EEL 4837
Programming for Electrical Engineers II

Qiangeng Yang
Teaching Assistant
Department of Electrical and Computer Engineering
University of Florida at Gainesville

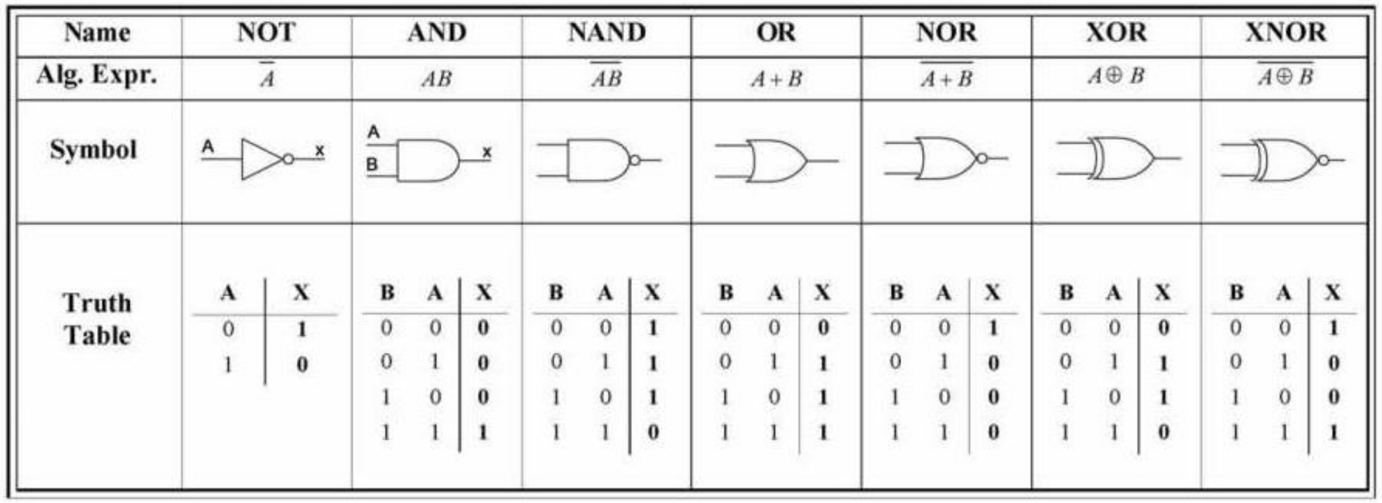
Excursion 2 – Technology Mapping

Reading:

- Excursion 2 Description

Logic Gate

- Logic gate
  - A device that acts as a building block for digital circuits.
  - One or more inputs (0/1) and only one output (0/1).
  - Performing a specific logic function.
- Some basic logic gates:



Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$
Symbol	A, x	A, B, x					

Truth Table

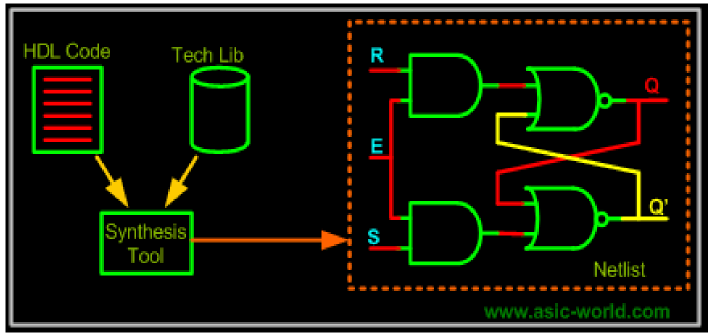
NOT:

A	X
0	1
1	0

B	A	X (AND)	X (NAND)	X (OR)	X (NOR)	X (XOR)	X (XNOR)
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Logic Synthesis

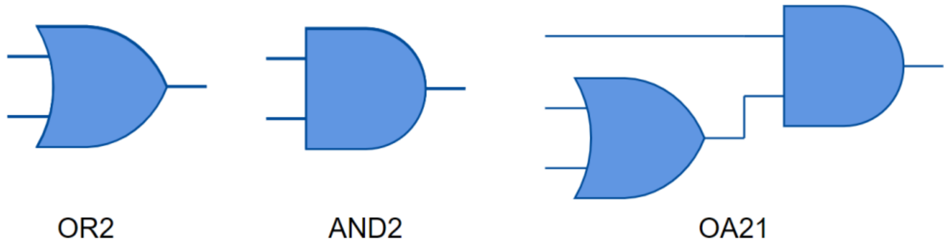
- **Logic synthesis**
    - The process of automatic production of logic components, in particular digital circuits.
- It includes:
    - **Logic optimization**
        - Simplifies logic expressions theoretically.
    - **Technology mapping**
        - Implements logic gates with physical layouts.
        - **The topic of Excursion 2**



Source: http://www.asic-world.com/verilog/synthesis1.html

Technology Mapping

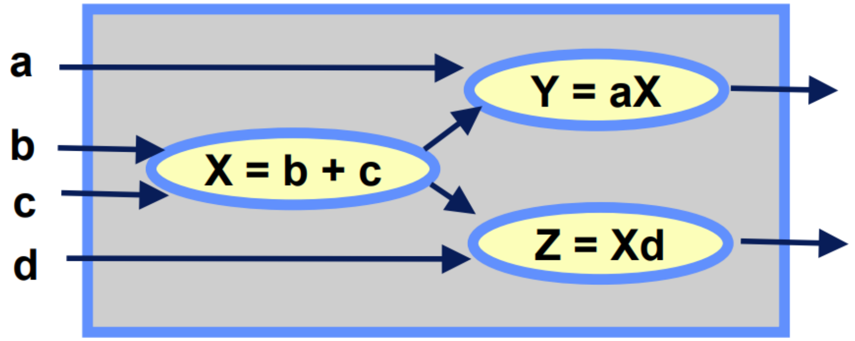
- Suppose we have these gates in our "library":



OA21: OR-AND, a so-called complex gate.

Number like *2* and *21*: the count of direct inputs.

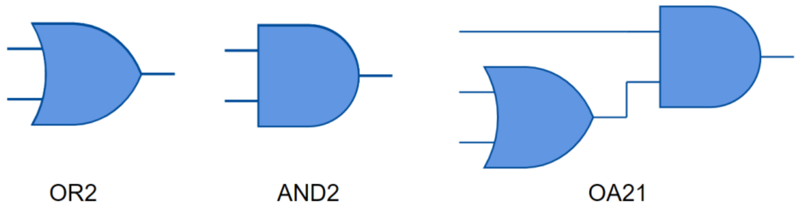
- Suppose we need to build a model as follows:



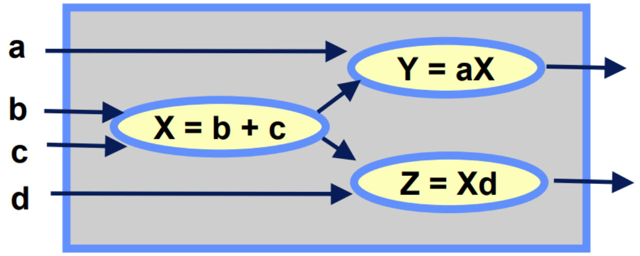
Q: How can we realize it by using the gates in our library?

Technology Mapping

- Library:

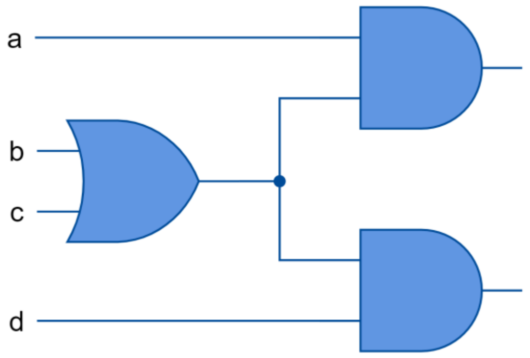


Model:

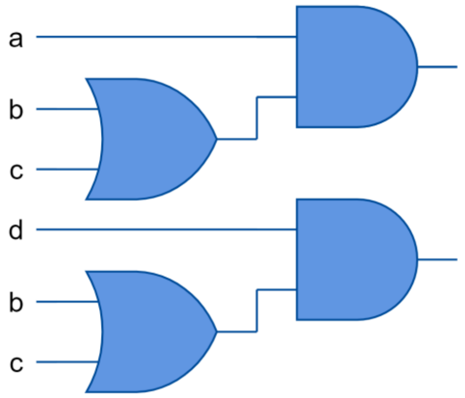


- Solutions:

Solution 1: suppose we have all what we need :)



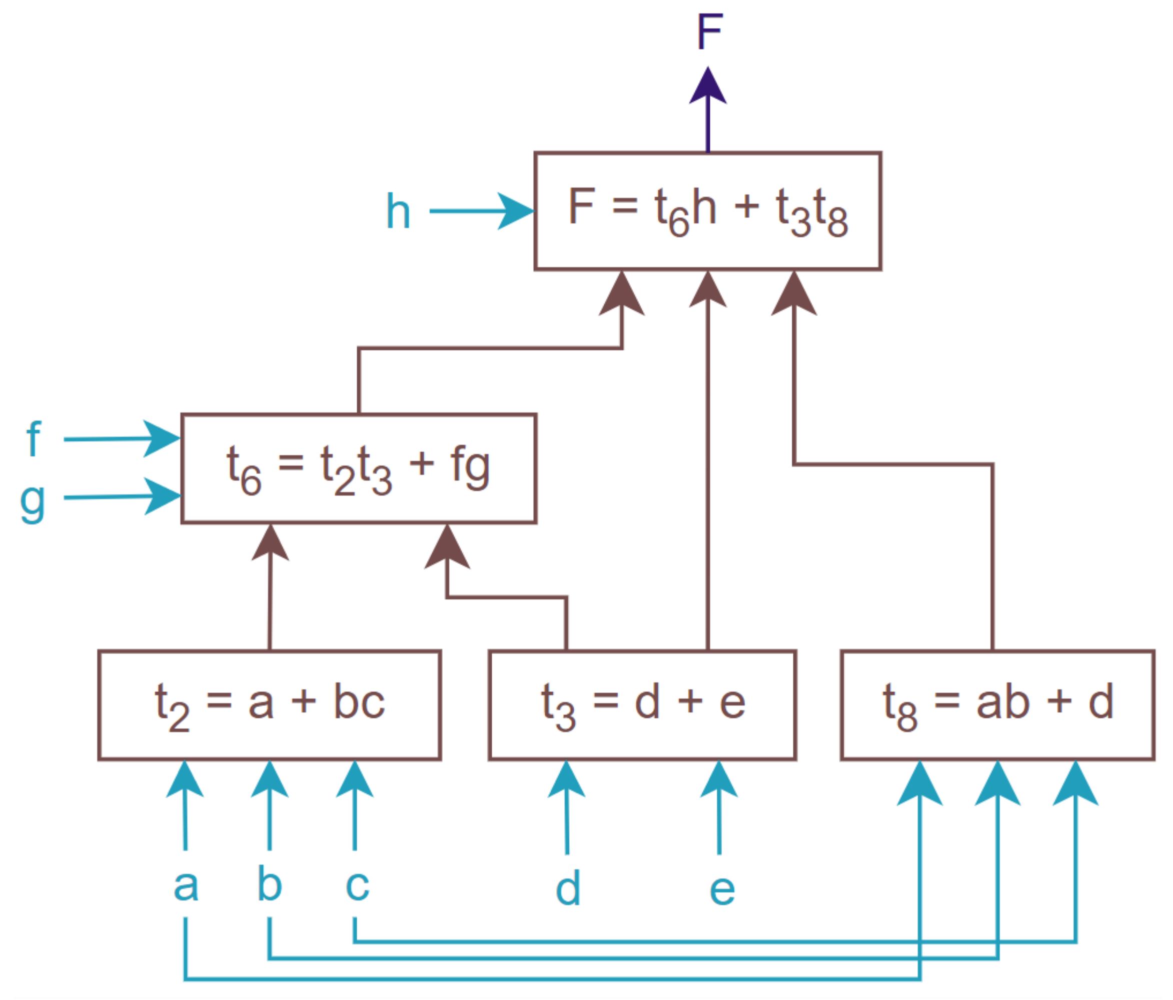
Solution 2: suppose we only have OA21 :(



Excursion 2

- Goal:
  - Implement a tool for technology mapping.
  - Convert the original logic expressions into NAND and NOT gates.
  - Compute the minimal cost.
- General steps:
  1. Read a Boolean netlist.
  2. Convert the original structure into a NAND-NOT tree.
  3. Recursively compute the minimal cost.

Step 1: Read a Boolean netlist



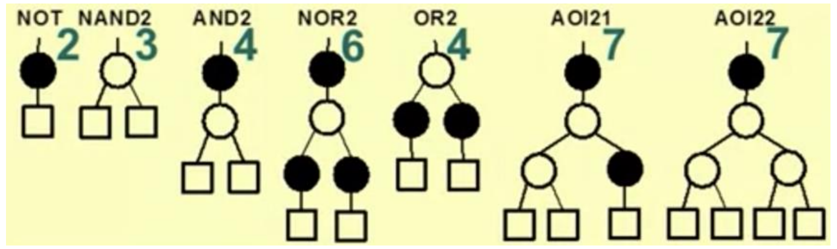
Note: some middle steps were combined only for elegance.

Netlist format:

```
a  INPUT                t8 = OR t7 d
b  INPUT                t9 = AND t3 t8
c  INPUT                t10 = AND t6 h
d  INPUT                F = OR t9 t10
e  INPUT
f  INPUT
g  INPUT
h  INPUT
F  OUTPUT
t1 = AND b c
t2 = OR a t1
t3 = OR d e
t4 = AND f g
t5 = AND t2 t3
t6 = OR t4 t5
t7 = AND a b
```

Step 2: Conversion into a NAND-NOT tree

- Only NAND and NOT gates are available.
- Represent a gate of AND/OR/NOT by NANDs and NOTs only.
  - Transforming equations (De Morgan's laws):
    - **NOT A = NOT A**
    - **A AND B = NOT (A NAND B)**
    - **A OR B = (NOT A) NAND (NOT B)**

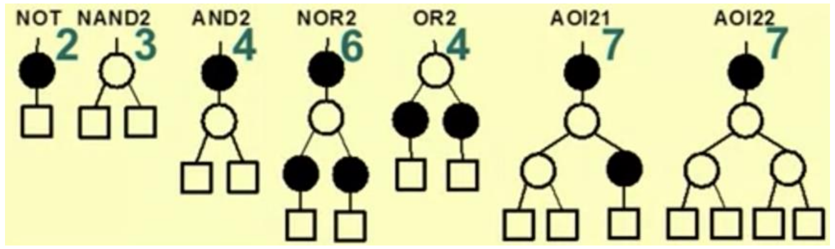


Black Circle = NOT
White Circle = NAND
White Square = input

- Best practice: topologically traverse from output to input.

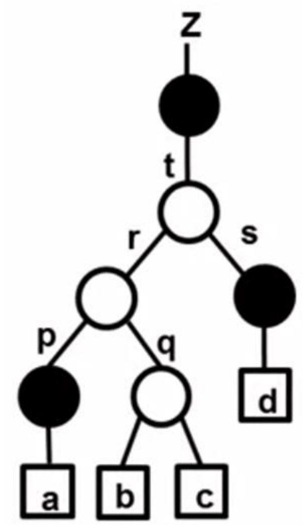
Step 3: Recursively compute minimal cost

- Library:

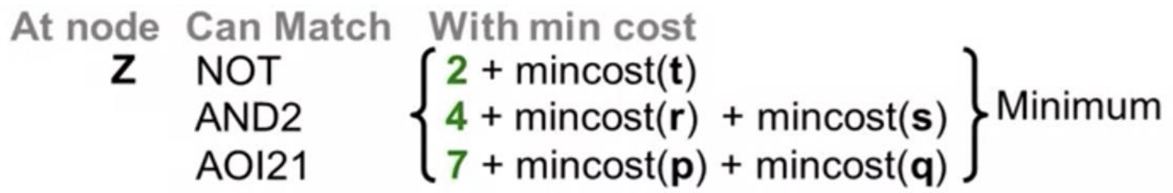


Black Circle = NOT
White Circle = NAND
White Square = input
Number: cost

- Consider a tree like this:



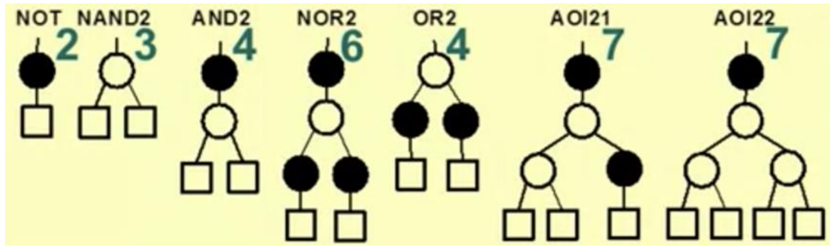
How can we describe the tree?
What components provided in the library were used?



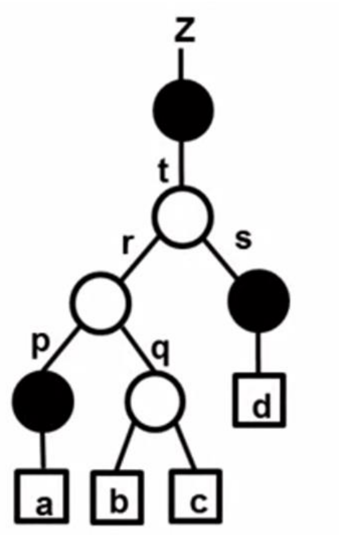
At node	Can Match	With min cost	
z	NOT	2 + mincost(**t**)	Minimum
	AND2	4 + mincost(**r**) + mincost(**s**)	
	AOI21	7 + mincost(**p**) + mincost(**q**)	

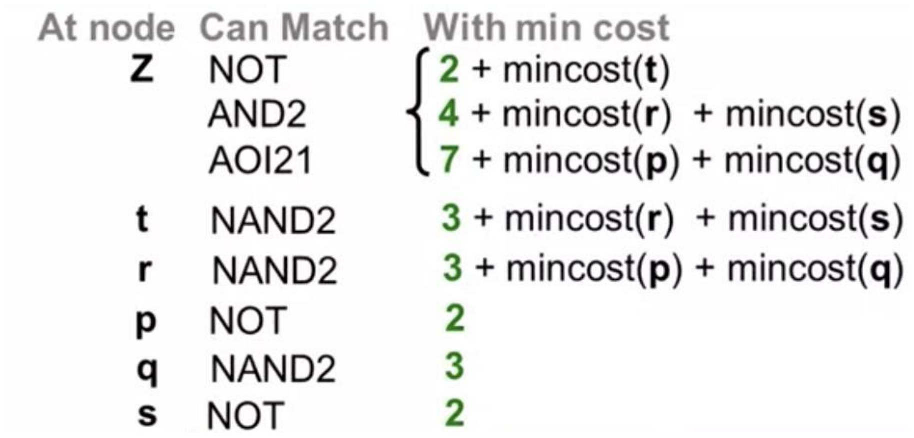
Step 3: Recursively compute minimal cost

- Continue the recursion to compute the minimal cost at the other nodes.



Black Circle = NOT
White Circle = NAND
White Square = input
Number: cost

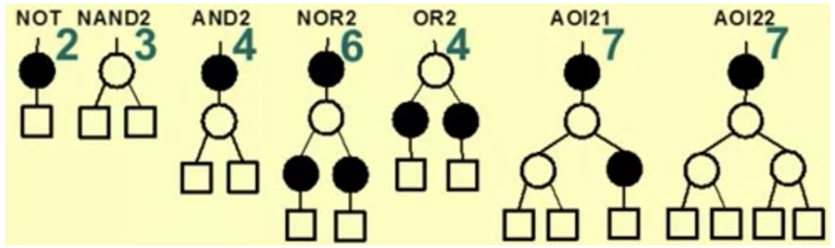




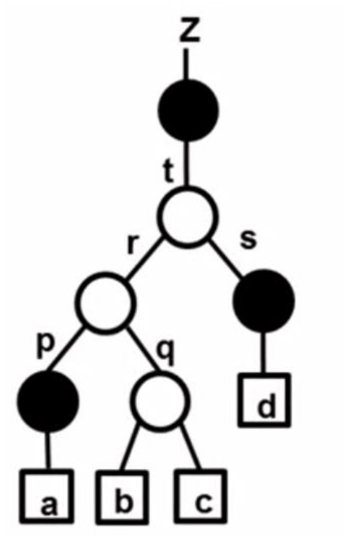
At node	Can Match	With min cost
Z	NOT	2 + mincost(**t**)
	AND2	4 + mincost(**r**) + mincost(**s**)
	AOI21	7 + mincost(**p**) + mincost(**q**)
t	NAND2	3 + mincost(**r**) + mincost(**s**)
r	NAND2	3 + mincost(**p**) + mincost(**q**)
p	NOT	2
q	NAND2	3
s	NOT	2

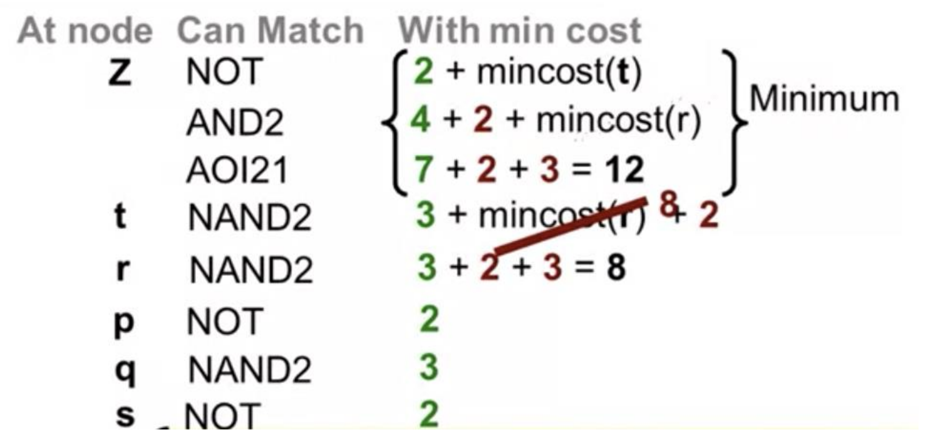
Step 3: Recursively compute minimal cost

- Continue the recursion to compute the minimal cost at the other nodes.



Black Circle = NOT
White Circle = NAND
White Square = input
Number: cost

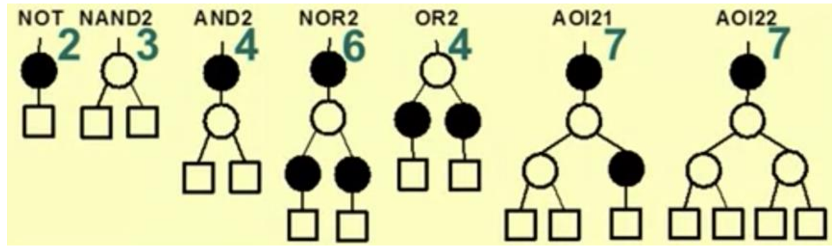




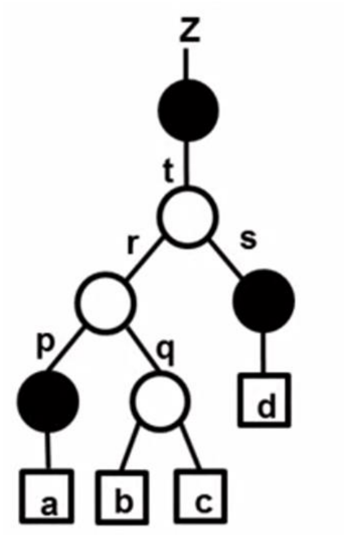
At node	Can Match	With min cost	
z	NOT	2 + mincost(**t**)	Minimum
	AND2	4 + 2 + mincost(r)	
	AOI21	7 + 2 + 3 = **12**	
t	NAND2	3 + ~~mincost(r)~~ 8 + 2	
r	NAND2	3 + 2 + 3 = **8**	
p	NOT	2	
q	NAND2	3	
s	NOT	2	

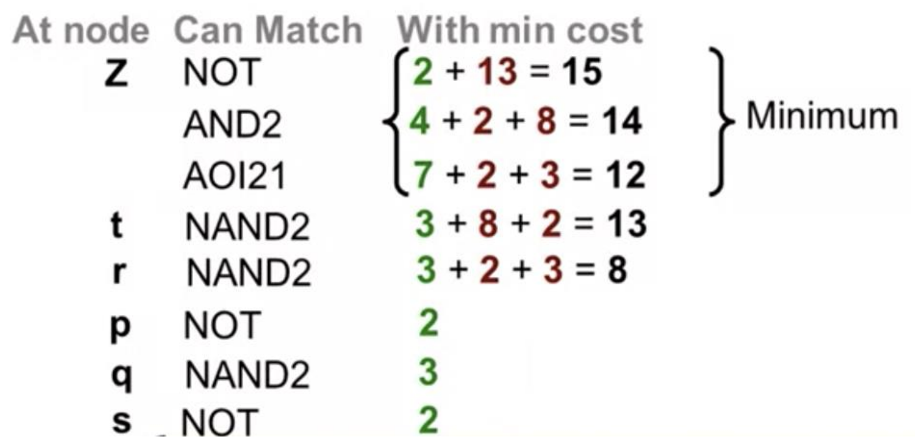
Step 3: Recursively compute minimal cost

- Continue the recursion to compute the minimal cost at the other nodes.



Black Circle = NOT
White Circle = NAND
White Square = input
Number: cost

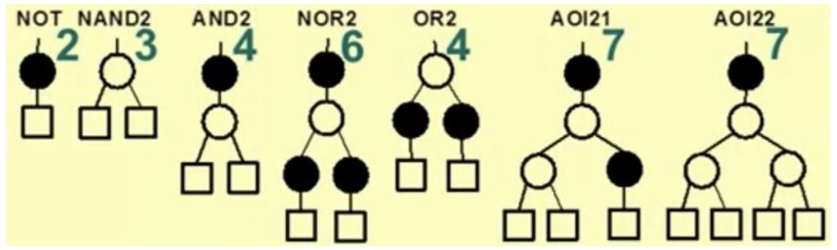




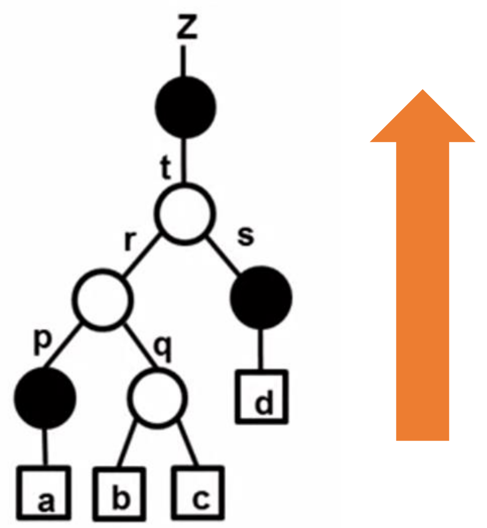
At node	Can Match	With min cost	
z	NOT	2 + 13 = 15	Minimum
	AND2	4 + 2 + 8 = 14	
	AOI21	7 + 2 + 3 = 12	
t	NAND2	3 + 8 + 2 = 13	
r	NAND2	3 + 2 + 3 = 8	
p	NOT	2	
q	NAND2	3	
s	NOT	2	

Step 3: Recursively compute minimal cost

- (Optional, the other ways, **bonus points**)
  - Bottom-up
  - Dynamic programming

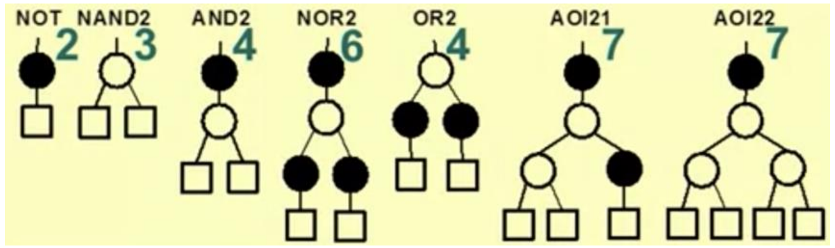


Black Circle = NOT
White Circle = NAND
White Square = input
Number: cost

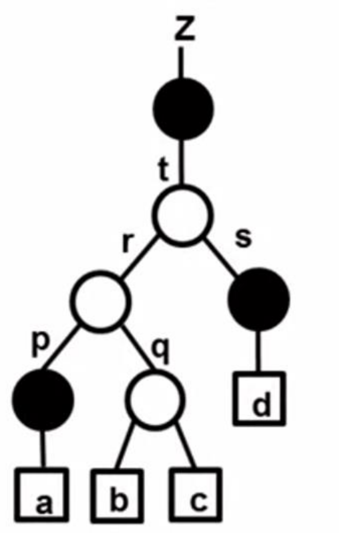


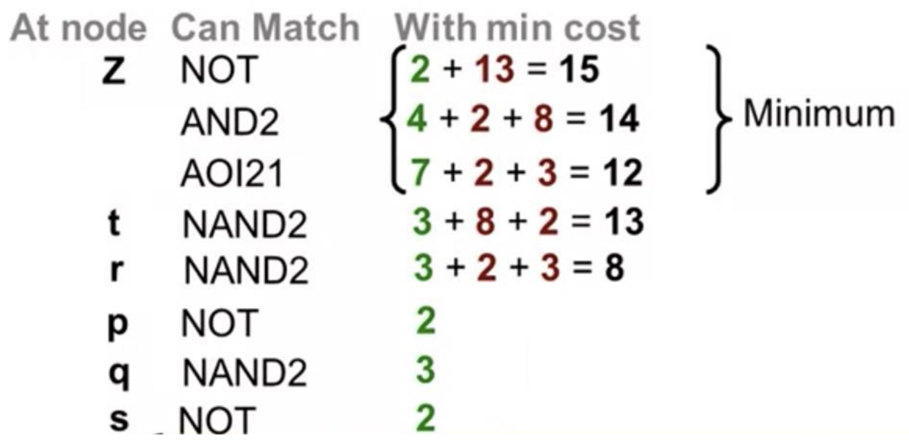
Step 3: Recursively compute minimal cost

- Return the calculated optimal cost.



Black Circle = NOT
White Circle = NAND
White Square = input
Number: cost





At node	Can Match	With min cost
Z	NOT	2 + 13 = 15
	AND2	4 + 2 + 8 = 14
	AOI21	7 + 2 + 3 = 12
t	NAND2	3 + 8 + 2 = 13
r	NAND2	3 + 2 + 3 = 8
p	NOT	2
q	NAND2	3
s	NOT	2

Minimum

Min cost: 12

Quick Review

- Goal:
  - Implement a tool for technology mapping.
  - Convert the original expressions into NAND and NOT gates.
  - Compute the minimal cost.
- General steps:
  1. Read a Boolean netlist.
  2. Convert the original structure into a NAND-NOT tree.
  3. Recursively compute the minimal cost.

Thank you!

Reading:

- Excursion 2 Description