CMSC389E:

Digital Logic Design Through Minecraft

Logic Gates

Fall 2020

Akilesh Praveen

Administrivia

- Install & Setup 389E Minecraft
  - Installer coming out very soon
  - For help w/ installation, send a message on Discord
- Turn in Project 0 on **ELMS** (will open on Monday)

- Keep checking the course website! (will be updated tonight) (http://www.cs.umd.edu/class/fall2020/cmsc389E/)

- **Special Setup & Installation Friday Happy Hour! (alcohol consumption not recommended)**
  - **Aki will be available on Discord today 3PM-5PM (and this weekend with appointment) to assist with installations**

Announcements

- Project 1
    - Conceptual knowledge in today’s lecture + online textbook
    - Project spec will be released on ELMS and course website **tonight / Saturday**

Inputs & Outputs

Inputs & Outputs

- The basic logic gates make up all of our circuits
    - AND, OR, NOT, XOR
- More advanced gates are just combinations of these gates
    - (NOR, XNOR, etc.)
- All require two inputs and produce **one** output (NOT being the exception)

Inputs & Outputs

- Logic gates = absolute most basic constructs of the redstone circuits we will be building in this class

- As you can imagine, there are fairly easy ways to represent those using Minecraft’s built-in redstone mechanics

  READ: Digital Logic & Comp Arch in Minecraft, C1.3

Inputs & Outputs

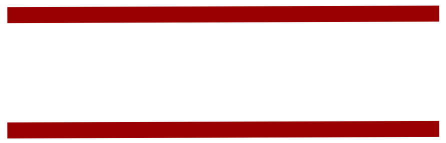
- How can we take our understanding of **logic gates** and translate it into **Redstone?**

- Go step by step, making clear connections with redstone concepts along the way

Inputs & Outputs

- **Input** for a logic gate is composed of **1 or 2** values, both of which can be either **True** or **False**

- Let's represent these inputs as **Redstone wires**; a wire with current is **True**, a wire without is **False**

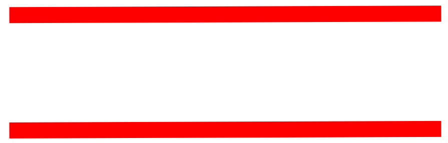


“Off”, or False

Inputs & Outputs

- **Input** for a logic gate is composed of **1 or 2** values, both of which can be either **True** or **False**

- Let’s represent these inputs as **Redstone wires**; a wire with current is **True**, a wire without is **False**



“On”, or True

Inputs & Outputs

- Q: Can we think of outputs in the same way?

- A: Yes, we can!

Inputs & Outputs

- **Output** can be represented by a single Redstone wire coming out of our logic gate.

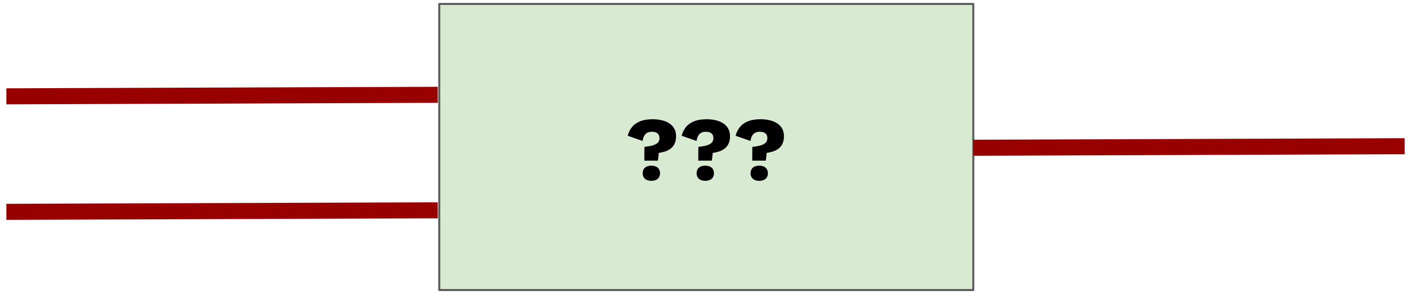
- If the wire carries current, it has a state of **True**. If it does not, it has a state of **False**.

“Off”, or False

“On”, or True

Inputs & Outputs

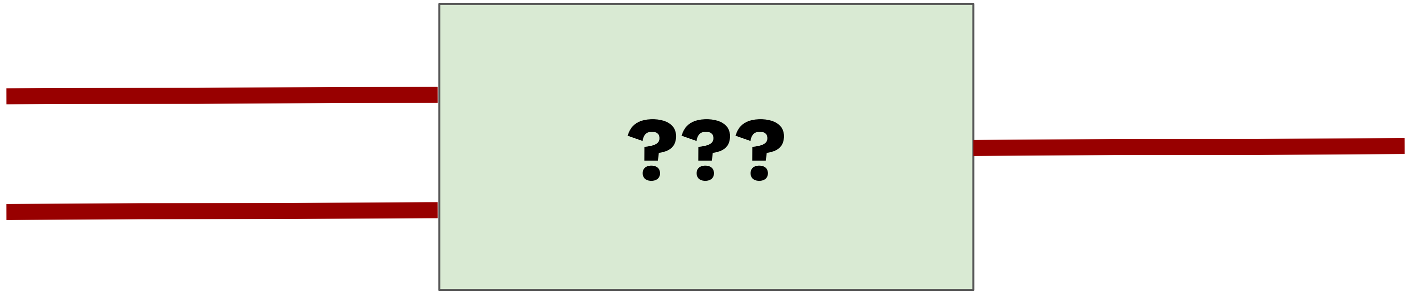
- Here’s what we have so far- a way to represent input and output for our logic gates using Redstone wires.



Basic Logic Gates

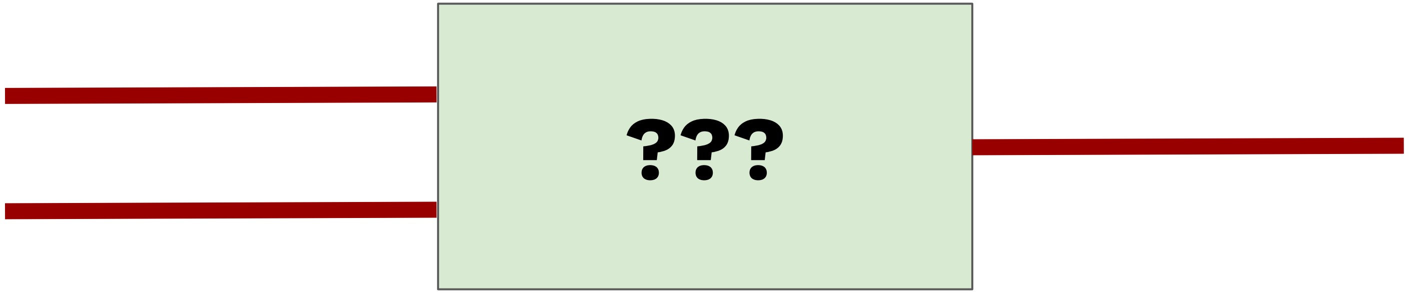
The Logic Gates

- Now, let's figure out what goes into the mystery box when we want to build **NOT, AND, OR** and **XOR**.



The Logic Gates

- Live demos- NOT, AND, OR, XOR



READ: Digital Logic & Comp Arch in Minecraft, 2.1, 2.2, 2.3, 2.4

The Logic Gates

- **If** you’ve taken 250 already, this is stuff you should already know
- If you have yet to catch up on 250 or need a refresher:
    - Detailed overview of the logic gates can be found in the online textbook.
    - Ashwath is also a 250 TA :)

The Logic Gates

- **Nuances to remember**
  - A redstone torch, placed correctly, will invert current
  - Repeaters will extend current **and** act as diodes
    - Useful to prevent backflow!
  - You are welcome to use comparators, but it is not encouraged

Logical Complements

Logical Complements

- Let’s talk about logical complements
- Occasionally, we have a gate or circuit that produces the **exact opposite** of what we want
- In this case, use the logical complement of whatever gate or circuit you’ve got
- Using ‘dot’ notation is less cumbersome sometimes

READ: Digital Logic & Comp Arch in Minecraft, C2.5

An Exercise

- Let’s make your 250 TAs proud
- Construct an XOR gate using only ORs, ANDs, and NOTs
  - Try to be as efficient as possible. In real life, circuits are made by hand, and every piece has a cost!
  - More gates = more built time, so learn to think efficiently?

An Exercise

- Why is this so important?

An Exercise

- Why is this so important?
- **Some gates can be built using other gates**

Applications In Computing

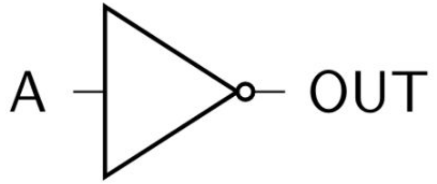
The NAND Gate’s Importance

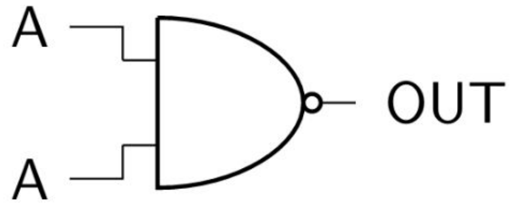
- Let’s talk about the NAND Gate
- They are the most commonly found gates in computers (*NAND Flash?*)
- Why is that?

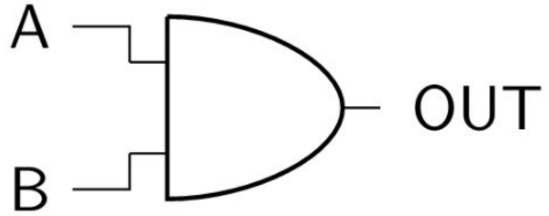
The NAND Gate’s Importance

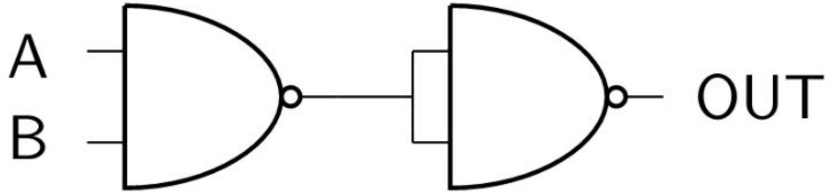
- Let’s talk about the NAND Gate
- They are the most commonly found gates in computers (*NAND Flash?*)
- Why is that?
- **The NAND gate is ‘functionally complete’**
    - So is the NOR gate!

The NAND Gate’s Importance

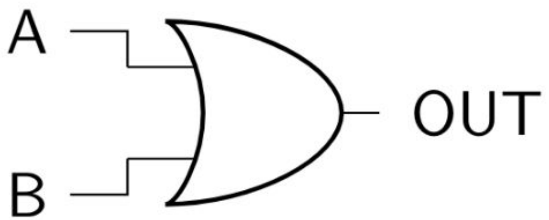


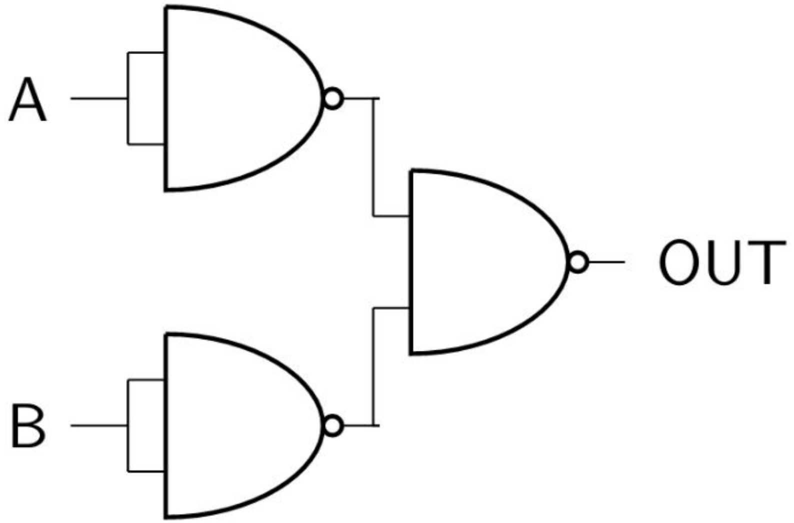




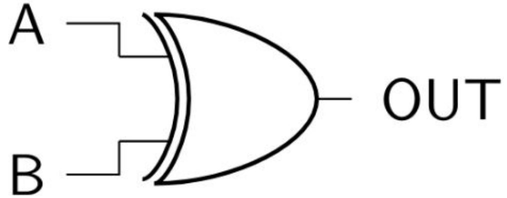


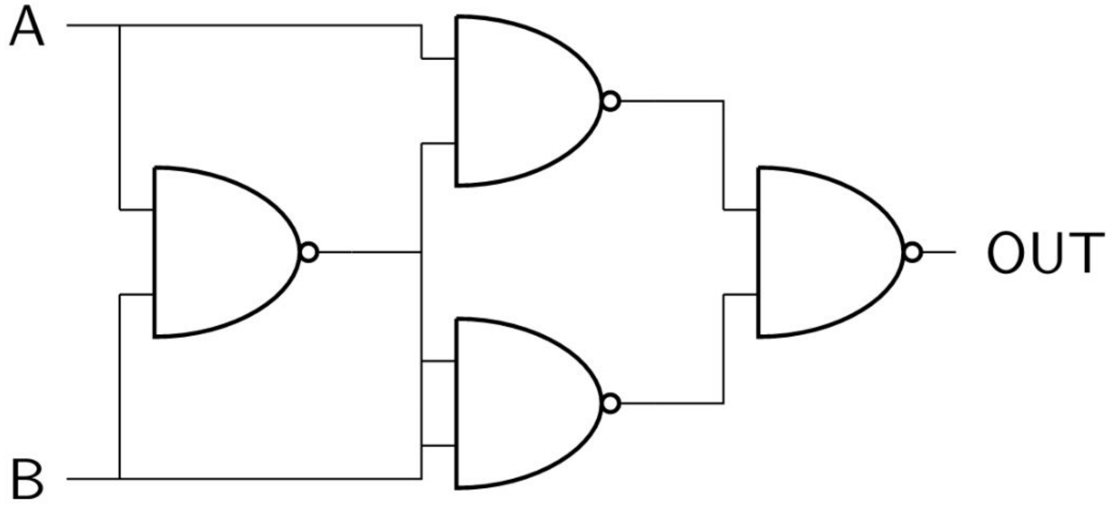
The NAND Gate’s Importance





The NAND Gate's Importance





The NAND Gate’s Importance

- What’s the big deal with NAND being functionally complete?
    - If a gate is functionally complete, all other gates can be constructed using it
    - In CMOS (a common circuit fabrication process), NAND is smaller and faster than a NOR gate.

The NAND Gate’s Importance

- TL;DR: Thanks to already set industry precedents + the initial ease to manufacture them, NAND gates are the pick over NOR gates for the universal gate that engineers used to build other gates, and eventually other full sized circuits.

READ: Digital Logic & Comp Arch in Minecraft, C2.6

Applications in Minecraft

Why is this big for us?

- We are able to create a NAND gate in Minecraft!

Why is this big for us?

- We are able to create a NAND gate in Minecraft!
  - *READ: We are able to create a fully functional computer in Minecraft!*
- **Provided** that we can create a NAND gate in Minecraft, and **knowing** that a NAND gate can be used to build **every other gate**, we have thusly proven that we can build a computer within Minecraft! Q.E.D.

NAND for everything!

- Does this mean we will build everything using NAND gates in Minecraft?

NAND for everything!

- Does this mean we will build everything using NAND gates in Minecraft?
  - No, absolutely not. That’s ridiculous

NAND for everything!

- Does this mean we will build everything using NAND gates in Minecraft?
  - No, absolutely not. That’s ridiculous
- It’s important to notice the subtle, yet **key** differences between Minecraft and real life (in terms of digital logic and otherwise).

Efficient Circuit Design

Efficiency

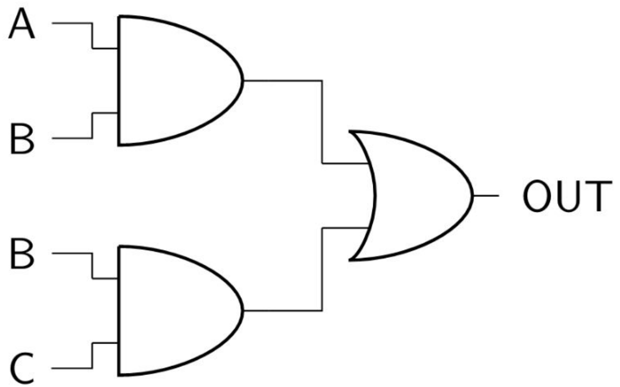
- We want to create circuits that, given input, produce the output that we want
  - With as few components as possible
- When discussing boolean logic, we should think about the logic rules that come with it.

Efficiency

- Digital Logic Circuits <-> Boolean Predicates
- We can use logic rules to perform simplifications on our circuits when needed (http://www.cs.umd.edu/class/spring2019/cmsc250-020X/files/logic-laws.pdf)
- Now, you'll give it a try

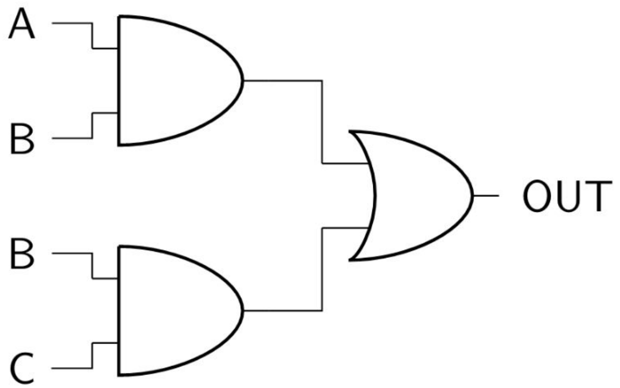
Efficiency

- Suppose you are given the following circuit. How would you go about simplifying this?



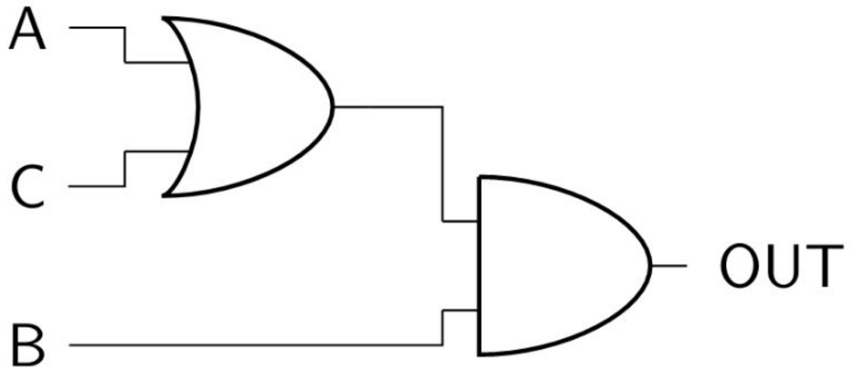
Efficiency

- Write it out using boolean logic:
    - $(A \land B) \lor (B \land C)$



Efficiency

- Apply the distributive law to get the following: (simpler!)
    - B ∧ (A ∨ C)



Key Takeaways

- Knowing the basics of boolean logic opens the doors for us to make more advanced circuits!

- Understanding functional completeness gives us the theoretical knowledge to know just **how** computers can be created in Minecraft

Key Takeaways

- Boolean logic rules are essential to designing circuits in the most efficient way possible
  - Saving time, money and materials in the real world
  - Saving us from carpal tunnel in Minecraft

<u>Project 1</u>