

9/23/26



COMP 311: *Computer Organization!*

Lecture 9: Transistors and CMOS Gates



The University
of North Carolina
at Chapel Hill

<https://tinyurl.com/comp311-fa26>

Logistics Update



- Lab 1 is due next week!
- I'm pushing back quiz 3. The others are staying!
- WA2 going out today

Plan for Class Today

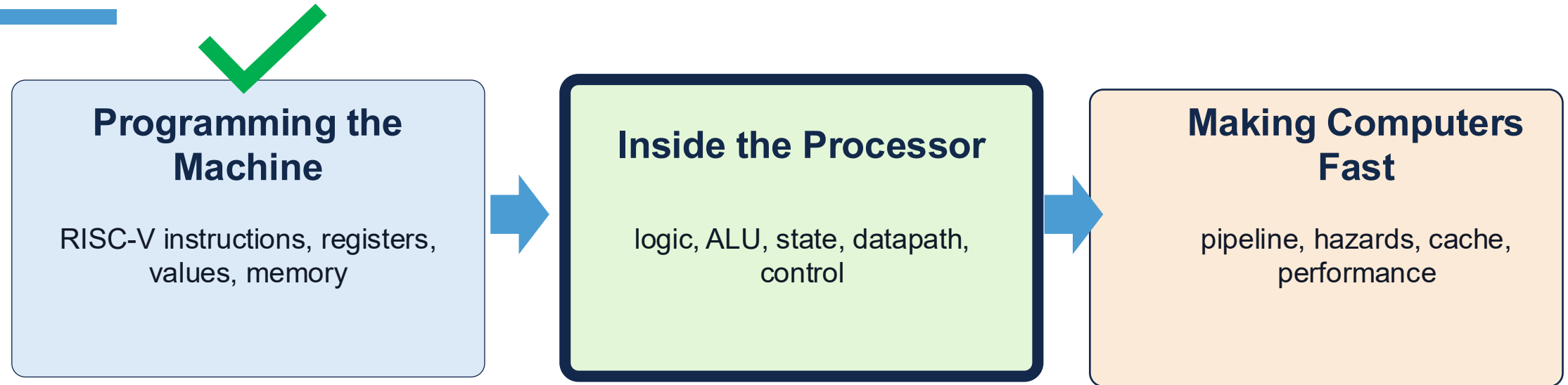


- 3:35-4:00: Basic Circuits, Switches
- 4:00-4:15: Transistors
- 4:15-4:20: Break
- 4:20-4:50: More Transistors, CMOS Gates

Guiding Question 😊

*How do we build
useful computation
from simple on/off
components?*

Onto the next chapter!! 😊



- We now know how to program in assembly, according to the ISA.
 - How can hardware actually make that happen?
- We'll build up from logic + state to the ALU, data path, and control.
- **Goal:** connect each assembly instruction to the hardware actions needed to execute it.



UNIT 2: INSIDE THE PROCESSOR

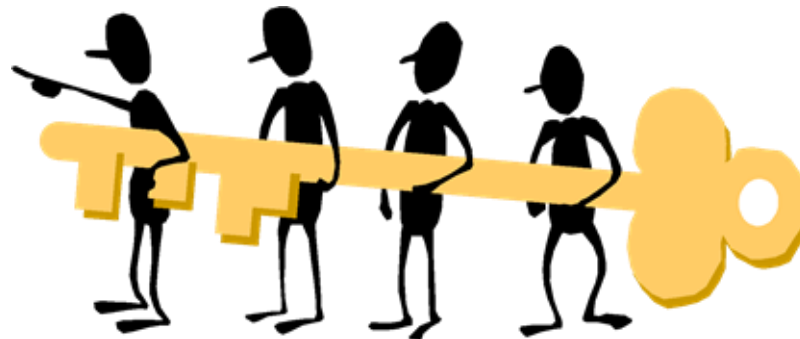
Basic Circuits

Contracts: Key to System Design



A SYSTEM is a structure that is “guaranteed” to exhibit a specified behavior, assuming all of its components obey their specified behaviors.

How is this achieved?



Through Contracts

Every system component will have clear obligations and responsibilities. If these are maintained we have every right to expect the system to behave as planned. If contracts are violated all bets are off.

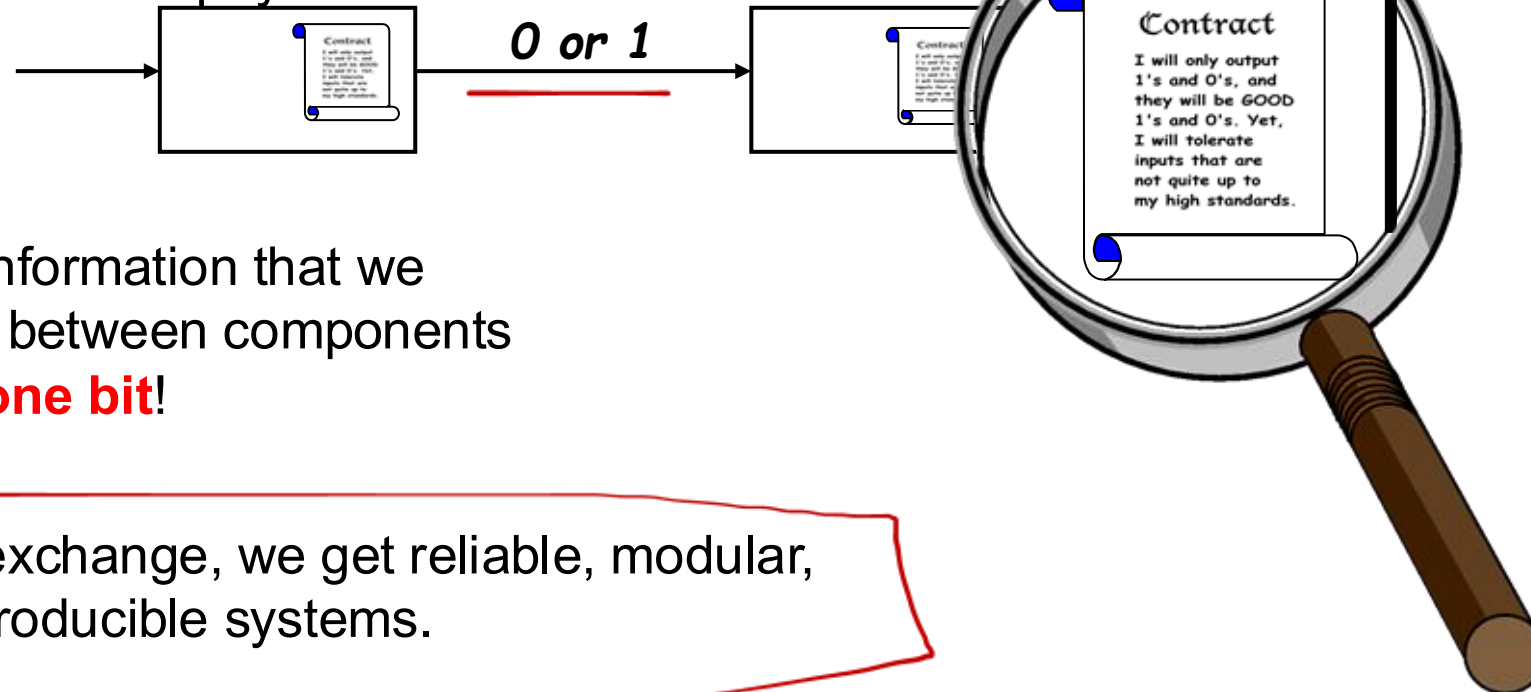
Digital Contracts



Why DIGITAL?

... because it keeps the contracts SIMPLE!

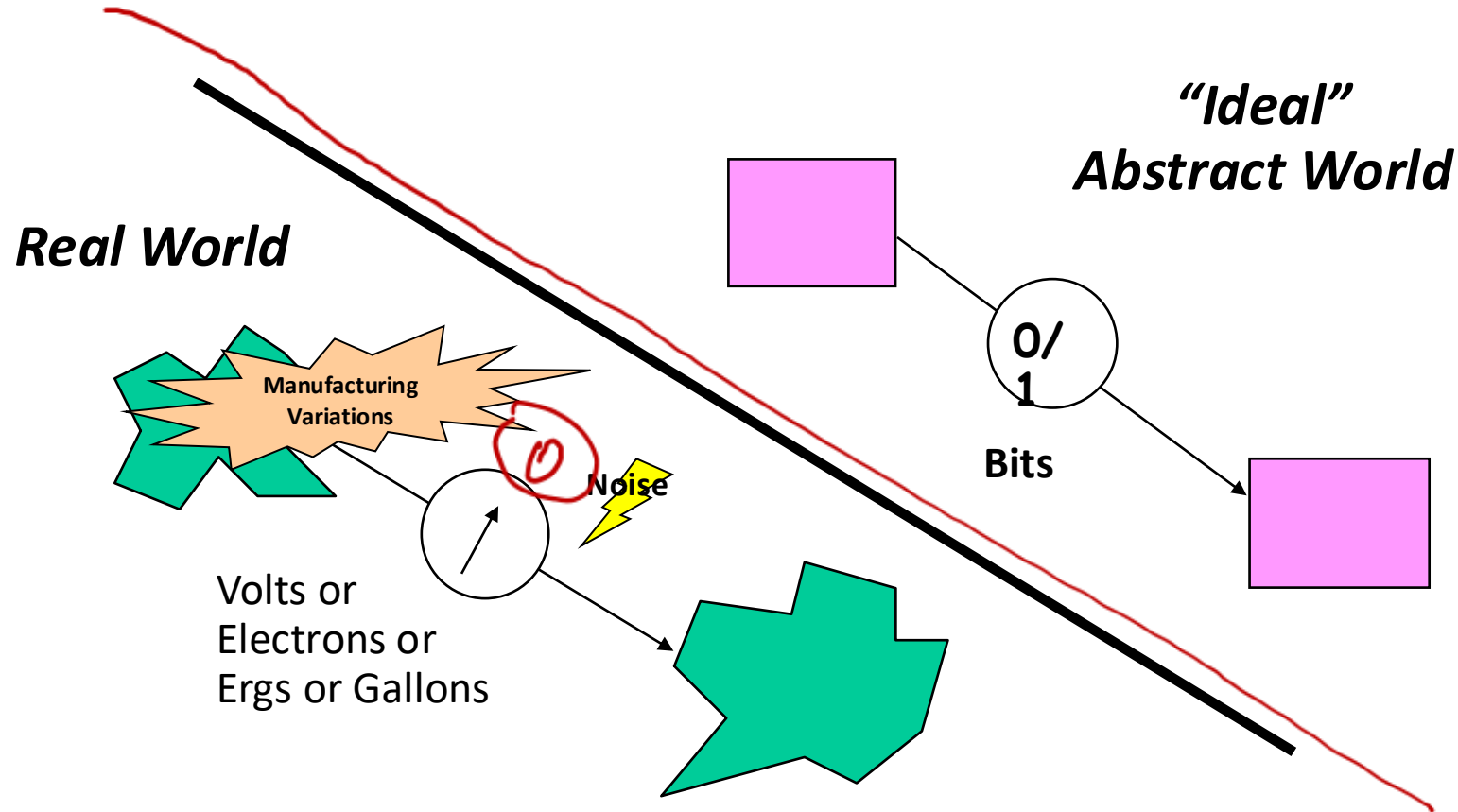
It's the price we pay for this robustness?



All the information that we transfer between components is only **one bit**!

But, in exchange, we get reliable, modular, and reproducible systems.

The Digital Abstraction



Keep in mind, **the real world is not digital, we engineer it to behave that way.**
We **coerce** real physical phenomena to implement digital designs!

Nose Margins



- Key idea:

Don't allow "0" to be mistaken for a "1" or vice versa

- Use the same "uniform bit-representation convention", for every component in our digital system
- To implement devices with high reliability, we outlaw "close calls" via a representation convention which forbids a range of voltages between "0" and "1".

- Ensure the valid input range is more tolerant (larger) than the valid output range

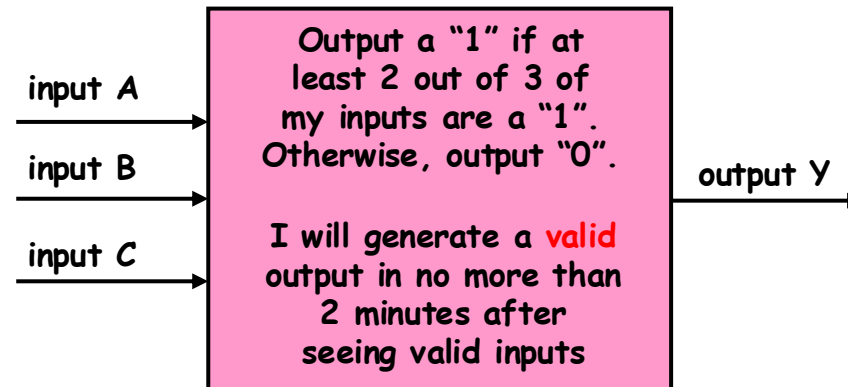


Our definition of valid does not preclude inputs and outputs from passing through invalid values. In fact, they must, but only during transitions. Our specifications allow for this (i.e. outputs are specified sometime (T_{pd}) after inputs become valid).

A Digital Processing Element



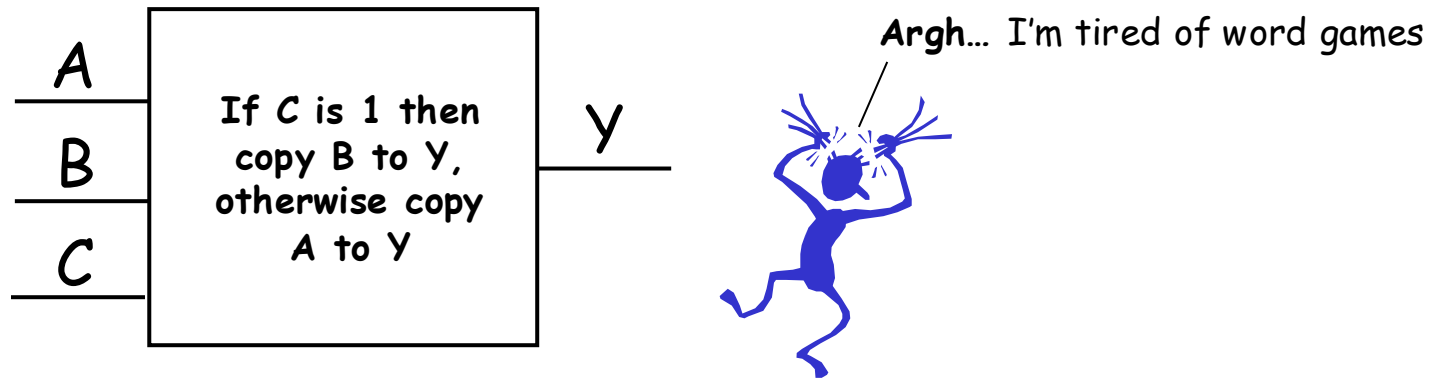
- A **combinational device** is a digital element that has
 - one or more digital inputs *0 / 1*
 - one or more digital outputs
 - a functional specification that details the value of each output for **every possible combination of valid input values**
 - a timing specification consisting (at a minimum) an upper bound propagation delay, t_{pd} , on the required time for the device to compute the specified **valid** output values from an arbitrary set of stable, **valid** input values



Functional Specifications: Truth Tables



We need to start somewhere – usually with a functional specification



Every combinational function can be expressed as a table!

“Truth tables” are a concise description of the combinational system’s function, where an output is specified for **every** input combination.

Q1 on WS



A Combinational Digital System



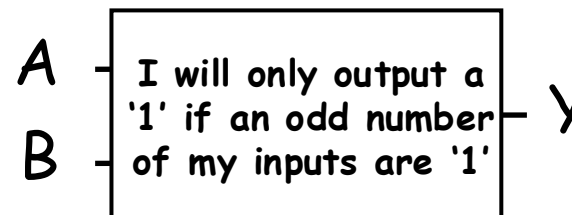
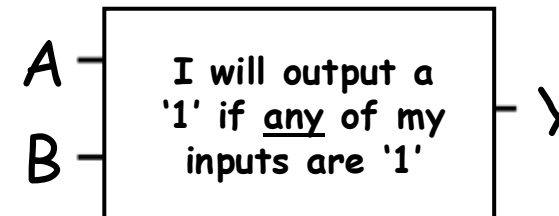
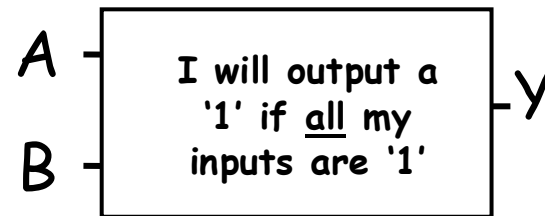
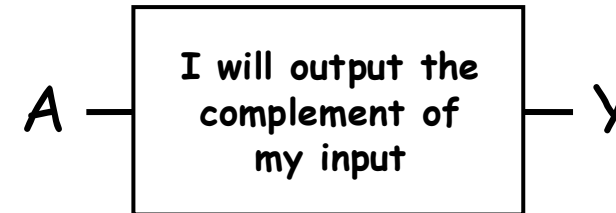
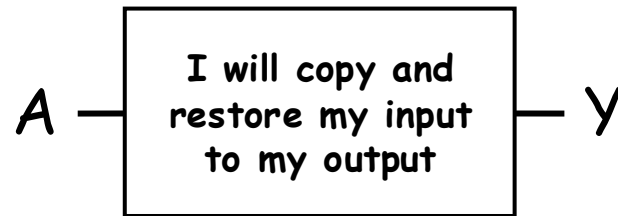
A system of interconnected elements is combinational if:

- Each circuit element is combinational
- Every input is connected to exactly one output or directly to some source of 0's or 1's
- The circuit contains no directed cycles
- Adheres to some noise margin threshold → AKA we can expect good 0s and good 1s! 😊

Digital Processing Elements (Gates)



Some digital processing elements occur so frequently that we give them special names and symbols



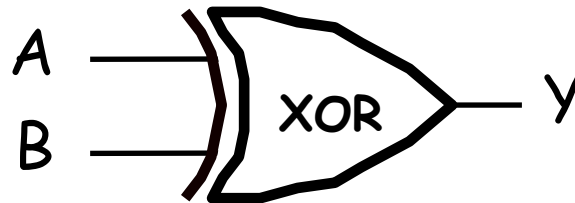
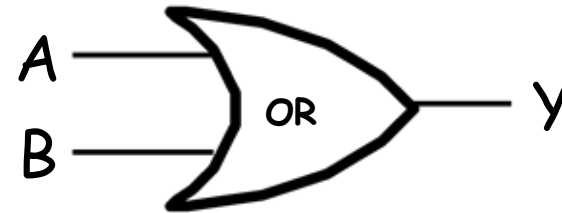
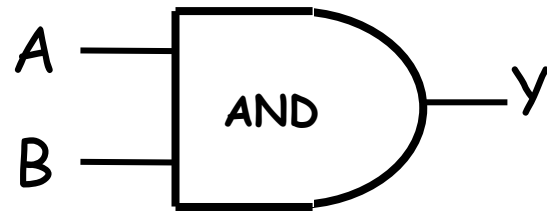
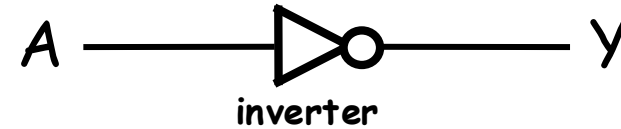
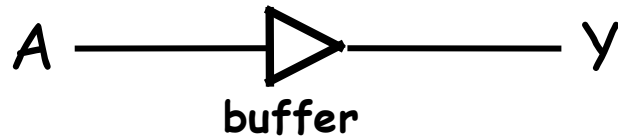
Q: What is the point of a buffer?
Doesn't a wire do the same thing?
A: A buffer restores marginal digital signals, because the output is as good or "better" than the input (i.e. it solves that bad image problem from slide 7).



Digital Processing Elements (Gates)



Some digital processing elements occur so frequently that we give them special names and symbols

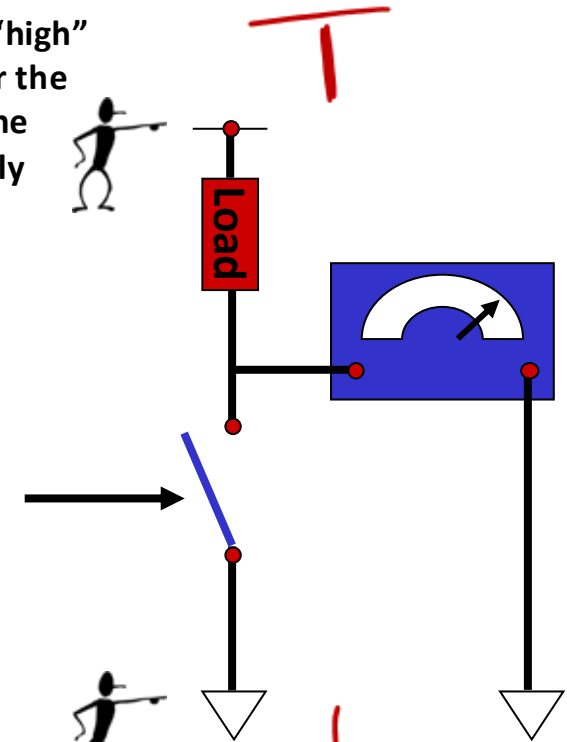


How do we make gates?



- A controllable switch is the common link of all computing technologies
- Switches control voltages by creating and opening paths between higher and lower potentials
- Closed circuit:
 - *fully connected, allows electricity to flow uninterrupted*
- Open circuit
 - *A circuit that contains a broken connection*
 - *Electricity stops flowing at the point where the connection was lost*

This symbol indicates a “high” potential, or the voltage of the power supply

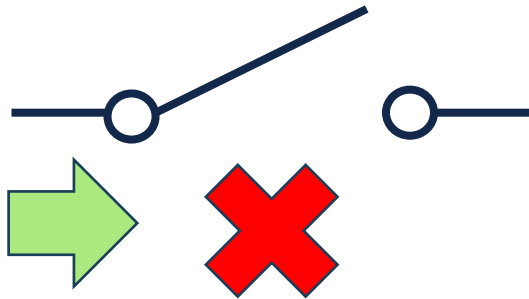


This symbol indicates a “low” or ground potential

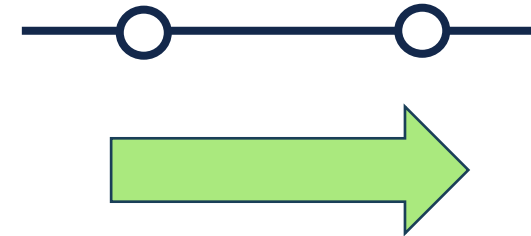
Switches



Open (off)

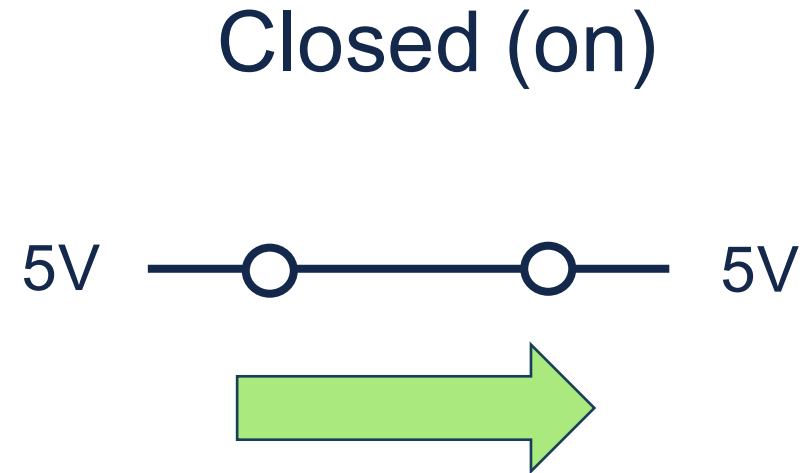
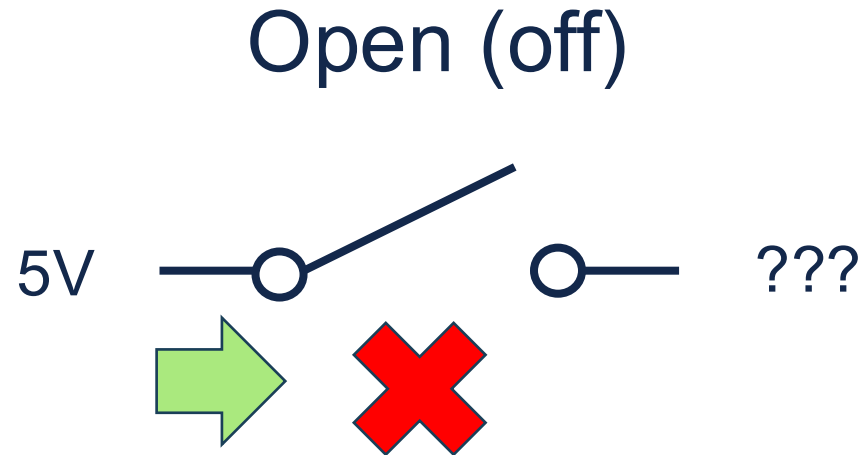


Closed (on)



Electricity can flow through a closed switch, it cannot flow through an open switch

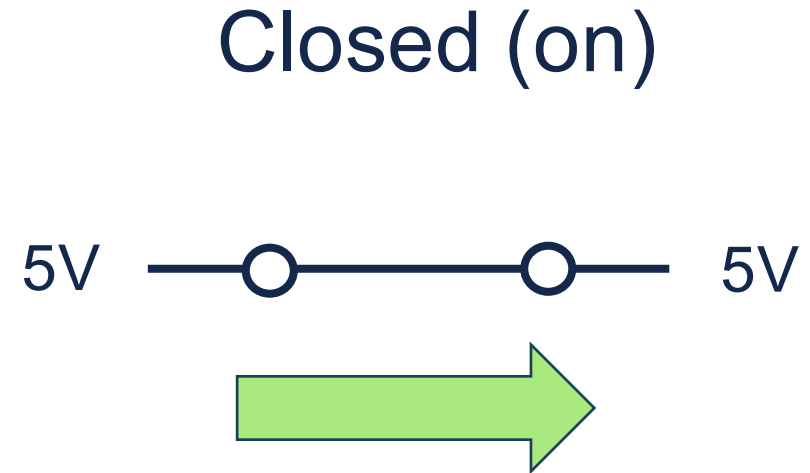
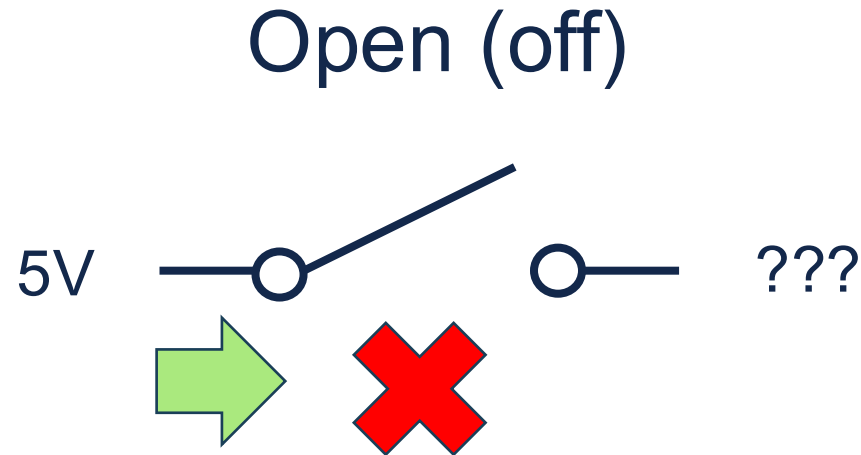
Switches



Let's say we apply 5V to the left side of the switch

- If the switch is open, we don't necessarily know the voltage on the right
 - Would need to see the full circuit to determine the voltage
- If the switch is closed, the voltage on the right is 5V

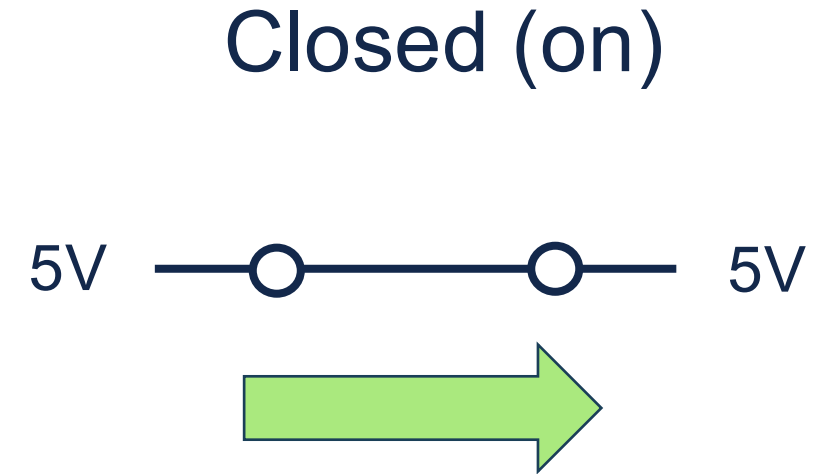
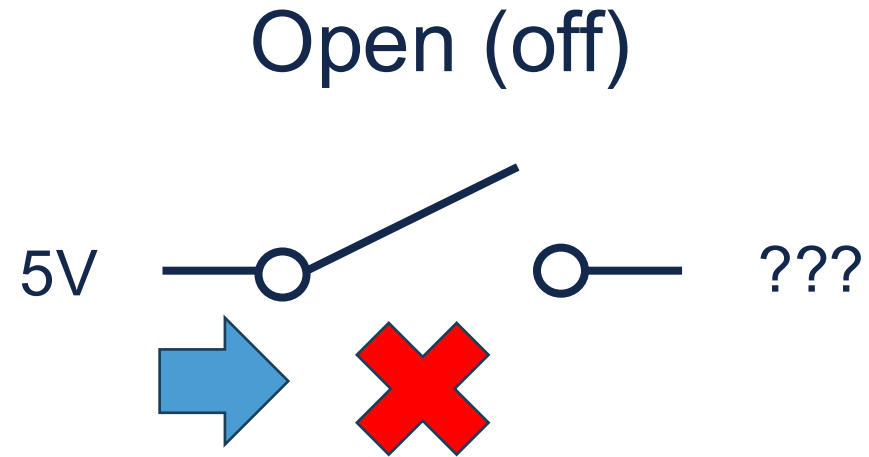
Switches



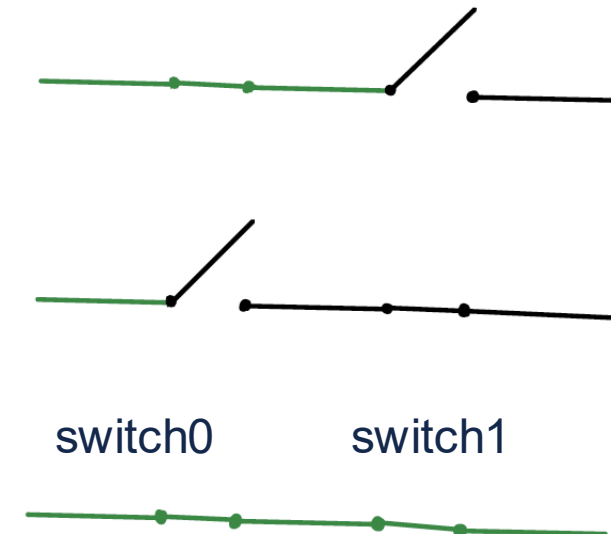
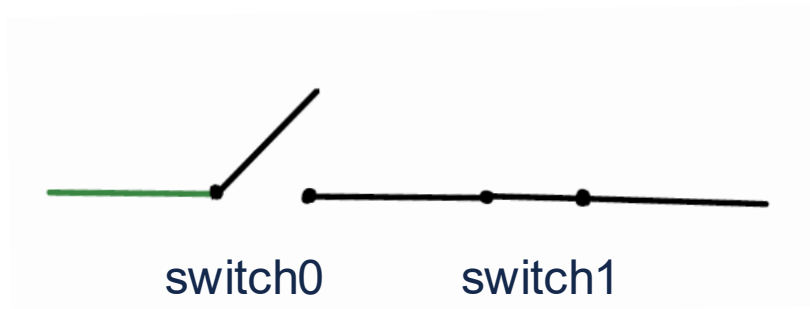
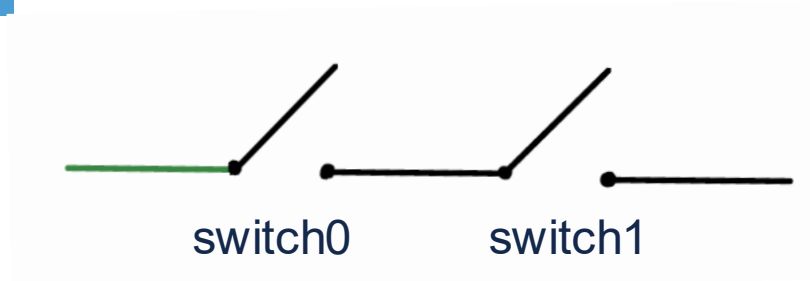
Let's say we apply 0V to the left side of the switch

- If the switch is open, we don't necessarily know the voltage on the right
 - Would need to see the full circuit to determine the voltage
- If the switch is closed, the voltage on the right is 0V

Switches



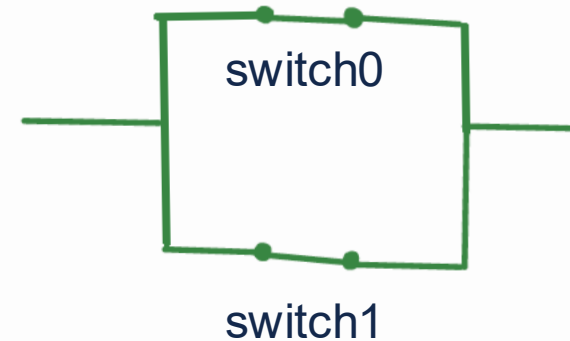
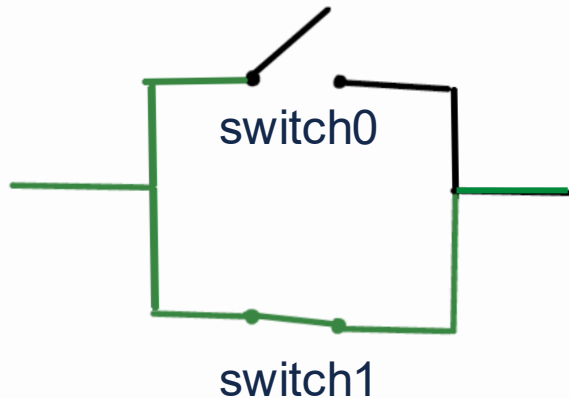
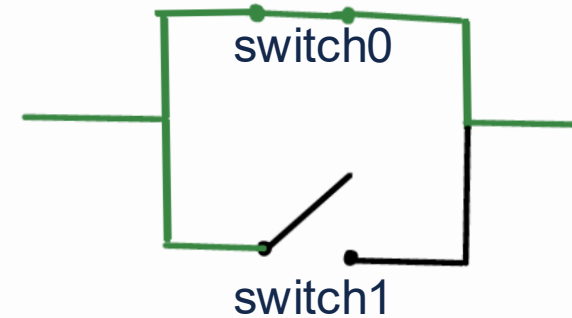
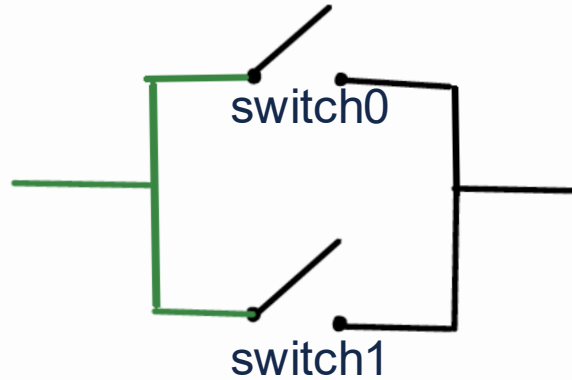
Switches in Series



Current can only flow between the two ends if both switches are closed

$$\text{current_flow} = \text{switch}_0 \text{ AND } \text{switch}_1$$

Switches in Parallel



Current can flow between the two ends if either of the switches is closed
 $\text{current_flow} = \text{switch}_0 \text{ OR } \text{switch}_1$

UNIT 1: PROGRAMMING THE MACHINE

Transistors 😊

Transistors are the physical building blocks of modern computers!



Number of Transistors in CPUs over Time



A transistor is the physical device we use to turn continuous electrical behavior into digital 0s and 1s.

You can think of it as an electrically controlled switch that turns current on or off.

Transistors are like switches



Switches are controlled by **physical contact**

Transistors are controlled by a **voltage**

Two States: Binary!



- Transistors can be in two distinct states: on or off
- This is why our machines communicate in binary (1s and 0s)!

On



Off



Logic Levels



- Instead of specifying the exact voltage level (e.g. 3V or 5V), we'll simply use binary terms when discussing transistors.
- We'll either have a **high voltage** or a **low voltage**.
- There are a few different terms for these voltage levels.
 - High voltage
 - logic high = logic 1 = 1
 - Low voltage
 - logic low = logic 0 = 0

Transistors

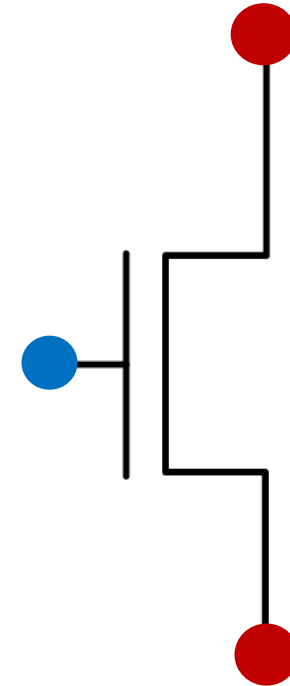


- We will be working with two types of transistors
 - nMOS transistors
 - pMOS transistors
- There are three terminals
 - **Gate**: controls whether the transistor is on or off
 - **Source** and **Drain**: endpoints

Transistor Symbol



Switch



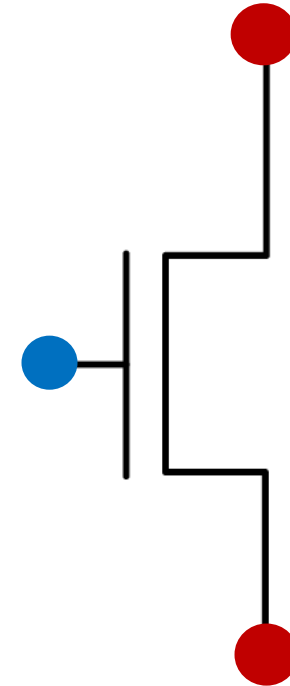
Transistor
(n-channel MOSFET)

The terminal labeled with the **blue** dot determines whether current can flow between the terminals labeled with the **red**

Transistor Symbol



Switch



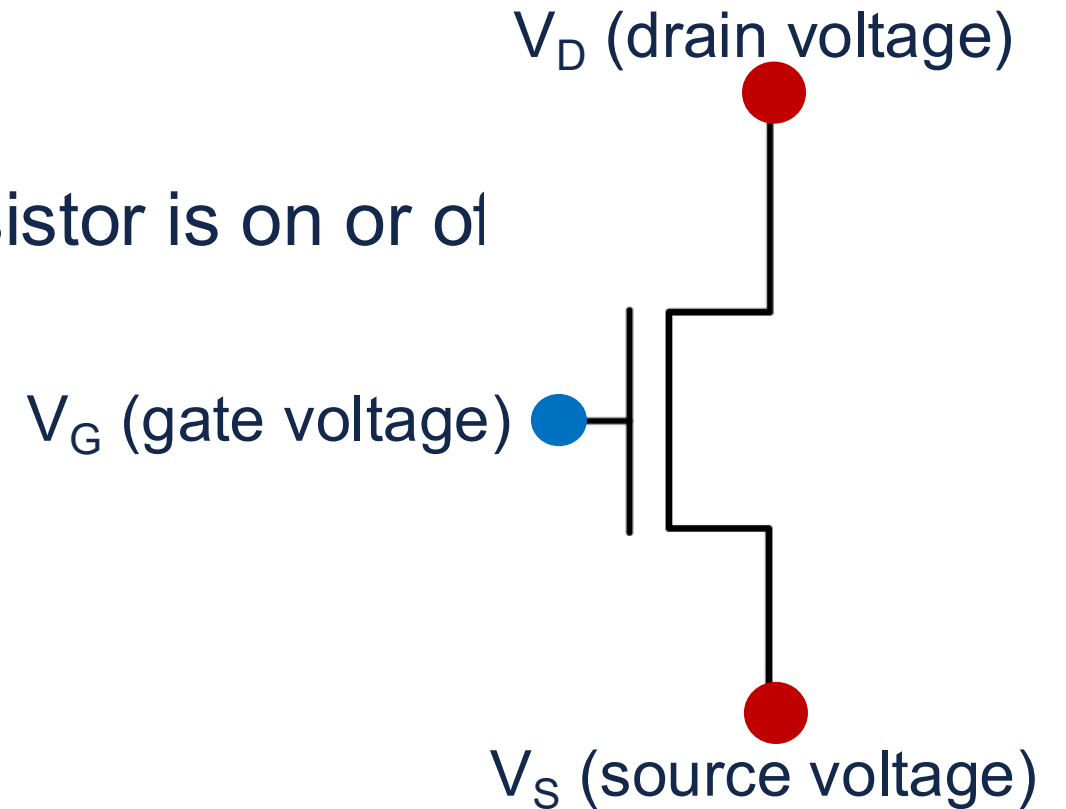
Transistor
(n-channel MOSFET)

The terminal labeled with the **blue** dot determines whether current can flow between the terminals labeled with the **red** dot.

NMOS Transistor



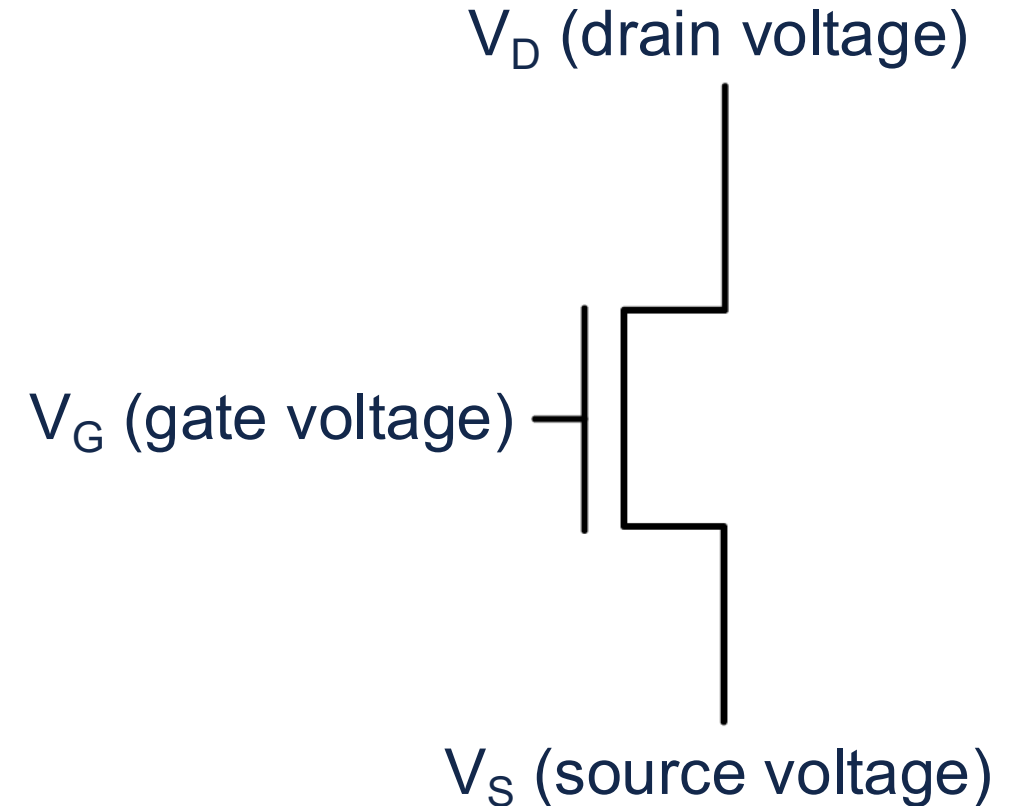
- There are three terminals
 - **Gate**: controls whether the transistor is on or off
 - **Source** and **Drain**: endpoints



NMOS Transistor



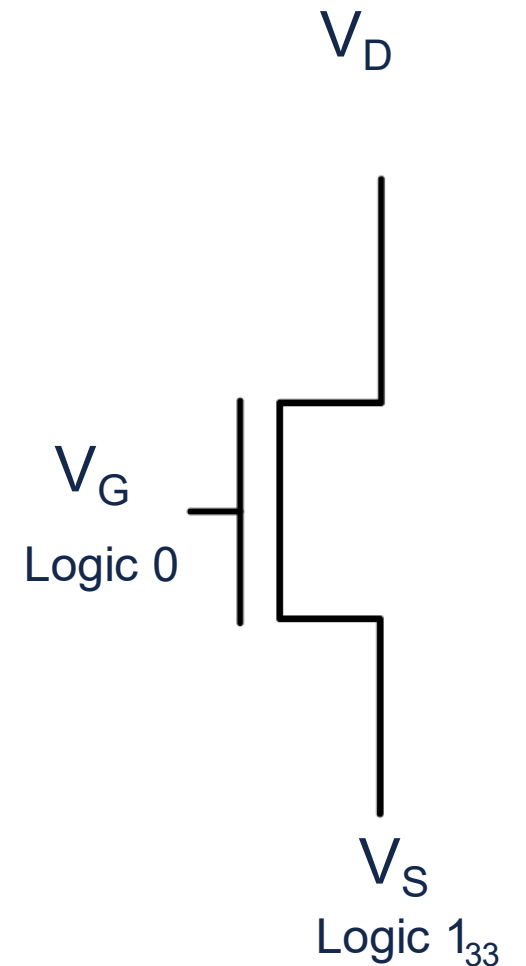
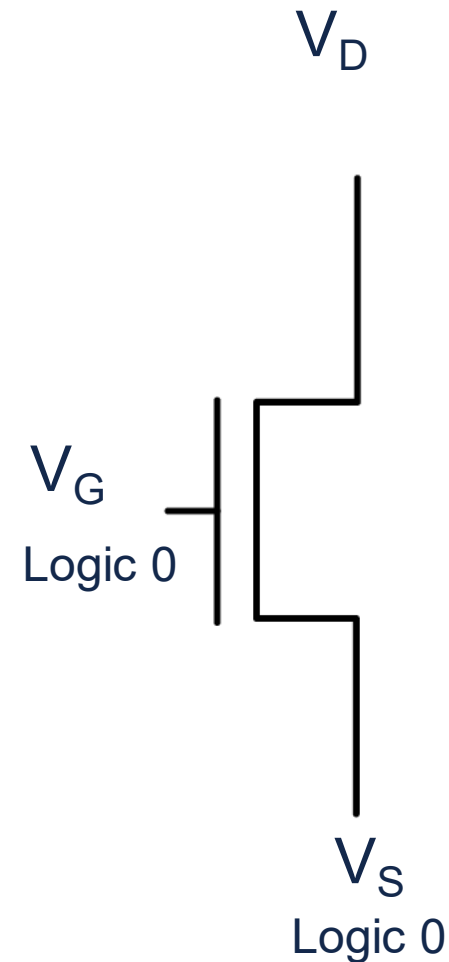
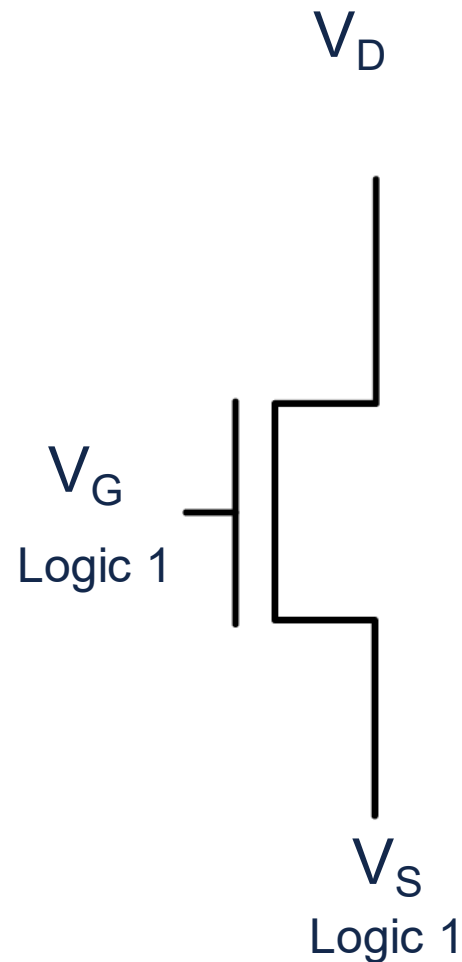
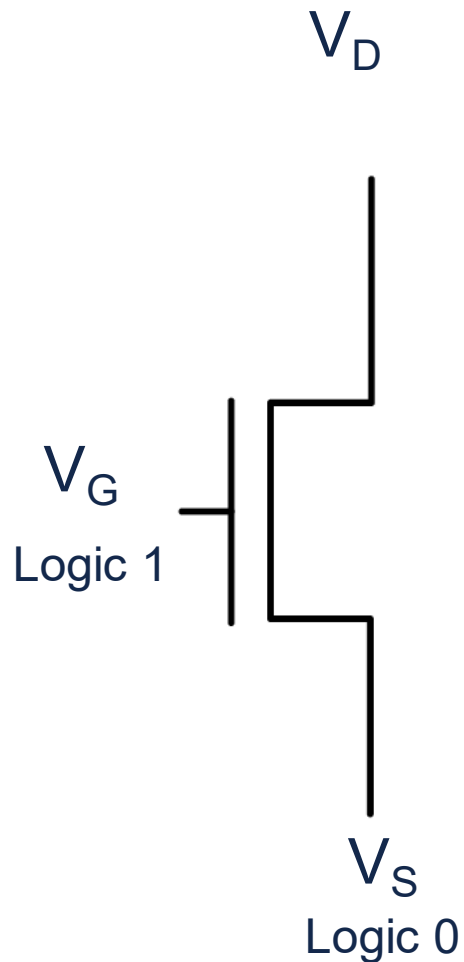
- If the gate voltage is **high (logic 1)**, current **can** flow between source and drain
 - The source and drain will be connected and will have the same voltage
- If the gate voltage is **low**, current **cannot** flow between source and drain
 - The source and drain are not connected



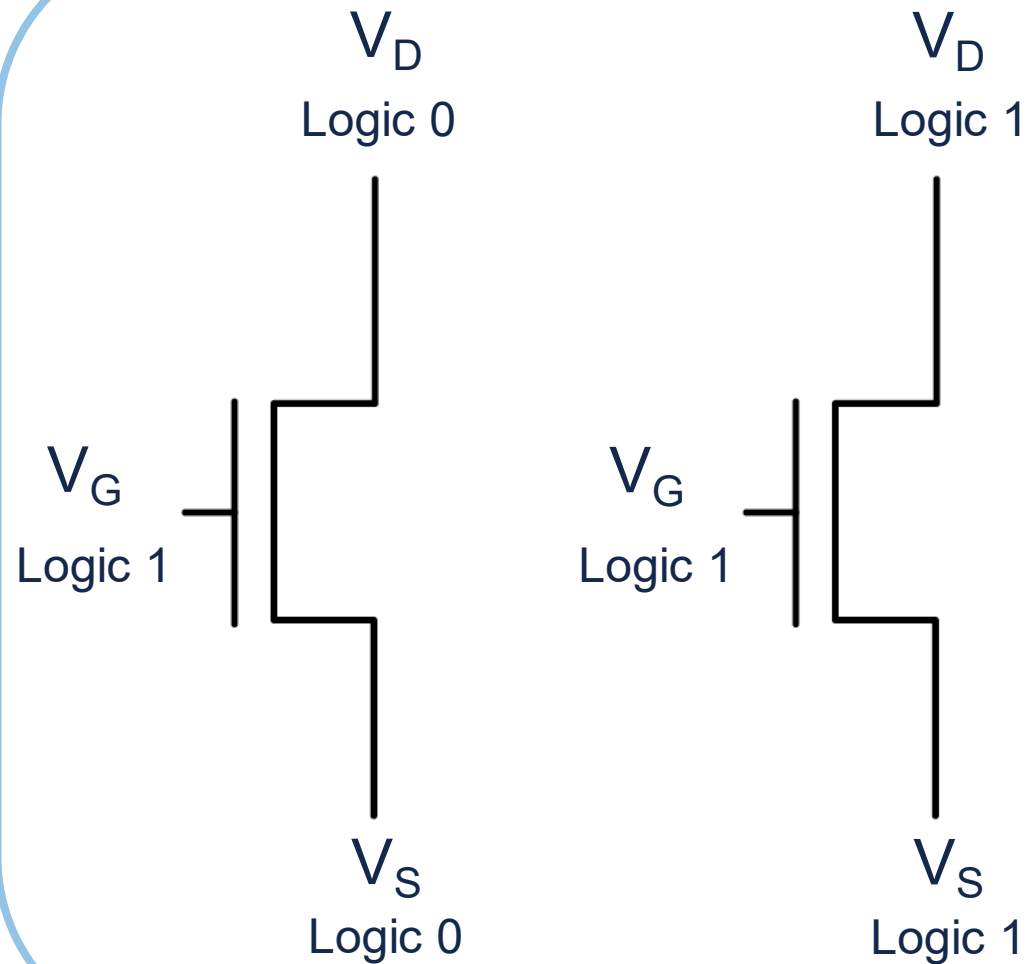
NMOS Transistor



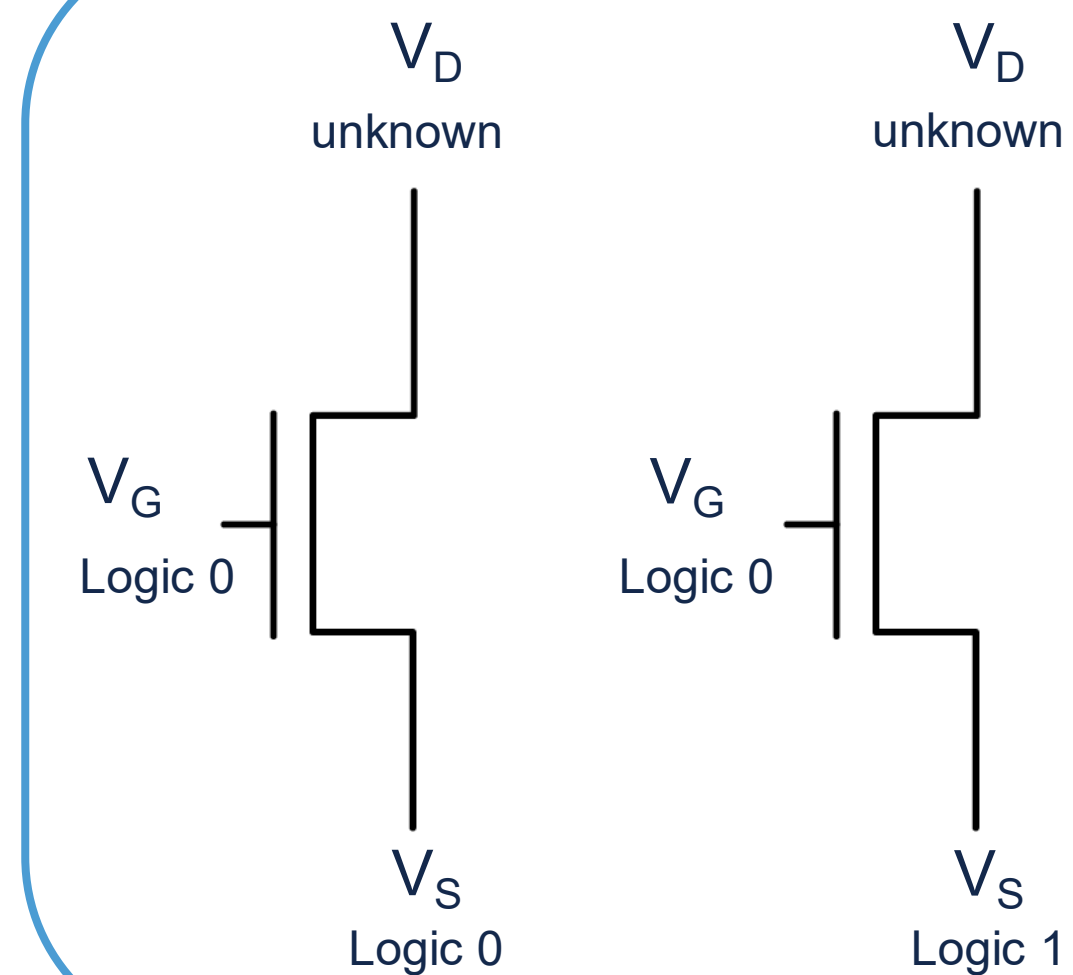
Let's say that we know the values of V_G and V_S . What is the value of V_D in each of the following diagrams?



NMOS Transistor



V_G is high, so source and drain are connected

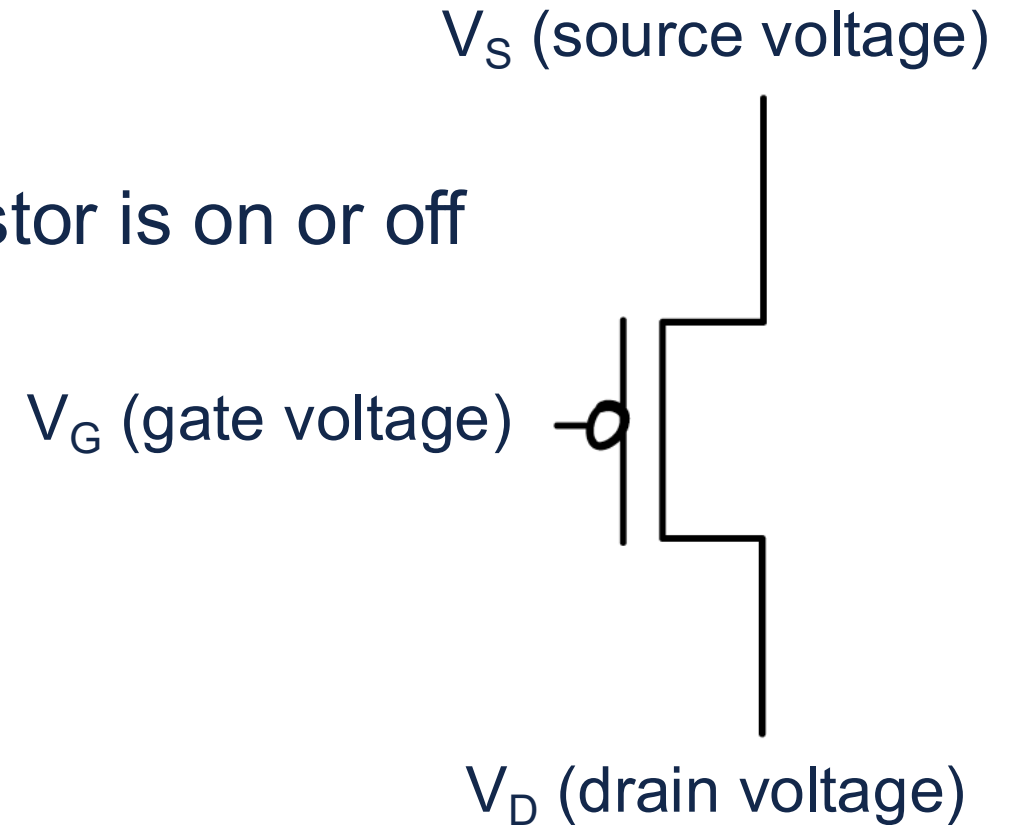


V_G is low, so source and drain are not connected

PMOS Transistor



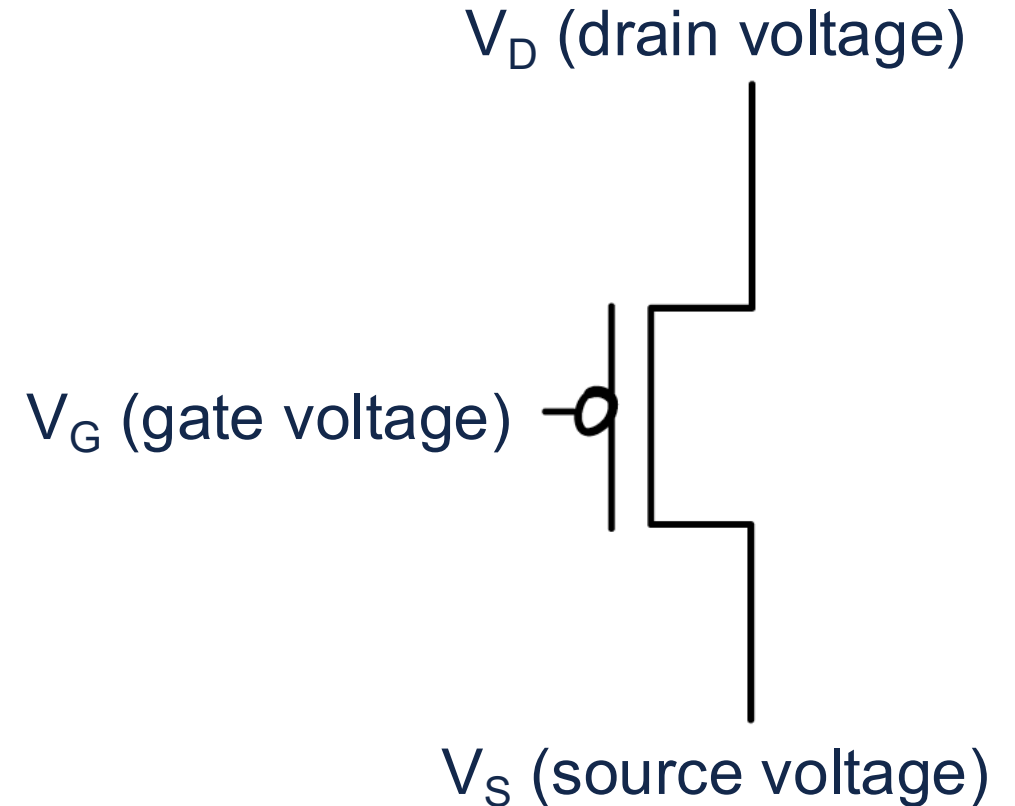
- There are three terminals
 - **Gate**: controls whether the transistor is on or off
 - **Source** and **Drain**: endpoints



PMOS Transistor



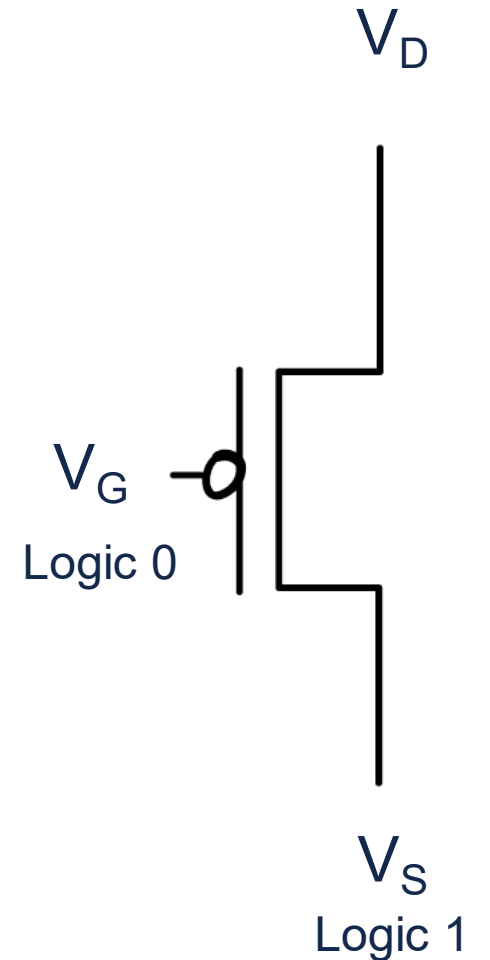
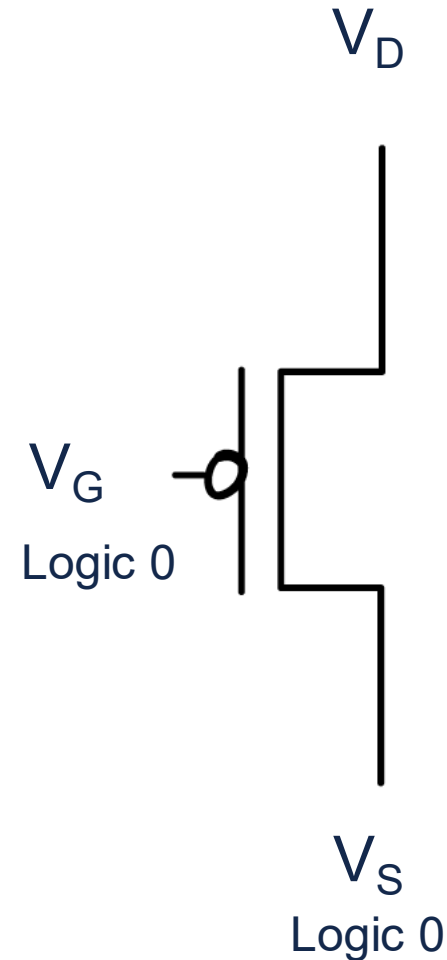
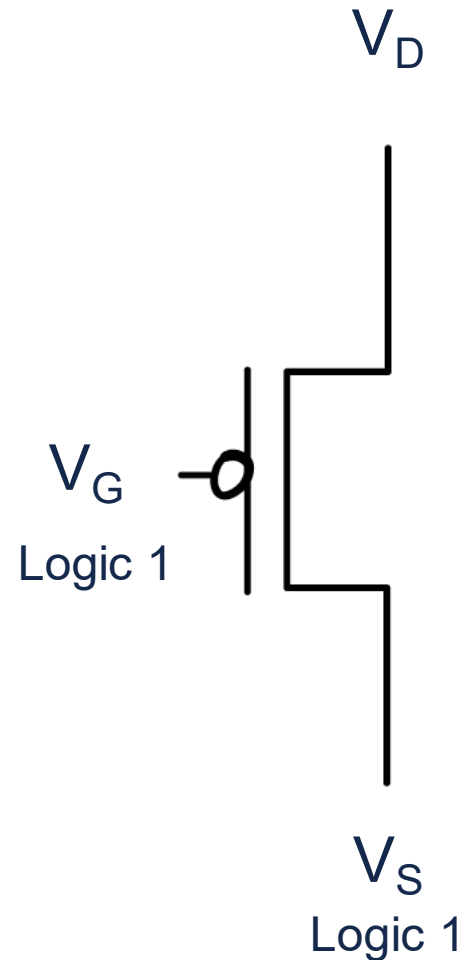
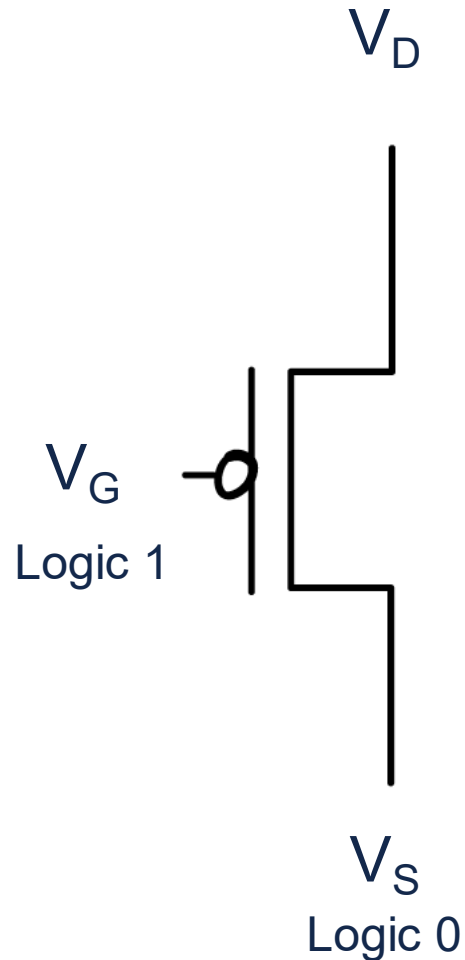
- If the gate voltage is **low (logic 0)**, current **can** flow between source and drain
 - This source and drain will be connected and will have the same voltage
- If the gate voltage is **high**, current **cannot** flow between source and drain
 - The source and drain are not connected



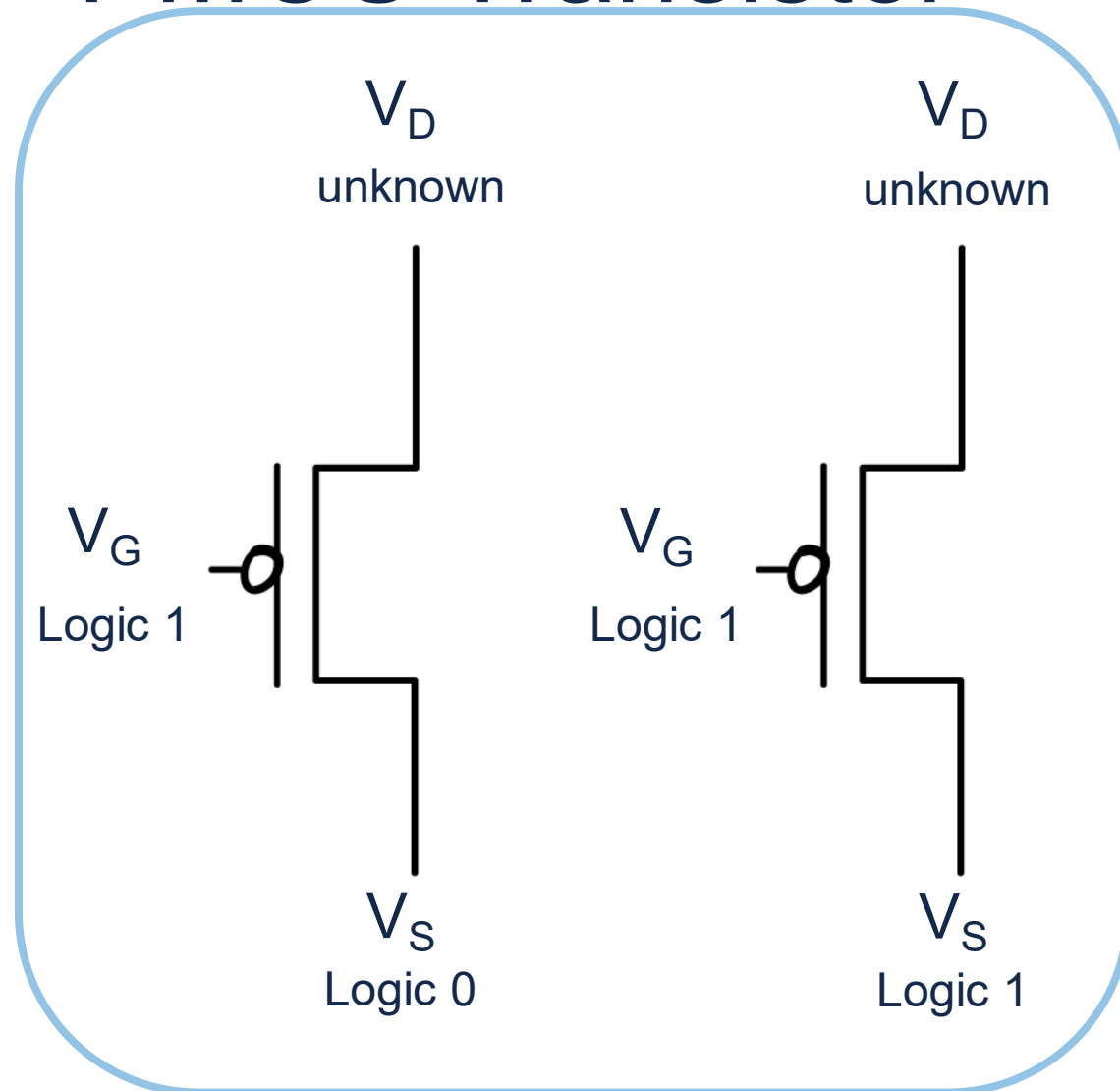
PMOS Transistor



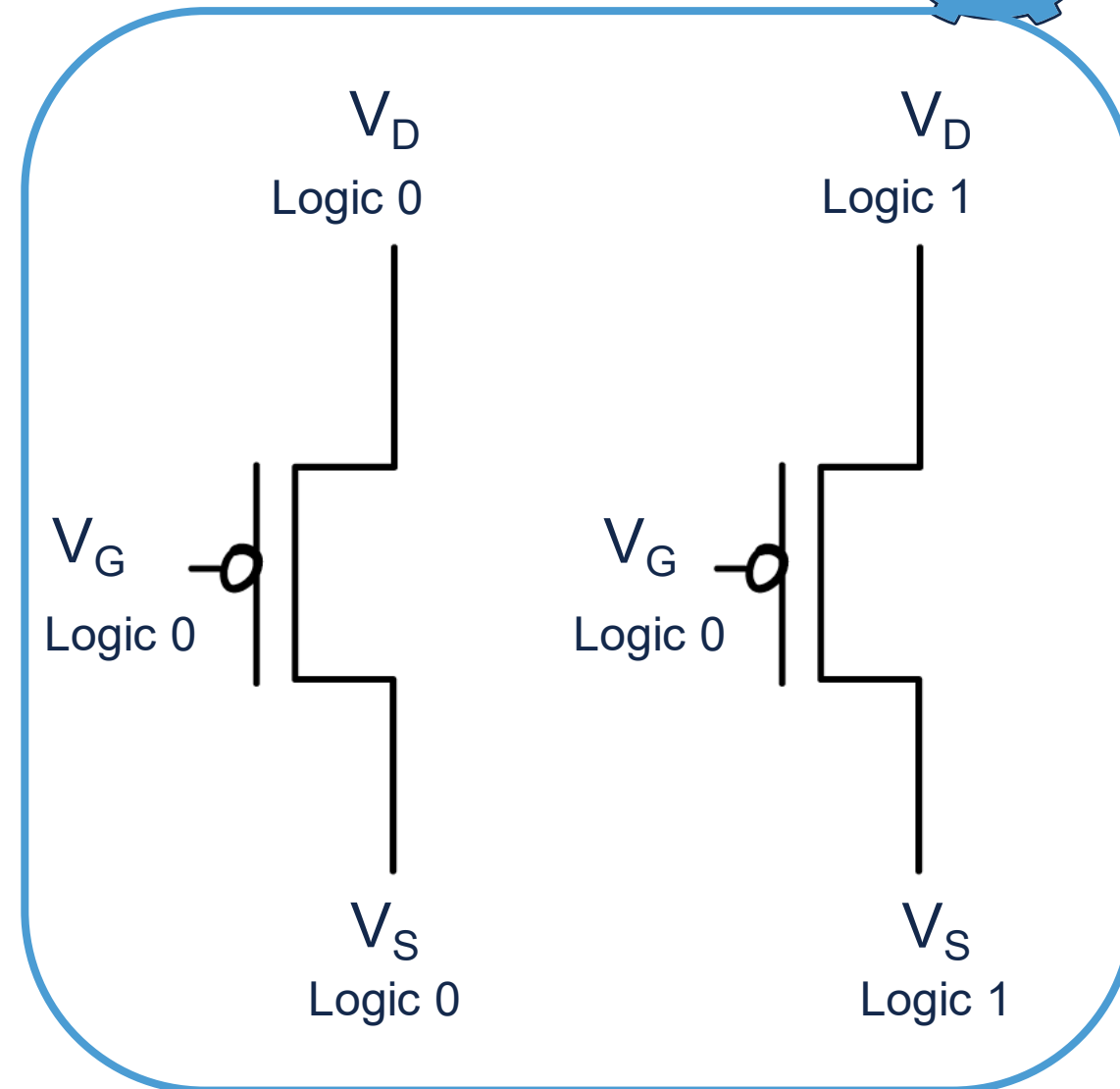
Let's say that we know the values of V_G and V_S . What is the value of V_D in each of the following diagrams?



PMOS Transistor



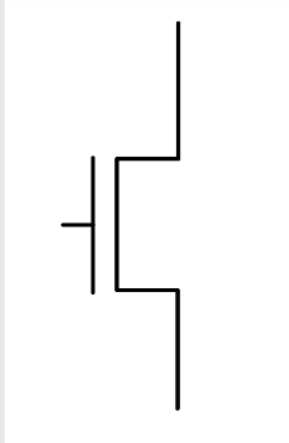
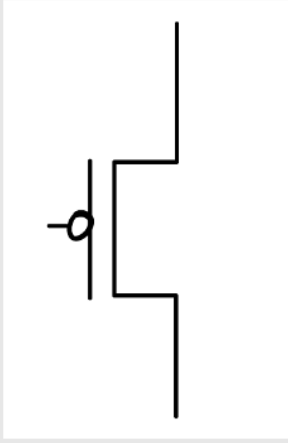
V_G is high, so source and drain are not connected



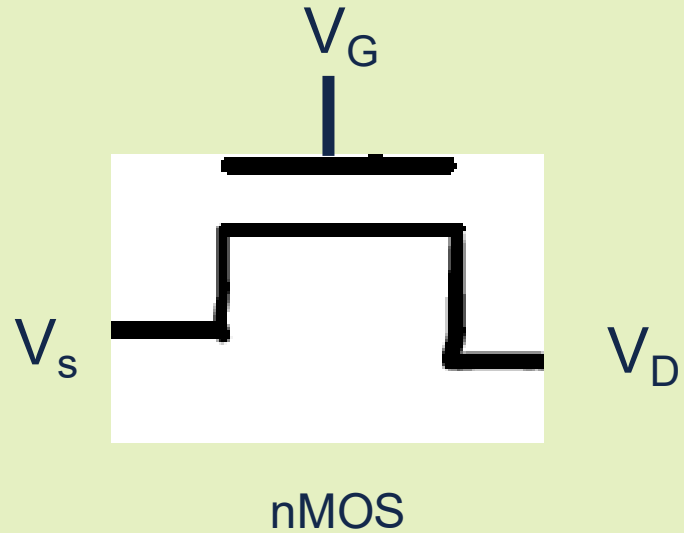
V_G is low, so source and drain are connected

Two Types of Transistors



	<u>nMOS</u>	<u>pMOS</u>
		
Behaves as an open switch when	V_g is low	V_g is high
Behaves as a closed switch when	V_g is high	V_g is low

nMOS Transistors



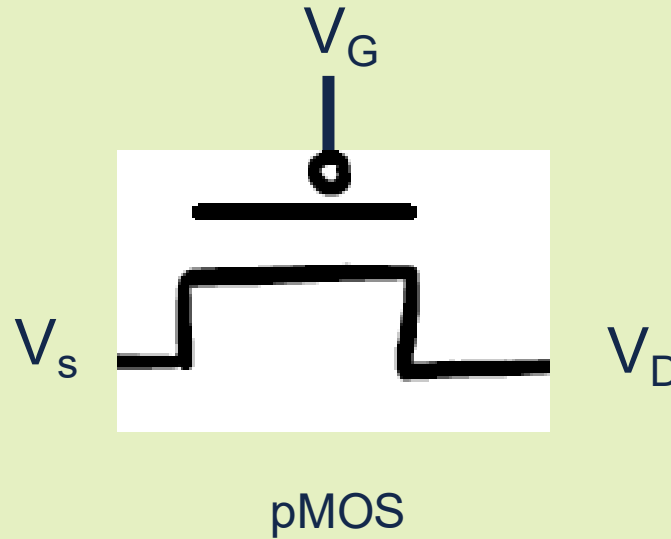
Acts as a(n) _____ (open/closed) switch when the gate voltage (V_g) is low.

Meaning that current _____ (can/cannot) flow between source and drain.

Acts as a(n) _____ (open/closed) switch when the gate voltage (V_g) is high.

Meaning that current _____ (can/cannot) flow between source and drain.

pMOS Transistors



Acts as a(n) _____ (open/closed) switch when the gate voltage (V_G) is low.

Meaning that current _____ (can/cannot) flow between source and drain.

Acts as a(n) _____ (open/closed) switch when the gate voltage (V_G) is high.

Meaning that current _____ (can/cannot) flow between source and drain.

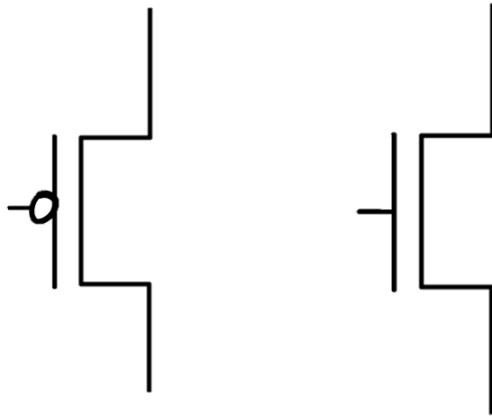
Building Logic Gates with Transistors

Abstraction 😊

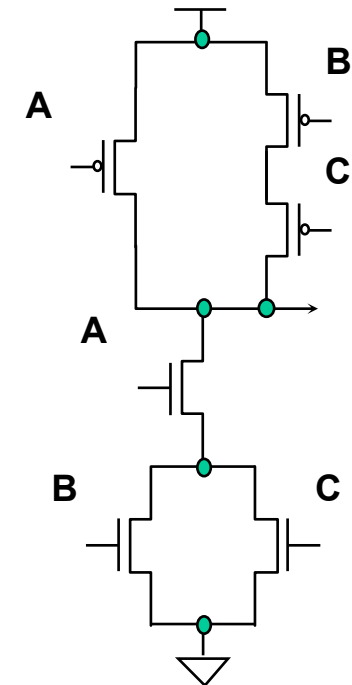
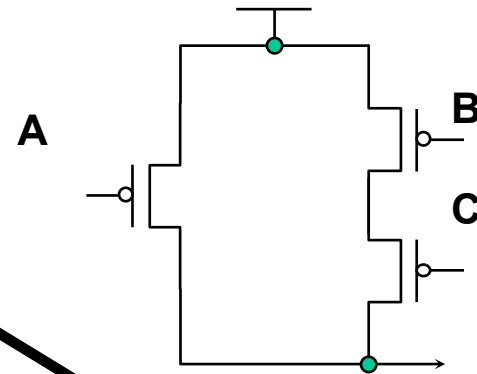


CMOS Gates

Physical Transistors



CMOS gates abstract away individual transistors, letting us design reliable logic at a higher level.



CMOS gates are the abstraction that lets many transistors behave like a single logical unit.

Let's Design an Inverter!



I want to design a circuit that takes as an input a one-bit value and outputs the complement of that number.

Symbol



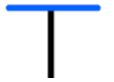
Truth Table

A	Y
0	1
1	0

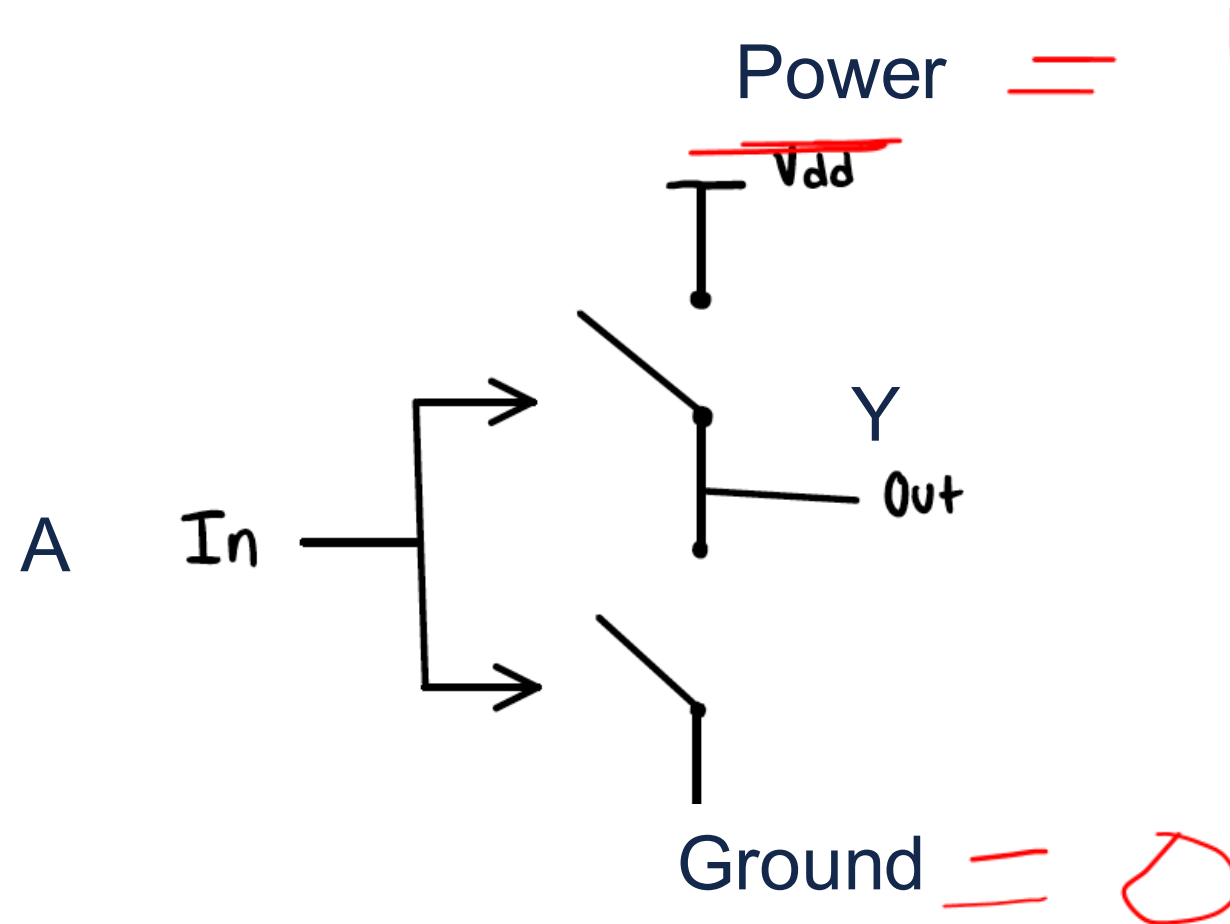
New Components



- Power
 - a component that produces a logic high value
- Ground
 - a component that produces a logic low value



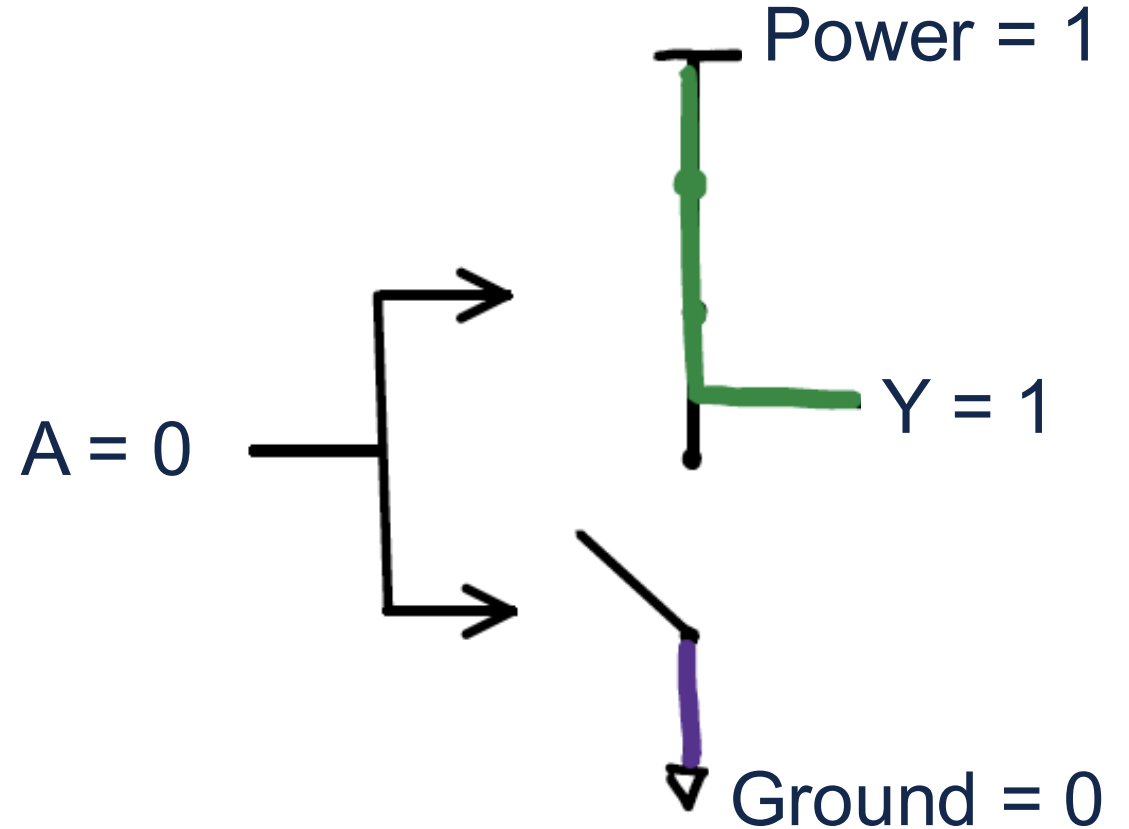
An Inverter with Switches



An Inverter with Switches



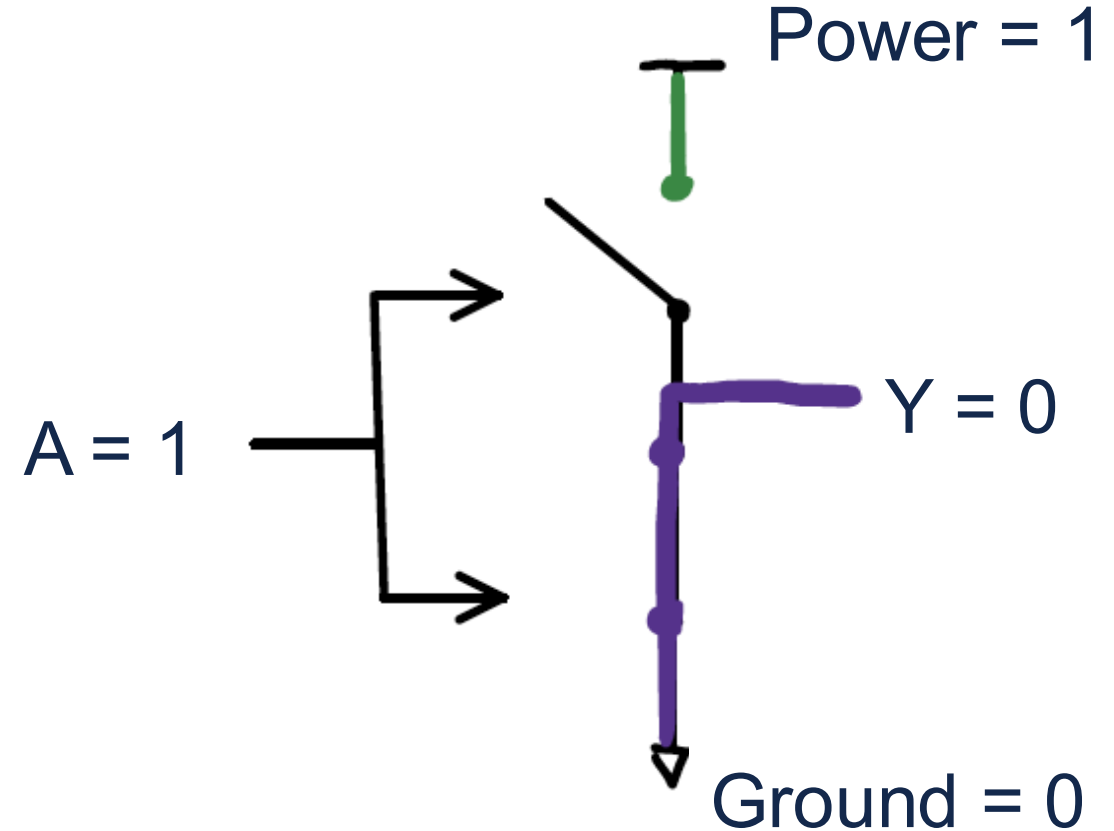
- When the input is 0
 - The top switch is closed.
 - The bottom switch is open.
 - This will connect power to the output which will make the output 1.



An Inverter with Switches



- When the input is 1
 - The bottom switch is closed.
 - The top switch is open.
 - This will connect ground to the output which will make the output 0.



Building Logic Gates with Transistors

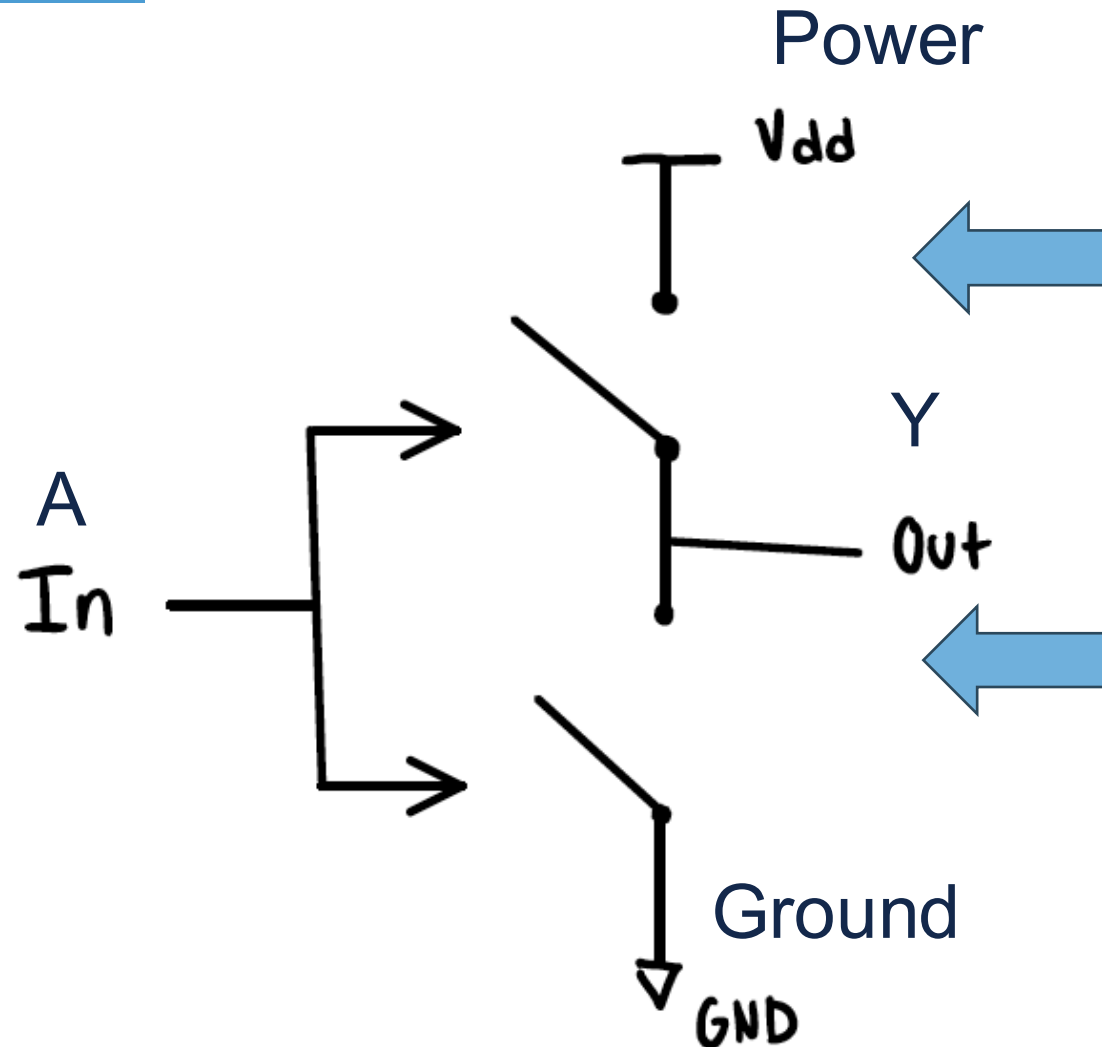


- Engineering Digital Behavior from Analog Devices
 - NMOS transistors act as controllable connections to ground
 - PMOS transistors act as controllable connections to power

CMOS gates work by carefully arranging these devices so physics is forced to produce clean digital logic!

Another way to say this: CMOS gates work by using NMOS transistors to pull outputs down and PMOS transistors to pull outputs up.

Building an Inverter with Transistors



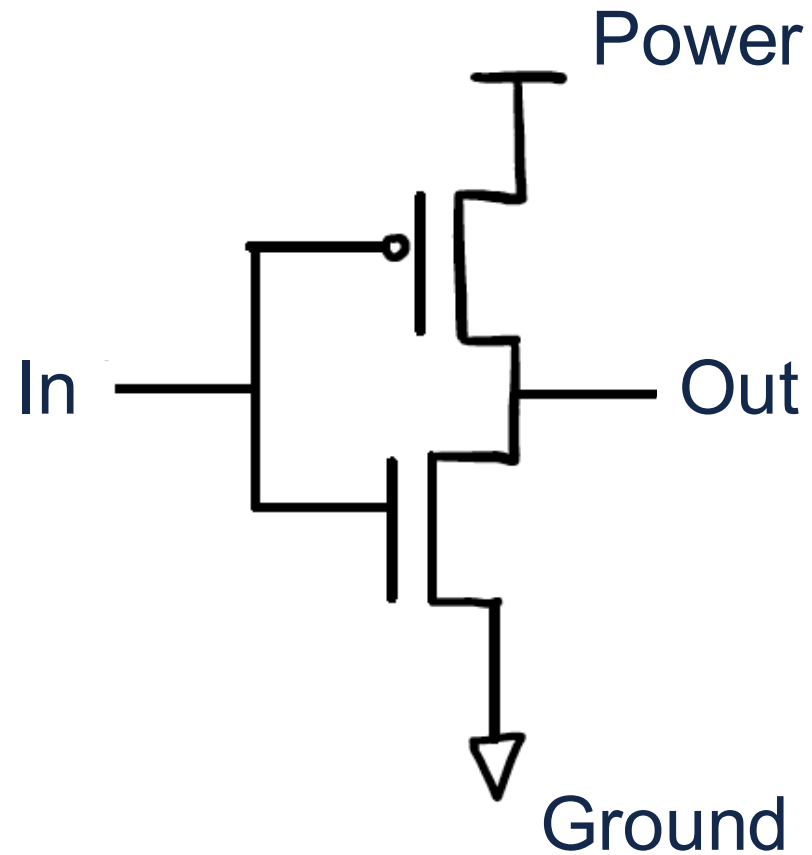
Should this switch be replaced by an nmos or pmos transistor?

Should this switch be replaced by an nmos or pmos transistor?

CMOS Inverter



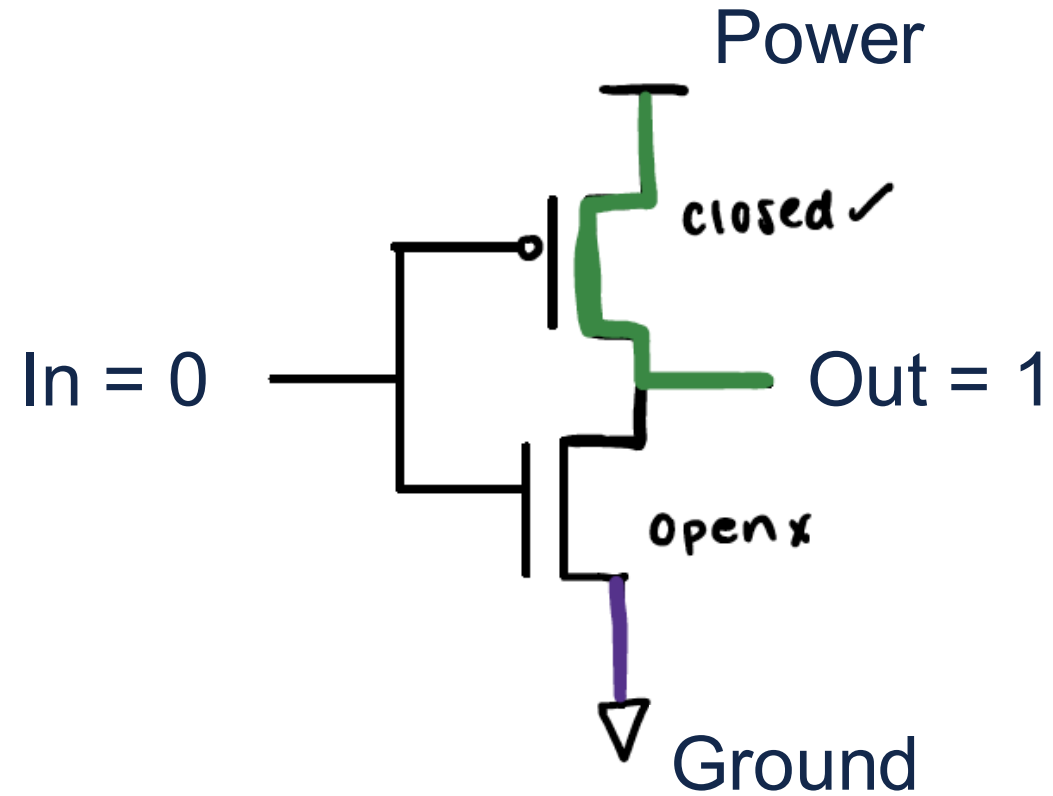
In	Out
0	1
1	0



CMOS Inverter



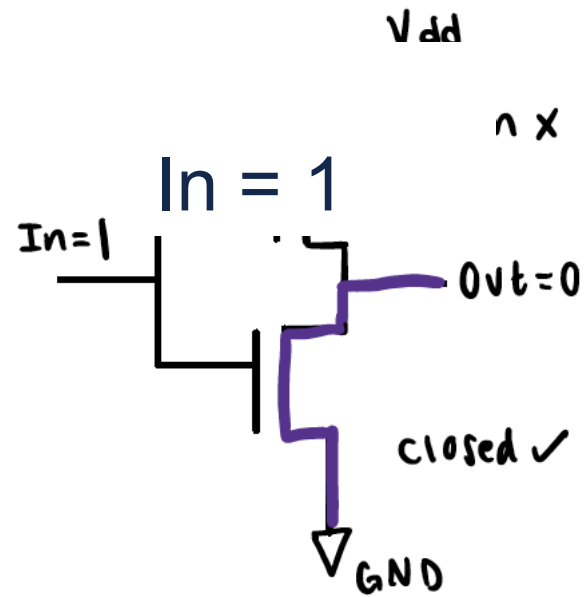
In	Out
0	1
1	0



CMOS Inverter



In	Out
0	1
1	0



Power

Out = 0

Ground

CMOS Gates



- CMOS = Complementary MOS
 - The pMOS and nMOS transistors complement each other to form the logic gate
- Common confusion
 - CMOS is not a type of transistor!
 - It is a technology that combines both nMOS and pMOS transistors

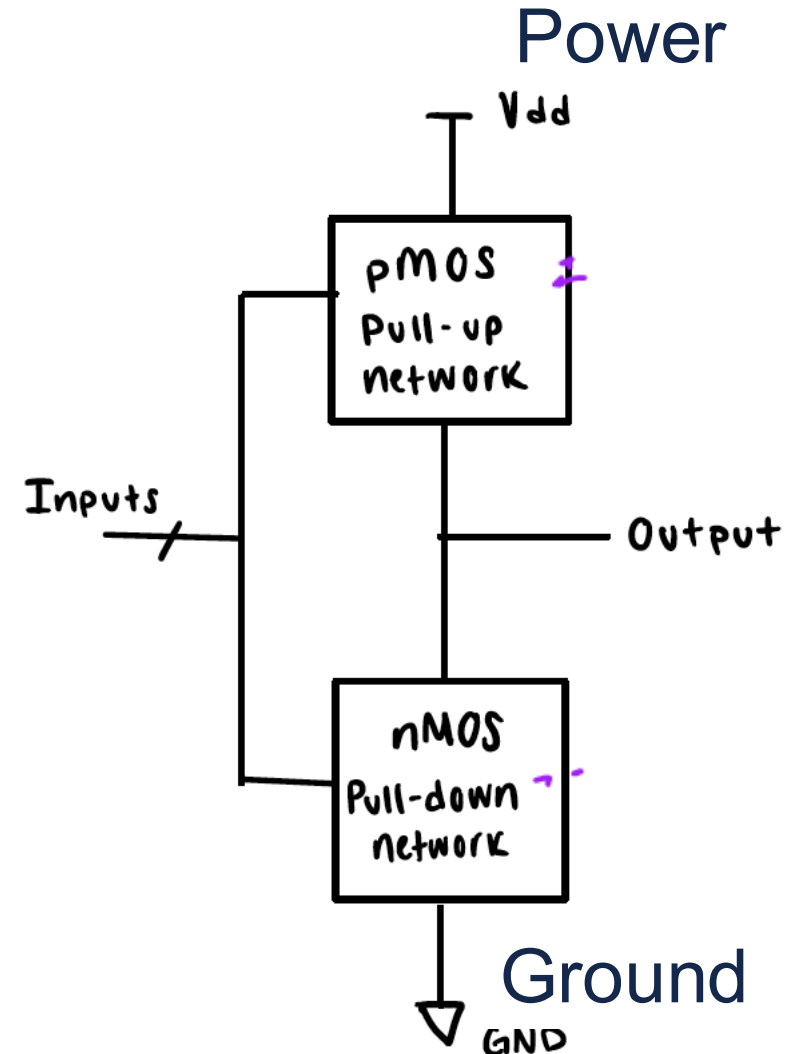
MOS: Metal Oxide
Semiconductor 😊

You might also see **MOSFET:**
Metal Oxide Semiconductor
Field-Effect Transistor

Pull-up and Pull-down Networks



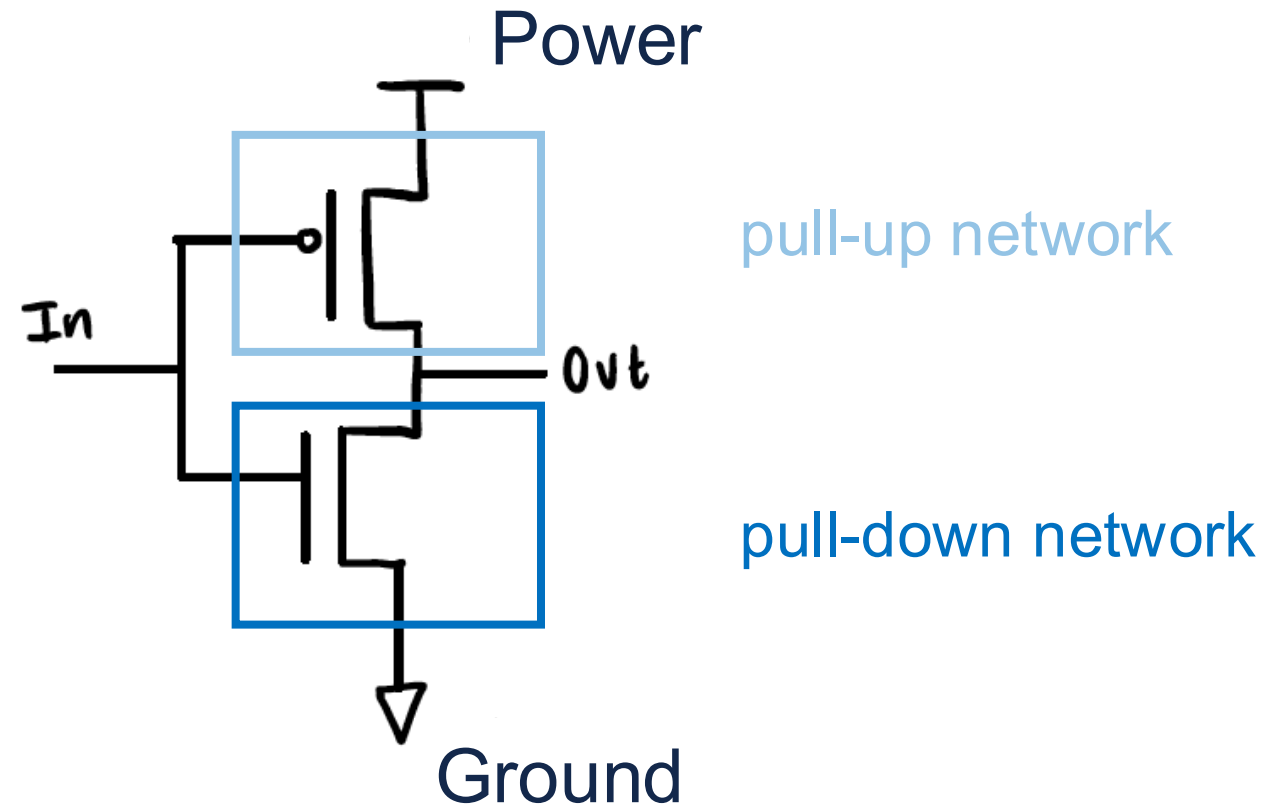
- Only one network will be on at a time
- The pull-up network drives the output when the output is 1
- The pull-down network drives the output when the output is 0



CMOS Inverter



In	Out
0	1
1	0



General CMOS Gate Recipe

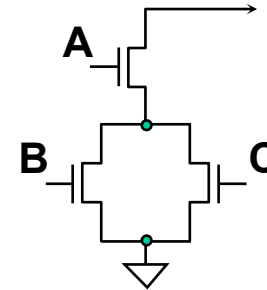


Step 1. Figure out pulldown network that does what you want (i.e the set of conditions where the output is '0')

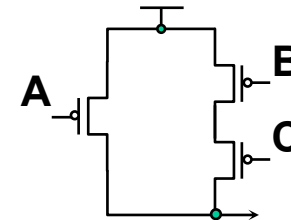
e.g., $F = A*(B+C)$

Step 2. Build the "opposite" version of that network to figure out when the output should be 1.

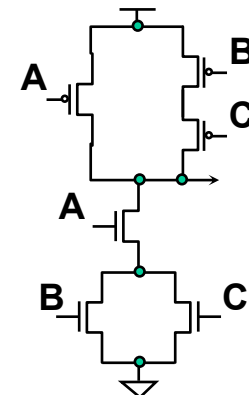
Step 3. Combine the two so you have a circuit that always pulls the output either up to 1 or down to 0.



Recall:
Series = *AND*
Parallel = *OR*



This comes from
DeMorgan's Laws! In
practice, we swap series
+ parallel connections,
use pMOS instead of
nMOS

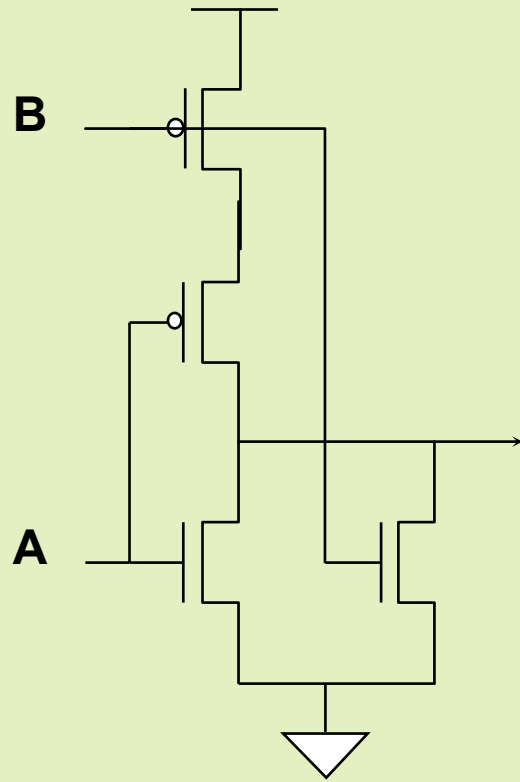


But isn't it
hard to wire
it all up? \



Yes it is!
This does
not
scale...

What function does this gate compute?



A	B	C
0	0	
0	1	
1	0	
1	1	