

Lecture 15 Technology Mapping

Technology Mapping

- Previously we performed logic minimization
 - Minimized size of equation and/or number of literals
 - Didn't consider how we will implement the circuit

$$F(w, x, y, z) = w'x'y'z' + w'x'yz' + w'x'yz' + w'xy'z' + w'xyz + w'xyz' + wxy'z + wxyz + wx'y'z + wx'yz$$



$$F = w'z' + wz + yz$$

- Technology mapping
 - Transforms technology independent logic network into gates implemented with a technology library
 - standard cell library, gate array library, look-up tables (for FPGAs)

$$F(a, b, c) = a + a'b'c' + bc'$$



$$F = a + c'$$

$$F(h, i, j, k) = h'i'j'k' + hi'j'k' + h'i'k + h'ij + jk$$



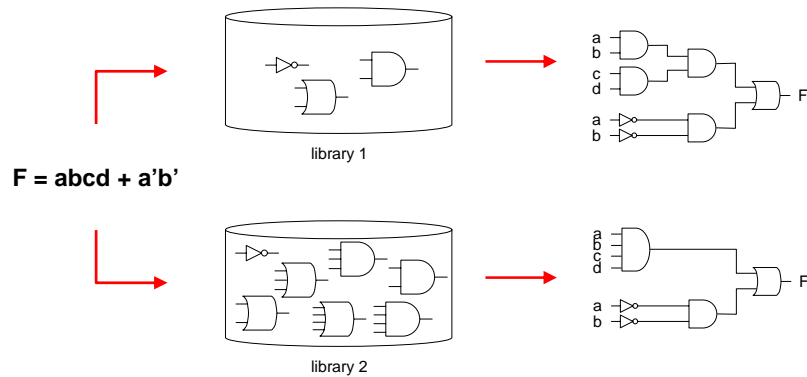
$$F = hi'j'k' + h'i' + jk$$



$$\left\{ \begin{array}{l} F = w'z' + wz + yz \\ F = a + c' \\ F = hi'j'k' + h'i' + jk \end{array} \right.$$

Technology Mapping Example

- Standard cell ASIC technology
 - Uses library of pre-layed-out gates or small pieces technology known as cells
 - Designer instantiates and connects these cell to implement a digital circuit



ECE 474a/575a
Susan Lysecky

3 of 23

Technology Mapping Phases

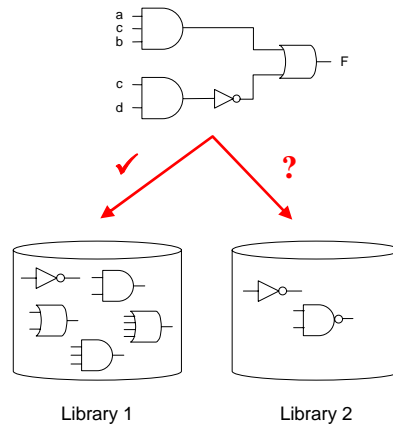
- Technology mapping broken down into 3 phases
 - Decomposition
 - Translate logic network into some primitive cells to create a simple structure that aids the mapping process
 - Pattern Matching
 - Analysis on the circuit library to determine the set of matches for all nodes in the circuit
 - Covering
 - Identify the best possible matches (based on particular cost function) for logic network so that every node is covered at least once and the functionality is maintained

ECE 474a/575a
Susan Lysecky

4 of 23

Decomposition

- Depending on the input function, circuit may need to be modified
- Cell library typically limited to a few primitives functions with few inputs
 - AND, OR, INV (2-input only)
 - NAND (2-, 3-, 4-input)
 - NOR (2-, 3-, 4-input)
- Most practical cell libraries limit primitives to 2-input NAND gates and INV
 - 2-input NOR gates work equally well



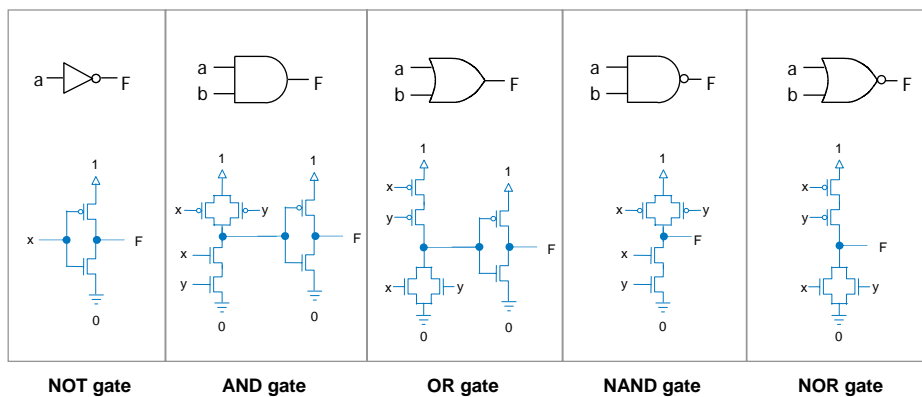
ECE 474a/575a
Susan Lysecky

5 of 23

Why NAND and NOR Gates?

CMOS Transistor Level Gate Implementation

- At the low level NAND/ NOR gates are require fewer transistors than AND/OR gates and are more desirable to use



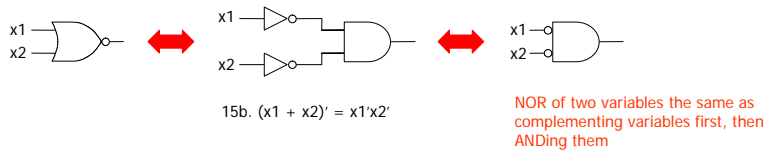
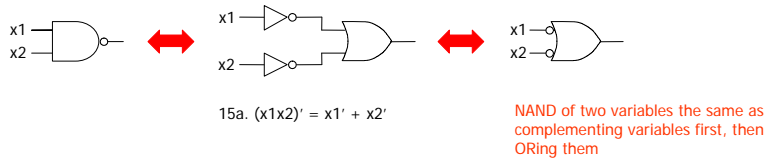
ECE 474a/575a
Susan Lysecky

6 of 23

NAND and NOR Logic Networks

DeMorgan's Theorem as Logic Circuits

- Can we use NAND and NOR gates to implement various logic circuits?
 - Let's look at logic gate implementation of DeMorgan's Theorem

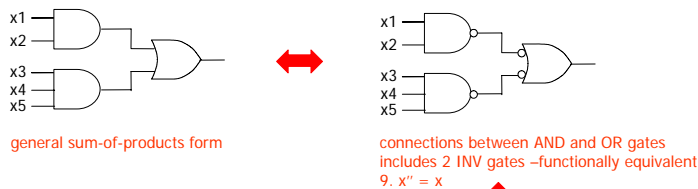


NOT gates represented with inversion bubbles

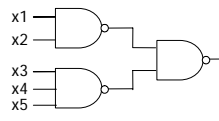
Example 1

NAND Logic Networks

- Transform sum-of-products function into a network using only NAND gates



Just showed this equivalency - DeMorgan's Theorem

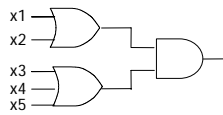


Network can be transformed into a network of NAND gates

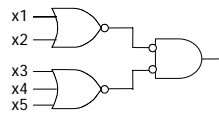
Example 2

NOR Logic Networks

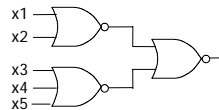
- Any function can be implemented in product-of-sums form
 - We can transform into a network using only NOR gates



general product-of-sums form



connections between OR and AND gates includes 2 INV gates –functionally equivalent
 $\bar{x''} = x$



Network can be transformed into a network of NOR gates

Just showed this equivalency – DeMorgan's Theorem

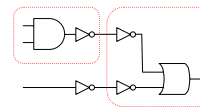


ECE 474a/575a
 Susan Lysecky

9 of 23

Decomposition

- Many ways to decompose logic network
 - Based on delay, size, power, etc.
- Typically not limited to 1 decomposition
 - Multiple versions provided to next step to allow more opportunity for pattern matching and covering phases
 - Additional strategy to increase potential pattern matches – add cascading inverters
 - Potential to be helpful
 - Make sure leftover cascading inverters covered by wires
- Decomposition phase greatly impacts resulting circuit
 - Tremendous work that considers timing-, layout-, area-driven decomposition techniques

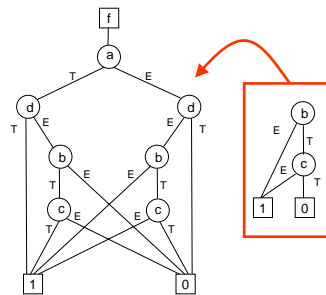
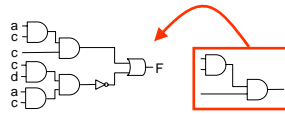


ECE 474a/575a
 Susan Lysecky

10 of 23

Pattern Matching

- Several pattern matching techniques available
 - Combination of matchers frequently used
- Structural matcher
 - Pure structural isomorphic tests
- Boolean matcher
 - Uses BDDs to find matches based on logic functions
 - Independent of the actual decomposition
- PLA matcher
 - Similar to Boolean matcher, differs in the representation of the function
 - Utilizes truth tables of gates (AND, OR)



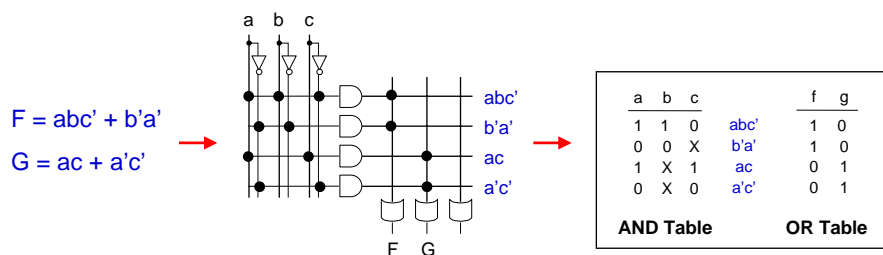
ECE 474a/575a
Susan Lysecky

11 of 23

Pattern Matching

PLA Matcher – AND/OR Truth Table Representation

- Programmable logic array (PLA)
 - Set of programmable AND array linked to programmable OR array
 - AND Table mimics AND array
 - Table as wide as number of entries
 - 1 = term positive, 0 = term negative, X = term doesn't apply
 - OR Table mimics OR array
 - Table as wide as number of outputs
 - 1 = term belong to output, 0 = term doesn't belong to output



ECE 474a/575a
Susan Lysecky

12 of 23

Covering

- After network decomposed and patterns generated need to select subset of patterns so entire network is covered
 - Objective function indicates which subset to choose (delay, cost, reliability, power)
- Represent these choices with covering matrix
 - Rows represent nodes in graph
 - Column represent pattern matches
 - "1" in matrix signifies node in row covered by pattern in column
- How to choose subset of columns?
 - Find set of columns so all rows covered
 - Choose columns to obtain minimal cost
 - Each match has inputs available from output of other matches

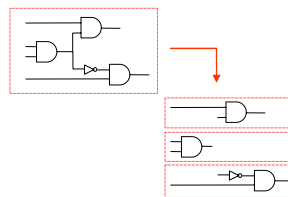
		patterns found			
		P ₁	P ₂	...	P _n
nodes in logic network	N ₁	1			1
	N ₂		1		
	...		1		1
	N _m	1			

Does this look familiar?
Binate covering problem

We already know this is hard to find exact solution – look at heuristics

Covering Dynamic Programming

- Propose mapping method based on tree-covering
 - Network partitioned into forest of trees
 - Solve problem by solving for each tree
 - Stitch trees back together
 - Motivated by existence of efficient dynamic programming algorithm available*
- General idea of dynamic programming
 - Optimal substructure
 - Overlapping sub-problems
 - Memoization



optimal solutions of sub-problems can be used to find the optimal solutions of the overall problem

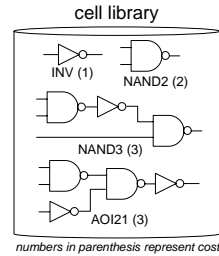
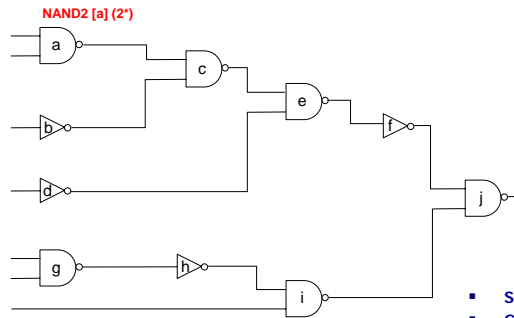
same sub-problems are used to solve many different larger problems.

savings and re-use of already-computed sub-solutions

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- Start with leaves to root
- Given node, look for best cover of subtree based on node
- Node a
 - NAND2 is only match, cost of 2
 - * indicates best solution for subtree
 - [] indicates which nodes are covered by pattern

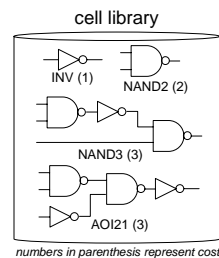
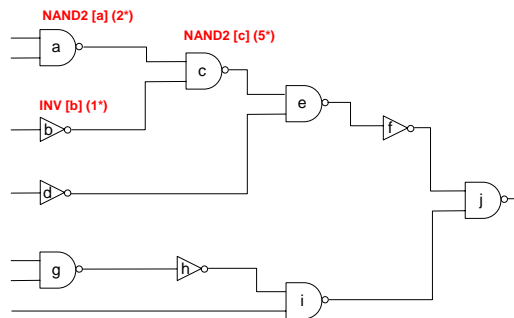
ECE 474a/575a
Susan Lysecky

15 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- Node b
 - INV is only match, cost of 1
- Node c
 - NAND2 is only match, cost of 2
 - Cost equal match for the node plus cost to cover subtrees
 - $2 \text{ (node a)} + 1 \text{ (node b)} + 2 \text{ (node c)} = 5$

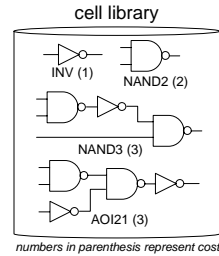
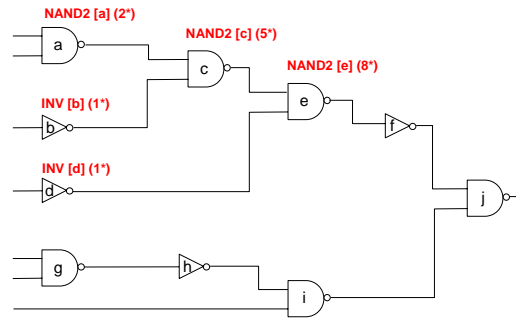
ECE 474a/575a
Susan Lysecky

16 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- **Node d**
 - INV is only match, cost of 1
- **Node e**
 - NAND2 is only match
 - cost = 5 (subtree c) + 1 (node d) + 2 (node e) = 8

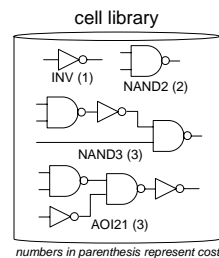
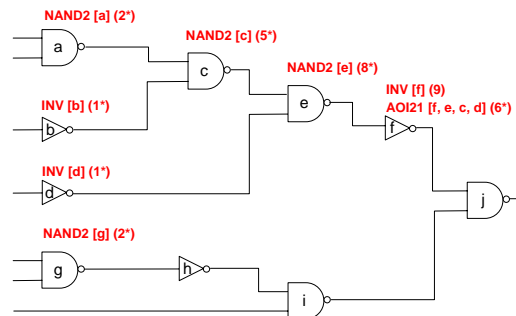
ECE 474a/575a
Susan Lysecky

17 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- **Node f**
 - INV is a possible match, cost of 9 (8 + 1)
 - AOI21 is a possible match, cost of 6 (3+2+1)
- **Node g**
 - NAND2 is only match, cost of 2

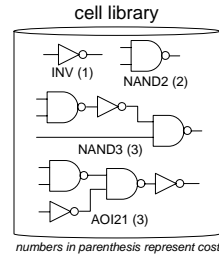
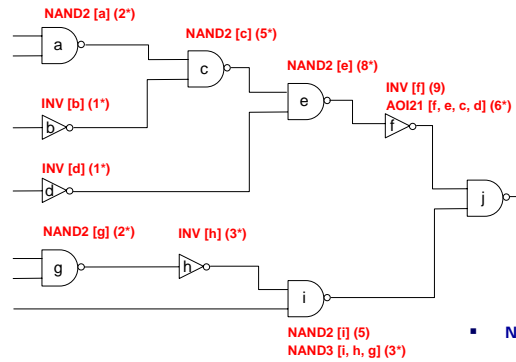
ECE 474a/575a
Susan Lysecky

18 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- **Node h**
 - INV is only match, cost of 3 (2 + 1)
- **Node i**
 - NAND2 is a possible match, cost is 5
 - NAND3 is a possible match, cost is 3

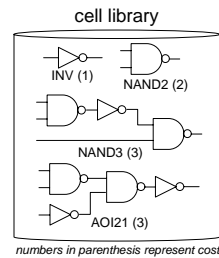
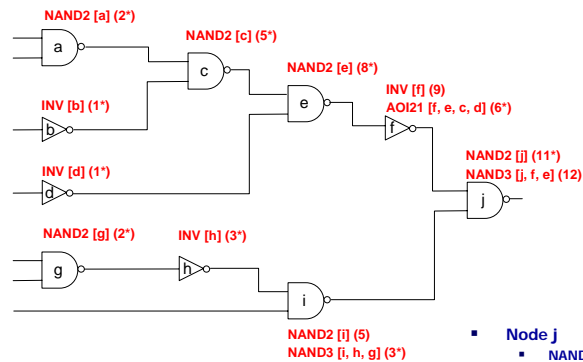
ECE 474a/575a
Susan Lysecky

19 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- **Node j**
 - NAND2 is a possible match, cost is 11
 - NAND3 is a possible match, cost is 12

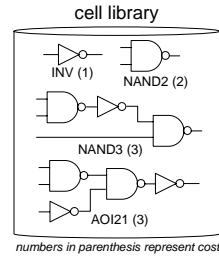
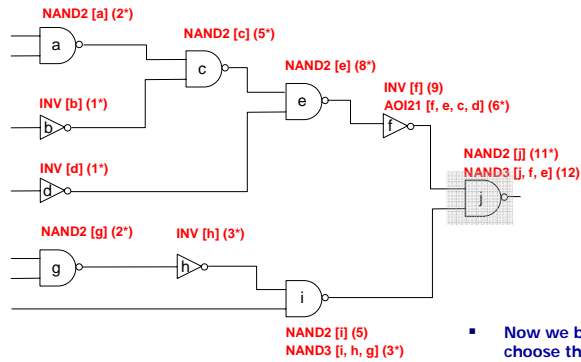
ECE 474a/575a
Susan Lysecky

20 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- Now we backtrack starting from root and choose the best covering seen
- At node j, NAND2 gives best cost

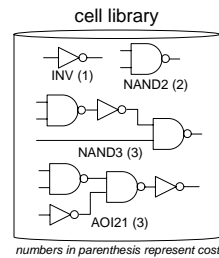
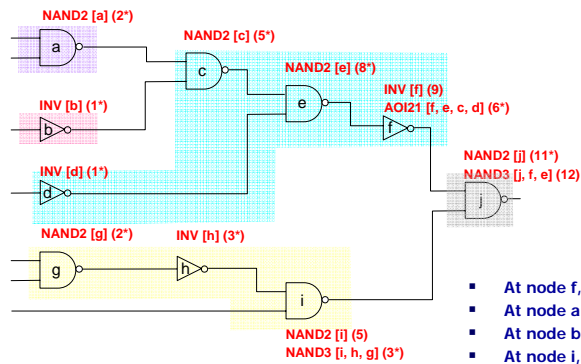
ECE 474a/575a
Susan Lysecky

21 of 23

Example 3

Dynamic Programming

Cover following circuit using dynamic programming approach



- At node f, AOI21 gives best cost
- At node a, NAND2 gives best cost
- At node b, INV gives best cost
- At node i, NAND3 gives best cost
- FINAL COST = 11

ECE 474a/575a
Susan Lysecky

22 of 23

Conclusion

- Technology mapping phases
 - Decomposition
 - Pattern Matching
 - Covering
- Only considered a few techniques
 - Many more exist with alternative constraints
 - Delay, Power, Reliability, Layout, Congestion, etc.
 - Others look at two phase technology mapping
 - Simplify technology mapping by finding a minimal area solution
 - Apply post technology mapping transformations to customize solution to specific interest