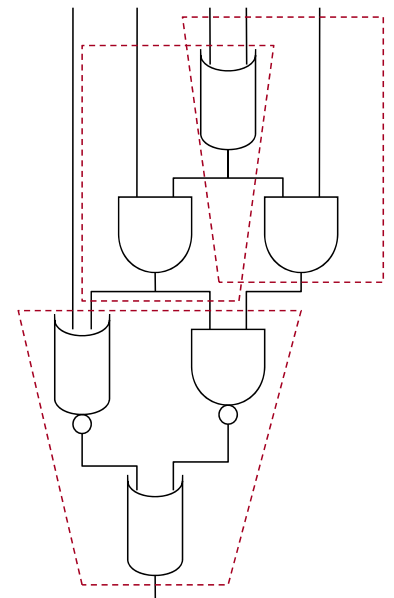


Technology Mapping

1

Outline

- Logic Synthesis
- Basics of Technology Mapping
- DP Tree-map algorithm
- Greedy Tree-map algorithm
- FlowMap algorithm
- DAOmap algorithm
- WireMap algorithm



2

HDL Synthesis

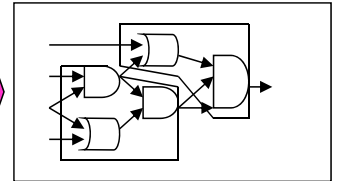
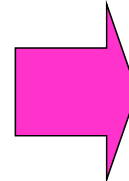
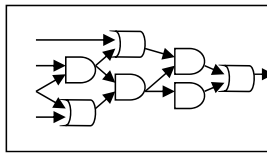
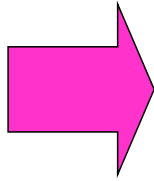
Synthesis = Domain Translation + Optimization

Domain
translation

Optimization
(area, timing, power...)

```
--VHDL
if(A='1') then
  Y<=C + D;
elseif (B='1') then
  Y<=C or D;
else Y<=C;
endif

//Verilog
if(A==1)
  Y=C + D;
else if(B==1)
  Y=C | D;
else Y=C;
```



Behavioral domain

Structural domain

3

Syntax-directed Translation

- Translate HDL into logic directly.
 - $ab + ac$
- Generally requires *optimization*.

4

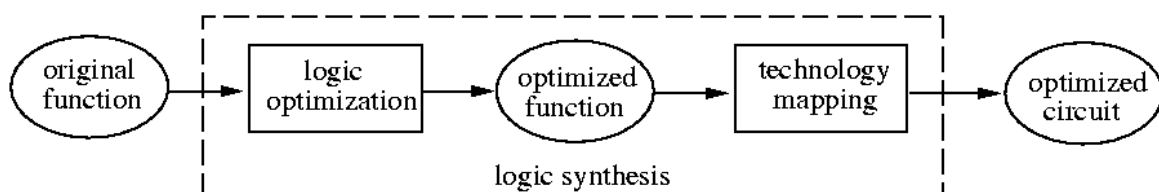
Macros

- Pre-designed components
 - Generally identified by language features.
 - E.g. + operator, xxx()
- *Hard macro*: includes placement.
- *Soft macro*: no placement.

5

Logic Synthesis Phases

- *Logic optimization* transforms current gate-level network into an optimized gate-level network.
- *Technology mapping* transforms the gate-level network into a network of cells in the target technology library.



6

Logic Optimization

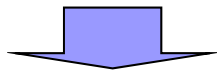
- Work on Boolean expression equivalent.
- Estimate size based on literal count.
- Estimate delay by simple delay models.
- Use *factorization, substitution, decomposition, extraction*, etc. to optimize logic.

7

Logic Optimization

1. Decomposition (single function)

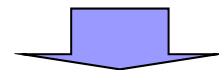
$$f = abc + abd + c'd' + a' + b'$$



$$\begin{aligned} f &= xy + (xy)' \\ x &= ab \\ y &= c + d \end{aligned}$$

2. Extraction (multiple functions)

$$\begin{aligned} f &= (az + bz')cd + e \\ g &= (az + bz')e' \\ h &= cde \end{aligned}$$



$$\begin{aligned} f &= xy + e \\ g &= xe' \\ h &= ye \\ x &= az + bz' \\ y &= cd \end{aligned}$$

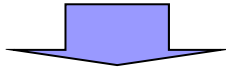
8

Logic Optimization

3. Factoring

(series-parallel decomposition)

$$f = ac + ad + bc + bd + e$$

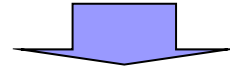


$$f = (a + b)(c + d) + e$$

4. Substitution

$$f = ga + g'b$$

$$g = c + d$$



$$f = ac + ad + bc'd'$$





$$g = c + d$$

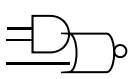
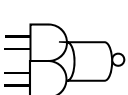
9

Technology Mapping

- Map Boolean expressions into a particular *cell library*.
- Mapping takes into account area, delay, etc.
- May perform some optimizations in addition to mapping e.g. retiming.
- Allow more accurate delay models.

An Example Cell Library

INV	2		a'
NAND2	3		$(ab)'$
NAND3	4		$(abc)'$
NAND4	5		$(abcd)'$

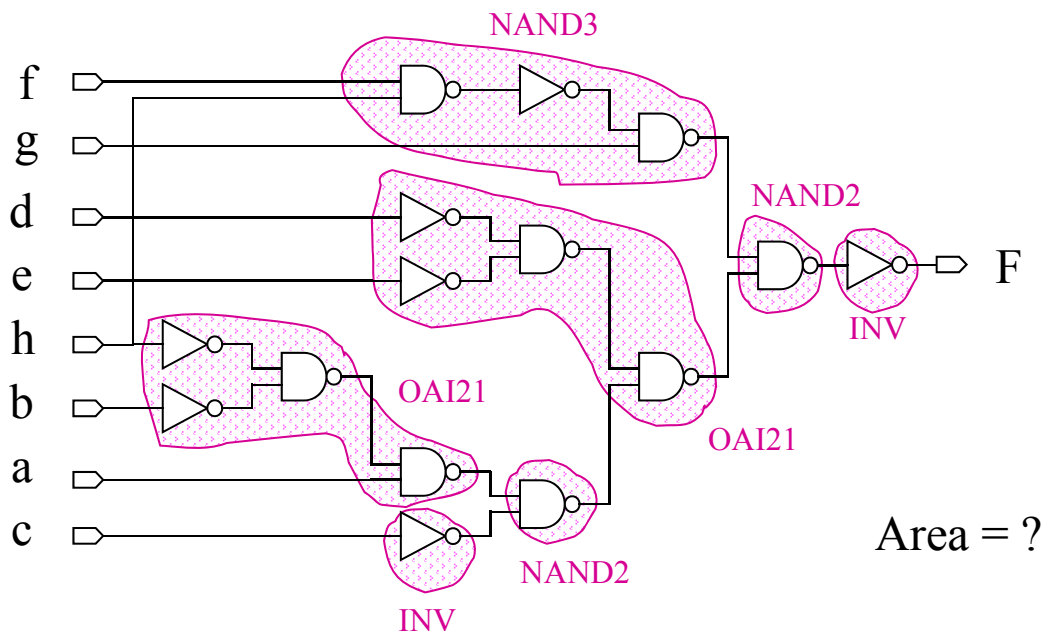
AOI21	4		$(ab+c)'$
AOI22	5		$(ab+cd)'$

Library Element

11

Technology Mapping

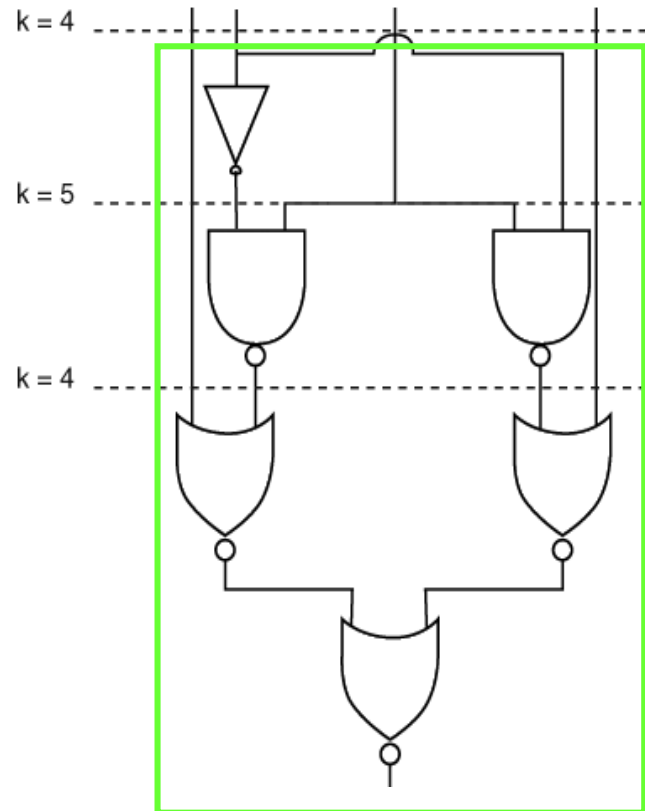
- Cover function with library cells
- To optimize area, delay, etc.



12

LUT-based Technology Mapping

- # inputs *doesn't* always increase with added functions.
- Useful to find the largest cone that will fit into a LUT.



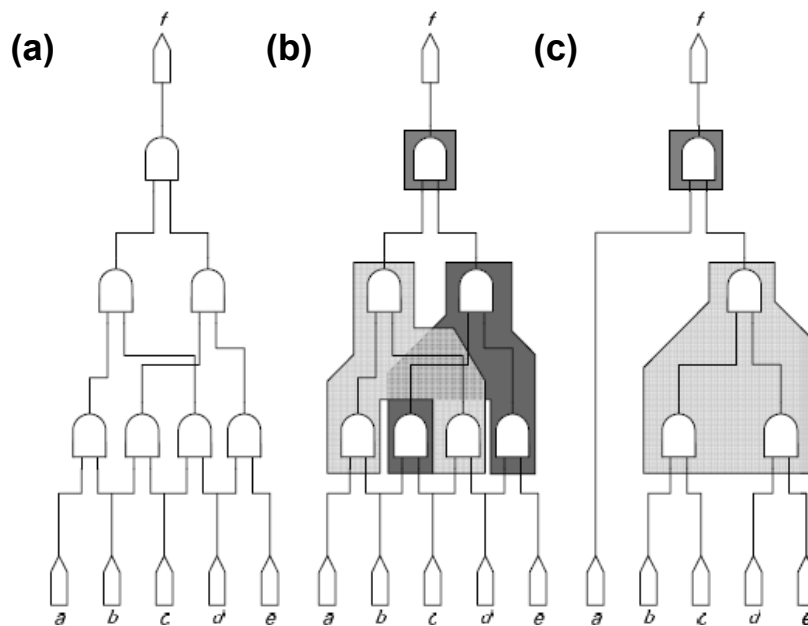
13

FPGAs vs. Custom Logic

- Area cost metric for static gates is literal:
 - $ax + bx'$ has four literals, requires 8 transistors.
- Area cost metric for FPGAs is logic element:
 - All functions that fit in an LE have the same cost.

14

Logic Optimization & Technology Mapping



- (a) Original network
(b) Mapping without logic optimization
(c) Mapping with logic optimization

15

Technology Mappers for LUT-Based FPGAs

- Input can be a sequential circuit.
- Most algorithms work on the combinational portions which are *directed acyclic graphs (DAGs)*.
- For mapping with K -input LUTs
 - Remove sequential elements to break into combinational portions
 - Covering K -bounded DAGs by K -feasible cones
- Property: a K -LUT can implement any function of $\leq K$ inputs.

16

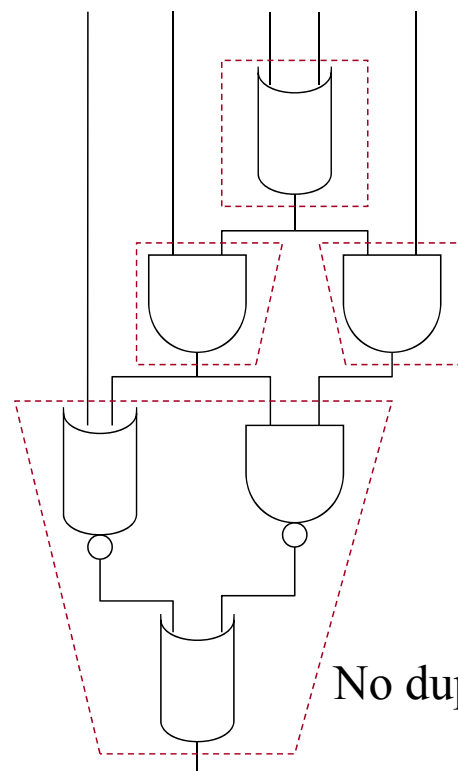
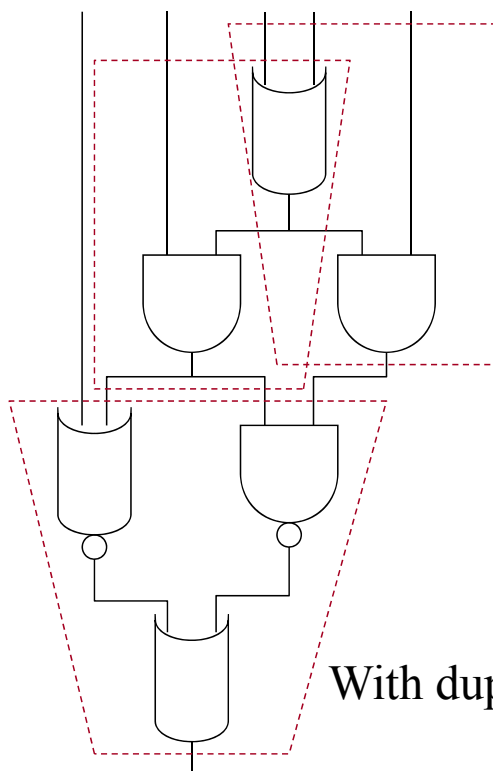
Technology Mapper Classification

- Classification based on objectives:
 - *Area* minimization
 - *Delay* minimization
 - *Power* minimization
 - *Routability* optimization
- Some algorithms perform duplication-free mapping, others allow *duplication* (i.e., a node may be covered by more than one LUT).
- Duplication can lead to reduced no. of LUTs and/or delay.

17

Node Duplication

- 3-LUTs mappings:



18

Hardness of Area /Power/Delay Minimization

- Area-optimal mapping problem (node duplication allowed) is NP-hard.
- Power-optimal mapping problem is NP-hard.
- But duplication-free area-optimal mapping is polynomial time solvable.
- Delay-optimal mapping problem is polynomial time solvable.

19

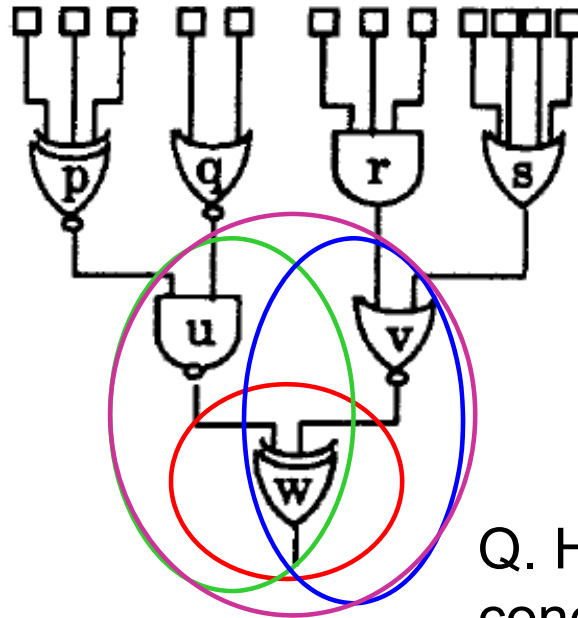
DP Tree-map: duplication-free area-optimal mapping for tree

- Cover a K -bounded tree with minimum no. of K -LUTs.
- Optimal *dynamic programming* approach.
- Process nodes in *topological order* starting from PIs.
- Recursive assumption: When computing the best mapping of tree T_i , the best mapping of all its subtrees are known.
- Enumerate all *K -feasible cones* rooted at node i .
- If mapping MT_i uses a LUT_i to implement node i :
 - $\text{Area}(MT_i) = 1 + \sum_{j \in \text{input}(LUT_i)} \text{Area}(MT_j)$.
- A best mapping MT_i^* is one such that Area is minimum.

20

DP Tree-map

- Min-area tree mapping with $K=5$



Q. How many 5-feasible cones rooted at w ?

21

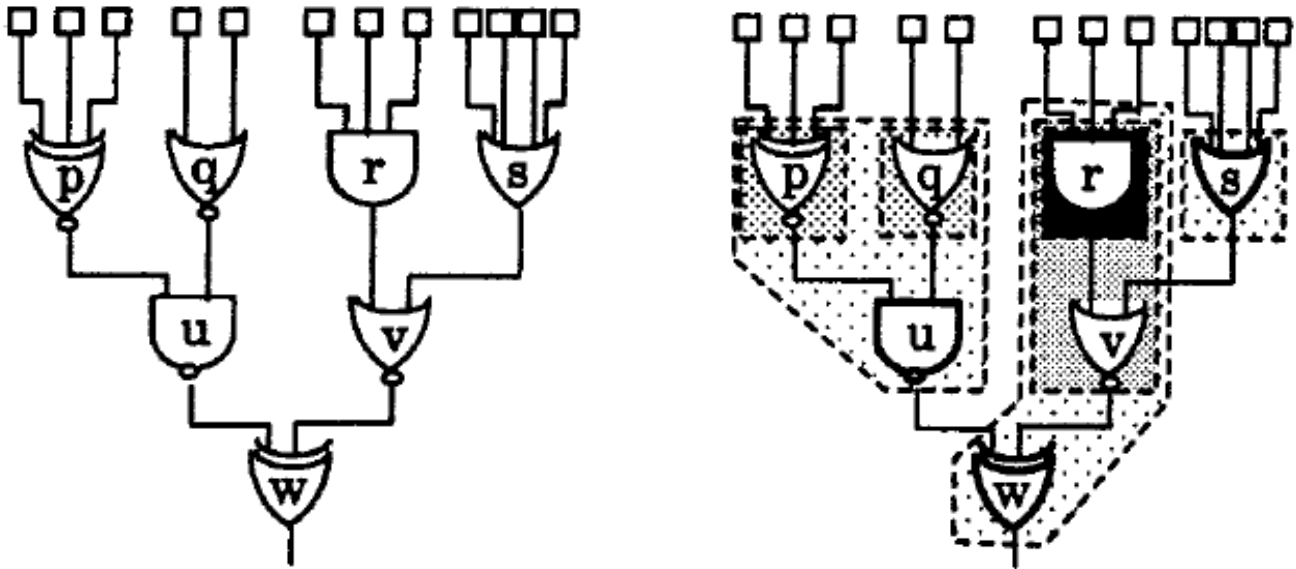
Greedy Tree-map: duplication-free area(delay)-optimal mapping for tree

- An optimal greedy algorithm without enumeration for min-area tree mapping.
- Let w_1, \dots, w_m be the fanin nodes of node i and $|\text{input}(\text{LUT}_{w_1})| \leq \dots \leq |\text{input}(\text{LUT}_{w_m})|$.
- Greedy packing:
 - Cover i by $\text{LUT}_i = \{i\} \cup_{j \leq s} \text{LUT}_{w_j}$ where s is the largest index s.t. LUT_i remains K -feasible
- Time complexity is $O(\max\{K, \log n\}n)$.
- Modification for depth-optimal tree mapping:
 - Order the fanin nodes of node i by their LUTs' depths in decreasing order.

22

Greedy Tree-map

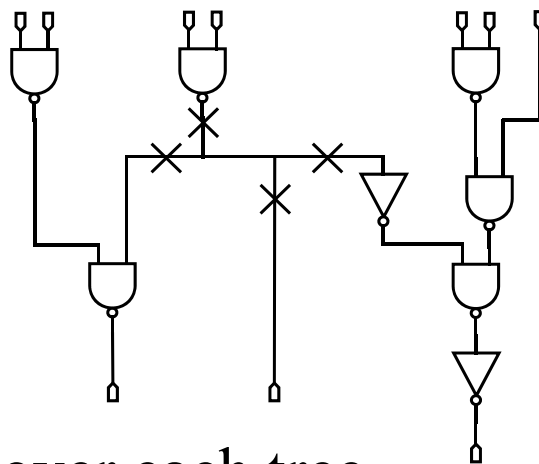
- Min-area tree mapping with $K=5$



23

Heuristic for General Network Mapping

- Partition a network into trees
 - Cut the network at all multiple fanout points

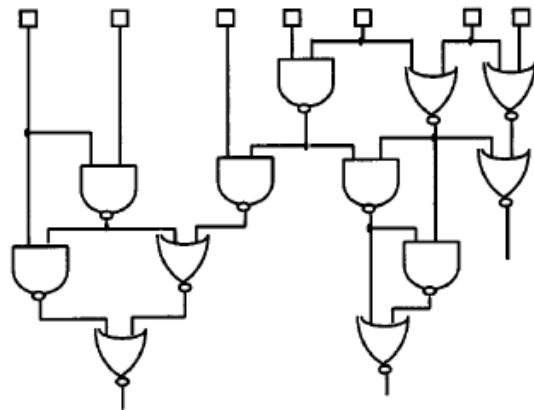


- Optimally cover each tree
- Piece the tree-covers into a cover for the subject graph

24

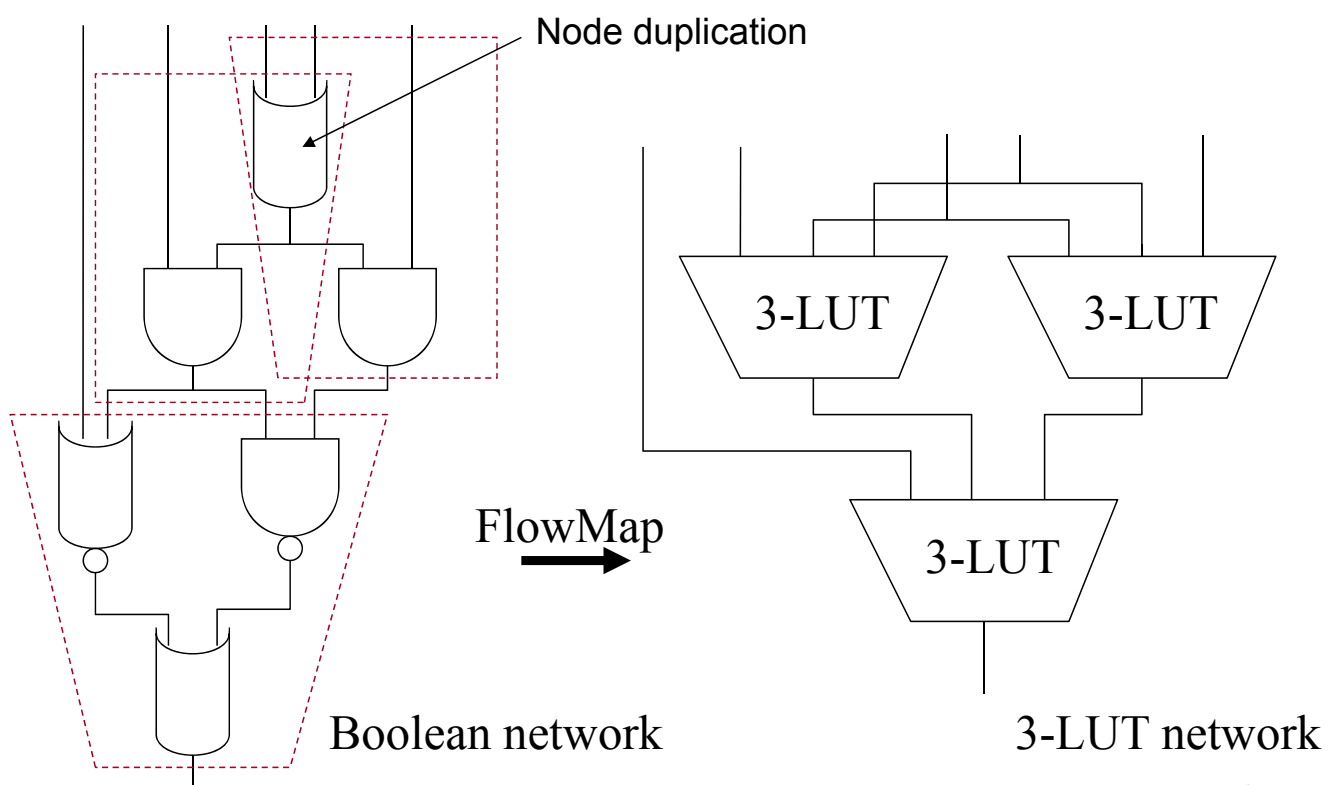
FlowMap: delay-optimal mapping for DAG

- Guarantee *depth-optimal* mappings for general DAGs.
- Automatically consider *duplication* for depth minimization(i.e., a node may be covered more than once).
- Use *network flow*.



25

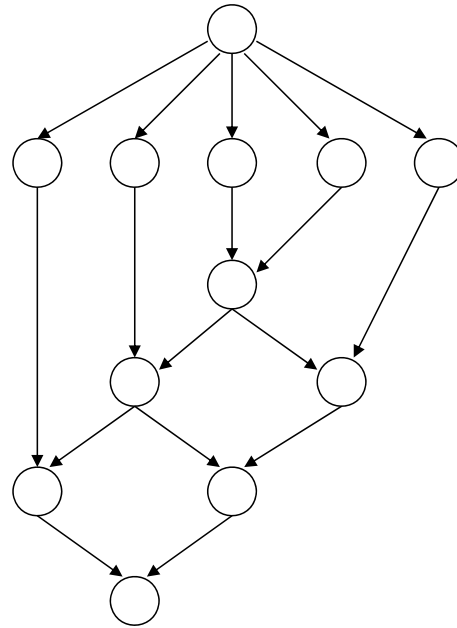
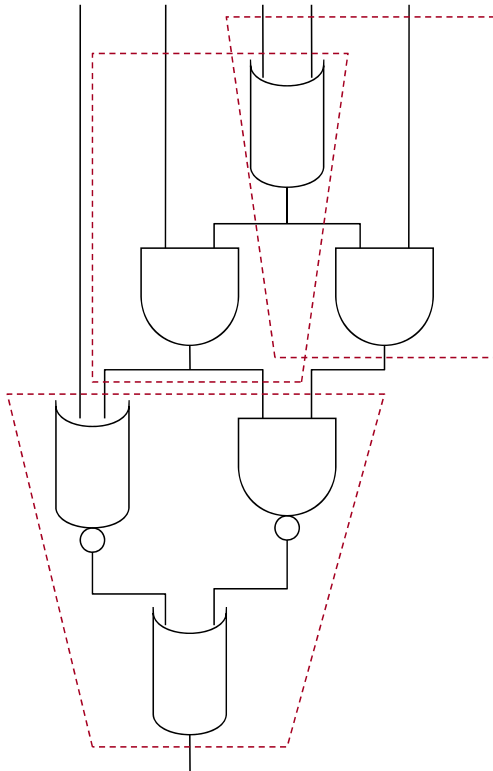
FlowMap



26

FlowMap

■ Graph model



Represent the network by a graph.

27

FlowMap

■ Terminologies

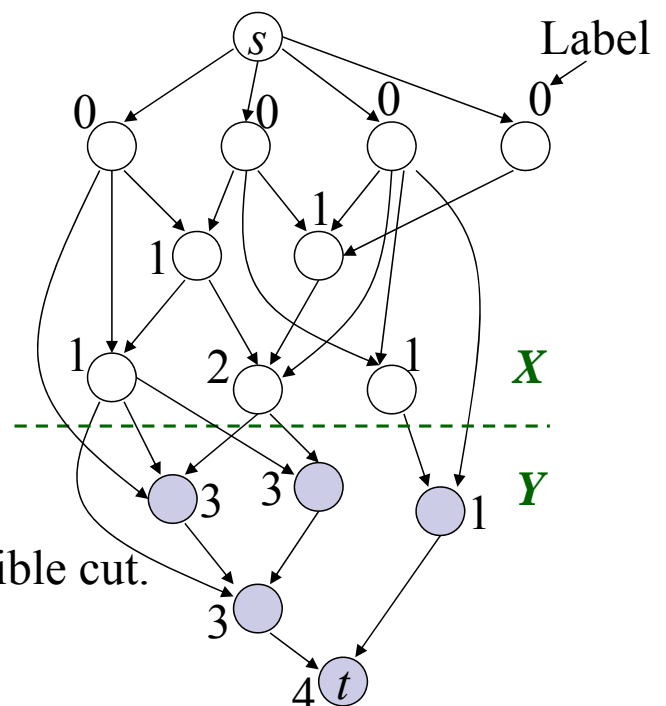
Node cut size $n(X,Y) = 5$

Edge cut size $e(X,Y) = 8$

Volume of a cut, $vol(X,Y) = |Y| = 5$

k -feasible cut: $n(X,Y) \leq k$

Height of a cut, $h(X,Y) = \max \{ \text{label}(x) : x \in X \} = 2$



This example shows a 5-feasible cut.

28

FlowMap

■ Assumptions:

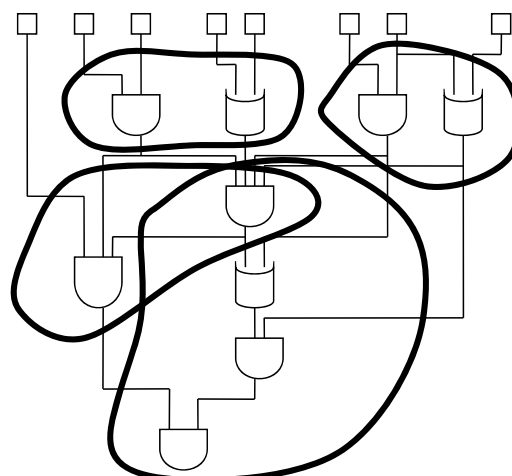
- Map to K -input LUTs.
- Input Boolean network is K -bounded (each node has $\text{fanin} \leq K$).
- *Unit delay* model.
- **Labeling**: Compute a label for each node, that reflects the level of the K -LUT implementing that node in an optimal mapping solution.
- **Matching**: Generate the K -LUT mapping solution based on the node labels.

29

Delay Model of FlowMap

■ *Unit delay model*

- No gate delay.
- No interconnect delay within a cluster.
- Delay $D = 1$ between two clusters.
- Delay minimization is equivalent to depth minimization under unit delay model



Delay = 3

30

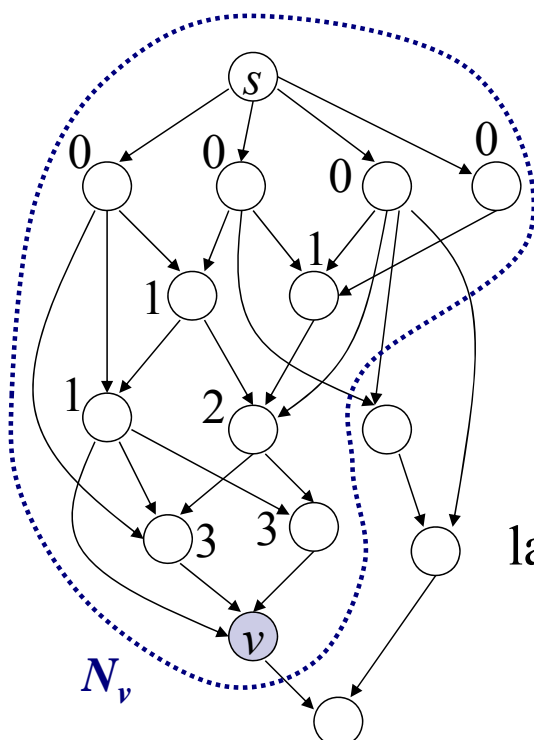
Labeling Phase of FlowMap

■ Labeling Phase

- Process the nodes in *topological order*.
- $\text{label}(v)$ = depth of the optimal K -LUT mapping solution for node v .

31

Labeling Phase of FlowMap



Want to find:

- (i) $\text{label}(v)$, and
- (ii) the K -LUT for v in the optimal mapping solution of N_v .

$$\text{label}(v) = \min_{\substack{(X,Y) \text{ is a} \\ K\text{-feasible cut in } N_v}} \{h(X,Y)\} + 1$$

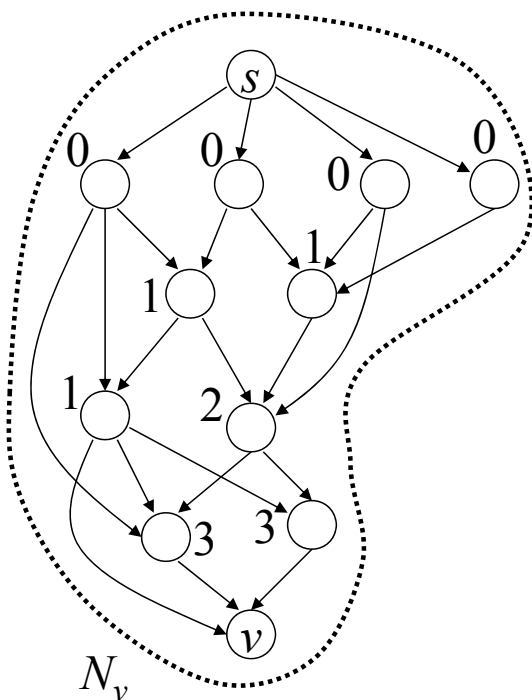
32

Labeling Phase of FlowMap

- Lemma 1: $\text{label}(v) = p$ or $p+1$, where p is the maximum label of the input nodes to v . (Why?)
- Computation of $\text{label}(v)$:
 - Check if there is a K -feasible cut (X, Y) in N_v of height $p-1$.
 - If yes, $\text{label}(v) = p$, Y and v are pushed into the same K -LUT.
 - Else, $\text{label}(v) = p+1$, use a new K -LUT for node v .

33

Labeling Phase of FlowMap

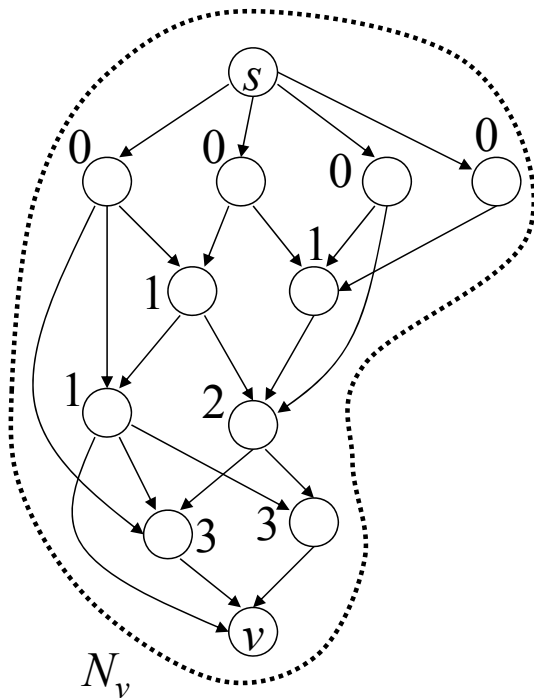


In this example, $p = 3$, i.e., $\text{label}(v) = 3$ or 4

How to determine if there is a K -feasible cut for N_v with height 2?

34

Labeling Phase of FlowMap



Let $K = 3$.

How to determine if there is a 3-feasible cut for N_v with height 2?

Note: If $\text{label}(v) = 3$, which nodes must be in the same LUT as v ?

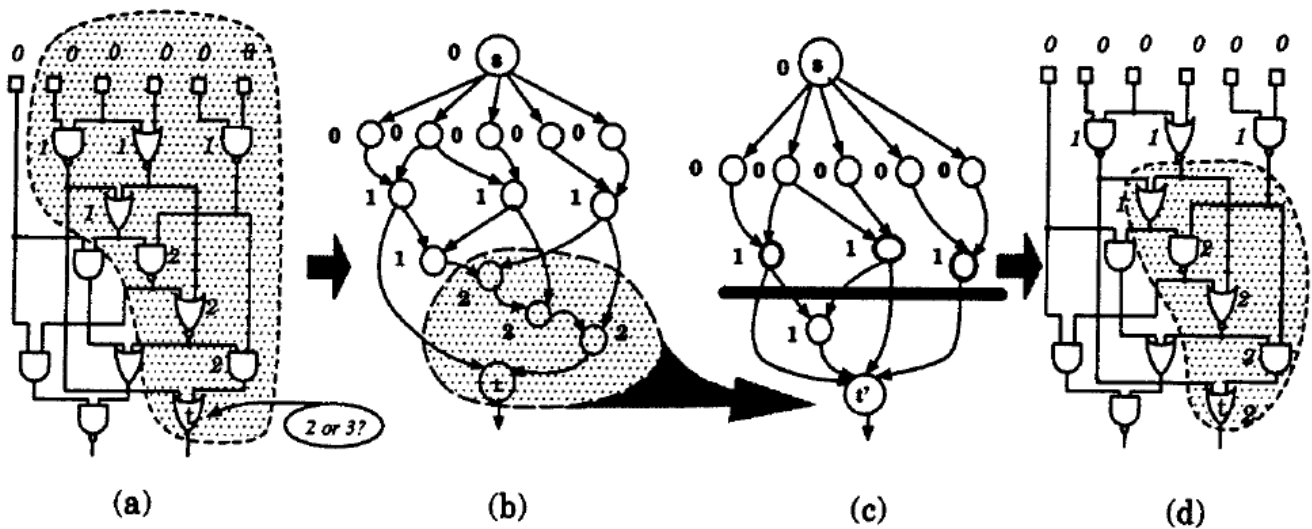
35

Labeling Phase of FlowMap

- Question: How to determine if there is a K -feasible cut of height $p-1$ in N_v ?
(Hint: You can use the *network flow* method.)
- What is the time complexity of this method?

36

Example



- (a) Original network and labels of nodes in N_t
- (b) Transform into flow network
- (c) Collapse nodes with largest labels, compute 3-feasible cut
- (d) Get label(t) and LUT(t)

37

Time Complexity of Labeling Phase

Labels of all the nodes in the subject tree can be computed in $O(Kmn)$ time where n = no. of nodes and m = no. of edges. Since $m = O(n)$ for any K -bounded input network, the time complexity is $O(n^2)$.

38



Mapping Phase of FlowMap

1. S = Set of primary output nodes
2. Choose any $t \in S$.
3. Use $K\text{-LUT}(t)$ as one of the $K\text{-LUTs}$.
4. Delete t from S .
5. Add $\text{Input}(K\text{-LUT}(t))$ to S .
6. Goto step 2.

39



FlowMap Complexity

1. Labeling phase takes $O(Kmn)$ time.
2. Mapping phase takes $O(n)$ time.
3. Hence, the total time complexity is $O(Kmn)$, i.e., $O(n^2)$.

40

Minimization of #LUTs in FlowMap

- In the Labeling Phase, we use the network flow method to obtain min-cuts.
- Should we take the *min-area min-cut* or the *max-area min-cut* in order to reduce the total no. of LUTs used?
- How to obtain a max-area min-cut?

41

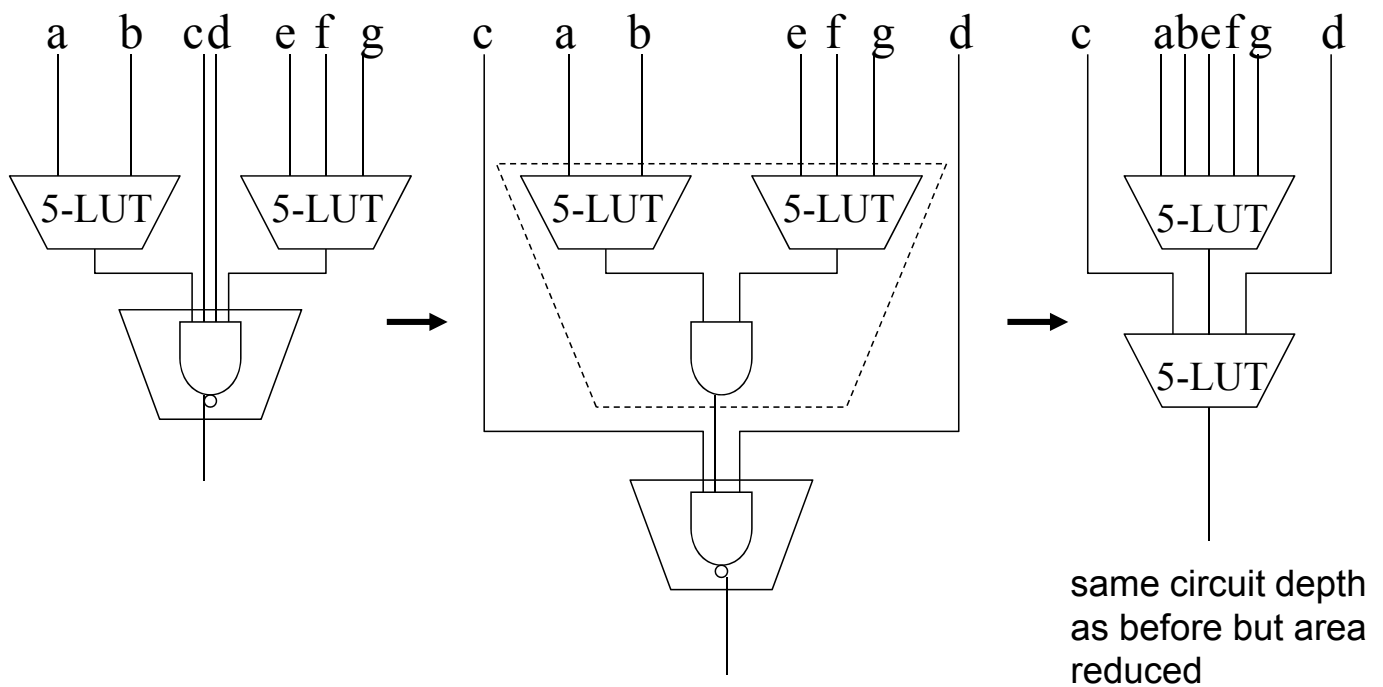
Minimization of #LUTs in FlowMap

- In order to reduce the total no. of LUTs used, we may maximize the cut volume during labeling phase.
- For each node, if min-cut size $\leq K$
 - Find a max-area min-cut.
 - Increase the volume of the cut until the node-cut size exceeds K .
 - Use the last K -feasible cut as the solution.

42

Post-Processing for Area Reduction

■ Gate decomposition

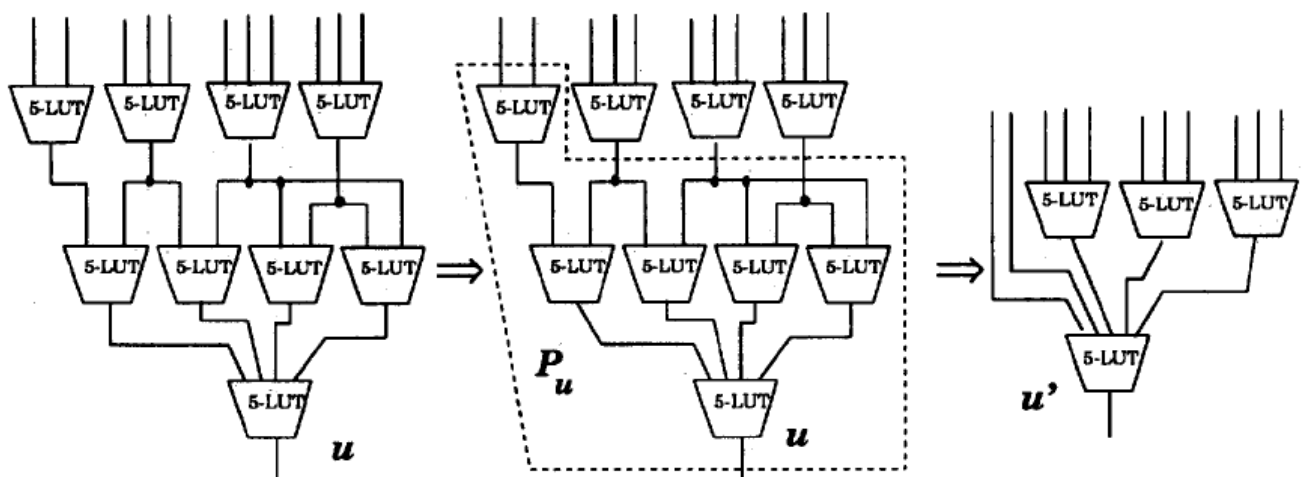


43

Post-Processing for Area Reduction

■ Flow pack

- Any K-feasible cone in an initial mapping solution can be replaced by a single K-LUT



44

DAOmap: optimal depth mapping with area optimization

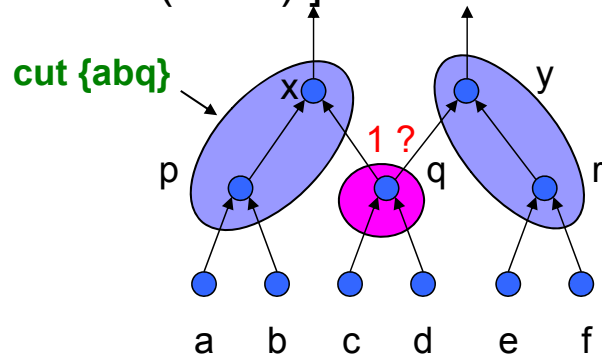
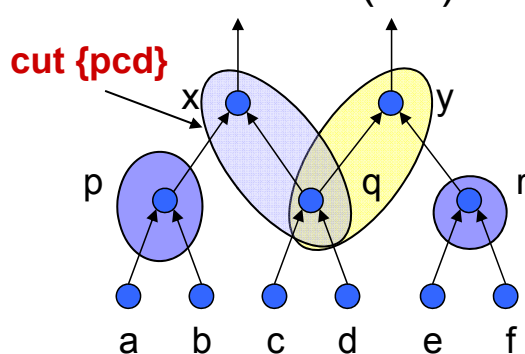
- In a solution of FlowMap, many nodes have extra *slack*.
- For node with slack
 - Pick a cut with *minimum area cost* s.t. timing increase does not exceed the slack
- For node with no slack
 - Among all cuts with *minimum timing cost*, pick one with *minimum area cost*
- Discourage unnecessary logic duplication.
- Re-run mapping phase multiple times and adjust area costs heuristically.

45

How to Measure Area?

Suppose we use the naïve definition:

$$\text{Area (cut)} = 1 + [\sum \text{area (fanin)}]$$



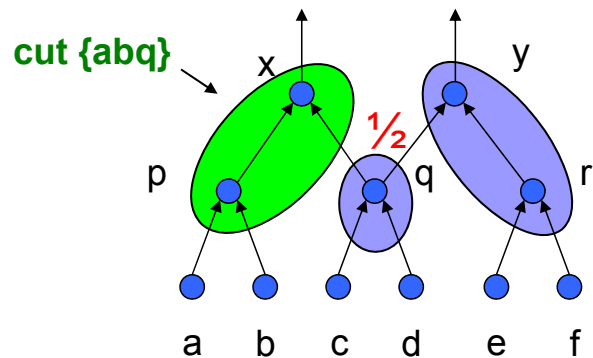
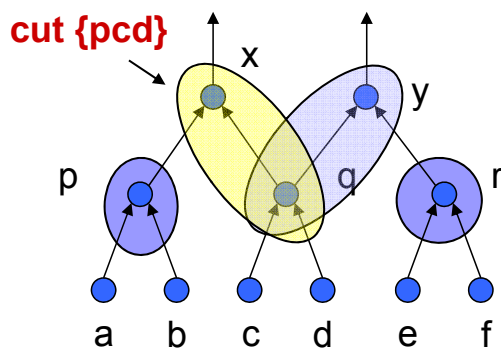
Naïve definition says both cuts are equally good in area

Naïve definition ignores sharing due to multiple fanouts

46

Area-flow (“Effective Area”)

area-flow (cut) = $[1 + \sum \text{area-flow (fanin)}] / \text{no. of fanouts}$



Area-flow recognizes that **cut {abq}** is better

Area-flow accounts for sharing

47

WireMap: mapping for improved routability

- Fewer pin-to-pin connections should make the design easier to place and route

- ☐ Could we come up with a mapping algorithm to minimize the total # of connections in a design?

- Consider *edge-flow* for cut ranking

- ☐ edge-flow (cut) = $[\text{cut size} + \sum \text{edge-flow (fanin)}] / \text{no. of fanouts}$

48

WireMap Algorithm

Input: And-Inverter Graph

1. Compute K -feasible cuts for each node
 2. Compute best arrival time at each node
 - In reverse topological order (from POs to PIs)
 - Compute the depth of all cuts and choose the best one
 3. Perform area **and edge** recovery w/o violating timing in topological order
 - Heuristic using area flow **and edge flow**
 - Heuristic using exact local area **and exact local edge count**
 4. Choose the best cover
- Output:** Mapped Netlist

Choose cut with smallest area-flow, break ties with lower edge-flow

Choose cut with smallest area, break ties with lower no. of edges

49

References

- “FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs”, *TCAD*, Jan. 1994
- “DAOmap: A Depth-Optimal Area Optimization Mapping Algorithm for FPGA Designs”, *ICCAD'04*
- “WireMap: FPGA Technology Mapping for Improved Routability”, in *FPGA'08*