COMSM1302
Overview of Computer Architecture

Lecture 3

Transistor logic and CMOS

In this lecture

Foundations	• Data representation, logic, Boolean algebra.
Building blocks	• **Transistors, transistor based logic**, simple devices, storage.
Modules	• Memory, simple controllers, FSMs, processors and execution.
Programming	• Machine code, assembly, high-level languages, compilers.
Wrap-up	• Operating systems, energy aware computing.

The switch - mechanical

- Mechanical switches are very useful
  - Turn things on/off
  - With several switches, we can **encode** more useful things.
- What's wrong with them?

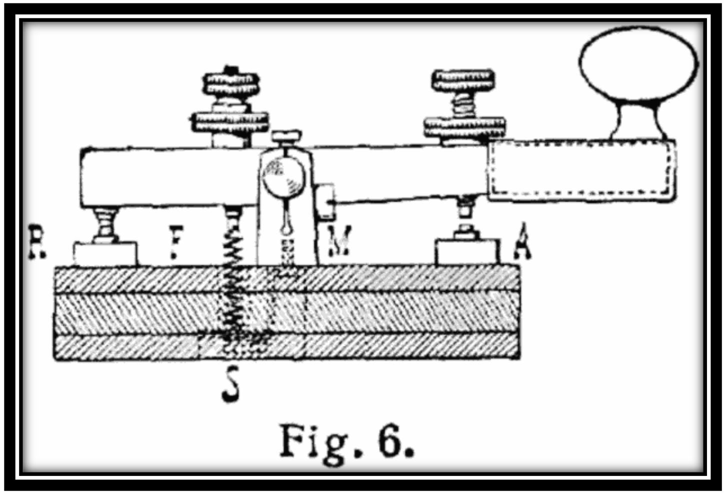
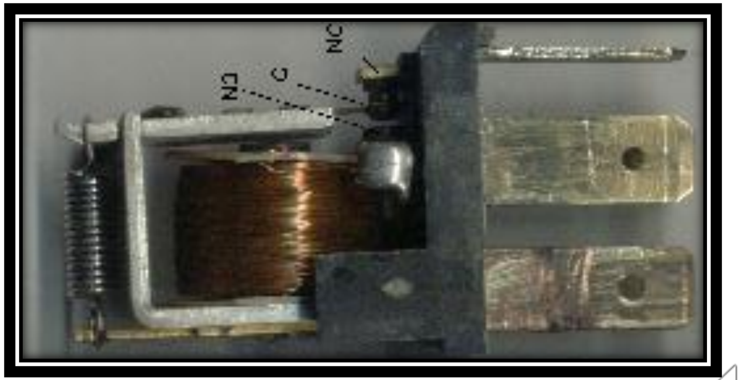
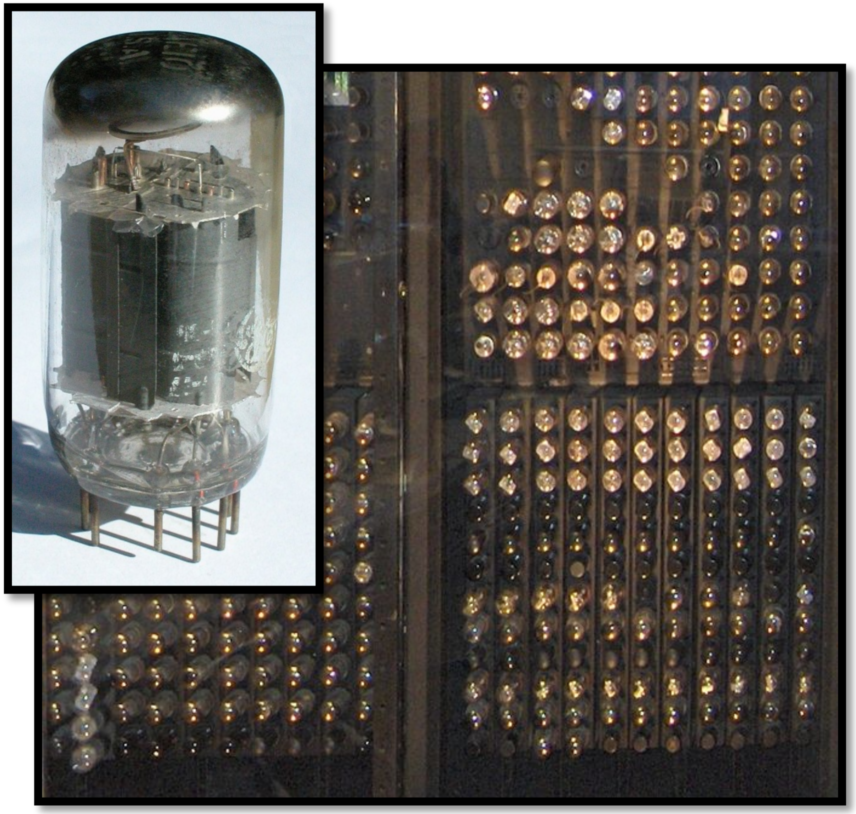


Fig. 6.



The switch - valve



Bank of valves from the ENIAC computer (1946).
Photo source: TexDex, Wikimedia Commons.

- Valves or vacuum tubes.
- Current controlled by **thermionic emission.**
  - They have a **heating element**.
- Quicker than mechanical switches.
- Fairly reliable
  - If they’re **kept on**!
  - *ENIAC (1946) used 17,468 vacuum tubes and consumed 150 kW of power.*

The switch - silicon

- Silicon, the element "Si"
  - The second most **abundant element** on Earth.
- A **semiconductor**
  - Can be constructed (using a process called **doping**) to pass electrons through a channel, when a voltage is applied to a **gate**.

The switch – silicon old

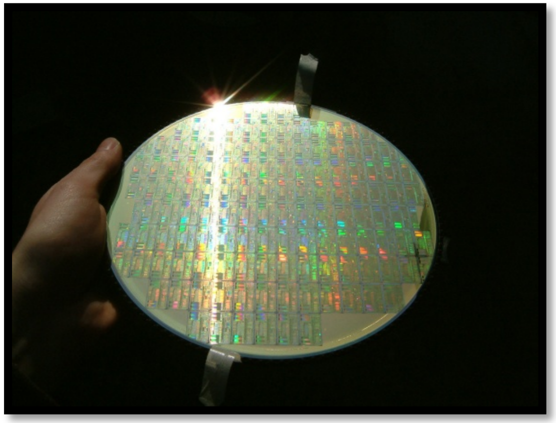
- Silicon, the element "Si"
  - The second most **abundant element** on Earth.
- A **semiconductor**
  - Can be constructed (using a process called **doping**) to pass electrons through a channel, when a voltage is applied to a **gate**.

Ore

Boule

Source: Stahlkocher, Wikimedia Commons

Wafer



Source: James Irwin, CC 2.0

The switch - silicon

- Silicon, the element “Si”
  - The second most **abundant element** on Earth.
- A **semiconductor**
  - Can be constructed (using a process called **doping**) to pass electrons through a channel, when a voltage is applied to a **gate**.

Ore

Source: Stahlkocher, Wikimedia Commons

The switch - silicon

- Silicon, the element “Si”
  - The second most **abundant element** on Earth.
- A **semiconductor**
  - Can be constructed (using a process called **doping**) to pass electrons through a channel, when a voltage is applied to a **gate**.

Ore

Boule

Source: Stahlkocher, Wikimedia Commons

The switch - silicon

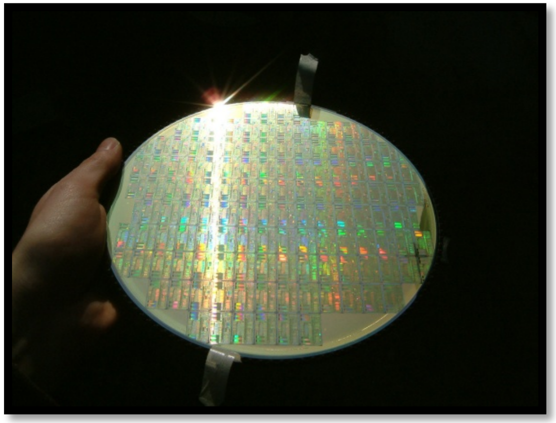
- Silicon, the element “Si”
  - The second most **abundant element** on Earth.
- A **semiconductor**
  - Can be constructed (using a process called **doping**) to pass electrons through a channel, when a voltage is applied to a **gate**.

Ore

Boule

Source: Stahlkocher, Wikimedia Commons

Wafer

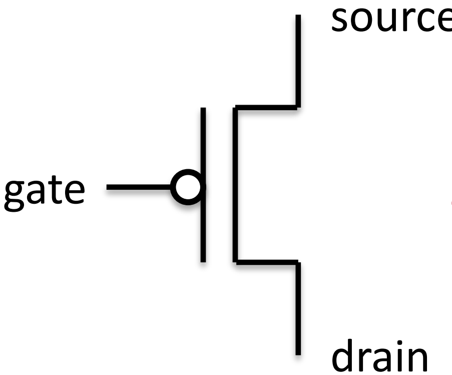


Source: James Irwin, CC 2.0

Making a switch out of silicon

- The **transistor**
  - Was invented in 1947 at AT&T's Bell Labs.
- Multiple uses
  - Amplification
  - **Switching**
- Multiple methods of construction
- We are interested in **Integrated Circuits** (ICs), so we want:

  Complementary Metal–Oxide–Semiconductor (CMOS) technology.

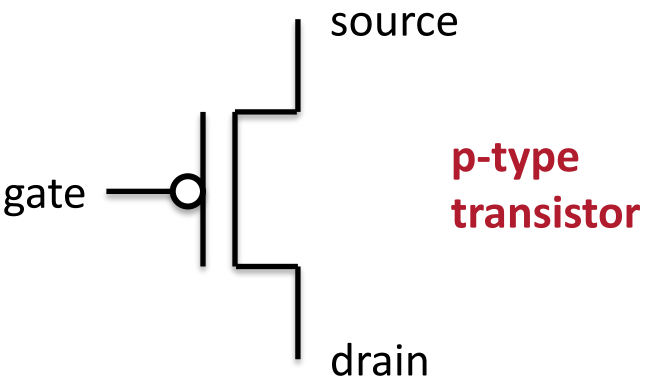


p-type transistor

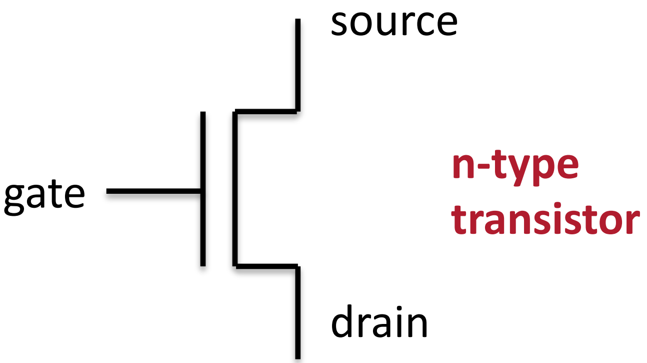
Making a switch out of silicon

- The **transistor**
  - Was invented in 1947 at AT&T's Bell Labs.
- Multiple uses
  - Amplification
  - **Switching**
- Multiple methods of construction
- We are interested in **Integrated Circuits** (ICs), so we want:

  Complementary Metal–Oxide–Semiconductor (CMOS) technology.



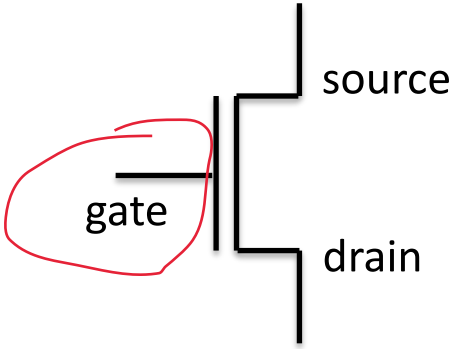
p-type transistor



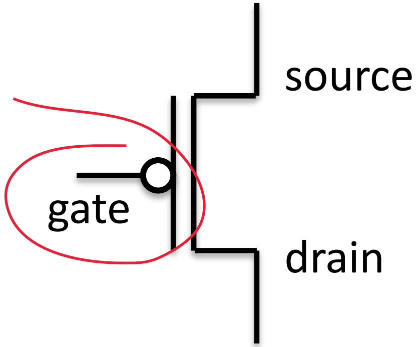
n-type transistor

n-type and p-type transistors

n-type transistor

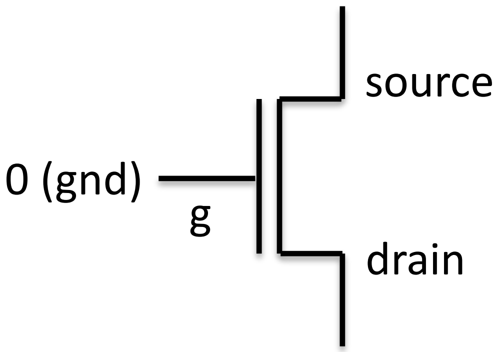


p-type transistor

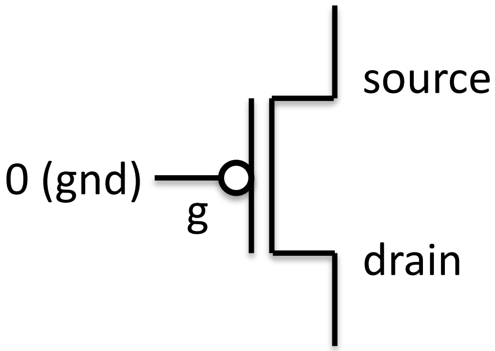


n-type and p-type transistors

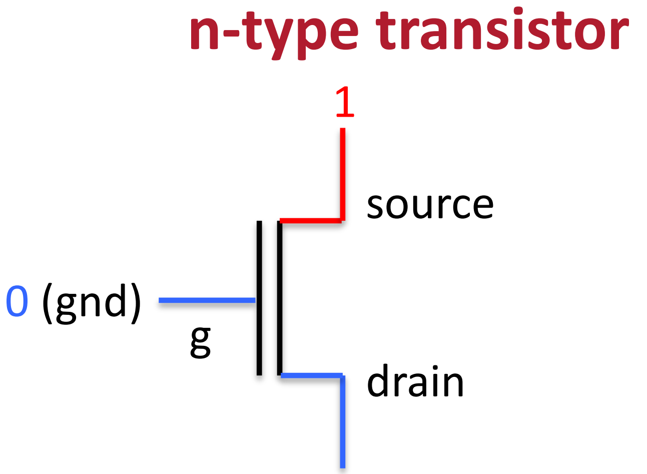
n-type transistor

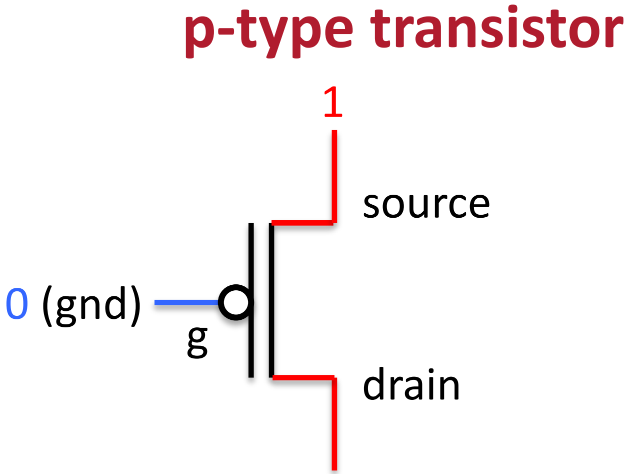


p-type transistor



n-type and p-type transistors

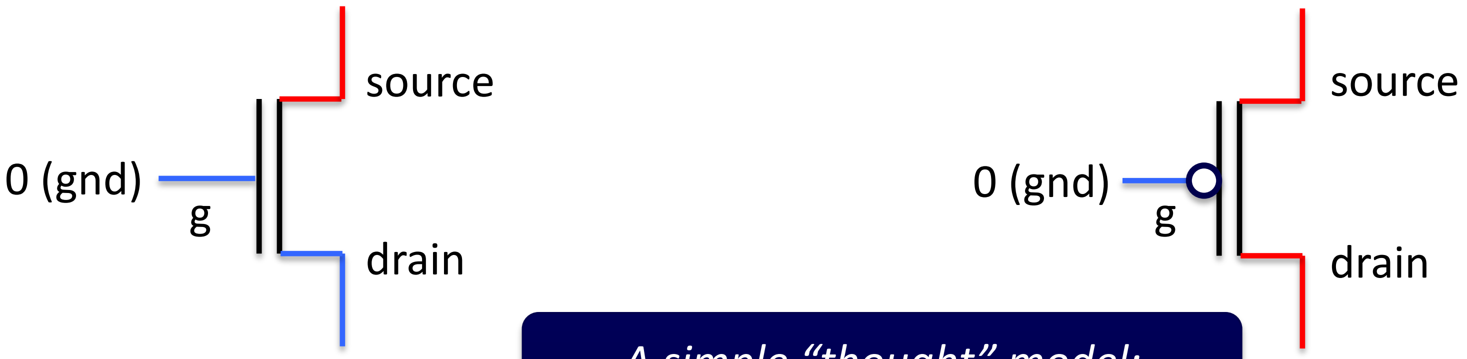




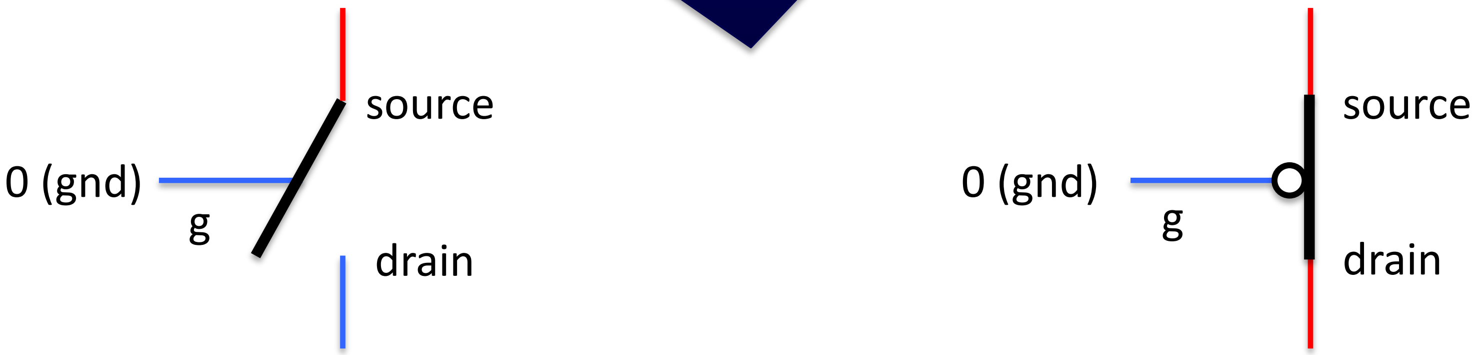
n-type and p-type transistors

n-type transistor

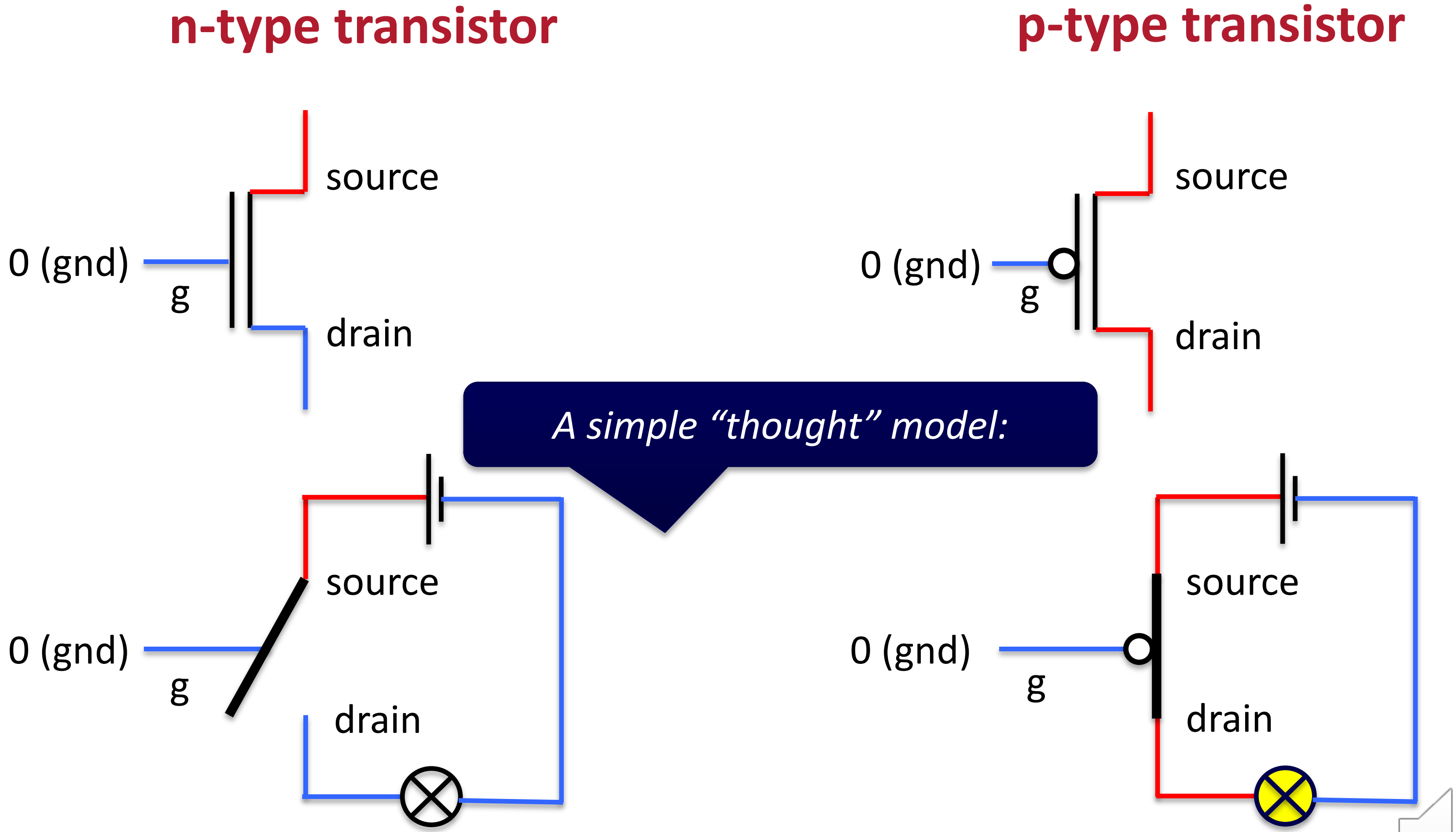
p-type transistor



A simple "thought" model:



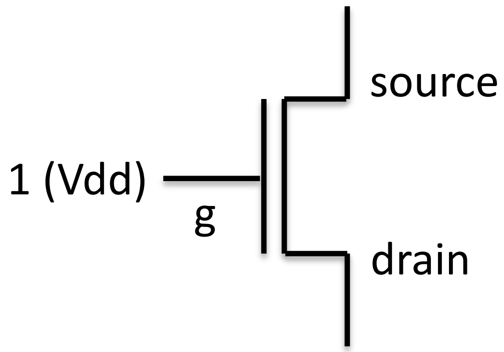
n-type and p-type transistors



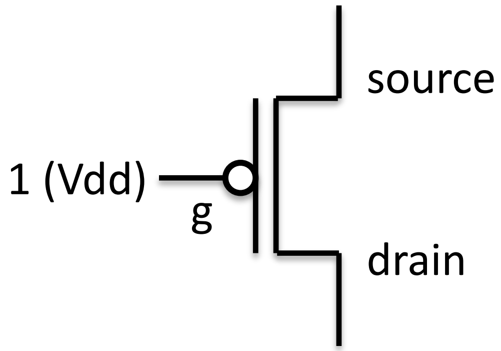
A simple "thought" model:

n-type and p-type transistors

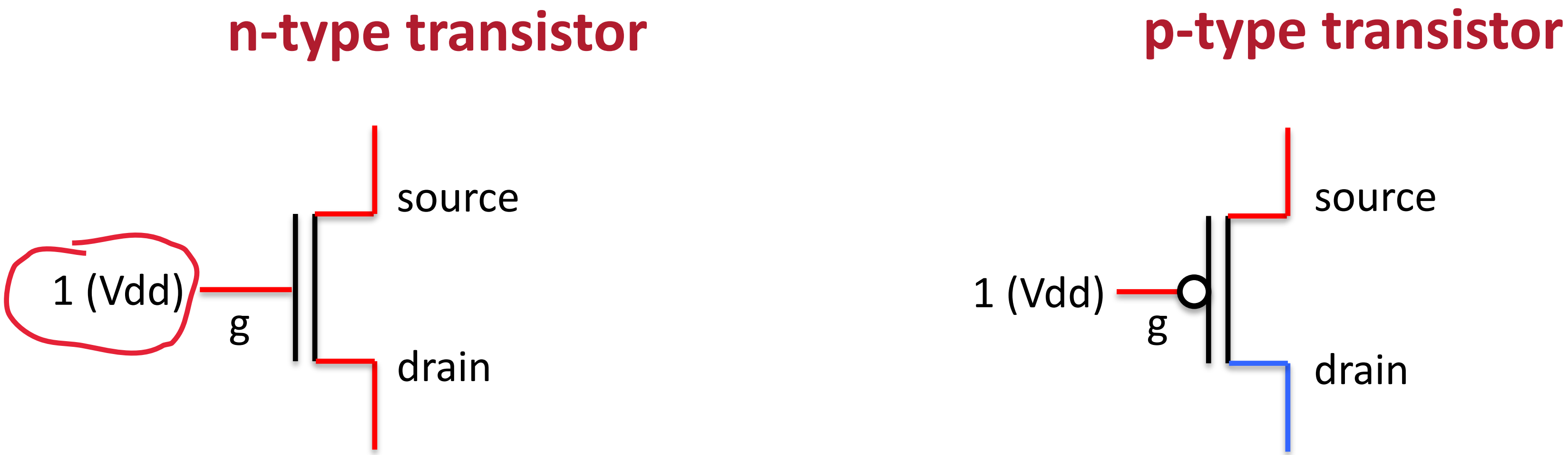
n-type transistor



p-type transistor



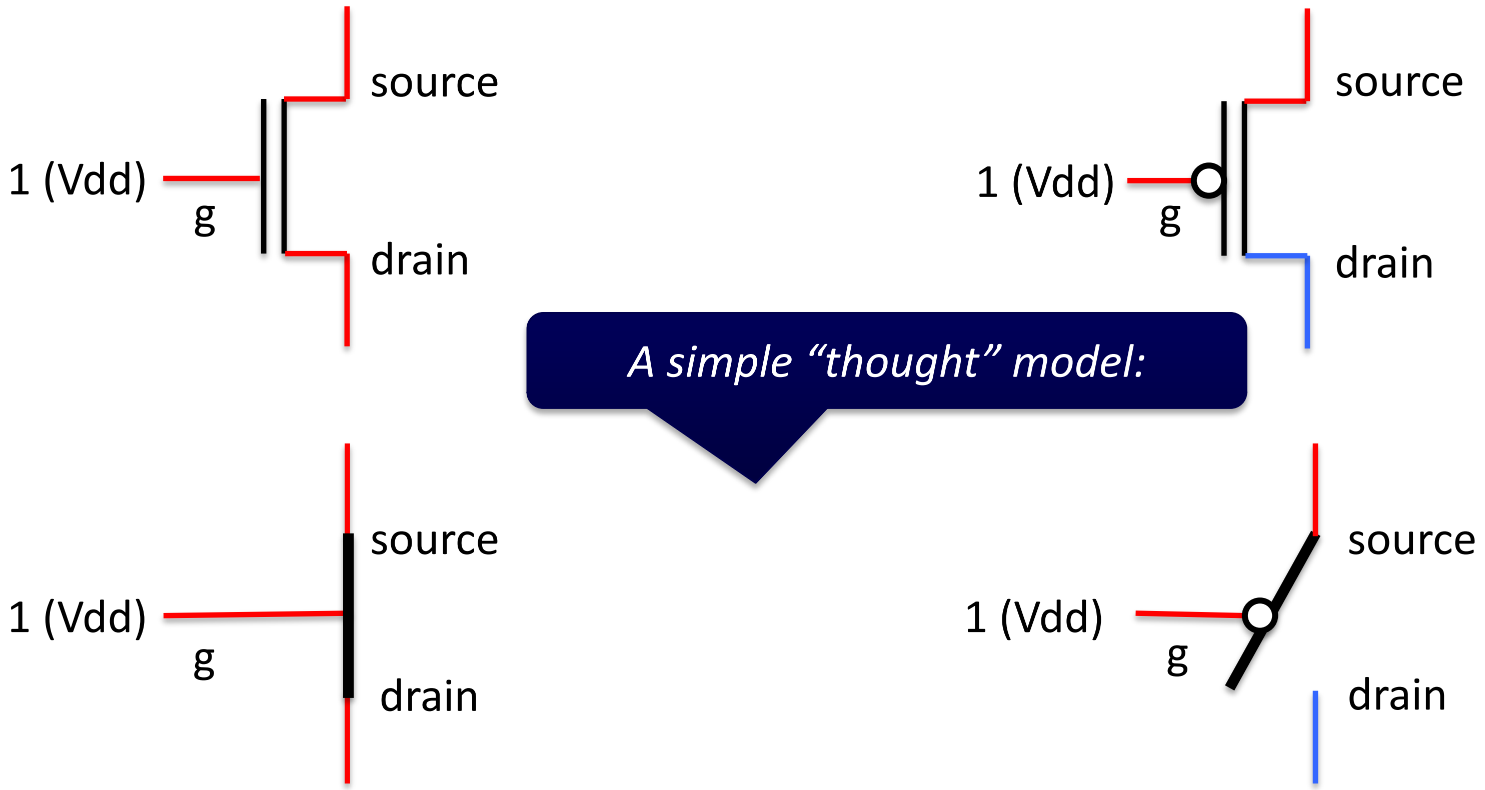
n-type and p-type transistors



n-type and p-type transistors

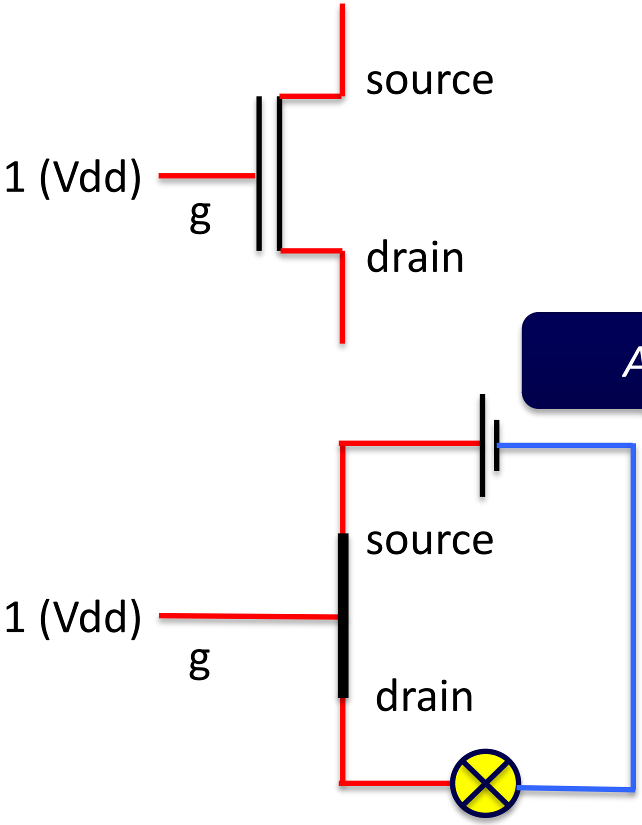
n-type transistor

p-type transistor

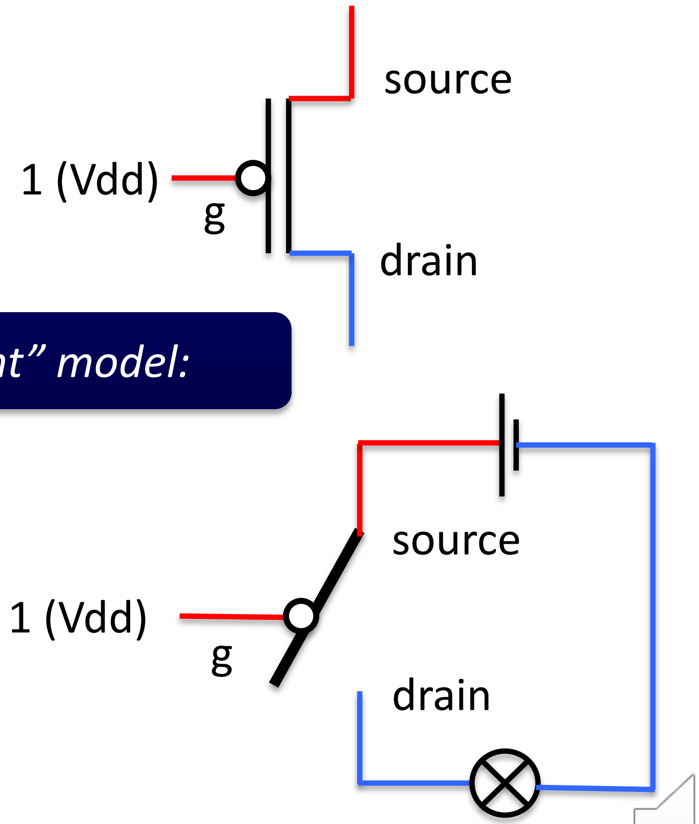


n-type and p-type transistors

n-type transistor



p-type transistor



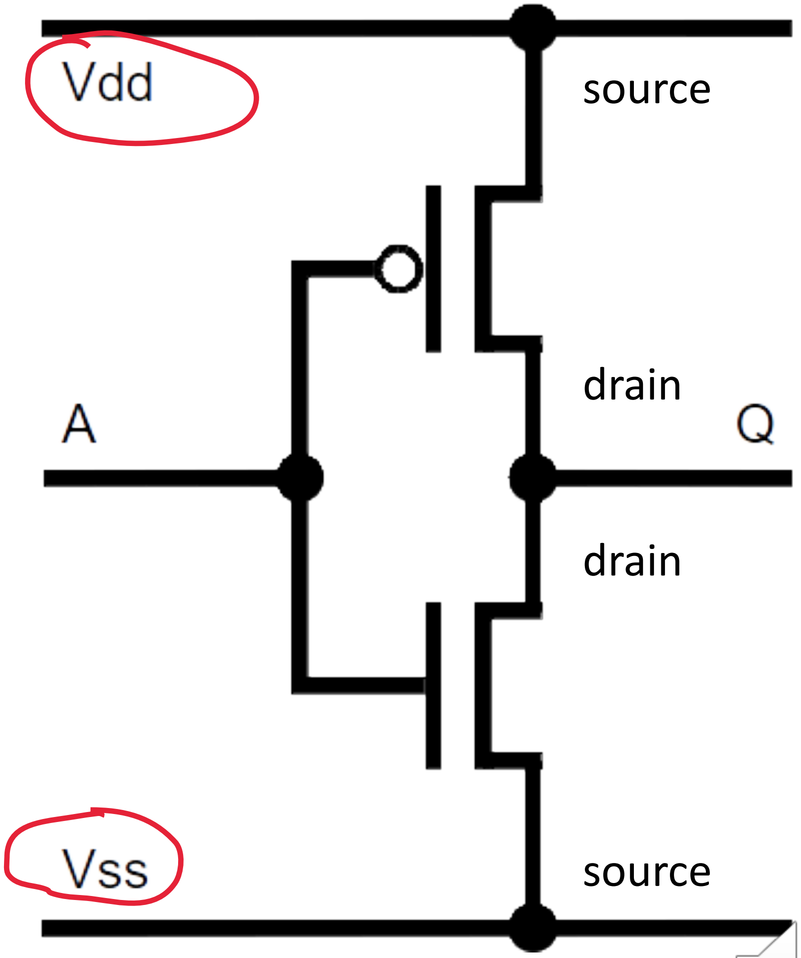
A simple "thought" model:

The most basic CMOS circuit

Composition of two **complementary** transistors

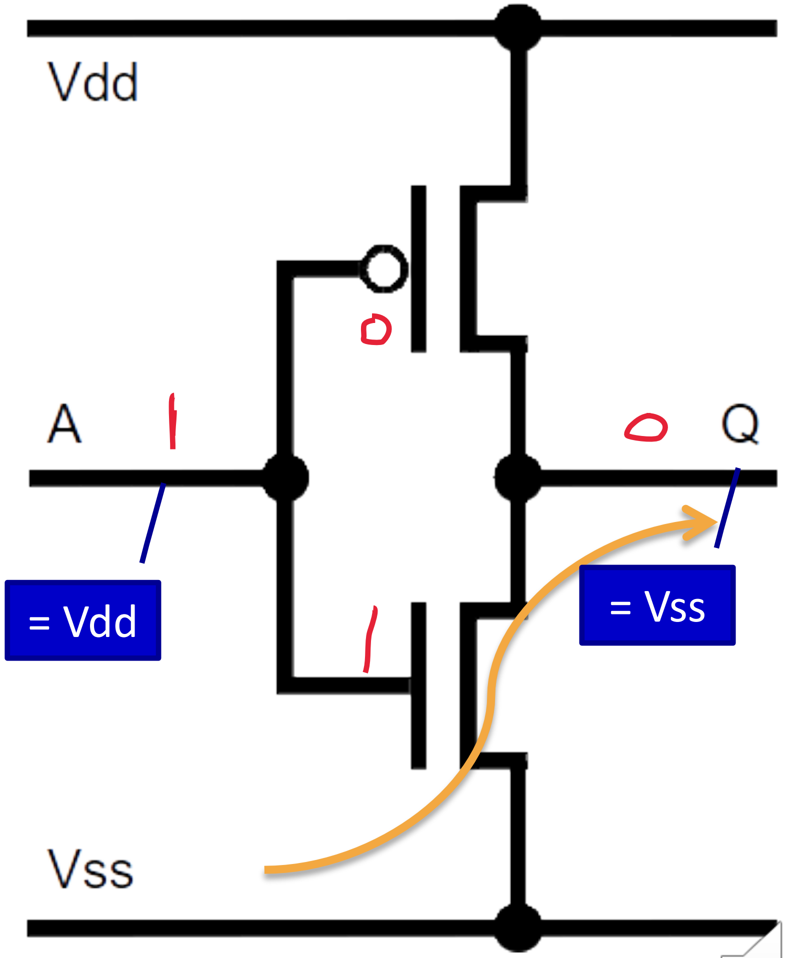
- The power supplies for CMOS are called V_{dd} and V_{ss}
  - Vdd = drain supply
  - Vss = source supply
    - "0 V" or "**ground**" voltage

How does this work?

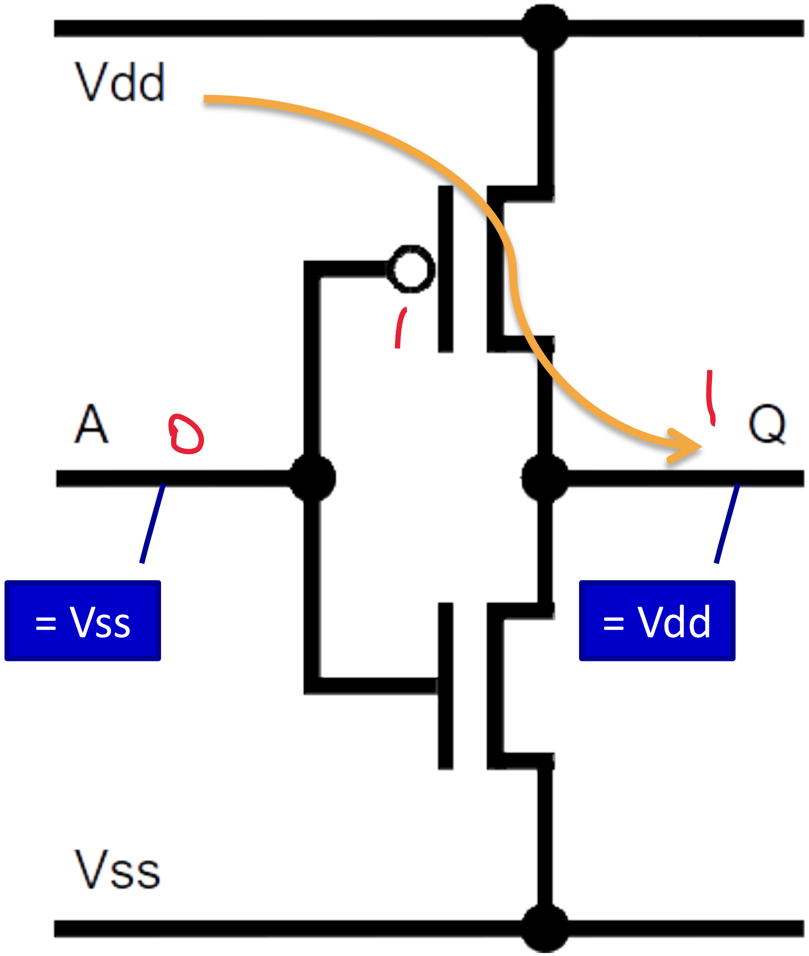


Modes of operation

- A = Vdd.
- The top “switch” is **off/open**.
  - PMOS transistor.
- The bottom “switch” is **on/closed**.
  - NMOS transistor
- Q is “connected” to Vss.



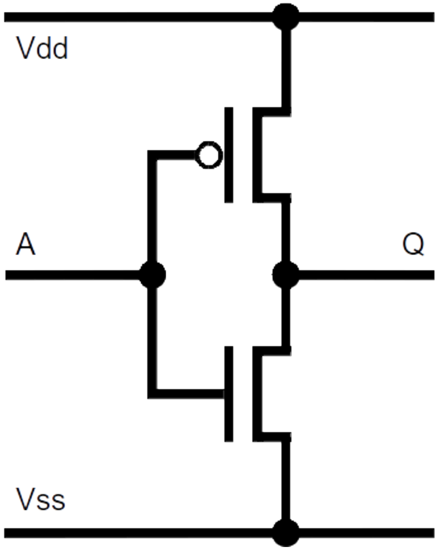
Modes of operation



- A = Vss.
- The top “switch” is **on/closed**.
- The bottom “switch” is **off/open**.
- Q is “connected” to Vdd.

What did we just make?

- The circuit has **four connections**.
- Two are **power-supply** related.
  - Providing Vdd and Vss.
- One is an **input, A.**
- One is an **output, Q.**



Symbolic

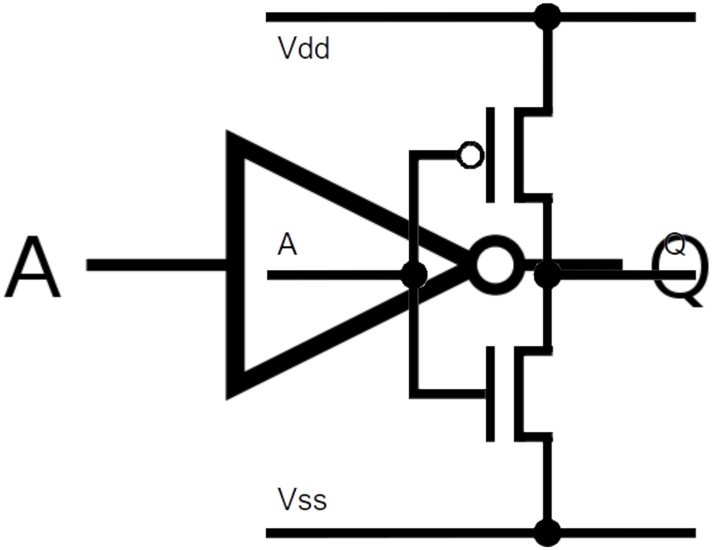
A	Q
Vss	Vdd
Vdd	Vss

Voltage

A	Q
0 V	3.3 V
3.3 V	0 V

What did we just make?

- The circuit has **four connections**.
- Two are **power-supply** related.
  - Providing Vdd and Vss.
- One is an **input, A.**
- One is an **output, Q.**



Symbolic

A	Q
Vss	Vdd
Vdd	Vss

Voltage

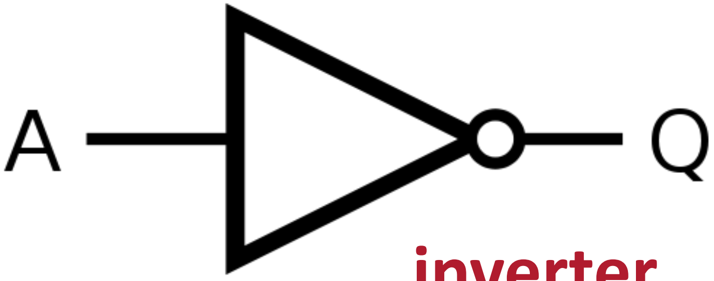
A	Q
0 V	3.3 V
3.3 V	0 V

Binary

A	Q
0	1
1	0

What did we just make?

- The circuit has **four connections**.
- Two are **power-supply** related.
  - Providing Vdd and Vss.
- One is an **input, A.**
- One is an **output, Q.**



inverter

Symbolic

A	Q
Vss	Vdd
Vdd	Vss

Voltage

A	Q
0 V	3.3 V
3.3 V	0 V

Binary

A	Q
0	1
1	0

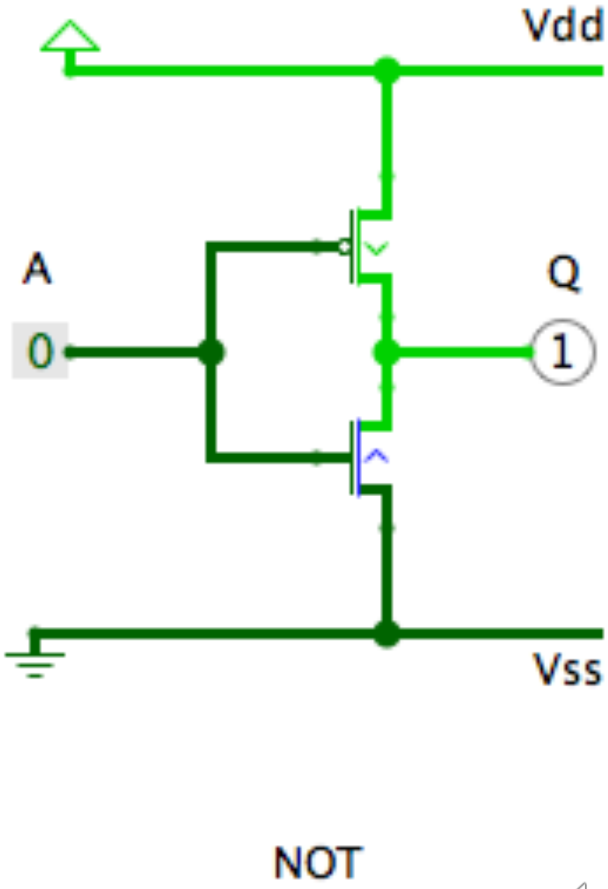
The CMOS inverter

Logisim design of a CMOS inverter for our next lab.

- You can download Logisim from:

http://sourceforge.net/projects/circuit/

- Install Logisim on your own computer and practice outside of lab hours.



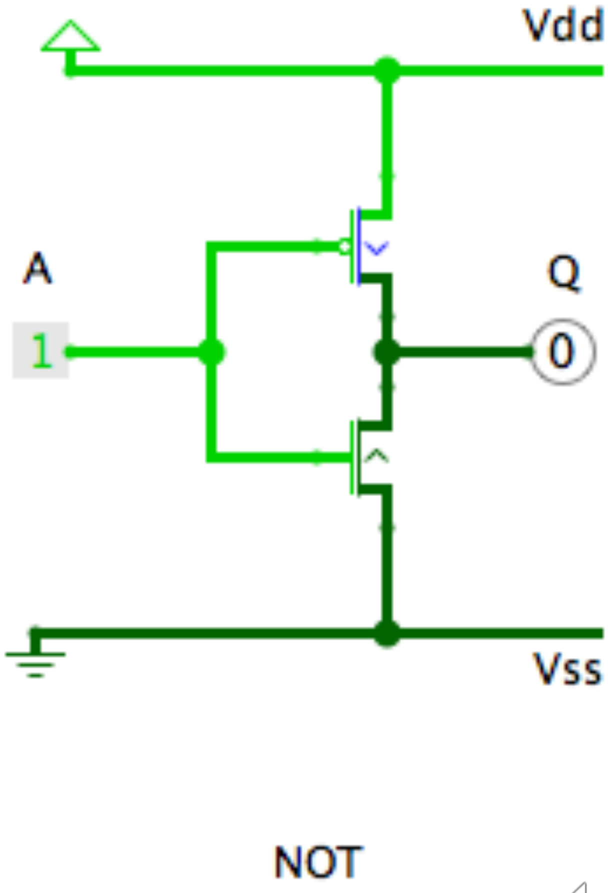
The CMOS inverter

Logisim design of a CMOS inverter for our next lab.

- You can download Logisim from:

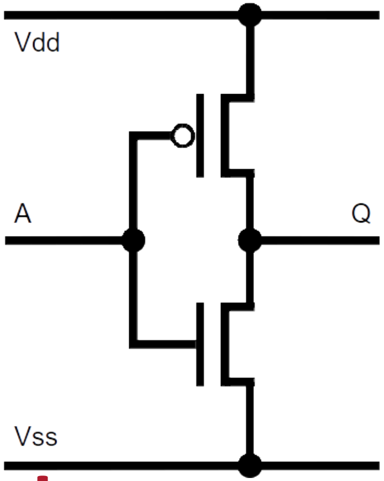
http://sourceforge.net/projects/circuit/

- Install Logisim on your own computer and practice outside of lab hours.



PMOS + NMOS = CMOS

- PMOS is good for making connections to **Vdd**. 1
- NMOS is good for making connections to **Vss**. 0
  - You can’t make a reliable switch (or inverter) with just one type.
  - We either get 1/? or ?/0; we want 1/0.
- So we use a **pair**, one PMOS, one NMOS.
- They are **complementary**.



CMOS

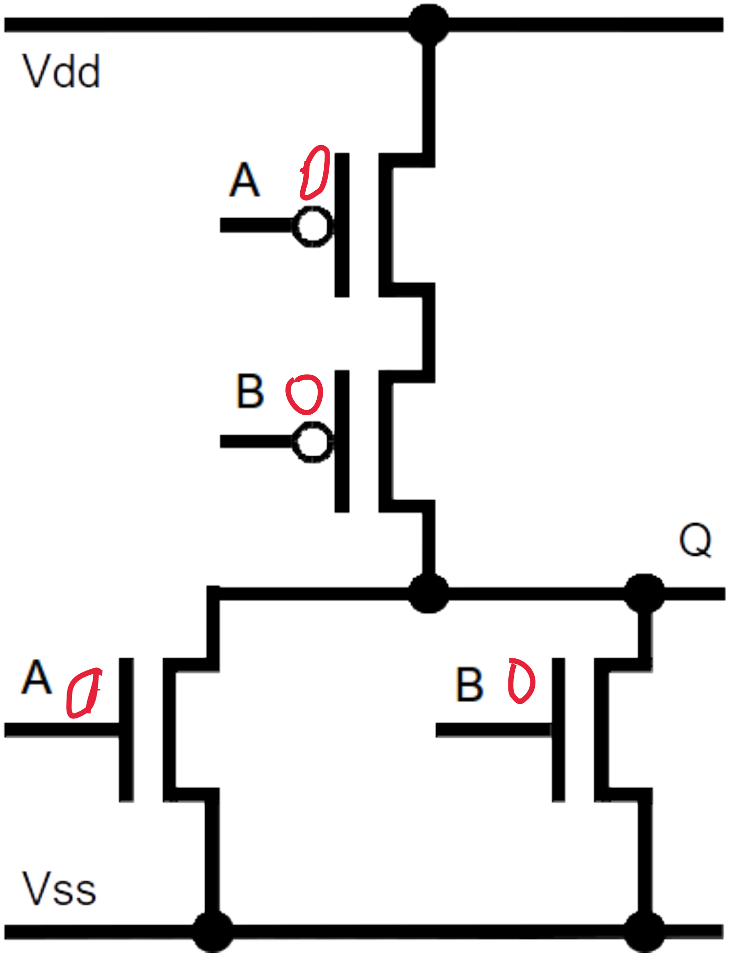
Complementary Metal Oxide Semiconductor

Boolean logic in CMOS

- We have a CMOS inverter.
  - Implements “not”
  - In a logic circuit, we call this a **NOT gate**.
- To build a more complex circuit, we need more than just a not gate.

So … what (else) can we make?

So … what (else) can we make?



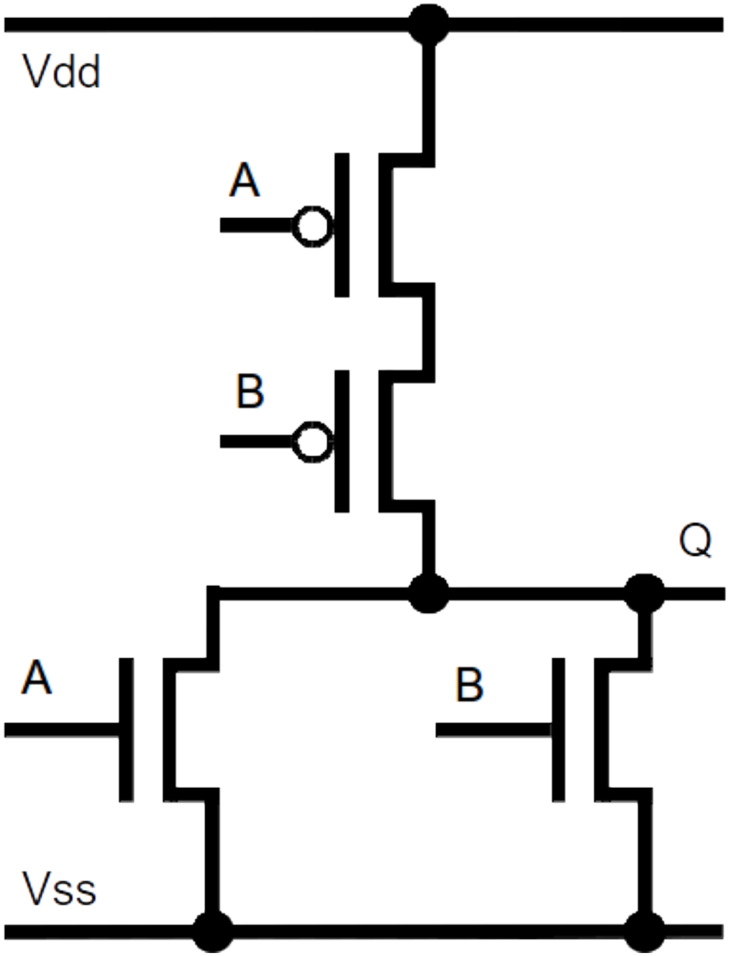
What is this?

How does it work?

NOR

A	B	Q
0	1	0
0	0	1
1	0	0
1	1	0

So … what (else) can we make?

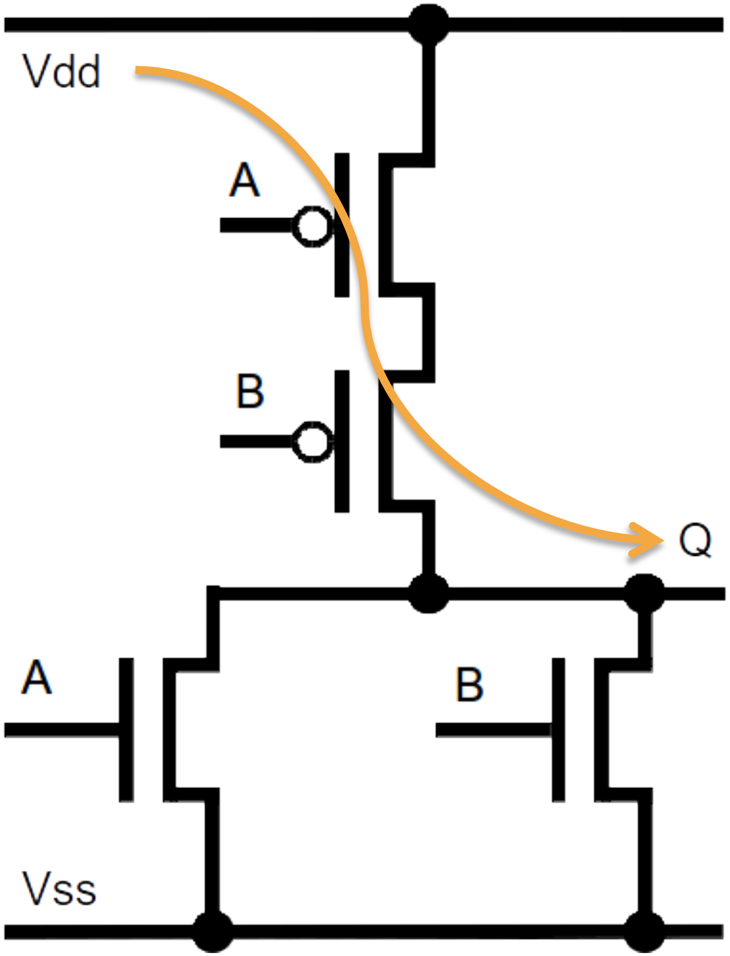


What is this?

How does it work?

Please pause to see whether you can work this out. Carry on only when you've tried.

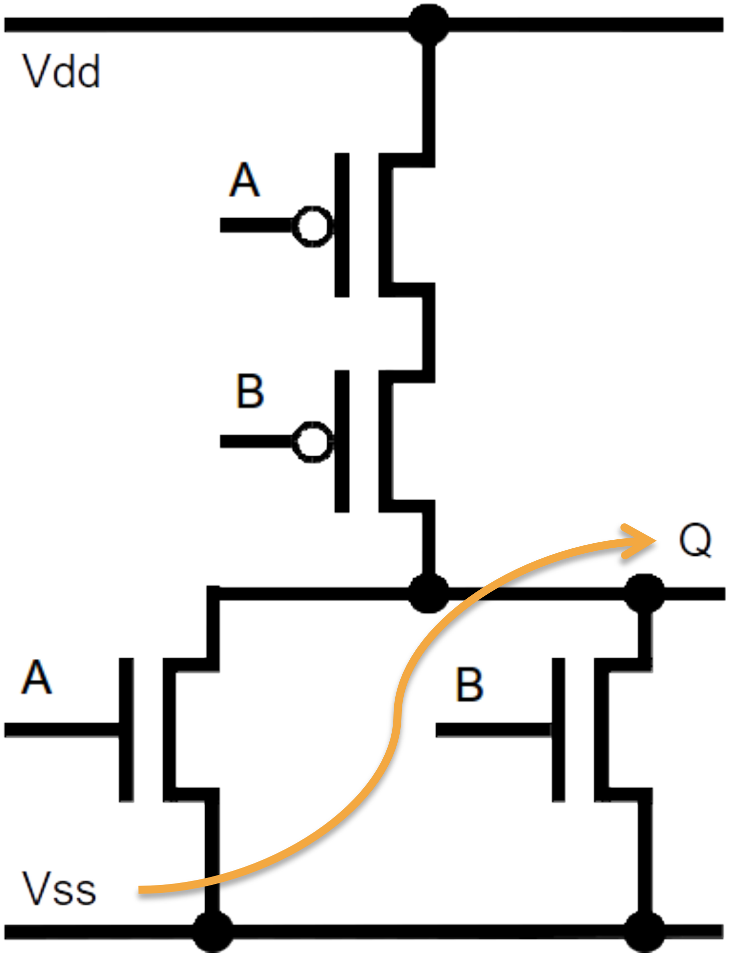
So … what (else) can we make?



What is this?

How does it work?

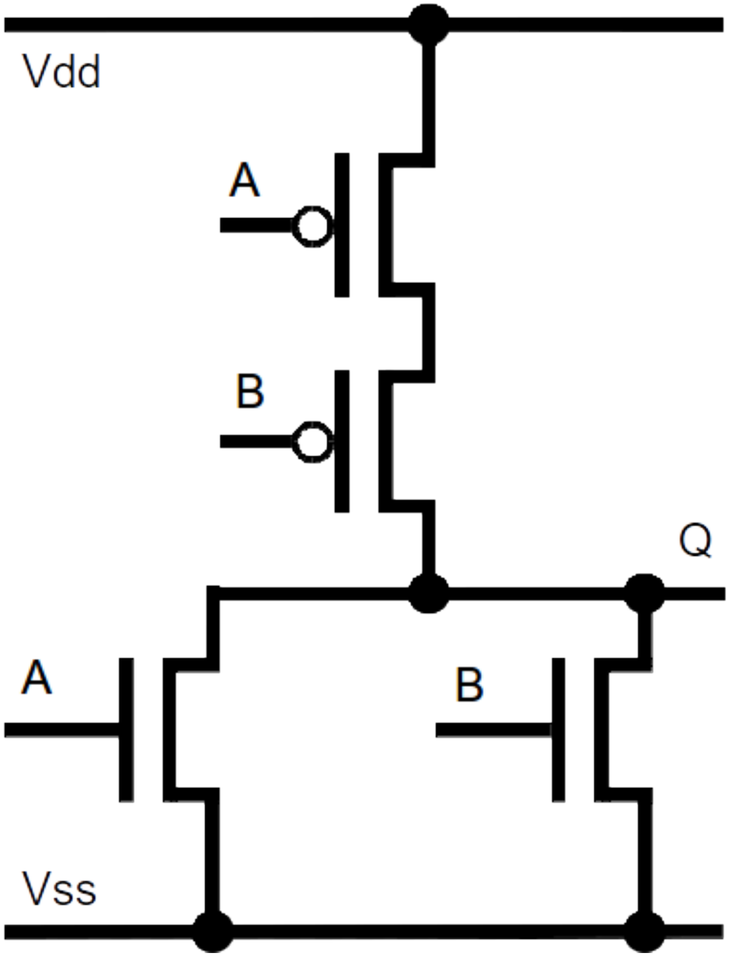
So … what (else) can we make?



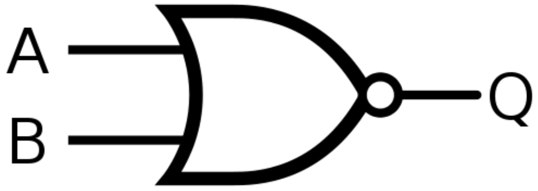
What is this?

How does it work?

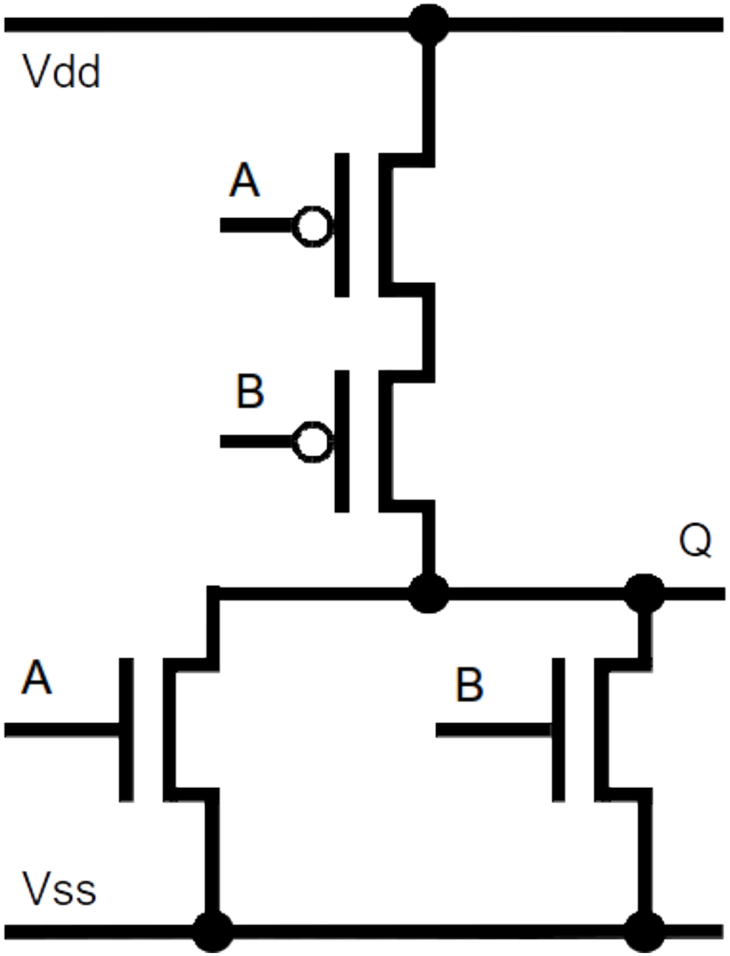
NOR (NOT-OR)



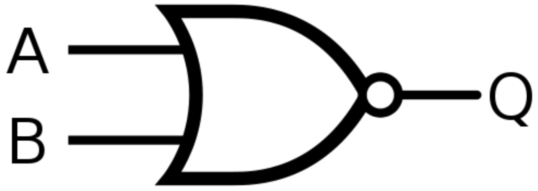
- This is a NOR gate.



NOR (NOT-OR)

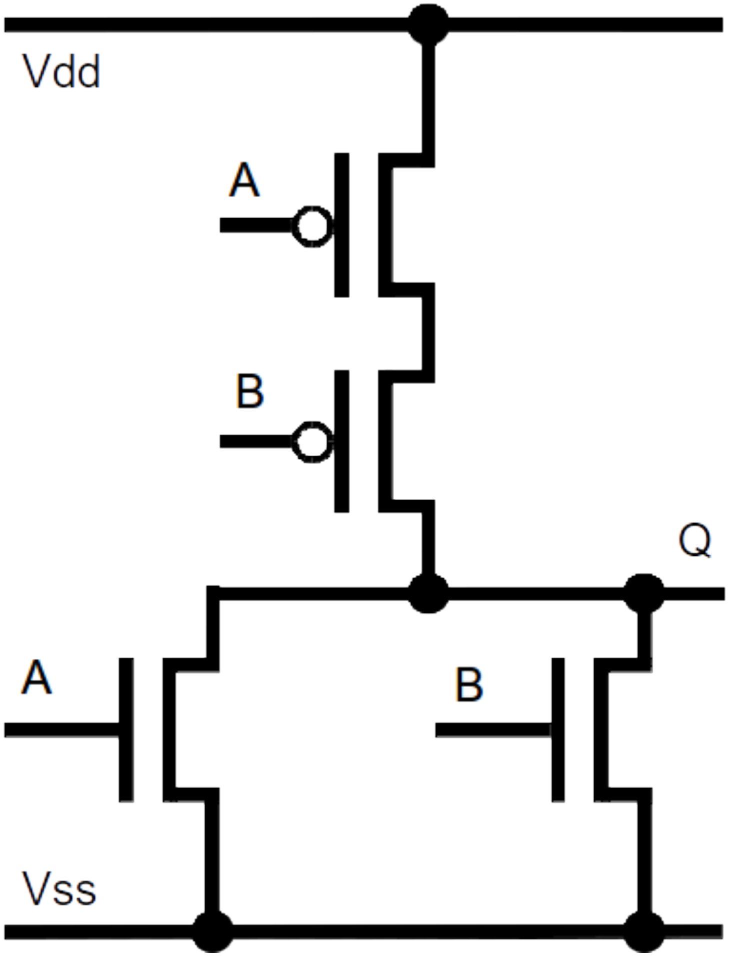


- This is a NOR gate.

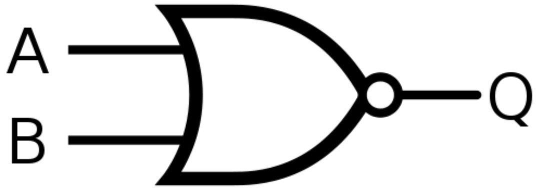


A	B	Q
0	0	
0	1	
1	0	
1	1	

NOR (NOT-OR)

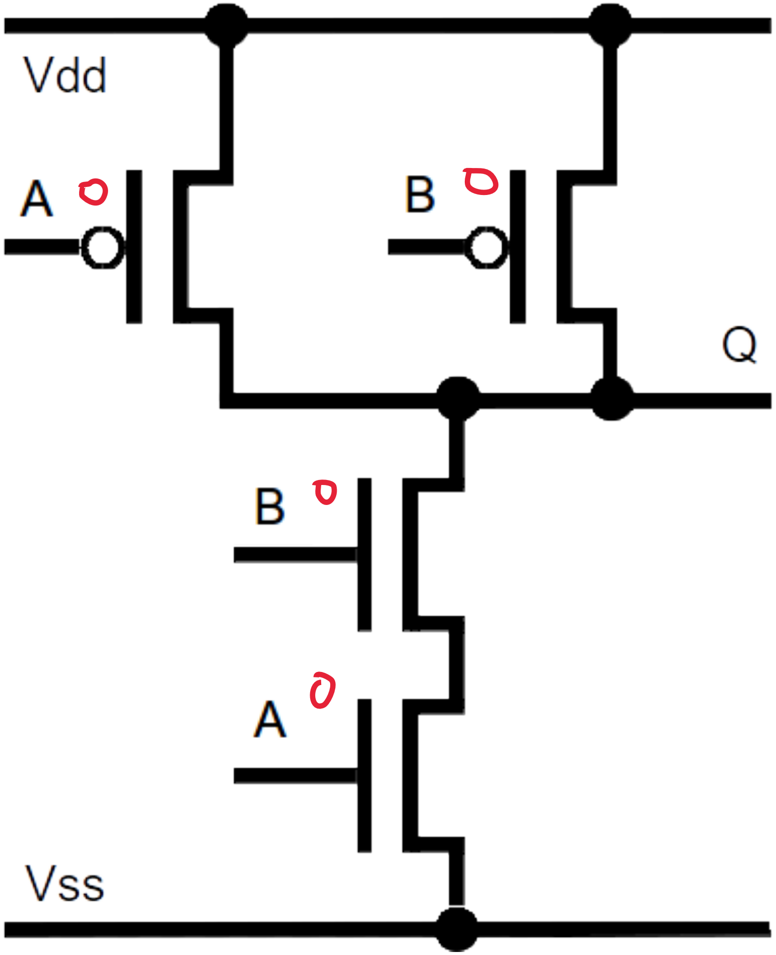


- This is a NOR gate.



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

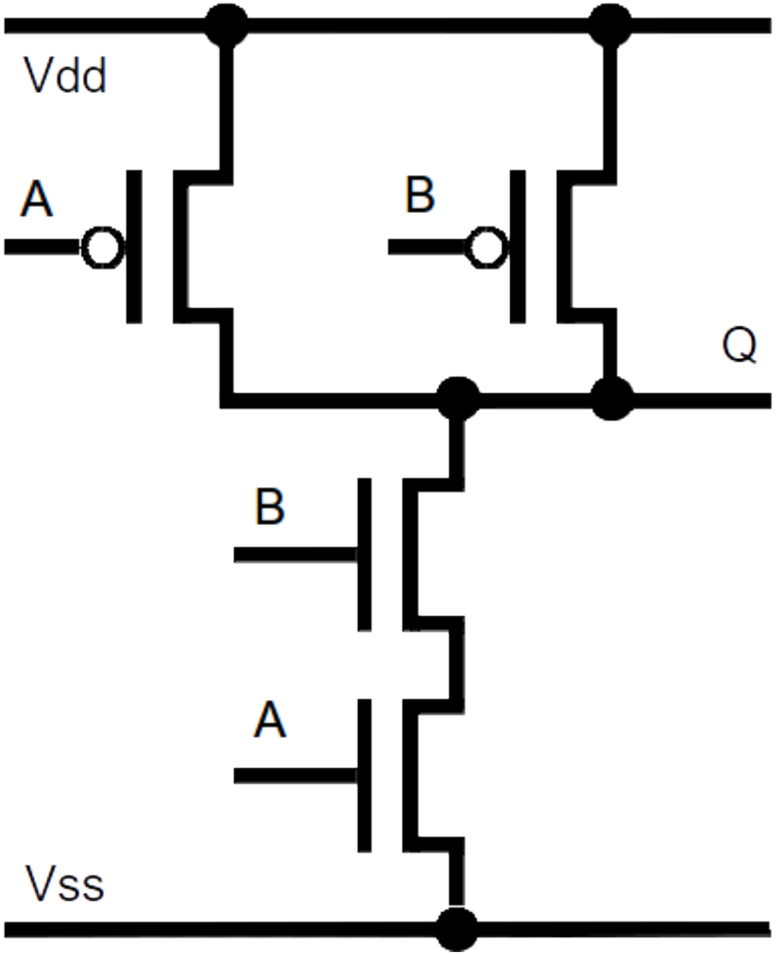
NAND (NOT-AND)



What is this?

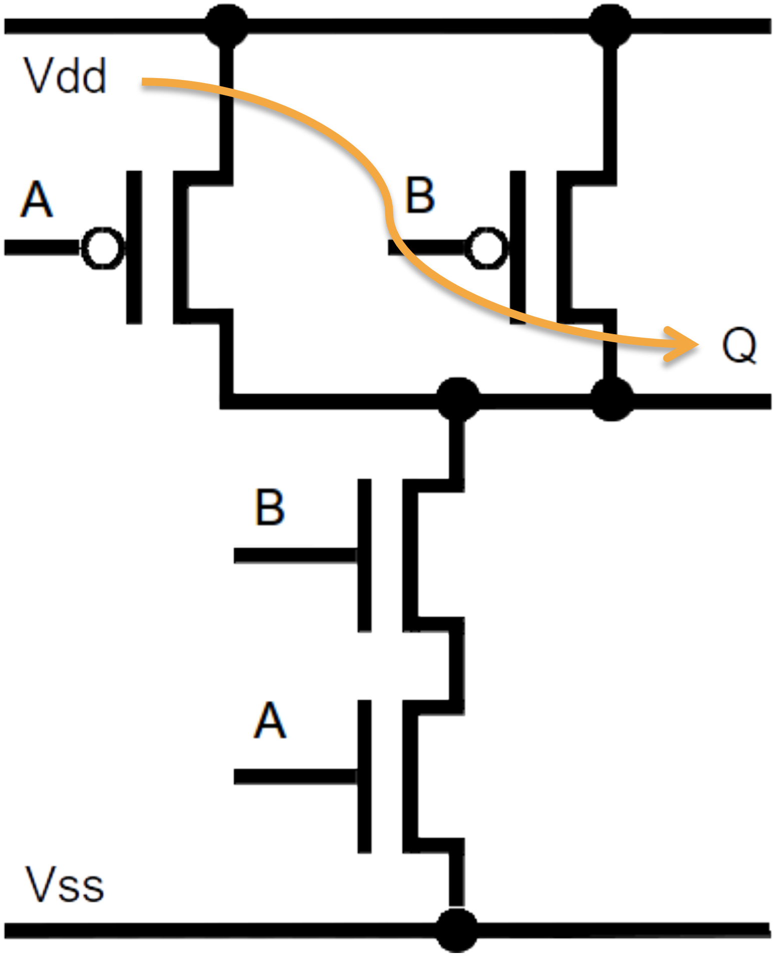
How does it work?

NAND (NOT-AND)

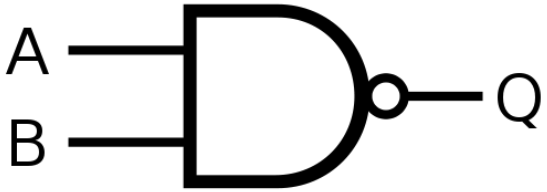


Can you work out under which conditions Q is connected to Vss?

NAND (NOT-AND)

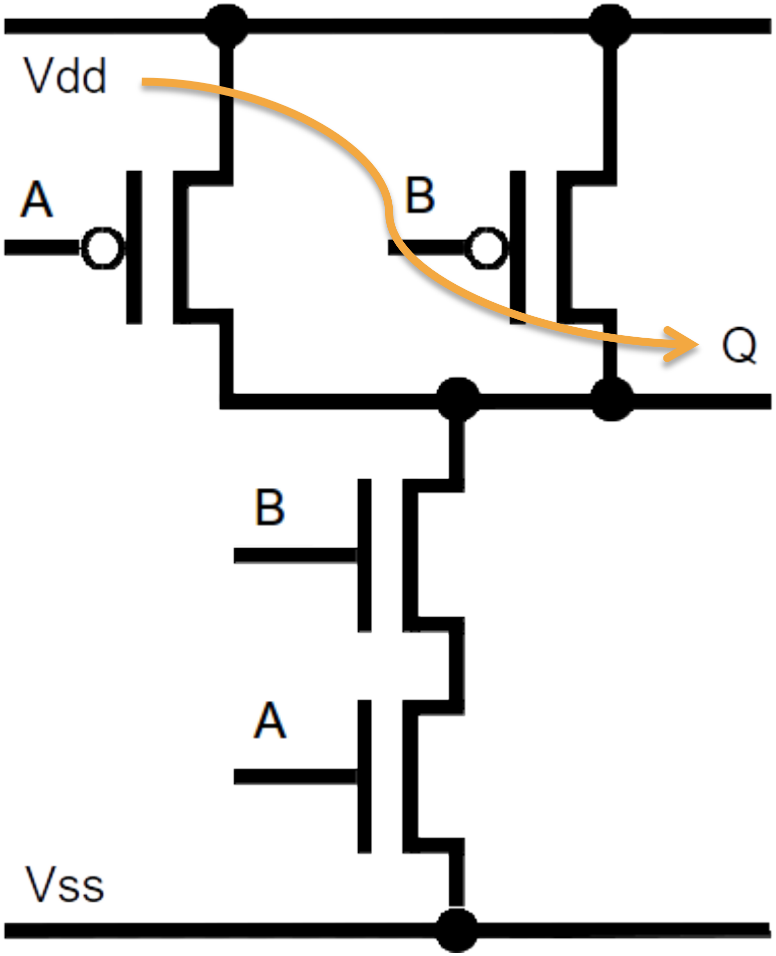


- We could also build a NAND gate.

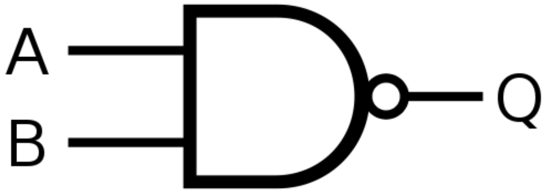


A	B	Q
0	0	
0	1	
1	0	
1	1	

NAND (NOT-AND)

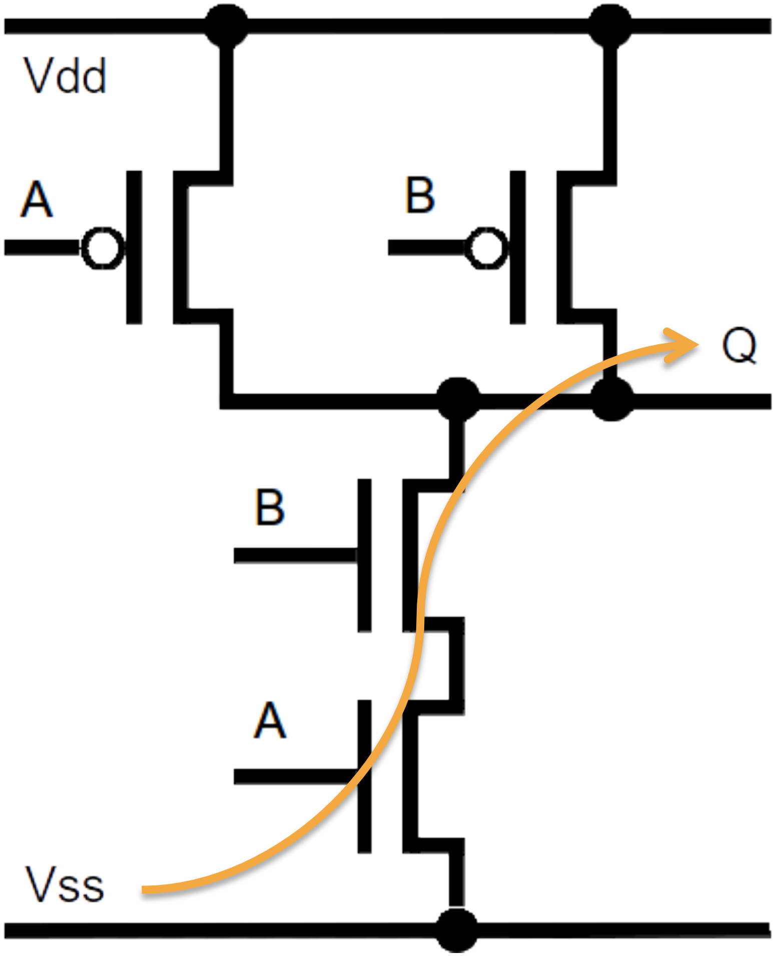


- We could also build a NAND gate.

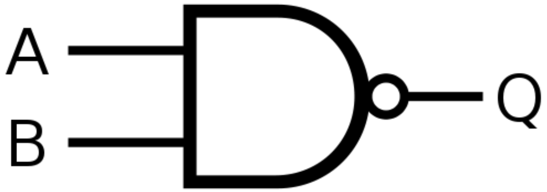


A	B	Q
0	0	1
0	1	1
1	0	1
1	1	

NAND (NOT-AND)

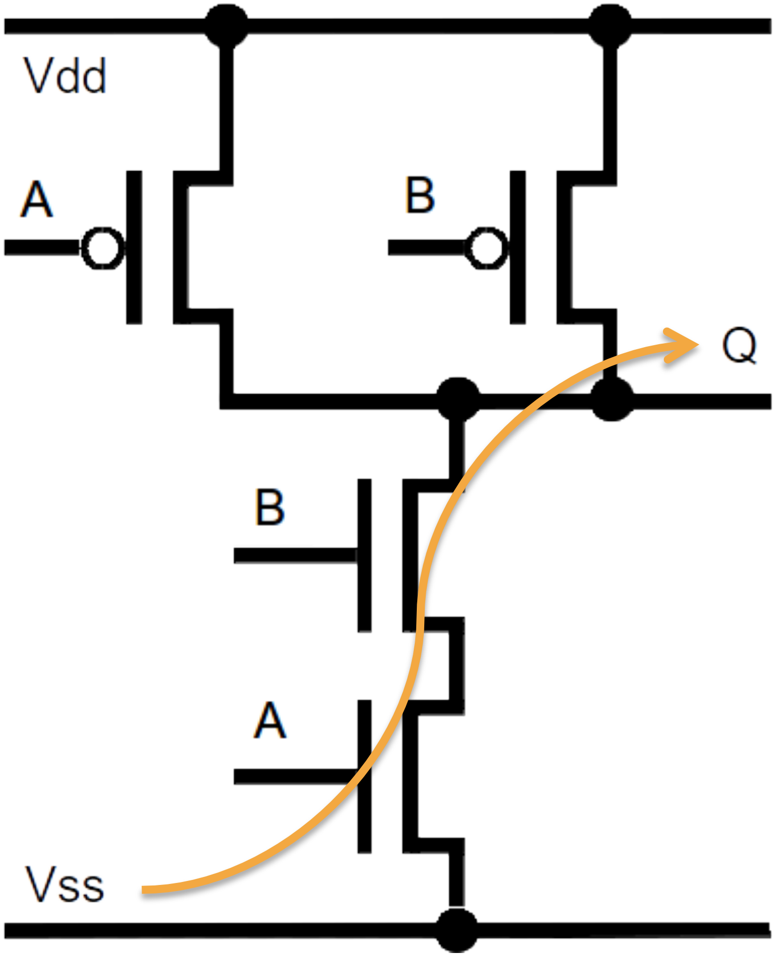


- We could also build a NAND gate.

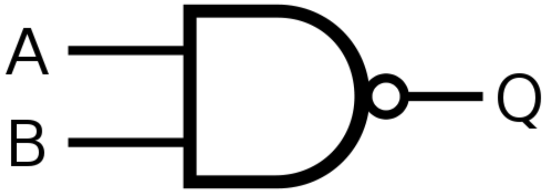


A	B	Q
0	0	1
0	1	1
1	0	1
1	1	

NAND (NOT-AND)

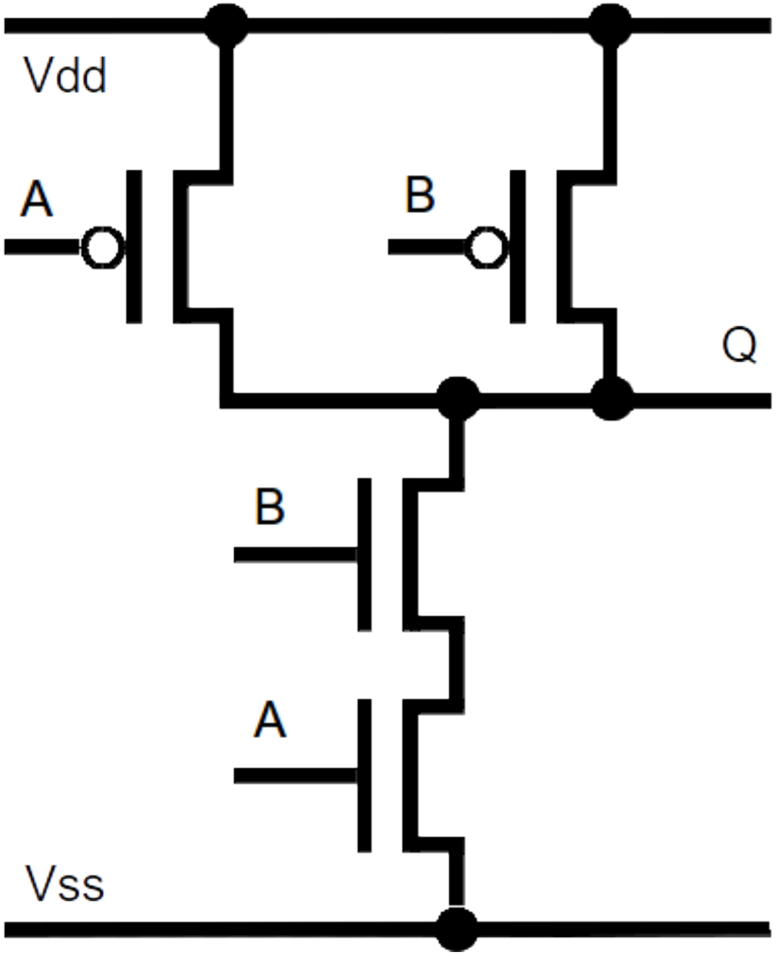


- We could also build a NAND gate.

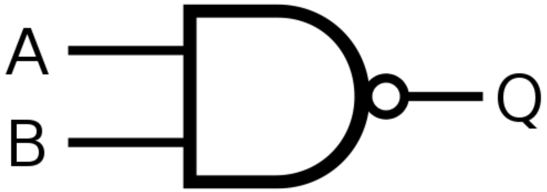


A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

NAND (NOT-AND)

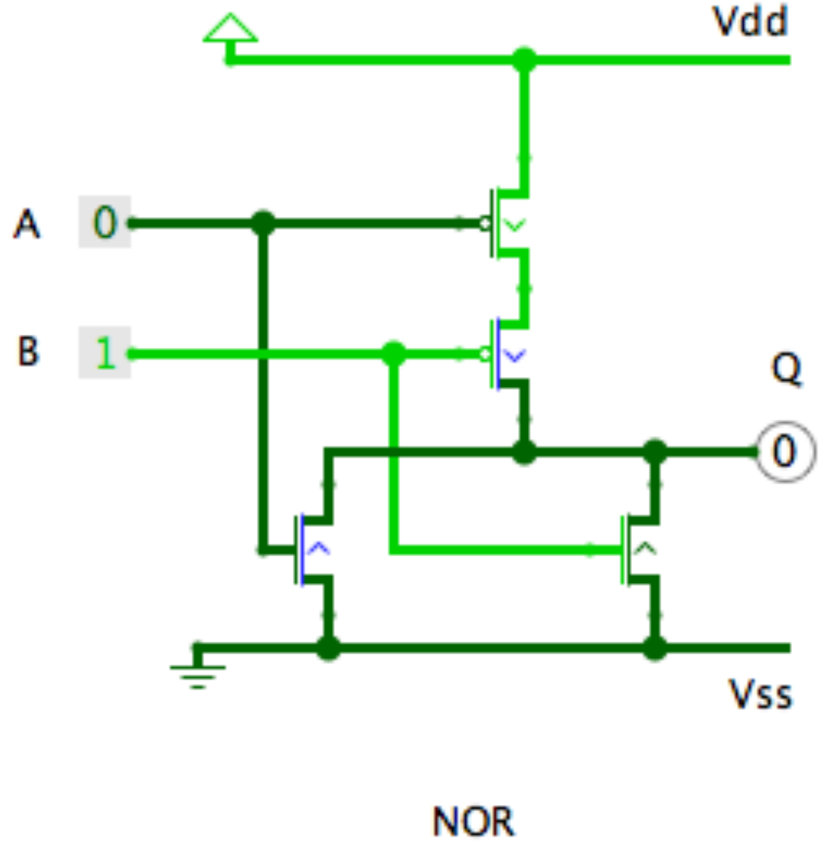


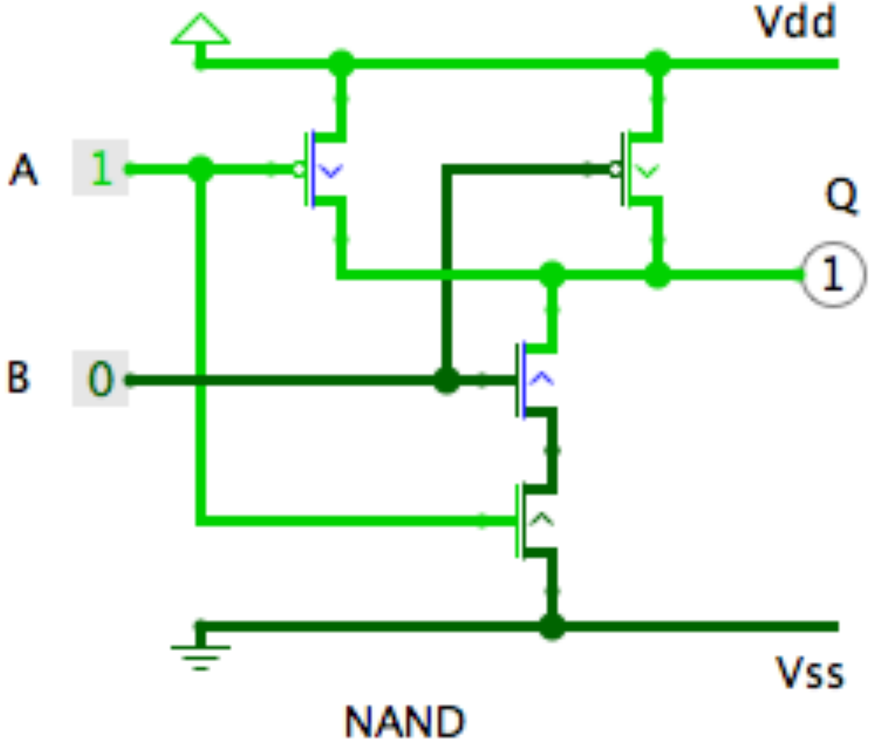
- We could also build a NAND gate.



A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

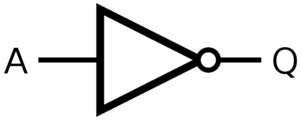
NOR and NAND in Logisim



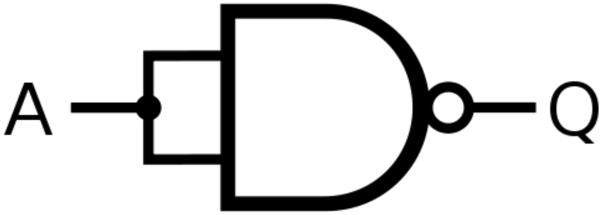


Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **NOT** 

Diagram



Boolean algebra

Q = ¬(A ∧ A)

- Idempotence axiom

Truth table

A	Q
0	1
1	0

Axioms in Boolean algebra

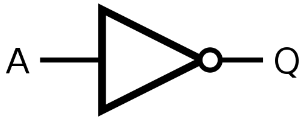
- Some rules help us avoid evaluating parts, because we can know the answer regardless of the values of the variables.

Rule	Axioms
Identity	$x \wedge 1 \equiv x$
Null	$x \wedge 0 \equiv 0$
Idempotence	$x \wedge x \equiv x$
Inverse	$x \wedge \neg x \equiv 0$

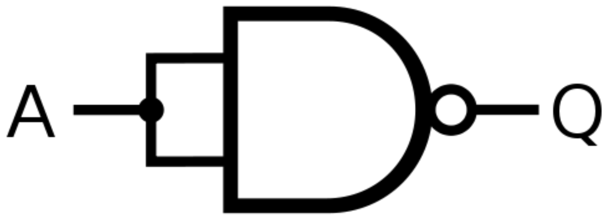
From Lecture 2 Boolean Algebra

- **Duality**: Swap 0s and 1s, conjunction and disjunction. Equivalence property is preserved.

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **NOT** 

Diagram



Boolean algebra

$Q = \neg(A \wedge A)$

- Idempotence axiom

$x \wedge x \equiv x$

$\neg(\underline{A \wedge A}) \equiv \neg(\underline{A}) \equiv \neg\underline{A}$

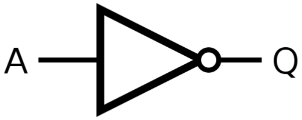
Truth table

A	Q
0	1
1	0

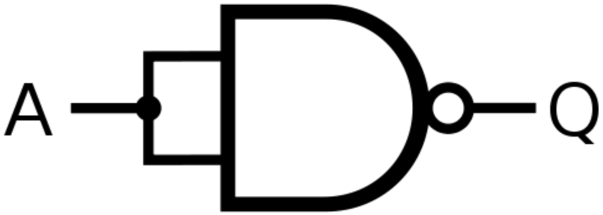
Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **NOT**



Diagram



Boolean algebra

Q = ¬(A ∧ A) = ¬A

- Idempotence axiom

  x ∧ x ≡ x

  ¬(<u>A ∧ A</u>) ≡ ¬(<u>A</u>) ≡ ¬<u>A</u>

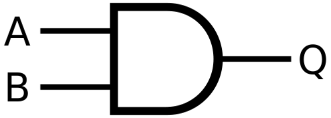
Truth table

A	**Q**
0	1
1	0

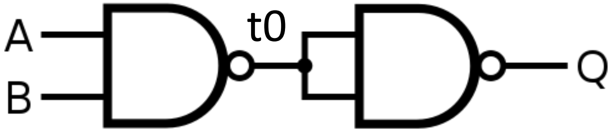
Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **AND**



Diagram



t0 = ¬(A ∧ B)

Algebra

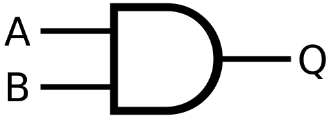
Q = ¬(t0)

Truth table

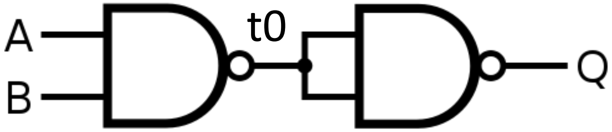
A	B	t0	Q
0	0		
0	1		
1	0		
1	1		

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **AND**



Diagram



t0 = ¬(A ∧ B)

Algebra

Q = ¬(¬(A ∧ B))

= A ∧ B

- Double negation axiom

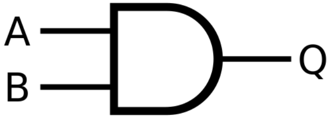
¬(¬X) ≡ ¬¬X ≡ X

Truth table

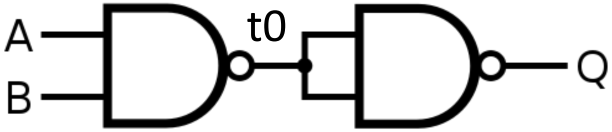
A	B	t0	Q
0	0		
0	1		
1	0		
1	1		

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **AND**



Diagram



Algebra

Q = ¬(¬(A ∧ B))

= A ∧ B

- Double negation axiom

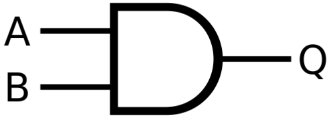
¬(¬X) ≡ ¬¬X ≡ X

Truth table

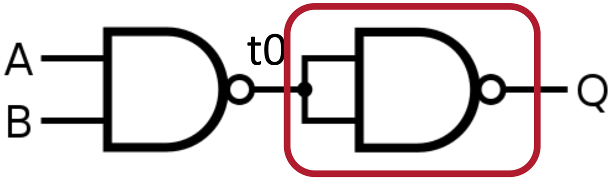
A	B	t0	Q
0	0	1	
0	1	1	
1	0	1	
1	1	0	

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **AND**



Diagram



NOT gate

Algebra

$$Q = \neg(\neg(A \land B))$$
$$= A \land B$$

- Double negation axiom

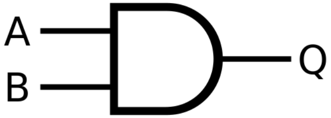
$$\neg(\neg X) \equiv \neg\neg X \equiv X$$

Truth table

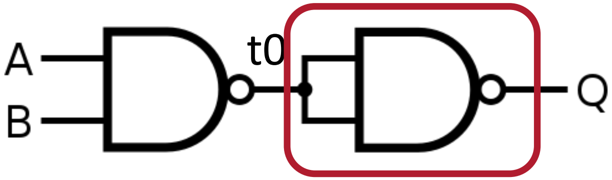
A	B	t0	Q
0	0	1	
0	1	1	
1	0	1	
1	1	0	

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **AND**



Diagram



NOT gate

Algebra

$Q = \neg(\neg(A \land B))$

$= A \land B$

- Double negation axiom

$\neg(\neg X) \equiv \neg\neg X \equiv X$

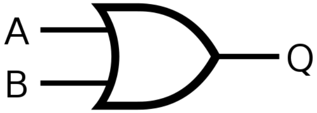
Truth table

A	B	t0	Q
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

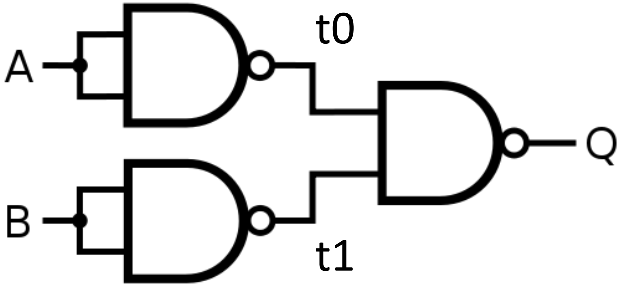
Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **OR**



Diagram



Algebra

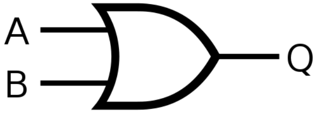
Q =

Truth table

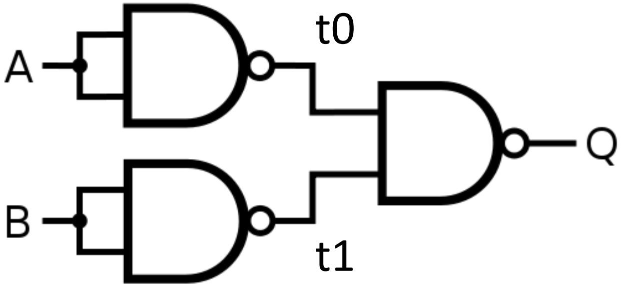
A	B	t0	t1	Q
0	0			
0	1			
1	0			
1	1			

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **OR**



Diagram



Algebra

$Q = \neg(t0 \land t1)$

$t0 = \neg(A \land A)$

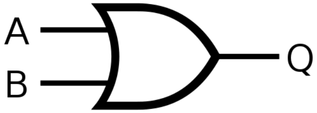
$t1 = \neg(B \land B)$

Truth table

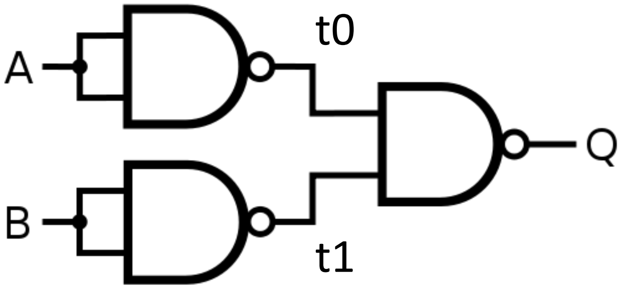
A	B	t0	t1	Q
0	0			
0	1			
1	0			
1	1			

Why NAND?

- NAND is an excellent **building block** for other Boolean logic.
- For example: **OR**



Diagram



Algebra

$$\begin{aligned} Q &= \neg(\neg(A \land A) \land \neg(B \land B)) \\ &= \neg(\neg A \land \neg B) \\ &= \neg(\neg(A \lor B)) \\ &= A \lor B \end{aligned}$$

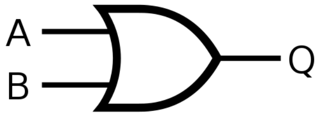
Truth table

A	B	t0	t1	Q
0	0			
0	1			
1	0			
1	1			

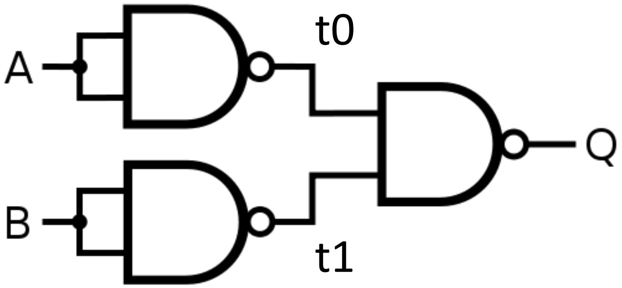
Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **OR**



Diagram



Algebra

$$
\begin{aligned}
Q &= \neg(\neg(A \land A)\land\neg(B \land B)) \\
&= \neg(\neg A \land\neg B) \\
&= \neg(\neg(A \lor B)) \\
&= A \lor B
\end{aligned}
$$

- Idempotence, de Morgan and double negation

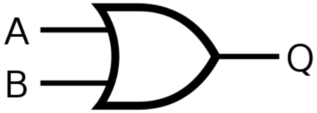
Truth table

A	B	t0	t1	Q
0	0			
0	1			
1	0			
1	1			

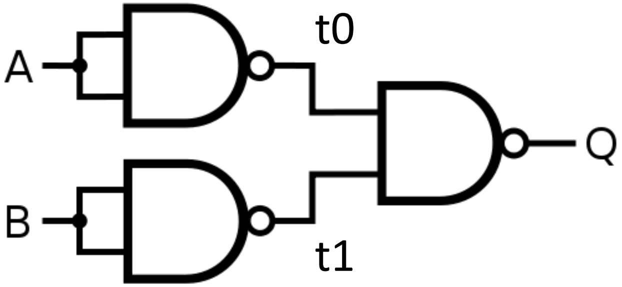
Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **OR**



Diagram



Algebra

$$Q = \neg(\neg(A \wedge A) \wedge \neg(B \wedge B))$$
$$= \neg(\neg A \wedge \neg B)$$
$$= \neg(\neg(A \vee B))$$
$$= A \vee B$$

- Idempotence, de Morgan and double negation

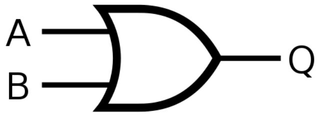
Truth table

A	B	t0	t1	Q
0	0	1	1	
0	1	1	0	
1	0	0	1	
1	1	0	0	

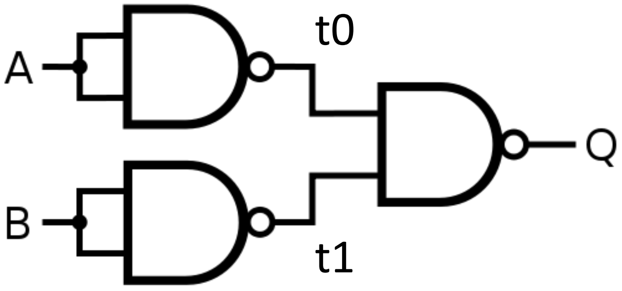
Why NAND?

- NAND is an excellent **building block** for other Boolean logic.

- For example: **OR**



Diagram



Algebra

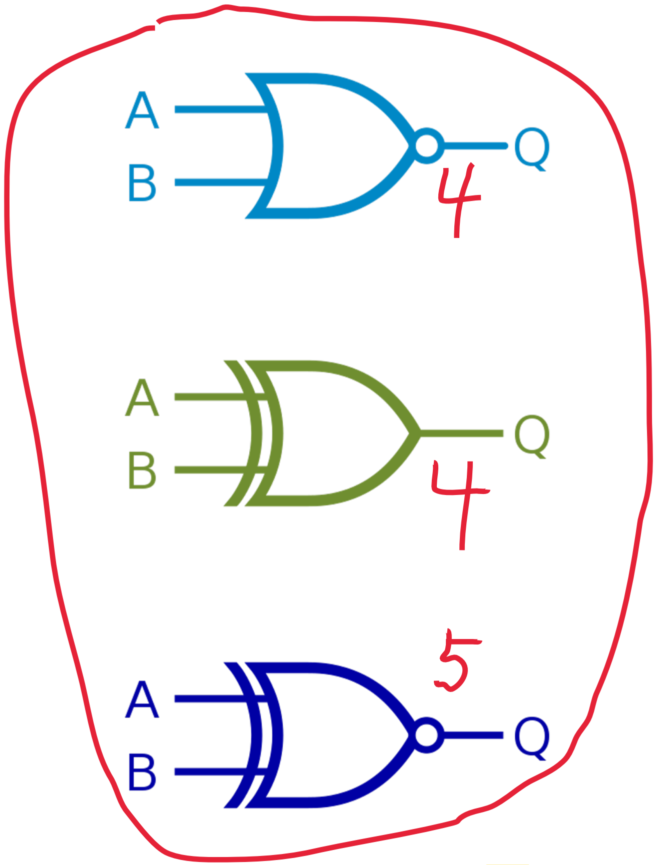
$$Q = \neg(\neg(A \land A) \land \neg(B \land B))$$
$$= \neg(\neg A \land \neg B)$$
$$= \neg(\neg(A \lor B))$$
$$= A \lor B$$

- Idempotence, de Morgan and double negation

Truth table

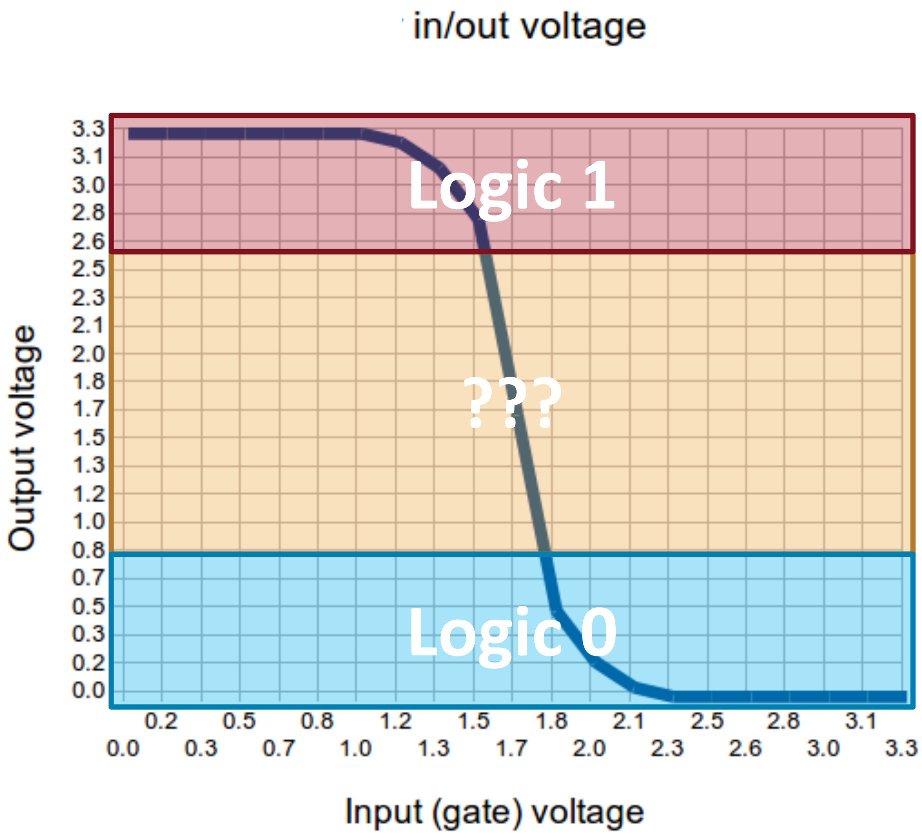
A	B	t0	t1	Q
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

Why NAND?



- NAND is an excellent **building block** for other Boolean logic.
- **NAND is functionally complete.**
  - All gates can be expressed with NAND gates arranged in various ways.
- **NOT, AND, OR** as previously shown.
- As well as **NOR**, **XOR**, **XNOR**.
  - **NOR** is also functionally complete.
  - **NOR** structure is slower than NAND.

Voltages and logic levels



- Simplified view so far.
- **Ideal** would be a **right-angled** response.
- Steepness and position of curve depend on transistor **properties**.
- There is also a **delay** between a change in input producing a change in output.
- This delay accounts for the **physical signal propagation time** in silicon.

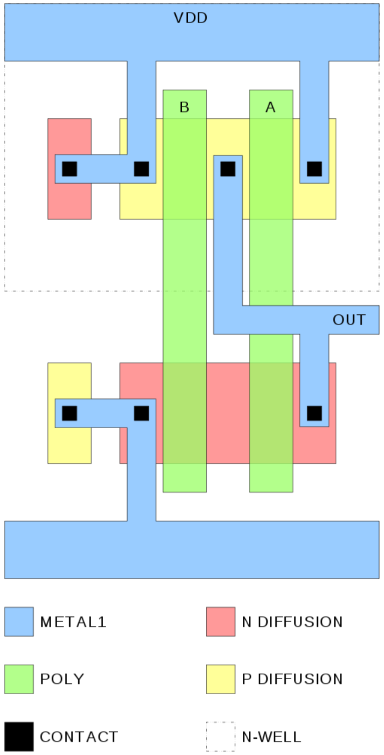
COMSM1302 NAND Lab 1-3

- NAND boards have either been sent to you or are ready for you to collect.
- The NAND boards are for you to work on your lab sheets during scheduled lab hours or on your own.
- **Please be careful with the NAND boards. They have sharp edges and are sensitive electronic devices.**

Summary

- Switches
  - Mechanical
  - Thermionic
  - Silicon
- Transistors
- CMOS
  - Inverter
  - NAND (functionally complete)
  - Making other gates from NAND
- More about how chips are made:

  https://www.youtube.com/watch?v=qm67wbB5GmI
  https://www.youtube.com/watch?v=4FLBtQC0F0c
  https://www.youtube.com/watch?v=i8kxymmjdoM
  and this is also quite cool:
  https://www.youtube.com/watch?v=Y33cf-lcq-g



The physical layout of a NAND circuit. [Source: By Jamesm76 at English Wikipedia - Transferred from en.wikipedia to Commons., Public Domain, https://commons.wikimedia.org/w/index.php?curid=2944[illegible]2]

In this lecture

Foundations	Data representation, logic, Boolean algebra.
Building blocks	**Transistors, transistor based logic**, simple devices, storage.
Modules	Memory, simple controllers, FSMs, processors and execution.
Programming	Machine code, assembly, high-level languages, compilers.
Wrap-up	Operating systems, energy aware computing.

In the next lecture

Foundations	Data representation, logic, Boolean algebra.
Building blocks	Transistors, transistor based logic, **simple devices**, storage.
Modules	Memory, simple controllers, FSMs, processors and execution.
Programming	Machine code, assembly, high-level languages, compilers.
Wrap-up	Operating systems, energy aware computing.