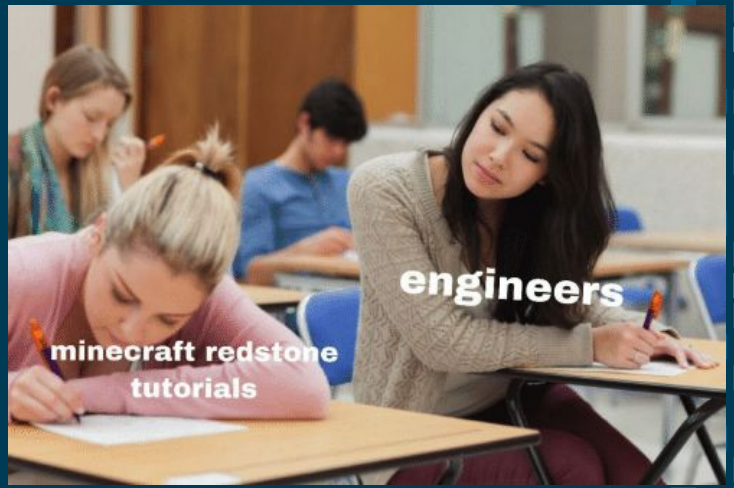
How Minecraft Teaches Complex Machine Logic Circuits

Way before we even know what machine logic circuits are.



Thomas Quig and Rima Bouhal

What you will learn in this class

- That you should not be afraid of computer architecture, because many of you have already learned the basics through Minecraft.
- A summary of the first two weeks of CS 233
  - What logic gates are
    - The four basic logic gates
  - What truth tables are
  - What boolean expressions are
  - The first logic circuits that are critical to computer architecture
    - One Bit Full Adder
    - Multiplexor
    - Logic Unit
- Basic Minecraft Redstone
  - How everything in CS 233 can (technically) be represented in Minecraft.

https://bit.ly/2YP8e1H

Icebreaker

Name

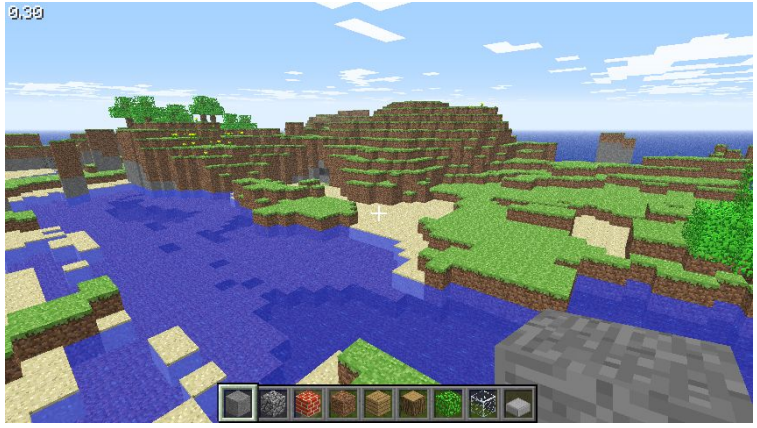
Favorite operating system

Who Are We?

- Thomas Quig (whoami.quig.dev)
  - CS Junior
  - Interested in Computer Security, Network Security, Systems Design, Computer Architecture
  - President of SIGPwny, involved in DDR Club
- Favorite Games
  - Borderlands 2
  - Titanfall 2
  - Risk Of Rain 2
  - Minecraft
- Rima Bouhal
  - CS Junior
  - Interested in Artificial Intelligence, Computational Photography, Computer Architecture
  - Involved in WCS, Illini Swing Dance Society
- Favorite game
  - Chrome Dinosaur Game

What is Minecraft? (Alright come on guys,,,)

- Popular game released in 2009
- Open world, (almost) everything is cubical



Quick Vocabulary List

- **Logic gate:** A circuit that takes inputs (>=1) and gives an output based on a simple operation.
- **Binary:** Base 2 arithmetic, formed by 0's and 1's.
  - If you need more help with binary ask an instructor.
- **Bit:** One binary value, either on or off.
- **Byte:** Eight binary values in sequence, usually split into 4s (`1001 0010`).
- **Truth Table:** Table of inputs and outputs, another way to represent logic gates
- **Gate Schematics:** A standard drawing of wires, inputs, and outputs most commonly associated with logic gates.

What are logic circuits?

- A logic circuit is a sequence of gates
  - Takes input(s), usually in binary
    - 1 (On)
    - 0 (Off)
  - These gates are combined to do more complex tasks
- Logic circuits are the foundations of all computing
  - In the most simple terms, everything in your computer is a combination of logic gates, and wires.
    - This is not completely true, but it's enough for the scope of this class

How this is represented in Minecraft

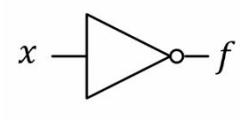
- **Redstone Dust**
  - Found in the Minecraft world naturally, equivalent to a wire
  - Signal strength (0-15) and a powered state (on or off)
- **Redstone Torches**
  - Produces signal strength of 15, or full power
- **Redstone Repeaters**
  - Can extend a signal by returning it to full strength
  - Used as a one way Gate
- **Redstone Lamp**
  - Lamp that turns on with a redstone signal, better visual representation of signals

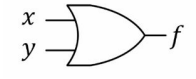
Lets Play A Video Game

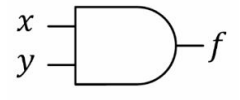
World Download: https://bit.ly/2YP8e1H

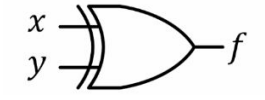
Version 1.13.2 Required

The Four Most Important Logic Components

- NOT 

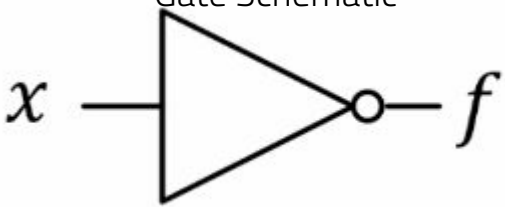
- OR 

- AND 

- XOR 

Not Gate

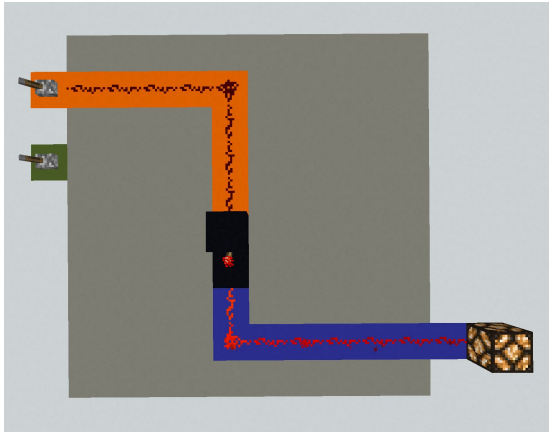
Gate Schematic

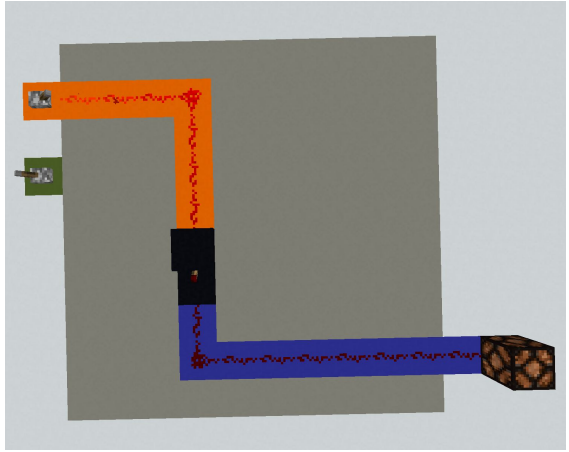


Boolean Expression

$$f(x) = x'$$

- Inverts the input. Output is the opposite of the input. (`1 -> 0`), (`0 -> 1`)

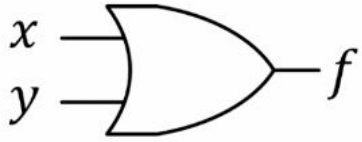




Truth Table

x	f(x)
0	1
1	0

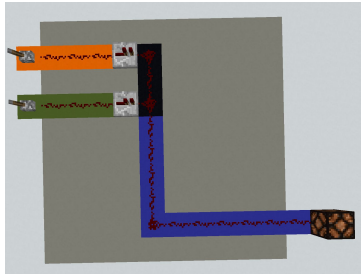
OR Gate

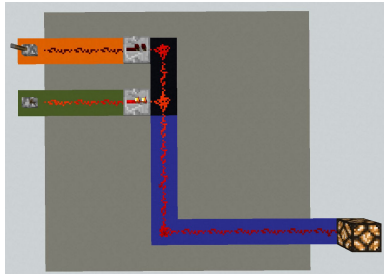


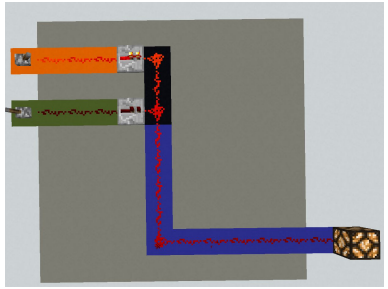
$$f(x, y) = x + y$$

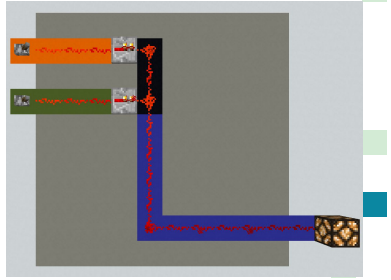
- Takes two inputs
- Returns 1 if one or both of the inputs are 1

x	y	f(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

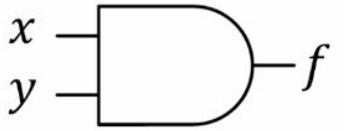








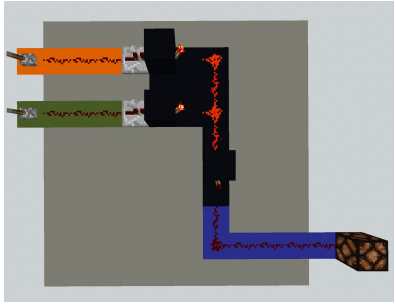
AND Gate

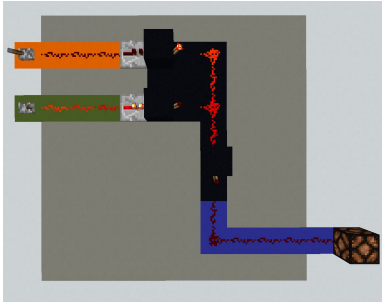


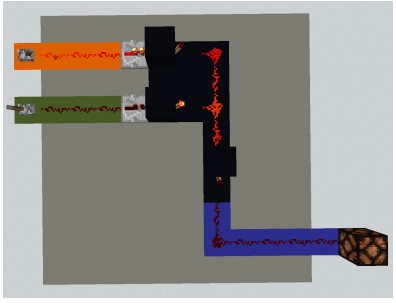
$$f(x, y) = x \cdot y$$

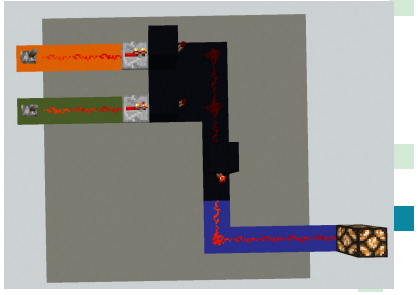
x	y	f(x,y)
0	0	0
0	1	0
1	0	0
1	1	1

- Takes two inputs
- Returns 1 if and only if both of the inputs are 1



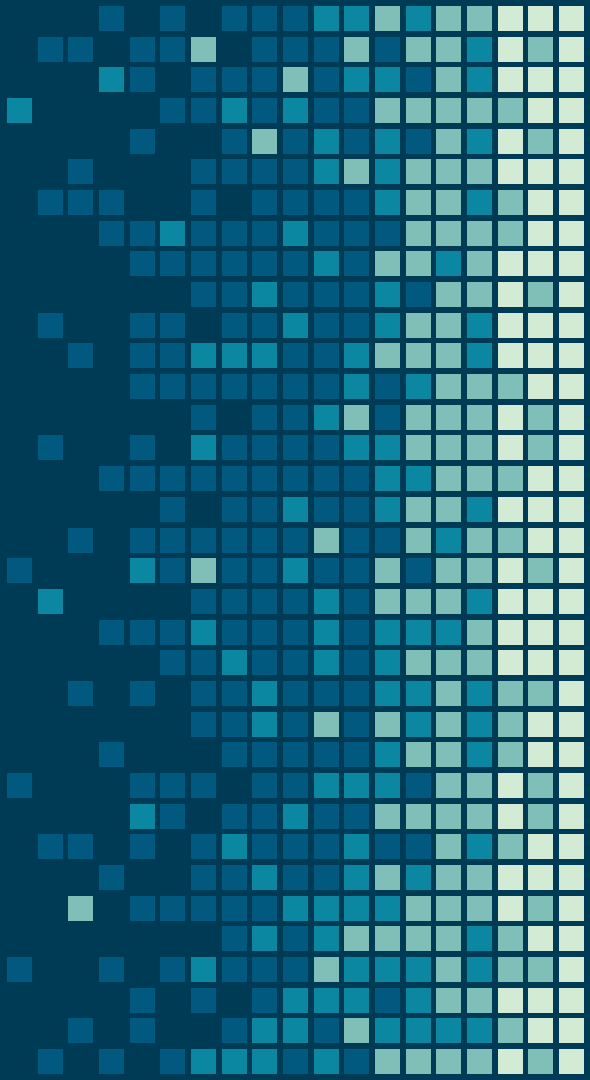






Go to Minecraft now

If you don't have Minecraft thats okay, discuss with someone who does.



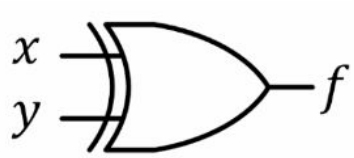
Practice Time

Draw the gate schematics and truth tables for the following boolean expressions (5 Minutes)

1. (XY′)
2. (X′Y)+(X′Y′)
3. Design a NAND gate (Not and)
4. Design a NOR gate (Not or)

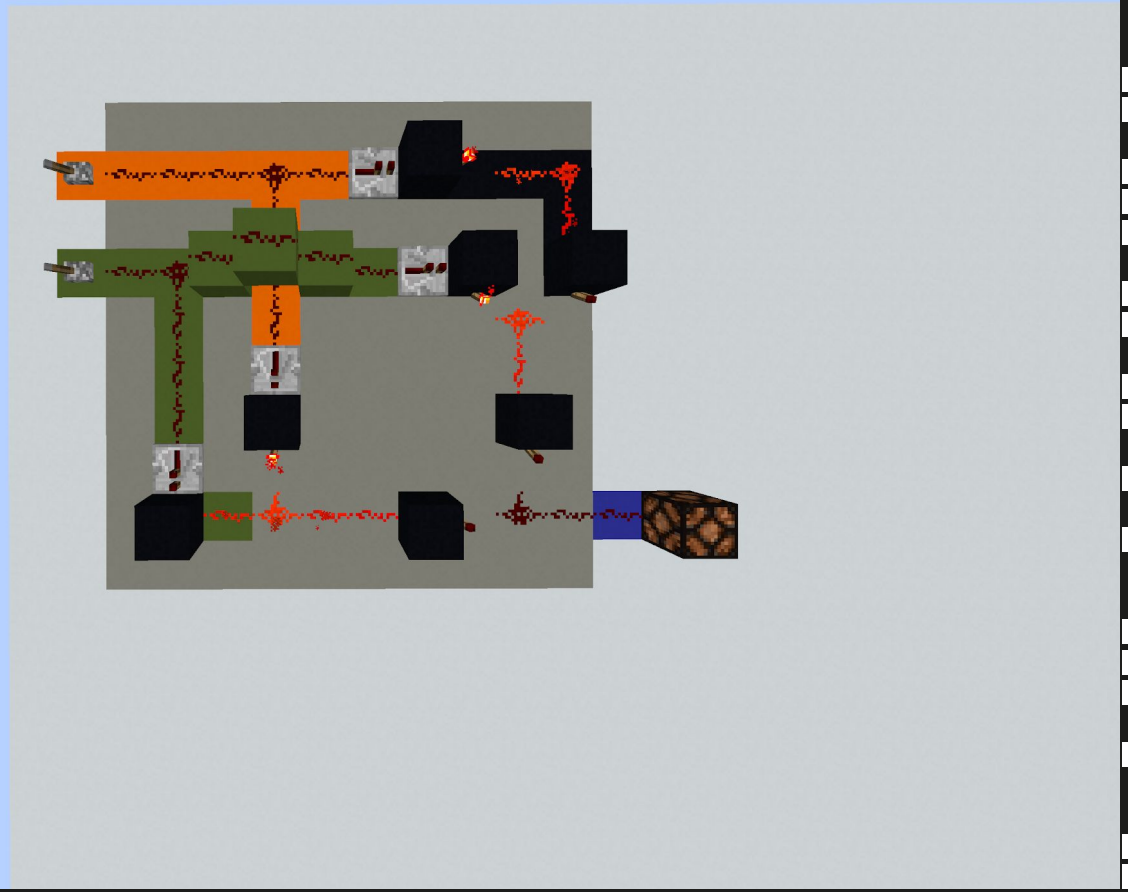
XOR Gate

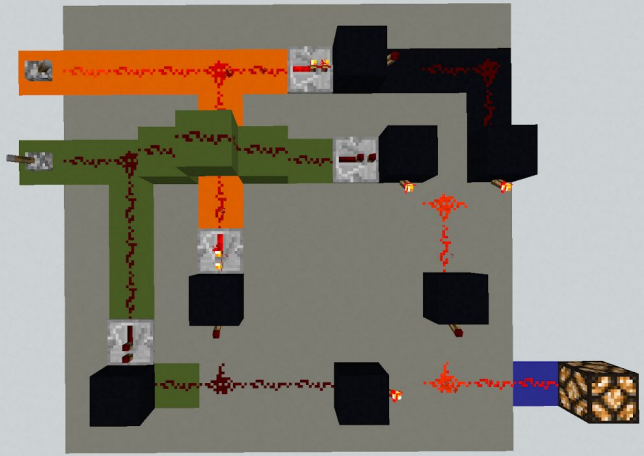
- Also known as Exclusive-OR gate
- Takes two inputs, returns 1 only if one of the two inputs is 1.
- Logically equivalent to (AB')+(A'B)
- Many ways to design, varying levels of compactness.

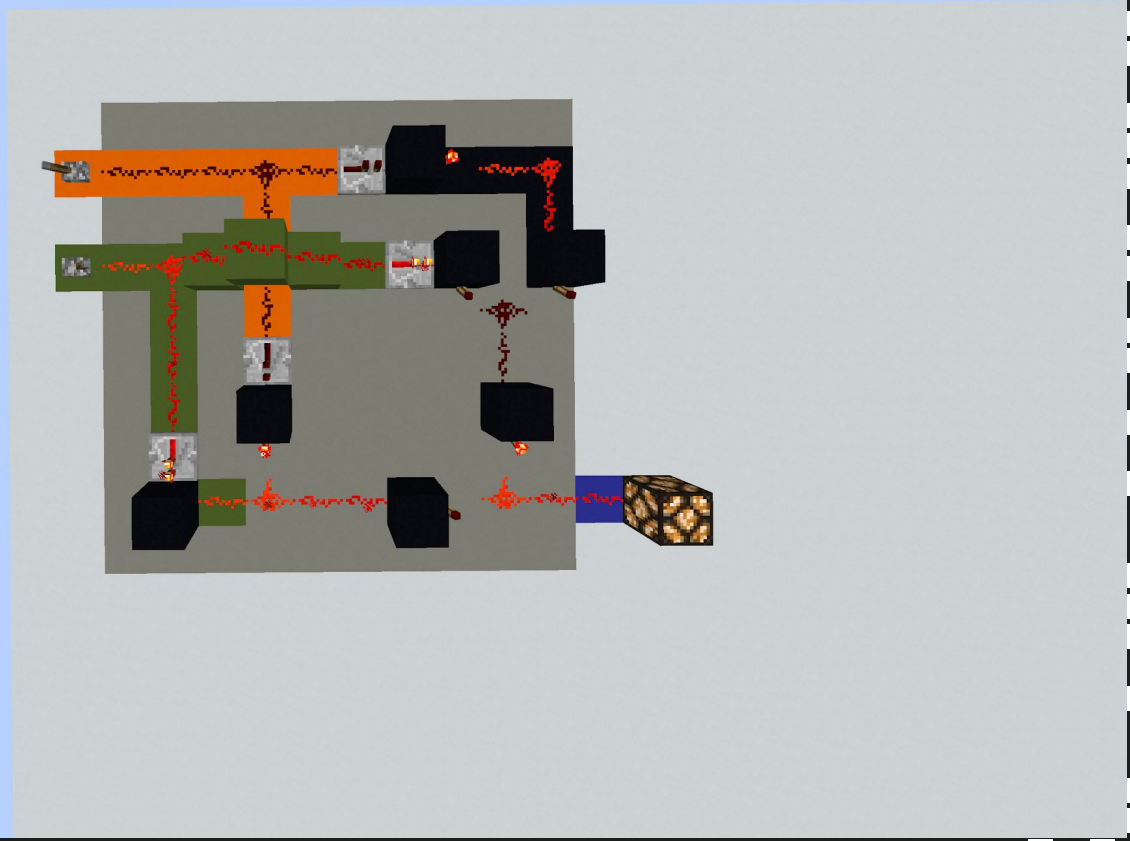


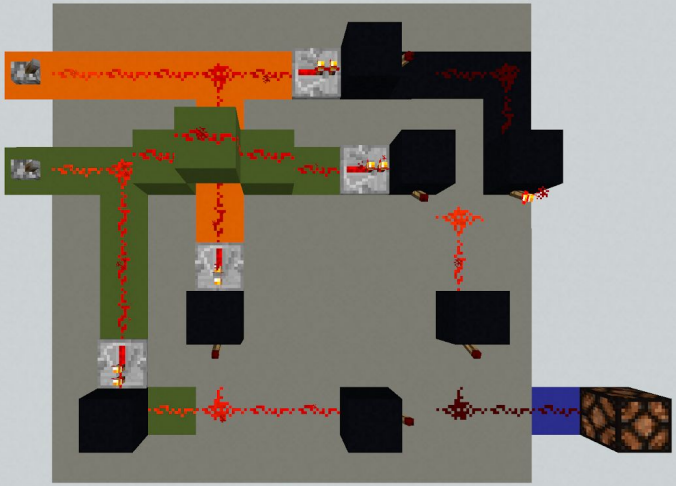
$$f(x, y) = x \oplus y$$

x	y	f(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

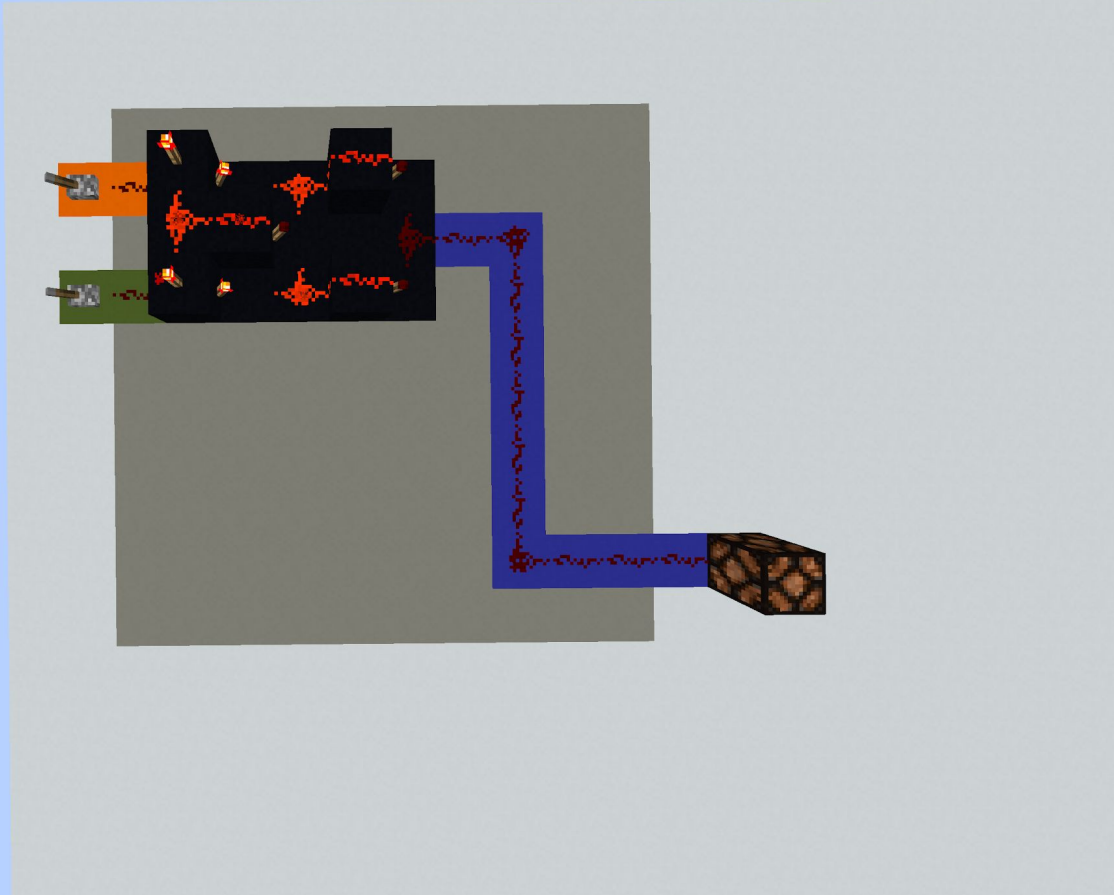


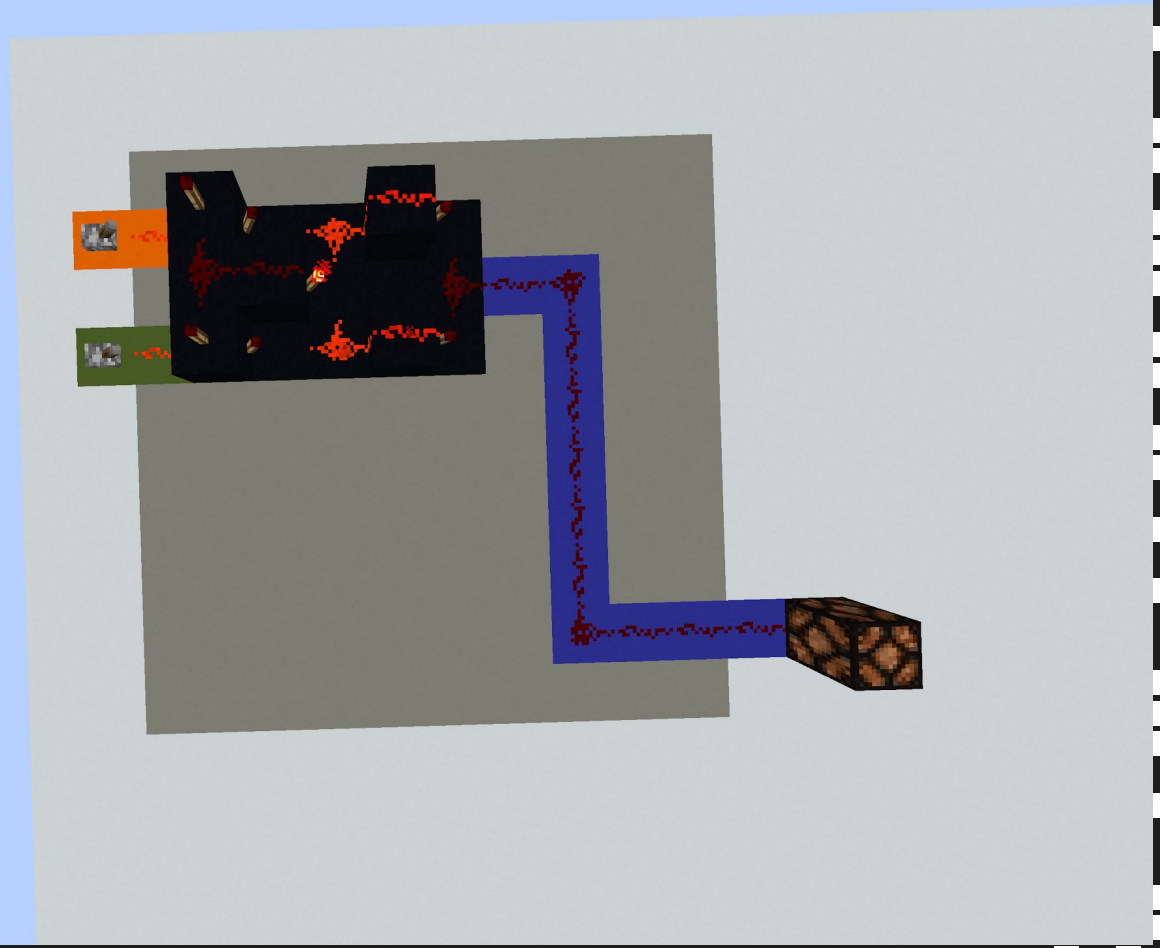


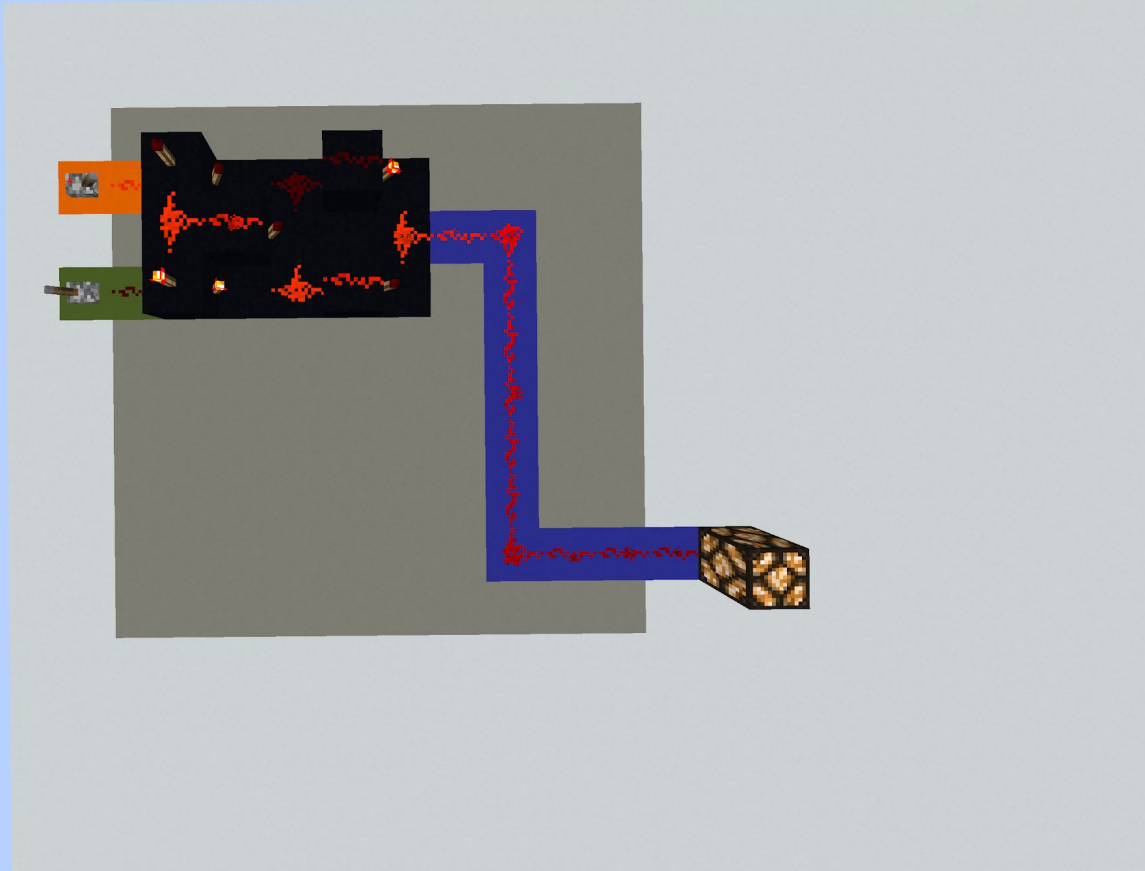




Compact XOR Gate

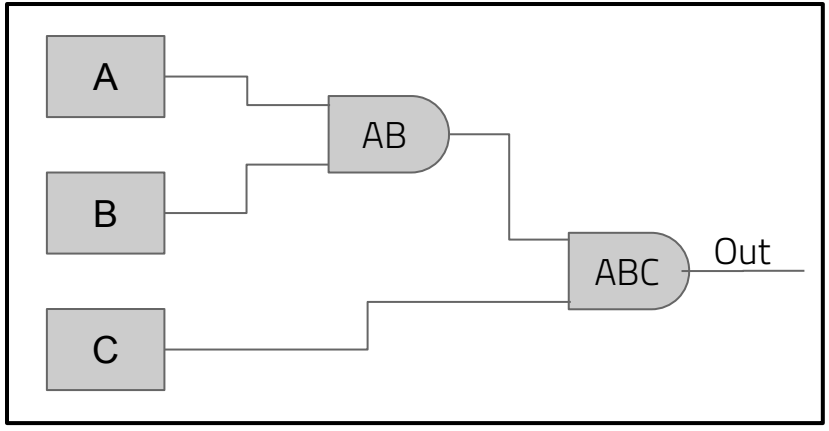




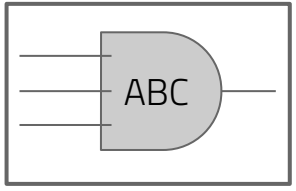


N -> 1 bit AND/OR/XOR gates.

- Represented as a gate with N input bits and one output bits.
- Chain gates together similar to below



Is same as



Multiplexor

Chooses between N inputs using a Control signal.

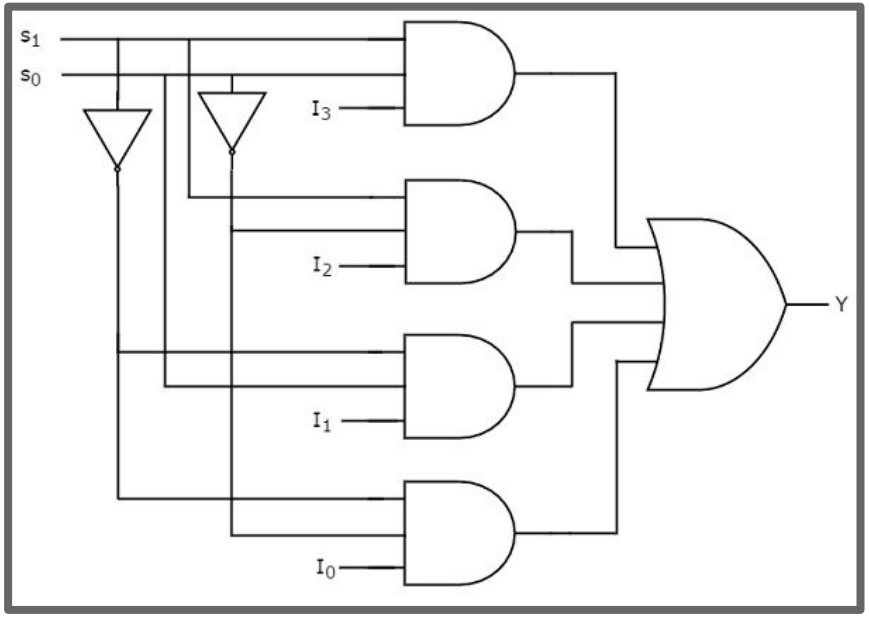
In the picture:

$S_0 = 0, S_1 = 0 \rightarrow I_0$

$S_0 = 1, S_1 = 0 \rightarrow I_1$

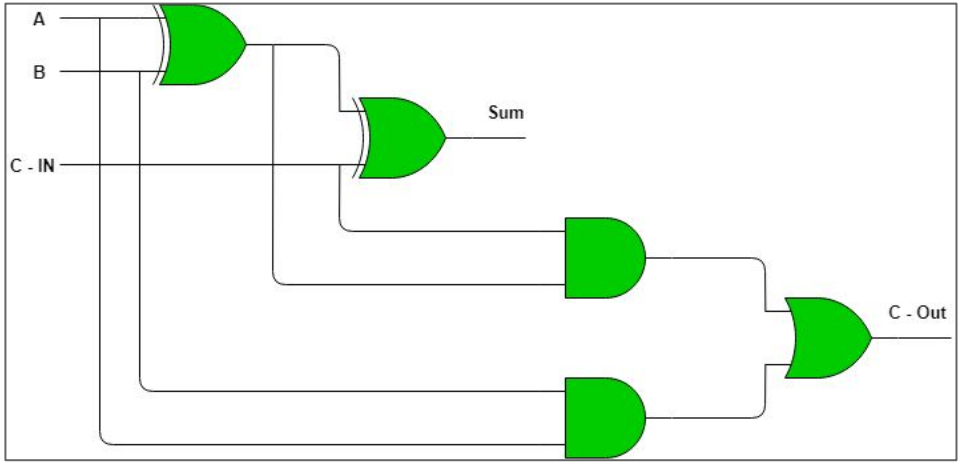
$S_0 = 0, S_1 = 1 \rightarrow I_2$

$S_0 = 1, S_1 = 1 \rightarrow I_3$



Full Adder

Adds two bits, and returns a sum as well as a carry out value, the carry out can be used if the user wants to chain multiple full adders together.

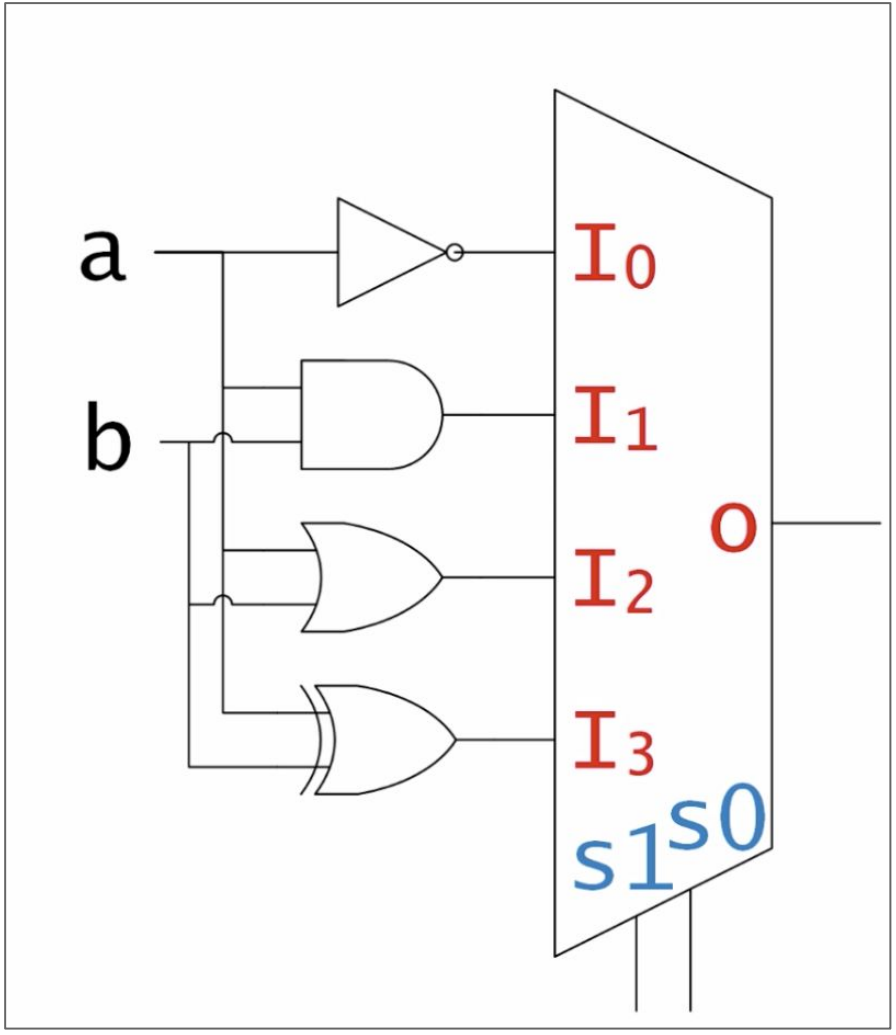


Logic Unit

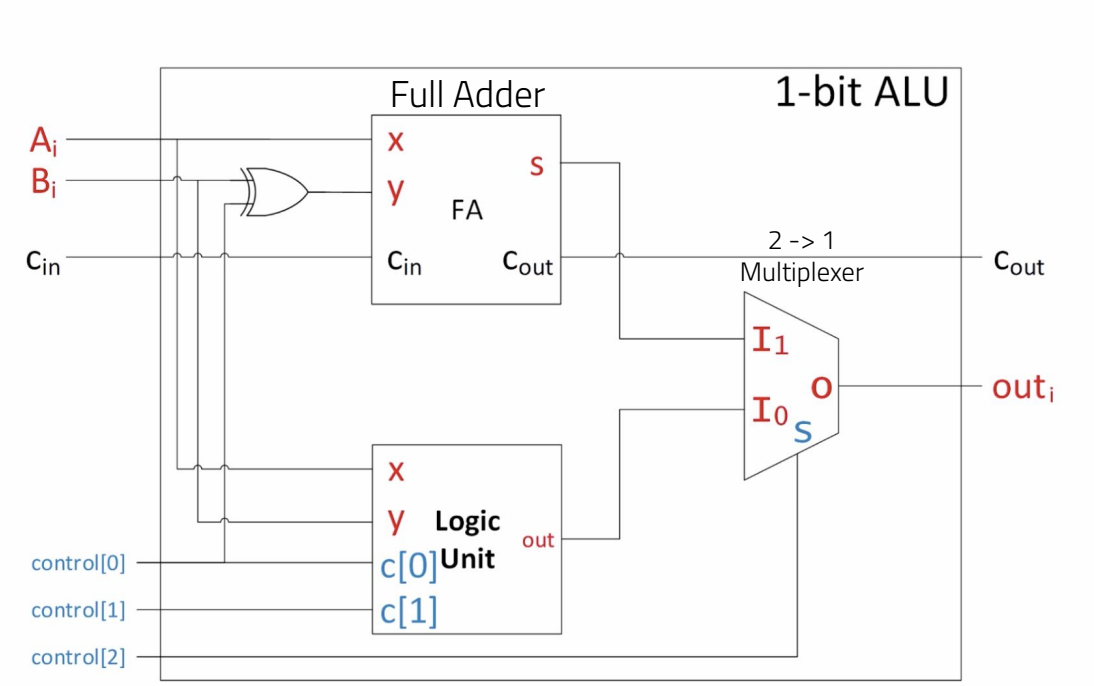
Takes two inputs.

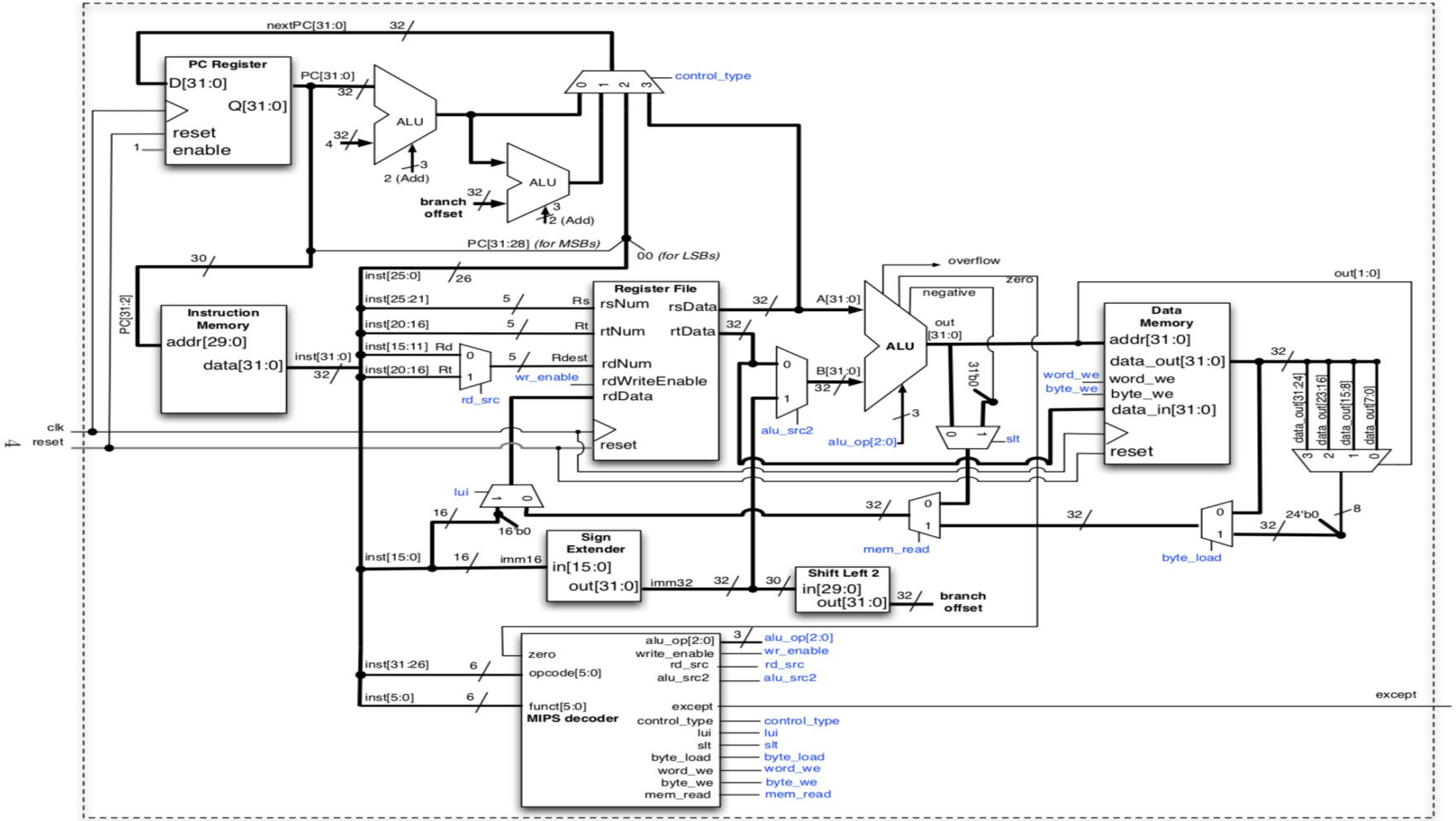
The control signals determine what kind of logic gate is used on the inputs.

Implemented using a 4-to-1 multiplexor.



One Bit ALU


Full Adder
1-bit ALU
A_i
B_i
C_{in}
x
y
FA
s
C_{in}
C_{out}
2 -> 1
Multiplexer
C_{out}
I_1
I_0
o
s
out_i
x
y
Logic
Unit
out
c[0]
c[1]
control[0]
control[1]
control[2]


nextPC[31:0]
32
PC Register
D[31:0]
Q[31:0]
reset
enable
PC[31:0]
ALU
control_type
2 (Add)
branch offset
PC[31:28] (for MSBs)
00 (for LSBs)
30
PC[31:2]
Instruction Memory
addr[29:0]
data[31:0]
inst[31:0]
inst[25:0]
26
inst[25:21]
Rs
inst[20:16]
Rt
inst[15:11] Rd
inst[20:16] Rt
Rdest
wr_enable
rd_src
Register File
rsNum
rtNum
rdNum
rdWriteEnable
rdData
rsData
rtData
reset
A[31:0]
B[31:0]
alu_src2
alu_op[2:0]
overflow
zero
negative
out [31:0]
slt
word_we
byte_we
Data Memory
addr[31:0]
data_out[31:0]
word_we
byte_we
data_in[31:0]
out[1:0]
data_out[31:24]
data_out[23:16]
data_out[15:8]
data_out[7:0]
clk
reset
lui
16'b0
inst[15:0]
imm16
Sign Extender
in[15:0]
out[31:0]
imm32
mem_read
byte_load
24'b0
Shift Left 2
in[29:0]
out[31:0]
branch offset
inst[31:26]
inst[5:0]
zero
opcode[5:0]
funct[5:0]
MIPS decoder
except
control_type
lui
slt
byte_load
word_we
byte_we
mem_read
write_enable
rd_src
alu_src2
alu_op[2:0]
wr_enable

Upper End of Basic Minecraft Redstone

https://www.youtube.com/watch?v=LGkkyKZVzug

https://www.youtube.com/watch?v=3sIMqgfKCf0

https://www.youtube.com/watch?v=Sb00tqH8f5I