CS294-7: Reconfigurable Computing

Day 19: Tuesday, April 1

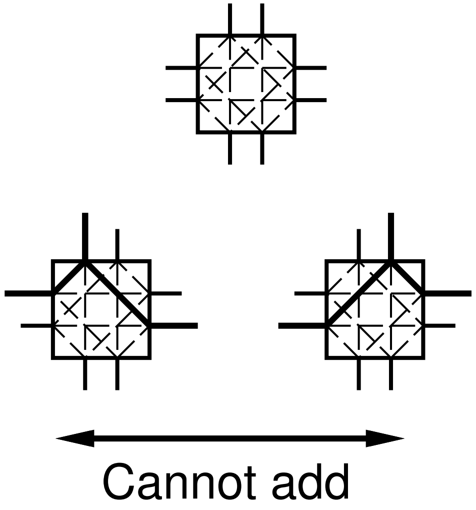
Compute Blocks

- Clarification on Universal Switch Blocks
- LUT Mapping

Universal Switch Blocks

Universal **switchbox can connect any set of inputs to their target output channels simultaneously**

N.B. **Cannot route all requests requiring fanout.**



No guarantee that the requests for a set of switchboxes can be unified?

LUT Mapping

Task: Map netlist to LUTs, minimizing area and/or delay.

Technology mapping:

Could use library approach (*e.g.* Dagon)

...but 2^{2^K} gates in library ($K = 4$, 9014 unique).

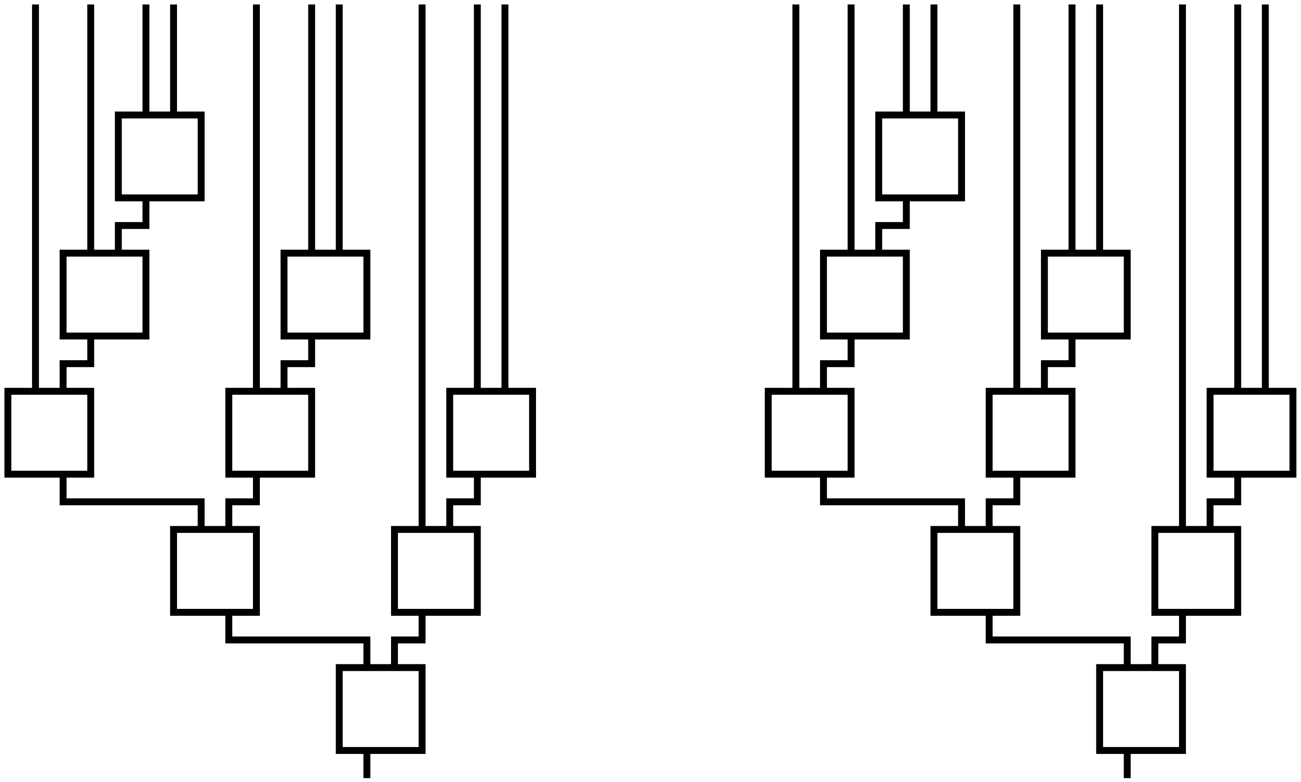
...and the fact that a K-LUT may implement any K-input function simplifies parts of the mapping problem.

LUT Mapping

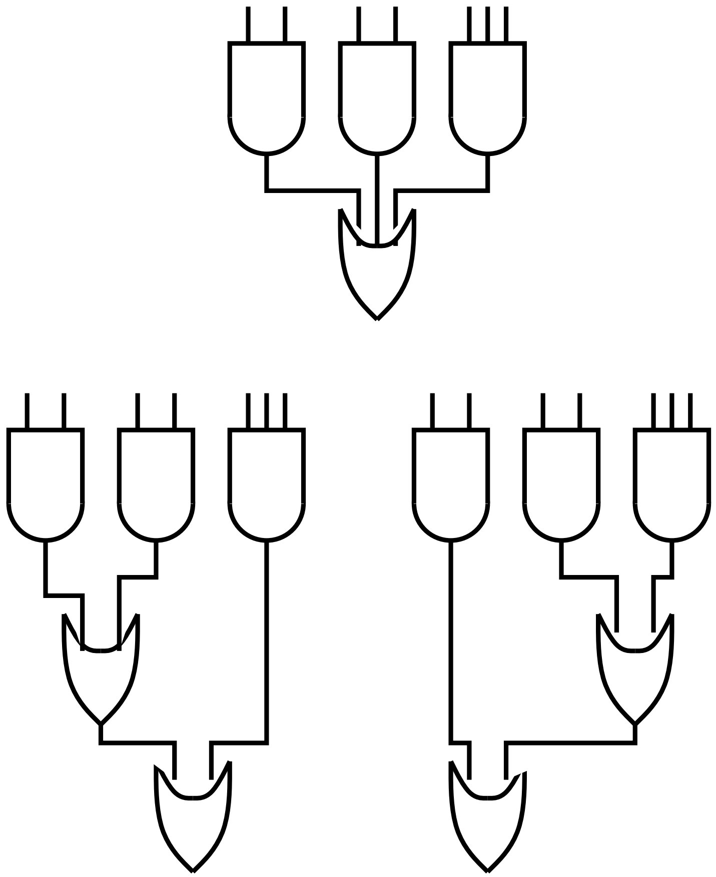
Technology Mapping

- General – NP-Hard
- Fanout-free – solve optimally given decomposition (but not clear which decomposition to use)
- Chortle – optimal delay (or area) for fanout free logic ($K \leq 5$)
- Flowmap – optimal delay (some sensitivity to decomposition)
- DF-map – area optimal duplication free (? sensitive to decomposition ?)

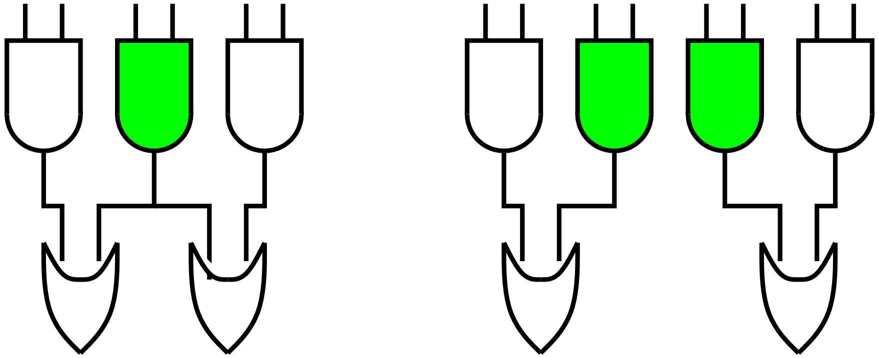
Area v/s Delay Mapping



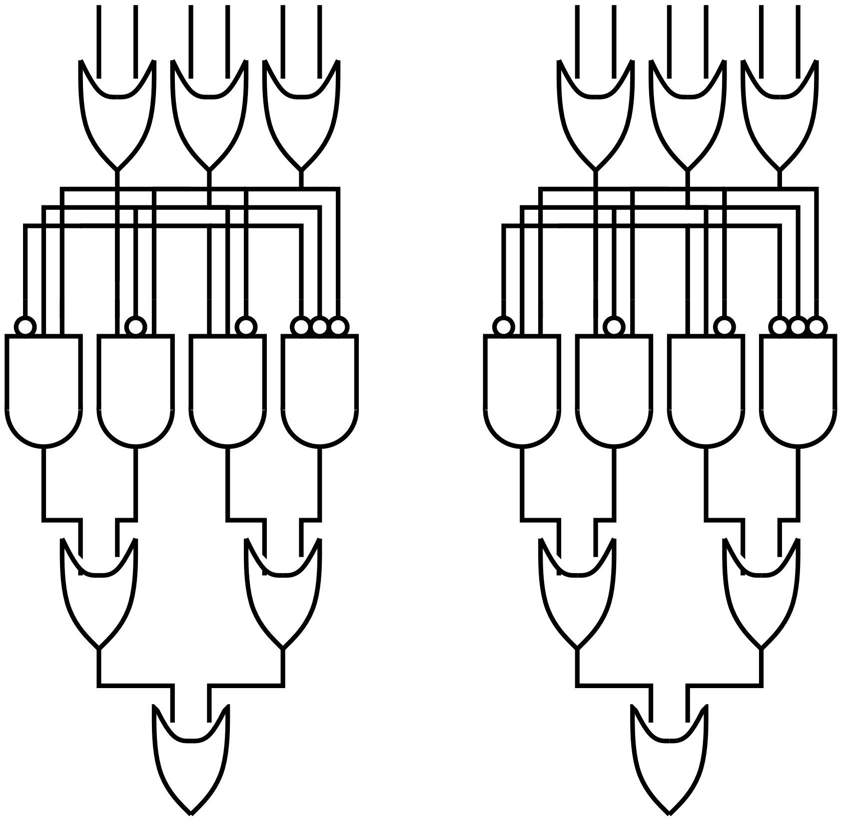
Decomposition



Why Replicate?



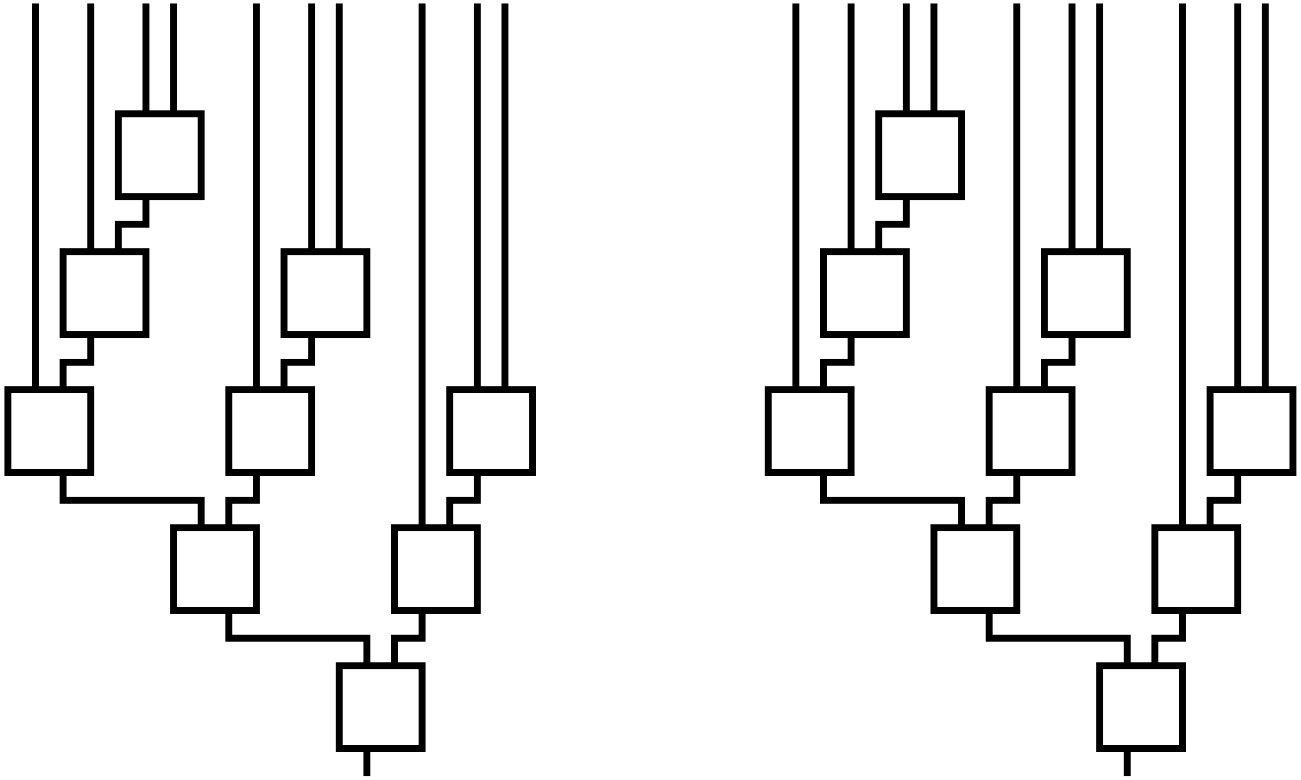
Reconvergence



Dynamic Programming

- Optimal covering of cone is:
  Mincost (possible coverings)
- Evaluate costs for each node in topological order
- Know optimal solutions to subproblems,
  so only have to evaluate covering options at each node as encountered

Dynamic Programming Example



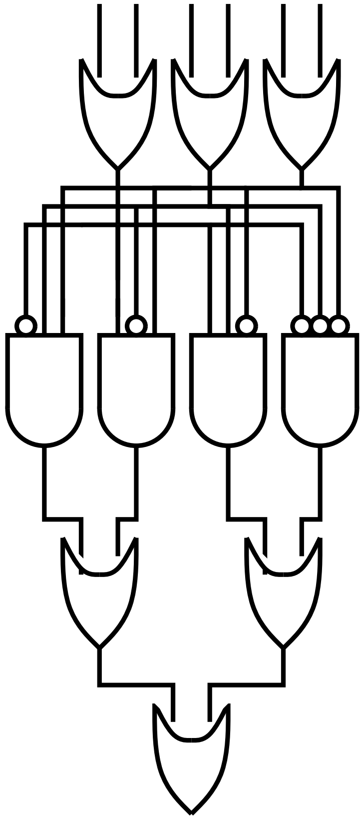
Chortle Decomposition/Bin Packing

Dynamic programming – at each node:

- Pack fanin LUTs into bins (implicit node decomposition)
  1. Find largest LUT input
  2. Place in largest available bin (allocate new bin if necessary)
  3. Repeat until all LUTs packed into bins
- Pack final node fanin
  1. Find largest unpacked LUT
  2. Link into first available bin (allocate new bin if necessary)
  3. Repeat until all fanin packed into bins

Flowmap

- Decompose to 2-input gates
- Also Dynamic Programming
  At each node:
  - Compute minimum height
    K-feasible cut
    (network flow)



min height K-feasible cut

- Height of node will be height of predecessors or one greater
- Check shorter height first
- Construct flow problem:
  - sink $\leftarrow$ target node being mapped
  - source $\leftarrow$ start set (PIs)
  - flow infinite into start set; one on each link

Flowmap

To simultaneously minimize area:
Compute max volume min-cut

- compute max flow
- remove edges consumed by max flow (partition at min cut)
- DFS from source
- complement of set is max volume set

Postprocess:

- Combine predecessors (remember min-cut, not K-cut)
- Gate decomposition

Maximum Fanout Free Cones

Duplication Free Map

- Partition graph into MFFCs
- Optimally map each MFFC
- In dynamic programming, at each node:
  - examine each K-feasible cut, P, selecting least cost covering:

$$cost(P) = 1 + \sum_{i=1}^{m} area\left(\mathrm{MFFC}_{v_i^p}\right)$$

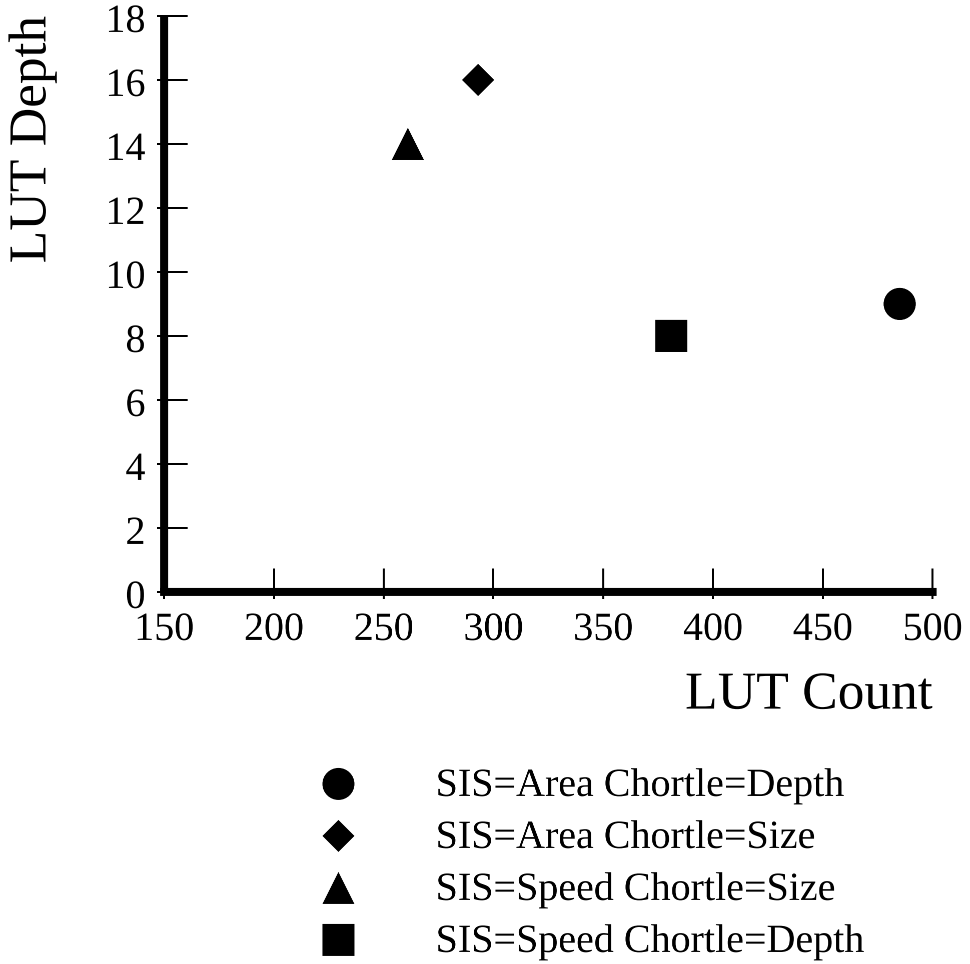
Flowmap r

N.B. In flowmap all nodes mapped to minimum height

$\oplus$ Often don't want endpoints of area-time curve.

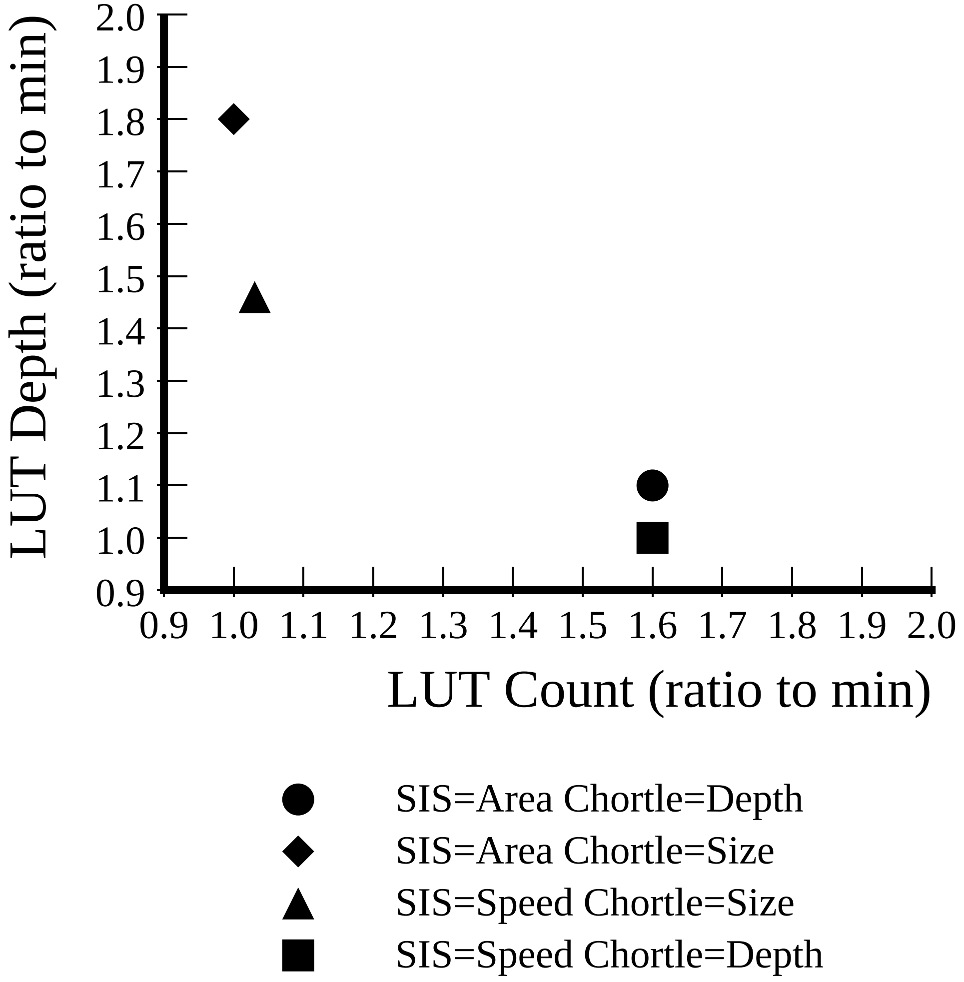
- Map with flowmap
- Greedy decomposition of "most promising" non-critical nodes
- DF-map (area) relaxed nodes

Technology Independent and LUT Mapping



(`rot` benchmark)

Technology Independent and LUT Mapping



(Average Ratios)

LUT Mapping Summary

- Can find depth-optimal mapping in polynomial time
- Area-optimal with replication NP-Hard
- Can find duplication free area-optimal in polynomial time
- Basic technique uses
  - dynamic programming
  - min-cut partitioning