

# CS137: Electronic Design Automation

Day 2: January 7, 2004  
Covering



CALTECH CS137 Winter2004 -- DeHon

## Why covering now?

- Nice/simple cost model
- problem can be solved well (somewhat clever solution)
- general/powerful technique
- show off special cases (harder/easier cases)
- show off things that make hard
- show off bounding

CALTECH CS137 Winter2004 -- DeHon

# Problem

- Implement a “gate-level” netlist in terms of some library of primitives
- General
  - easy to change technology
  - easy to experiment with library requirements (benefits of new cells...)

CALTECH CS137 Winter2004 -- DeHon

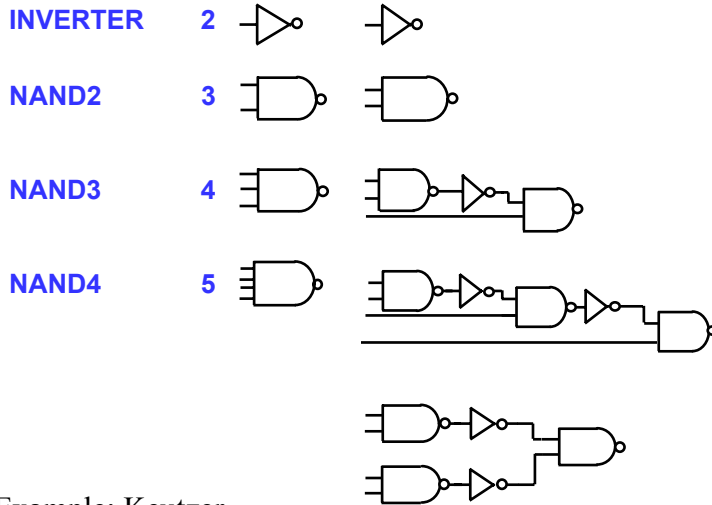
# Input

- netlist
- library
- represent both in normal form
  - nand gate
  - inverters

CALTECH CS137 Winter2004 -- DeHon

# Elements of a library - 1

Element/Area Cost    Tree Representation (normal form)

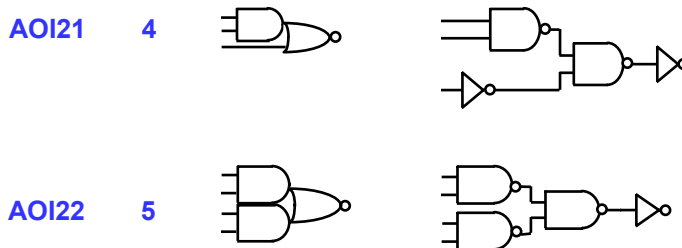


Example: Keutzer

CALTECH CS137 Winter2004 -- DeHon

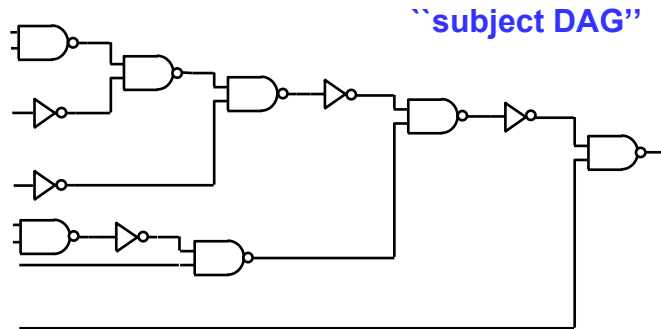
# Elements of a library - 2

Element/Area Cost    Tree Representation (normal form)



CALTECH CS137 Winter2004 -- DeHon

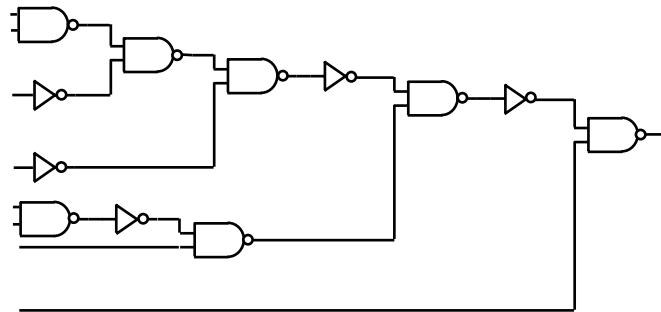
# Input Circuit Netlist



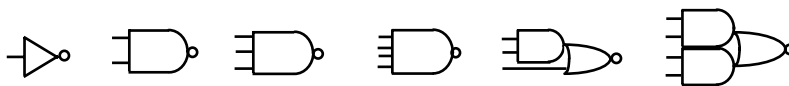
CALTECH CS137 Winter2004 -- DeHon

# Problem statement

Find an ``optimal`` (in area, delay, power) mapping of this circuit (DAG)



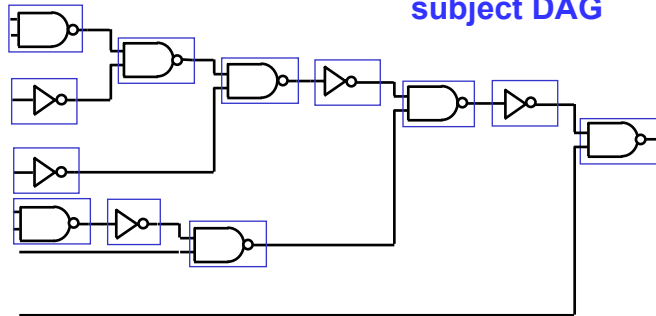
into this library



CALTECH CS137 Winter2004 -- DeHon

# What's the problem? Trivial Covering

subject DAG



7      NAND2 (3) = 21  
5      INV     (2) = 10  
Area cost 31

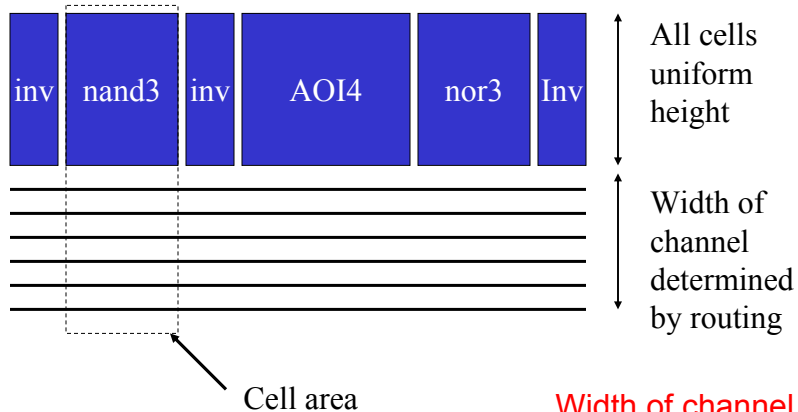
## Cost Models

## Cost Model: Area

- **Assume:** Area in gates
- or, at least, can pick an area/gate
  - so proportional to gates
- *e.g.*
  - Standard Cell design
  - Standard Cell/route over cell
  - gate array

CALTECH CS137 Winter2004 -- DeHon

## Standard Cell Area



Width of channel fairly constant?

CALTECH CS137 Winter2004 -- DeHon

# Cost Model: Delay

- Delay in gates
  - at least assignable to gates
    - $T_{\text{wire}} \ll T_{\text{gate}}$
    - $T_{\text{wire}} \approx \text{constant}$
  - delay exclusively/predominantly in gates
    - Gates have  $C_{\text{out}}$ ,  $C_{\text{in}}$
    - lump capacitance for output drive
    - $\text{delay} \sim T_{\text{gate}} + \text{fanout} \times C_{\text{in}}$
    - $C_{\text{wire}} \ll C_{\text{in}}$
    - or  $C_{\text{wire}}$  can lump with  $C_{\text{out}}/T_{\text{gate}}$

CALTECH CS137 Winter2004 -- DeHon

# Cost Models

- Why do I show you models?
  - not clear there's one "right" model
  - changes over time
  - you're going to encounter many different kinds of problems
  - want you to see formulations so can critique and develop own
  - simple make problems tractable
    - are surprisingly adequate
  - simple, at least, help bound solutions

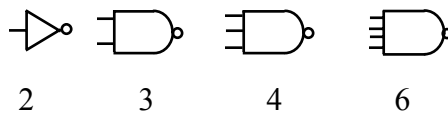
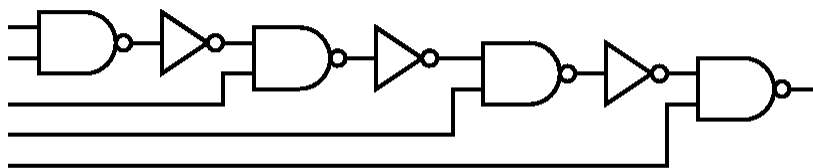
CALTECH CS137 Winter2004 -- DeHon

# Approaches

CALTECH CS137 Winter2004 -- DeHon

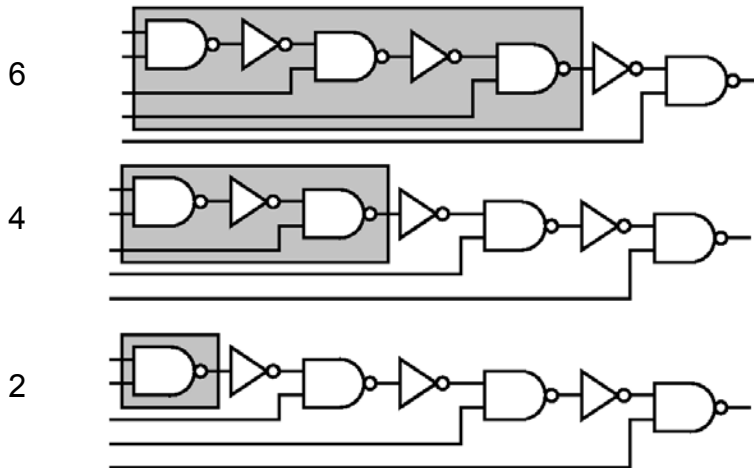
## Greedy work?

- Greedy = pick next locally “best” choice



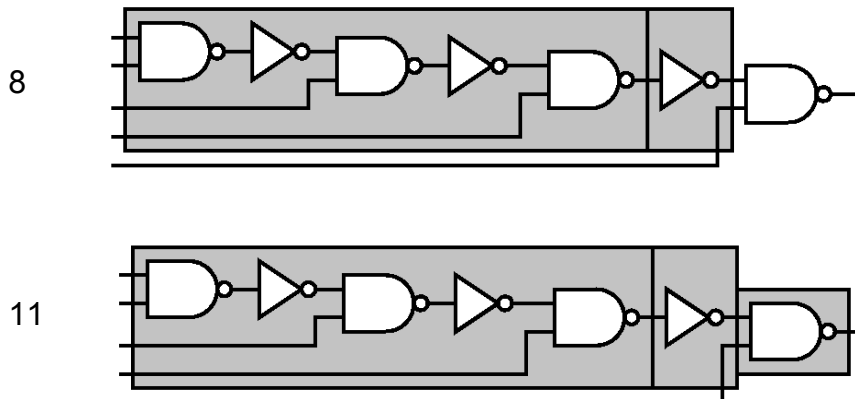
CALTECH CS137 Winter2004 -- DeHon

## Greedy In→Out



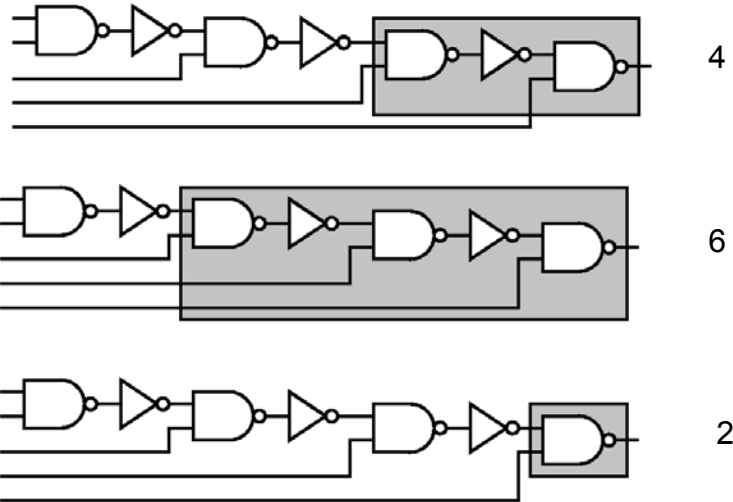
CALTECH CS137 Winter2004 -- DeHon

## Greedy In→Out



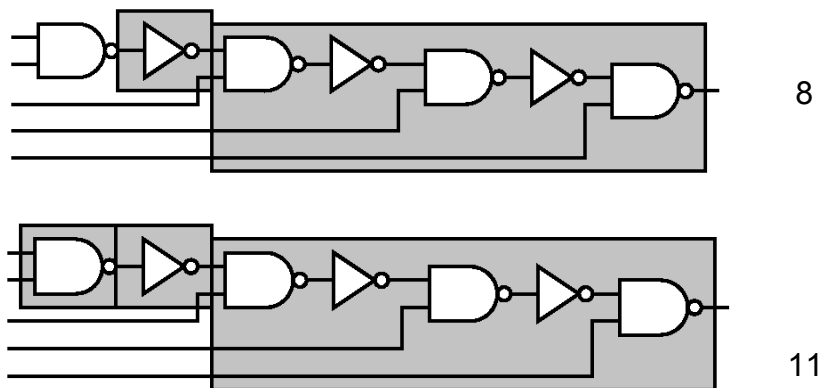
CALTECH CS137 Winter2004 -- DeHon

## Greedy Out→In



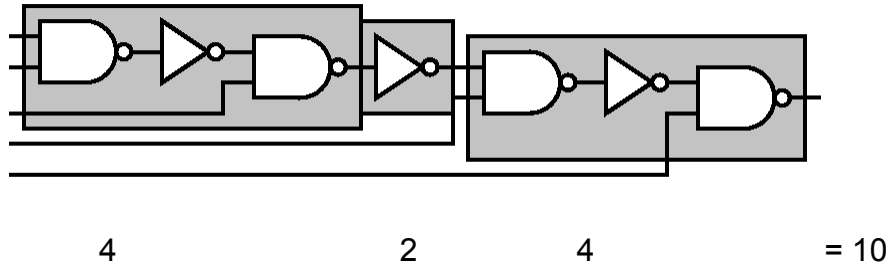
CALTECH CS137 Winter2004 -- DeHon

## Greedy Out→In



CALTECH CS137 Winter2004 -- DeHon

But...



CALTECH CS137 Winter2004 -- DeHon

## Greedy Problem

- What happens in the future (elsewhere in circuit) will determine what should be done at this point in the circuit.
- Can't just pick best thing for now and be done.

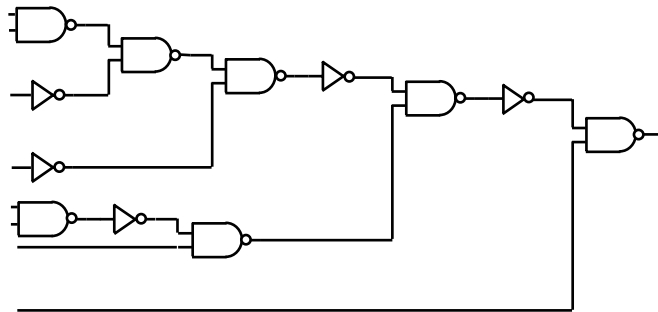
CALTECH CS137 Winter2004 -- DeHon

## Brute force?

- Pick a node (output)
- Consider
  - all possible gates which may cover that node
  - branch on all inputs after cover
  - pick least cost node

CALTECH CS137 Winter2004 -- DeHon

## Pick a Node



CALTECH CS137 Winter2004 -- DeHon

## Brute force?

- Pick a node (output)
- Consider
  - all possible gates which may cover that node
  - recurse on all inputs after cover
  - pick least cost node
- Explore all possible covers
  - can find optimum

CALTECH CS137 Winter2004 -- DeHon

## Analyze brute force?

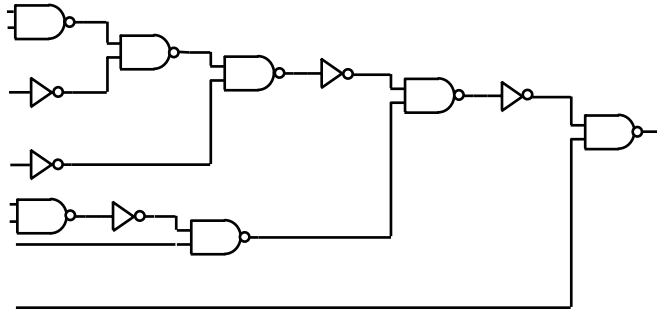
- Time?

$$T_{brute}(node) = \sum_{i=0}^{\max \text{ pattern}} \left( T_{match}(P_i) + \sum_{j=0}^{\max \text{ in}} (T_{brute}(\text{in } j)) \right)$$

- Say P patterns, linear time to match each
  - (can do better...)
- P-way branch at each node...
- ...exponential
  - $O(\text{depth}^P)$

CALTECH CS137 Winter2004 -- DeHon

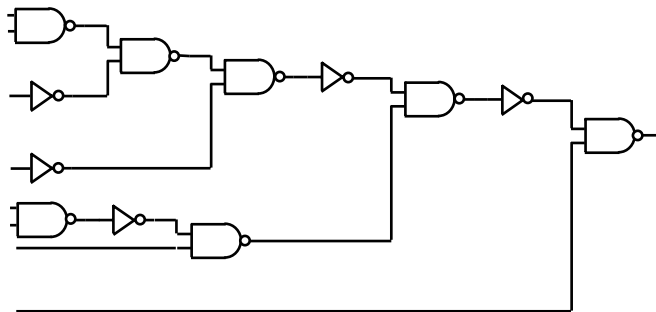
## Structure inherent in problem to exploit?



CALTECH CS137 Winter2004 -- DeHon

## Structure inherent in problem to exploit?

- There are only N unique nodes to cover!



CALTECH CS137 Winter2004 -- DeHon

## Structure

- **If** subtree solutions do not depend on what happens outside of its subtree
  - separate tree
  - farther up tree
- Should only have to look at N nodes.
- $\text{Time}(N) = N * P * T(\text{match})$ 
  - w/ P fixed/bounded, technically linear in N
  - w/ cleverness work isn't  $P * T(\text{match})$  at every node

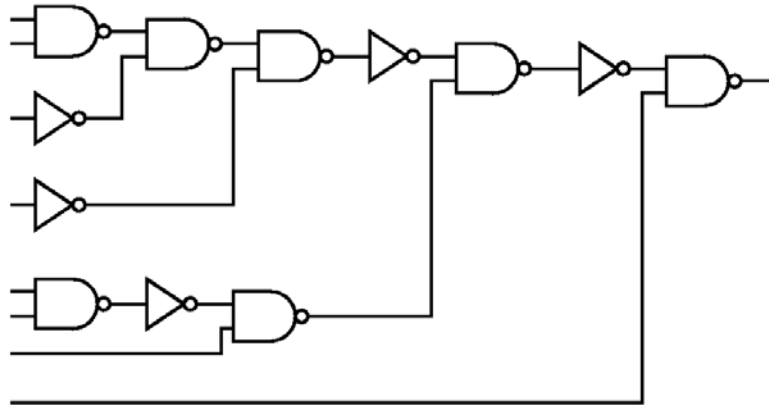
CALTECH CS137 Winter2004 -- DeHon

## Idea Re-iterated

- Work from inputs
- Optimal solution to subproblem is contained in optimal, global solution
- Find optimal cover for each node
- Optimal cover:
  - examine all gates at this node
  - look at cost of gate and its inputs
  - pick least

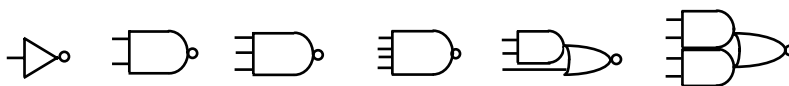
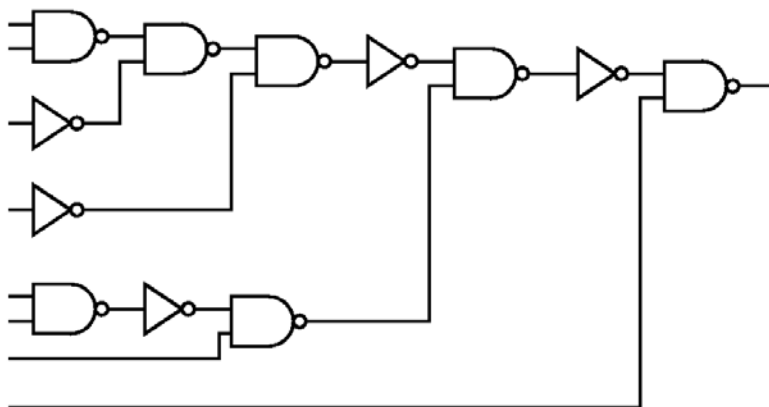
CALTECH CS137 Winter2004 -- DeHon

## Work front-to-back



CALTECH CS137 Winter2004 -- DeHon

## Work Example (area)



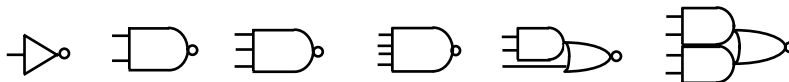
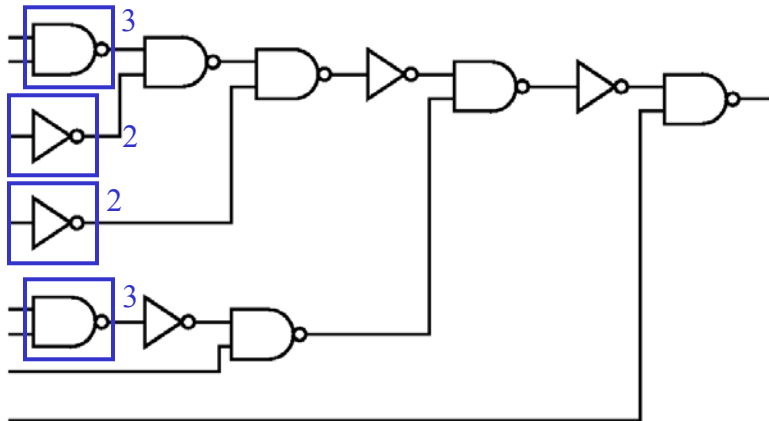
CALTECH CS137 Winter2004 -- DeHon

5

4

5

## Work Example (area)



CALTECH CS137 Winter 2004 -- DeHon

2

3

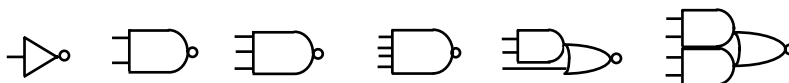
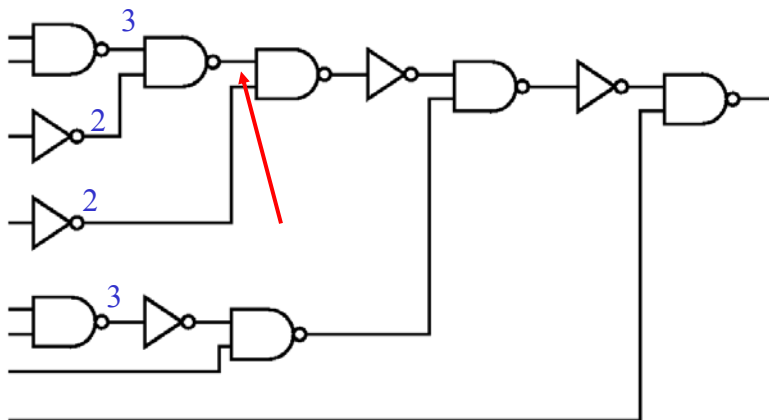
4

5

4

5

## Work Example (area)



CALTECH CS137 Winter 2004 -- DeHon

2

3

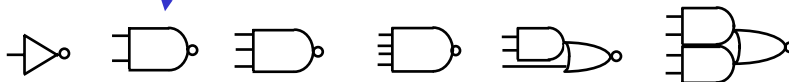
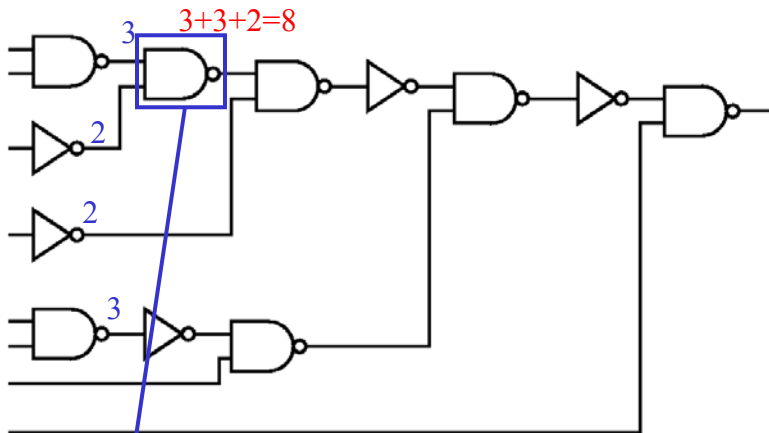
4

5

4

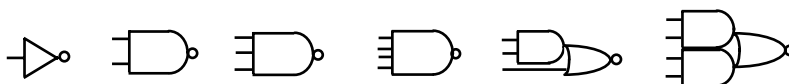
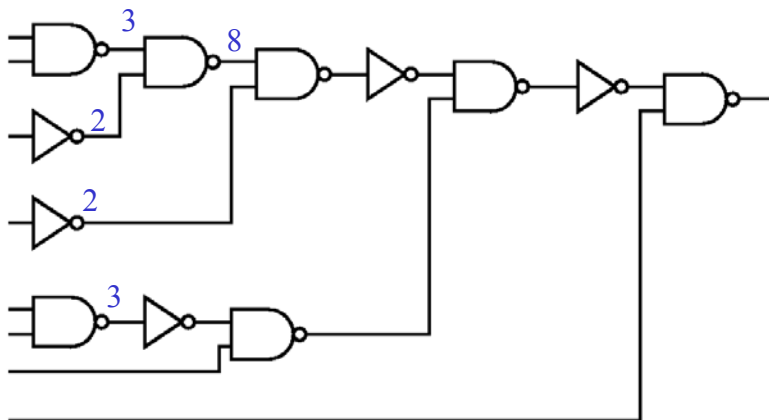
5

## Work Example (area)



CALTECH CS137 Winter 2004 -- DeHon

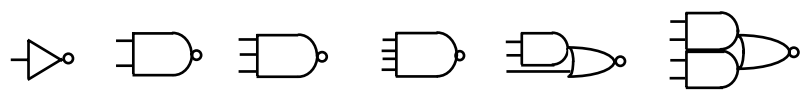
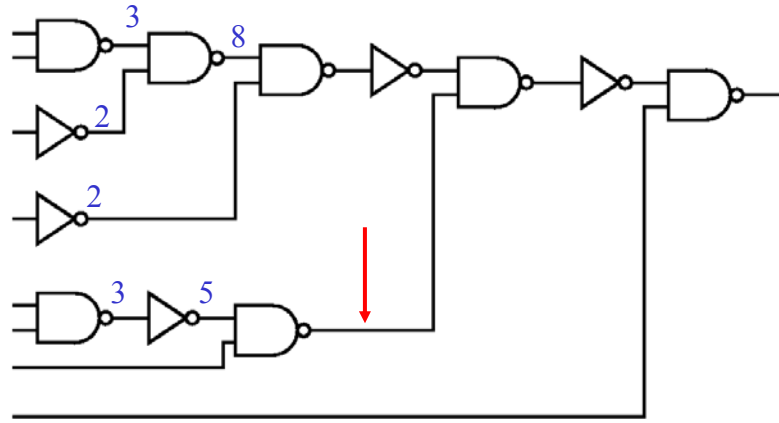
## Work Example (area)



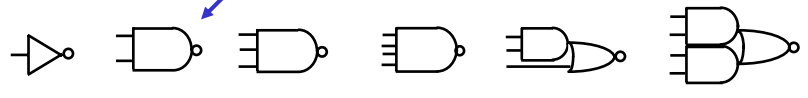
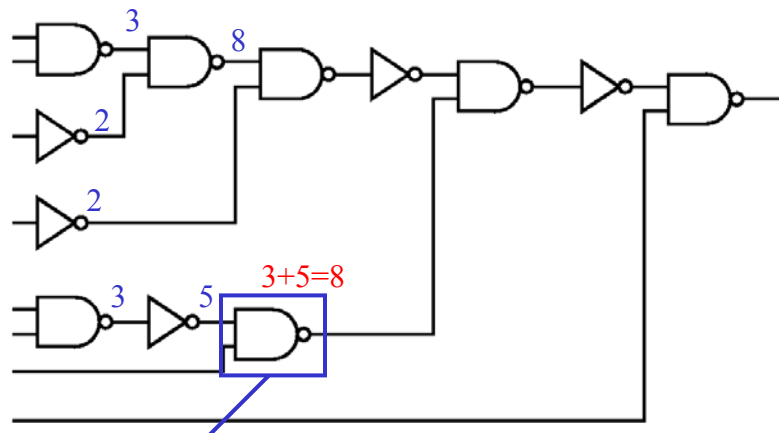
CALTECH CS137 Winter 2004 -- DeHon



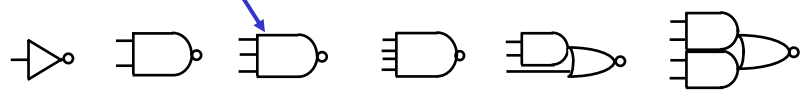
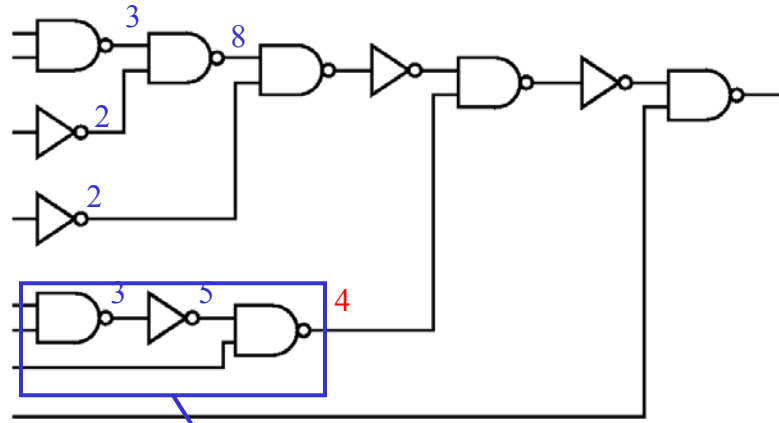
## Work Example (area)



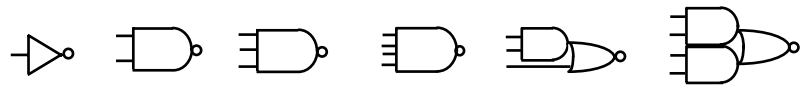
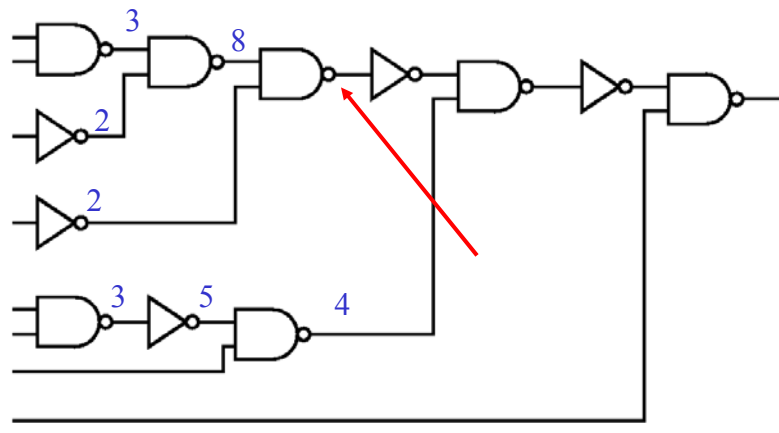
## Work Example (area)



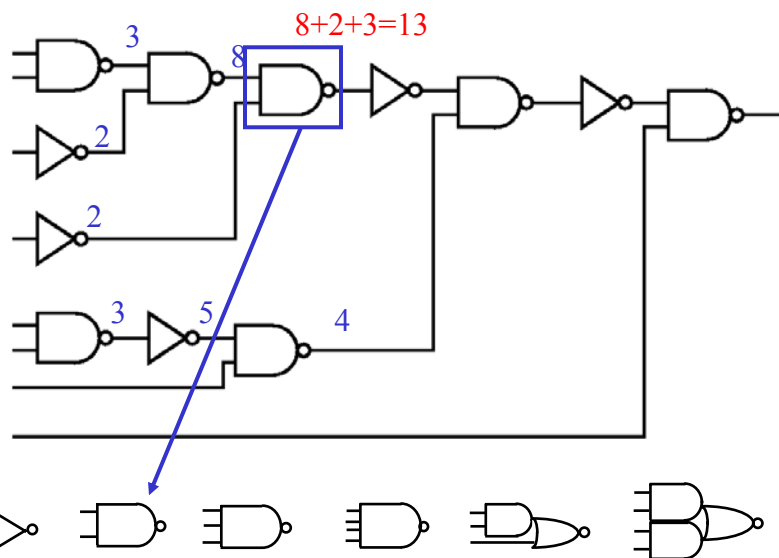
## Work Example (area)



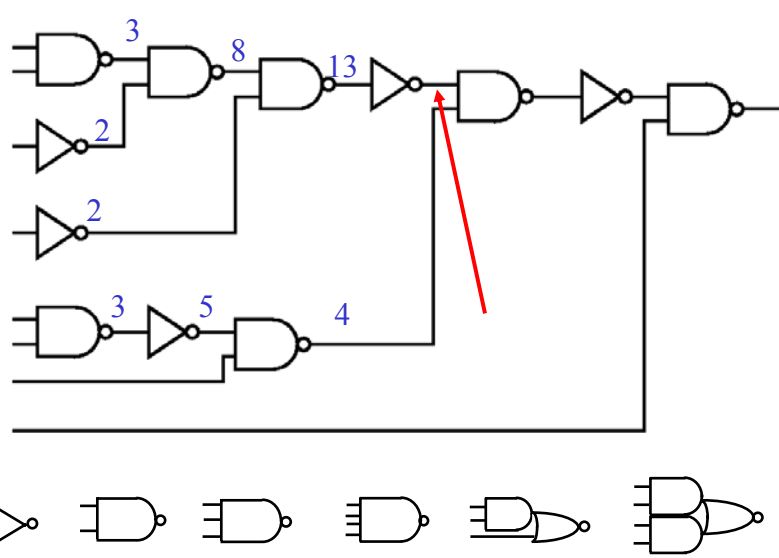
## Work Example (area)



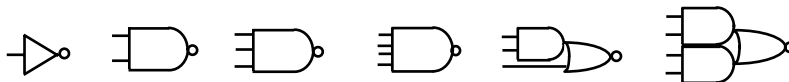
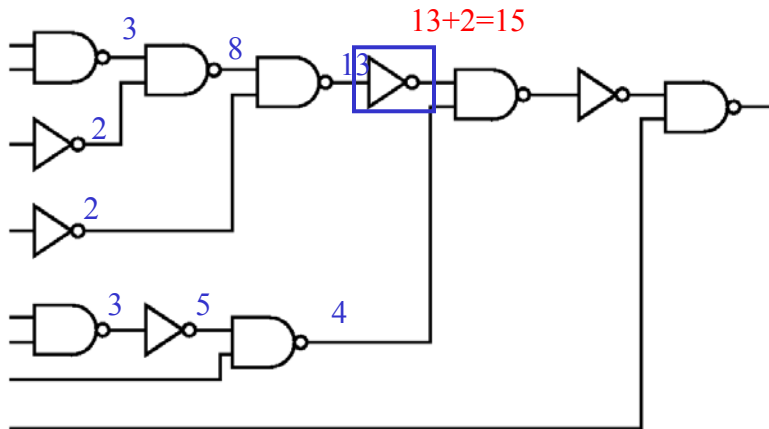
## Work Example (area)



## Work Example (area)

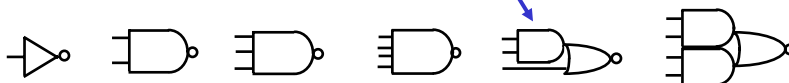
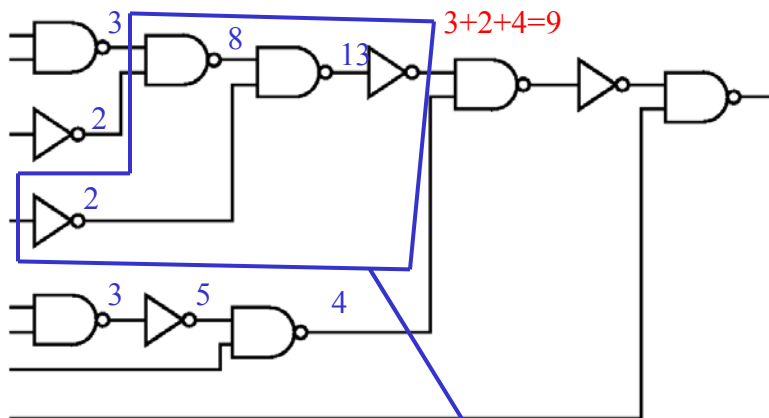


## Work Example (area)



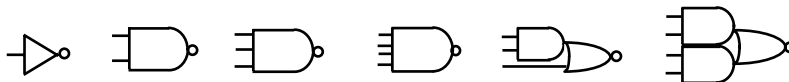
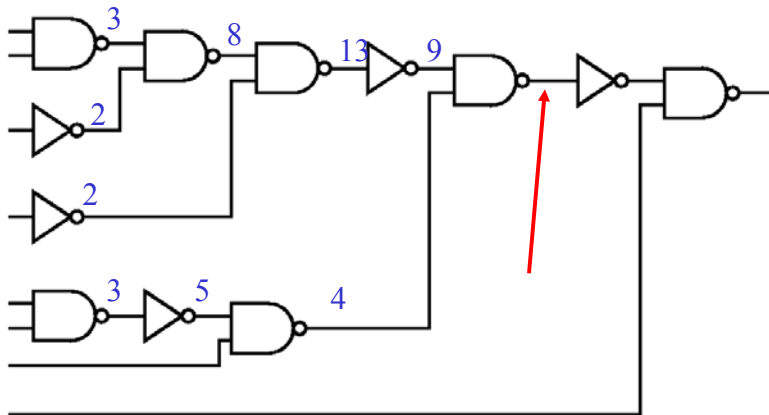
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



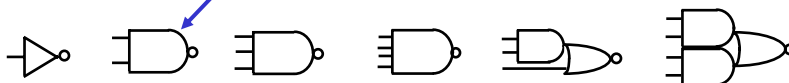
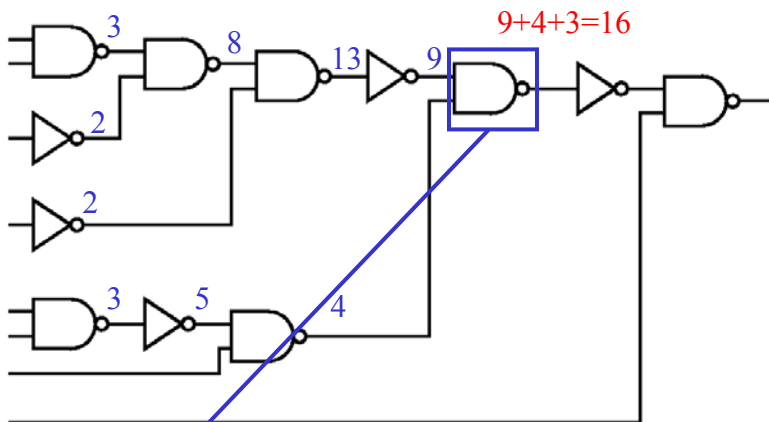
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



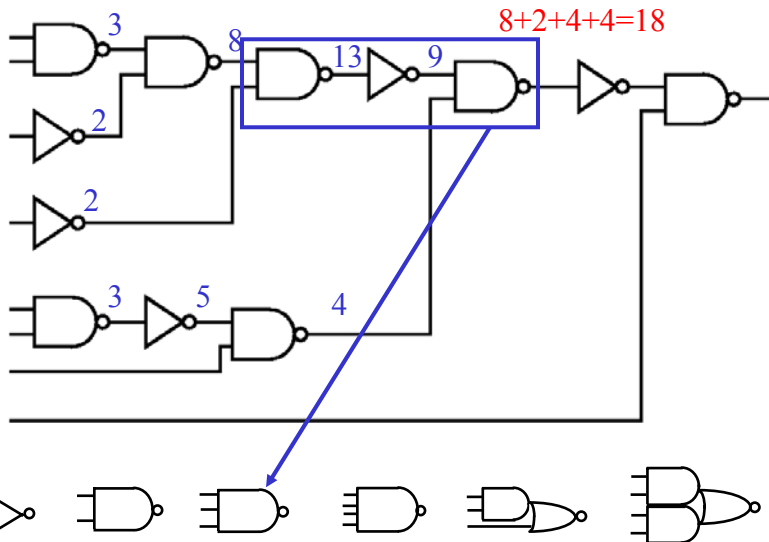
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



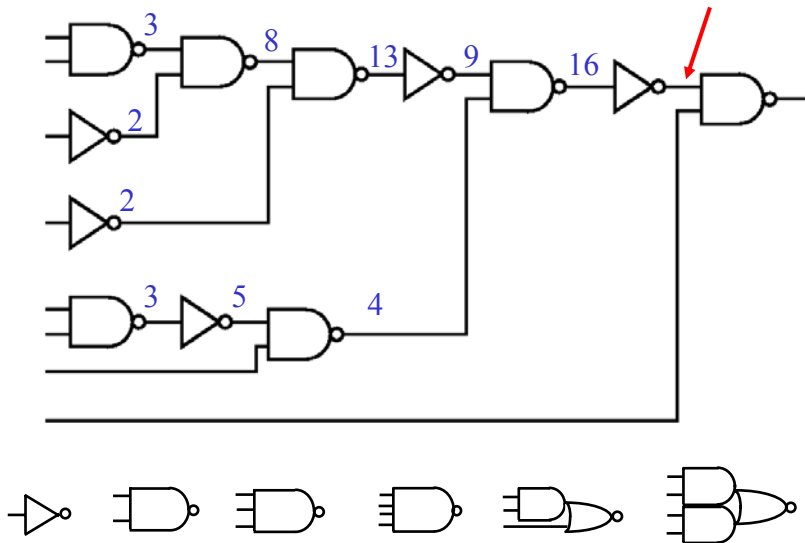
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



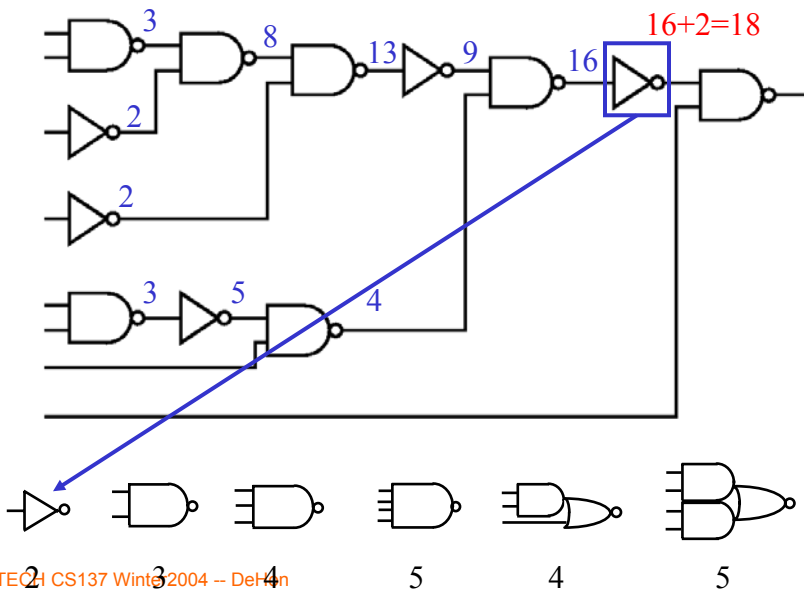
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)

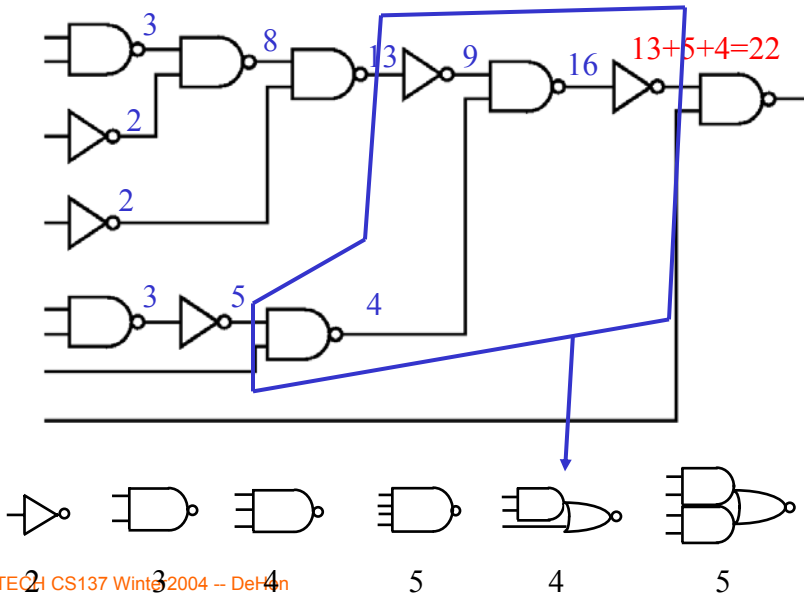


CALTECH CS137 Winter 2004 -- DeHon

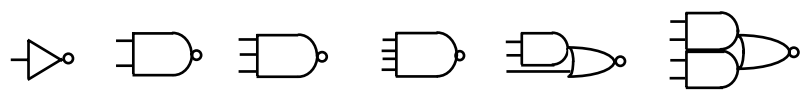
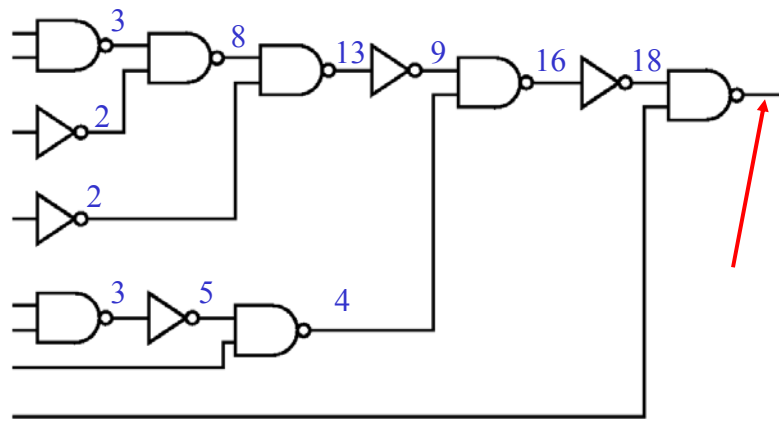
## Work Example (area)



## Work Example (area)

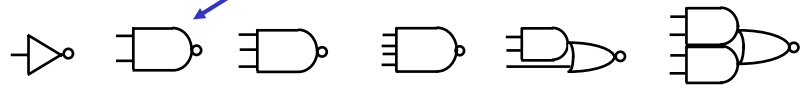
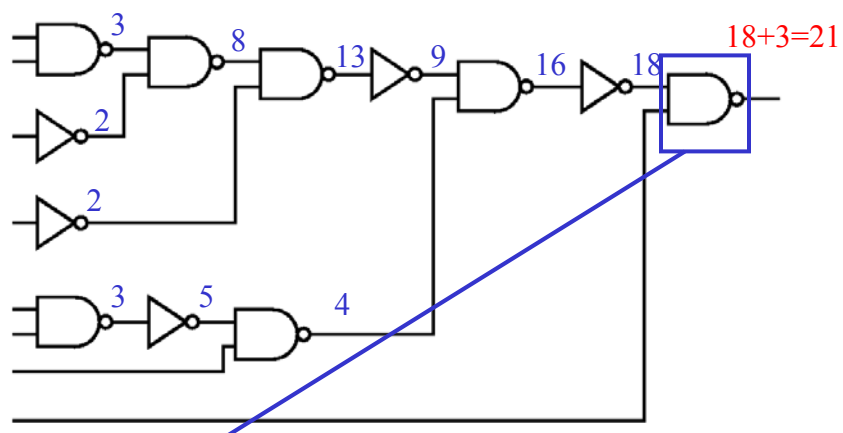


## Work Example (area)



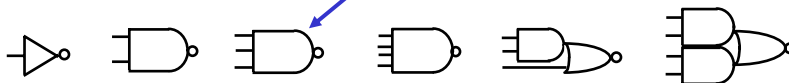
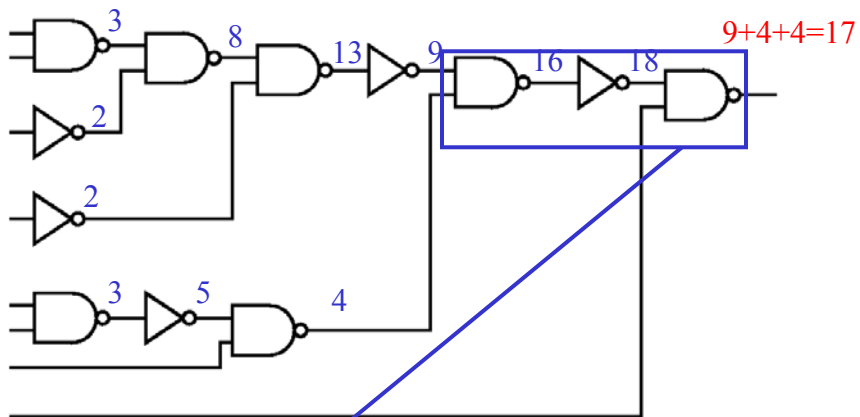
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



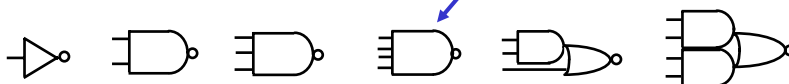
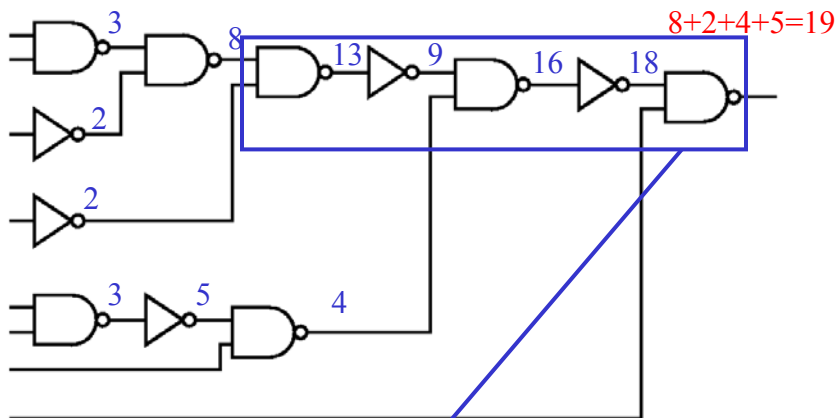
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



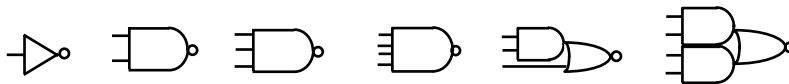
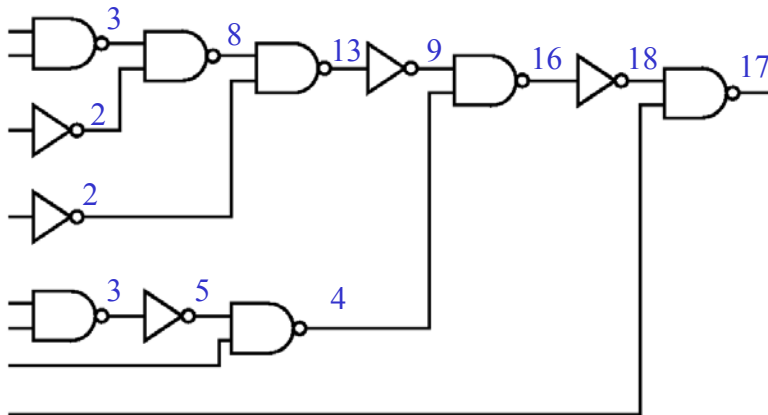
CALTECH CS137 Winter 2004 -- DeHon

## Work Example (area)



CALTECH CS137 Winter 2004 -- DeHon

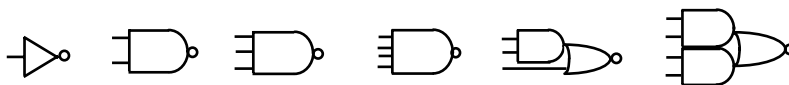
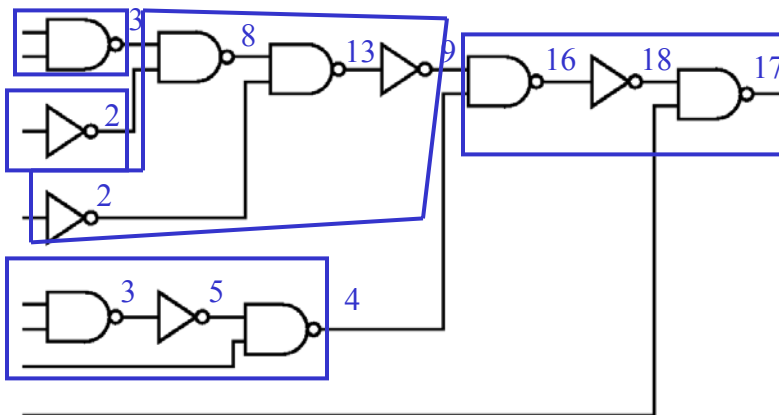
## Work Example (area)



CALTECH CS137 Winter 2004 -- DeHon

2 3 4 5 4 5

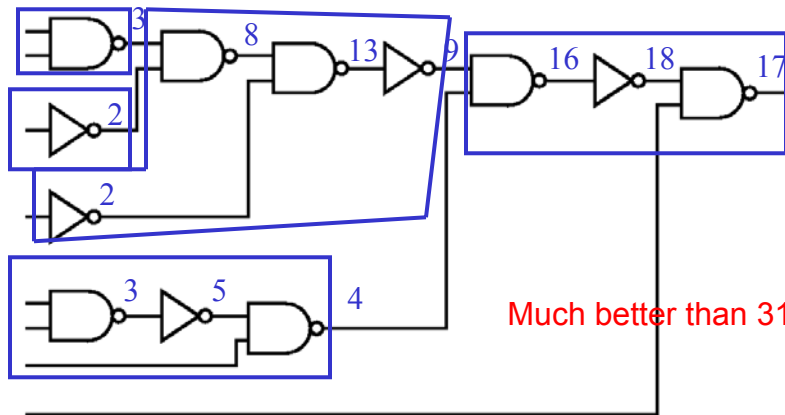
## Optimal Cover



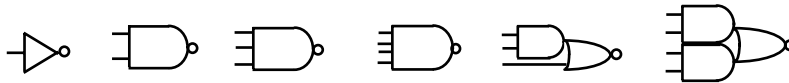
CALTECH CS137 Winter 2004 -- DeHon

2 3 4 5 4 5

## Optimal Cover



Much better than 31!



CALTECH CS137 Winter 2004 -- DeHon

## Note

- There are nodes we cover which will **not** appear in final solution.

CALTECH CS137 Winter 2004 -- DeHon

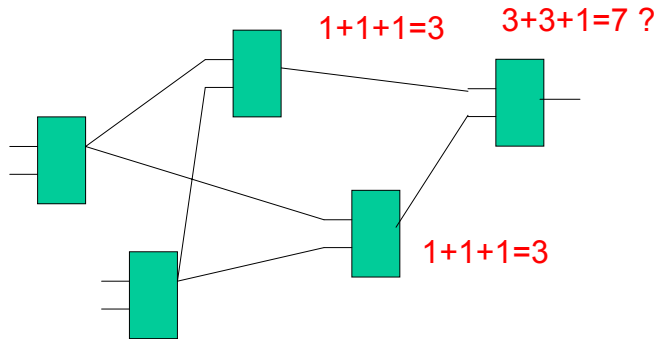






## Not optimal for DAGs

- Why?



CALTECH CS137 Winter2004 -- DeHon

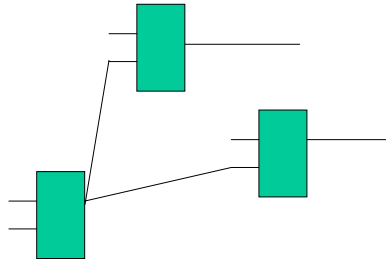
## Not Optimal for DAGs (area)

- $\text{Cost}(N) = \text{Cost}(\text{gate}) + \sum \text{Cost}(\text{input nodes})$
- think of sets
- cost is magnitude of set union
- **Problem:** minimum cost (magnitude) solution isn't necessarily the best pick
  - get interaction between subproblems
  - subproblem optimum not global...

CALTECH CS137 Winter2004 -- DeHon

## Not Optimal for DAGs

- Delay:
  - in fanout model, depends on problem you haven't already solved (delay of node depends on number of uses)



CALTECH CS137 Winter2004 -- DeHon

## What do people do?

- Cut DAGs at fanout nodes
- optimally solve resulting trees
- Area
  - guarantees covered once
    - get accurate costs in covering trees, made “premature” assignment of nodes to trees
- Delay
  - know where fanout is

CALTECH CS137 Winter2004 -- DeHon

# Bounding

- Tree solution give bounds (esp. for delay)
  - single path, optimal covering for delay
  - (also make tree by replicating nodes at fanout points)
- no fanout cost give bounds
  - know you can't do better
- delay bounds useful, too
  - know what you're giving up for area
  - when delay matters

CALTECH CS137 Winter2004 -- DeHon

## (Multiple Objectives?)

- Like to say, get delay, then area
  - won't get minimum area for that delay
  - algorithm only keep best delay
  - ...but best delay on off critical path piece not matter
    - ...could have accepted more delay there
  - don't know if on critical path while building subtree
  - (iterate, keep multiple solutions)

CALTECH CS137 Winter2004 -- DeHon

## Many more details...

- Implement well
- Combine criteria
  - (touch on some later)
- ...see literature
  - (put some refs on web)

CALTECH CS137 Winter2004 -- DeHon

## Big Ideas

- simple cost models
- problem formulation
- identifying structure in the problem
- special structure
- characteristics that make problems hard
- bounding solutions

CALTECH CS137 Winter2004 -- DeHon