8-Bit Computer **George Sleen**

Overview

Design Goal:
Create from scratch[1] an 8-bit Turing complete computer.

Timeline:

May: Design decisions	Logic family Proof of concept logic gates What are the macro components I need?
June: Logisim macro components, Breadboard macro components	
July: Full Logisim computer, Breadboard macro components	
August: Complete design, Manufacturing	

Logic Families

	Pros	Cons
Resistor Transistor Logic (RTL)	Incredibly simple BJT transistors Inexpensive components Static electricity resistant	Power inefficient Lots of components needed Slow switching speed Susceptible to noise Bad fan out
Diode Transistor Logic (DTL)	BJT transistors Inexpensive components Static electricity resistant Handles noise well	Power inefficient More complicated than RTL Lots of components needed Slow switching speed
Transistor Transistor Logic (TTL)	Power efficient BJT transistors Inexpensive components Static electricity resistant	Ideally uses multiple emitter transistors More complicated than RTL Lots of components needed
Complementary Metal Oxide Semiconductor Logic (CMOS)	Incredibly power efficient Very fast switching speed	Static electricity susceptible Expensive components Complicated logic gates

[1] I will not be manufacturing resistors and transistors.

There are of course more logic families, these are simply the most common and have differences that will be relevant to the project. Due to the low clock speed of the final computer, I don't have to worry too much about transistor saturation[2].

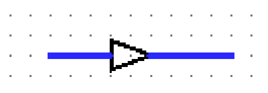
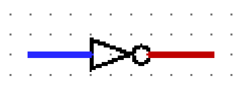
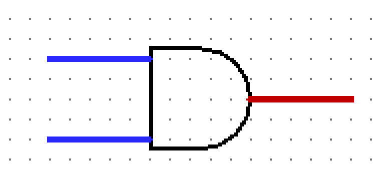
After screening[3] I cut DTL for the more complicated gate design and CMOS for the price and complexity again. Comparing TTL and RTL, RTL's simplicity won out for me. I do not have access to an oscilloscope currently and I want to minimize the amount of high-speed troubleshooting I need to do.

Digital Logic

Logic Gates

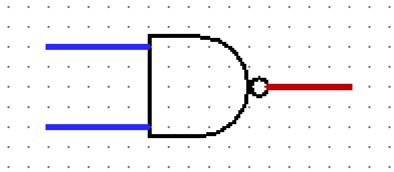
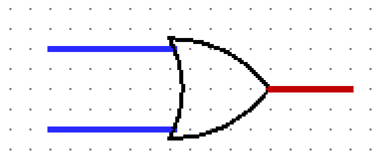
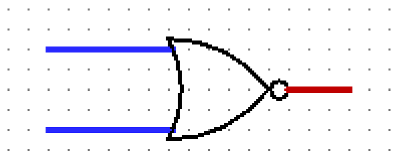
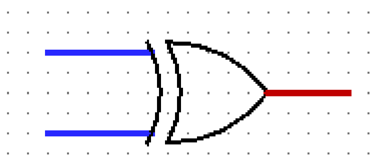
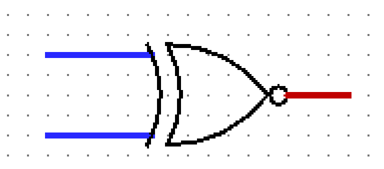
The fundamental building blocks of digital logic, these simply take in one or more input and give back one output.

Logic gates are commonly shown as block diagrams, which are summarized below:

<table>
<tr><th></th><th>Block Diagram</th><th>Truth Table</th></tr>
<tr><td>BUFFER</td><td></td><td><table><tr><th>IN</th><th>OUT</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>NOT</td><td></td><td><table><tr><th>IN</th><th>OUT</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>AND</td><td></td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
</table>

[2] If your project is going to run at a very high clock speed, or you care about transistor saturation you should look at Schottky TTL.

[3] Yes, we're using APSC 100-101 terminology.

<table>
<tr><td>NAND</td><td></td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>OR</td><td></td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>NOR</td><td></td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>XOR</td><td></td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
<tr><td>XNOR</td><td></td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
</table>

Boolean Algebra

Boolean algebra allows us to compute what will happen in a more complex circuit. All of the logic gates above can be represented through 4 mathematical operations:

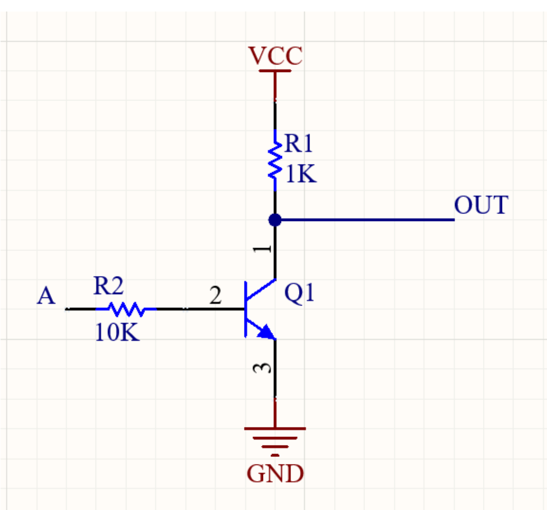
<table>
<tr><td>NOT</td><td>$\bar{A}$</td><td>Inverts the value of A</td><td><table><tr><th>IN</th><th>OUT</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td></tr>
</table>

<table>
<tr><td>OR</td><td>$A + B$</td><td>Returns 1 if either A or B is 1</td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>AND</td><td>AB</td><td>Returns 1 if both A and B are 1</td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td></tr>
<tr><td>XOR</td><td>$A \oplus \mathrm{B}$</td><td>Returns 1 if only A or B is 1</td><td><table><tr><th>A</th><th>B</th><th>OUT</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td></tr>
</table>

We can combine these operations to get any of the logic gates above.

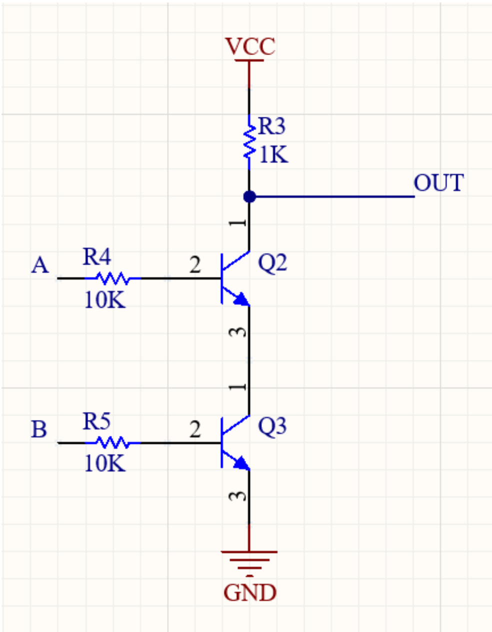
RTL Logic Gates

NOT



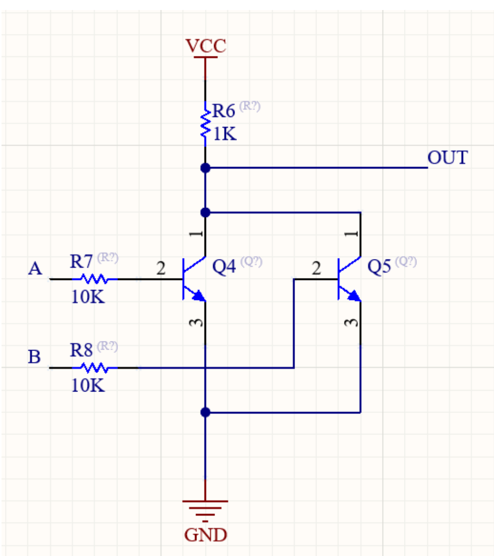
1 NPN Transistor
1 10K Resistor
1 1K Resistor

NAND



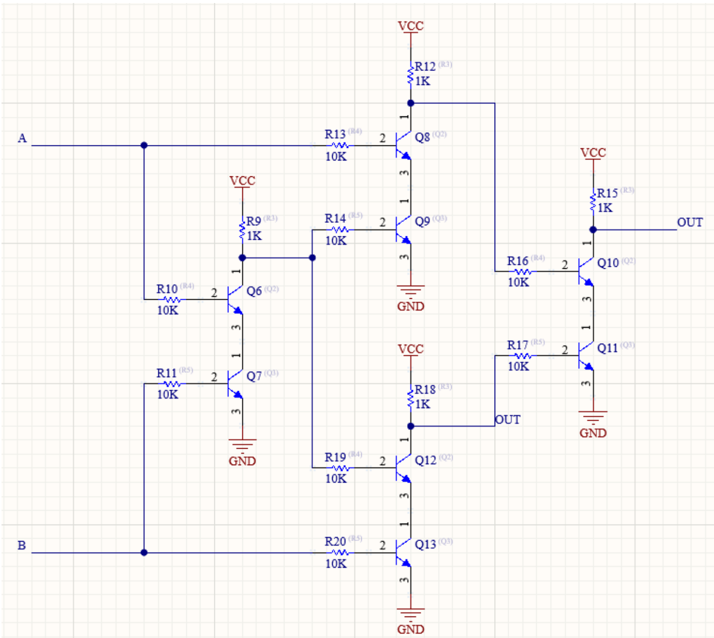
2 NPN Transistors
2 10K Resistors
1 1K Resistor

NOR



2 NPN Transistors
2 10K Resistors
1 1K Resistor

XOR (Find a more efficient circuit)



8 NPN Transistors
8 10K Resistors
4 1K Resistors

https://www.youtube.com/watch?v=nB6724G3b3E