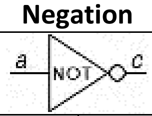
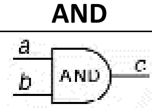
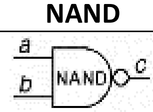
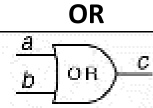
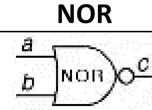
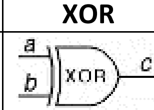
1. Analog vs. digital
    a. Analog – represent values by a continuously variable physical quantity
        i. Here, voltage
        ii. Key word is *continuous*
        iii. Suited to amplification of real world phenomena (sound)
        iv. Suited to calculating continuous function values (integrals)
        v. Subject to noise, difficult to debug
    b. Digital – use discrete (discontinuous) values to represent data
        i. Suited to discrete mathematics (like accounting)
        ii. Needs to sample continuous data
            1. Will miss data that fluctuates faster than sampling rate
        iii. Fixed 0 and 1, low and high, false and true
        iv. Far more resistant to noise, easier to debug

2. Boolean algebra
    a. Algebra of truth values 0 and 1, along with conjunction (AND), disjunction (OR), and negation
        i. George Boole, 1854
        ii. Claude Shannon, 1938, uses it to solve circuit design problems
    b. * = AND, + = OR, ~ = negation, ⊕ = XOR, variables A, B…
    c. Duality principle – swap all signs (+, *, 0, 1) and the underlying logic is still the same
    d. Operator precedence: NOT, AND, OR
    e. Logic types
        i. Combinational – output based solely on current input
        ii. Sequential logic – output based on input and previous stored values (memory)

| **Truth Tables for Digital Design Gates** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Operation:** | | **Negation** | | **AND** | **NAND** | **OR** | **NOR** | **XOR** |
| **Gates:** | |  | |  |  |  |  |  |
| A | B | ~A | ~B | A * B | ~(A * B) | A + B | ~(A + B) | A ⊕ B |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

3. Logical equivalence

| 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 |

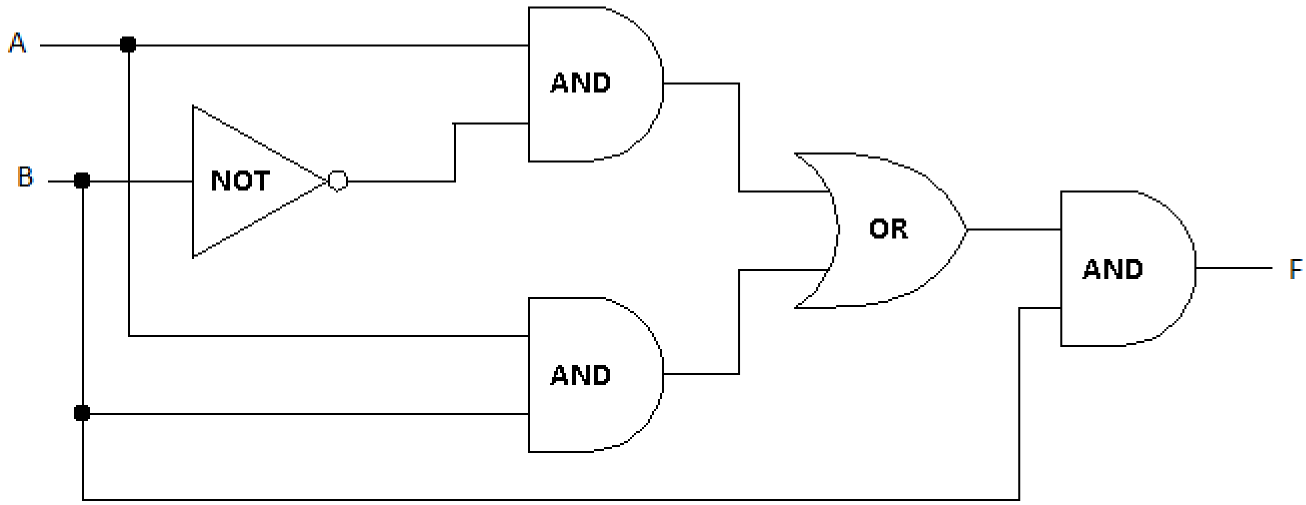
4. Examples
   a. A + ~A * B = A + B. Why?

| Assertion | Reason |
|---|---|
| A + ~A * B | Initial function |
| = (A + ~A) * (A + B) | Distributive Law for OR |
| = 1 * (A + B) | Complement Law for OR |
| = (A + B) * 1 | Commutative Law for AND (won't bother with this from now on) |
| = A + B | Identity Law for AND |

   b. 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 |

c. 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 OR |
| = (A * 1) * B | Complement Law for OR |
| = A * B | Identity Law for AND |