

Transistors, Boolean Logic and Logical Gates

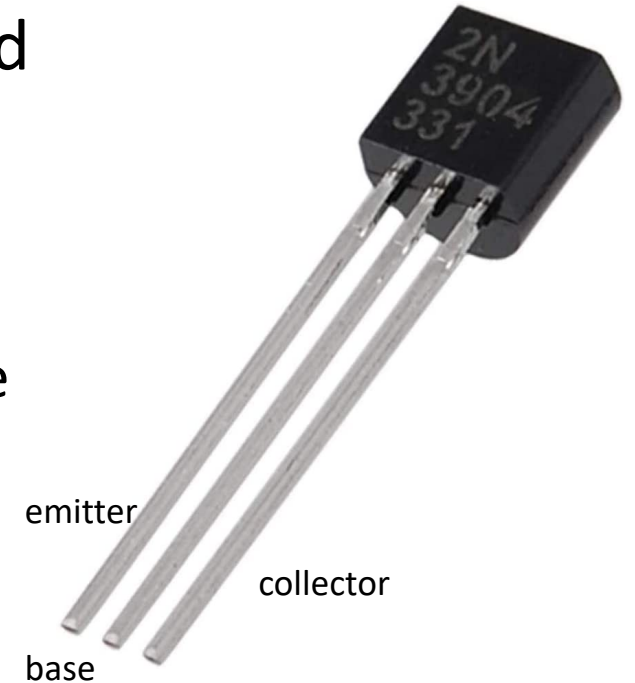
Garrett Dancik, PhD

Fall 2024

Course Notes: <https://gdancik.github.io>

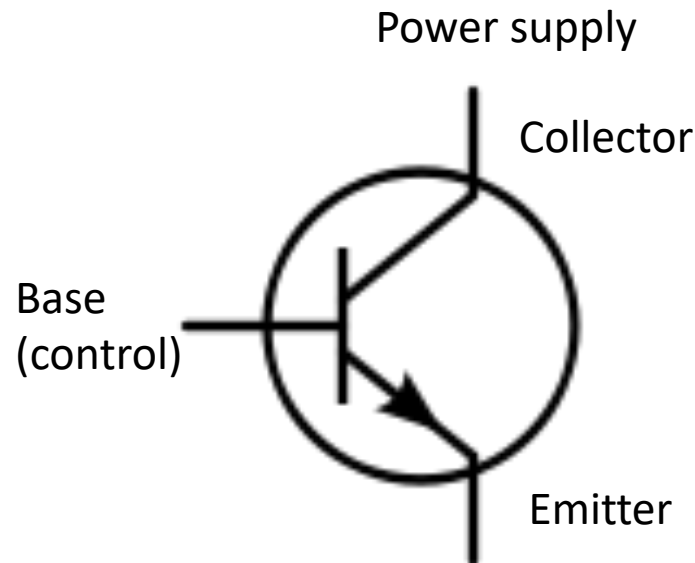
Why Binary?

- The foundation of computer hardware (processing and memory) are devices that can operate in two stable states.
 - A transistor can either turn current on (1) or off (0)
 - Magnetic core memory (common until 1975): a core can be magnetized in either the clockwise or counter-clockwise direction
 - A hard drive consists of regions of magnetic material which can be magnetized (1) or not (0)



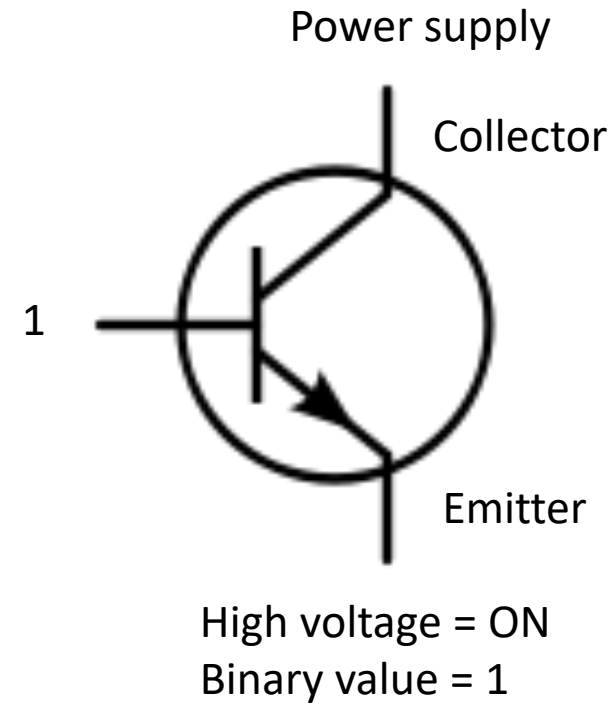
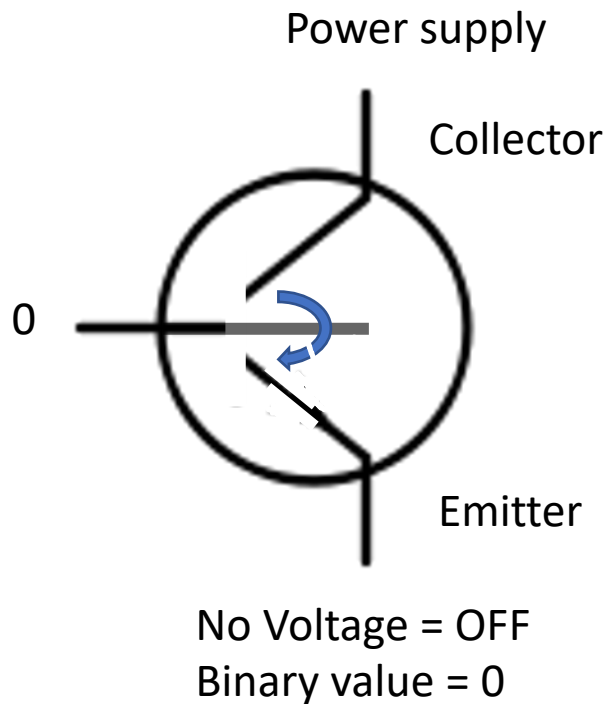
Transistor basics

- A transistor is a semi-conductor that can amplify or switch electronic signals.



- The base is used to open/close the switch
 - If the control line is set to 1, then the switch is closed, and current will flow from the collector to the emitter. The transistor is ON (1). (This is the state of the current figure)
 - If the control line is set to 0, then switch is open, and current will not flow from the collector to the emitter. The transistor is OFF (0)

A transistor can be OFF (0) or ON (1)



Additional details: <https://www.youtube.com/watch?v=stM8dgcY1CA>

Boolean logic

- Computer circuits are constructed based on Boolean logic
 - Boolean logic is a branch of mathematics (algebra) that deals with *true* and *false* values
- In computer logic,
 - *true* represents a binary 1 or a transistor that is ON
 - *false* represents a binary 0 or a transistor that is OFF
- Boolean operations include AND, OR, NOT, NOR, NAND, and XOR

Truth table for AND and OR

Truth table for: a AND b

Inputs		Output
a	b	a AND b (also written as $a \cdot b$)
True (1)	True (1)	True (1)
True (1)	False (0)	False (0)
False (0)	True (1)	False (0)
False (0)	False (0)	False (0)

The expression a AND b is True only if both a and b are True

Truth table for: a OR b

Inputs		Output
a	b	a OR b (also written as $a + b$)
True (1)	True (1)	True (1)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

The expression a OR b is True if either a or b are True (including if both are True)

Truth table for NOT

Truth table for: *NOT a*

Inputs	Output
a	<i>NOT a</i> (also written as \bar{a})
True (1)	False (0)
False (0)	True (1)

The expression *NOT a* is True if *a* is False and is False if *a* is True

- Boolean logic examples:
 - grade is between 90 and 100
 - `grade >= 90 AND grade <= 100`
 - User has typed 'Y' or 'y'
 - `userInput == 'Y' OR userInput == 'y'`
 - User has not typed 'Y' or 'y'
 - `userInput != 'Y' AND userInput != 'y'`

Truth table for NAND and NOR

Truth table for: $a \text{ NAND } b$

Inputs		Output
a	b	$a \text{ NAND } b$ (also written as $\overline{a \cdot b}$)
True (1)	True (1)	False (0)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	True (1)

The expression $a \text{ NAND } b$ is equivalent to $\text{NOT } (a \text{ AND } b)$ and is True if either a or b are False

Truth table for: $a \text{ NOR } b$

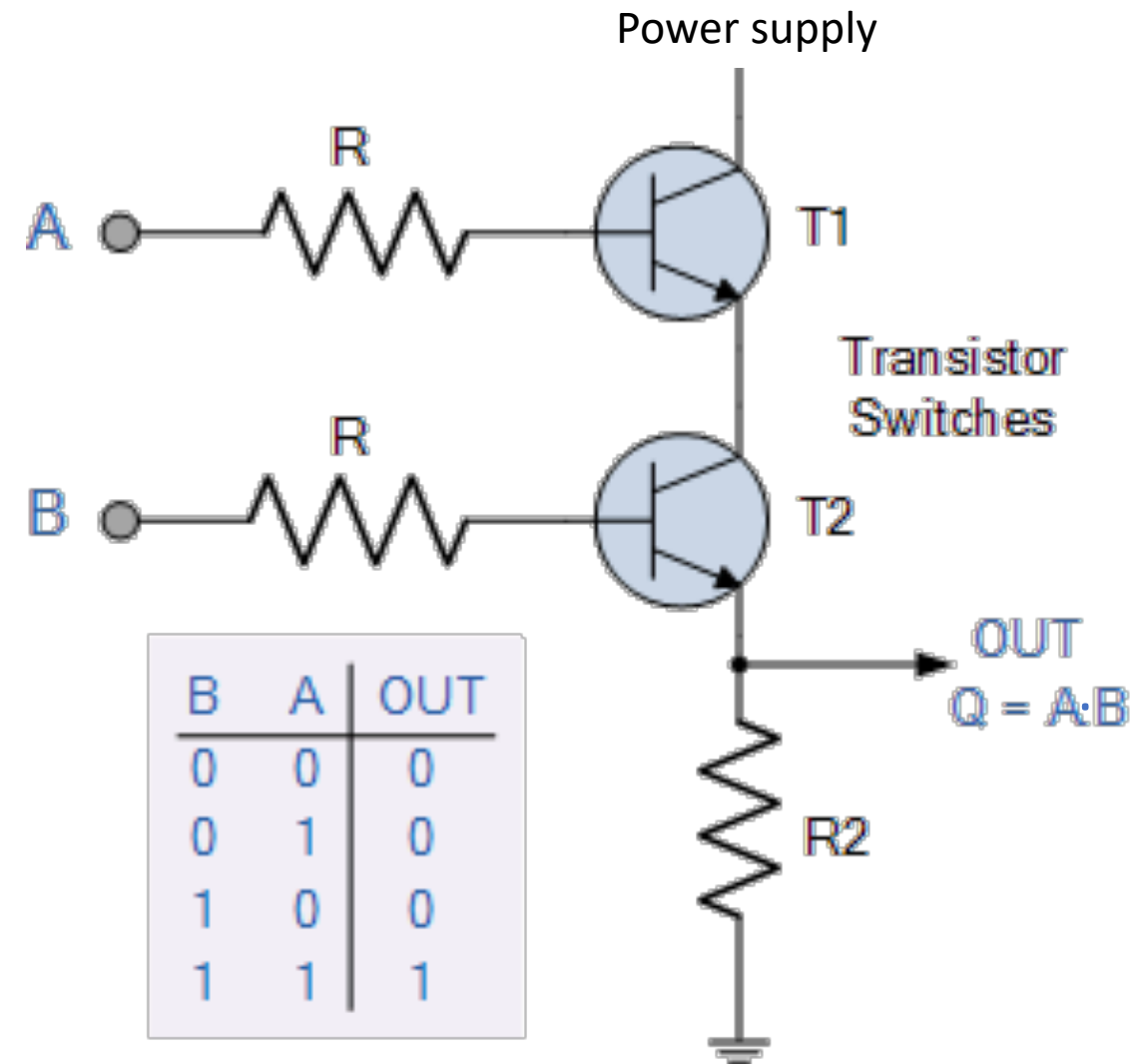
Inputs		Output
a	b	$a \text{ NOR } b$ (also written as $\overline{a + b}$)
True (1)	True (1)	False (0)
True (1)	False (0)	False (0)
False (0)	True (1)	False (0)
False (0)	False (0)	True (1)

The expression $a \text{ NOR } b$ is equivalent to $\text{NOT } (a \text{ OR } b)$ and is True only if both a and b are False

Logic Gates

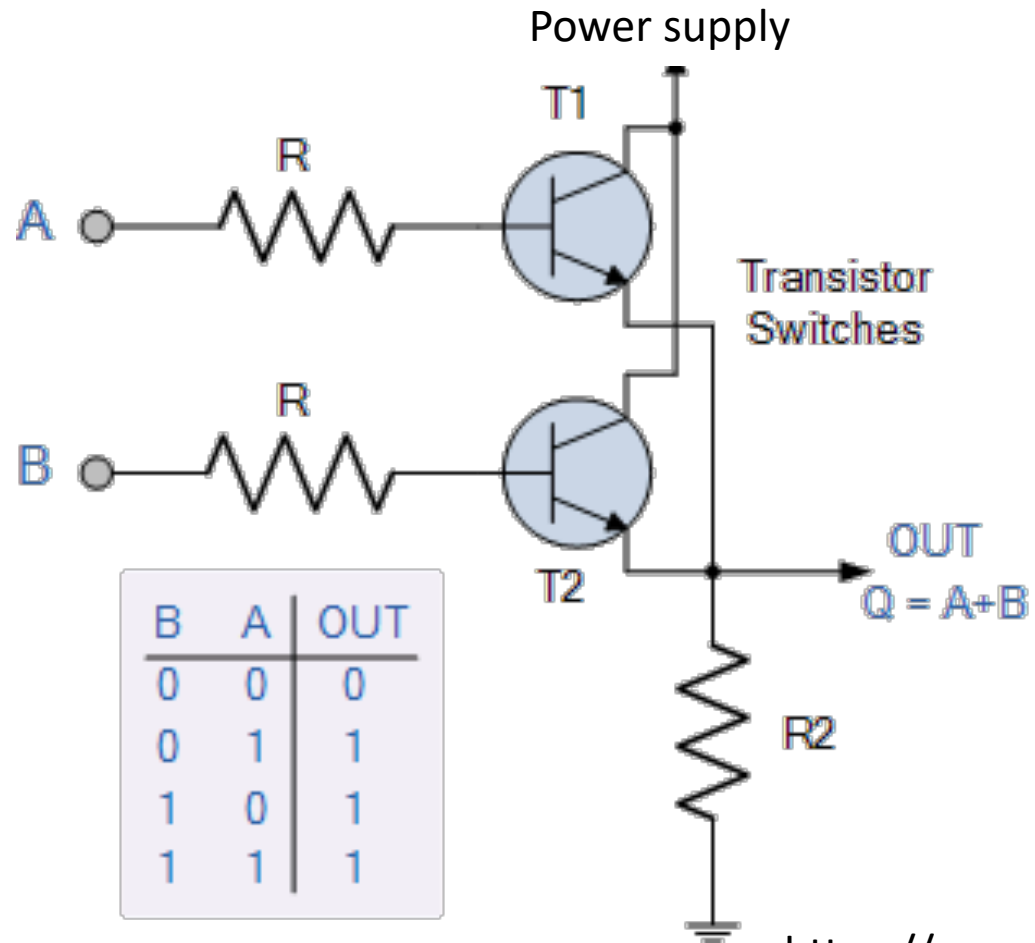
- A *logic gate* is an electronic device that takes one or more binary inputs and produces a single binary output.
- Gates are created from transistors
- Types of gates: AND, OR, NAND, NOR (and XOR and XAND)
- An AND gate is shown on the right
 - Recall: If the base (input) is 1, the transistor switch will be closed
 - Only if A and B are both 1, will current flow from the power supply through the transistors, resulting in voltage at the output (OUT)
 - This gives a digital implementation of the Boolean AND operator

Transistor AND Gate

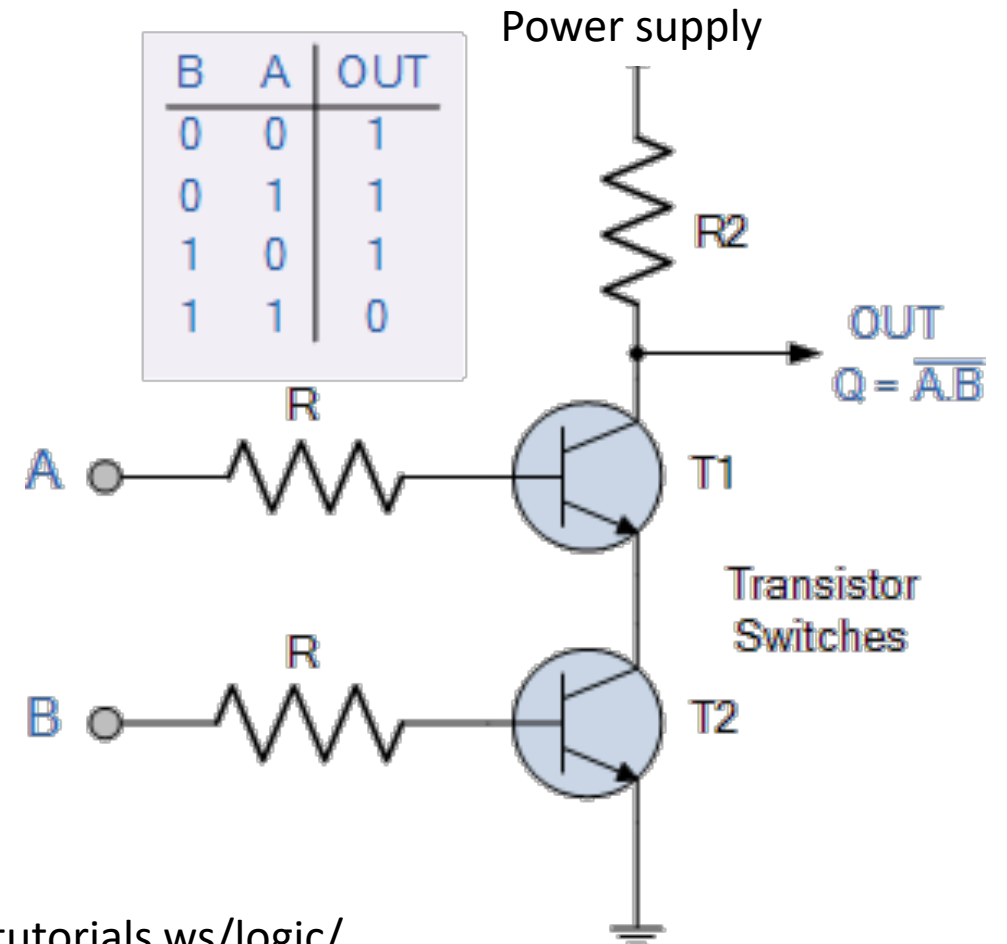


Additional Gates

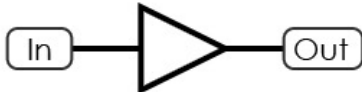
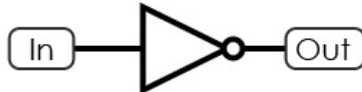






Transistor OR Gate



Transistor NAND Gate



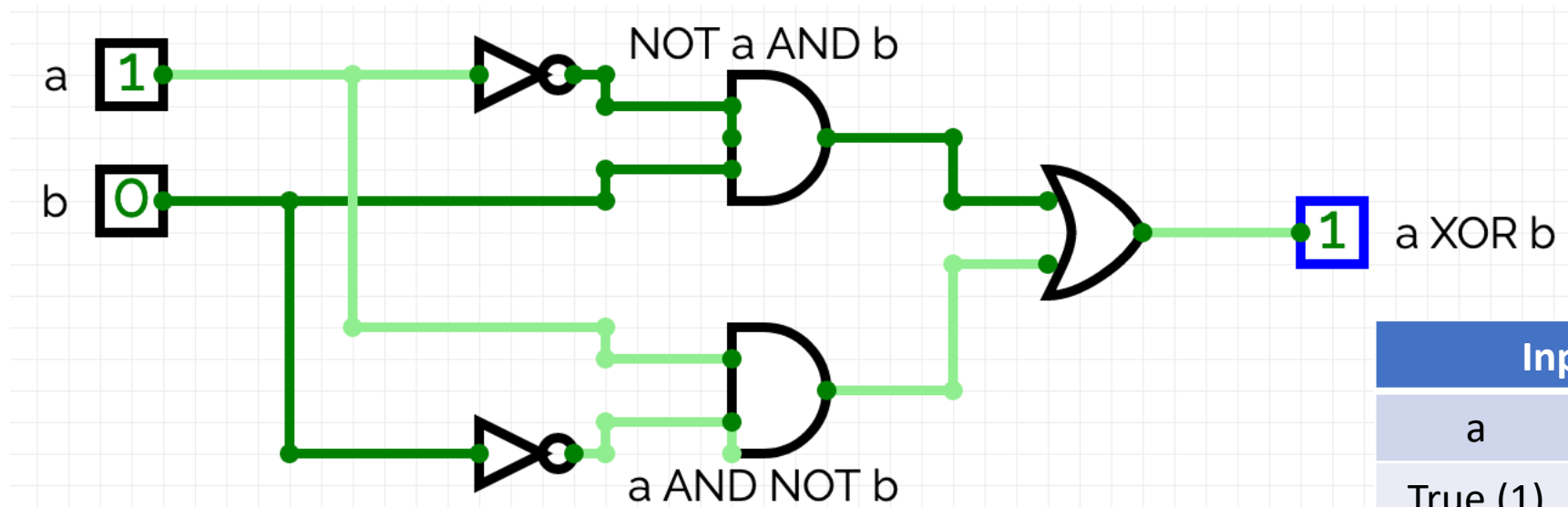
Logic Gates - Symbols and Truth Tables

<div>BUF (Buffer)</div> <div></div>	<table><tr><th>In</th><th>Out</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	In	Out	0	0	1	1	<div>NOT (Inverter)</div> <div></div>	<table><tr><th>In</th><th>Out</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	In	Out	0	1	1	0																		
In	Out																																
0	0																																
1	1																																
In	Out																																
0	1																																
1	0																																
<div>AND</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	In1	In2	Out	0	0	0	0	1	0	1	0	0	1	1	1	<div>NAND (NOT AND)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	In1	In2	Out	0	0	1	0	1	1	1	0	1	1	1	0
In1	In2	Out																															
0	0	0																															
0	1	0																															
1	0	0																															
1	1	1																															
In1	In2	Out																															
0	0	1																															
0	1	1																															
1	0	1																															
1	1	0																															
<div>OR</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	In1	In2	Out	0	0	0	0	1	1	1	0	1	1	1	1	<div>NOR (NOT OR)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	In1	In2	Out	0	0	1	0	1	0	1	0	0	1	1	0
In1	In2	Out																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	1																															
In1	In2	Out																															
0	0	1																															
0	1	0																															
1	0	0																															
1	1	0																															
<div>XOR (Exclusive Or)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	In1	In2	Out	0	0	0	0	1	1	1	0	1	1	1	0	<div>XNOR (NOT XOR)</div> <div></div>	<table><tr><th>In1</th><th>In2</th><th>Out</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	In1	In2	Out	0	0	1	0	1	0	1	0	0	1	1	1
In1	In2	Out																															
0	0	0																															
0	1	1																															
1	0	1																															
1	1	0																															
In1	In2	Out																															
0	0	1																															
0	1	0																															
1	0	0																															
1	1	1																															

A circle behind a symbol indicates that the output signal is inverted.

Logic gates are the building blocks of computer systems

Exclusive OR (XOR)



XOR is an *exclusive or* which returns True if only one of *a* or *b* are True; if both *a* and *b* are True, the *exclusive or* returns False

Inputs		Output
a	b	$a \text{ XOR } b$
True (1)	True (1)	False (0)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

Digital Circuits

- A circuit is a collection of logic gates that transform binary inputs into binary outputs.
 - If the outputs depend only on the current inputs, the circuit is a *combinational circuit*
 - If the outputs also depend on previous inputs, the circuit is a *sequential circuit*
- Every output in a circuit can be represented as a Boolean expression
- We will use <https://circuitverse.org/> to simulate circuits

CircuitVerse overview

- Let's use <https://circuitverse.org/> to simulate our own NAND circuit using an AND gate and a NOT gate
1. Select an AND gate from the Gates panel and add it
 2. Select two binary Input elements from the Input panel (the Input element is the first one)
 3. Draw lines from each input to the AND gate
 4. Add a NOT gate from the Gates panel
 5. Connect the output of the AND gate to the input of the NOT gate
 6. Add an Output element from the Output panel (the first element)
 7. Connect the output of the NOT gate to the Output element

Tricks:

- You can click on an element and hit delete to delete it.
- Click and drag an element to move it
- Hold shift and move the mouse to select multiple elements

Circuit Construction Algorithm

- Construct a Truth table defining the relationship between inputs and outputs
- Specify one or more Boolean expressions that hold for every row where the output is 1, and which never evaluate to 1 otherwise. This is always possible by using the AND operator and all inputs, some of which may be inverted.
- Use OR to combine the Boolean expressions identified in the step above

a XOR b

Inputs		Output
a	b	$a \text{ XOR } b$
1	1	0
1	0	1
0	1	1
0	0	0

← a AND NOT b

← NOT a AND b


Circuit is equivalent to:

$(a \text{ AND NOT } b) \text{ OR } (\text{NOT } a \text{ AND } b)$

Example – identity comparator

- Determine the Boolean expression that corresponds to the output from the below Truth Table for the identity comparator. Then construct the corresponding circuit. The identity comparator outputs a 1 if both of its inputs are the same (both 1 or both 0)

Inputs		Output
a	b	<i>output</i>
1	1	1
1	0	0
0	1	0
0	0	1

- Note: This is equivalent to NOT XOR, or XNOR 
- However, let's assume that we do not have an XNOR gate. How can we create a XNOR circuit from AND, NOT, and OR gates?