Boolean Algebra

CS 2130: Computer Systems and Organization 1

Xinyao Yi Ph.D.
Assistant Professor

UNIVERSITY of VIRGINIA | ENGINEERING

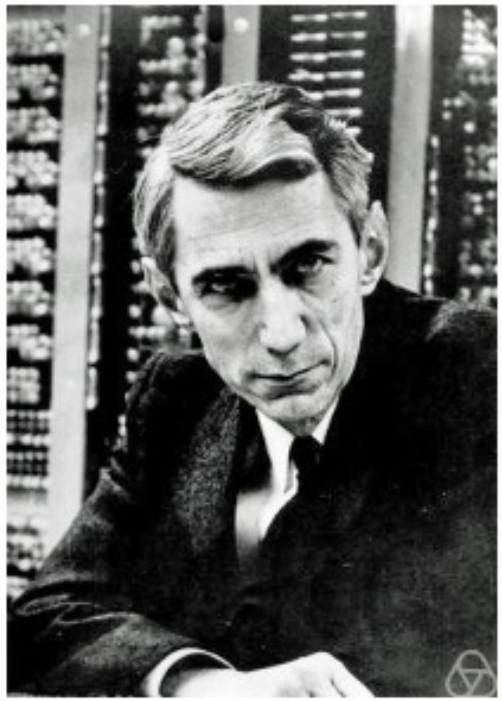
Announcement

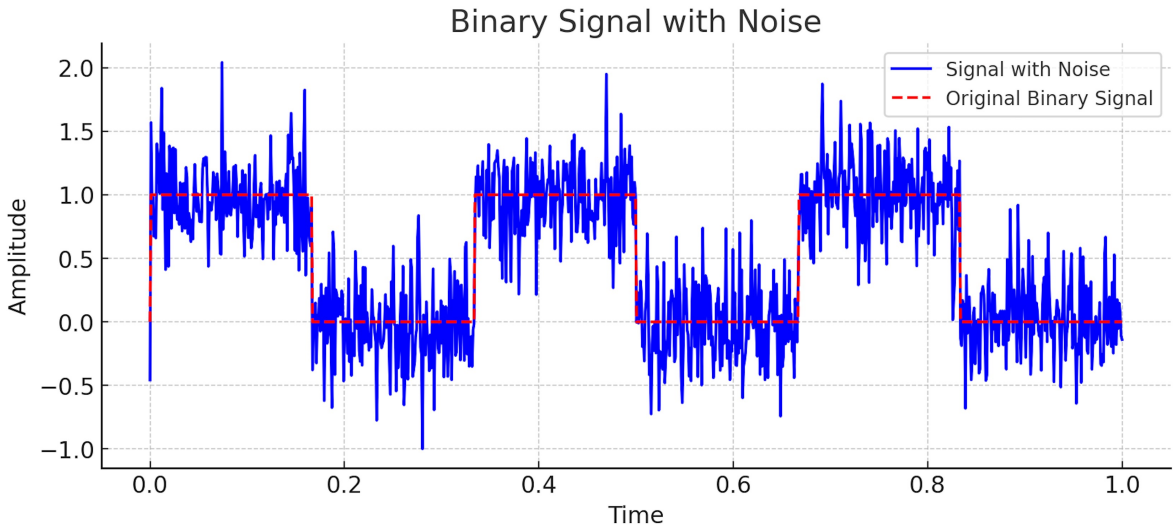
If you need to switch labs:

- Most lab sections still have available spots.
- Send me an email if you are unable to swap

Why only 0 and 1?

Claude Shannon





Why only 0 and 1?

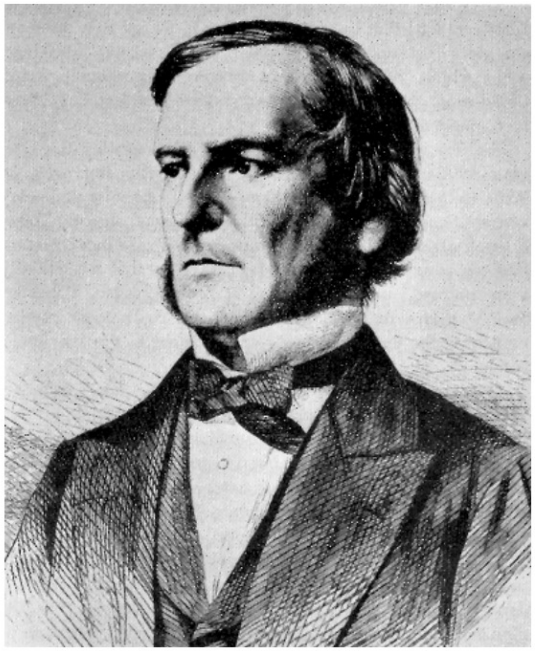
Vocabulary

- bit – either a 0 or 1

- binary - a system that has only two positions
- trinary - a system that has only three positions
- quadrinary - a system that has only four positions
- ...
- decinary - ...
- decimal - system that has ten positions

Boolean Algebra

George Boole



In Boolean Algebra, we live in a world with only two values:

- **True or False**
- **Yes or No**
- **1 or 0**

Boole showed that you could build an entire algebraic system using only these two values.

And that system uses three basic operations: **AND, OR, and NOT**."

Putting Them Together

Overall idea:

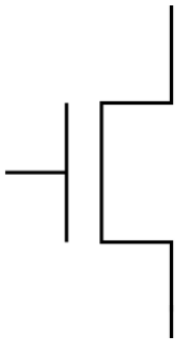
- Only need two things (Shannon)
- We can do math with two things (Boole)

Now we need a physical device that deals in two levels

Transistors

Electricity (conceptually) - involves flow of electrons or other charged carriers through a conductive material

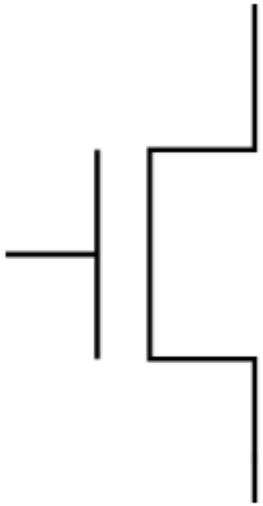
- current - rate of flow
- voltage - pressure of flow



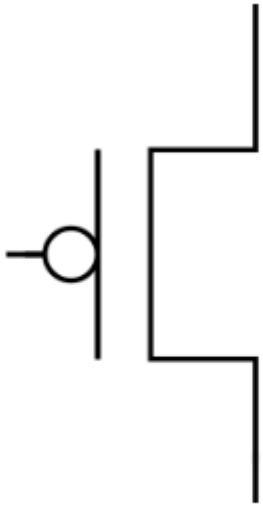
Transistors act like an electrically-triggered switch

- No voltage, no current
- Apply voltage to allow current to flow
- The amount of voltage needed to open the gate is boundary between 0 and 1
- Central technique for how we are going to build binary computers

Transistors

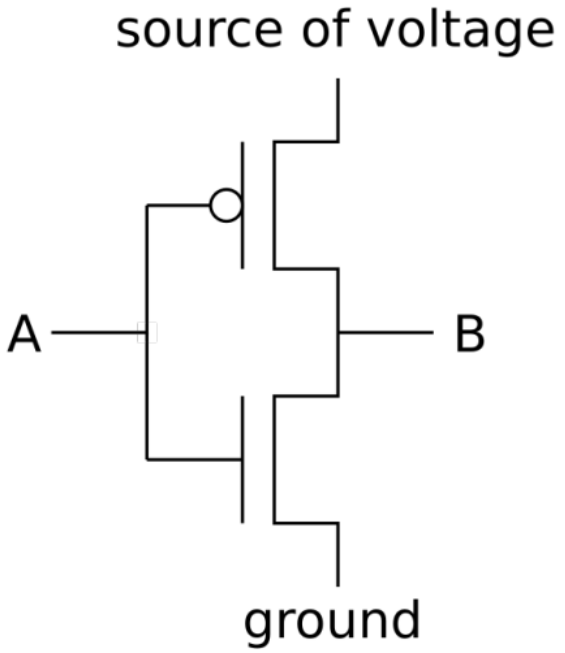


Push to close

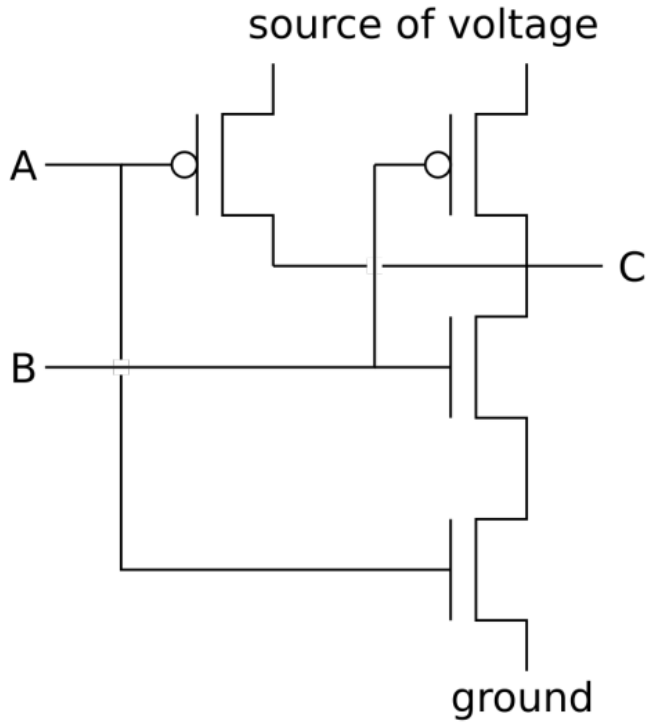


Push to open

Circuit Diagram



Circuit Diagram



Other Gates

Reading: https://uva-cs.github.io/cso1-s26/readings/bool.html

Building Up

Where we are now

- World with only 2 states: 0 and 1
- Re-developed Boolean logic: and, or, not

Gives us everything Boole talked about

- We can do a lot of interesting things!
- Next: build higher level ideas: the trinary operator

Trinary Operator

General idea

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

Trinary operator (expression if)

Python:

```
x=b if a else c
```

Java:

```
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)

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

x=a ? b : c

Can be built from and, or, and not

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

Mux will be the key when constructing a computer out of gates and circuits!

Multiplexer (mux)

x=a ? b : c

A multiplexer (mux) is commonly drawn as a trapezoid in circuit diagrams.

2-bit Multiplexer (mux)

2-bit values instead of 1-bit values

Any Questions?