

Logic Bloxⁱ

Alex Kessler

Olin College ENGR 3410: Computer Architecture

December 19, 2013

Note: This document was prepared with future Oliners in mind. Consequently, it is stuffed full of snark, sass, and irreverence (especially the end notes.) However, this document and the accompanying zip file should contain enough information for someone in a future class to pick up where I left off with only 1-3 hours of repeated effort.

Table of Contents

Table of Figures	2
The Documentation	3
What did you do?	3
Why did you do it?	3
How did you do it?.....	3
How can someone else build on it?.....	8
Difficulties.....	8
Work Plan Reflection	9
To Do	9
End Notes.....	10

Table of Figures

Figure 1: Schmitt triggered delay with OpAmp	4
Figure 2: AND gate, according to simulation	4
Figure 3: XOR Logic.....	5
Figure 4: OR gate.....	5
Figure 5: Half Adder Schematic	6
Figure 6: AND gate holder.....	7
Figure 7: OR gate holder	7
Figure 8: Holder size comparison	8
Figure 9: AND gate PCB Layout	8

The Documentation

What did you do?

I createdⁱⁱ Logic Blox, which allow anyone to learn about logic gates in an easy, accessible way.

Computers are incredibly complex machines, but at a fundamental level they are all comprised of logic gates and transistors. Logic Blox take a look at the individual logic gates, AND, OR, and XOR (exclusive or) – as well as how they can be converted to NAND-only logicⁱⁱⁱ to build larger components.

They also introduce users to the concept of propagation delay – that is, the amount of time it takes for a gate to change state. While this happens on the scale of a few hundred nanoseconds inside a computer, Logic Blox extends this to a more human-friendly ~ 0.7 seconds.

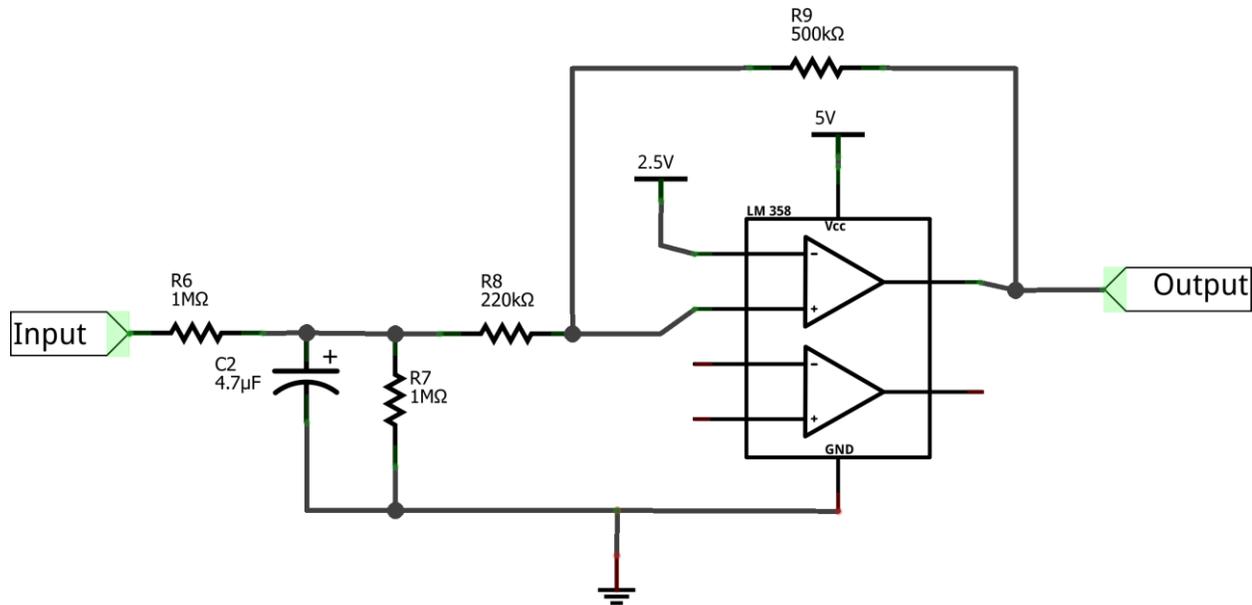
Why did you do it?

Since the first or second week of class, I'd thought it might be interesting if there was a way to play with logic gates like LEGOs. To be fair, I didn't come up with the idea (sorry FBE team from last semester, I don't know who you are to credit you with the idea.) Nonetheless, the idea of being able to combine them into a piece of technical wall art that you could rearrange at will to create new structures was pretty intriguing. Trying to make it really easy to visually follow the path of "data" through the circuit was also a big priority.

How did you do it?

Since I knew I wanted to start at the transistor level and work up from there, I began by doodling what I thought might make AND, NAND, and OR in my notebook during classes. I also looked up the diagrams for half and full adders from the class notes, and took a stab at converting them to NAND logic. I had no idea how to create XOR, so I wound up just Googling it^{iv}. Later on in the process, when actually trying to test out gates on a breadboard, I realized that the way I implemented my propagation delay almost completely killed the signal. I wound up stealing and modifying an amplifier I found during that trip through Google.

While I was putting everything down on paper, I discovered Little Bits^v – snap-togetherable circuit pieces of exactly the variety I was trying to make. Nonetheless, once I had all my circuitry down on paper, I tried actually implementing things on breadboards. Since I now wanted to make mine different from the commercially available option^{vi}, I focused heavily on the propagation delay component. Thus, I began with trying to make a Schmitt Trigger. Using the Wikipedia page^{vii} as a guide, I wound up with the following:

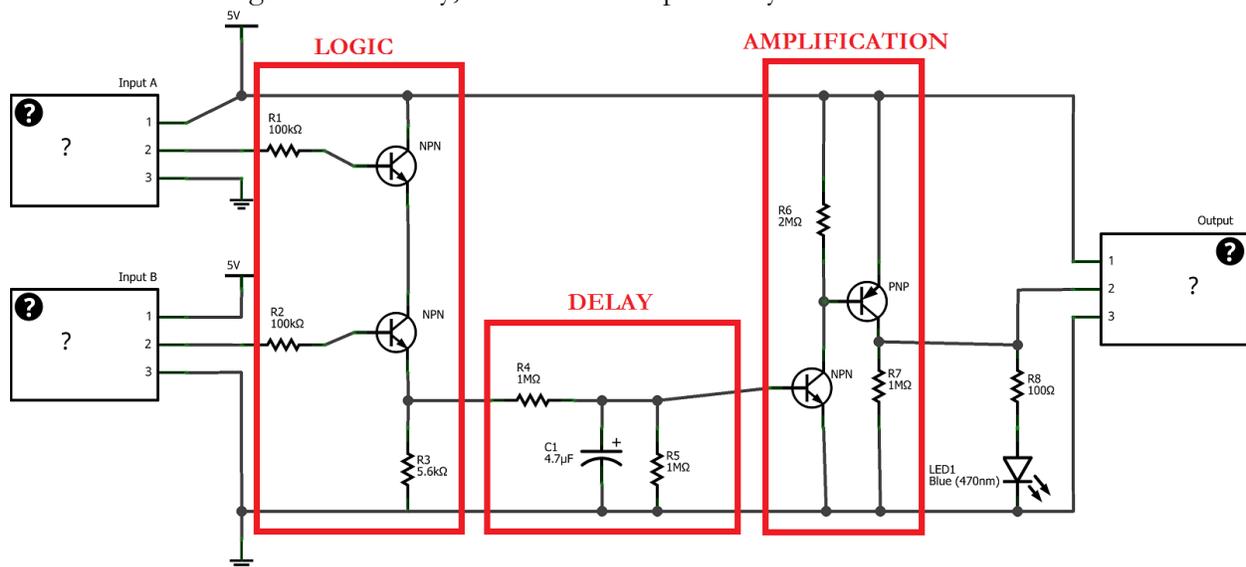


fritzing

Figure 1: Schmitt triggered delay with OpAmp

While this worked, I looked back at the plan I'd put down on paper and realized I'd need 50 OpAmps to make this work. It looked less and less cost effective the more I stared at it.

Instead, I forged ahead, choosing to ignore the Schmitt trigger problem in hopes that it would just go away. Starting with the AND gate, I used falstad's circuit simulation software^{viii} to test out my drawings. According to my simulations, I found that I should be able to use the following to construct an AND gate with a delay, and have its output always ~5V.



fritzing

Figure 2: AND gate, according to simulation

Another benefit to this setup was that the delay and amplification stages could remain the same between gates – the only place I needed to make changes was in the logic portion.

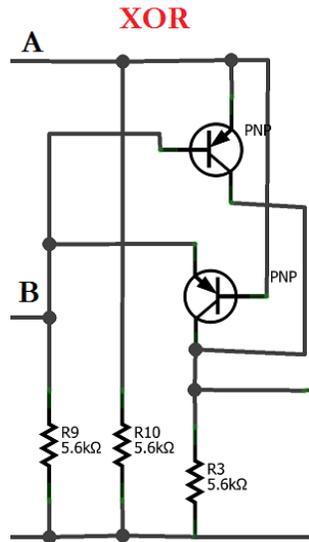


Figure 3: XOR Logic

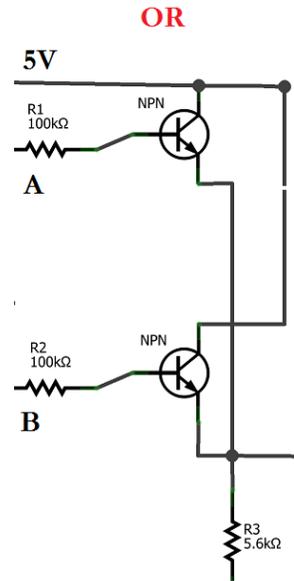


Figure 4: OR gate

Additionally, this architecture was (accidentally) beneficial because the transistors in the amplification stage have a natural activation threshold. While this isn't as tunable as the Schmitt triggers, it is considerably cheaper and more board-space effective. Additionally, I discovered that switching to FETs makes the transition between output states sharper.

From here, moving on to the half adder was pretty trivial. I used CD4093BCN chips, which have four NAND gates with Schmitt triggers built in. Unfortunately, these are not tunable, so I ran into a slight snag where placing the propagation delay before each NAND gate left the voltage too low to flip the state from low to high. This was solved by changing the resistor values surrounding the RC time delay such that it would no longer be a 1:1 voltage divider. However, this had the side effect of creating a slight imbalance between the charging and discharging propagation timing. One way I considered fixing this was to buffer the delay from the gate with a FET. However, the time difference was low enough I chose to ignore it instead. The final circuit is shown below.

The other key feature that I figured out in this stage and subsequently retconned into all the diagrams previously included in this report is how the individual blocks would connect to one another. It is important that the blocks be able to connect in an arbitrary order with an arbitrary quantity. In order to accomplish this, I decided to have a three-wire connector between blocks, one to carry +5V, one to carry GND, and one to carry the "data" signal. Thanks to our amplifier, the data signal should be approximately 5V when entering any arbitrary block. Additionally, the "registers" or original input signals (which I didn't implement) can be blocks with a SPDT switch to select between 0 and 5 volts.

One of my goals in this project was to not only create something that functioned, but something that people would want to have around the house. If Logic Blox looked pretty enough, people might actually put them on the wall and display them, creating art that people can learn from (which is awesome.) Thus, the next step was to try and create a wrapper for the electronics that would be visually interesting.

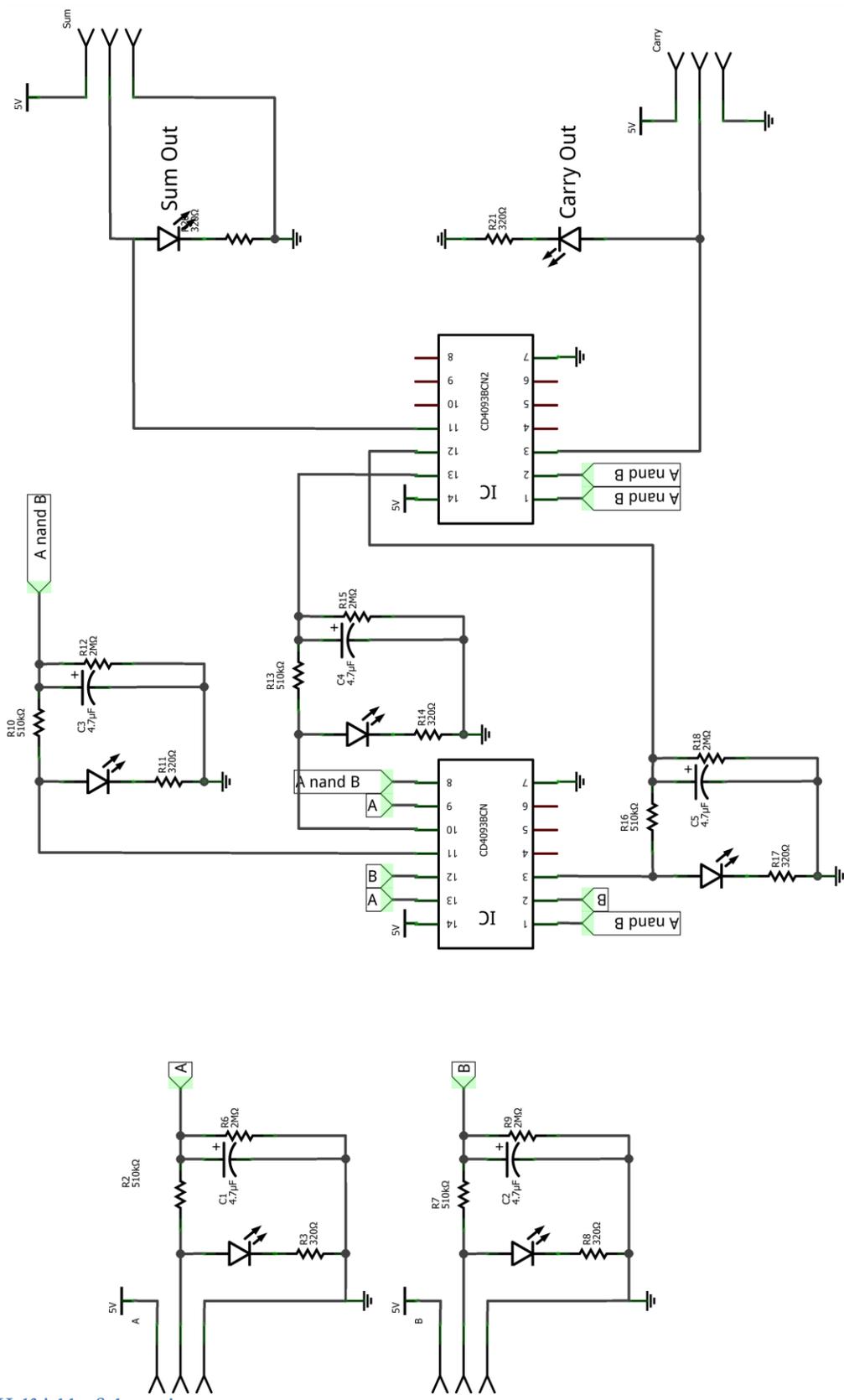


Figure 5: Half Adder Schematic

fritzing

Though I ran out of time to actually fabricate them, the holders would likely have looked something like the following CAD renders^{ix}. The holders are specifically designed to allow the user to see the circuitry involved and be able to manually trace the path of the data through the gates. They also attempt to convey the idea that the more complex structures (the half and full adders) use more gates, and would require more space within a computer.

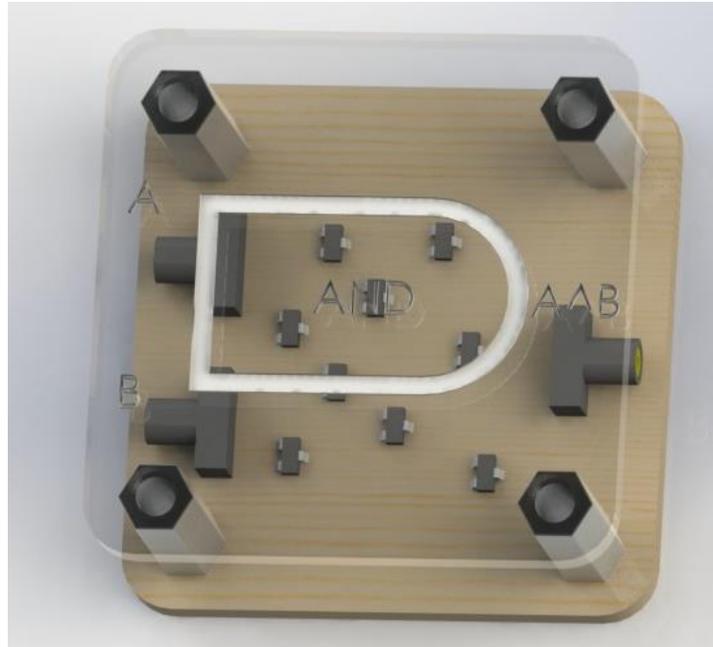


Figure 6: AND gate holder

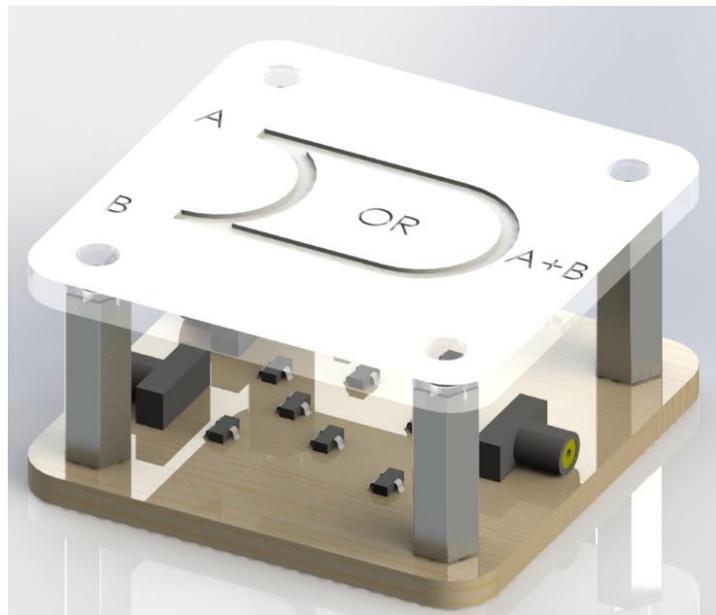


Figure 7: OR gate holder

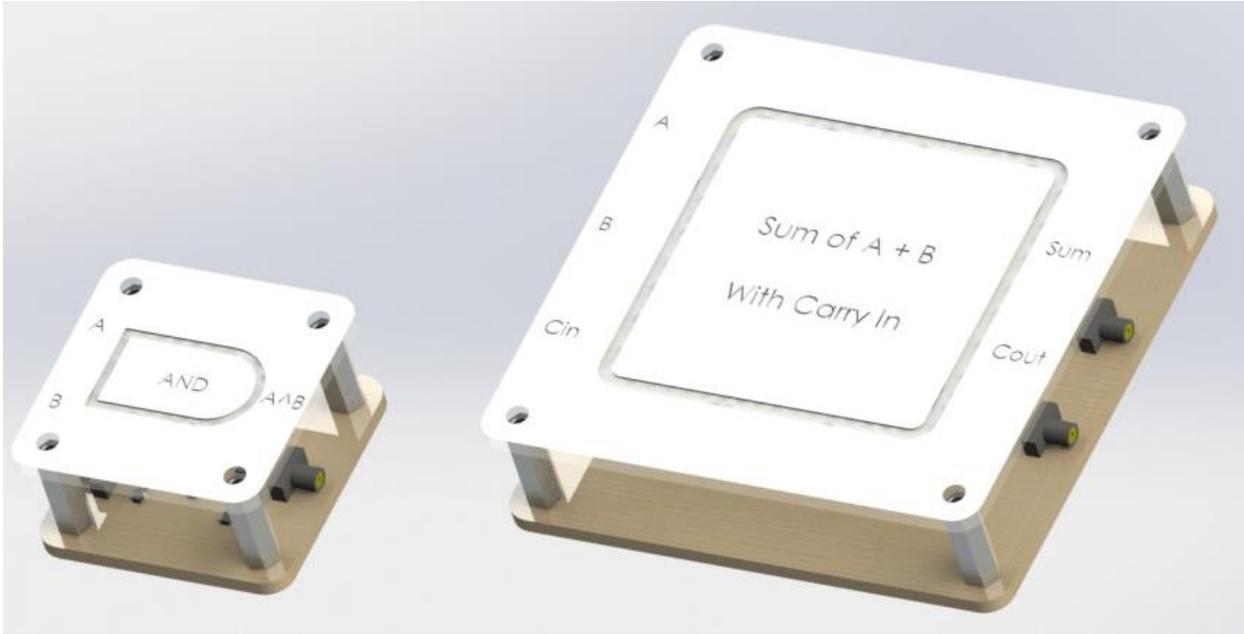
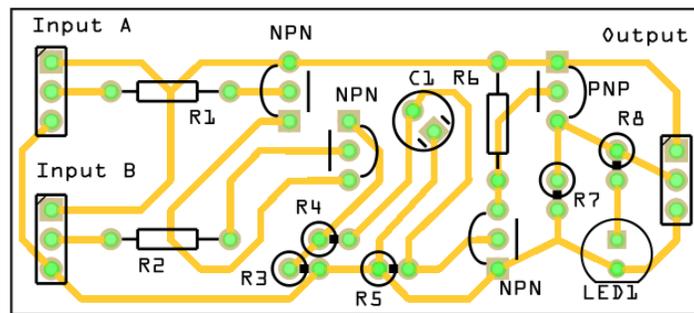


Figure 8: Holder size comparison

The last step I tried along the “make things pretty” track was actually putting the circuits onto custom-etched PCBs. I tried a different fabrication process^s than what is standard at Olin, with mixed results. That is to say, it didn’t work, but not because the process was flawed. I got impatient during the etching process and accidentally destroyed some of my traces with a paper towel. I will be trying this again sometime, and (if it’s successful) will update this document with an appendix detailing the actual process used. The AND gate’s PCB layout is included below, just for funzies. The others can be found in the accompanying folder.



fritzing

Figure 9: AND gate PCB Layout

How can someone else build on it?

Since the bulk of the difficulties and the design decisions were outlined above, I will simply add a few difficulties, work plan reflection, and some suggestions for future development.

Difficulties

The main difficulties previously addressed include:

- Expensive. Cut costs down.

The gates, as designed, are not cost effective on small scales. I'm not even sure if they're cost effective on large scales, but I know they're not at this scale. Ironically, I think printing the PCBs professionally^{xi} would actually *reduce* cost at this point.

- Cannot laser copper
This was a pipedream of mine – if the copper on the blank PCBs was thin enough, we could theoretically cut through it and quickly manufacture PCBs without the environmental hazards associated with ferric chloride. Turns out our lasers can't do this.
- Transistors and FETs are not the same.
Who knew? This caused serious delays when I was freely mixing and matching them without really paying attention to their characteristics.
- P-channel is (illogically) not NPN
Okay, seriously, this one's just annoying. The channel is in the middle of the FET, so it should be the middle letter. Again, transistors and FETs are not strictly interchangeable (though many parts of these circuits can use either, with very slight modification) and they don't make any sense. If someone with experience levels similar to my own (read: none) wants to take this project ahead, I would recommend just using the diagrams provided.

Work Plan Reflection

- GOAL: To create a semi-artistic, hands-on representation of various logic gates to teach people basic Boolean logic. This should be polished enough that it could be displayed in a home (That is, something people want to use and show off.)
- Stretch Goal 1: Include propagation delays in the representation.
- Stretch Goal 2: Show several higher-order components, with some way of visually seeing their comparative size to the smaller gates.

So...I incorporated both stretch goals, but missed the "Polished" bit a little. Everything is ready for fabrication (in theory) but I didn't have time to actually fabricate it. There are two main reasons for this – while I accurately predicted the amount of time it would take to accomplish each sub-task, I didn't plan on things like creating PCB layouts. While I normally would have just doubled down on the project and tried to complete it anyway, another class' final project ate up several days more than expected. Having done so much legwork to get this project to the point it is now, I fully intend to push it over the hump next semester. I want my wall art!

To Do

- Finish polishing the last few corners and actually fabricate the thing.
- Make a latch? That would be neat.
- Figure out what it would take to create a GIANT (binary?) wall clock from these gates. IT WOULD BE SO BRIGHT!
- Run a test with the acrylic layer and see if clear with frosted details (shown) or the inverse would look better. The first would allow you to clearly see the circuit, but the second might be better at diffusing the LEDs and making it possible to actually look at the propagation delay without blinding yourself.

End Notes

ⁱ With an x. That's important.

ⁱⁱ "Created" is a pretty strong term. I laid (almost) all of the foundations for Logic Blox. As is noted several times throughout this, I ran out of time for fabrication. However, in the spirit of Oliners building off the successes (and failures) of past Oliners, I've included every file I generated during this project so that someone in the future (myself or others) can take it from here.

ⁱⁱⁱ As it turns out, NAND (not-AND) has "functional completeness." This means that any other logic gate can be created by combining NAND gates. The Wikipedia page on how to build each gate exclusively out of NAND gates gives complete diagrams of NOT, AND, OR, NOR, XOR, XNOR. Check it out: http://en.wikipedia.org/wiki/NAND_logic

^{iv} After several conflicting answers on the "minimum" number of transistors needed to create an XOR gate, I stumbled upon this one (<http://www.lastrayofhope.com/tag/pnp/>) that used two. His blog post says that this version doesn't work, but it worked for me.

^v While it was a little annoying to find exactly the thing I was trying to make already available on the internet, it did (sort of) provide some validation that the idea was neat. Also, their kits only have AND and OR gates. Theirs snap together with magnets though. I wish I had magnets. Check them out! www.littlebits.cc

^{vi} A laudable goal that is AMAZINGLY difficult to do. Just this Tuesday (two days before the project deadline) someone else discovered Sparkfun's LogicBlocks kit (<https://www.sparkfun.com/products/11006>). Not only do these perform every lower level circuit element I was trying to include, they also have a propagation delay and pretty circuit boards to boot.

^{vii} Seriously, is there anything Wikipedia doesn't know? http://en.wikipedia.org/wiki/Schmitt_trigger

^{viii} This little java applet is amazing. You can download it and play with it all the time. I've only found one scenario when it differed from what actually happened on my breadboard. <http://www.falstad.com/circuit/>

^{ix} If it renders, it's manufacturable, right?

^x Hint: it involved lasers.

^{xi} OSH Park (www.oshpark.com) has what appear to be really reasonable prices. Also, their boards are purple. Hard to beat the cool factor of purple boards.