Practice Problem

- Boolean Function: $F = \overline{X}YZ + XZ$

Truth Table:

X	Y	Z	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Simplification:

$=$

Why Is This Useful?

- Logic minimization: reduce complexity at gate level
  - Allows us to build smaller and faster hardware
  - Care about both # of gates, # of literals (gate inputs), # of gate levels, and types of logic gates

DeMorgan's Law

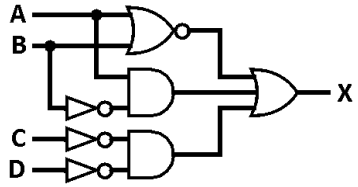
- $\overline{X + Y} = \overline{X} \cdot \overline{Y}$
- $\overline{X \cdot Y} = \overline{X} + \overline{Y}$

				NOR		NAND	
X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X + Y}$	$\overline{X} \cdot \overline{Y}$	$\overline{X \cdot Y}$	$\overline{X} + \overline{Y}$
0	0	1	1	1		1	
0	1	1	0	0		1	
1	0	0	1	0		1	
1	1	0	0	0		0	

- In Boolean Algebra, converts between AND-OR and OR-AND expressions
  - $Z = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C$
  - $\overline{Z} = (A + B + \overline{C}) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + B + \overline{C})$
- At gate level, can convert from AND/OR to NAND/NOR gates
  - "Flip" all input/output bubbles and "switch" gate

DeMorgan's Law Practice Problem

- Simplify the following diagram:



- Then implement with only NAND gates: