COMP311:
COMPUTER ORGANIZATION!

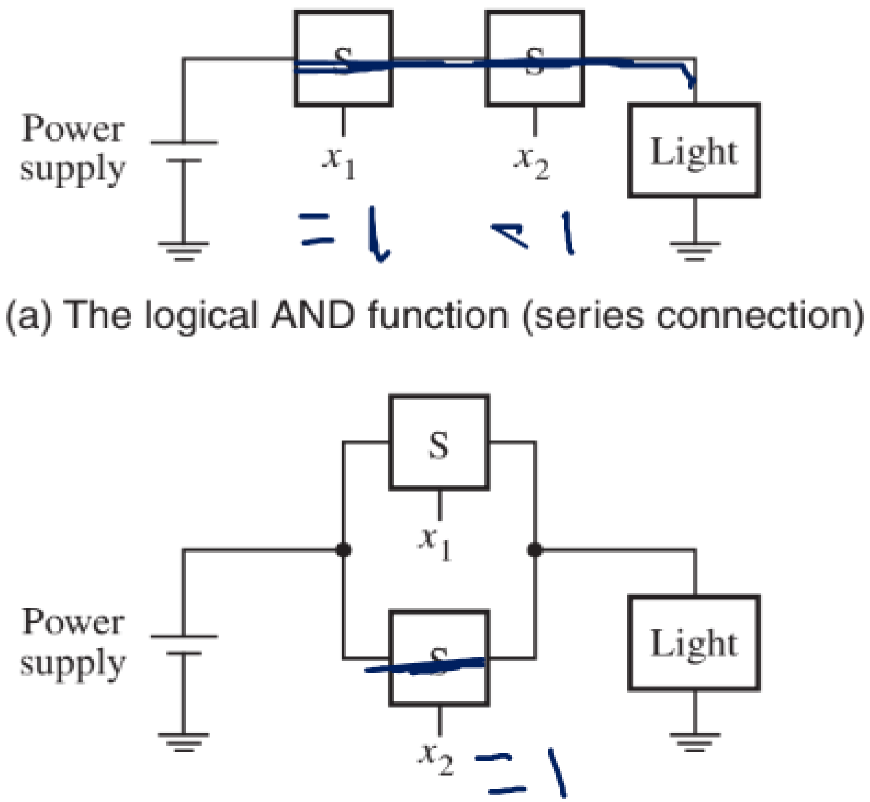
Lecture 4: Transistors, Logic Gates

tinyurl.com/comp311-fa25

Functions with Two Switches

AND

- Two switches can be connected either in ***series*** or in ***parallel***
- Using a **series** connection, the light will be turned on only if **both switches are closed.** If either switch is open, the light will be off.
  - $L(x1, x2) = x1 \cdot x2$ *where L = 1 if x1 = 1 and x2 = 1, L = 0 otherwise.*
- · is logical AND



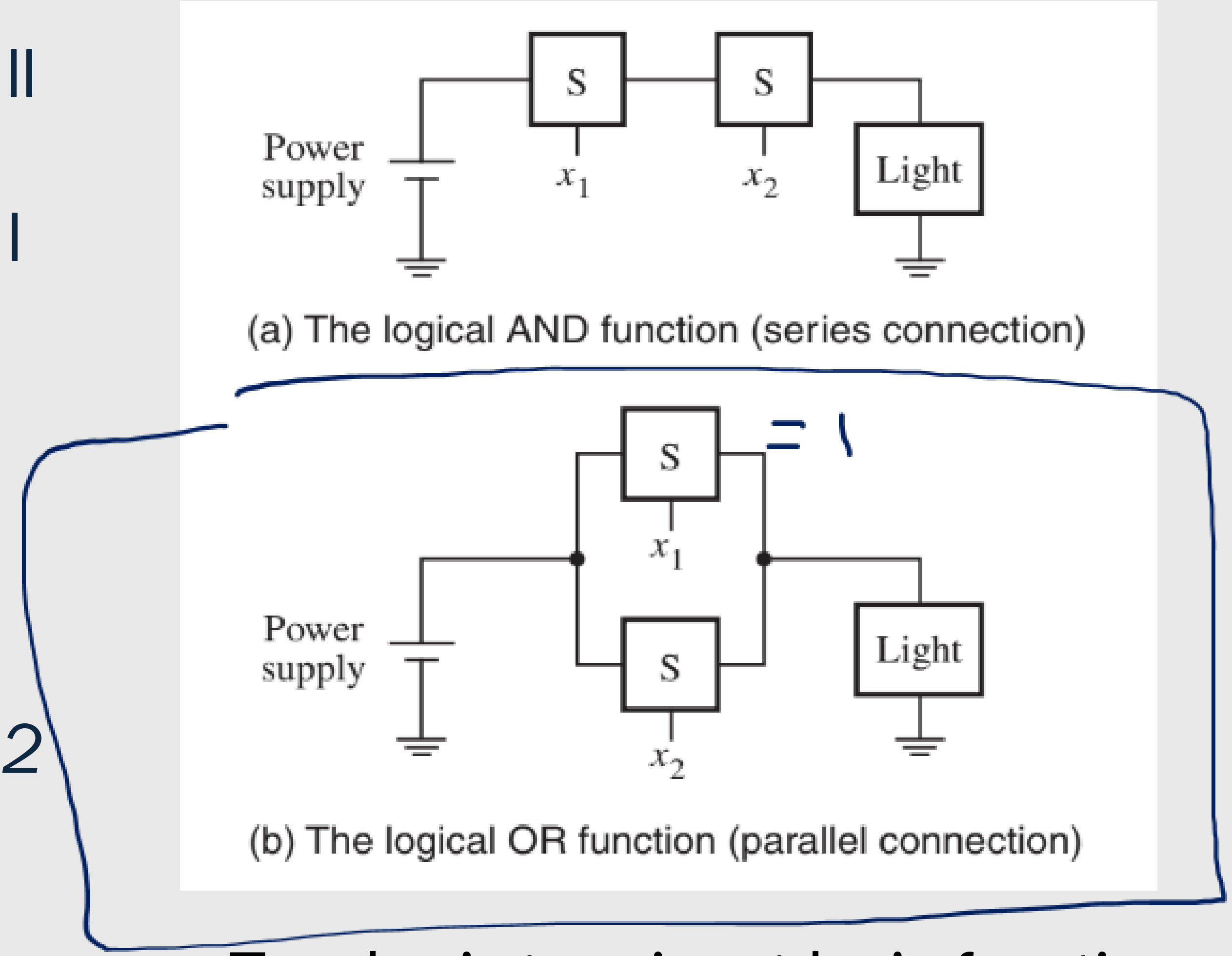
(a) The logical AND function (series connection)

(b) The logical OR function (parallel connection)

Two basic two-input logic functions

Functions with Two Switches

- With a **parallel** connection of the two switches, the light will be on if **either the x1 or x2 switch is closed.** The light will also be on if both switches are closed. **The light will be off only if both switches are open.**
  - *L(x1, x2) = x1 + x2*
    where L = 1 if x1 = 1 or x2 = 1 or if x1 = x2 = 1,
    L = 0 if x1 = x2 = 0.
- ”+” is logical OR



(a) The logical AND function (series connection)

(b) The logical OR function (parallel connection)

Two basic two-input logic functions

A Three-Switch Function

- Three switches can be used to control the light in more complex ways. This series-parallel connection of switches realizes the logic function:
  - $L(x1, x2, x3) = \underbrace{(x1 \underline{+} x2)}\,\underline{\cdot}\, x3$
- The light is on if x3 = 1 and, at the same time, at least one of the x1 or x2 are 1.

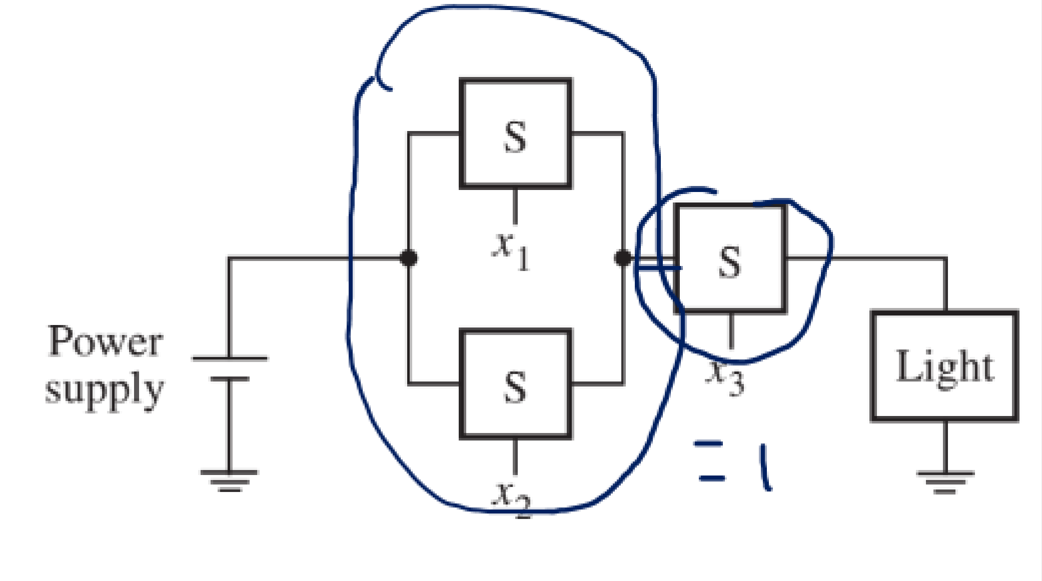
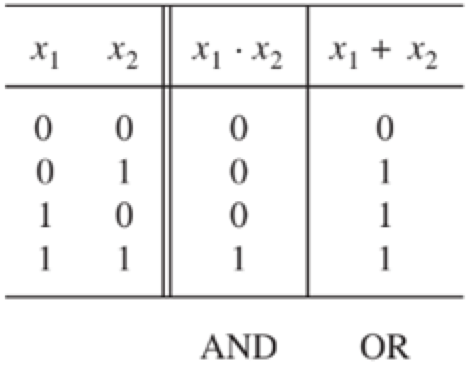


Figure 2.4 A series-parallel connection.

Truth Tables

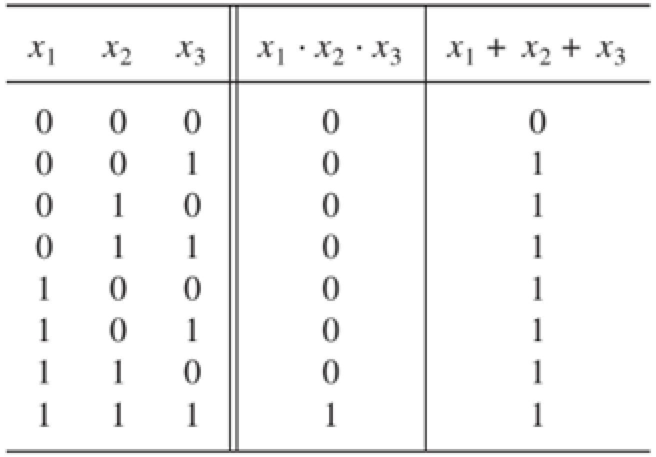
- Every logical function over a finite set of inputs can be described in a tabular fashion that enumerates all possible input combinations and provides the associated output.

- Functions expressed this way are called ***truth tables***.



x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1
		AND	OR

Figure 2.6 A truth table for the AND and OR operations.



x_1	x_2	x_3	$x_1 \cdot x_2 \cdot x_3$	$x_1 + x_2 + x_3$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

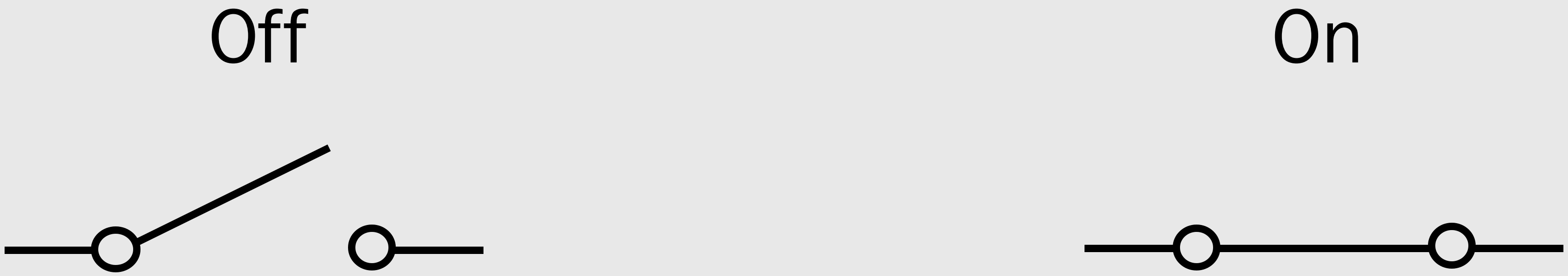
Figure 2.7 Three-input AND and OR operations.

Pondering Truth Tables

WS 2 Q5

TRANSISTORS

Transistors are like switches



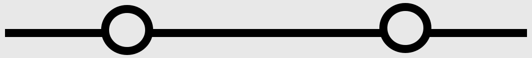
Switches are controlled by physical contact

Transistors are controlled by a voltage

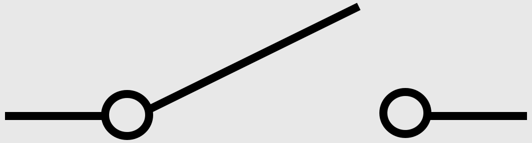
Two States: Binary!

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

On



Off



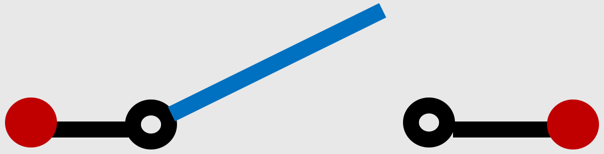
Logic Levels

- 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

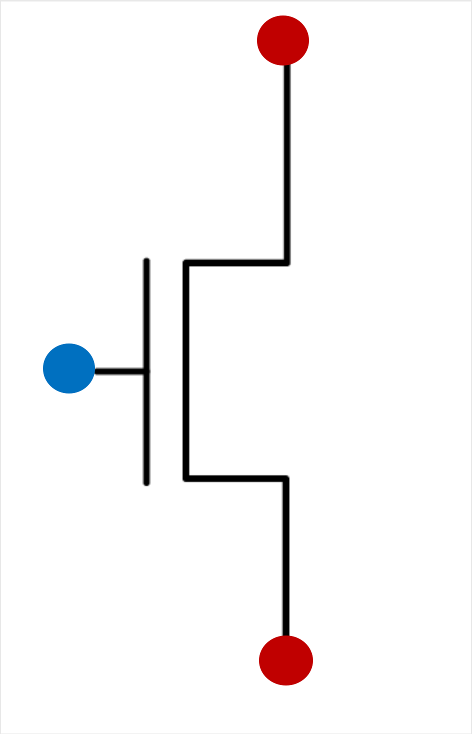
Transistors

- 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

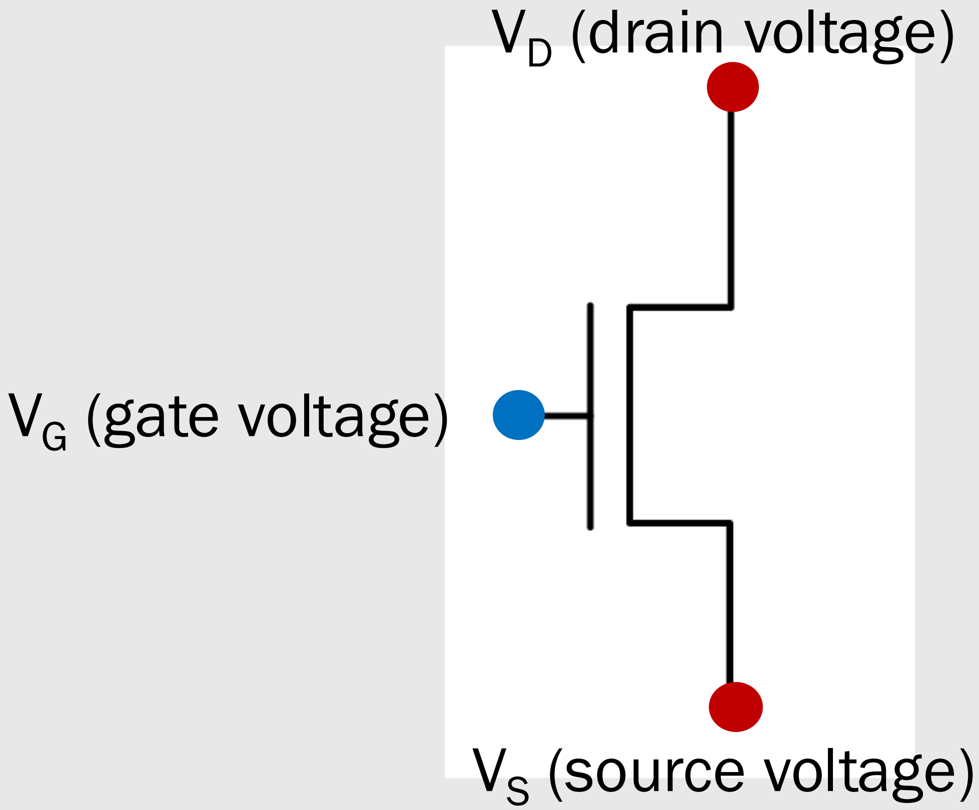


Transistor
(n-channel MOSFET)

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

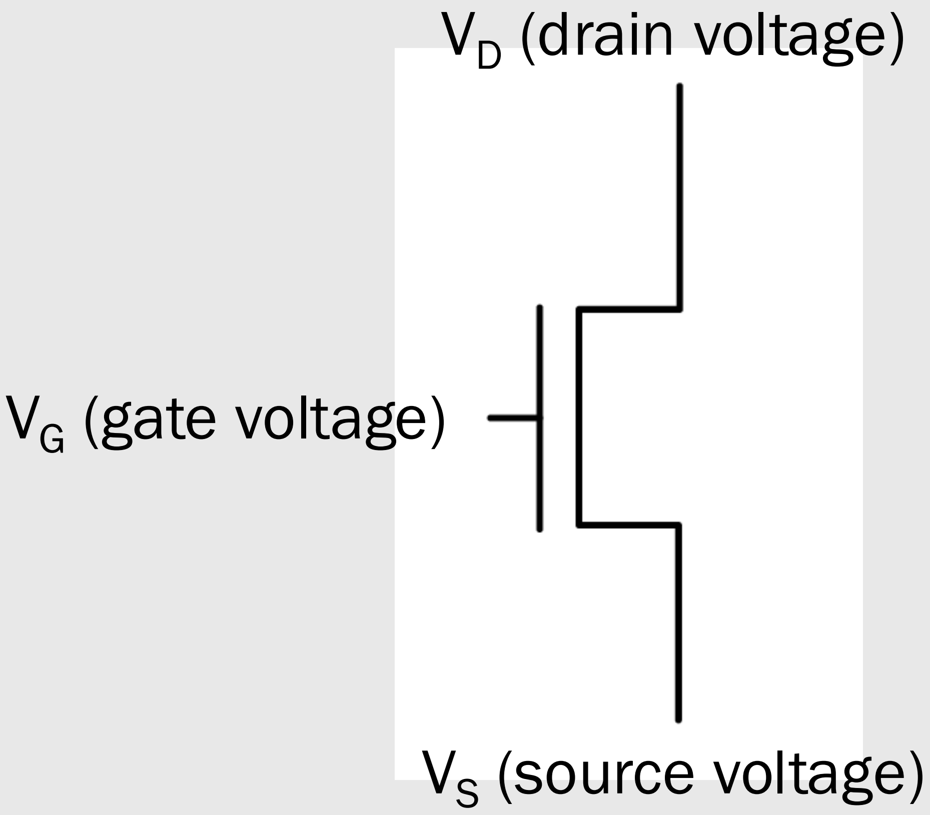
NMOS Transistor

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



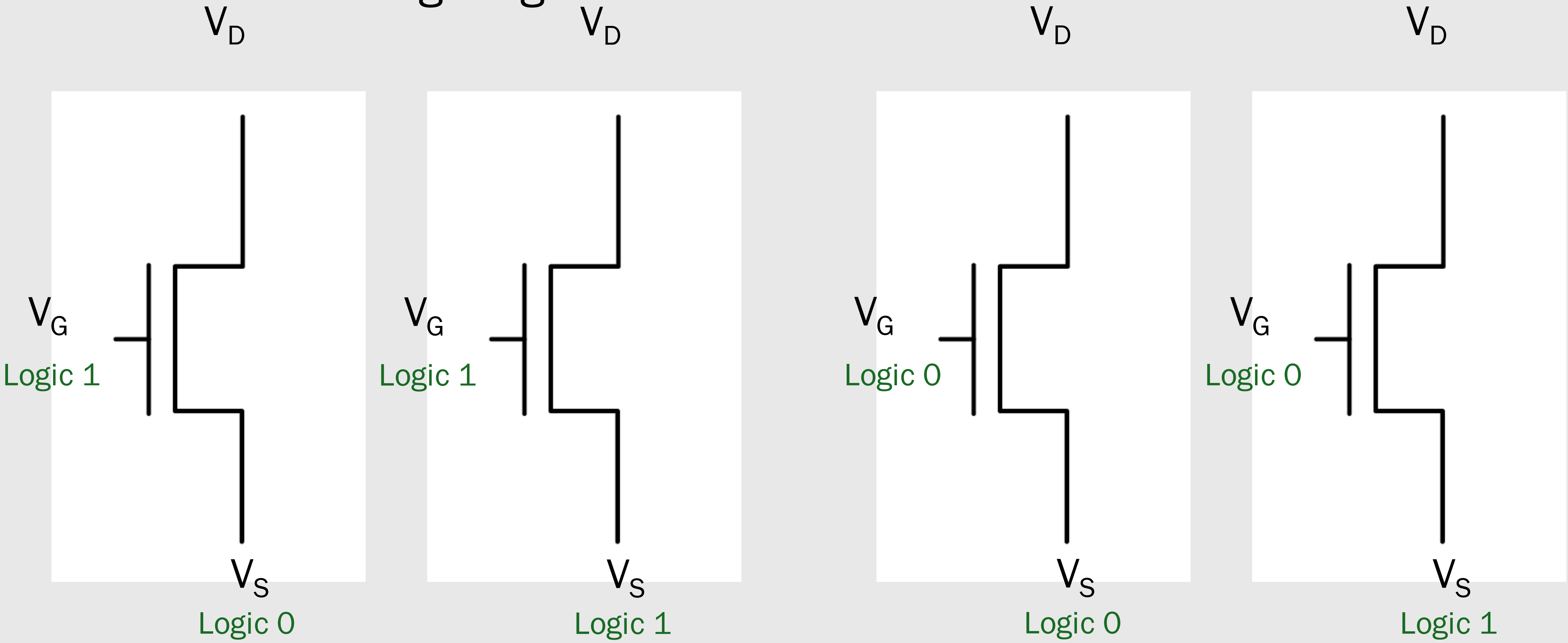
NMOS Transistor

- 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*

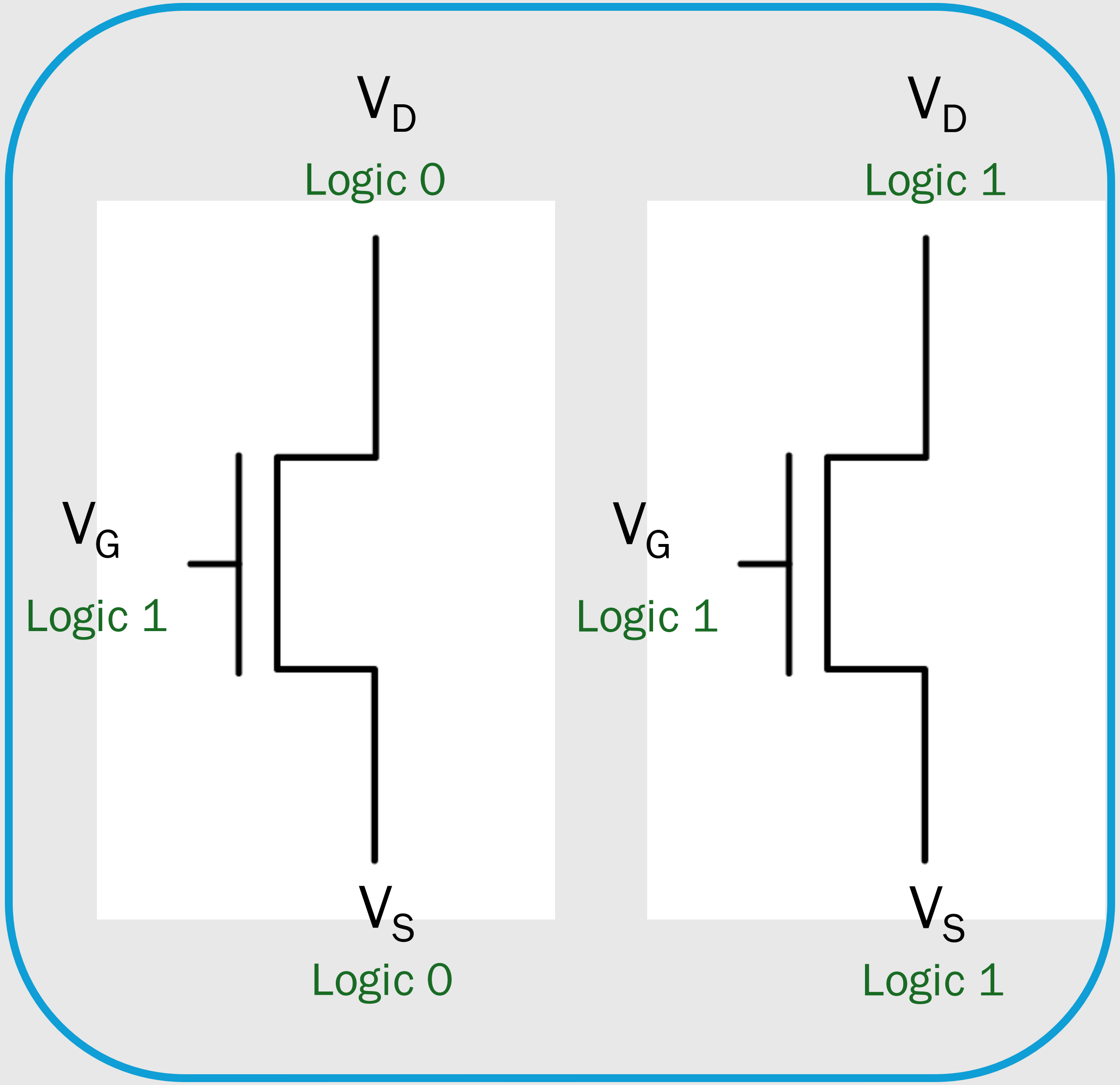


NMOS Transistor

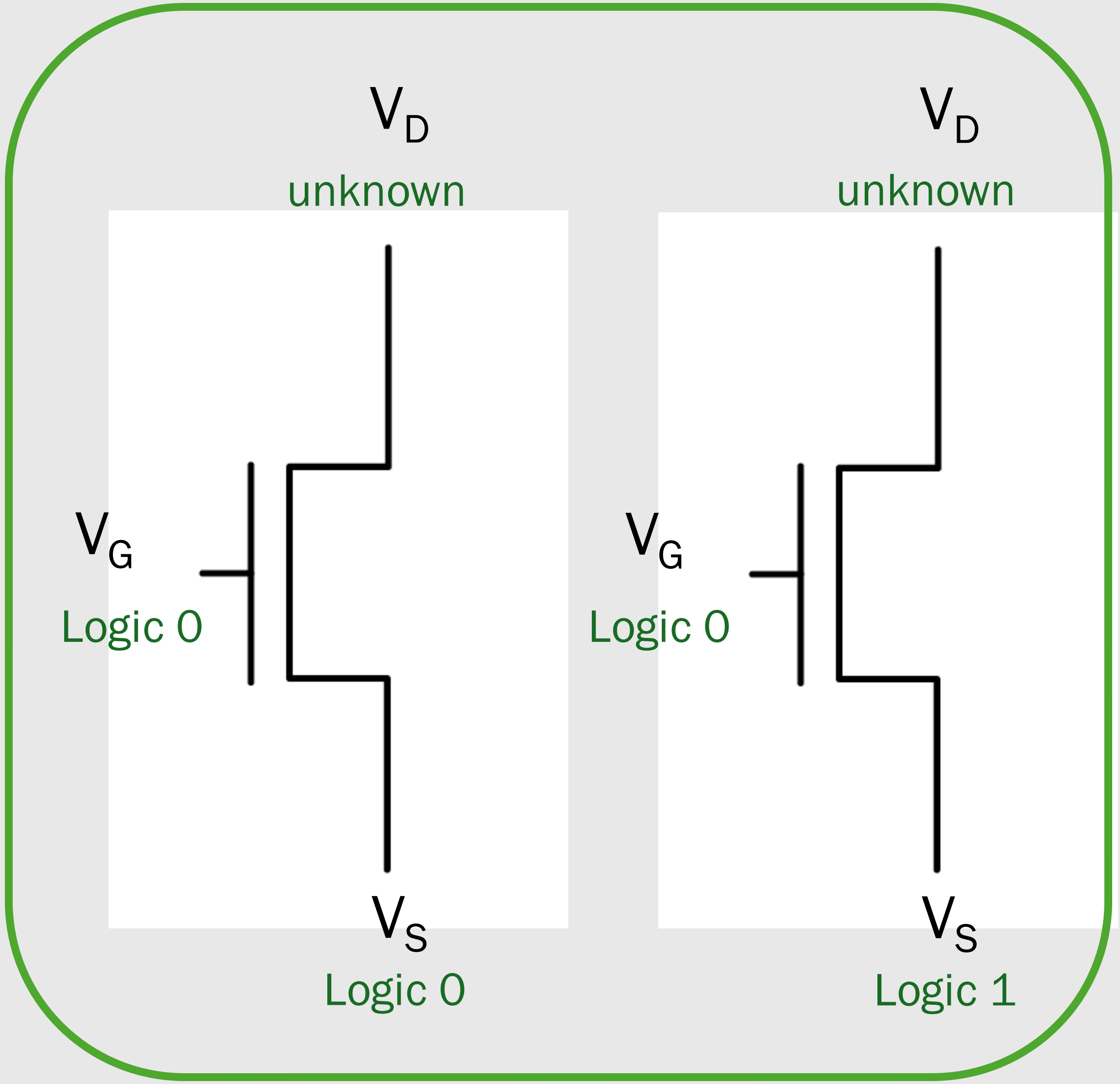
Let’s say that we know the values of V_G and V_S. What is the value of V_D in each of the following diagrams?



NMOS Transistor



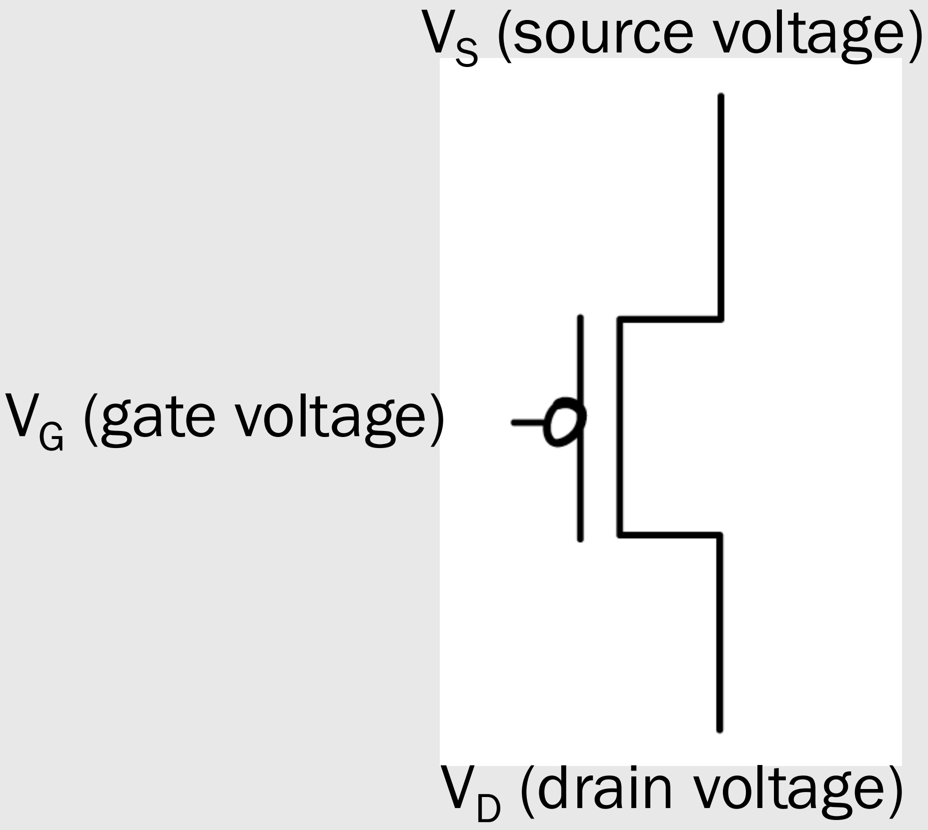
V_G is high, so source and drain are connected



V_G is low, so source and drain are not connected

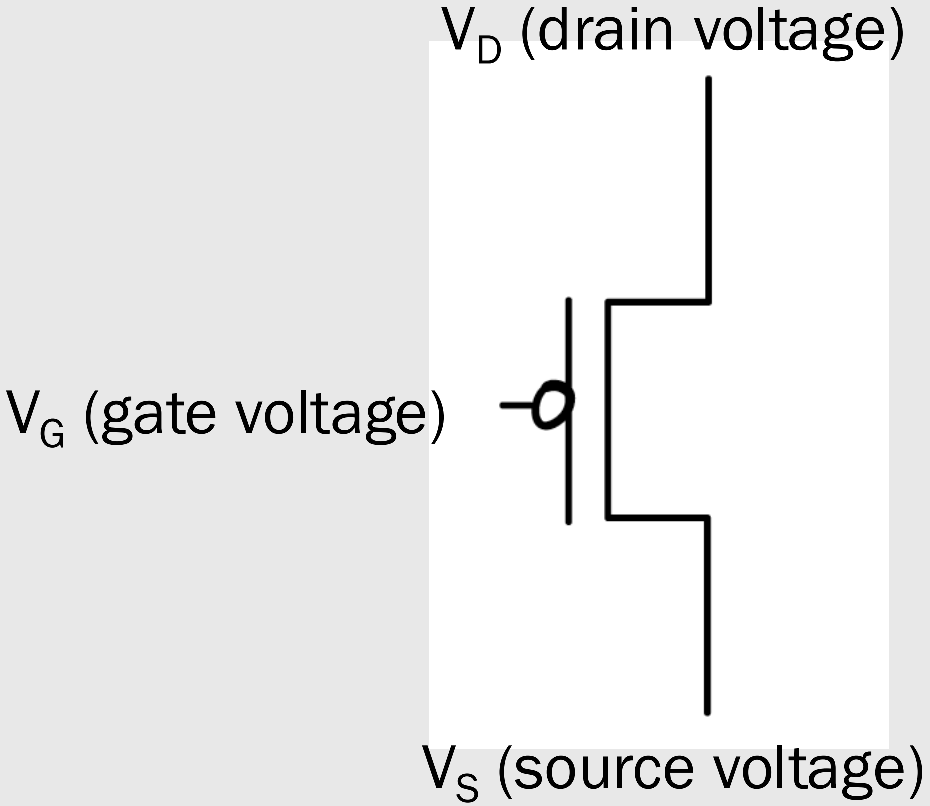
PMOS Transistor

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



PMOS Transistor

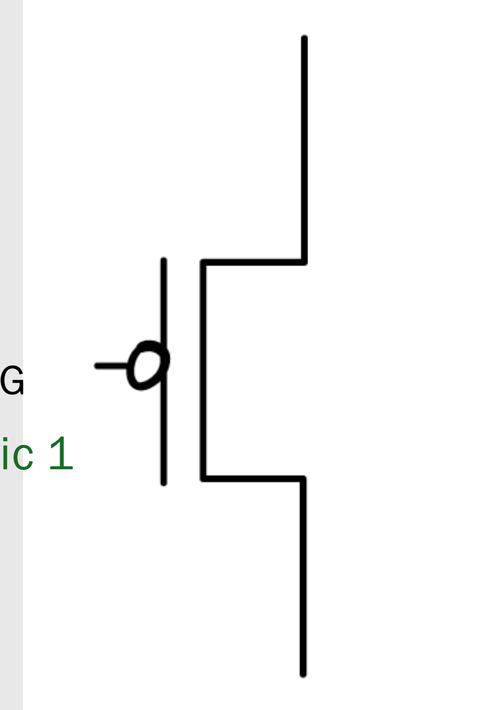
- If the gate voltage is low (logic 0), current can flow between source and drain
  - *This source and drain will be connected and will have the same voltage*
- If the gate voltage is high, current cannot flow between source and drain
  - *The source and drain are not connected*



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?

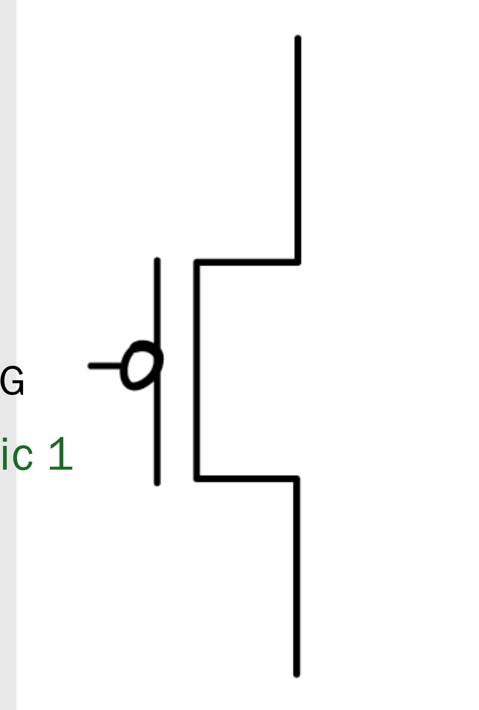
V_D



V_G
Logic 1

V_S
Logic 0

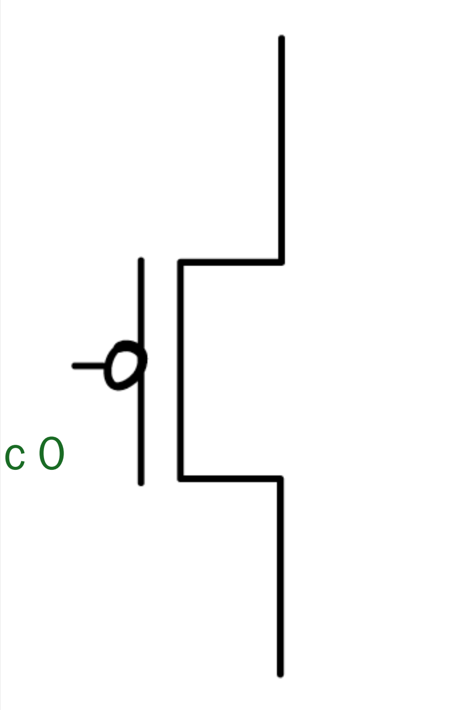
V_D



V_G
Logic 1

V_S
Logic 1

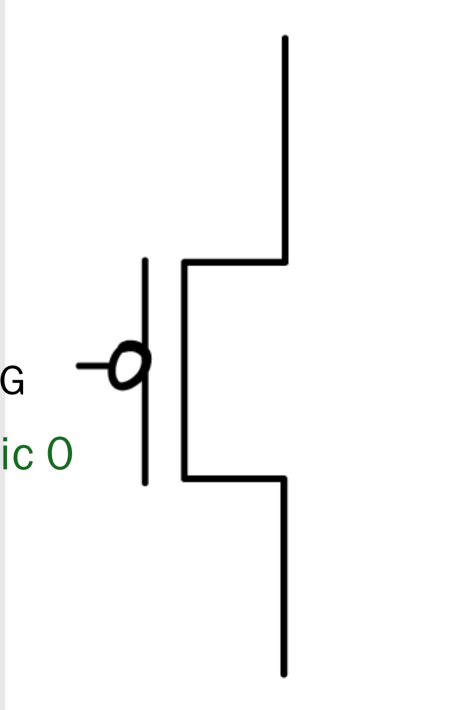
V_D



V_G
Logic 0

V_S
Logic 0

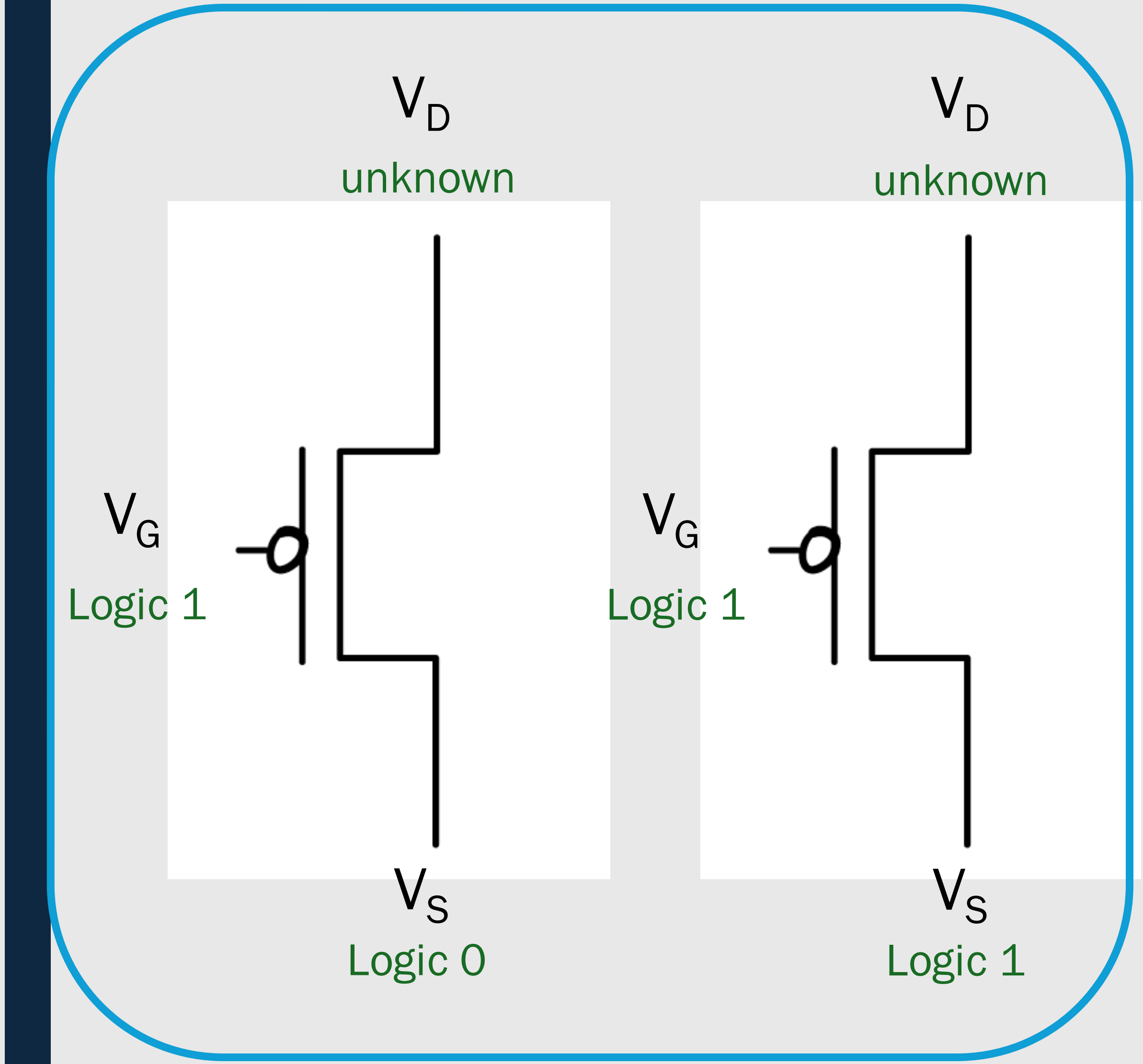
V_D



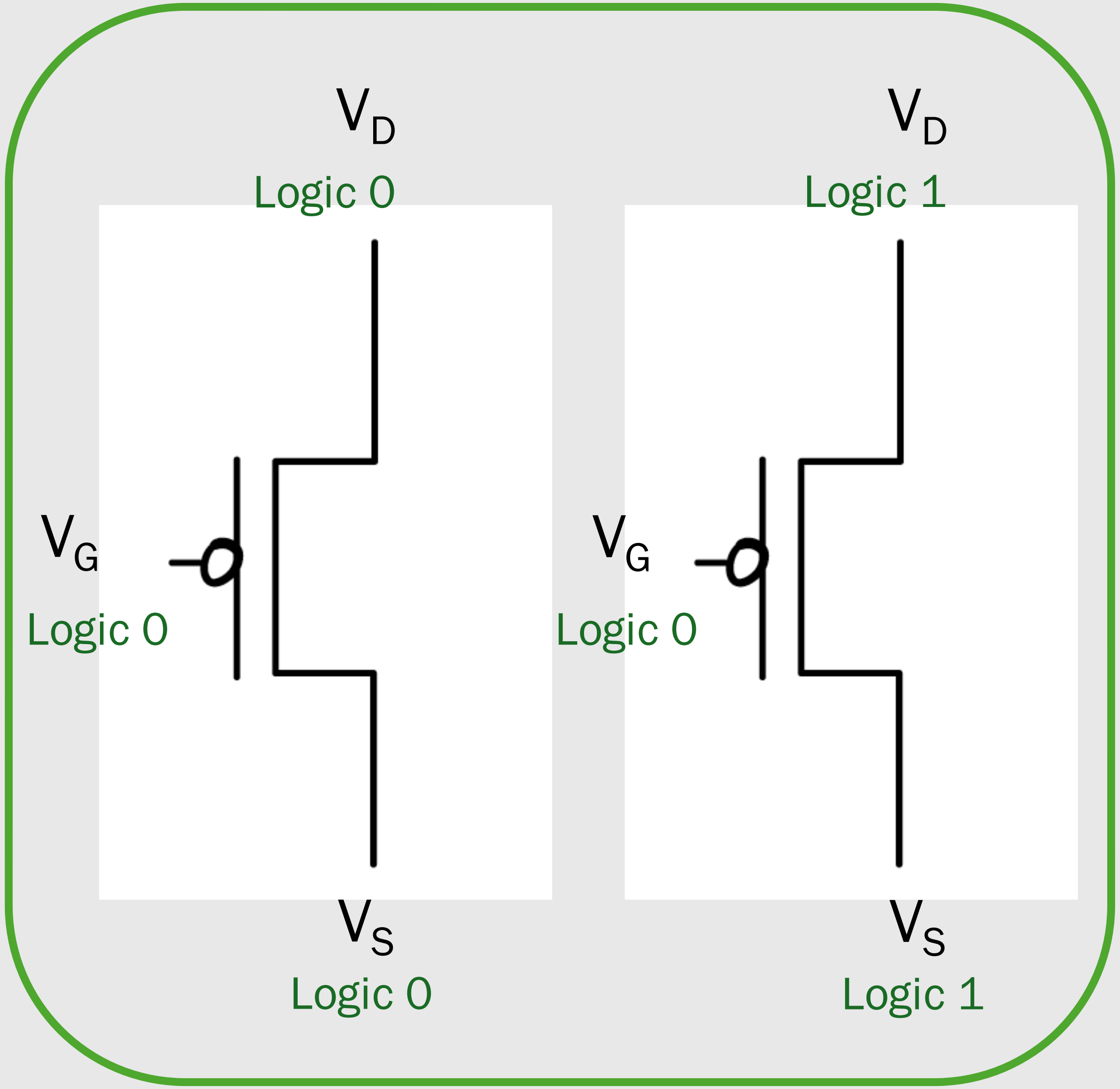
V_G
Logic 0

V_S
Logic 1

PMOS Transistor

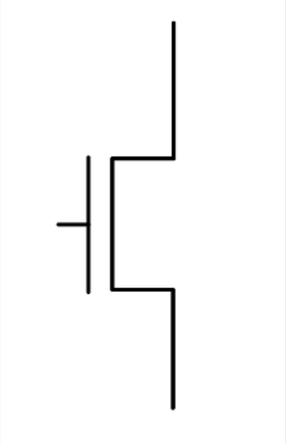
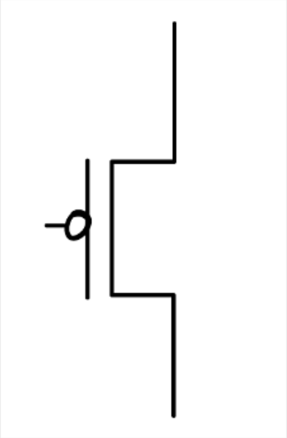


V_G is high, so source and drain are not connected



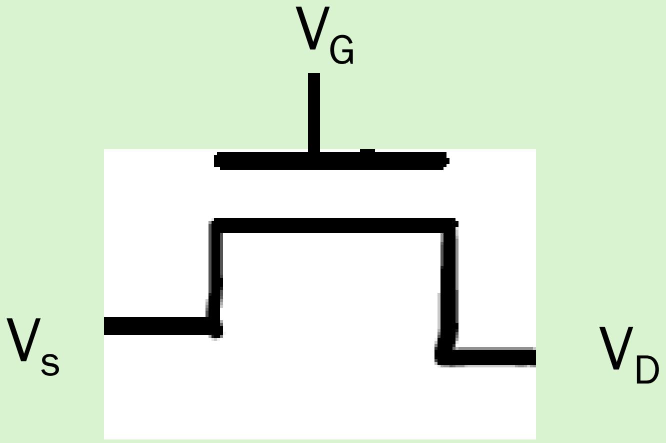
V_G is low, so source and drain are connected

Two Types of Transistors

	nMOS	pMOS
		
Behaves as an open switch when	Vg is low	Vg is high
Behaves as a closed switch when	Vg is high	Vg is low

nMOS Transistors

Q1



nMOS

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

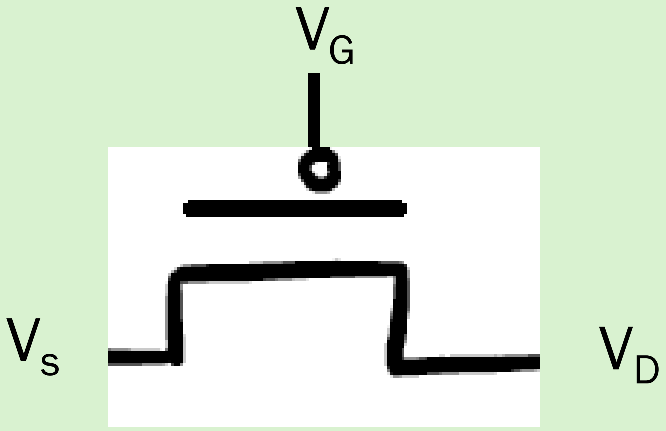
Meaning that current _________ (can/cannot) flow between source and drain.

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

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

pMOS Transistors

Q2



pMOS

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

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

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

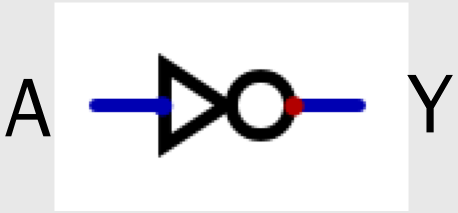
Meaning that current _______ (can/cannot) flow between source and drain.

BUILDING LOGIC GATES WITH TRANSISTORS

Designing 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

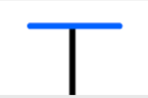


Truth Table

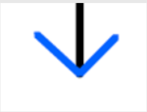
A	Y
0	1
1	0

New Components

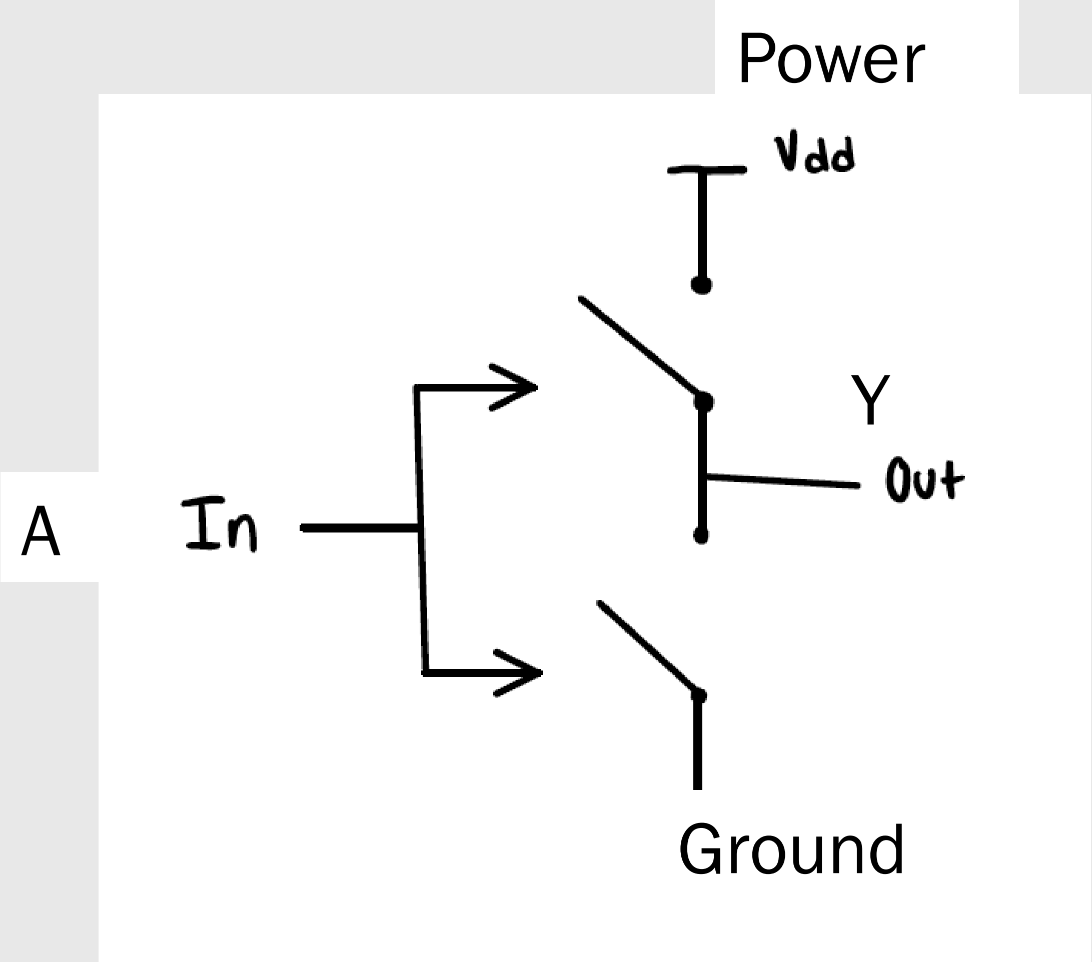
- Power
  - *a component that produces a logic high value*



- Ground
  - *a component that produces a logic low value*

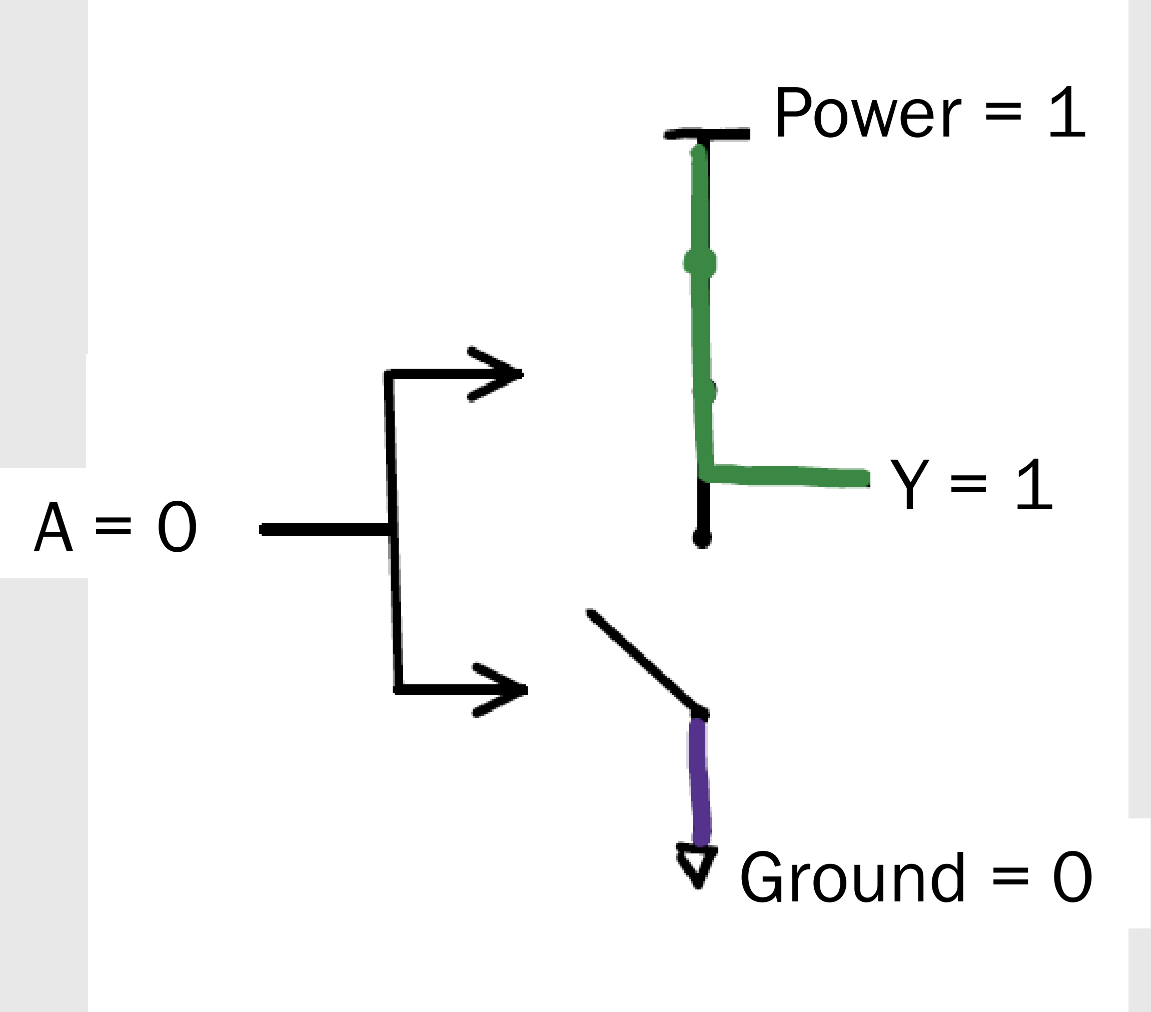


An Inverter with Switches



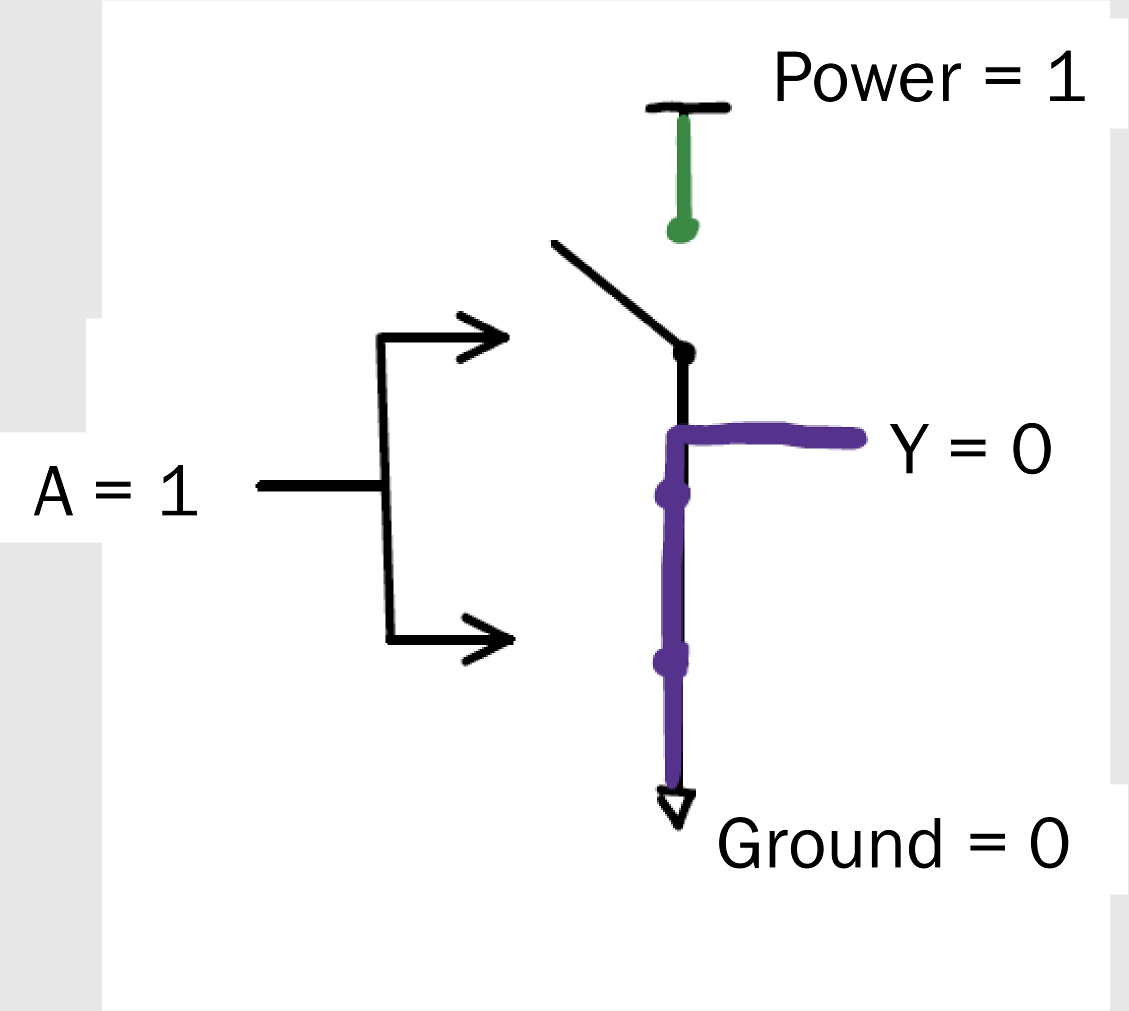
An Inverter with Switches

- 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

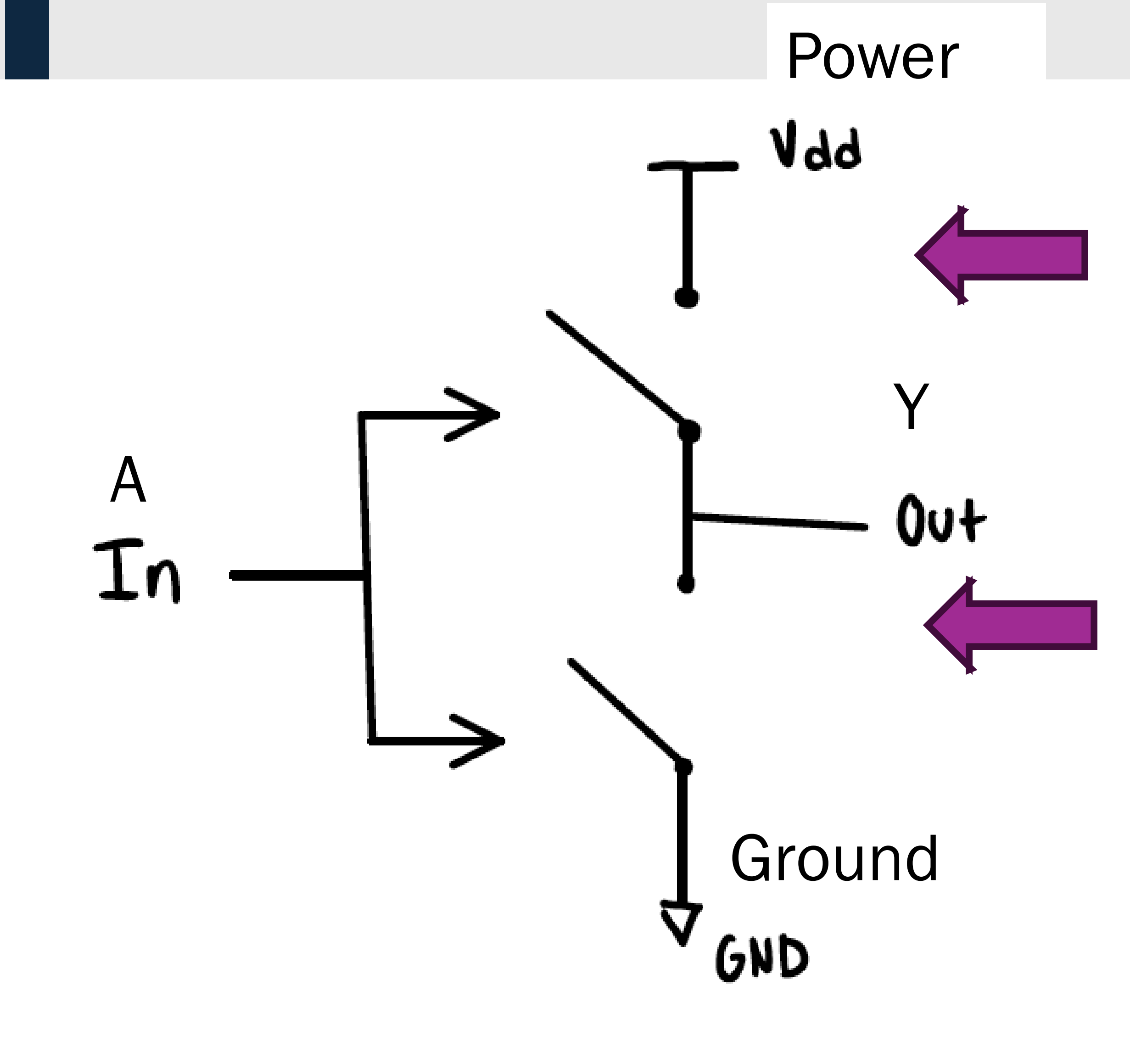
- When the input is 1
  - *The bottom switch is closed.*
  - *The top switch is open.*
  - *This will connect ground to the output which will make the output 0.*



Building Logic Gates with Transistors

- Due to the physical characteristics of the transistors (which we don’t have the physics background to get into in this class)…
  - *We use nmos transistors to connect the output to ground*
  - *We use pmos transistors to connect the output to power*

Building an Inverter with Transistors

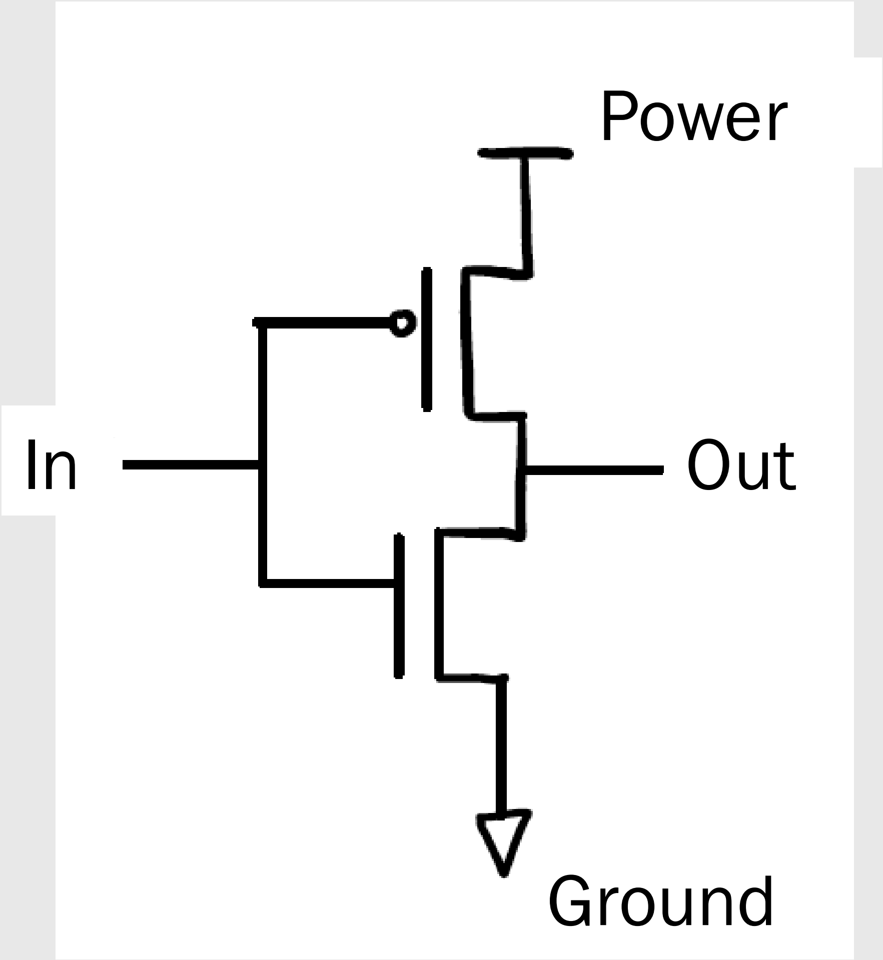


Should this switch be replaced by an nmos or pmos transistor?

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

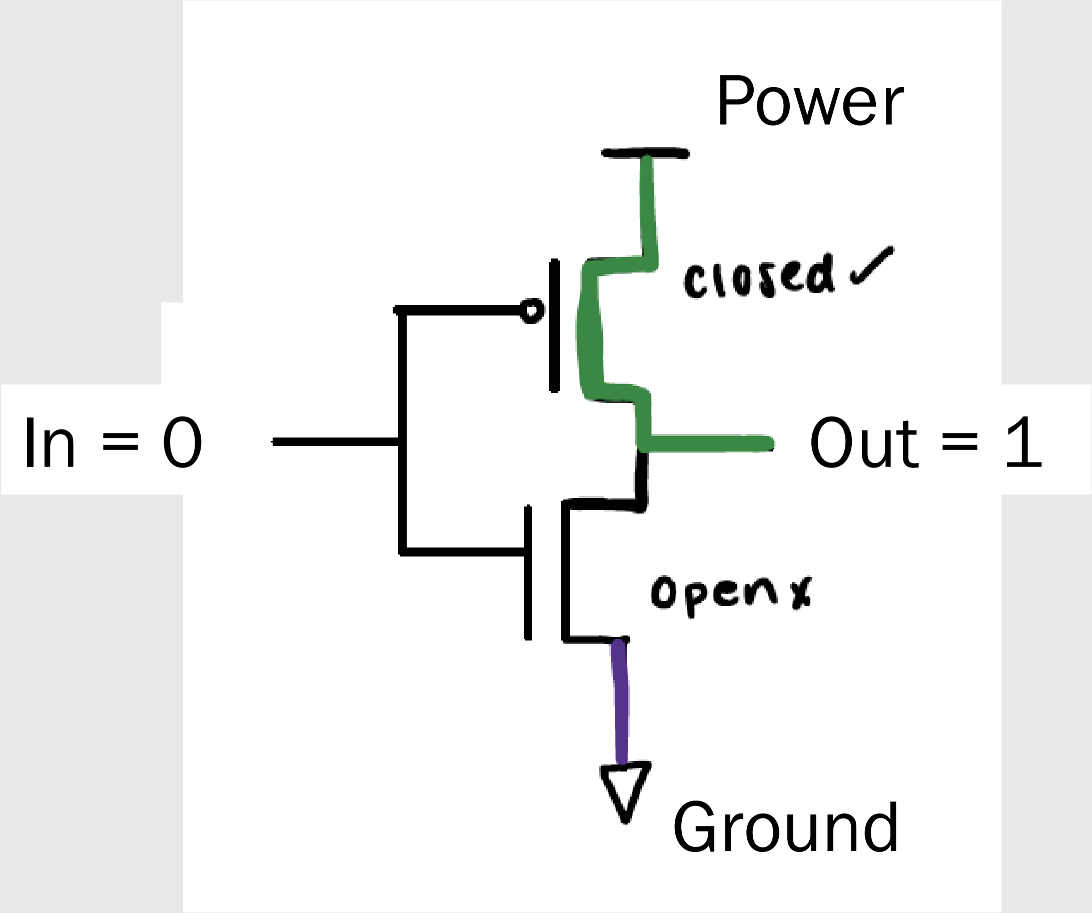
CMOS Inverter

In	Out
0	1
1	0



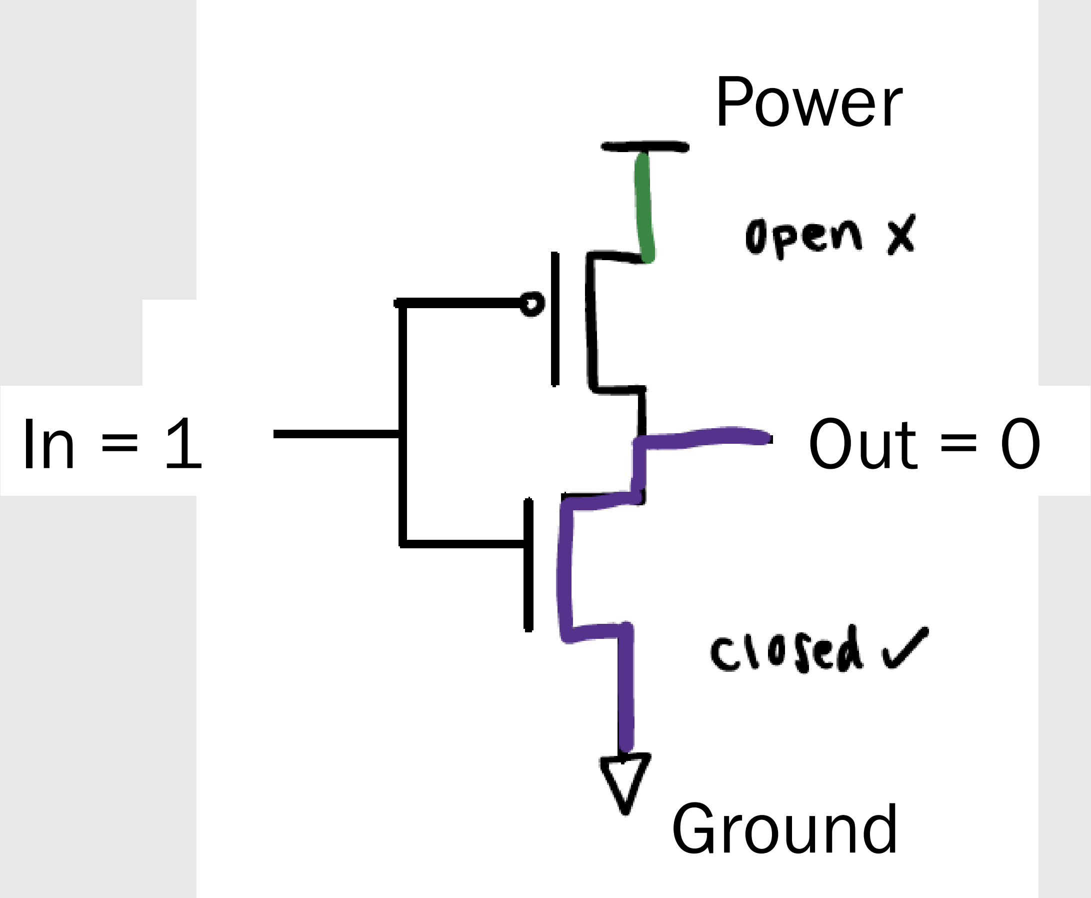
CMOS Inverter

In	Out
0	1
1	0



CMOS Inverter

In	Out
0	1
1	0

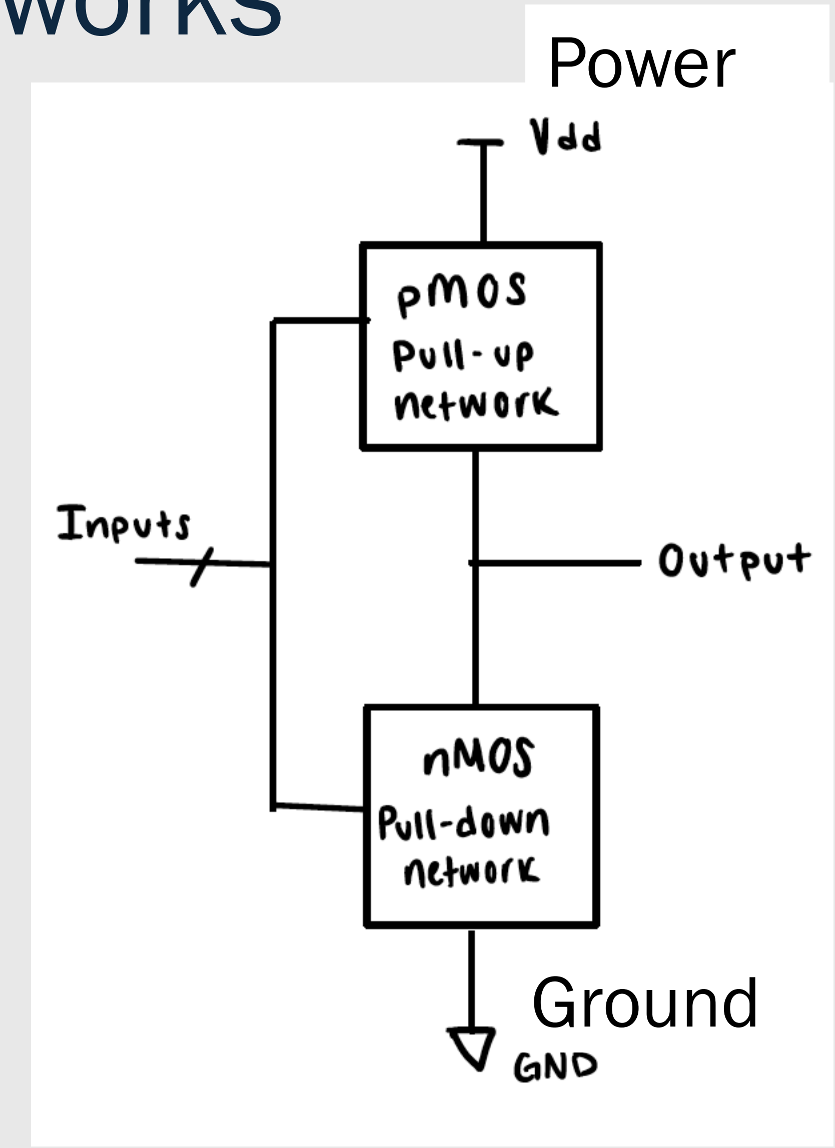


CMOS Gates

- 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*

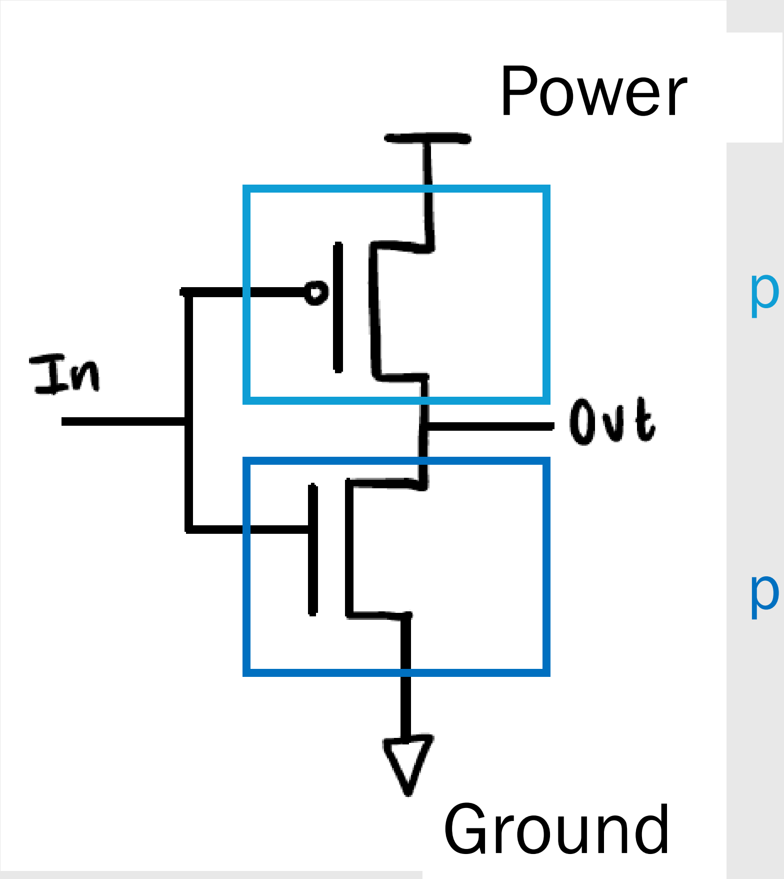
Pull-up and Pull-down Networks

- 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

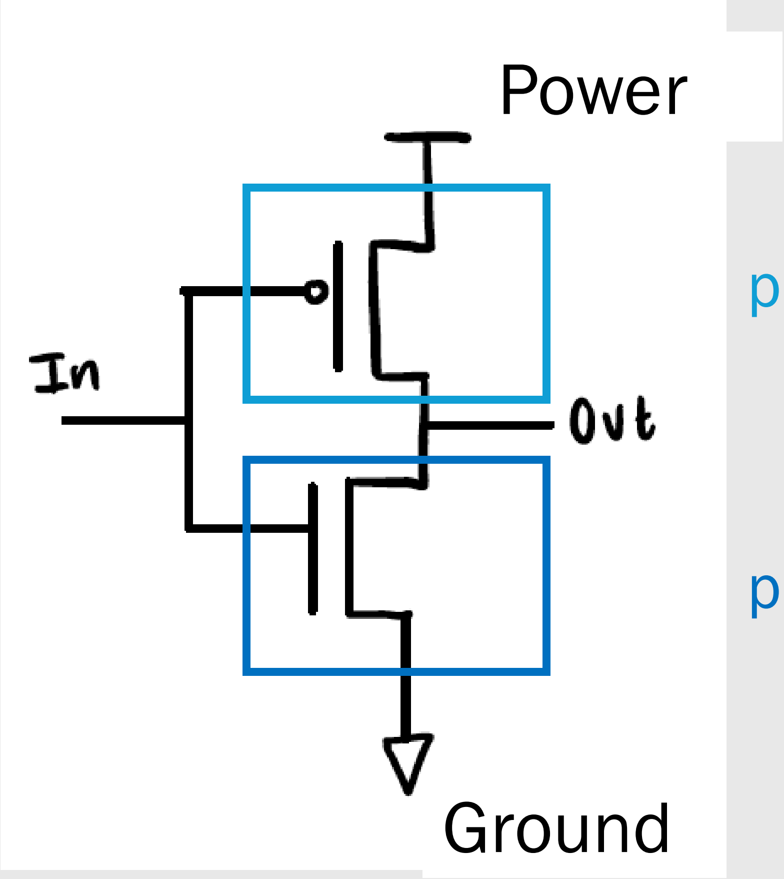


pull-up network

pull-down network

CMOS Tracing

In	Out
0	1
1	0



pull-up network

pull-down network

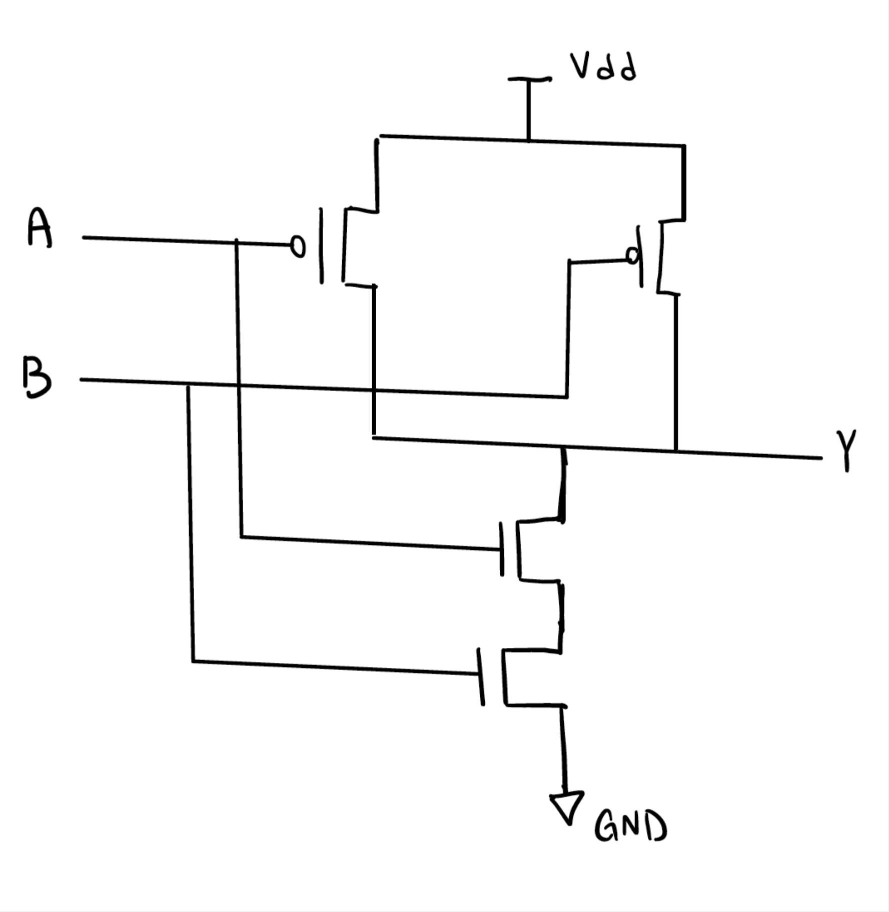
CMOS NAND

CMOS NAND

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

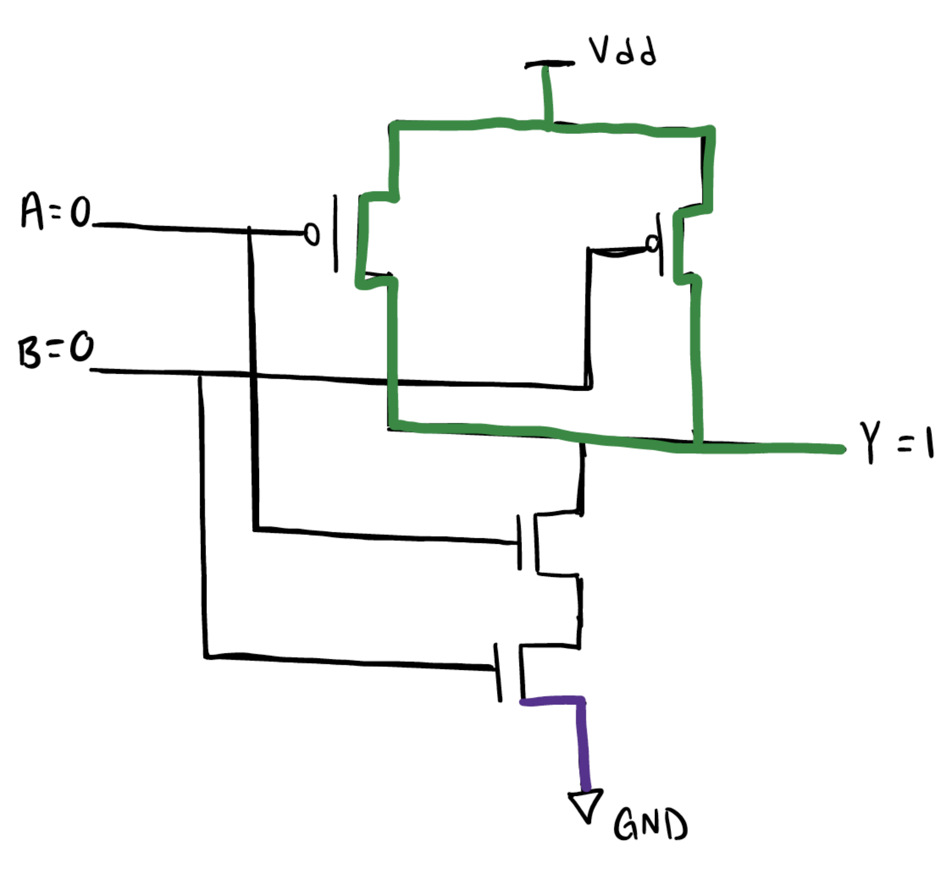
CMOS NAND

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



CMOS NAND

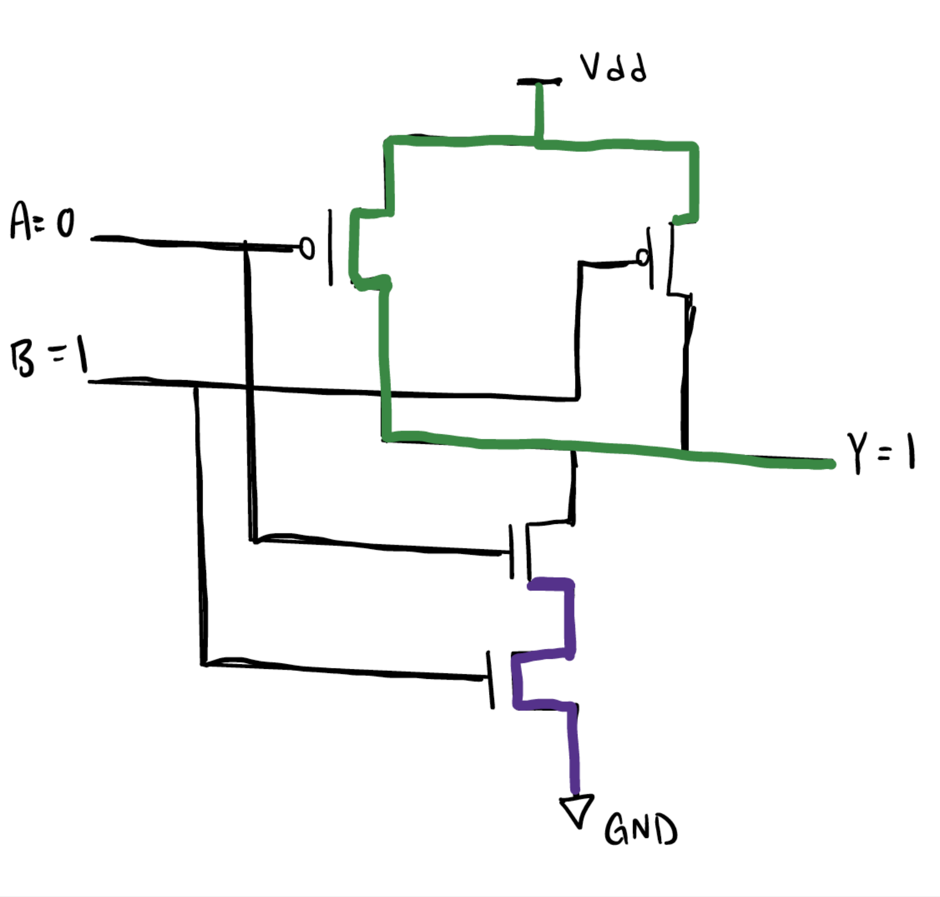
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



both pMOS on, pull-up works

CMOS NAND

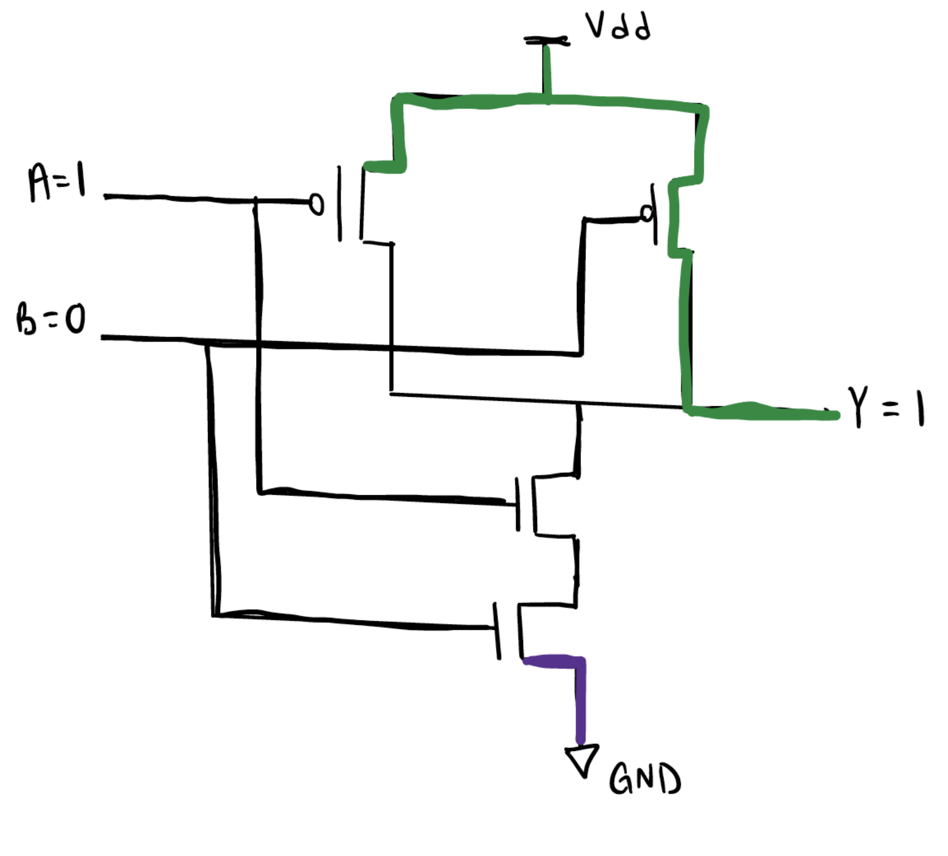
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A’s pMOS still on, output pulled high

CMOS NAND

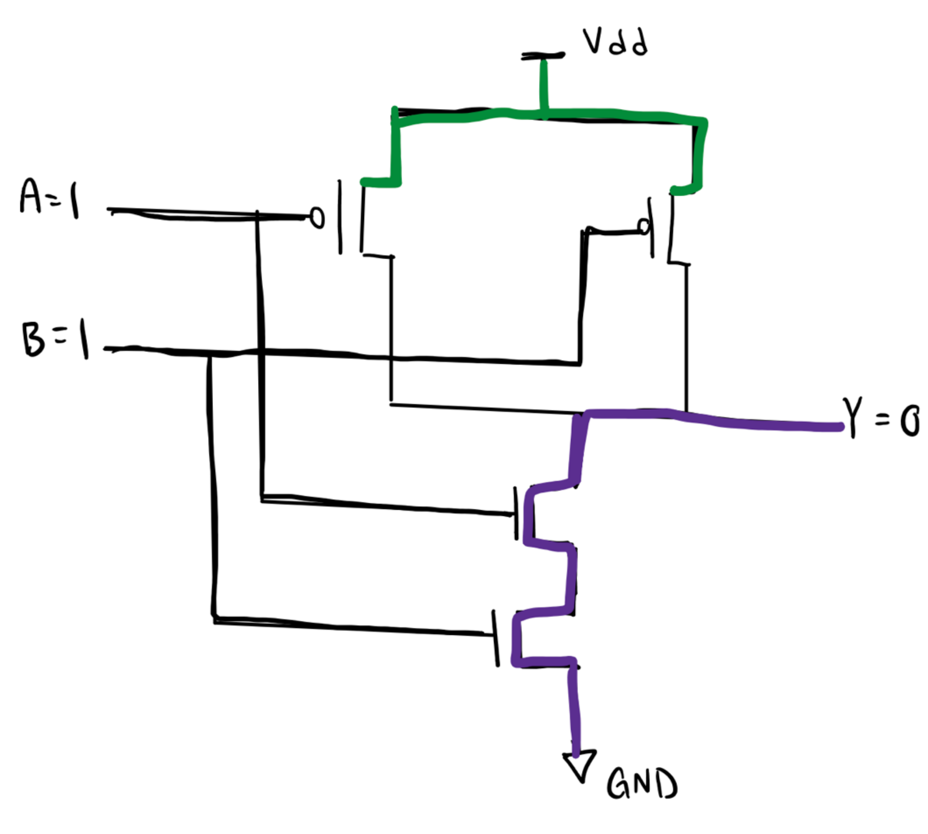
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



B's pMOS still on, output pulled high

CMOS NAND

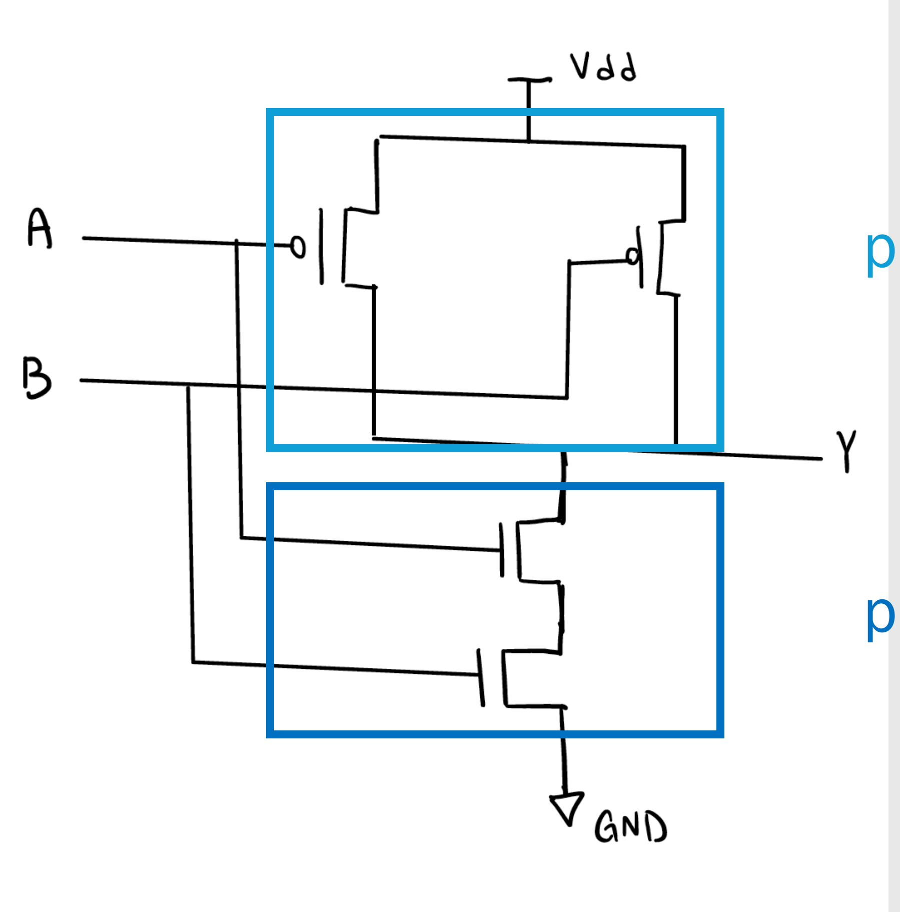
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



both pMOS off, both nMOS on ⟶ path to ground

CMOS NAND

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



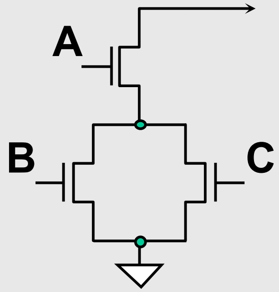
pull-up network

pull-down network

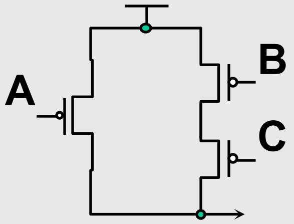
General CMOS Gate Recipe

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

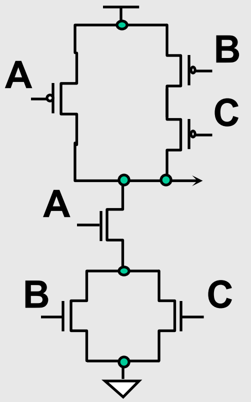
e.g., F = A*(B+C)

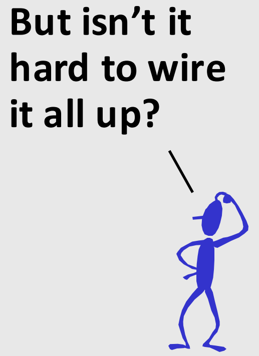


Step 2. Build the "opposite" version of that network to figure out when the output should be **1.**



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





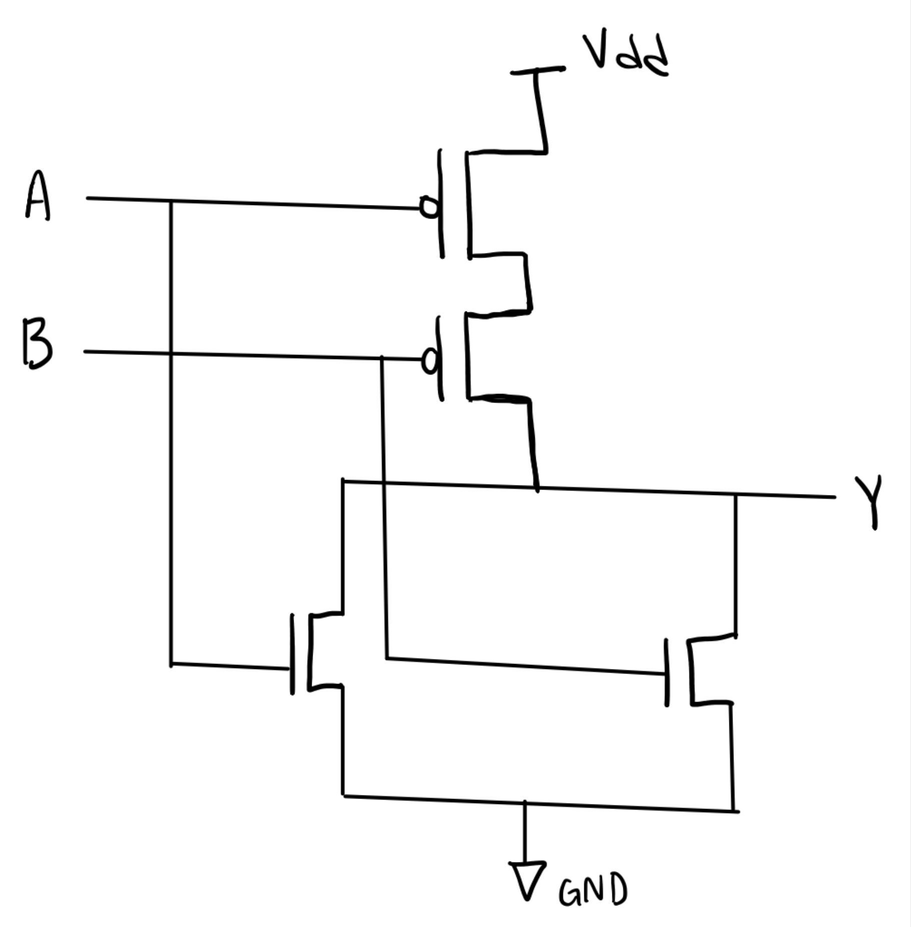
CMOS NOR

CMOS NOR

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

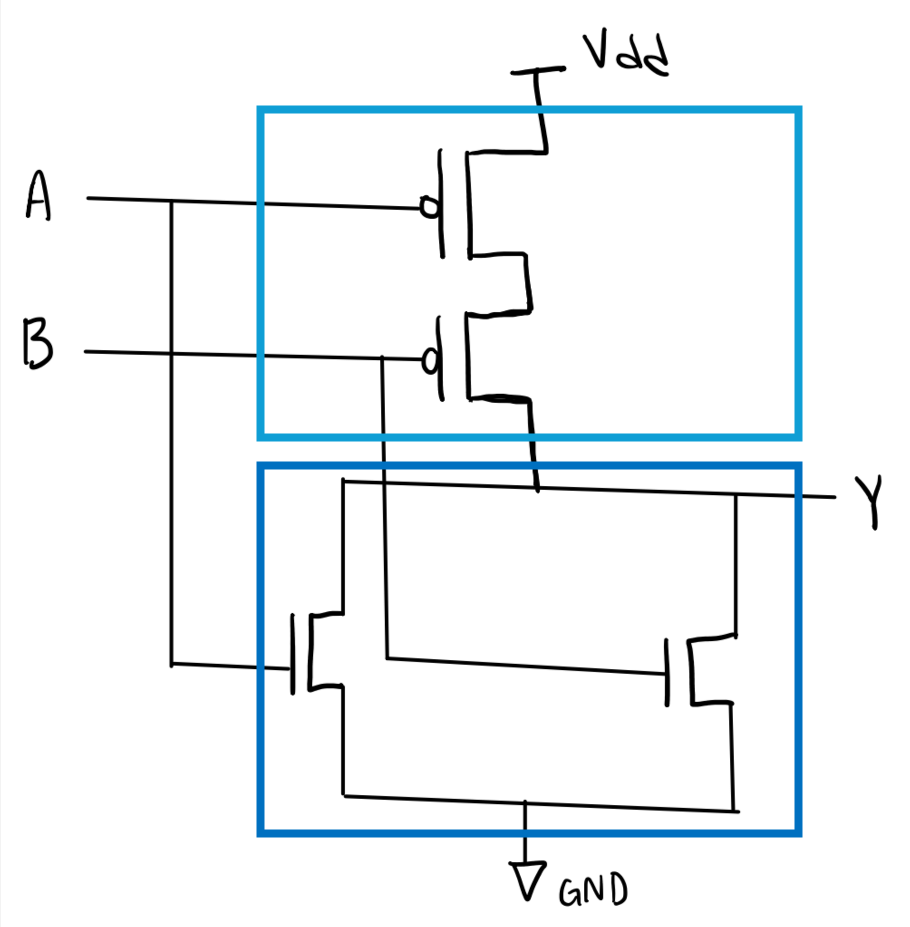
CMOS NOR

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



CMOS NOR

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

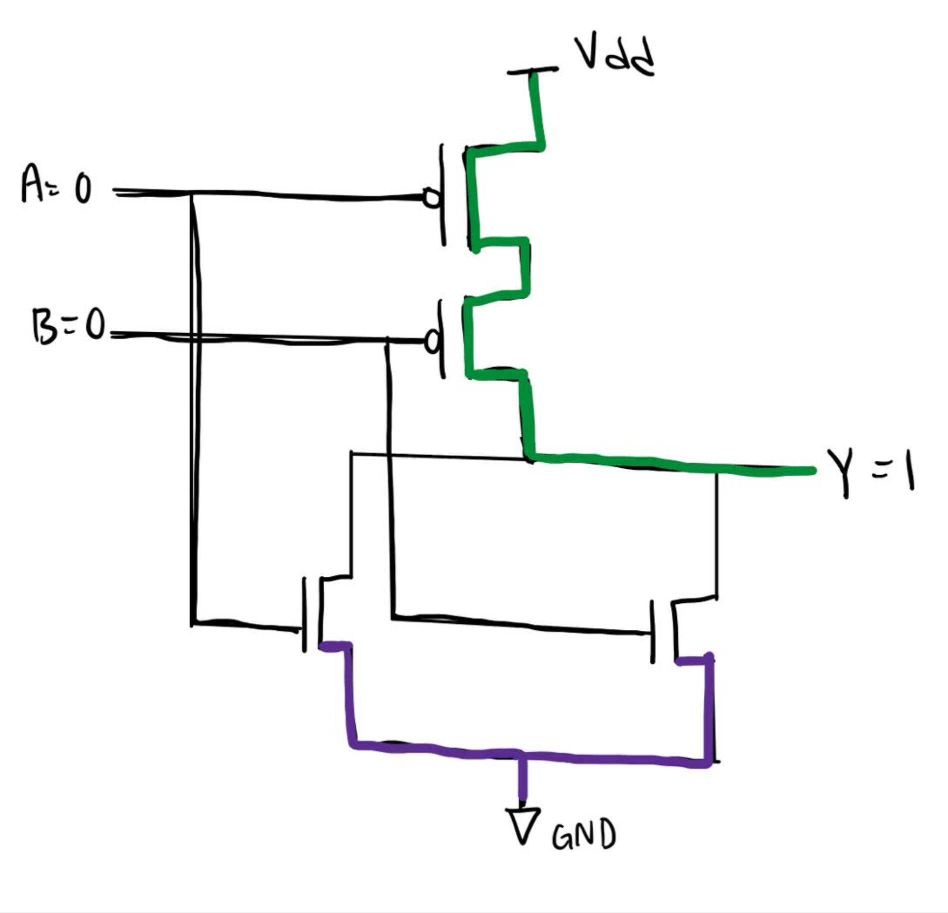


pull-up network

pull-down network

CMOS NOR

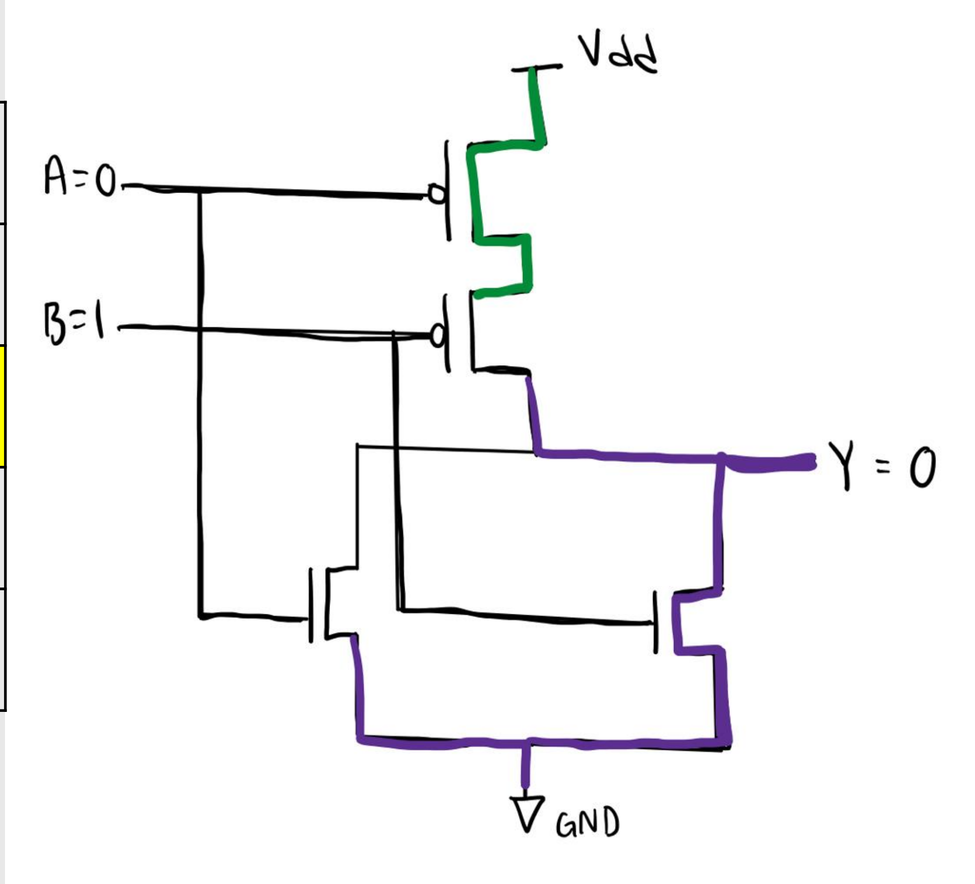
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



both pMOS on, output pulled up

CMOS NOR

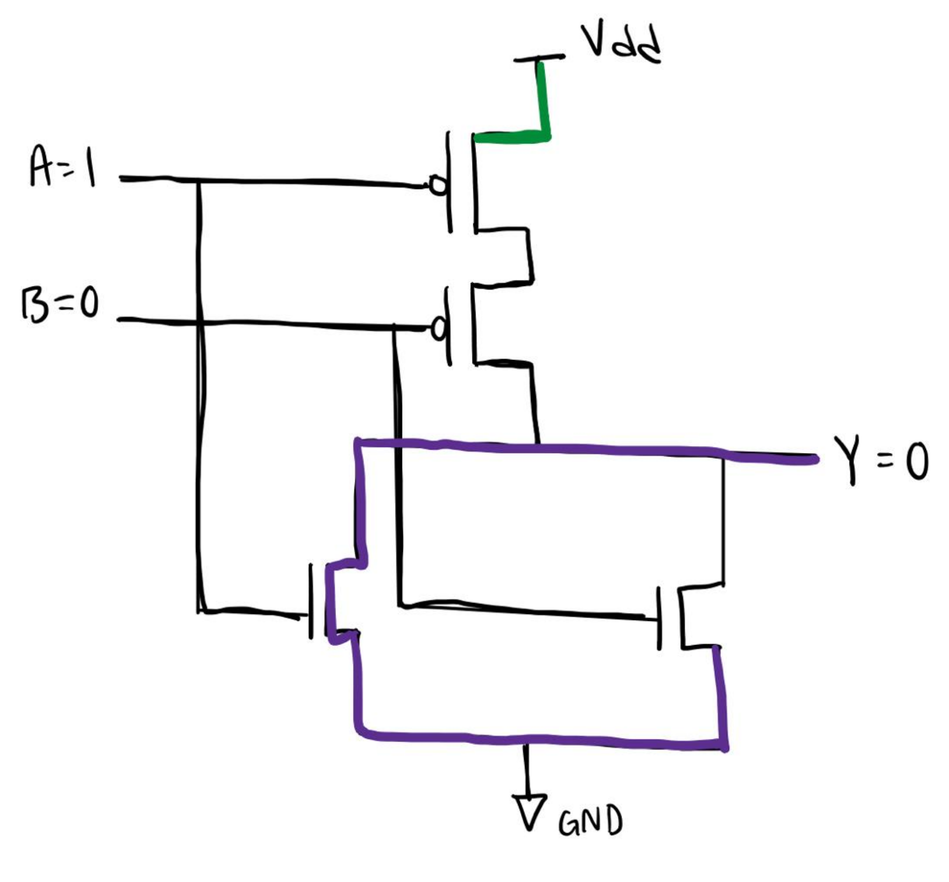
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



B’s nMOS on, pulls down

CMOS NOR

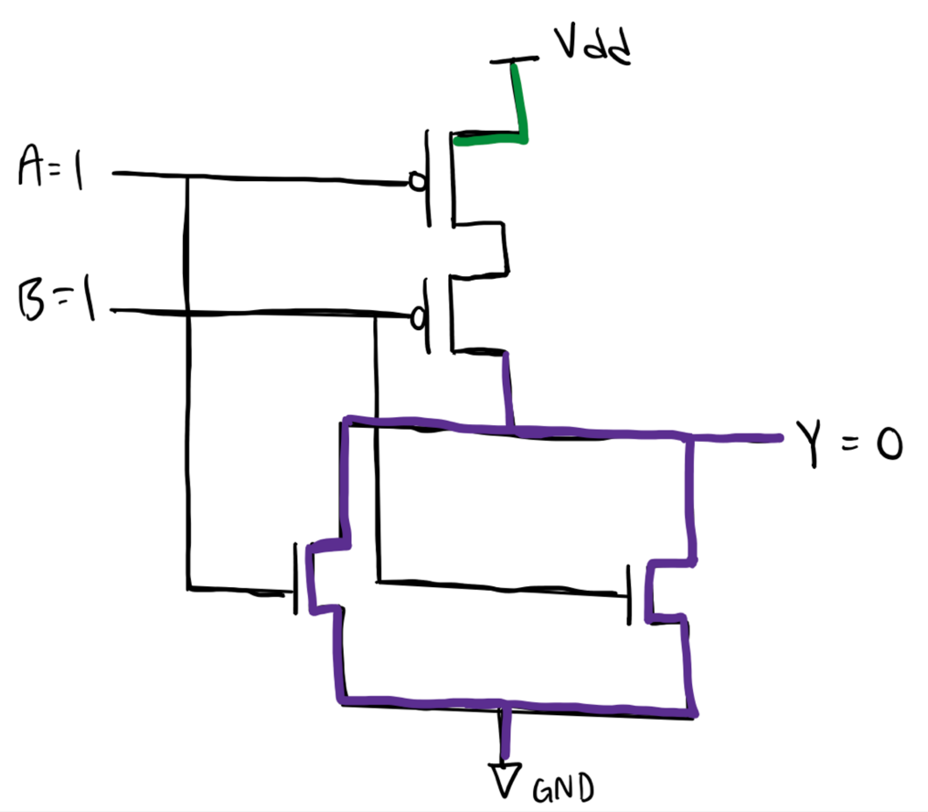
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



A’s nMOS on, pulls down

CMOS NOR

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



both nMOS on, strong pull down

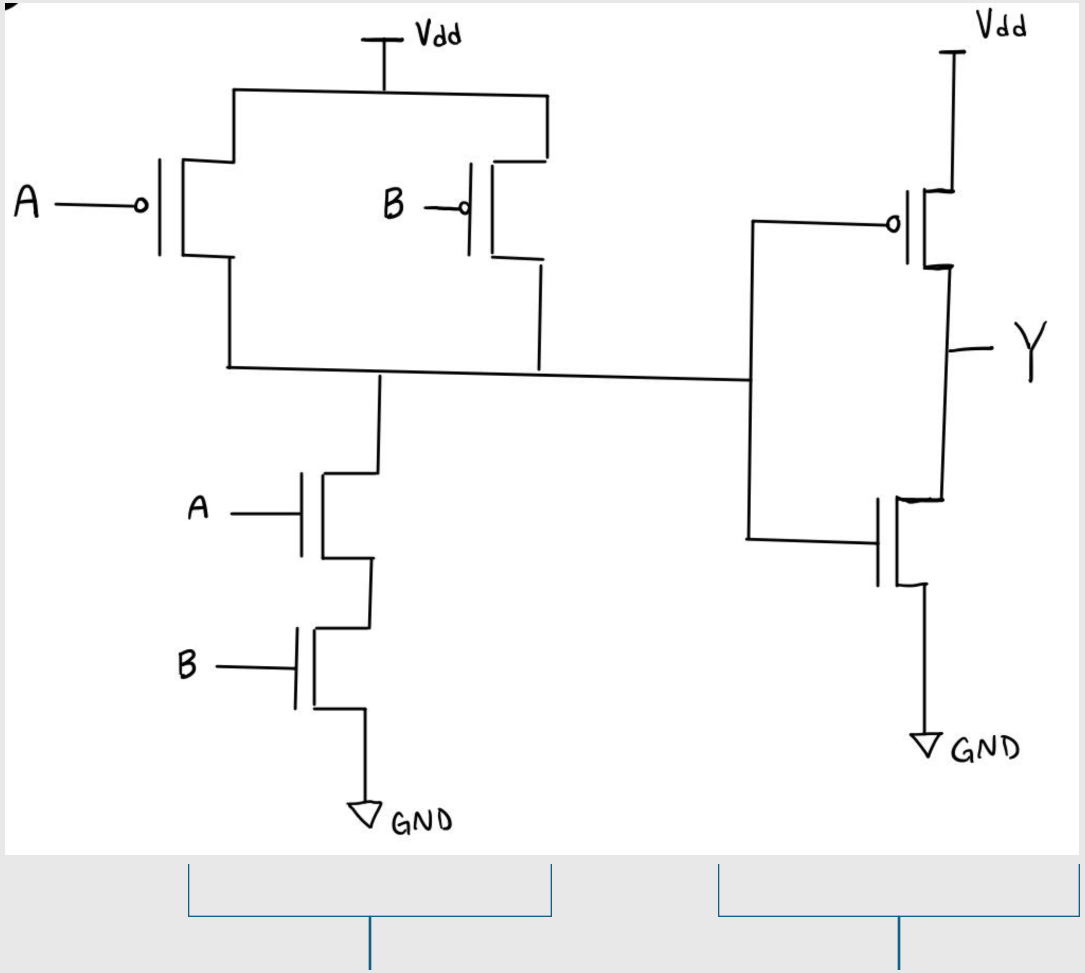
CMOS AND

CMOS AND

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

CMOS AND

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



NAND

INVERTER

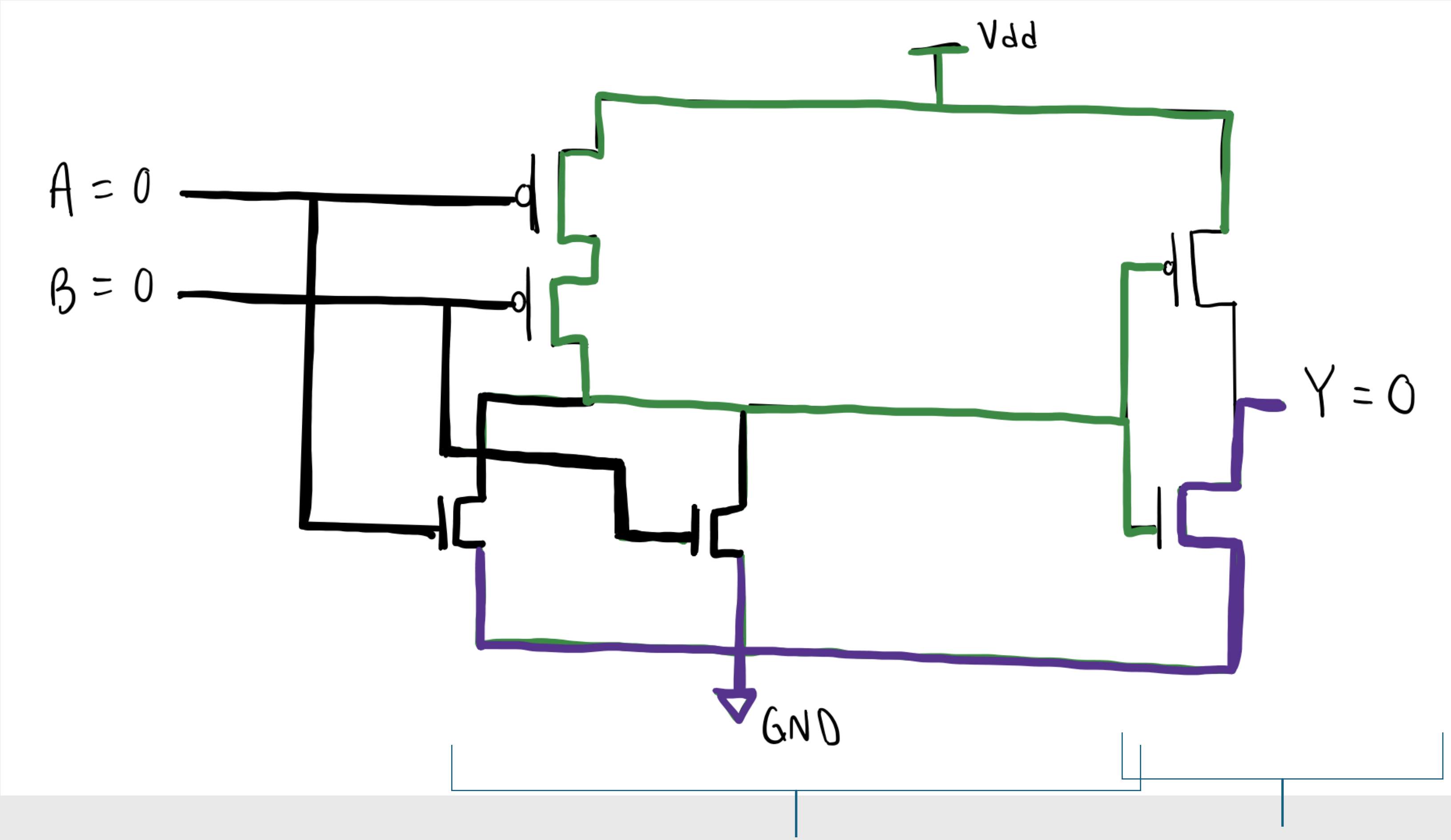
CMOS OR

CMOS OR

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

CMOS OR

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



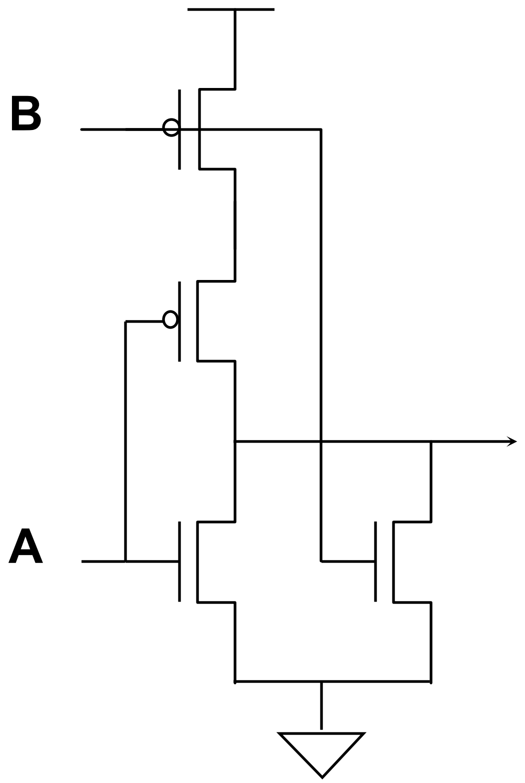
NOR

INVERTER

Transistor Count

Gate	Number of Transistors
NOT	2
AND	6
OR	6
NAND	4
NOR	4
XOR	12
XNOR	12

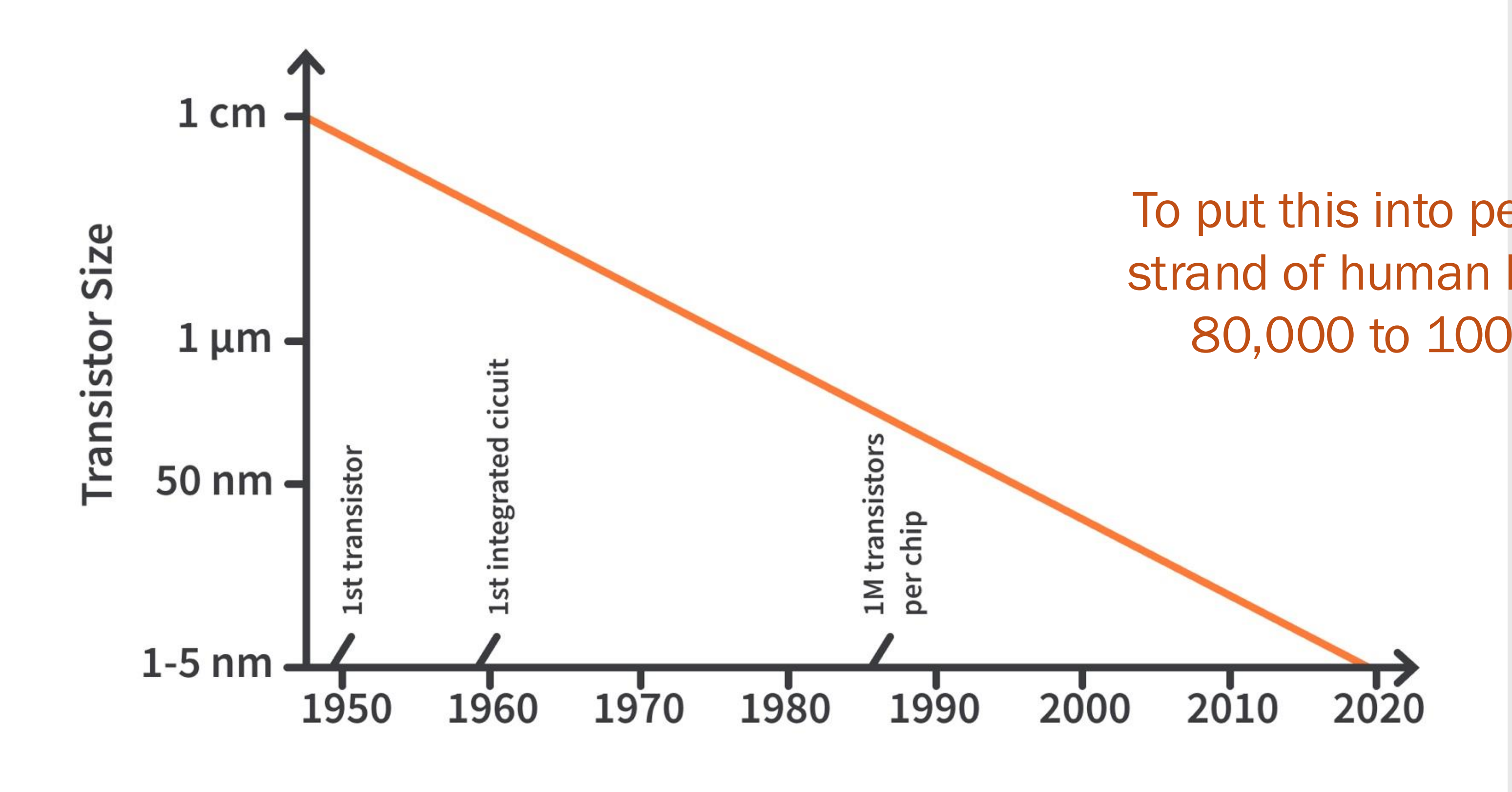
What function does this gate compute?



A	B	C
0	0	
0	1	
1	0	
1	1	

Transistor Size

- The first transistor was successfully demonstrated in 1947, at Bell Laboratories (the research arm of AT&T)
- Transistor size has exponentially decreased year after year leading to huge gains in performance and processing capabilities

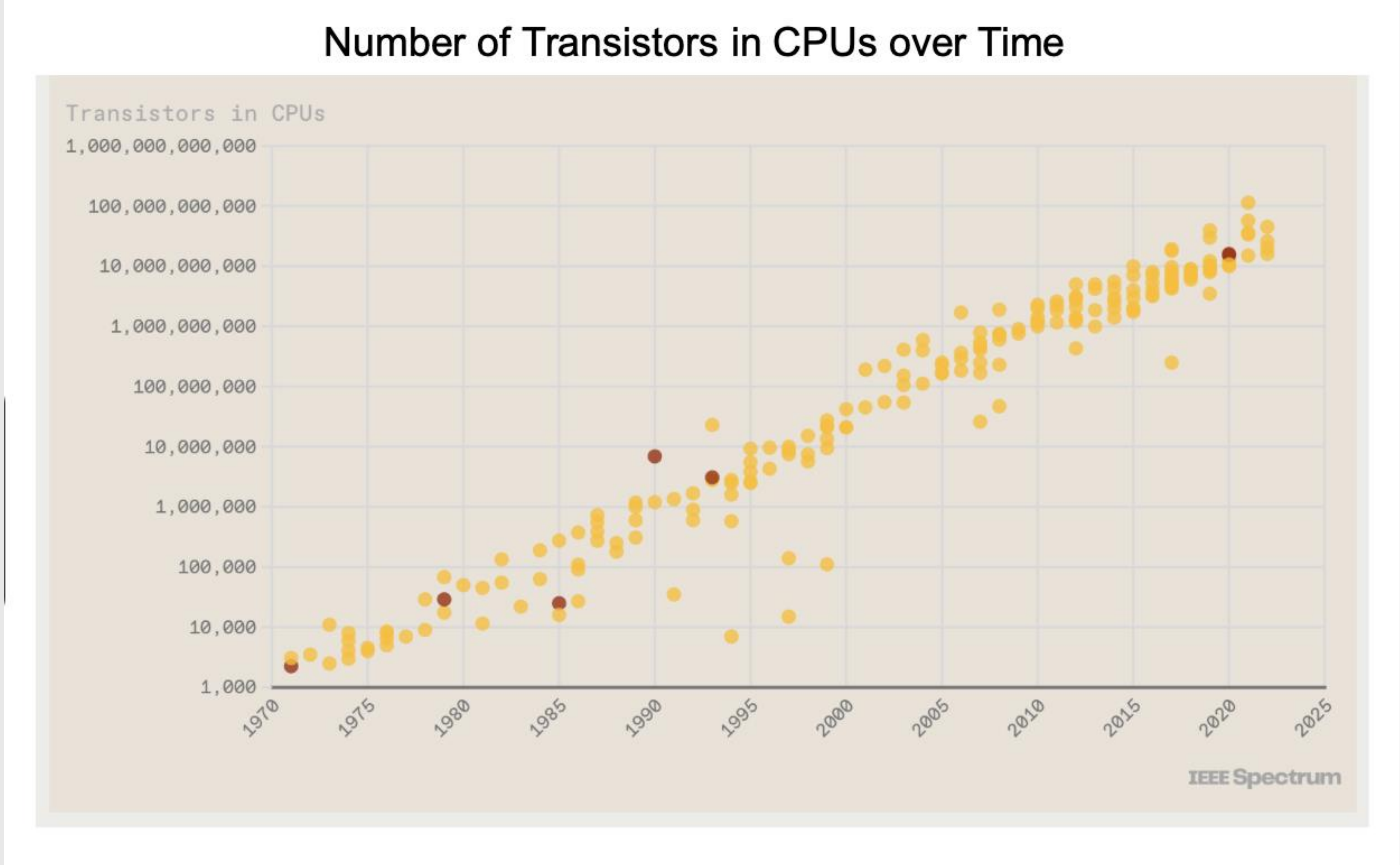


To put this into perspective, a strand of human hair is about 80,000 to 100,000 nm!

Technology Node (nanometer size)

- You may have heard that the new M4 chips use a 3nm technology node. What does that mean?
- Historically, the node name was closely related to the physical distance between the source and drain terminals (known as the gate length).
  - We are reaching the limits of how small we can make the gate length.
  - The performance i*mprovements that we see in transistor technology today are based on more factors than just the gate length.*

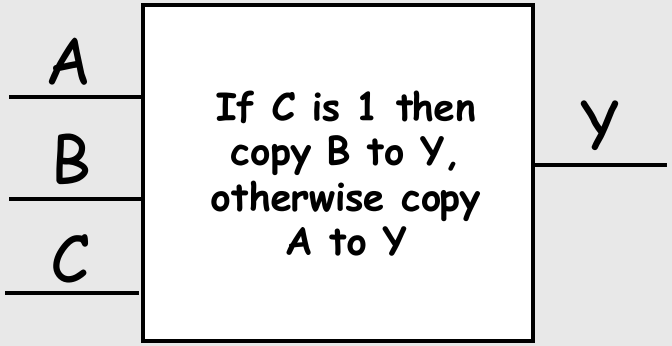
Recall: Moore’s Law



LOGIC GATES

Now can we design larger systems!

We need to start somewhere –
usually with a functional specification



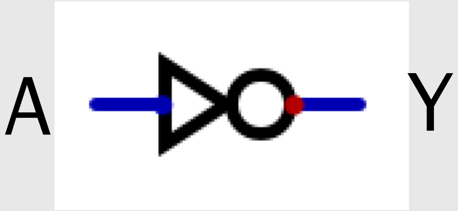
If C is 1 then copy B to Y, otherwise copy A to Y

If you are like most pragmatists you'd rather be given a table or formula than solve a puzzle to understand a function. The fact is, **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.

Inverter/Not Gate

Symbol



Truth Table

A	Y
0	1
1	0

Equation

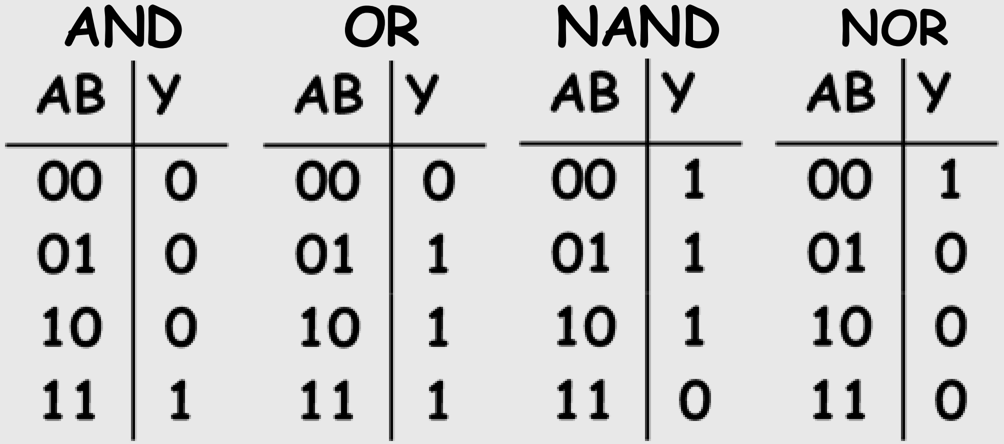
$$Y = \overline{A}$$

What Gates can we build?

Recall, we need to design our gates using a pull-up network of pMOS transistors and a pull-down network of nMOS transistors

What gates can we
- build?
- define?

Let’s start by considering only 2-input gates.



AND

AB	Y
00	0
01	0
10	0
11	1

OR

AB	Y
00	0
01	1
10	1
11	1

NAND

AB	Y
00	1
01	1
10	1
11	0

NOR

AB	Y
00	1
01	0
10	0
11	0

How many possible 2-input gates are there?

KEY IDEA: As many as there are 2-input truth tables.

All the gates!

There are only 16 possible 2-input gates… Let’s examine all of them. Some we already know, others are just silly.

INPUT AB	ZERO	AND	A > B	A	B > A	B	XOR	OR	NOR	XNOR	NOT 'B'	A <= B	NOT 'A'	B <= A	NAND	ONE
00	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~	0	~~1~~

How many of these gates can be implemented using a single CMOS gate?

Do we really need all of these gates?

Nope! Once we realize that we can describe all of them using just AND, OR, and NOT

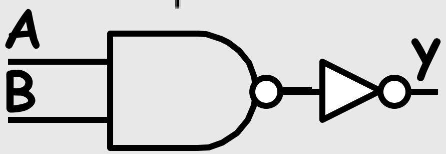
Composing gates: AND and OR

Each can be constructed using a pair of CMOS gates

AND is just NAND with an inverter, and OR is just NOR with an inverted output.

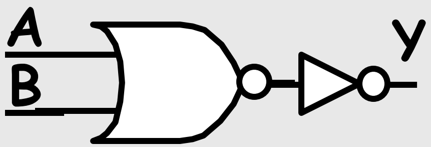
AND

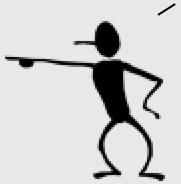
AB	Y
00	0
01	0
10	0
11	1



OR

AB	Y
00	0
01	1
10	1
11	1





These two gates are particularly important. Using them will allows us to develop a systematic approach for constructing any combinational function.

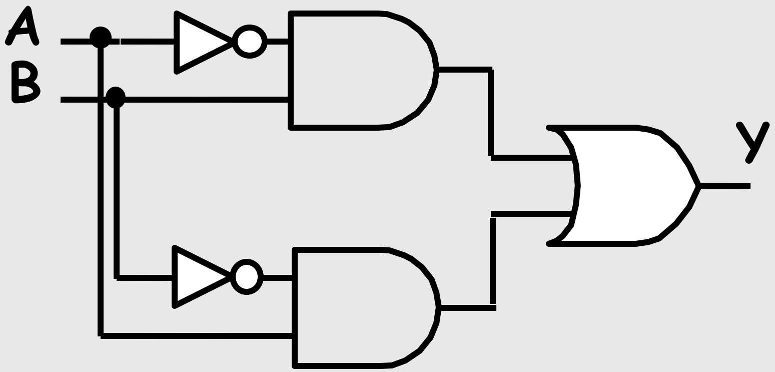
Composing gates

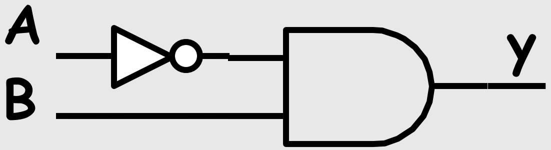
B>A

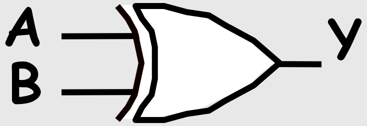
AB	Y
00	0
01	1
10	0
11	0

XOR

AB	Y
00	0
01	1
10	1
11	0





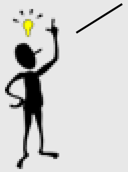


How many different gates do we really need?

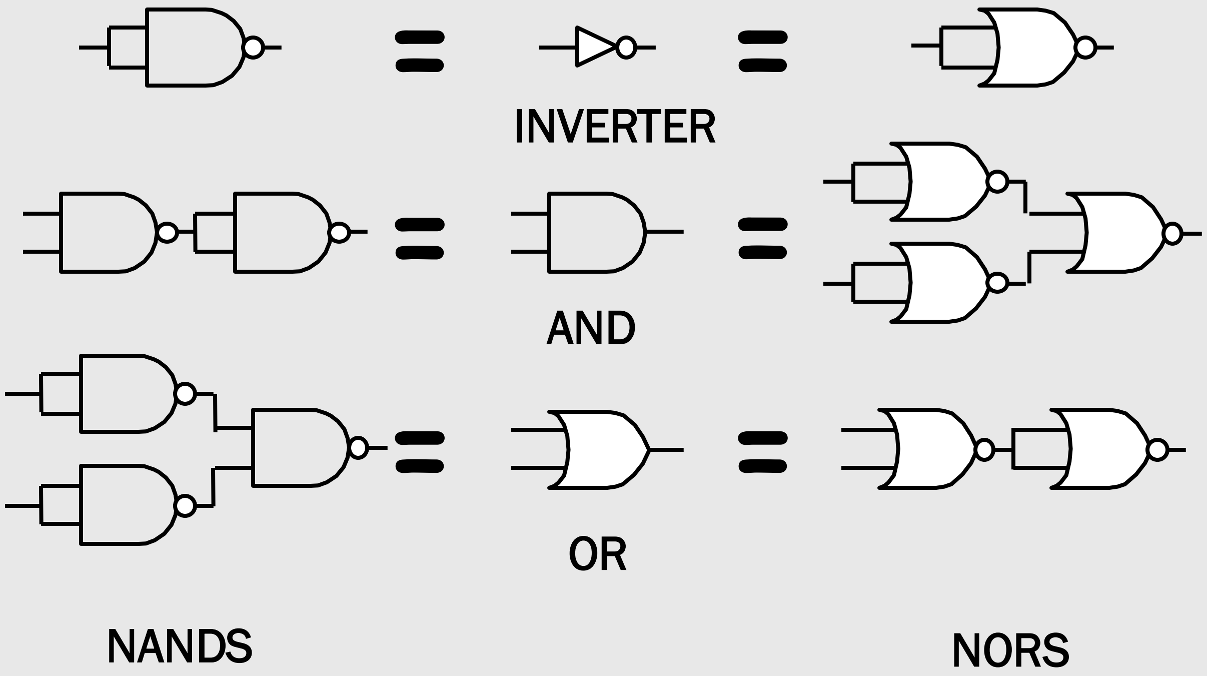
We can always do it with 3 different types of gates (AND, OR, INVERT), and sometimes with 2, but, can we use fewer?

One will do!

A UNIVERSAL gate is one that can be used to implement *ANY* COMBINATIONAL FUNCTION. There are many UNIVERSAL gates, but not all gates are UNIVERSAL.

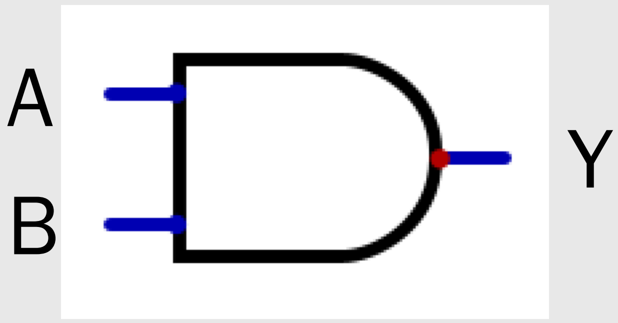


NANDs and NORs are UNIVERSAL!



AND Gate

Symbol



Truth Table

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

Equation

$$Y = A \text{ x } B$$

or

$$Y = AB$$