Logic Gates

Kevin "Yama" Keyser[1, a)]
University of Missouri-Kansas City

In Laboratory 6, we cover logic gates and Boolean algebra. To accomplish this, we use NAND and NOR gates with LEDs to show when values are true (high) or false (low).

I. BACKGROUND

For this Lab, we need to know the basics of how Boolean Algebra works. You start off with 3 operators, (AND represented by $\wedge$, OR represented by $\vee$, and NOT represented by $\neg$), and using truth tables to figure out what the output would be.

The is a binary operator (takes two inputs) and only puts out a true (high) value if both inputs are true (high). The truth table for an AND gate looks as follows

AND gate Truth Table		
A	B	A$\wedge$B
0	0	0
0	1	0
1	0	0
1	1	1

The truth table takes all possible values of A and B, and lists them out, and what the corresponding output would be. Whether you start off your values as false or true depends on your general outlook in life. Most Computer Science classes will have you write the false values first (pessimists as they can be), and Philosophical Logic classes will have you write the truth values first (blind optimists as they are). Since each variable can have two separate values (0 and 1) in Boolean Algebra, you will have 2^n rows or possibilities.

Next, the $\vee$ gate is a binary operator that only puts out a false (low) value if both inputs are false (low). The truth table for the OR gate looks as follows

OR gate Truth Table		
A	B	A$\vee$B
0	0	0
0	1	1
1	0	1
1	1	1

Finally the $\neg$ gate. This is a unary operator (one input) and gives the inverse of what your input was. As from Laboratory 5, you should notice that its symbol is similar to the Op Amps that we used for our experiments. An Op Amp can be used as a NOT gate if you put your input through the inverting pin, have your voltage supply for VCC+, but ground both the non-inverting, and VCC- pins. The truth table for the NOT gate looks as follows

NOT gate Truth Table	
A	$\neg$A
0	1
1	0

This is everything we need to be able to analyze any equation that uses Boolean algebra, which is what our circuits that we are testing today use.

II. PROCEDURE

The procedures for Laboratory 6 is very straight forward, so the Laboratory 6 sheet will be attached at the end of this write up.

III. PRESENTATION OF DATA

A. 6-1: Logic Levels

Voltage Cutoff Threshold for 7400	
Bound	Voltage (V)
Lower Bound	0.650
Upper Bound	1.21

Frequency Cutoff for Stability	
Wave Type	Frequency (MHz)
Triangle	2.3
Squre	7.7
Sine	3.4

[a)] Electronic mail: kk8r8@mail.umck.edu

FIG. 1. Instability point for Triangle Wave.

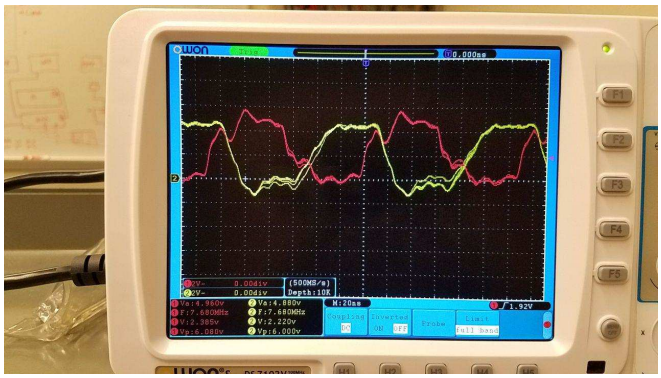

FIG. 2. Instability point for Square Wave.

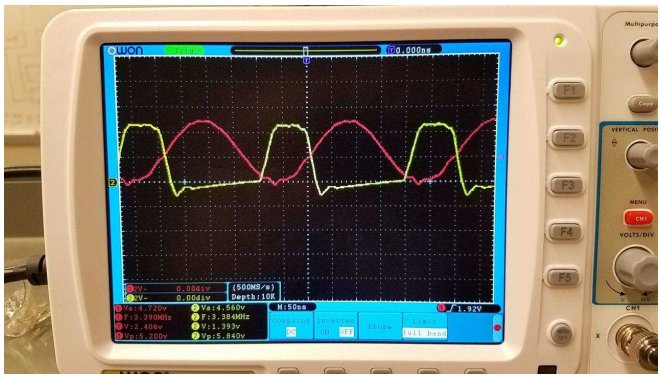

FIG. 3. Instability point for Sine Wave.

B. 6-2: NAND gates

NAND gate Truth Table		
A	B	¬(A∧B)
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate Truth Table		
A	B	¬(A∨B)
0	0	1
0	1	0
1	0	0
1	1	0

4 input NAND Gate				
A	B	C	D	¬(A∧B∧C∧D)
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

IV. DISCUSSION

A. 6-1: Logic Levels

For the first part of this experiment, we used trial and error to figure out the threshold for when the NAND gate went from a 1 (high) to a 0 (low). What we found was that any lower than 0.650V, and your input counted as a 0. Any higher than 1.21V and your input counted as a 1. This in-between area, inside the high and low threshold gave us inaccurate readings that fluctuated repeatedly. This may be due to tolerances, sensitivity, and the design of the gate to mainly have pulse or square waves as its input.

The frequency dependency section, we raised the frequency of our function generator until the output signal no longer looked like a square wave. Looking at our table, you can see that the Square Wave had the most stability (up to 7.7 MHz) where the Triangle had the lowest (2.3 MHz), with the Sine Wave in between (3.4 MHz). When you look at the pictures, and consider how much time is spent in the the state of both being low, and that our window of stability from our previous experiment showed the voltage had to be below 0.650V, then the more time the input signal spends at the low state, the more stable our output signal will be. The square wave spends half of its time in the high state, and half of the time in the low state. Less so for the Sine Wave, and even less for the

Triangle Wave. This means that the gate has less time to respond to changes, and the changes are more gradual, creating these inconsistencies and instabilities in our measurements.

B. 6-2: NAND gates

What we get with our circuits with the LED is that it matches our truth table as we would expect. For the 2 input NAND and NOR gates, notice how they are just the opposite of the truth tables we had in the BACKGROUND section of our lab writeup.

For the 4 input NAND gate (or even a 4 input AND gate) we could easily break down each into their own 2 input tables (A∧B, C∧D), and then combine their output into another 2 input gate. This is essentially what is happening on the inside (with oversimplifications due to higher level abstractions). You could also use De Morgan's Law, but that's outside the scope of this Lab.

V. CONCLUSION

This Lab was a very straight forward one, that will be expanded upon in the next Lab. The biggest takeaway was the use of truth tables to solve Boolean Algebra problems. There isn't an easier way to solve these as far as we know either (the 3-SAT problem has been labeled an NP-complete problem in the field of Algorithm Analysis), so the use of truth tables is the best we can do at this moment.