# Johns Hopkins Engineering

## Module 1: Boolean & Digital Logic

EN605.204 Computer Organization

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

# Introduction

- How do bits change/flow through digital circuits?
- What is Boolean logic?
- How do digital circuits implement Boolean logic?

# What is a "bit"?

- Computer circuits are based on Boolean algebra
- Boolean algebra operates on 1's and 0's
- Claude Shannon showed the relationship between Boolean algebra and digital circuits in 1930's

# Logical NOT

- Takes in a single bit as input and negates ("flips") it
- Can be denoted using the ~ or ¬ character

| Input (X) | Output (¬X) |
| --- | --- |
| 0 | 1 |
| 1 | 0 |

# Logical OR

- Takes two bits as input and outputs a 1 if either input was a 1
- Can be denoted using the ∨, ∪, or + character

| Input (X, Y) | | Output (X∨Y) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Logical AND

- Takes two bits as input and outputs a 1 if both inputs were 1
- Can be denoted using the ∧, ∩, or * character

| Input (X, Y) | | Output (X∧Y) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Logical XOR

- Two bits as input and outputs a 1 if exactly one input was a 1
- Can be denoted using the ⊗ character

| Input (X, Y) | | Output (X⊗Y) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Logical NOR

- Two bits as input and outputs a 1 if no inputs were a 1

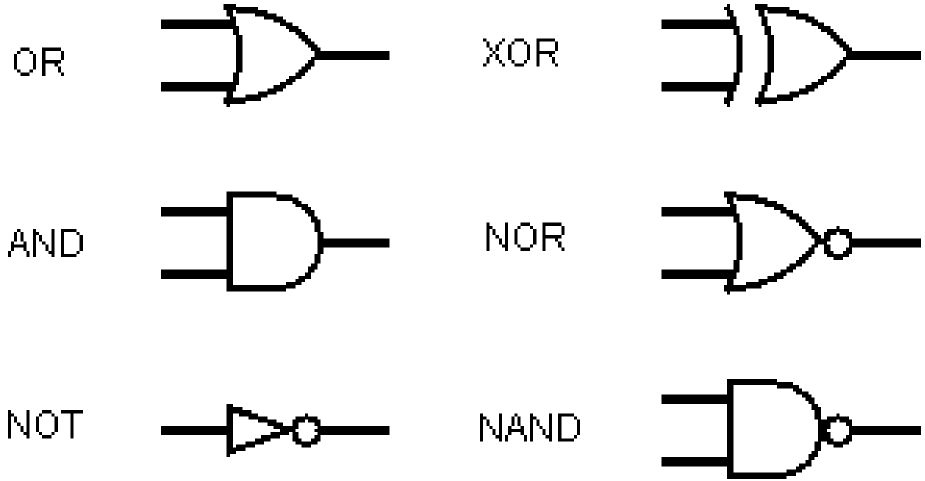
| Input (X, Y) | | Output ¬(X∨Y) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Logical NAND

- Two bits as input and outputs a 1 if at least 1 bit was a 0

| Input (X, Y) | | Output ¬(X∧Y) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

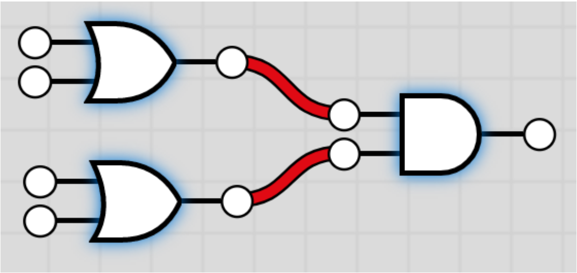
# Logic Gates



# Combinational Logic

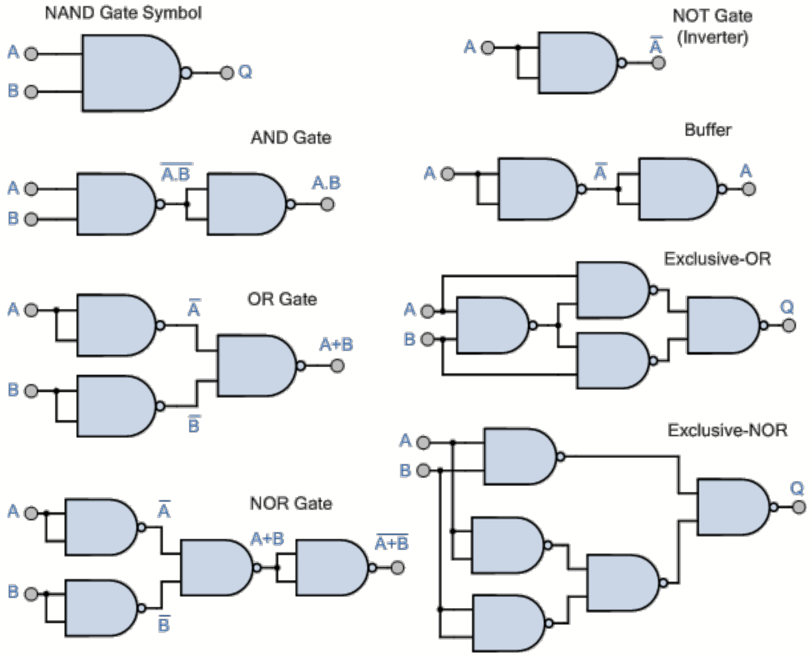
- Computer logic is implemented using physical "gates"
- Gates can be strung together to build more complex logic

- Example: AND(OR(W, X), OR(Y, Z))
  - Input: {0, 0, 0, 1}
  - = AND(OR(0, 0), OR(0, 1))
  - = AND(0, 1)
  - = 0



# Universal Logic Gates

- NOR and NAND gates are "universal"; they can simulate any logical operation



# Laws of Boolean Logic

| Input (X, Y) | Output $\neg(X \wedge Y)$ | |
|---|---|---|
| **Identity** | $X \wedge 1 = X$ | $X \vee 0 = X$ |
| **Complementation** | $X \wedge \neg X = 0$ | $X \vee \neg X = 1$ |
| **Double Negation** | $\neg(\neg X) = X$ | |
| **Idempotent** | $X \wedge X = X$ | $X \vee X = X$ |
| **Dominance** | $X \wedge 0 = 0$ | $X \neg 1 = 1$ |
| **Commutative** | $X \wedge Y = Y \wedge X$ | |
| **Associative** | $X \wedge Y \wedge Z = X \wedge (Y \wedge Z) = (X \wedge Y) \wedge Z$ | |
| **Distributive** | $X \wedge (Y \vee Z) = (X \wedge Y) \vee (X \wedge Z)$<br>$X \vee (Y \wedge Z) = (X \vee Y) \wedge (X \vee Z)$ | |

# DeMorgan's Law

- (X OR Y) and (X AND Y) are called "propositions"
- ¬(X OR Y) = ¬X AND ¬Y
  - The proposition X OR Y fails if X and Y are false
  - Ex: "I don't like the Cowboys (X) or Patriots (Y)" = "I don't like the Cowboys (X) and I don't like the Patriots (Y)."
- ¬(X AND Y) = ¬X OR ¬Y
  - The proposition fails if either X is false or Y is false
  - Ex: "I don't want a sandwich unless it has both ham (X) and cheese (Y)." = "I don't want a sandwich if it doesn't ham (X) or if it doesn't have cheese (Y)."