

All Computers are Built Upon
the Same Building Blocks

Building Blocks of Modern Computers

Transistors

Transistors

Problem

Algorithm

Program/Language

Runtime System
(VM, OS, MM)

ISA (Architecture)

Microarchitecture

Logic

Devices

Electrons

Transistors

- **Computers are built from very large numbers of very small (and relatively simple) structures: transistors**

- ❑ Intel 4004, in 1971, had **2300 MOS transistors**
- ❑ Intel's Pentium IV microprocessor, 2000, was made up of more than **42 Million MOS transistors**
- ❑ Apple's M2 Max, offered for sale in 2022, is made up of more than **67 Billion MOS transistors**
- ❑ Micron's V-NAND flash memory (2022), is made up of more than **5 Trillion MOS transistors**

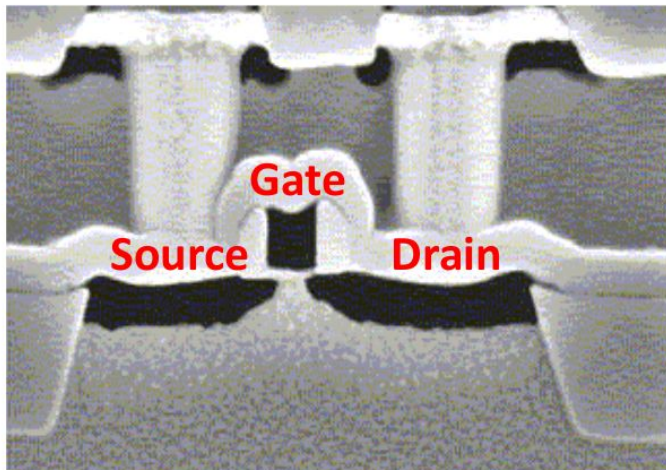
- **This lecture**

- ❑ How the MOS transistor works (as logic element)
- ❑ How transistors are connected to form logic gates
- ❑ How logic gates are interconnected to form larger units that are needed to construct a computer

Problem
Algorithm
Program/Language
Runtime System (VM, OS, MM)
ISA (Architecture)
Microarchitecture
Logic
Devices
Electrons

MOS Transistor

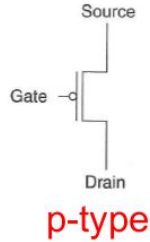
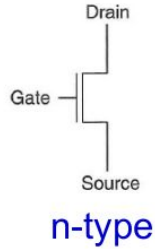
- By combining
 - Conductors (**M**etal)
 - Insulators (**O**xide)
 - **S**emiconductors
- We get a Transistor (MOS)
- Why is this useful?
 - We can combine many of these to realize simple logic gates
- The **electrical properties** of metal-oxide semiconductors are well **beyond** the scope of what we want to understand in this course
 - They are below our lowest level of abstraction



1 H Hydrogen																2 He Helium	
3 Li Lithium	4 Be Beryllium											5 B Boron	6 C Carbon	7 N Nitrogen	8 O Oxygen	9 F Fluorine	10 Ne Neon
11 Na Sodium	12 Mg Magnesium											13 Al Aluminum	14 Si Silicon	15 P Phosphorus	16 S Sulfur	17 Cl Chlorine	18 Ar Argon
19 K Potassium	20 Ca Calcium	21 Sc Scandium	22 Ti Titanium	23 V Vanadium	24 Cr Chromium	25 Mn Manganese	26 Fe Iron	27 Co Cobalt	28 Ni Nickel	29 Cu Copper	30 Zn Zinc	31 Ga Gallium	32 Ge Germanium	33 As Arsenic	34 Se Selenium	35 Br Bromine	36 Kr Krypton
37 Rb Rubidium	38 Sr Strontium	39 Y Yttrium	40 Zr Zirconium	41 Nb Niobium	42 Mo Molybdenum	43 Tc Technetium	44 Ru Ruthenium	45 Rh Rhodium	46 Pd Palladium	47 Ag Silver	48 Cd Cadmium	49 In Indium	50 Sn Tin	51 Sb Antimony	52 Te Tellurium	53 I Iodine	54 Xe Xenon
55 Cs Cesium	56 Ba Barium	*	72 Hf Hafnium	73 Ta Tantalum	74 W Tungsten	75 Re Rhenium	76 Os Osmium	77 Ir Iridium	78 Pt Platinum	79 Au Gold	80 Hg Mercury	81 Tl Thallium	82 Pb Lead	83 Bi Bismuth	84 Po Polonium	85 At Astatine	86 Rn Radon
87 Fr Francium	88 Ra Radium	**	104 Rf Rutherfordium	105 Db Dubnium	106 Sg Seaborgium	107 Bh Bohrium	108 Hs Hassium	109 Mt Meitnerium	110 Ds Darmstadtium	111 Rg Roentgenium	112 Cn Copernicium	113 Nh Nihonium	114 Fl Flerovium	115 Mc Moscovium	116 Lv Livermorium	117 Ts Tennessine	118 Og Oganesson
			57 La Lanthanum	58 Ce Cerium	59 Pr Praseodymium	60 Nd Neodymium	61 Pm Promethium	62 Sm Samarium	63 Eu Europium	64 Gd Gadolinium	65 Tb Terbium	66 Dy Dysprosium	67 Ho Holmium	68 Er Erbium	69 Tm Thulium	70 Yb Ytterbium	71 Lu Lutetium
			89 Ac Actinium	90 Th Thorium	91 Pa Protactinium	92 U Uranium	93 Np Neptunium	94 Pu Plutonium	95 Am Americium	96 Cm Curium	97 Bk Berkelium	98 Cf Californium	99 Es Einsteinium	100 Fm Fermium	101 Md Mendelevium	102 No Nobelium	103 Lr Lawrencium

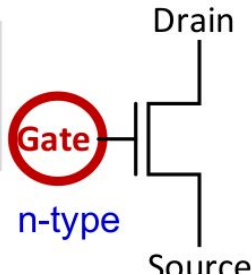
Different Types of MOS Transistors

- There are two types of MOS transistors: **n-type** and **p-type**

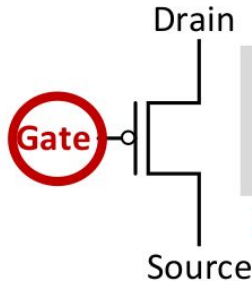


- They both operate "logically," very similar to the way wall switches work

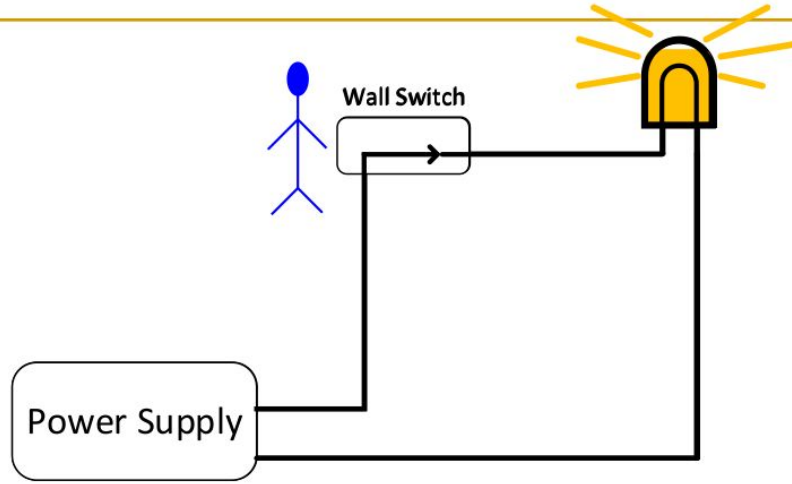
The circuit is closed when the gate is supplied with 3V



The circuit is closed when the gate is supplied with 0V



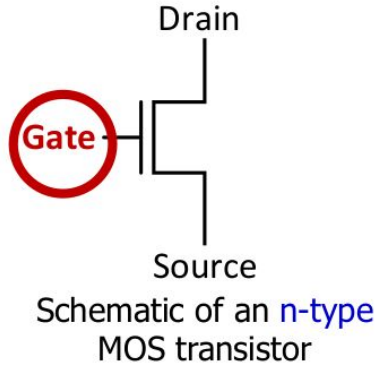
How Does a Wall Switch Work?



- ❑ In order for the lamp to glow, **electrons must flow**
- ❑ In order for electrons to flow, there must be a **closed circuit** from the power supply to the lamp and back to the power supply
- ❑ The lamp can be **turned on and off** by simply manipulating the wall switch to make or break the closed circuit

Transistor Works As a Switch

- Instead of the wall switch, we could use an **n-type** or a **p-type** MOS transistor to make or break the closed circuit



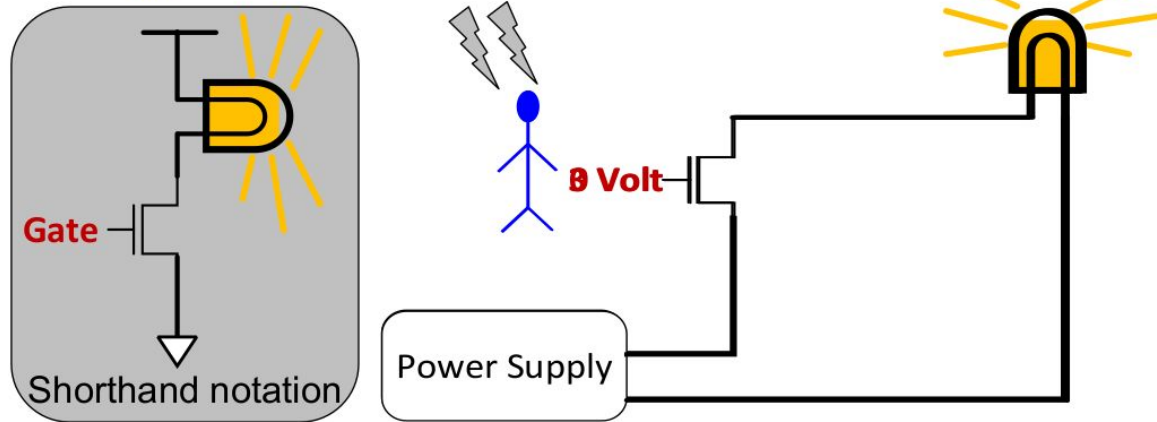
If the gate of an **n-type** transistor is supplied with a **high** voltage, the connection from source to drain acts like a **piece of wire** (i.e., the circuit is closed)

Depending on the technology, high voltage can range from 0.3V to 3V

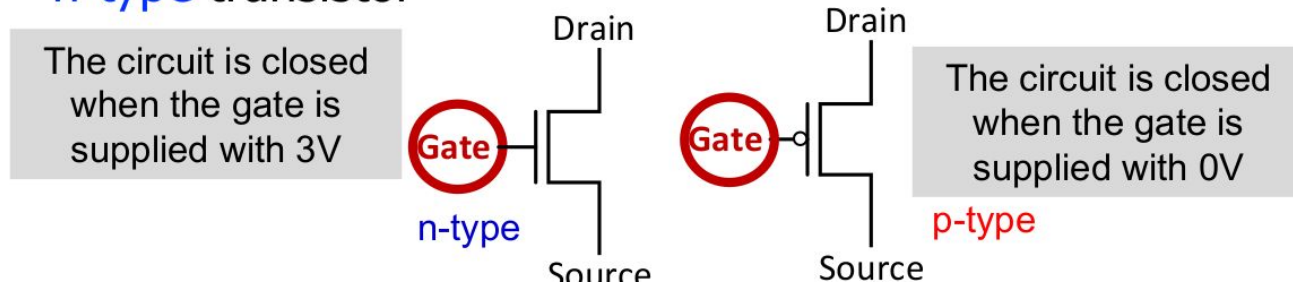
If the gate of the **n-type** transistor is supplied with **zero** voltage, the connection between the source and drain is **broken** (i.e., the circuit is open)

Abstraction: Transistor As a Switch

- **n-type** transistor in a circuit with a battery and a bulb



- **p-type** transistor works in exactly the opposite fashion from **n-type** transistor



Logic Gates

One Level Higher in the Abstraction

- **Now, we know how a MOS transistor works**
- How do we build logic structures out of MOS transistors?
- We construct basic logical units out of individual MOS transistors
- These **logical units** are called **logic gates**
 - They implement simple **Boolean** functions

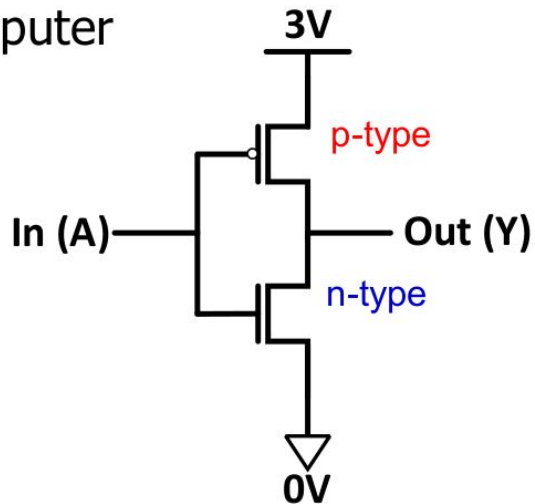
Problem
Algorithm
Program/Language
Runtime System (VM, OS, MM)
ISA (Architecture)
Microarchitecture
Logic
Devices
Electrons

Making Logic Blocks Using CMOS Technology

- Modern computers use both **n-type** and **p-type** transistors, i.e. Complementary MOS (**CMOS**) technology

$$\text{nMOS} + \text{pMOS} = \text{CMOS}$$

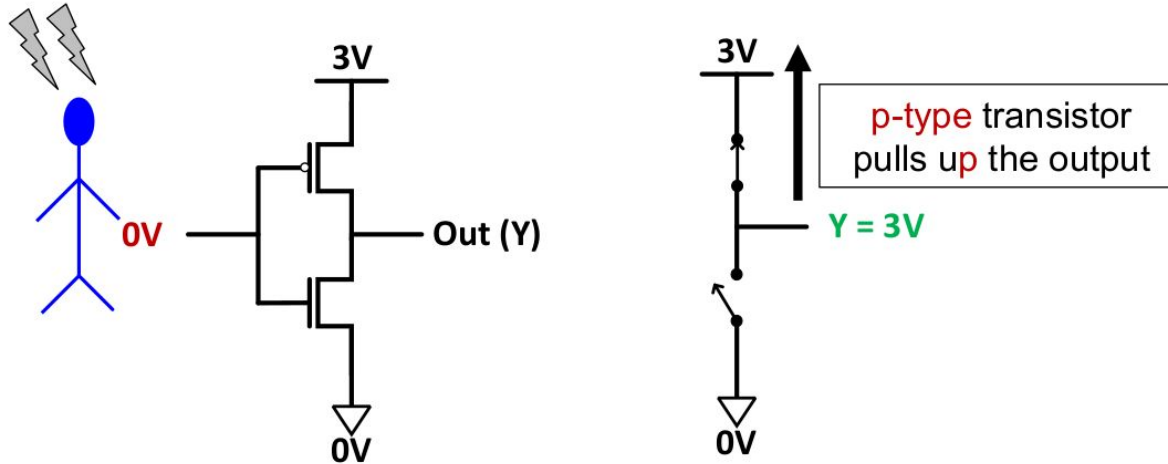
- The simplest logic structure that exists in a modern computer



What does this circuit do?

Functionality of Our CMOS Circuit

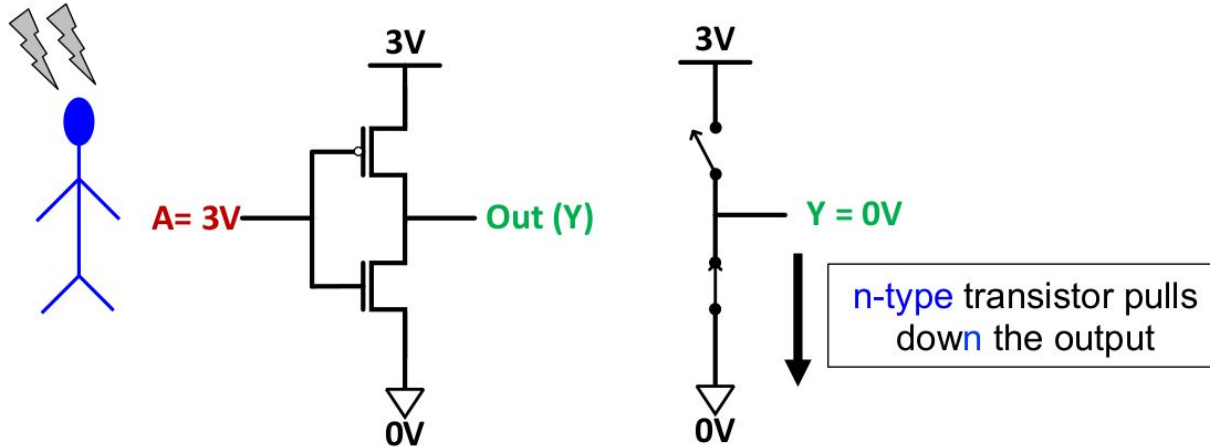
What happens when the input is connected to 0V?



p-type transistors are good at **pulling up** the voltage

Functionality of Our CMOS Circuit

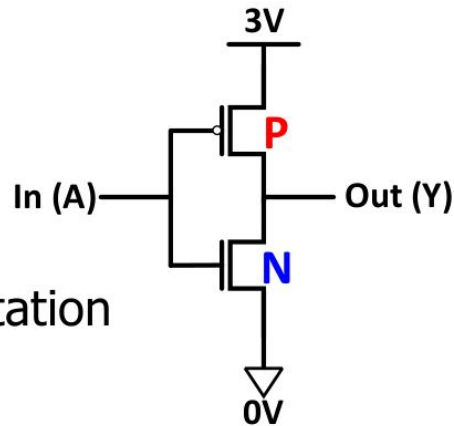
What happens when the input is connected to 3V?



n-type transistors are good at pulling down the voltage

CMOS NOT Gate (Inverter)

- This is the **CMOS NOT Gate**
- Why do we call it NOT?
 - If $A = 0V$ then $Y = 3V$
 - If $A = 3V$ then $Y = 0V$
- **Digital circuit:** one possible interpretation
 - Interpret **0V** as logical (binary) **0** value
 - Interpret **3V** as logical (binary) **1** value

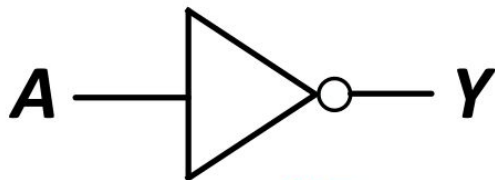
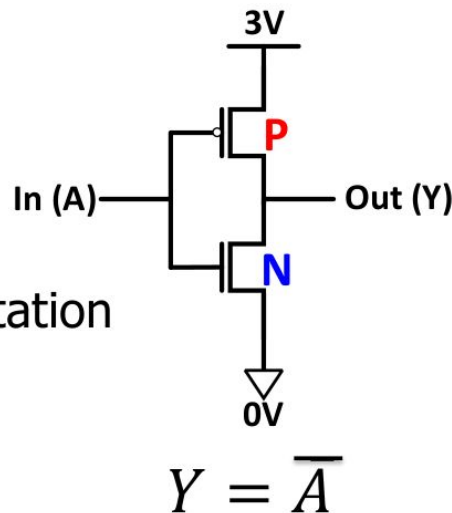


A	P	N	Y
0	ON	OFF	1
1	OFF	ON	0

$$Y = \overline{A}$$

CMOS NOT Gate (Inverter)

- This is actually the CMOS NOT Gate
- Why do we call it NOT?
 - If $A = 0V$ then $Y = 3V$
 - If $A = 3V$ then $Y = 0V$
- **Digital circuit:** one possible interpretation
 - Interpret $0V$ as logical (binary) 0 value
 - Interpret $3V$ as logical (binary) 1 value



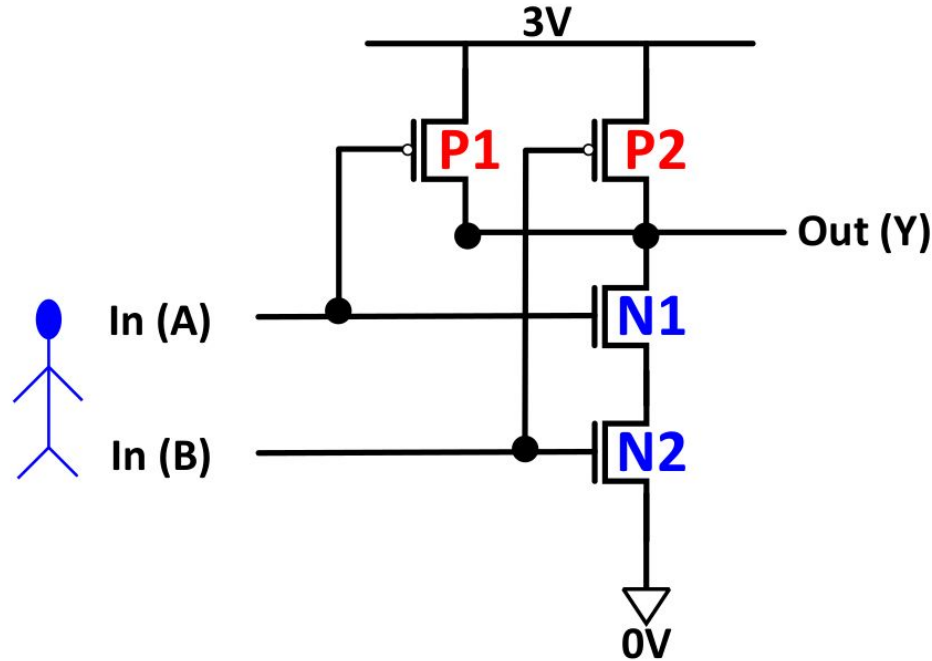
We call this a **NOT** gate
or an **inverter**
(bubble indicates inversion)

Truth table: shows the logical output of the circuit for each possible input

A	Y
0	1
1	0

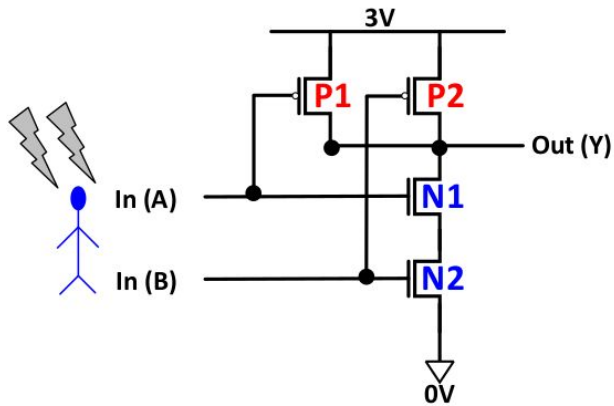
Another CMOS Gate: What Is This?

- Let's build more complex gates!



CMOS NAND Gate

- Let's build more complex gates!



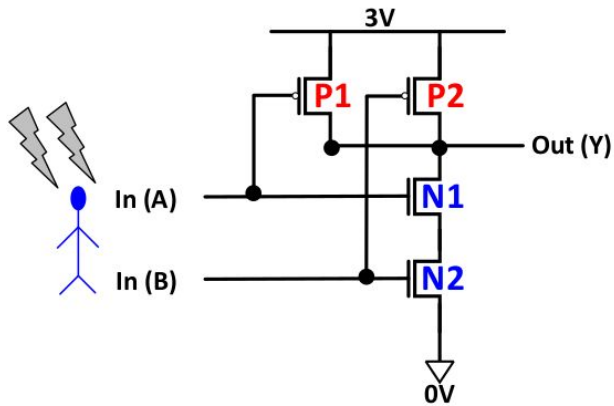
$$Y = \overline{A \cdot B} = \overline{AB}$$

A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

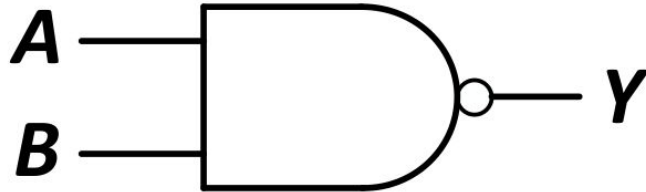
- P1 and P2 are in parallel; only one must be ON to pull up the output to 3V
- N1 and N2 are connected in series; both must be ON to pull down the output to 0V

CMOS NAND Gate

- Let's build more complex gates!



$$Y = \overline{A \cdot B} = \overline{AB}$$



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

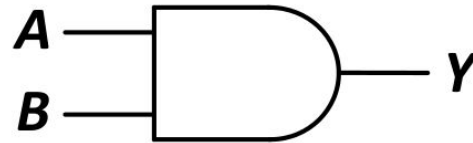
We call this a **NAND** gate

CMOS AND Gate

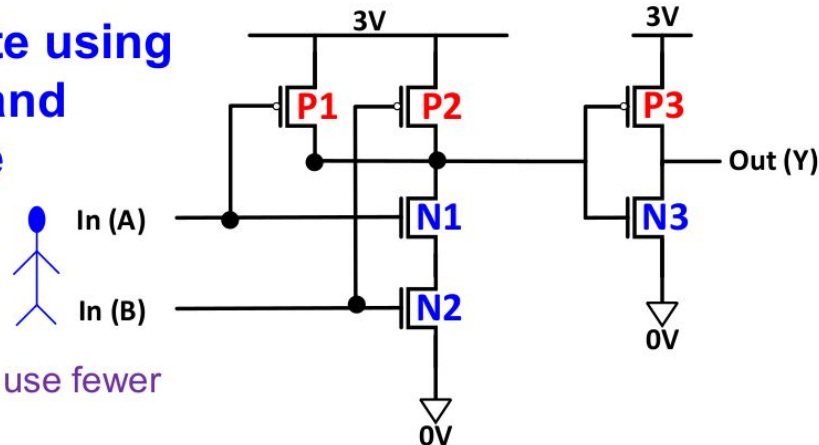
- How can we make an AND gate?

$$Y = A \cdot B = AB$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

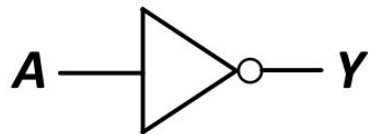


We make an **AND** gate using
one **NAND** gate and
one **NOT** gate

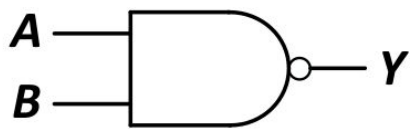


Food for thought: Can we not use fewer transistors for the AND gate?

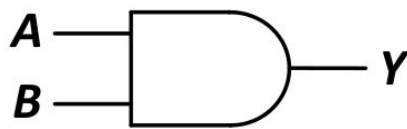
CMOS NOT, NAND, AND Gates



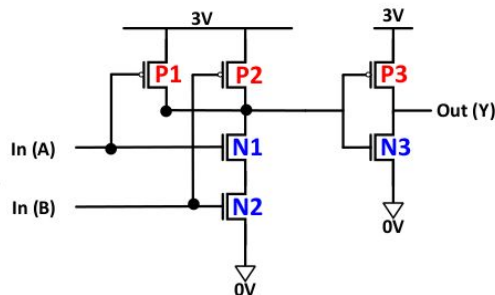
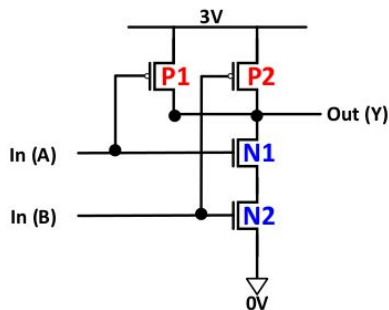
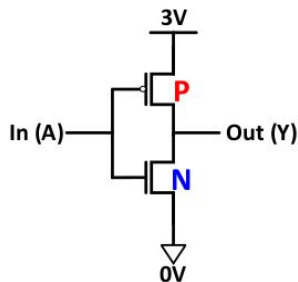
A	Y
0	1
1	0



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

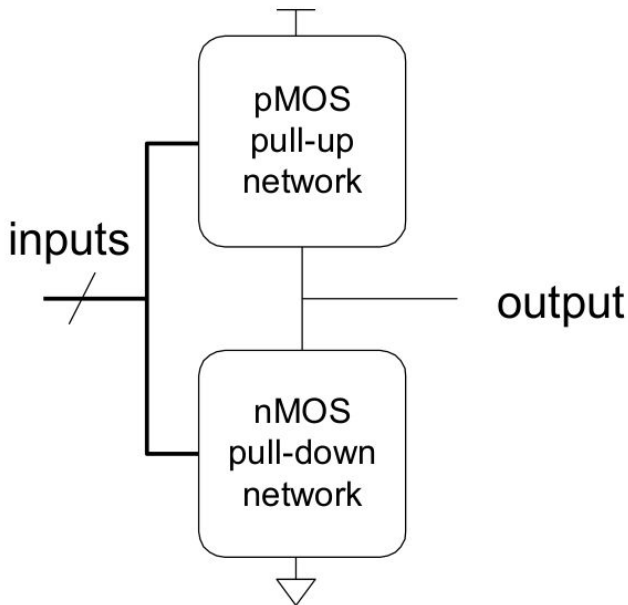


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



General CMOS Gate Structure

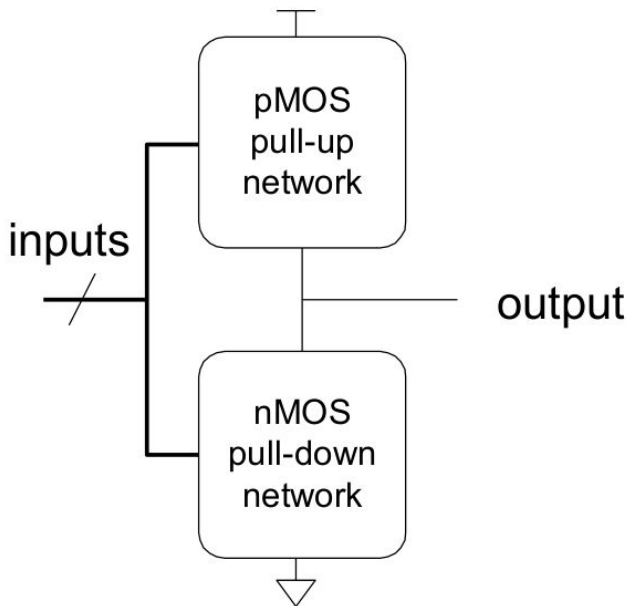
- The general form used to construct any inverting logic gate, such as: NOT, NAND, or NOR
 - The networks may consist of transistors in series or in parallel
 - When transistors are in **parallel**, the network is **ON** if **one** of the transistors is **ON**
 - When transistors are in **series**, the network is **ON** only if **all** transistors are **ON**



pMOS transistors are used for pull-up
nMOS transistors are used for pull-down

General CMOS Gate Structure (II)

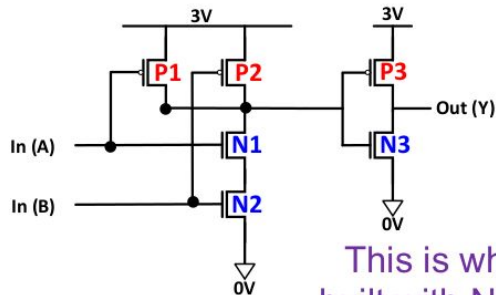
- Exactly one network should be ON, and the other network should be OFF at any given time
 - If both networks are ON at the same time, there is a **short circuit** → likely incorrect operation
 - If both networks are OFF at the same time, the output is **floating** → undefined



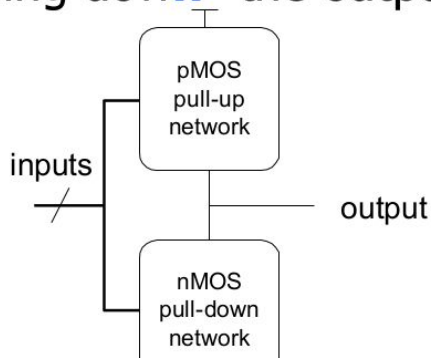
pMOS transistors are used for pull-up
nMOS transistors are used for pull-down

Digging Deeper: Why This Structure?

- MOS transistors are **imperfect** switches
- pMOS transistors pass 1's well but 0's poorly (holes carry charge)
- nMOS transistors pass 0's well but 1's poorly (electrons carry charge)
- **p**MOS transistors are good at "pulling up" the output
- **n**MOS transistors are good at "pulling down" the output



This is why AND is
built with NAND + NOT



Digging Deeper: Power Consumption

■ Dynamic Power Consumption

- Power used to charge capacitance as signals change ($0 \leftrightarrow 1$)
- $C * V^2 * f$
 - C = capacitance of the circuit (wires and gates)
 - V = supply voltage
 - f = charging frequency of the capacitor

■ Static Power Consumption

- Power used when signals do not change
- $V * I_{\text{leakage}}$
 - supply voltage * leakage current

■ Energy Consumption

- Power * Time

Our World
in Data

This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.



Licensed under [CC-BY-SA](#) by the author Max Roser.

Common Logic Gates

Buffer



A	Z
0	0
1	1

AND



A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

XOR



A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Inverter



A	Z
0	1
1	0

NAND



A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

XNOR



A	B	Z
0	0	1
0	1	0
1	0	0
1	1	1

We Can Now Build Logic Circuits

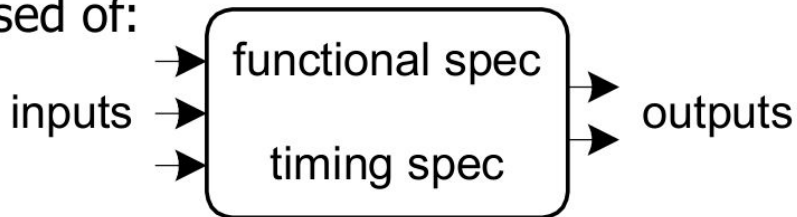
Now, we understand the workings of the basic logic gates

What is our next step?

Build some of the logic structures that are important components of the microarchitecture of a computer

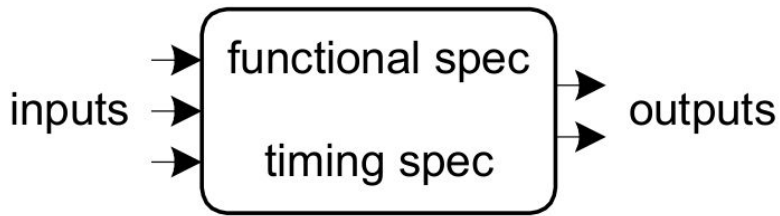
- A logic circuit is composed of:

- ❑ Inputs
- ❑ Outputs



- *Functional specification* (describes relationship between inputs and outputs)
- *Timing specification* (describes the delay between inputs changing and outputs responding)

Types of Logic Circuits



■ Combinational Logic

□ Memoryless

- Outputs are strictly dependent on the combination of input values that are being applied to circuit *right now*

■ Later we will learn: Sequential Logic

□ Has memory

- Structure stores history → Can "store" data values
- Outputs are determined by previous (historical) and current values of inputs