Diodes, Transistors, Etc. . . Logic

Lab 01B

6.S188 IAP 2026

In the previous lab, we made foundational logic circuits using nothing but relays. Amazing. Now we'll make a few of them using slightly more modern components.

Inputs

Before we get to that, an aside on digital inputs. In this class we'll be using a lot of buttons. And we need those buttons to give High and Low Voltages reliably. One way to do this would be using a button or switch that connects to our high and low voltages (1 or 0, respectively) directly like shown:

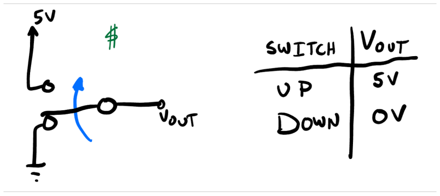

Figure 1: A (relatively expensive) SPDT switch

It turns out that a good Single-pole-double-throw switch like this is a bit more money (C.R.E.A.M.) than one would like and also is more complicated than it really needs to be. We can get effectively the same behavior with a single-pole-single-throw (SPST) switch. How would you hook it up?

One attempt might be something lke this:

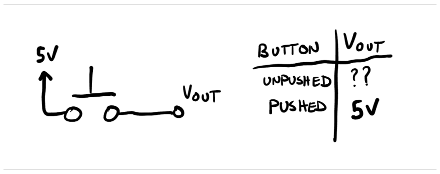

Figure 2: A SPST switch that either floats or connects to 1

At first glance, this might look "good". . . it connects to 1 or *not* 1. . . but it actually doesn't. It really connects to 1 or an undefined value. That undefined value could be a 1 or a 0. . . in is potentially random and based off of static electricity, the moons of Jupiter, etc. . . just not very reliable.

You could try this. . .

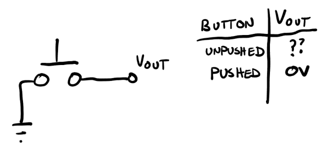

Figure 3: A SPST switch that either floats or connects to 0

But this is the same sort of problem as before, just in reverse. You can definitely say 0 or undefined (which may end up being 0 or 1).

The solution comes in using a resistor to electrically "position" the circuit in such a way that the voltage levels are defined in both states. Here is one way to do it. We call this a pull-down resistor. When the switch is unpushed, the output of the circuit is 0 (through the resistor path). When the switch gets pushed you then get a "strong" direct connection to 5V (aka 1) which is passed through to the output. Beautiful... no ambiguity in the two states. We call this an "active high" button since it has a high output when the button is "active"/pushed.

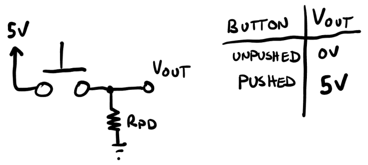

Figure 4: An active-low switch

You can do the same thing with a pull-up resistor as well in the configuration below. When unpushed, the circuit outputs a 1 (via that pull-up resistor's connection). When you then push, you get a direct connection to ground/0V/binary 0... We call this an "active low" button since it has a low output when pushed.

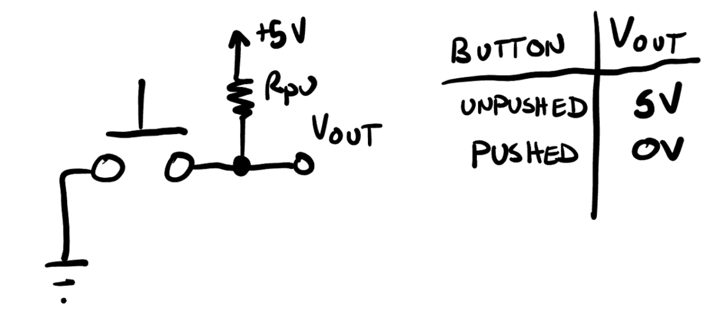

Figure 5: An active-high switch

What value should be used for the resistor? Well theoretically any value can be used, but in reality, they'll usually pick something in the order of 10 kilo-Ohms or so. That will ensure that a relatively small amount of current flows through the resistor during switch circuit's active mode. If you go too low, then you burn a lot of power in that resistor....if you go too high in value, then parasitics in the circuit may prevent it from working properly.

Anyways, you'll be using these inputs a lot in the labs. Being able to make these circuits quickly will be useful as we build various digital circuits.

Now onto digital logic building blocks...

Vacuum-Tube Logic

So relays were how they first did digital logic. They gave non-linearity and amplification, but they were slow and used a lot of power. The first replacement technology were vacuum tubes.

The key benefit that tubes brought (compared to relays) was speed and nominal power/efficiency improvements. Since vacuum tubes are purely electronic devices they're not dealing with the relatively slow behavioural mechanics that relays had (which limit their switching speed to 100's or 10's or even single Hz). Tubes gave us gain and non-linearity and could deliver it at very high frequencies (100's of kiloHertz or even MegaHertz). This was very important. The downside of vacuum tubes were two-fold though:

- Tubes were delicate. Vacuum tubes were literally glass tubes with vacuums in them and nature hates a vacuum, as they say, so the deck was stacked against them. Since one value proposition of digital electronics is you make up for binarization of signals by scaling up, this means vacuum-tube based computers needed lots of tubes…since the failure rates on vacuum tubes were relatively high, this meant it was very hard to keep a machine going for very long without at least one failure showing up and needing to be fixed. Very often a digital system would only work if *everything* was perfect and would fail to work if *anything* wasn't working (insert Anna Karenina happy family quote reference here).
- Tubes needed very high voltage. Compared to modern electronics, vacuum tubes had relatively high impedances in all their input and output ports, which meant that the only way you could get legitimate power out of them was using high voltages. As a result, it wasn't uncommon to run tube-based computers at 300V, which is easily lethal to humans. This made debugging very risky.

So tubes were better for sure… but not necessarily the best.

A replacement was needed for tubes and in 1947, the transistor was invented to fill in that gap. Transistors themselves were an outgrowth of early semiconductor research that had been going on since the last decade of the 1890s or so. The foundational device in that research was the semiconductor semiconductor diode.

Diode-Resistor Logic (DRL)

Diodes are simple semiconductor devices that were first developed in the very late 1800s. Interestingly, the first diodes were literally made by finding magical crystals on the ground (usually something like Lead-Sulfide II) and poking them with wires… almost like you would do in some sort of video game crafting mechanic.

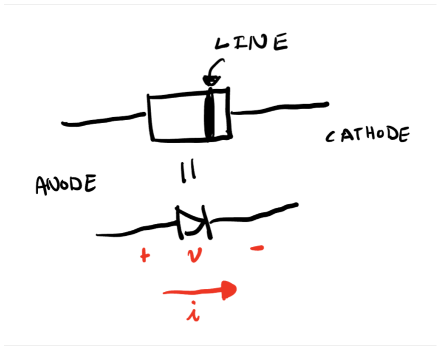

Figure 6: Diodes look like this… usually a tube with a line on one end to indicate their cathode.

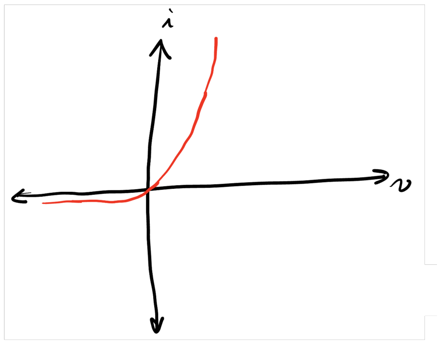

Figure 7: The I-V curve of a diode. Compared to something like a resistor or voltage or current source which will all be lines, the diode was the first reliable non-linear electrical component.

Diodes are important because their electrical characteristics can be *non-linear*. This is actually a big deal. Most regular electronics (resistors, etc. . .) are linear devices and linear devices cannot give you the sharp on-off behavior that we desire when making digital electronics. . . think about the I-V curve of the relay in the previous lab. . . you can't get that from just resistors. . . but you can get something pretty close using diodes. Often times in designing we'll just think of diodes in their "ideal form" which has a I-V curve like shown below:

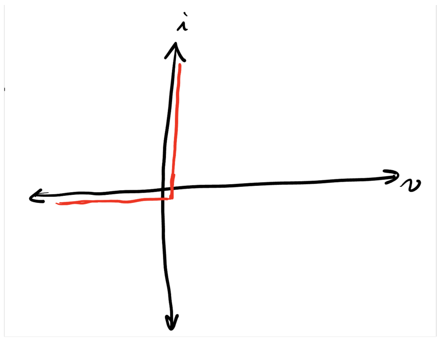

Figure 8: An ideal diode I-V curve

What this means is the diode will conduct electricity in one direction but not in the other. It is like a one-way wire, if you will. This weird asymmetric behavior allows interesting circuits to be built in a variety of fields (including digital logic).

Diode-resistor logic (DRL) represents the simplest form of solid-state logic implementation, using only passive components (diodes and resistors) to create basic logic functions. In a diode logic AND gate (shown below), multiple diodes have their anodes connected to separate inputs and their cathodes joined at a common output node with a pull-up resistor to the positive supply. The output is high only when all inputs are high; if any input is low, that diode becomes forward-biased and clamps the output to the low voltage. Similarly, an OR gate connects diode cathodes to inputs and anodes to a common output with a pull-down resistor.

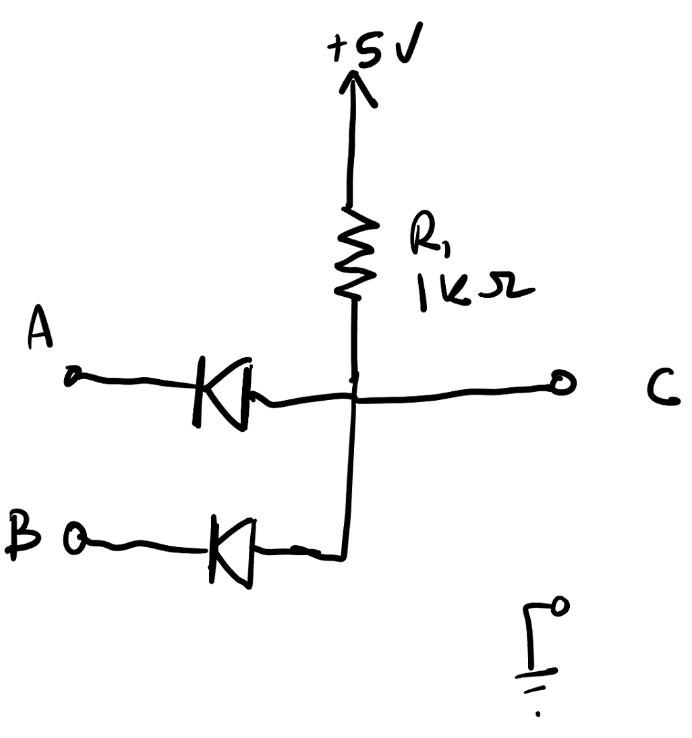

Figure 9: A Diode-Resistor AND Gate.

If you drop in our ideal diode model for the four possible "digital" input values for this circuit you'll get the following which verifies that it is indeed an AND gate in its behavior (output 1 only when both inputs are 1).

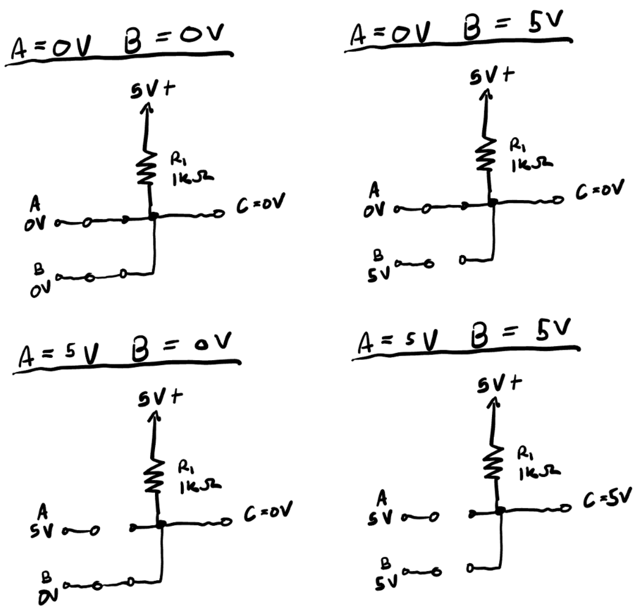

Figure 10: A Diode-Resistor AND Gate analyzed using ideal diode models

While simple, DRL suffered from fundamental limitations that prevented its widespread adoption. Each stage introduced a voltage drop due to the fact that real diodes don't act exactly like ideal diodes. . . each diode only starts conducting after a certain amount of forward-voltage has been built up. The forward-biased diode (approximately 0.7V for silicon, 0.2V for Germanium) means actual output logic levels degraded progressively through cascaded stages until signals became unreliable. . . .you couldn't stack one logic gate onto another and then onto another. The signal drop from the diodes built up too much. For example, here's the four cases above with more realistic results from the built-up drop that each diode introduces:

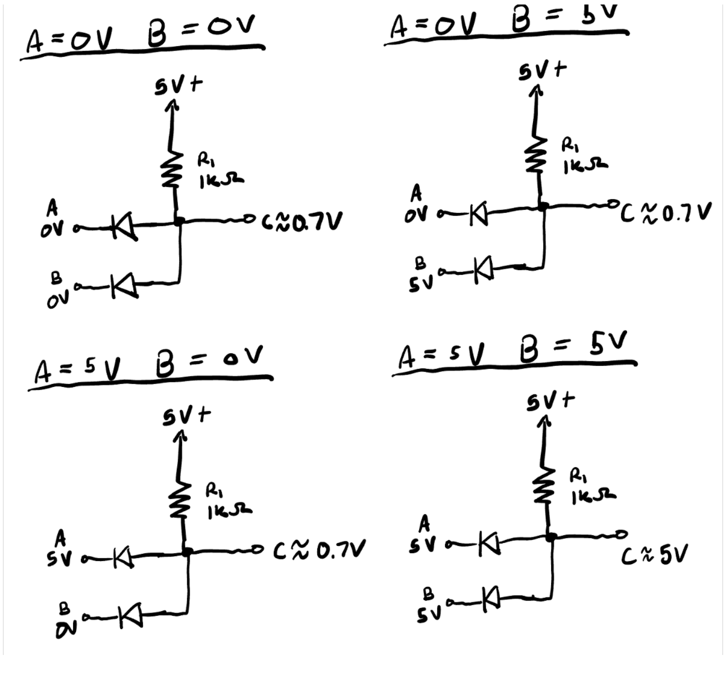

Figure 11: A Diode-Resistor AND Gate analyzed using non-ideal diode behavior

Now you look at that and say, "Who cares. 0.7V is still low enough to count at binary"0" so who cares. I don't care. Good enough" But If you wanted to stack this logic into more complicated systems like this…eventually you'll not be able to feed low enough voltages into the next stage to count as "0".

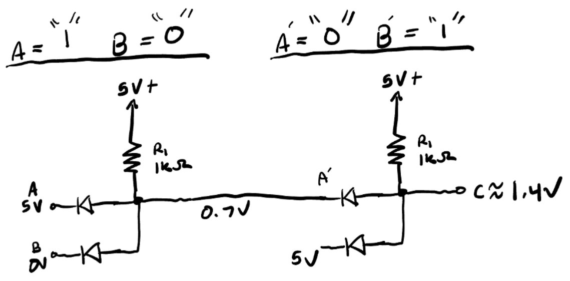

Figure 12: Chaining DRL circuits together would lead to signal integrity degradation

The inability of these circuits to "restore" signals back to very "pure" binary 0 or 1, was a real problem.

Still, for small circuits, you could build diode-Resistor forms of AND gates, OR gates, and XOR gates. The AND gate above is here mostly just as a curiosity / thought exercise, but you can build it if you want; that said, if you do want to build it, you probably want to use different resistor values (maybe like 100kΩ for R_1 and 1kΩ for your pull-down resistors on A and B).

But even if you don't build the AND gate, one thing you *should* build is an XOR gate (shown below…note that the little diode over by "C" is an LED to show us the output value visually, but the other diodes are just regular diodes):

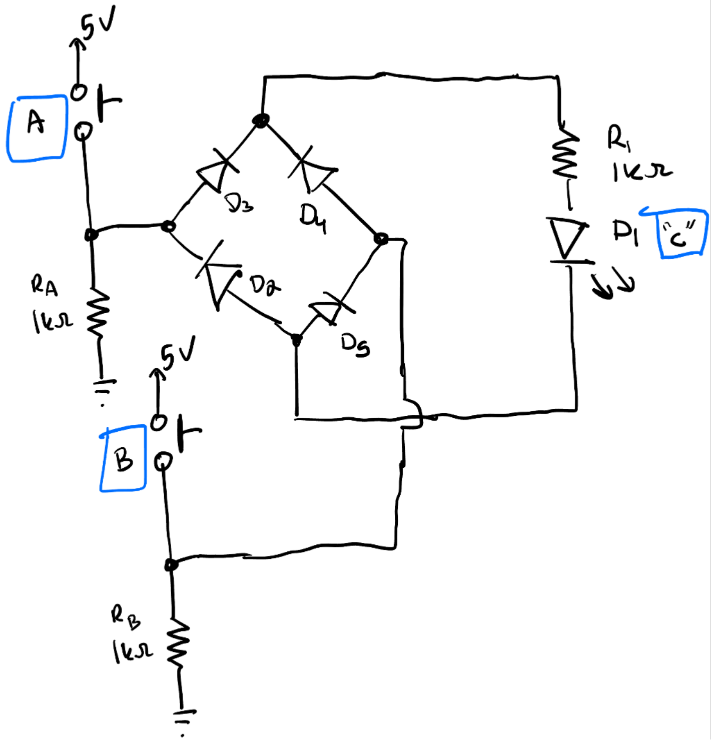

Figure 13: A Diode-Resistor XOR Gate. Build this with parts in lab. Make sure it actually does Exclusive OR! Feel free to ask us if you want to talk about how it does what it's doing.

All of these logic gate implementations can operate at very high speeds (perhaps not by modern standards, but definitely when compared to relay-based logic) and could work at ridiculously low voltages (several volts as opposed to the 100's of volts of vacuum tube-based logic or even the tens of volts of most relays). So somewhat of a win.

Importantly though, diode-based logic on its own cannot provide signal inversion (NOT function as well as the related NOR, NAND, and XNOR gates), making it impossible to build a complete logic family on its own (yes XOR can yield NOT, but since the XOR circuit above has a somewhat differential output, it isn't quite usable like we'd want). So aside from certain special-use-case situations you very rarely saw widespread deployment of diode-resistor logic…just wasn't scalable.

The signal drop and the lack of inversion could both be solved by an active amplifying device, which for a little while was the vacuum tube before eventually be replaced by the transistor when it became more reliable.

As a result you very rarely saw just diode-resistor logic…it was usually diode-tube logic (where the tubes would provide amplfication that could restore signals to proper 1's and 0's) or by the early 1950's, you'd have diode-transistor logic.

Diode Tranistor Logic

Transistors were invented in 1947 and pretty quickly came in as replacements to vacuum tubes in many areas. In spirit, they worked like scaled down vacuum tubes (small input electrical signal could influence larger output electrical signal).

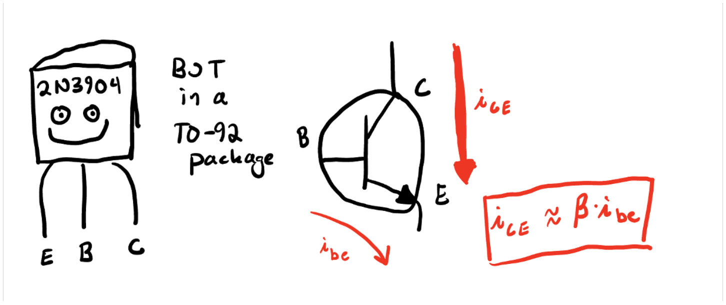

Figure 14: The 2N3904 NPN Bipolar Junction Transistor. Small base current modulates much larger emitter current

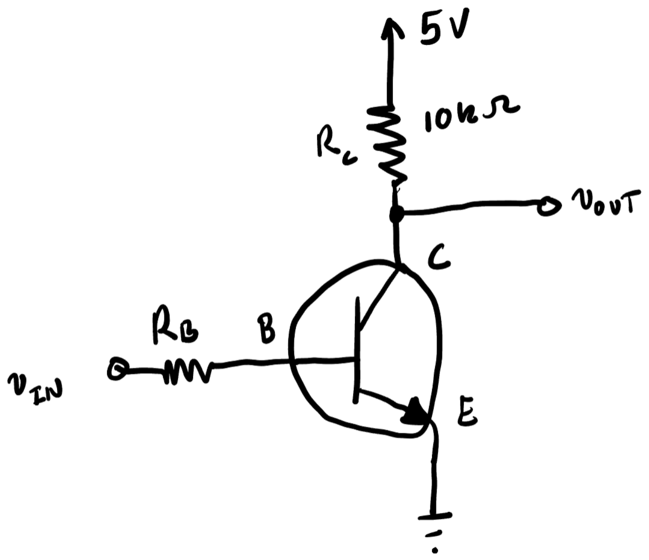

Figure 15: A BJT-based Inverting Amplifier. A high input voltage induces significant base current which causes a lot of collector current causing a large voltage drop in R_C, leading to a low output voltage. A low input voltage induces no collector current, meaning the output stays a high voltage. This is an inverter in the digital sense.

Diode-transistor logic represents one of the earliest practical approaches to implementing digital logic gates using solid-state components. In DTL circuits, diodes perform the basic logic functions while transistors provide signal amplification and inversion. A typical DTL NAND gate uses diodes at the inputs to implement the AND function—when all inputs are high, current flows through all diodes to charge a capacitor or drive the base of an output transistor. The transistor then inverts this signal, completing the NAND operation. DTL offered significant advantages over earlier diode logic by providing signal restoration and the ability to drive multiple subsequent stages without degradation. However, DTL circuits suffered from relatively slow switching speeds due to charge storage in the transistor base region.

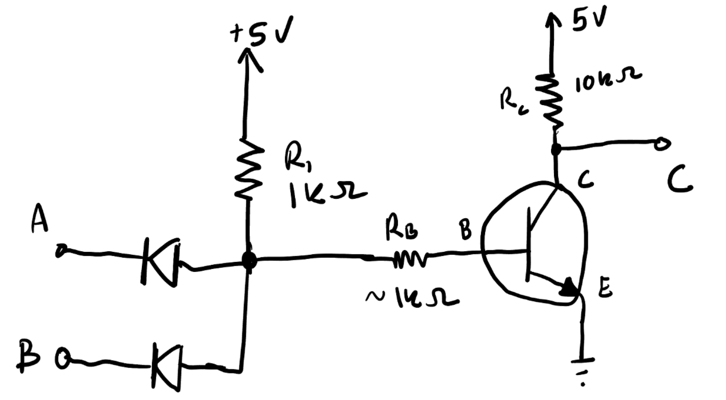

Figure 16: A Diode-Transistor NAND Gate. (don't worry about building this, just including it for reference)

Resistor-Transistor Logic (RTL)

Resistor-Transistor Logic (RTL) simplified the DTL approach by eliminating the input diodes and using resistors to combine input signals directly at the base of a transistor. In a basic RTL NOR gate, multiple input resistors connect to the base of a common transistor—if any input goes high, the transistor turns on and pulls the output low, implementing the NOR function. If you want better input isolation (so A doesn't affect B), multiple transistors could be used instead like shown below. RTL circuits were simple and economical, requiring only resistors and transistors, which made them attractive for early integrated circuits. The primary advantages of RTL included its simplicity of design and low manufacturing cost.

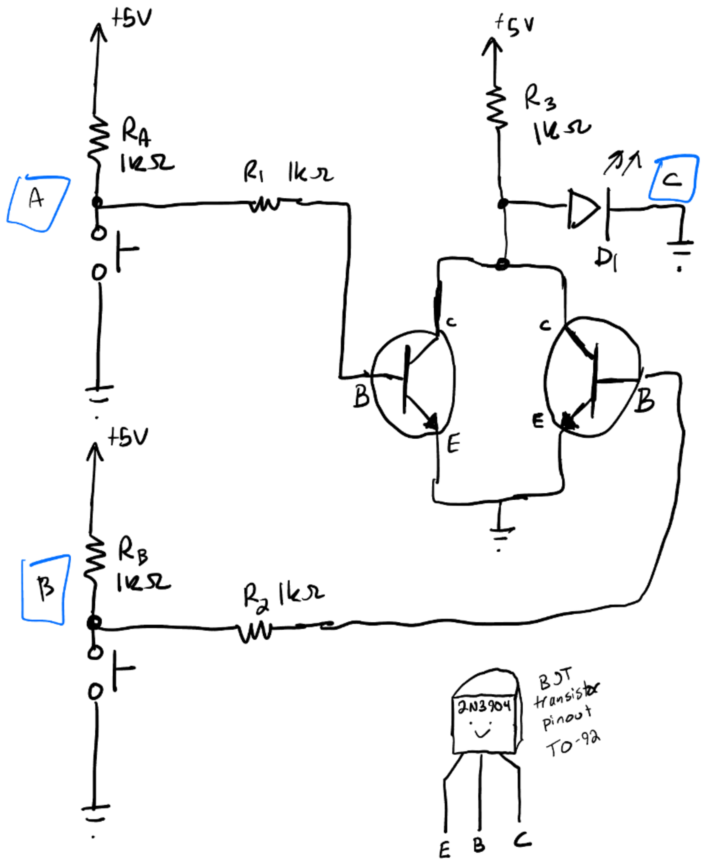

Figure 17: An RTL logic gate

Transistor-Transistor Logic and CMOS

Transistor-transistor logic (TTL) emerged as a major improvement over both DTL and RTL by using transistors for both the input logic function and output drive. In true TTL, a multi-emitter transistor at the input effectively performs the AND operation, while subsequent transistor stages provide inversion and strong output drive capability through a totem-pole output configuration. TTL became the dominant logic family for decades due to its excellent speed, good noise immunity, and ability to drive many loads. Many famous logic series (discussed below) like the 7400, 74LS, etc.. were all TTL.

However, the evolution of digital logic ultimately led to complementary metal-oxide-semiconductor (CMOS) technology, which represents a fundamentally different approach. While TTL uses bipolar junction transistors that conduct current continuously when active, CMOS employs complementary pairs of n-channel and p-channel MOSFETs arranged so that only one path conducts at a time, with near-zero static current flow in either logic state. This complementary arrangement means CMOS circuits consume power primarily only during switching transitions, resulting in dramatically lower power consumption—often orders of magnitude less than TTL. CMOS also proved extremely friendly to scaling down in size. Most modern digital systems overwhelmingly use CMOS technology because its low power consumption enables complex integrated circuits with billions of transistors, though TTL's still persists in certain in the edges in bespoke applications.

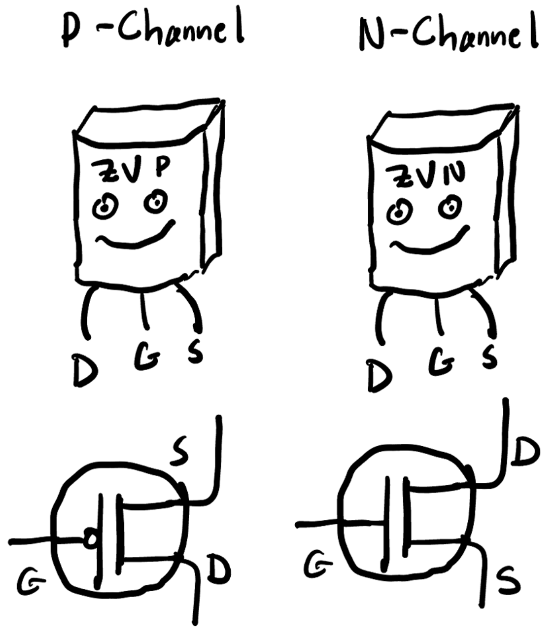

Figure 18: MOSFETs. Two types. They complement one another. Voltage modulates current flow in complementary ways.

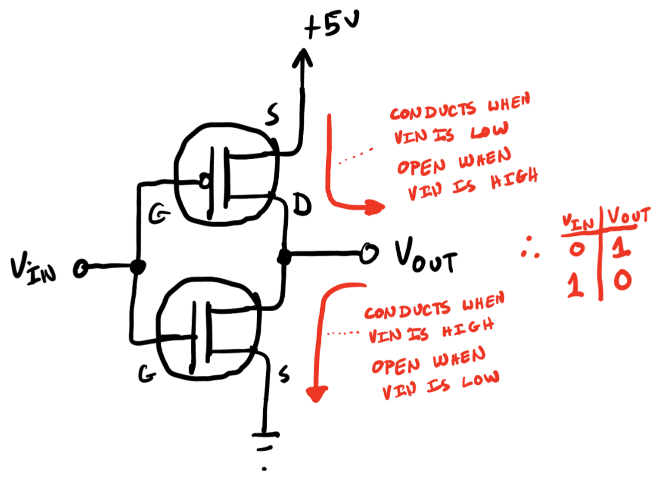

Figure 19: The CMOS Logical Inverter. Feel free to build!

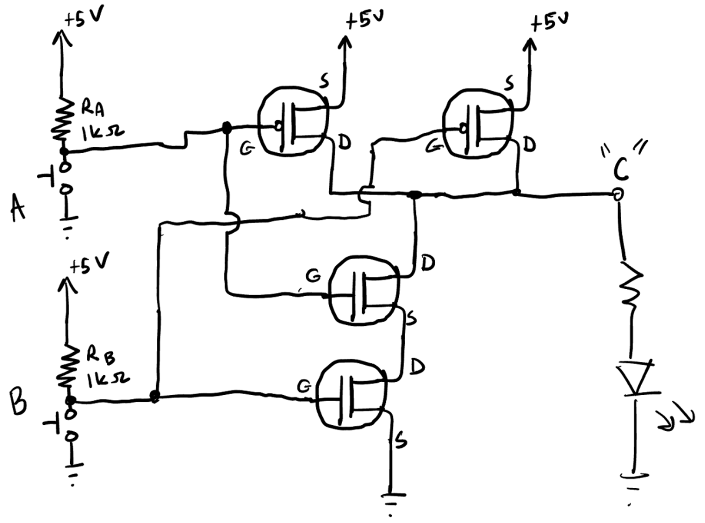

Figure 20: A CMOS Gate. Feel free to build! What kind of operation is this gate performing?

When you're building these, make sure you're using the right components! We also have some BJT's lying around, so **don't use anything labeled 3904**! The p-channel MOSFETs are ZVP4105A, and the n-channel MOSFETS are ZVN3306A (there are little numbers printed on them; they might be a bit hard to see but they'll tell you what part you're working with!).

Integrated Circuits

In the early 1960s Texas Instruments finally got transistors and their process technology under control to the point that they started selling pre-made logic gates to the wider public. This was revolutionary since it meant digital designers didn't have to build their logic designs at the transistor/circuit level like you've been doing in this lab. They took care to make sure all the gates would work together almost like lego bricks. In order to achieve this they established logic "families" which were series of chips that all were guaranteed to work together (had the same definitions of "Low" and "High" voltage), etc…

The 7400 series was groundbreaking in its influence and reach. The original few chips in the family were just two-input logic gates covering all your bases (NANDs, NORs, NOTs, ANDs, XORs, etc…). These integrated circuits were usually called "Small Scale Integration" or "SSI". After a little while they started adding more and more into single chips making more complicated circuits like adders, shift buffers, etc… these became known as "Medium Scale Integration" or "MSI". A few years after that when early processors (like the Intel 4004 which was their first processor) things were so complicated they were getting labeled "Large Scale Integration" or "LSI". Then after that things just got weird with the kinda vague term "Very Large Scale Integration" (VLSI), and all of its follow-ons and interpretations.

The 7486 and its variants was an early SSI chip featuring four two-input XOR gates (and a shared +5V/GND connection for all four) in a simple fourteen pin package. We have some 74LS86 chips in lab. Grab one, power it up, and test out its inputs and outputs with an LED. Look you didn't even need to touch a transistor or relay or diode…it was just a black amorphous magic box.

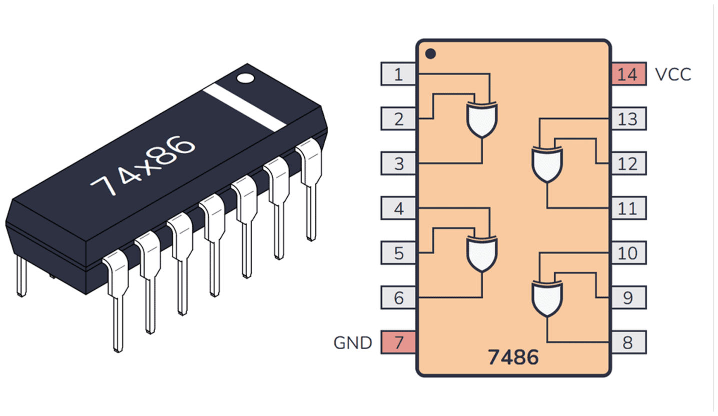

Figure 21: A 74LS86 quad XOR gate. Simply apply power and you have four two-input XOR gates at your disposal. This was luxurious. When building this, use 1K ohm resistors for your pull-down resistors! The 74LS86 has non-negligible input current which in conjunction with larger pull-up/down resistors can sometimes lead to unreliable behavior.

Logic packages like this was the first time components were getting made with lots of pins on them. Early integrated circuit packaging was usually in weird can-like things similar to transistors, but this didn't scale. The Dual-Inline-Package (DIP) was created to make routing easier.

And with that... we now have all the logic that we'll be working with this in 6.S188. And next week we'll start thinking about putting these together to build even more complex and even more cooler circuits.

Remember Your Roots (Optional)

The digital logic that emerged in the 1960s in DIP form (and then later smaller and smaller packages) worked so well that it was easy to forget that they actually were still just transistors wired together at the end of the day. And transistors are NOT digital devices. They are analog devices that we assemble into certain circuits and make behave in digital ways. Much of the "battle" with transistors as we have scaled them over the past decades is focused on trying to make them act like switches when they really don't want to... we make a smaller transistor... it now acts less like an ideal switch... so we modify the design (both device dimensions and circuits)... and repeat.

As a reminder of the actual situation with digital logic, let's take a very simple "digital" chip... the 74HC04. It is a 14-pin six inverter package of the 74HC CMOS family of chips released in the mid 1970s. Each one of its inverters is basically the CMOS inverter circuit you built up above (just a single complementary pair of MOSFETs).

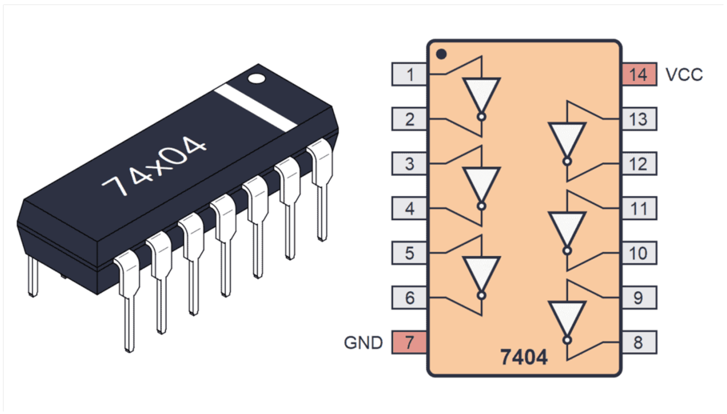

Figure 22: A 74HC04 six-count inverter package. To use, make sure to give it 5V and 0V.

Now normally this device just acts as a digital inverter... you put in a high voltage... it makes a low output voltage... and you put in a low input voltage... you get a high output voltage.

But it really is just a transistor inverting amplifier in its actual heart of hearts. If we instead, incorporate it into this circuit below (with negative feedback just like you'd do with an operational amplifier), you can actuall get an analog amplifier with a gain of ~10X.

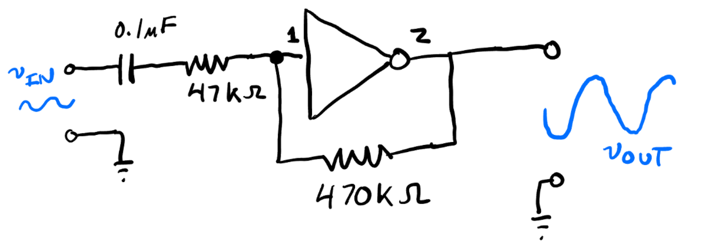

Figure 23: With some resistors and a capacitor and some negative feedback we can reveal that there's really just a nice little analog amplifier underneath here. Its parents wanted it to be a digital circuit, but it went to college and found its true self.

We should have some function generators and scopes floating around lab in case you're curious to try this... but it is pretty crazy... here's a scope grab of the input and output with very clear *analog* gain being demonstrated from a nominally digital device.

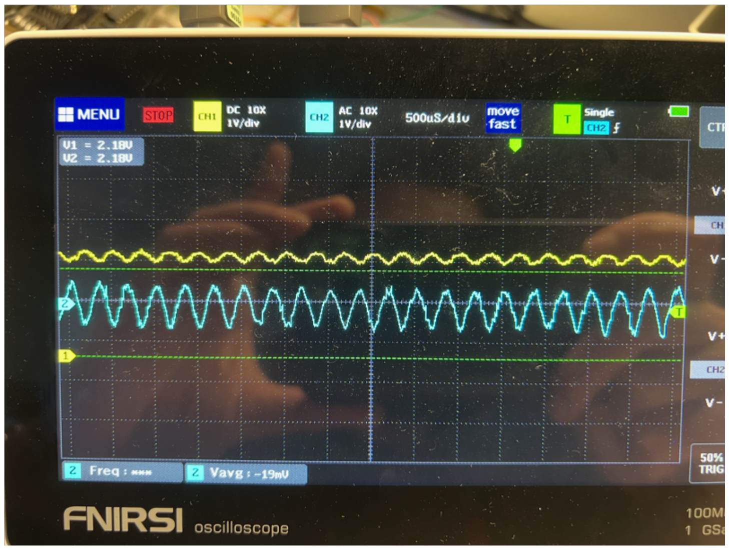

Figure 24: We should have s

OK next week we start building with these chips.