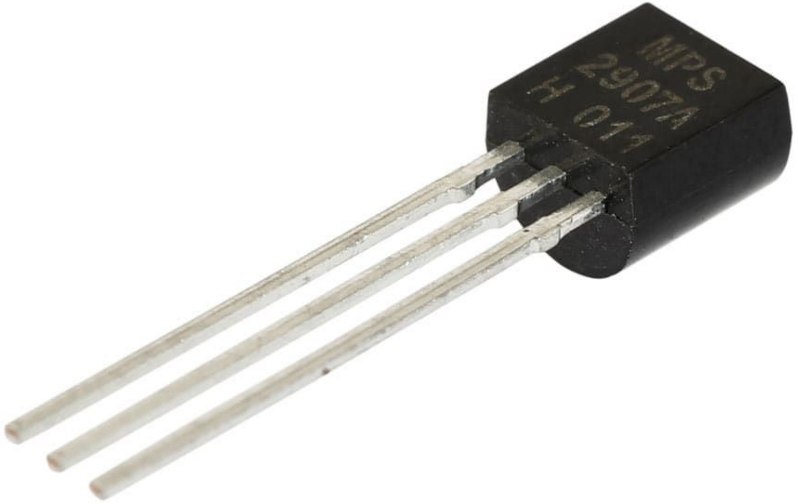
MONTANA
STATE UNIVERSITY

Transistors & Logical Gates

Getting Down to The Metal

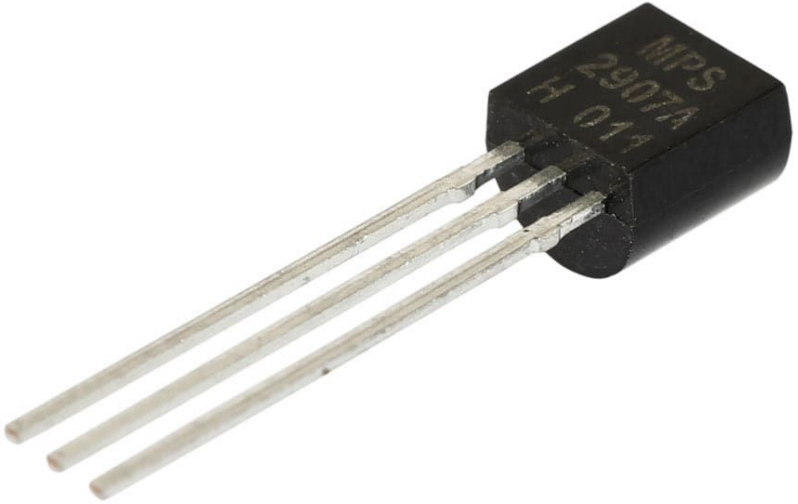
Hardware

- OK, it's time to get down to brass tacks
- We are going to look at how computers work at the physical level
- We will start by looking at *transistors*



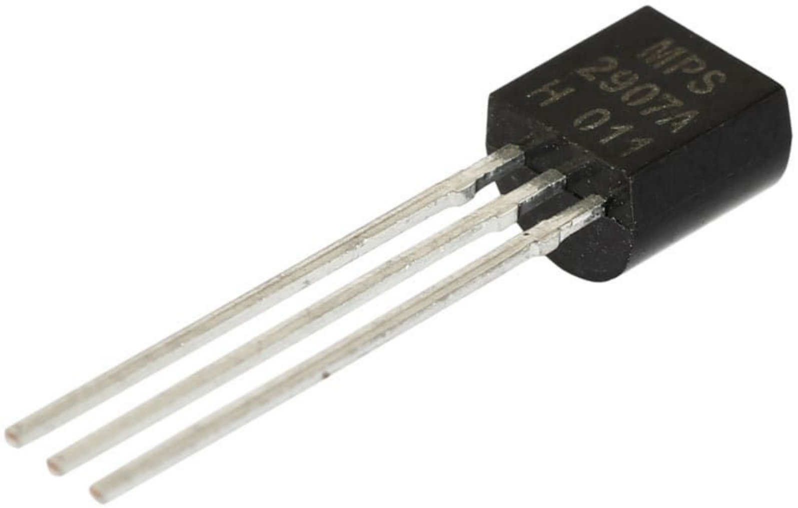
Hardware

- Note: I am not an electrical engineer
- I do not expect you to be an electrical engineer
- However, I do expect you to understand a bit about the underlying hardware in your computer



Transistors

- Pictured right is a *transistor*
- A transistor is semiconductor device that can be used to amplify or switch electronic signals
  - We will be concerned with switching, since we are discussing computer hardware



Transistor History

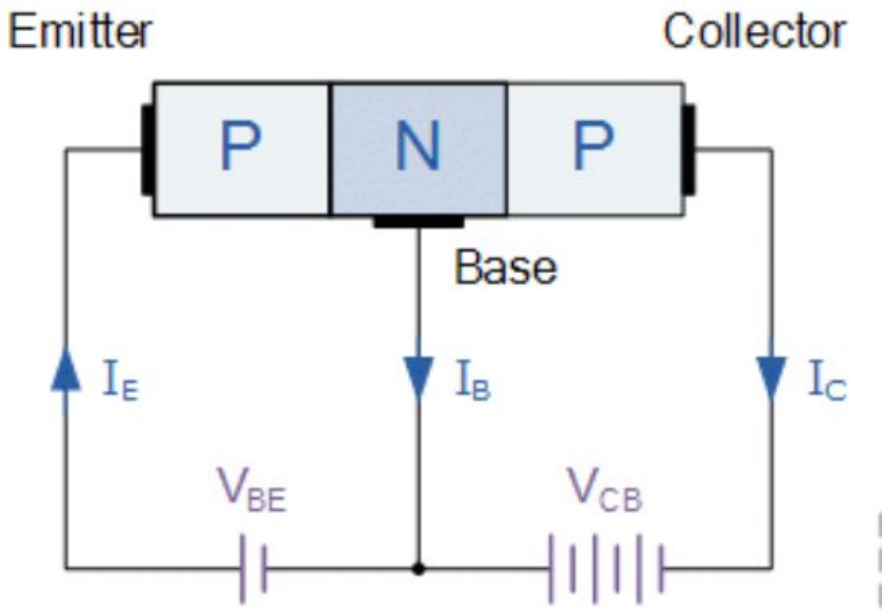
- Proposed in 1926 by Austrian Julius Lilienfeld
- First working transistor built at Bell Labs in 1947
  - John Bardeen
  - Walter Brattain
  - William Shockley

Transistor History

- Proposed in 1926 by Austrian Julius Lilienfeld
- First working transistor built at Bell Labs in 1947
    - John Bardeen
    - Walter Brattain
    - William Shockley

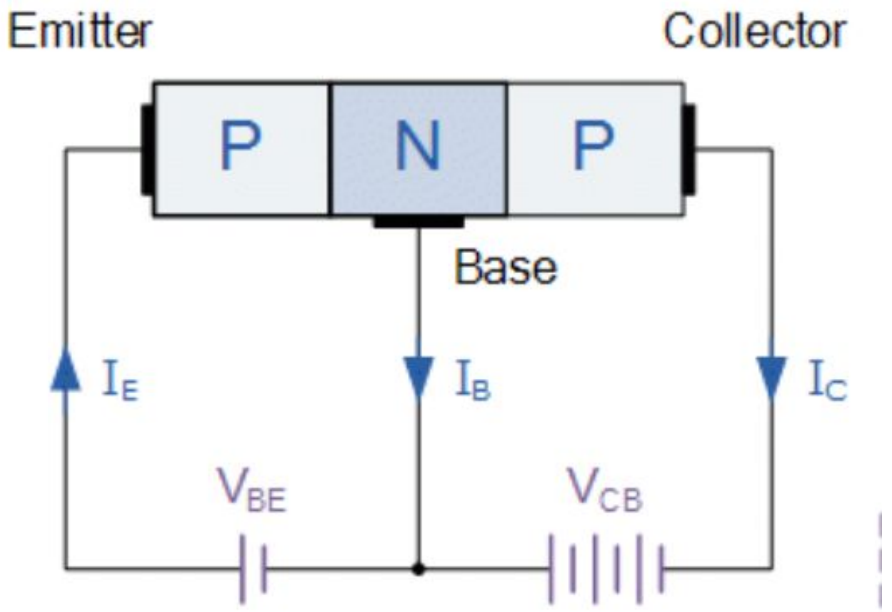
Transistor

- The Basic *Digital Transistor* Concept
  - Three input wires
    - Signal In 1 (Collector)
    - Signal In 2 (Base)
    - Signal Out (Emitter)
  - When a current is applied to the base wire, current will flow from the Collector to the Emitter wire
    - Or *not flow*, depending on the transistor type



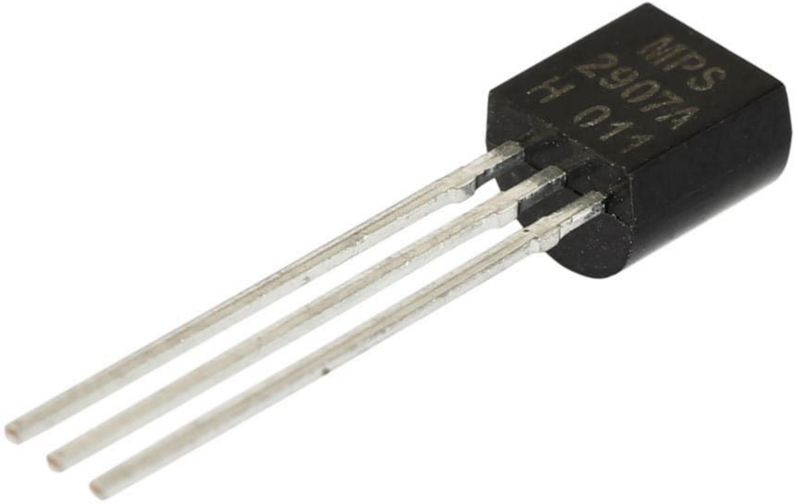
Transistor

- This is accomplished via materials engineering that makes the N region, picture here conductive when a signal is received from the base wire



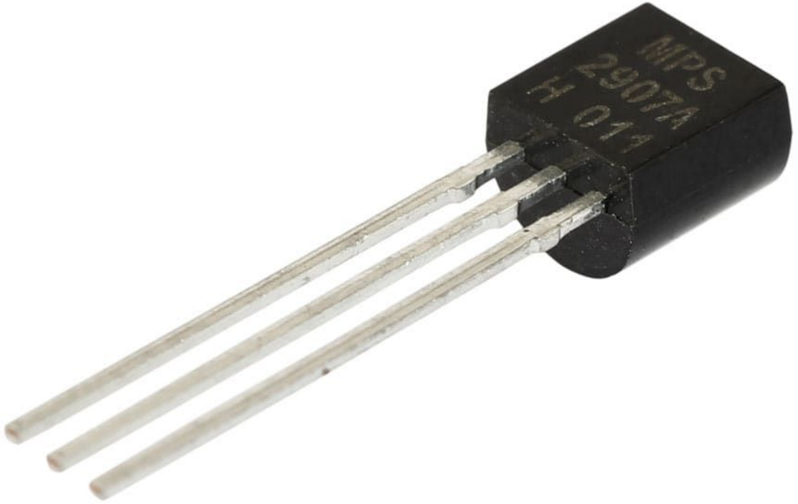
Transistor History

- Transistors then can be used as the basis for building digital systems
  - High (or low) voltage indicates a 1
  - How (or high) voltage indicates a 0



Logical Gates

- Using this idea we can create something called a *NAND gate*
- But first, let us review our logical operations and truth tables



AND

- By convention
  - 1 as high voltage = True
  - 0 as low voltage = False
- Recall the bitwise AND operator:

  1 if A and B are both 1

A && B

A	B	(A ∧ B)
F	F	**F**
F	T	**F**
T	F	**F**
T	T	**T**

OR

- The bitwise OR operator:

  1 if either A or B are 1

```
A || B
```

A	B	(A ∨ B)
F	F	**F**
F	T	**T**
T	F	**T**
T	T	**T**

NOT

- The bitwise NOT operator:

  1 if A is 0
  0 if A is 1

```
!A
```

A	¬A
F	**T**
T	**F**

NAND

- The bitwise NOT AND operator:

  1 if either A or B are 0

```
!A || !B
```

A	B	(¬A ∨ ¬B)
F	F	**T**
F	T	**T**
T	F	**T**
T	T	**F**

NAND

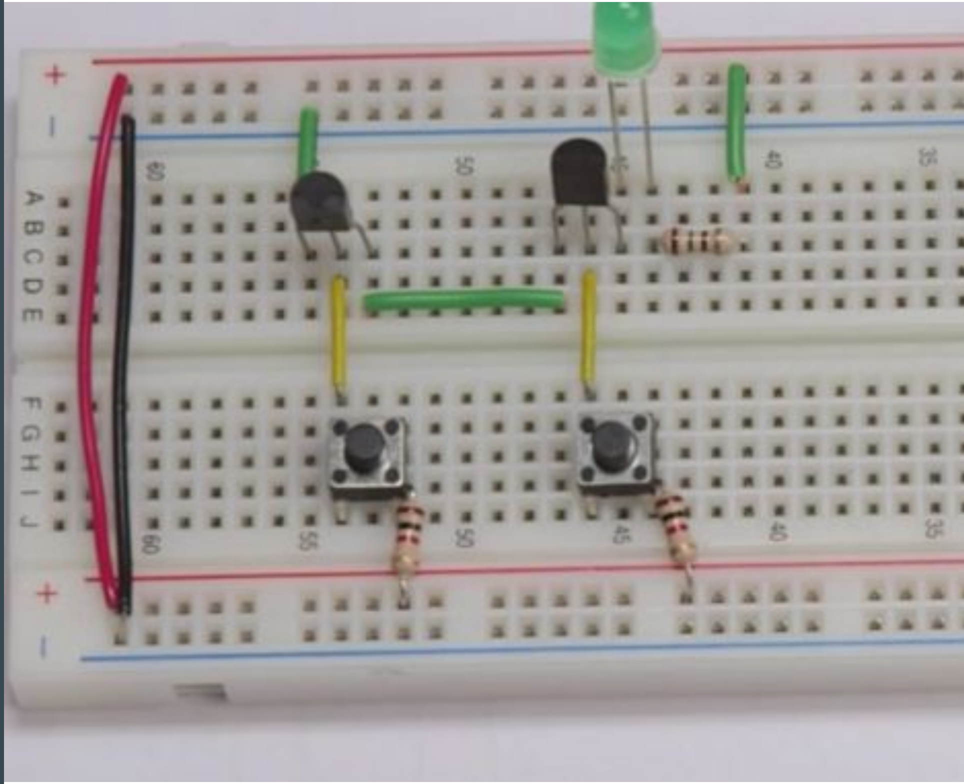
- NAND is very important
- It turns out that *all other logical operations* can be created using a NAND operator
    - This is also true of NOR
    - NAND and NOR are *functionally complete*
    - More on this gate type in a bit

```
!A || !B
```

A	B	(¬A ∨ ¬B)
F	F	**T**
F	T	**T**
T	F	**T**
T	T	**F**

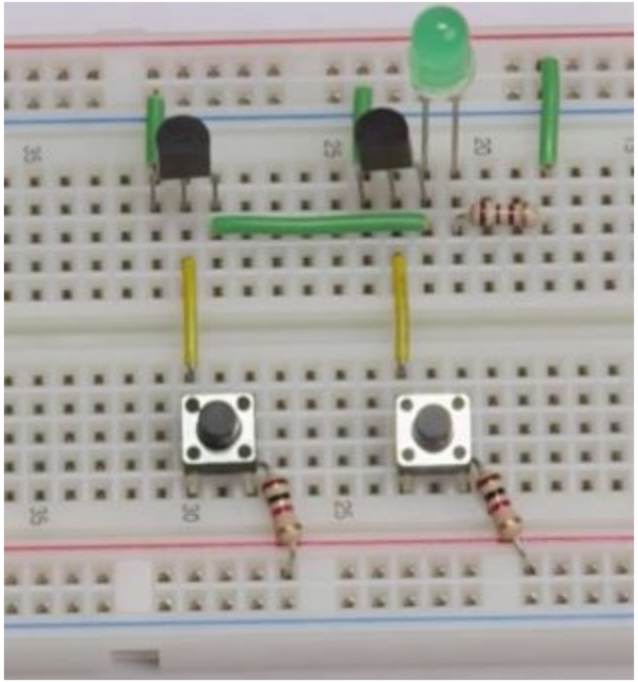
Transistor Implementations

- Using transistors, we can implement these logical operations
- Pictured right is a logical AND
- If both buttons are pressed, currency will flow through the transistors
- This allows electricity to flow through the led, lighting it up



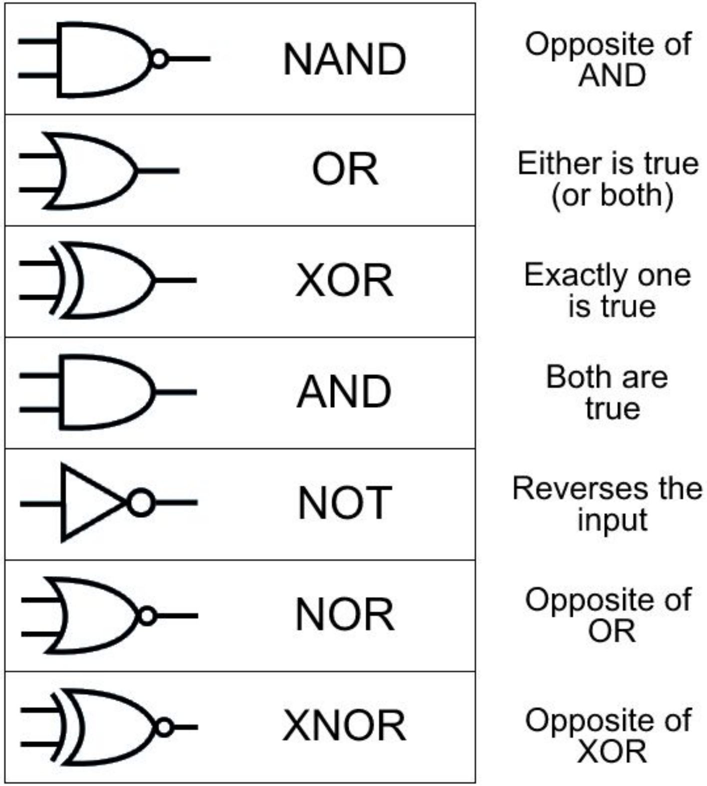
Transistor Implementations

- Here is an example of an OR
- If no button is pressed, no current is available to the LED
- If either button is pressed, currency is available to the LED



Logical Symbols

- When designing digital systems, symbols are used for the various logical gates
  - NAND
  - OR
  - XOR
  - AND
  - NOT
  - NOT
  - XNOR



NAND Gate

- Recall that I said that NAND gates are special
  - NAND gates are functionally complete
  - All other gates can be constructed with them

Q = **A** NAND **B**

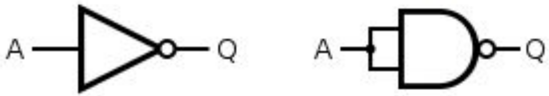
Truth Table

Input A	Input B		Output Q
0	0		1
0	1		1
1	0		1
1	1		0

NOT Gate

- To create a NOT, simply put *both* inputs into a NAND

Desired NOT Gate | **NAND Construction**



Q = NOT(**A**) | = **A** NAND **A**

Truth Table

Input A	Output Q
0	1
1	0

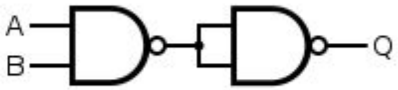
AND Gate

- AND gate construction
- Easy: NOT (A NAND B)

Desired AND Gate

Q = **A** AND **B**

NAND Construction



= (**A** NAND **B**) NAND (**A** NAND **B**)

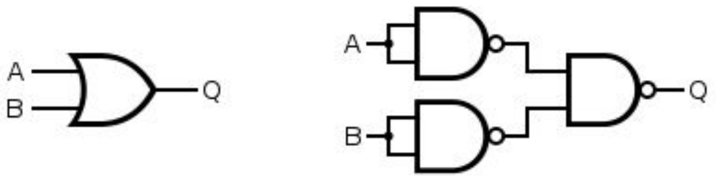
Truth Table

Input A	Input B		Output Q
0	0		0
0	1		0
1	0		0
1	1		1

OR Gate

- A little more complicated
- (NOT A) NAND (NOT B)

Desired OR Gate | **NAND Construction**



Q = **A** OR **B**

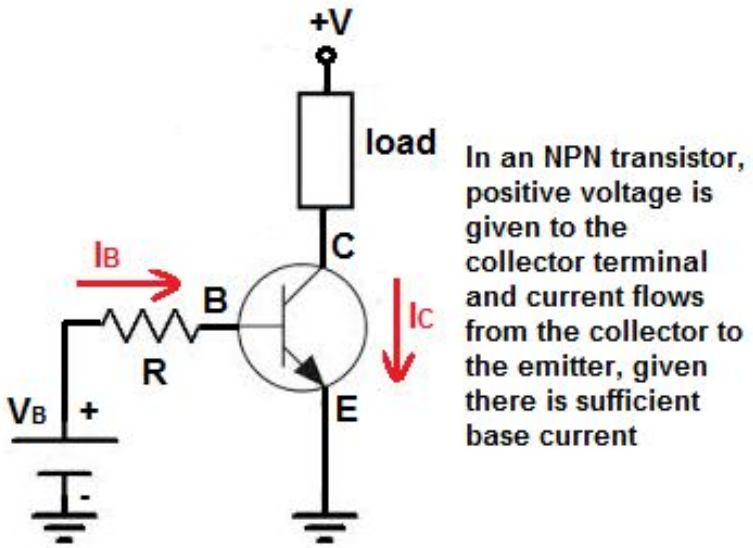
= (**A** NAND **A**) NAND (**B** NAND **B**)

Truth Table

Input A	Input B		Output Q
0	0		0
0	1		1
1	0		1
1	1		1

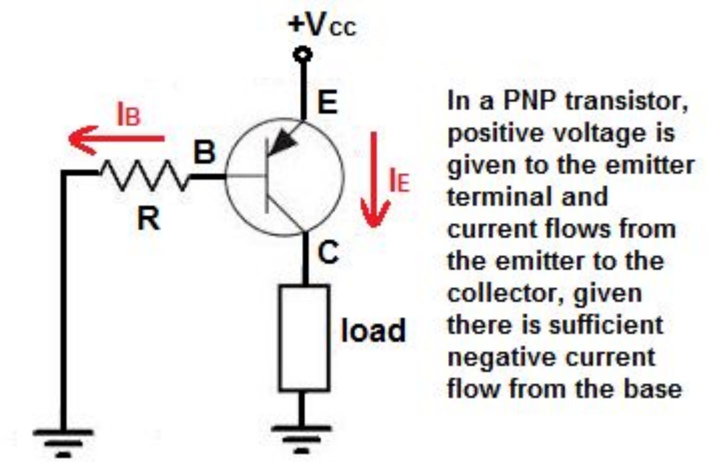
NPN vs PNP Transistors

- It turns out there are two different major types of transistors
- NPN Transistors
  - Positive voltage allows current to flow
  - This is what we have been discussing mainly so far



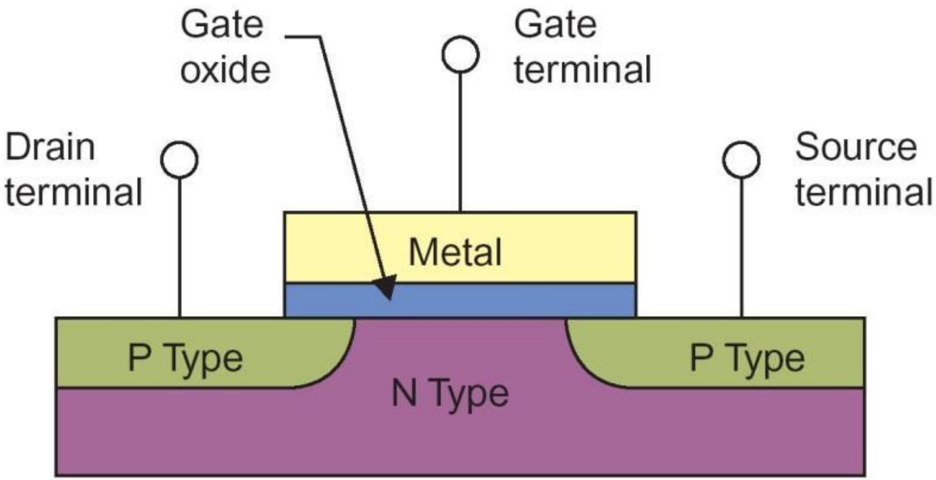
NPN vs PNP Transistors

- PNP Transistors
  - Application of voltage *prevents* current from flowing
  - Inverts the logic of the NPN transistor



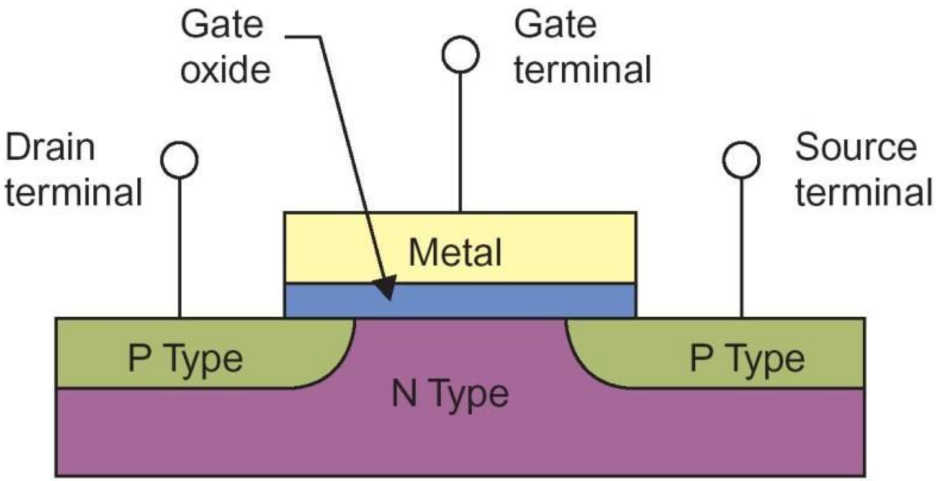
Field-Effect Transistors

- In order to make transistors smaller, our friends in EE have been creating smaller and smaller implementations of these basic building blocks
- MOSFET technology has been used to reduce and pack in transistors in mind bending amounts



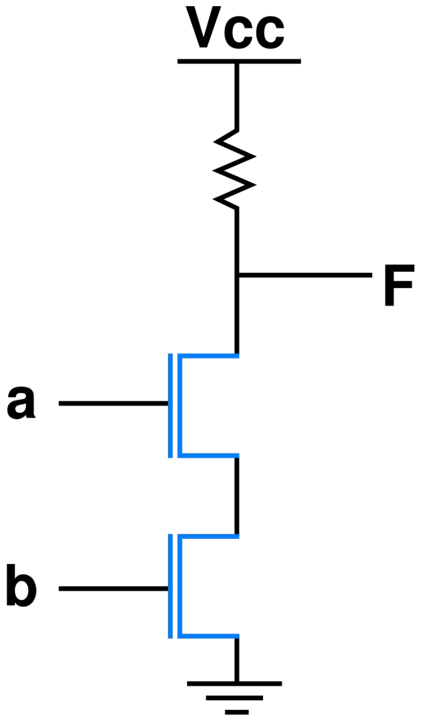
Field-Effect Transistors

- The NPN and PNP equivalent in MOSFETs are called NMOS and PMOS



NAND Gates & Transistors

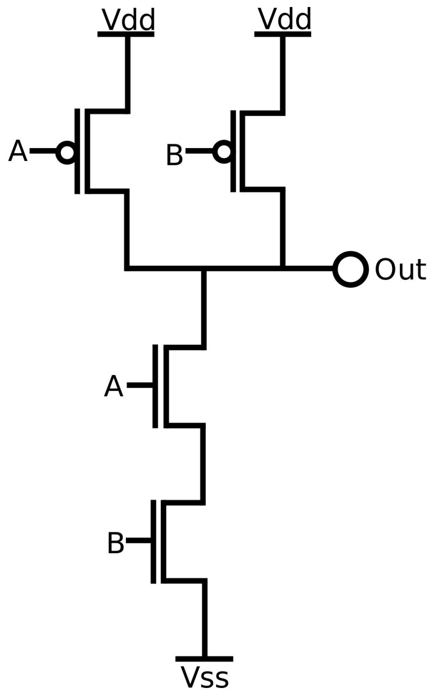
- How could we implement a NAND using NPN transistors?
- Here is the diagram for an NMOS NAND gate
  - If A and B are high, current will flow from Vcc to ground
  - Otherwise it will flow out (F)
- A simple NAND implementation!



NMOS NAND gate

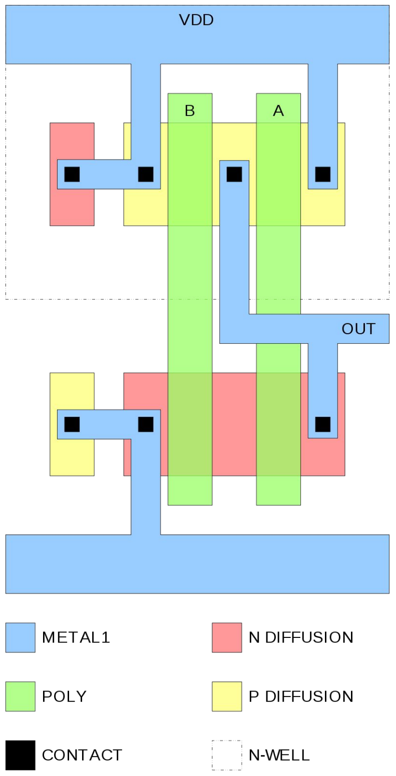
NAND Gates & Transistors

- A more modern and slightly more complex CMOS NAND gate
- Note that the top transistors are PNP gates (thus voltage *stops* current) while the bottom gates are NPN transistors



NAND Gates & Transistors

- A physical layout of a NAND gate
- CMOS uses *both* NPN/NMOS and PNP/PMOS type semiconductors
  - Again, this is not an EE class
  - Just know that
    - NPN/NMOS - currency flows if voltage is applied
    - PNP/PMOS - currency flows if voltage is NOT applied



Transistors & Logical Gates

- Today took a beginners tour of the lowest level of digital computing
- We discussed transistors, an electrical device that can be used for digital switching
- We revisited the logical operators
    - We discussed how NAND can be used to create other logical operators
- We looked at how to implement these logical operators using transistors
- *REMEMBER: IT'S JUST LIGHTNING RUNNING THROUGH SAND*

MONTANA
STATE UNIVERSITY