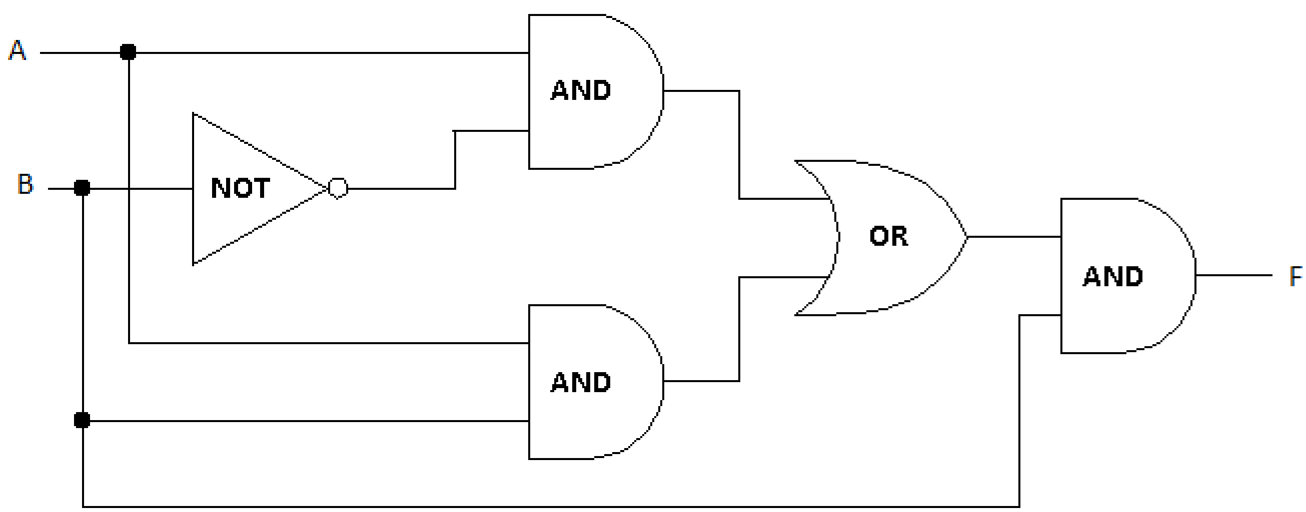
1. More logical equivalence examples

Laws of Logical Equivalence		
Name	**OR version**	**AND version**
Commutative	A + B = B + A	A * B = B * A
Associative	(A + B) + C = A + (B + C)	(A * B) * C = A * (B * C)
Distributive	A + (B * C) = (A + B) * (A + C)	A * (B + C) = (A * B) + (A * C)
Idempotent	A + A = A	A * A = A
Identity	A + 0 = A	A * 1 = A
	A + 1 = 1	A * 0 = 0
Complement	A +~A = 1	A * ~A = 0
	~1 = 0	~0 = 1
Double Negative	~(~A) = A	
De Morgan's	~(A + B) = ~A * ~B	~(A * B) = ~A + ~B
Absorption	A + (A * B) = A	A * (A + B) = A

a. Prove the OR version of the Absorption Law, A + A * B = A.

Assertion	Reason
A + A * B	Initial function
= (A * 1) + (A * B)	Identity Law for AND
= A * (1 + B)	Distributive Law for AND
= A * (1)	Identity Law for OR
= A	Identity Law for AND

b. Simplify the following digital logic circuit using propositional algebra.



Assertion	Reason
f = ((A * ~B) + (A * B)) * B	Initial circuit logic
= (A * (~B + B)) * B	Distributive Law for AND
= (A * 1) * B	Complement Law for OR
= A * B	Identity Law for AND

2. More on gates
    a. Functionally complete sets – a set of gates that can implement any Boolean function
    b. Universal gate – a single gate that is a functionally complete set on its own
        i. Less gates, easier fabrication
        ii. AND, OR, NOT
            1. XOR takes five gates: $A \oplus B = A * {\sim}B + {\sim}A * B$
        iii. AND, NOT
            1. OR requires four gates to implement DeMorgan's law
            2. $A + B = {\sim}({\sim}A * {\sim}B)$
        iv. OR, NOT
            1. AND requires four gates to implement DeMorgan's law
            2. $A * B = {\sim}({\sim}A + {\sim}B)$
        v. NAND - $\uparrow$ is the NAND operator
            1. ${\sim}A = A \uparrow A$
            2. $A + B = (A \uparrow A) \uparrow (B \uparrow B)$
            3. $A * B = (A \uparrow B) \uparrow (A \uparrow B)$, this requires only 2 NANDs because $(A \uparrow B)$ is used twice
            4. $A \oplus B = (A \uparrow (A \uparrow B)) \uparrow (B \uparrow (A \uparrow B))$, just 4 NANDs needed because $(A \uparrow B)$ is used twice
        vi. NOR - $\downarrow$ is the NOR operator.
            1. ${\sim}A = A \downarrow A$
            2. $A * B = (A \downarrow A) \downarrow (B \downarrow B)$,
            3. $A \oplus B = ((A \downarrow A) \downarrow (B \downarrow B)) \downarrow (A \downarrow B)$
            4. $A + B = (A \downarrow B) \downarrow (A \downarrow B)$, just 2 NOR gates needed
    c. Implement using transistors

3. Truth tables
    d. Boolean function with *n* variables has 2^n rows
    e. Entire set resembles counting upwards in binary, e.g. 000, 001, 010, 011… with 3 variables
    f. Minterms
        i. Product term in which each of the *n* variables appears once (in either complemented or uncomplemented term)
        ii. Minterm results in a 1 for output of a single cell expression, 0s for all other rows in truth table
        iii. Boolean function can be represented by sum of all minterms for which function is true
            1. Sum-of-products form (SOP)
    g. Maxterms
        i. Like minterms, variables can only appear once in complemented or uncomplemented form
        ii. Maxterm results in a 0 for the output of a single cell expression, 1s for all other rows in truth table
        iii. The complement of the corresponding row's minterm
        iv. Boolean function can be represented as product of all maxterms for which function is false
            1. Product-of-sums form (POS)
    h. SOP easier to work with, more natural, but sometimes POS can lead to simpler logic
    i. Term indices correspond to binary concatenation of row variable's truth values
        i. Minterms represented with lower case m, e.g. m_2 for inputs 010
        ii. Maxterms represented with upper case M, e.g. M_5 for inputs 101
    j. Example: 3-variable Boolean function true when A is true, B is true, and C is false
        i. Minterm $m_6 = AB\overline{C}$ (110), maxterm $M_6 = \overline{A} + \overline{B} + C$
        ii. $m_0 = \overline{ABC}$ (000), and $m_7 = ABC$ (111)

4. Synthesizing using gates
    k. Consider a Boolean function of three variables
        i. True when either, but not both, of the first two variables is true
        ii. Truth table below

Index	**A**	**B**	**C**	**f(A, B, C)**	**Minterm**	**Maxterm**
0	0	0	0	0	$m_0 = \overline{ABC}$	$M_0 = A + B + C$
1	0	0	1	0	$m_1 = \overline{AB}C$	$M_1 = A + B + \overline{C}$
2	0	1	0	1	$m_2 = \overline{A}B\overline{C}$	$M_2 = A + \overline{B} + C$
3	0	1	1	1	$m_3 = \overline{A}BC$	$M_3 = A + \overline{B} + \overline{C}$
4	1	0	0	1	$m_4 = A\overline{BC}$	$M_4 = \overline{A} + B + C$
5	1	0	1	1	$m_5 = A\overline{B}C$	$M_5 = \overline{A} + B + \overline{C}$
6	1	1	0	0	$m_6 = AB\overline{C}$	$M_6 = \overline{A} + \overline{B} + C$
7	1	1	1	0	$m_7 = ABC$	$M_7 = \overline{A} + \overline{B} + \overline{C}$

    l. Sum-of-products
        i. f = $\overline{A}B\overline{C}$ + $\overline{A}BC$ + $A\overline{BC}$ + $A\overline{B}C$ = $m_2 + m_3 + m_4 + m_5$