map and Delay)

N.B. no lecture Thursday

## Today

- How do we map to LUTs?
- What happens when 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? (last week)
  - Library approach require  $2^{2^K}$  gates in library

## Simplifying Structure

- K-LUT can implement any K-input function

## 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

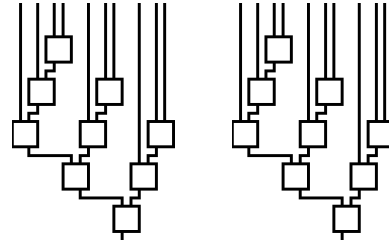
## LUT Mapping

- 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

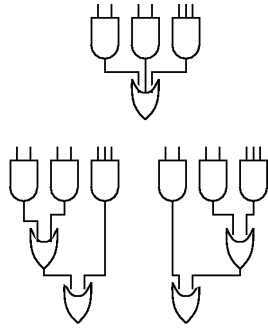
## Preliminaries

- What matters/makes this interesting?
  - Area / Delay target
  - Decomposition
  - Fanout
    - replication
    - reconvergent

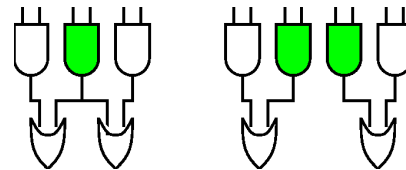
## Area vs. Delay



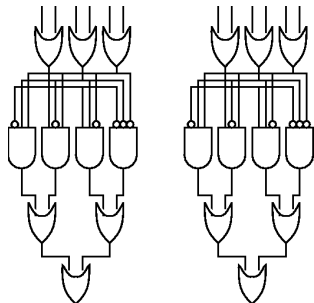
## Decomposition



## Fanout: Replication



## Fanout: Reconvergence



## Monotone Property

- Does cost function increase monotonically as more of the graph is included?
  - Delay?
  - gate count?
  - I/o?
- Important?
  - How far back do we need to search?

## Delay

## Dynamic Programming

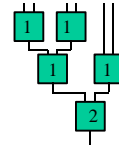
- Optimal covering of a logic cone is:
  - Minimum cost (all possible coverings)
- Evaluate costs of each node based on:
  - cover node
  - cones covering each fanin to node cover
- Evaluate node costs in topological order
- **Key:** are calculating optimal solutions to subproblems
  - only have to evaluate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

## Flowmap

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

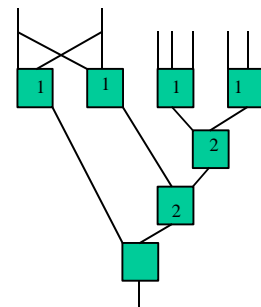
## Flowmap

- Delay objective:
  - minimum height, K-feasible cut
  - *i.e.* cut no more than K edges
  - start by bounding fanin  $\leq K$
- Height of node will be:
  - height of predecessors *or*
  - one greater than height of predecessors
- Check shorter first



## Flowmap

- Construct flow problem
  - sink  $\leftarrow$  target node being mapped
  - source  $\leftarrow$  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

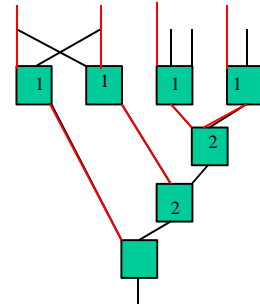
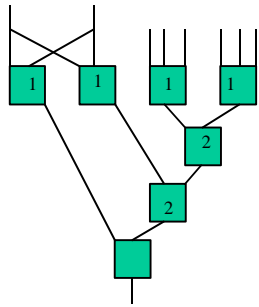
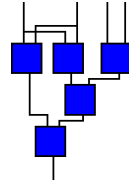


## Flowmap

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

## Flowmap

- Min-cut may not be unique
- To minimize area achieving delay optimum
  - find max volume min-cut
    - Compute max flow  $\Rightarrow$  find min cut
    - remove edges consumed by max flow
    - DFS from source
    - Compliment set is max volume set



## 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

Admin

No lecture Thursday

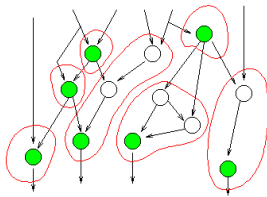
Area

## 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

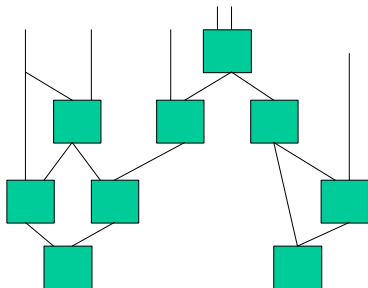


MFFC: bit more general than trees

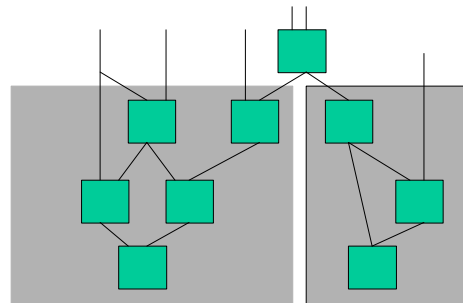
## MFFC

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

## MFFC example



## MFFC example



## DF-Map

- Partition into graph into MFFCs
- Optimally map each MFFC
- In dynamic programming
  - for each node
    - examine **each** K-feasible cut
    - pick cut to minimize cost
      - $1 + \sum$  MFFCs for fanins

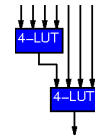
## Composing

- 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 => done
    - no => 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 works for partitioning to components
    - typically: partitioning onto multiple FPGAs
    - assumption: inter-FPGA delay  $\gg$  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