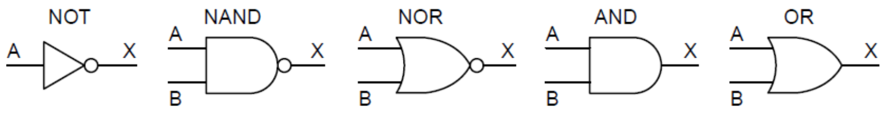
CS 210: Principles of Computer Organization

Gates and Boolean Algebra

Gate symbols (left) and corresponding truth symbols (below)



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Function notation

- **NOT** function: $f = \overline{A}$
- **OR** function: $f = A + B$
- **AND** function: $f = AB$
- **NOR** function: $f = \overline{A + B}$
- **NAND** function: $f = \overline{AB}$

Example:

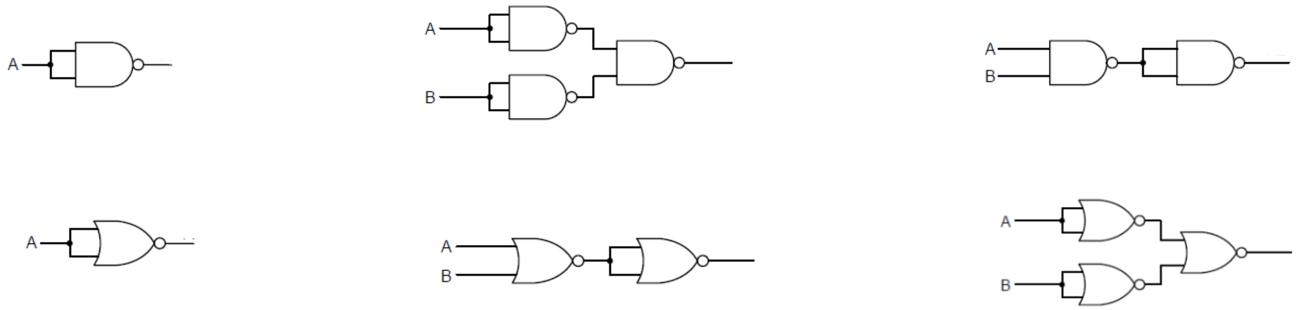
1. Implement a circuit for Boolean function *f* which takes three inputs, *A, B, C,* and outputs a 1 if and only if no more than one of the inputs is a 1.

Functional completeness

A **functionally complete** set of gates is one which can be used to express all possible truth tables by combining members of the set into Boolean expression.

- {NOT, AND, OR} is functionally complete.
- {NAND} is functionally complete.
- {NOR} is functionally complete.

1. Label the circuits below with their equivalence to NOT, AND, and OR gates.



<u>Circuit Equivalence:</u>

Two circuits/functions are **equivalent** if and only if they have the same output for every possible input.

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

1. Build a circuit for XOR. Can you build it in more than one way?

2. Use a truth table to show that X = (X AND Y) OR (X AND NOT Y)