Logic Gates, Mux, Binary Arithmetic

CS 2130: Computer Systems and Organization 1
September 1, 2025

Announcements

- Lab 1 tomorrow!

Putting it together

Overall idea:

- Only need two things (Shannon)
- We can do math with two things (Boole)

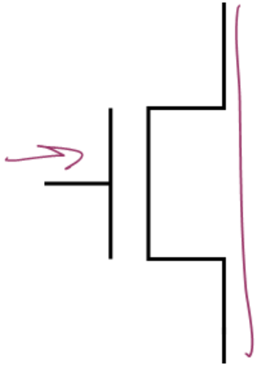
Putting it together

Overall idea:

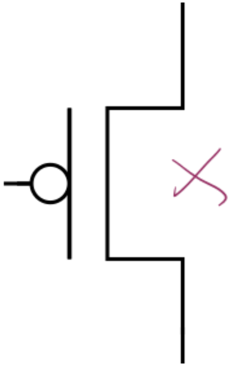
- Only need two things (Shannon)
- We can do math with two things (Boole)

Now we need a physical device that deals in two levels

Transistors



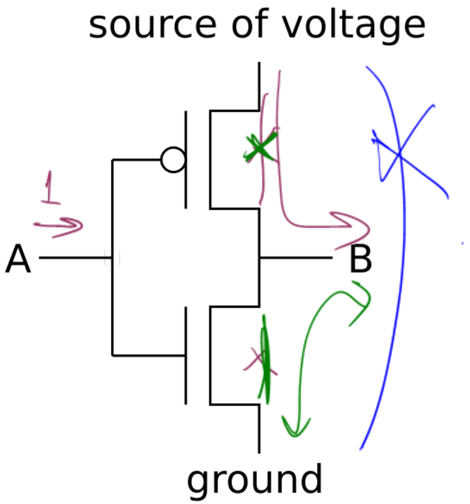
push to close
push to connect
push to allow current flow



push to open
push to disconnect
push to stop current flow

Circuit Diagram

A	B
0	1
1	0



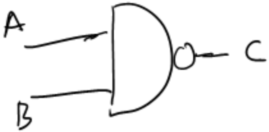
Circuit Diagram

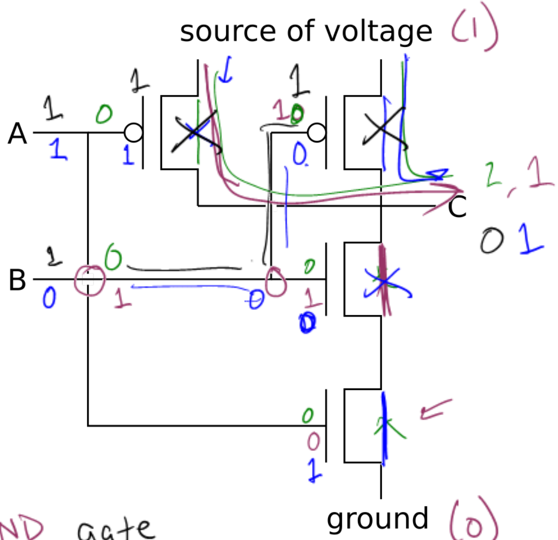
Welcome! Try this w/ your neighbors!

given values for A, B, what is C?

truth table

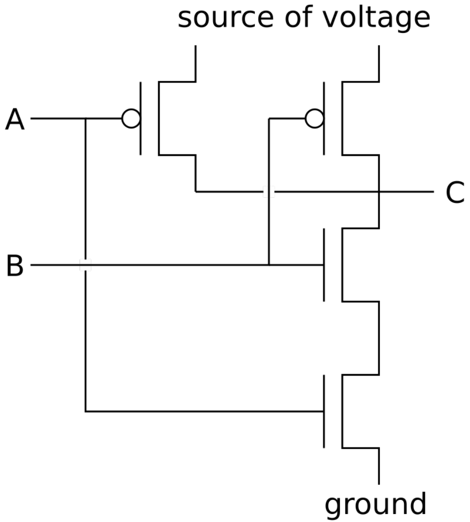
A	B	C	A & B
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1





NAND gate

Circuit Diagram

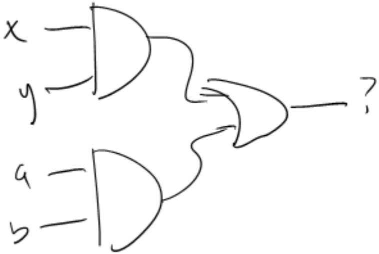


Other Gates (reading)

Building Up

Where we are now

- World with only 2 states: 0 and 1
- Re-developed Boolean logic (gates):
  - `and`, `or`, `not`
  - `nand`, `nor`, `xor`



Gives us everything Boole talked about

- Next: build higher level ideas, something powerful!

Trinary Operator

this will be key when we are constructing our computer out of these gates and circuits

Trinary Operator

- General idea:

```
if ( ... ) {
    ...
} else {
    ...
}
```

this will be key when we are constructing our computer out of these gates and circuits

Trinary Operator

- Python: `x = b if a else c`

this will be key when we are constructing our computer out of these gates and circuits

Trinary Operator

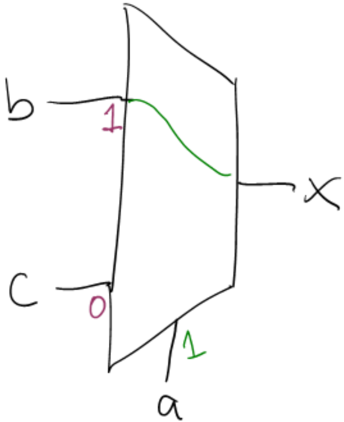
a	b	c	x
0			c
1			b

- Python: `x = b if a else c`
- Java: `x = a ? b : c`

this will be key when we are constructing our computer out of these gates and circuits

Multiplexer (mux)

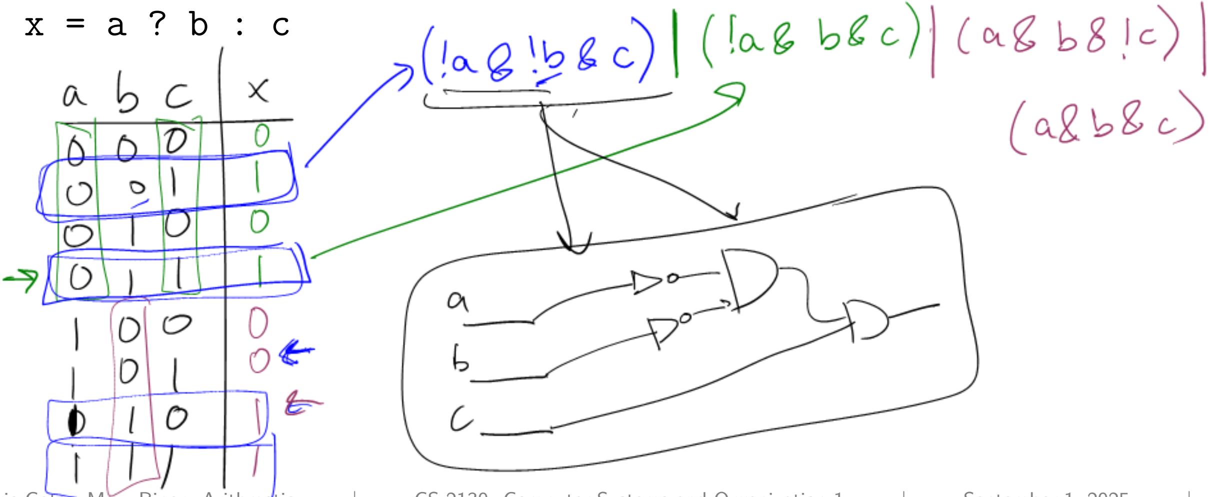
`x = a ? b : c`



Multiplexer (mux)

How can we build a mux out of what we have learned so far?

`x = a ? b : c`



Multiplexer (mux)

Can be built from `and`, `or`, and `not`

- Can be built using transistors
- Can physically put it in silicon!

Questions?

More bits!

2-bit Multiplexer (mux)

2-bit values instead of 1-bit values

Multi-bit Values

- So far, only talking about 2 things
- Numbers, strings, objects, ...

Numbers

From our oldest cultures, how do we mark numbers?

Numbers

From our oldest cultures, how do we mark numbers?

- **unary** representation: make marks, one per "thing"

Numbers

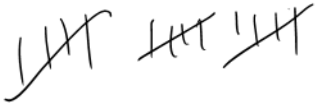
From our oldest cultures, how do we mark numbers?

- **unary** representation: make marks, one per ”thing”
  - Awkward for large numbers, ex: CS 2130?
  - Hard to tell how many marks there are

Numbers

From our oldest cultures, how do we mark numbers?

- **unary** representation: make marks, one per "thing"
  - Awkward for large numbers, ex: CS 2130?
  - Hard to tell how many marks there are
- Update: group them!



Numbers

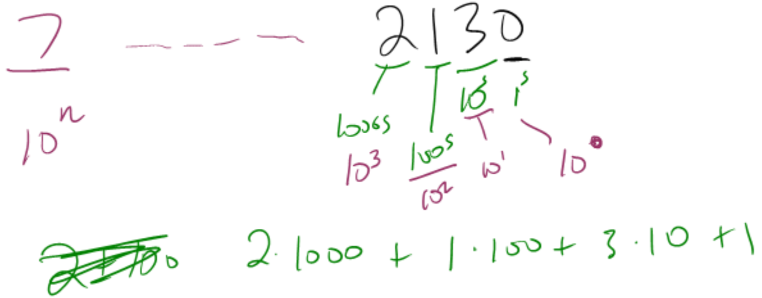
From our oldest cultures, how do we mark numbers?

- **unary** representation: make marks, one per "thing"
  - Awkward for large numbers, ex: CS 2130?
  - Hard to tell how many marks there are
- Update: group them!
- Romans used new symbols: V - X L C M

Numbers

From our oldest cultures, how do we mark numbers?

- Arabic numerals
  - Positional numbering system



Numbers

From our oldest cultures, how do we mark numbers?

- Arabic numerals
  - Positional numbering system
  - The 10 is significant:
    - 10 symbols, using 10 as base of exponent

0
1
2
3
4
5
6
7
8
9

Numbers

From our oldest cultures, how do we mark numbers?

- Arabic numerals
  - Positional numbering system
  - The 10 is significant:
    - 10 symbols, using 10 as base of exponent
  - The 10 is *arbitrary*
  - We can use other bases! $\pi, 2130, 2$, ...

Base-8 Example

Try to turn 134_8 into base-10:

$8^0 = 1$

$8^1 = 8$

$8^2 = 64$

$= 1 \cdot 64 + 3 \cdot 8 + 4 \cdot 1$

$= 92_{10}$

Bases

We will discuss a few in this class

- Base-10 (decimal) - talking to humans
- Base-8 (octal) - shows up occasionally
- Base-2 (binary) - most important! (we've been discussing 2 things!)
- Base-16 (hexadecimal) - nice grouping of bits