Advanced Digital Logic Design – EECS 303

http://ziyang.eecs.northwestern.edu/eecs303/

Teacher:	Robert Dick
Office:	L477 Tech
Email:	dickrp@northwestern.edu
Phone:	847–467–2298

NORTHWESTERN UNIVERSITY

Today's topics

- Technology mapping
  - Binate covering
  - Tree mapping
- Arithmetic circuits
  - Number systems
  - Adders

Technology mapping

- We know how to minimize two-level and multi-level logic
- Generally expressed in terms of simple gates, e.g., NAND2s and NOTs
- Need to map to efficient implementation in target technology
- Target technology consists of a set of gates
- Each has area, power consumption, and timing properties
- Map to those gates minimizing some combination of area, power, and delay

Technology mapping

Given:

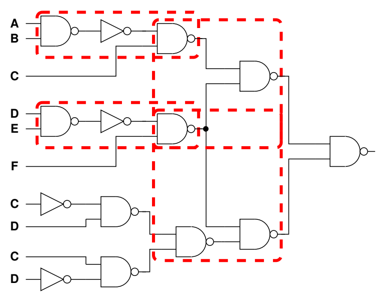
- Simplified and decomposed (reduced) logic graph (DAG)
  - Tree makes things easier
- Composed of *NOT* and *NAND2* gates
- Library: *NOT*, *NAND2*, *NAND3*, *ANDOR4 (AO4)*, and *XNOR* gates

Decide minimal area/delay/power cost mapping from logic tree to technology library

Technology mapping example library

gate	area cost
NOT	1
NAND2	2
NAND3	3
NAND4	4
XNOR2	5
ANDOR4	4

Potentially overlapping maps



Technology mapping for area as binate covering

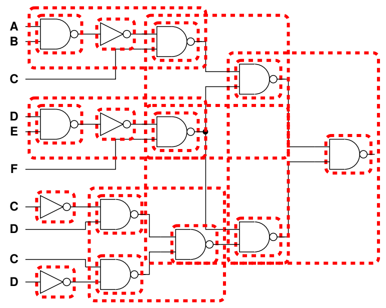
- Find all possible matches
- However, complete cover insufficient
  - Need match outputs to align with inputs of other matches
- Can represent problem as binate covering

Binate covering

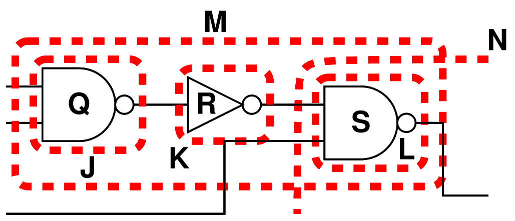
term	column			
	A	B	C	D
J_1	1			
J_2	1	1		
J_3		1	1	
J_4			1	1
J_5	1		0	

- Find a set of columns, S, such that, for every row
  - A 1-colum in the row is in S or. . .
  - . . . a 0-column in the row is not in S

DAG mapping binate covering formulation



DAG mapping binate covering formulation



$(\overline{J} \vee K) \wedge (\overline{K} \vee J) \wedge (\overline{K} \vee L \vee N) \wedge (\overline{L} \vee K) \wedge (\overline{N} \vee K)$

DAG mapping binate covering formulation

Outputs feed directly into next gate's inputs

$$(\overline{J} \vee K) \wedge (\overline{K} \vee J) \wedge (\overline{K} \vee L \vee N) \wedge (\overline{L} \vee K) \wedge (\overline{N} \vee K)$$

Also ensure primary outputs are available (omitted from example)

constraint	gate				
	J	K	L	M	N
Q	1			1	
R		1		1	
S			1	1	1
$\overline{J} \vee K$	0	1			
$\overline{K} \vee J$	1	0			
$\overline{K} \vee L \vee N$		0	1		1
$\overline{L} \vee K$		1	0		
$\overline{N} \vee K$		1			0

Problems with binate covering

- Although there are good heuristics to speed up unate covering, binate covering appears to be a harder problem
  - Intractable for large problem instances
- Cost function must be independent of other portions of solution
  - Can use area but can't use delay or power
- Can use a fast alternative for tree, not DAG covering

Tree covering

- Split DAG at multiple output nodes to form trees
- Optimally and quickly map trees using dynamic programming
- Reconnect result into a DAG
- Locally improve connection points
- Result: Nearly (but not quite) optimal, fast DAG mapping

Technology mapping for area complexity

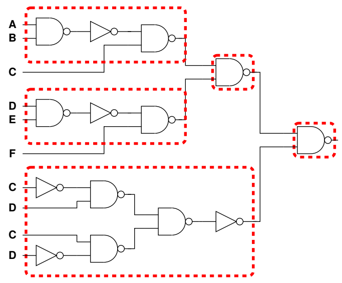
- Reduced logic tree $RLT = (V, E)$
- Technology library with T gates
- Maximum gate size of S, i.e., no gate covers more than S vertices

Algorithm is

- Linear in $|V|$
- Linear in T
- Exponential in S

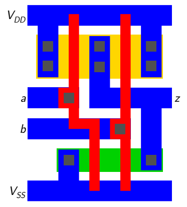
Tree technology mapping for area (detailed)

$2 \cdot NAND3(3) + XNOR2(5) + 2 \cdot NAND2(2) = 15$



Technology mapping for speed

- If each gate had a different constant delay, could use simple dynamic programming algorithm
  - This works pretty well but is suboptimal
- However, each gate's speed depends on next gate's input load



Two-pass delay optimization

Problem: Optimize area under timing constraint

1. Find the set of all possible pin-loads
2. From leafs, build array of minimal-area solutions, one for each pin-load
3. Calculate the arrival time for each possible mapping (pin-load)
4. For each point, select the optimal area solution for each pin-load
5. Propagate optimal solution backwards from output load

- Note, that this isn't efficient given a large number of pin-loads
  - Discretize

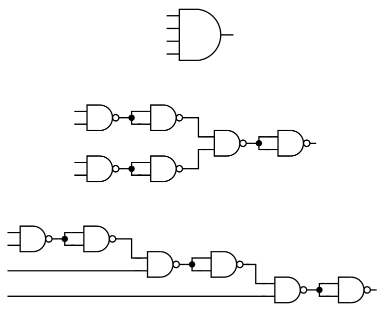
Technology mapping

- Are circuits really trees?
- What happens when they are not?

Decomposition for technology mapping

- Start from an arbitrary DAG of gates
- Want canonical DAG for technology mapping
  - E.g., NAND2 gates only
    - NAND2 with inputs tied is a NOT
- However, decomposition is not unique

AND4 $\rightarrow$ NAND2



Non-canonical decomposition

- Could decide to allow more complex gates in subject DAG
  - However, little opportunity for optimization
- Instead, consider all equivalent mappings for each library gate
- Can't consider numerous subject DAG
  - Complexity exponential
- Restated: Store many matches for each library gate but use only one subject DAG decomposition

Notes on optimality

- It is common for algorithm designers to come up with elegant and optimal algorithms
- Look closely at their assumptions!
- It is often, first, necessary to formulate a problem in a simplified form to arrive at an optimal solution
  - An optimal solution to the simplified form is not necessarily an optimal solution to the original problem

Notes on optimality

- The technology mapping algorithms in this lecture are polynomial-time (fast) and optimal…
- …however, they aren't polynomial-time (fast) in every term
  - Maximum library gate size
- …or optimal for the original problem
  - The original problem can be a DAG, instead of a tree
  - The decomposition is not unique
    - Better solutions might be possible for other decompositions

Notes on optimality

- The algorithms are high-quality and efficient
- However
  - They are only optimal with respect to the simplified inputs
  - They are only polynomial in the most important terms
- This situation is very common in EDA/CAD because the original problems are often $\mathcal{NP}$-complete
  - Too slow to solve for large problem instances
- Whenever you see or use the word "optimal", think very carefully about what is meant

Technology mapping summary

- Technology mapping is taking a set of functions and determining how to implement them using gates from a library
- Optimal and fast approaches exist for trees
- However, no optimal solution is known for arbitrary circuit topologies
- This is what you are doing when you type “map” in SIS

Next lecture

- Multipliers and ALUs
- Sequential logic networks
- Latches (RS Latch)
- Flip-flops (D and JK)
- Timing issues (setup and hold times)

Reading assignment

- M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, third edition, 2004
- Chapter 5