Digital Clock Project

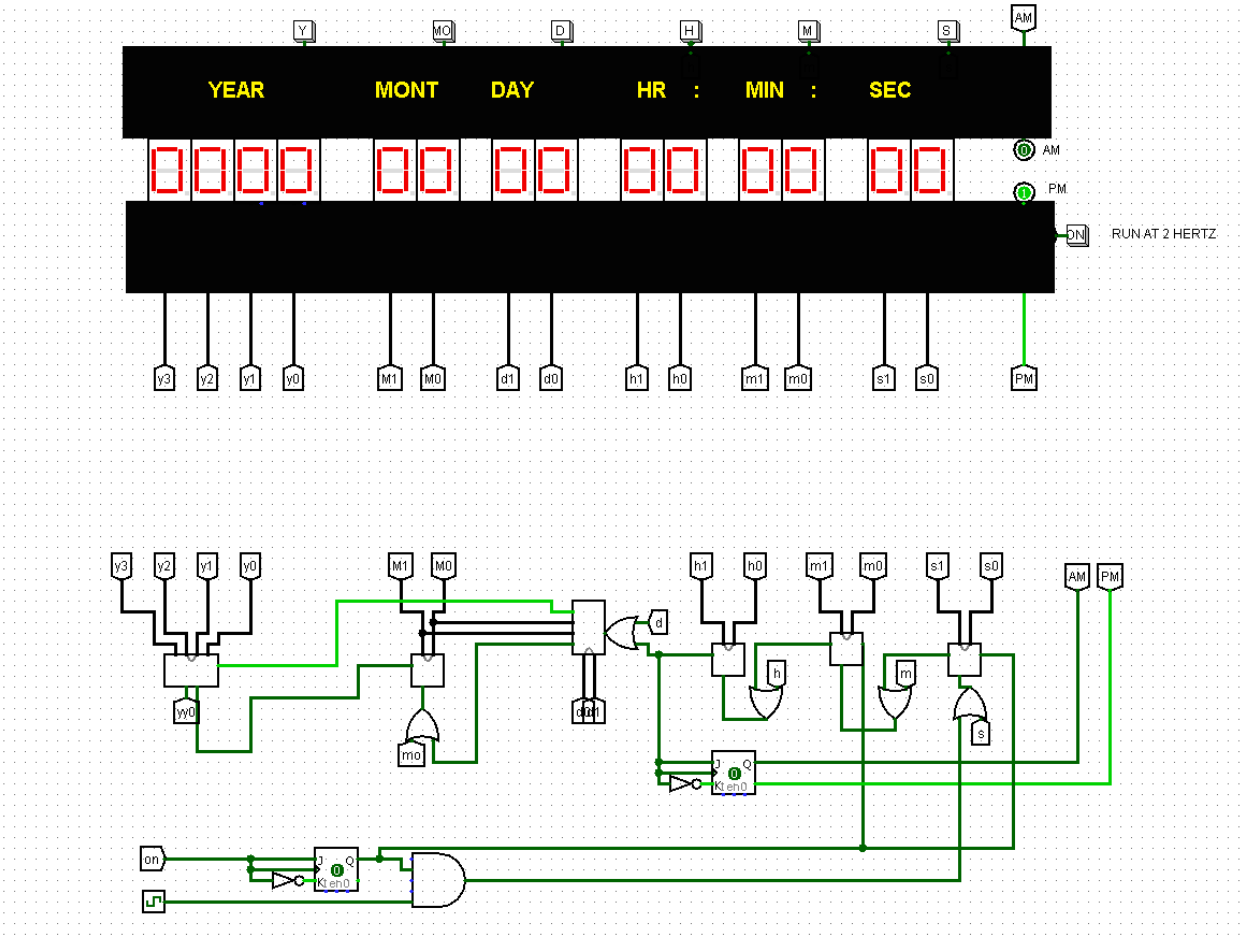
Gabriel Kim-Perez
CDA3103
Jahani

Outline:

I. **Introduction**

II. **Instructions**

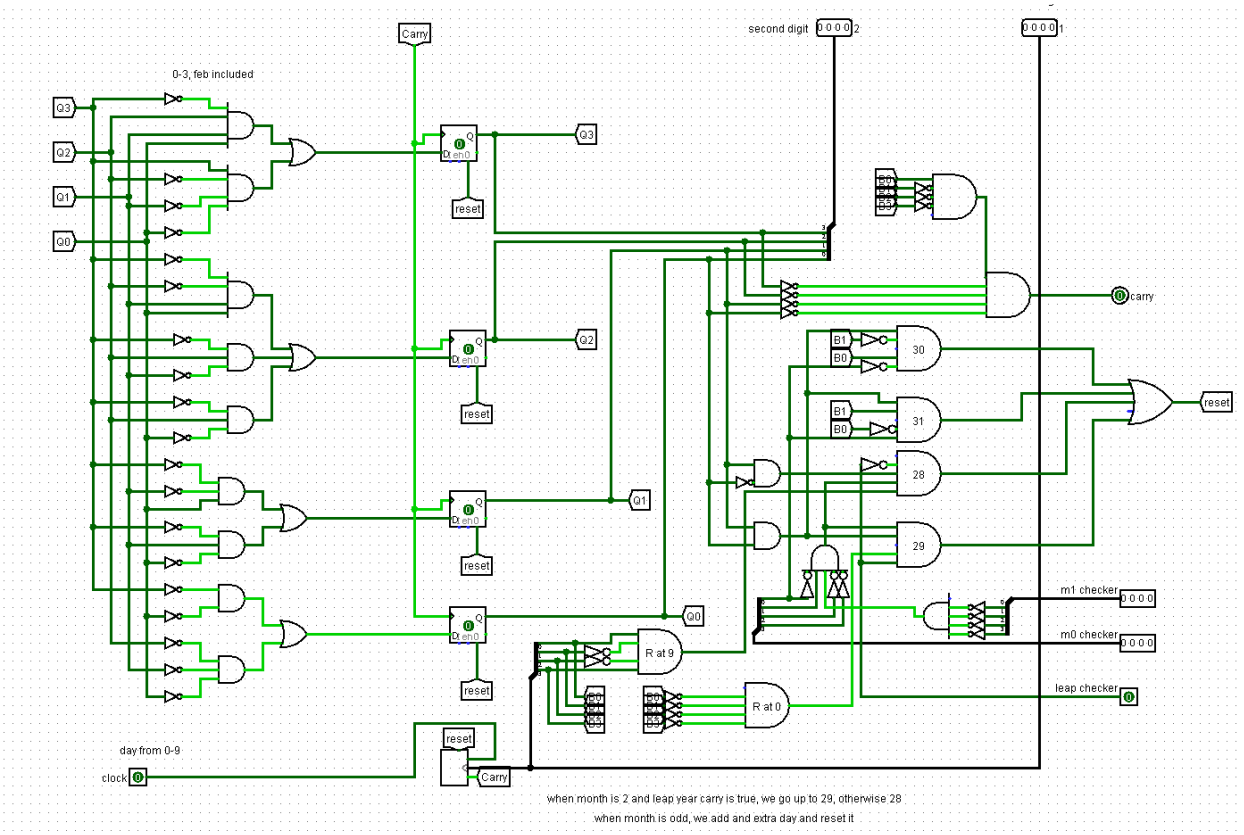
III. **Design and Components**



Introduction

Many hours were put into the different implementations, live demonstrations, failures, and ultimately successes that were achieved throughout the course of this project. There will be an extensive overview of each subcircuit and design I decided to use in order to make this clock work as seamless as possible. This clock features

the abilities of any standard digital clock: a second, minute, hour, day, month, and year counter, with AM/PM differentiation, **an ON and OFF switch**, **a leap year tracker, of which adds 29 days into February and cycles every four years**, and **adjustable buttons** to set the time appropriately,. Overall, everything works in a robust way with little bumps in implementation

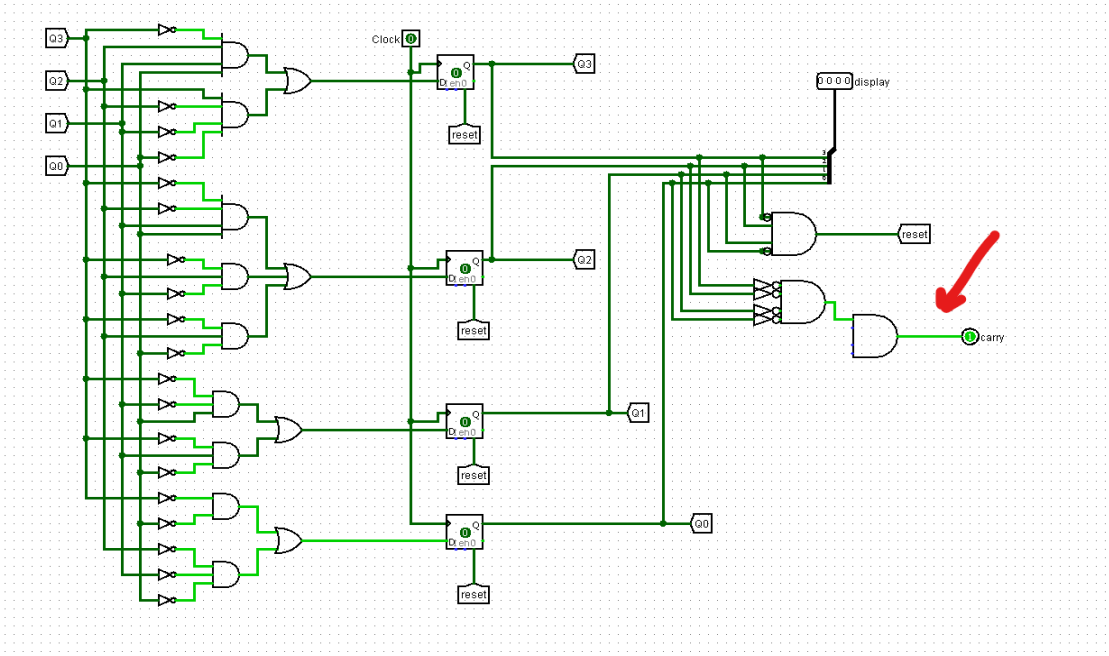


Instructions

As with all digital clocks, the setup is pretty much accessible from the get go. I recommend that in order to perfectly simulate real world measurements of time, you should clock the simulation at 2 hertz. Next, you're going to want to look at the top bar and set the time and date to your liking. This must be done before you press the ON button. Once the simulation is running and still at zero, press the ON button on the side of it to begin tracking time.

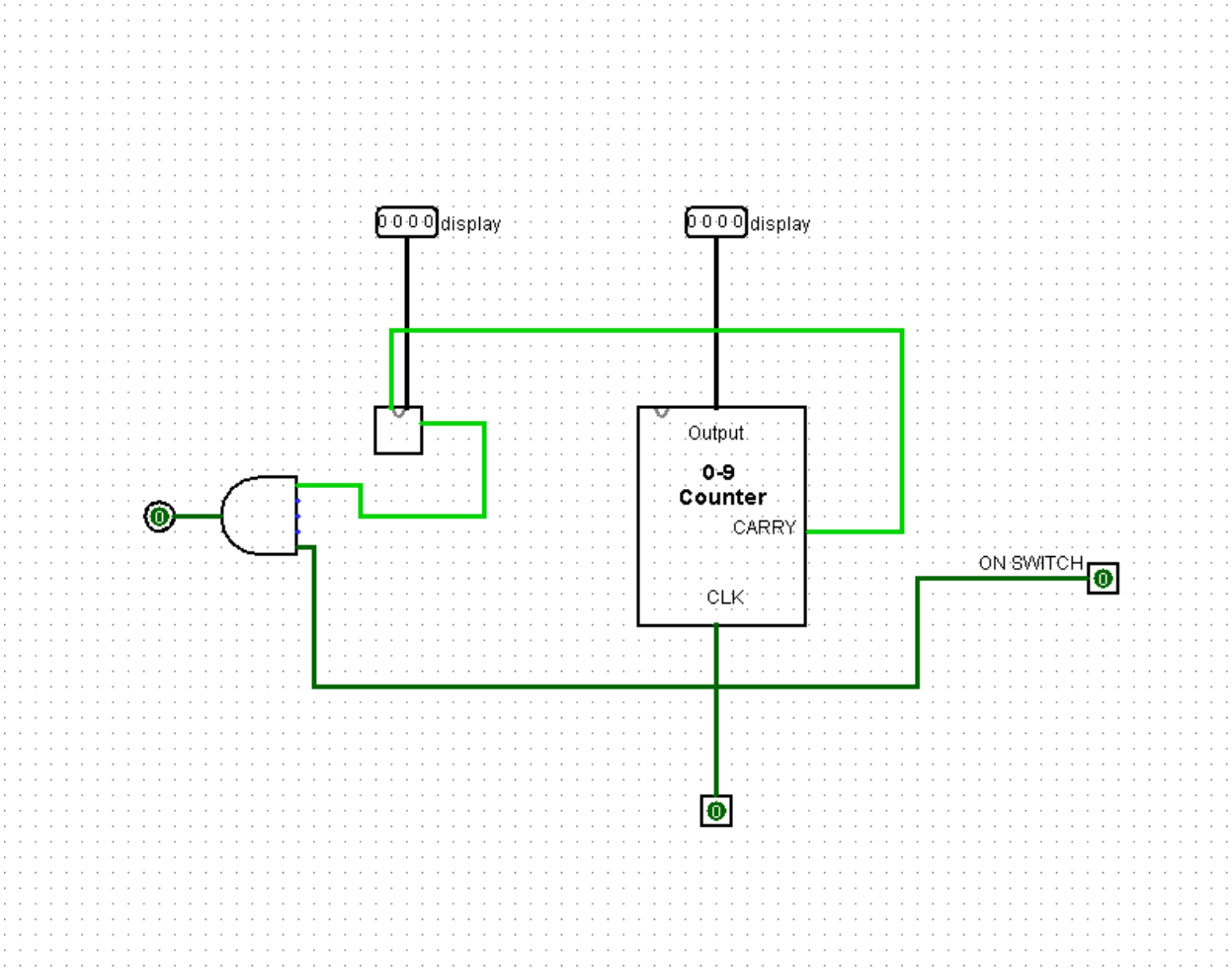
Design Plan

When looking at how to tackle this project, there were some things I had to adjust first and foremost in order to be as modular as possible with my implementation. Firstly, I integrated a carry feature in the 0-5 counter subcircuit that was provided by default. That way, I could not only use the 0-9 subcircuit to carry the signal to the 0-5 circuit, but I could adequately "carry" the signal to the next section of time, as seen in the second and minute function with the 5-9 counter subcircuit.

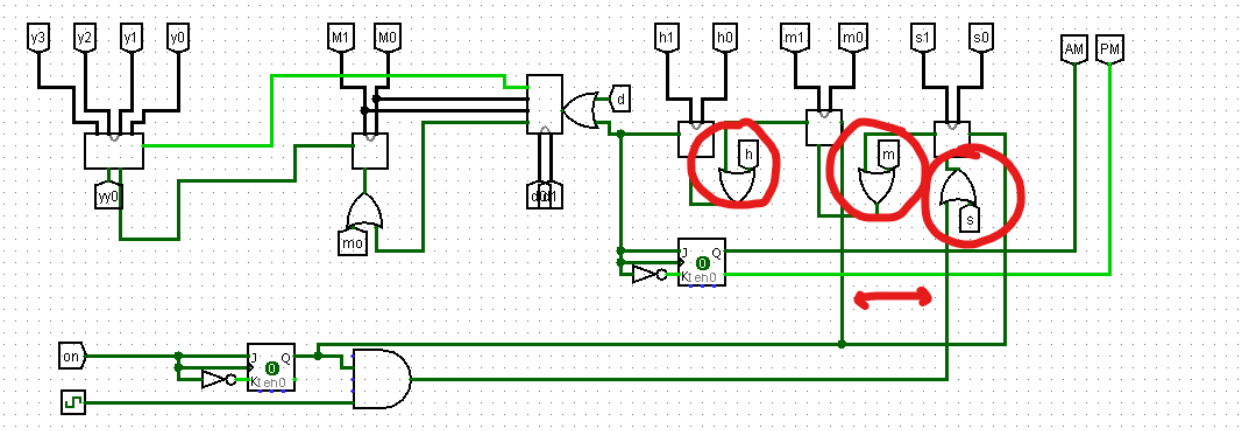


Attaching additional logic gates to the 4 bit splitter to effectively carry a signal appropriately.

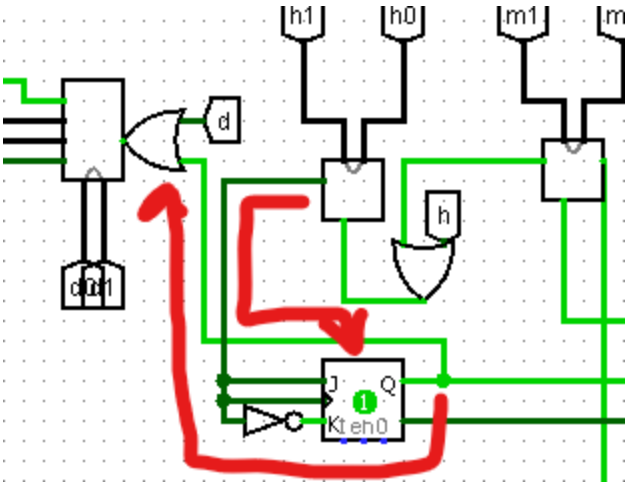
Within the latter half of this project, I noticed that within state 0 of the simulation once reset, all 5-9 counters were showing carry signals by default. The reason this was initially a setback was mainly because of the adjustable clock feature I implemented. I had to add an additional input (the ON Switch), as well as an AND gate at the output, to effectively hinder any signal carrying before any simulations began. That way, the OR gates implemented in the main circuit would lend signal to the adjustable buttons on the top.

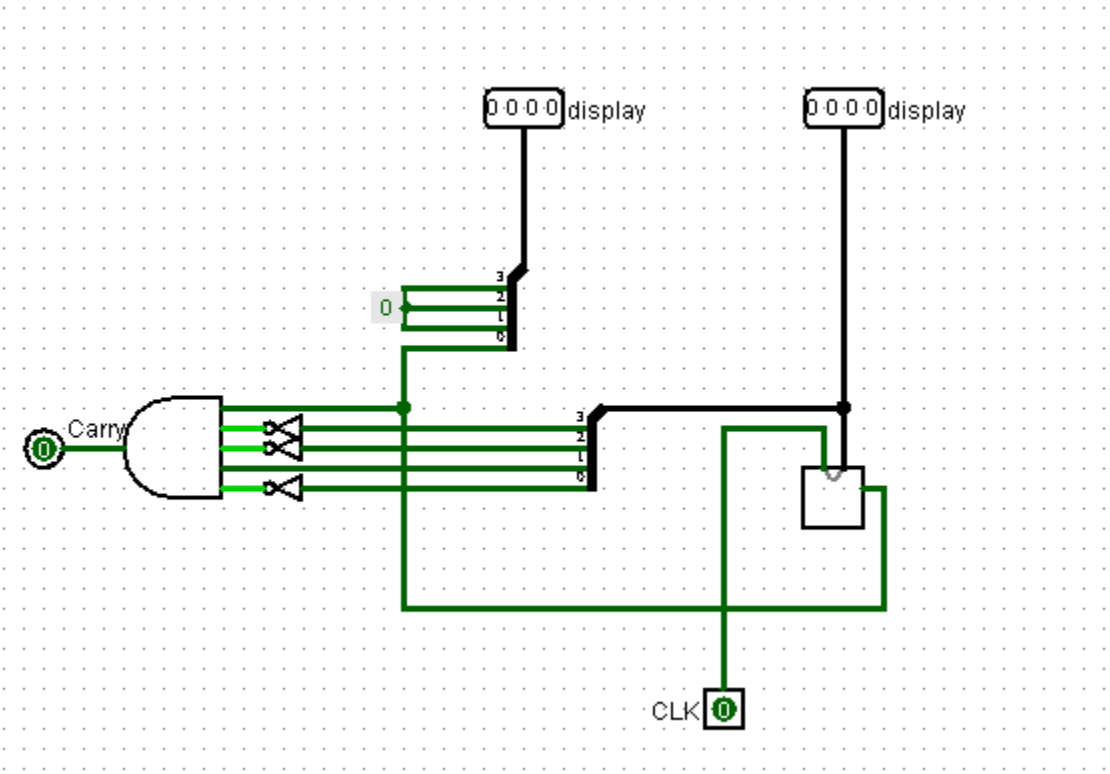


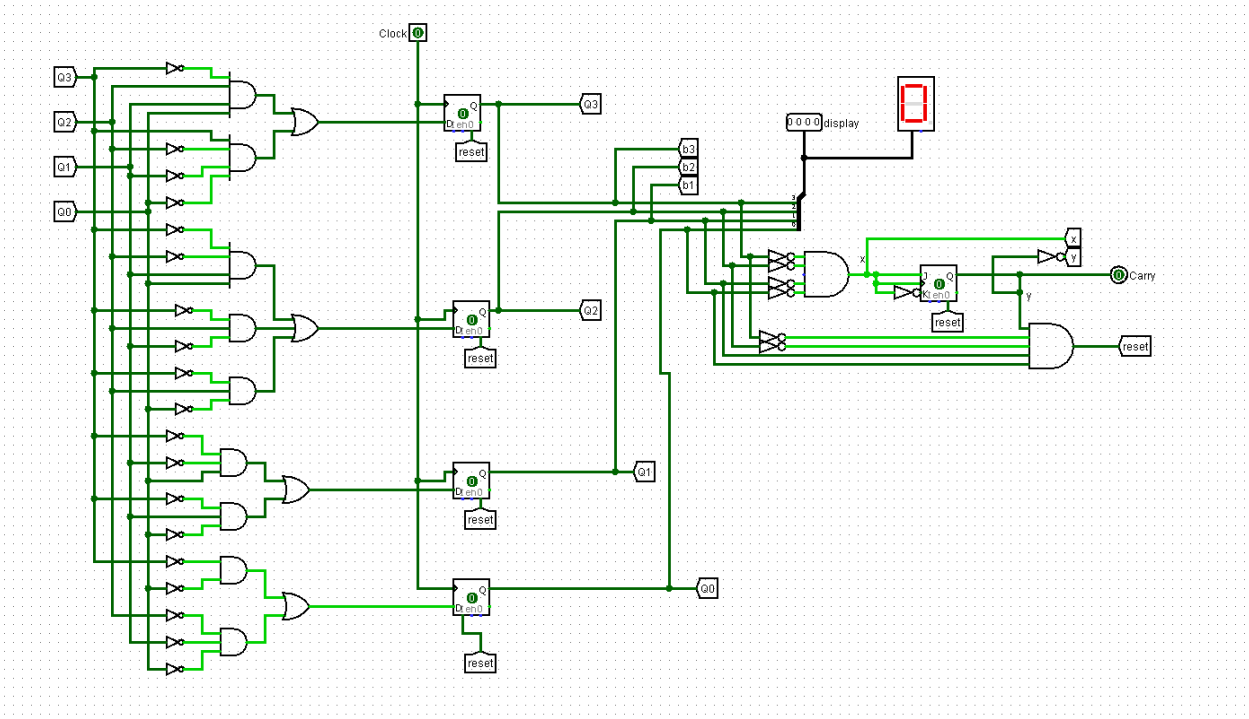
I also had to implement a JK flip flop in order to keep the signal on and allow them to carry



After the implementation of the seconds and hour, I had to modify circuits more heavily in order to tackle the hour switch. One ultimatum I was faced with designing the hour switch was to make it a 0-23 hour counter (as per international and military time standard) or the standard 12/12 AM/PM. I ultimately decided to go with the AM/PM approach, because not only would the counter circuit help me integrate the MONTH function seamlessly, it would also serve as a switch to change the DAY counter by having AM be the carry signal. The hour carry signal had to be fed into another JK flip flop, to balance the states of AM/PM in alternating fashion.

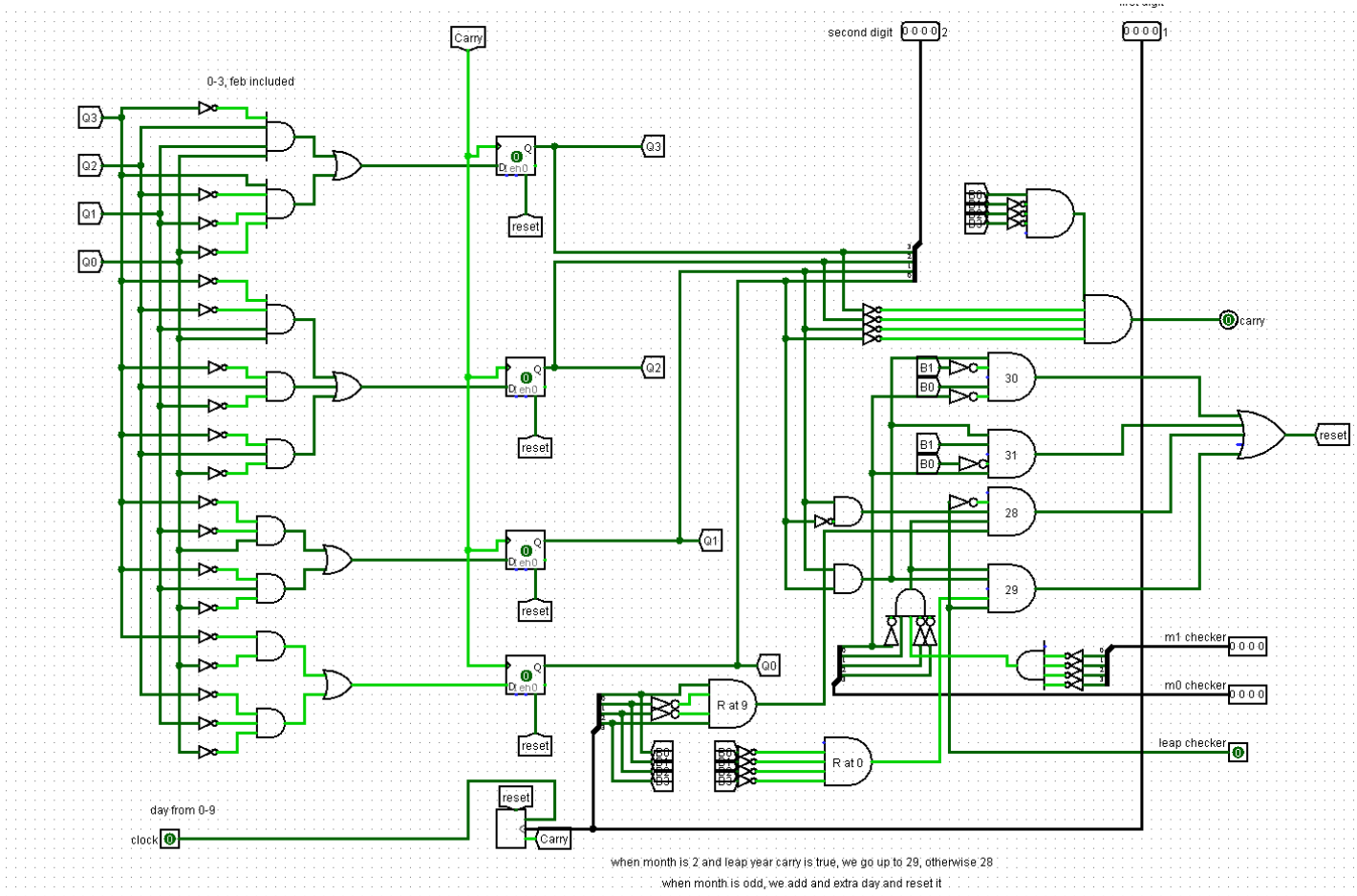

h1
h0
m1
m
d
h
J
Q
Kt eh0


0 0 0 0 display
0 0 0 0 display
3
2
1
0
0
Carry
3
2
1
0
CLK

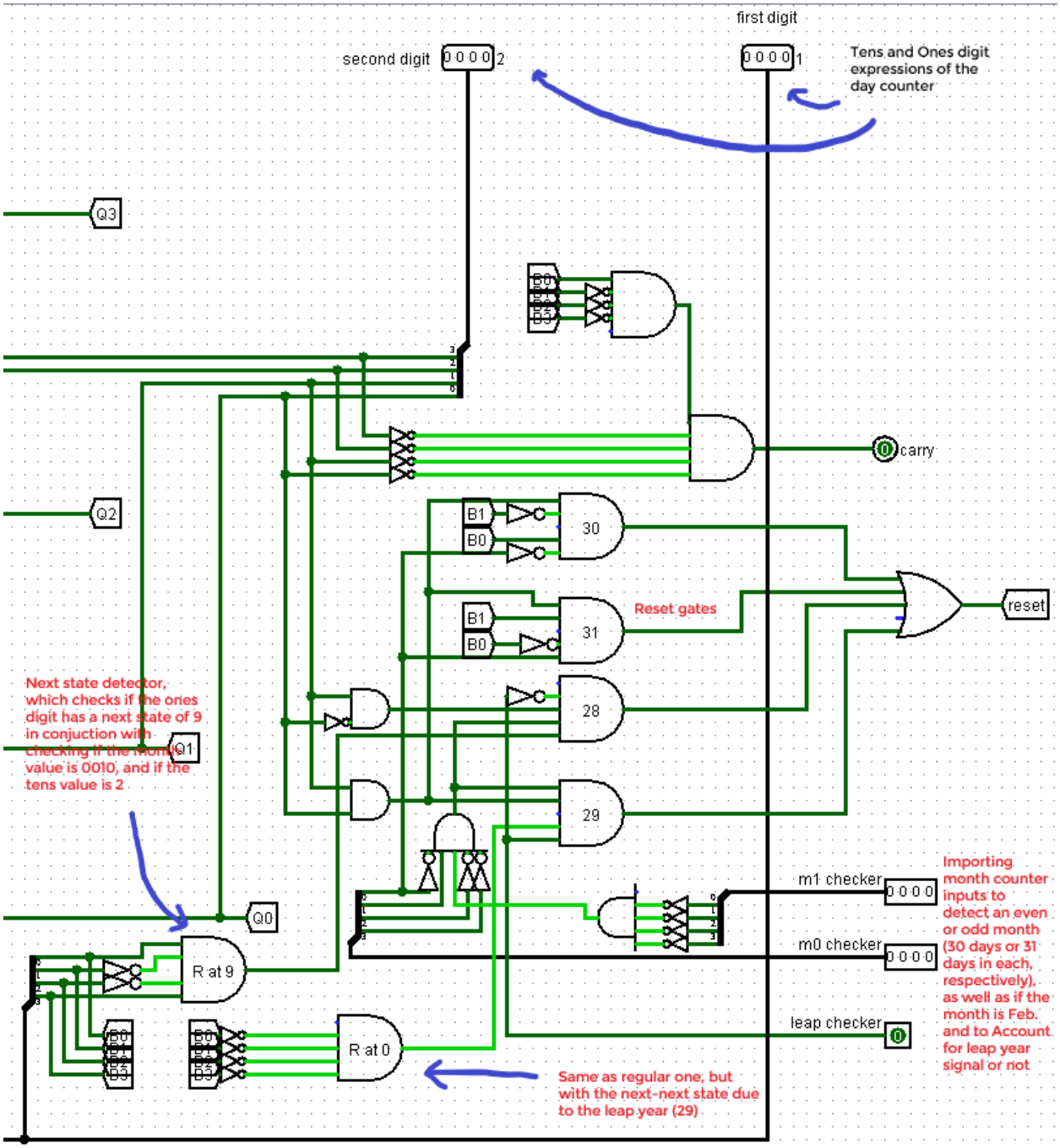


Next up was the Day counter. Possibly my most complex (and proudest) design, that has an integration that detects leap years and adds an extra 1 day to the 28 days of February. In addition, it also accurately shows the number of days for

every month:

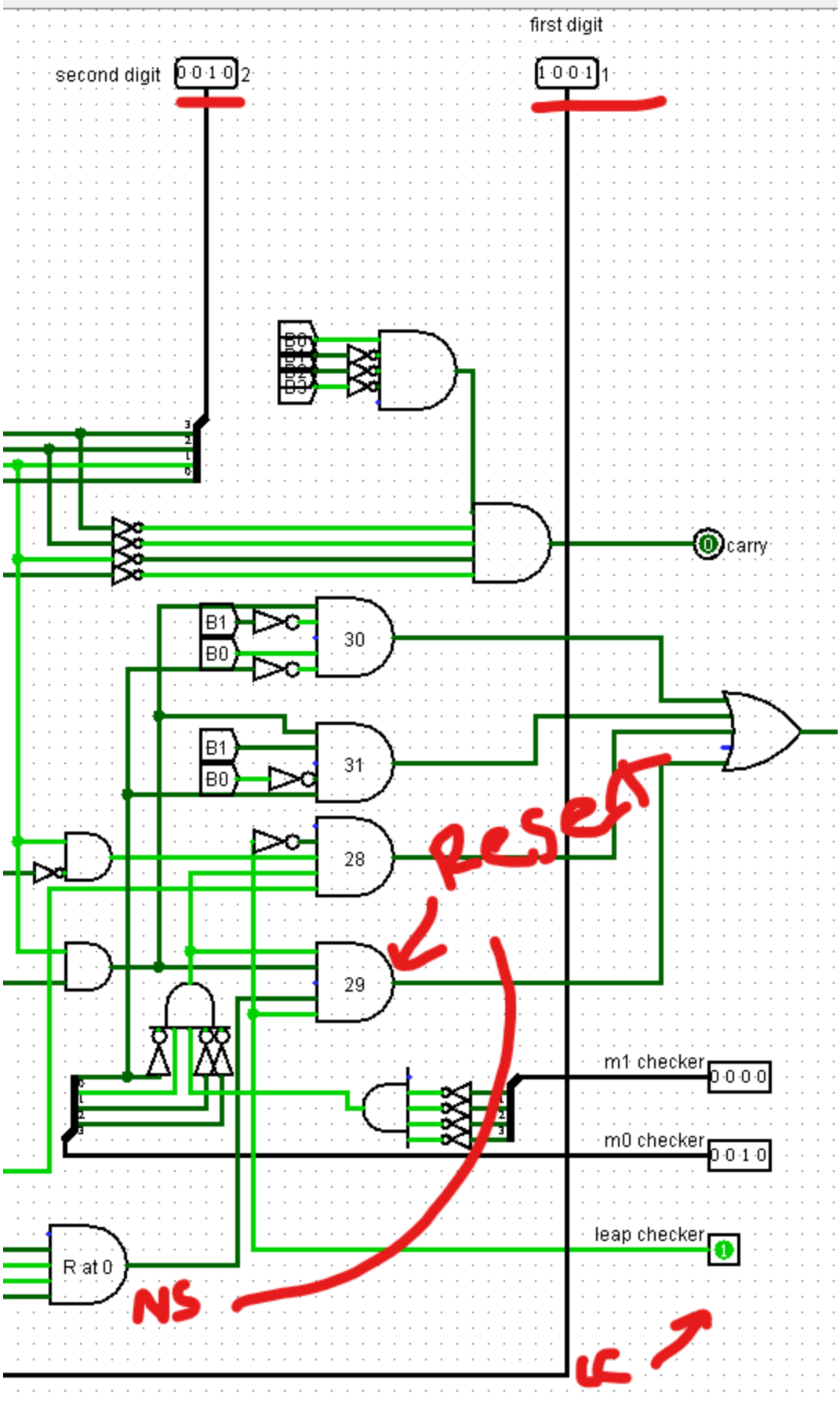


Let's break down the implementation. First we needed the standard 1-3 + 0-9 counter implementation. This would have been pretty easy to integrate EXCEPT for February. February not only has <30 days, but every four years, it switches from 28 to 29 days as per leap year. My first step to integrate this was to effectively branch off the 1-3 + 0-9 counters and create a different gate group.



when month is 2 and leap year carry is true, we go up to 29, otherwise 28

when month is odd, we add and extra day and reset it


first digit
second digit
0·0·1·0 2
1·0·0·1 1
B0
B5
B3
3
2
1
0
carry
B1
B0
30
B1
B0
31
28
29
RESET
m1 checker
0·0·0·0
m0 checker
0·0·1·0
leap checker
1
R at 0
NS
IF

The Gate group truth expressions are as follows:

Assume that m0 checker has 4 bits, m00, m01, m02, m03, and m1 checker has 4 bits as well, m10, m11, m12, m13.

Now, there are the 0-9 counters bits, which we will say are b0, b1, b2, b3.
There are the 0-2 counter bits, which are denoted as c0,c1,c2,c3.

Leap checker is L.

Gate of next state will be denoted as R (meaning the ones digit of the day counter)

For the 28 counter (no leap year), the truth expression is:

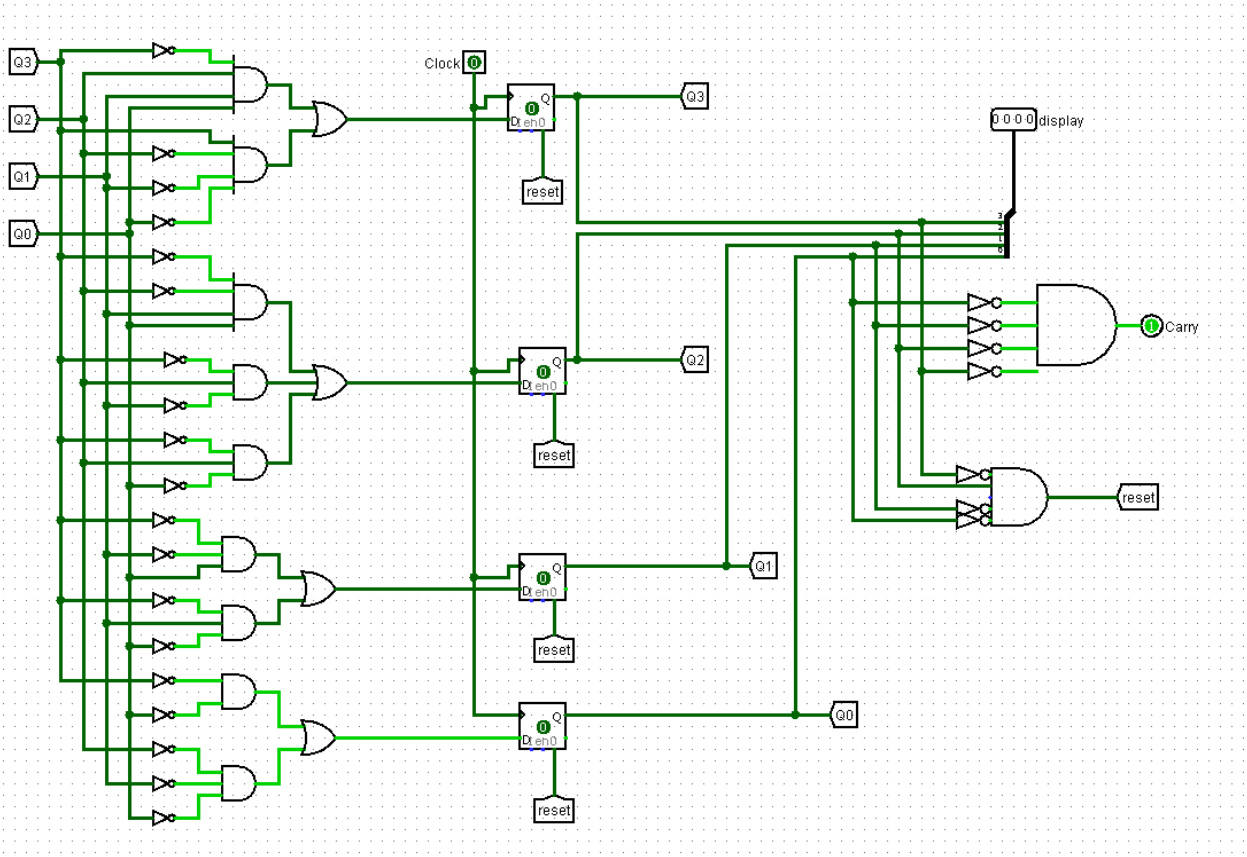
(L')(c1 c0')[(m13' m12' m11' m10') (m03' m02' m01 m00')] (b3 b2' b1' b0)

For the 29 counter (with leap year), the truth expression is:

(L) *(c1 c0')[(m13' m12' m11' m10') (m03' m02' m01 m00')] (b3' b2' b1' b0')*

Note how the last expression for each of these contain the NEXT state for triggering the reset.

The leap counter below just resets every 4 iterations for bits b0-b3 *(b3'b2b1'b0') before the reset gate.*



Finally, what was left of the day counter was to simply detect if the month ones counter was odd or not. You could easily just use m00 as a sentinel, because every other bit combination would be even.

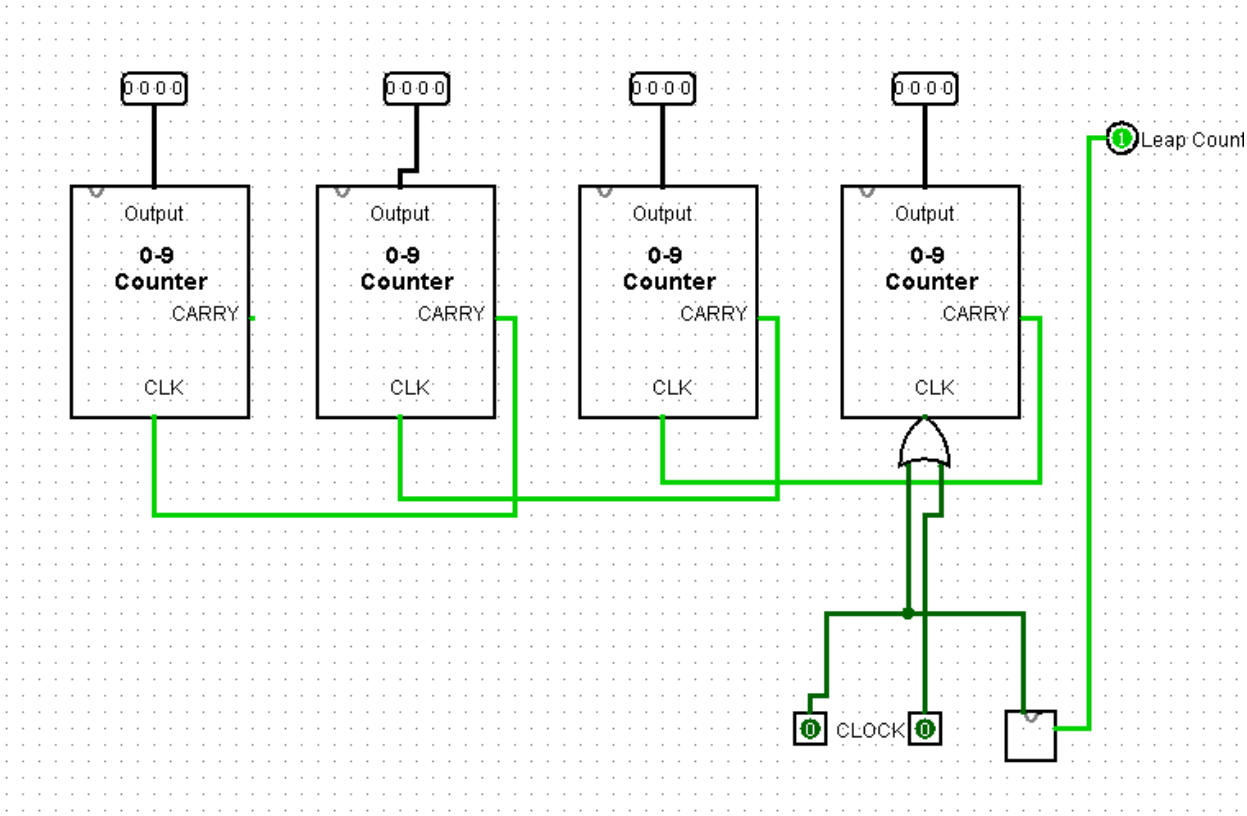
For odd months:

(Q1 Q0 B1 B0')

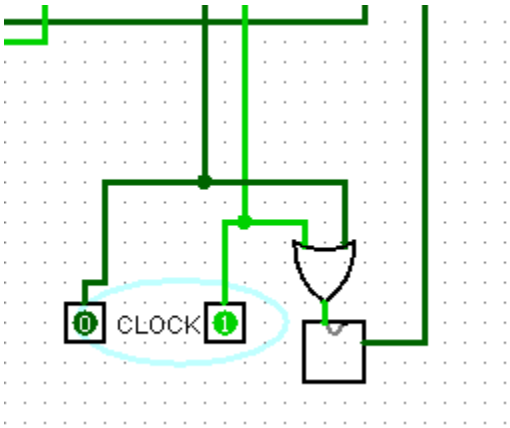
For even months:

(B1' B0 m00' Q1Q0)

For the month counter, I simply reused my 1-12 counter, but added extra outputs so the day counter can monitor the change any month is february or odd/even. For my year counter, I sequentially used 4, 0-9 counters, and attached the leap counter carrier to it.

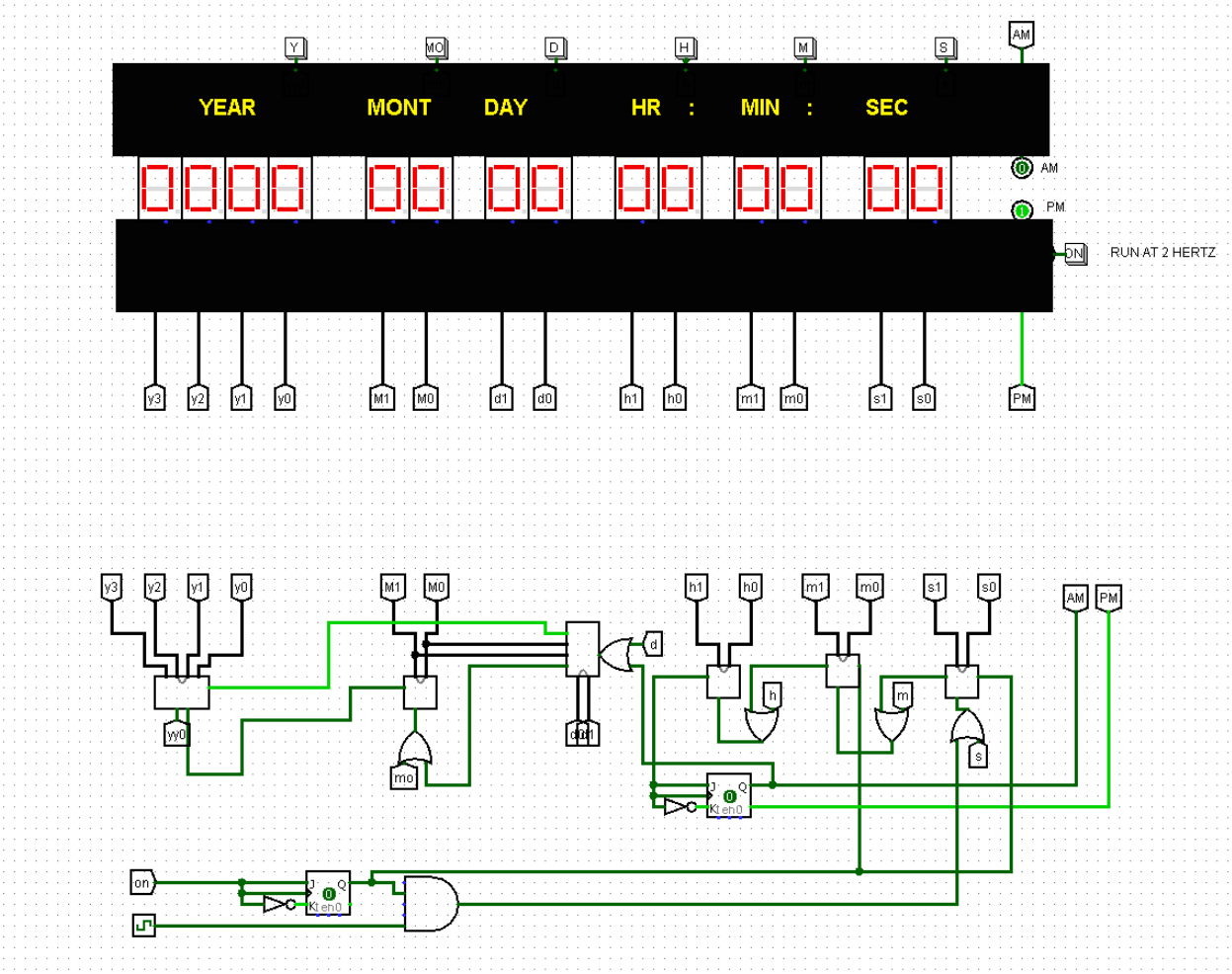


You'll notice that the clock in this image had to be altered in the next:



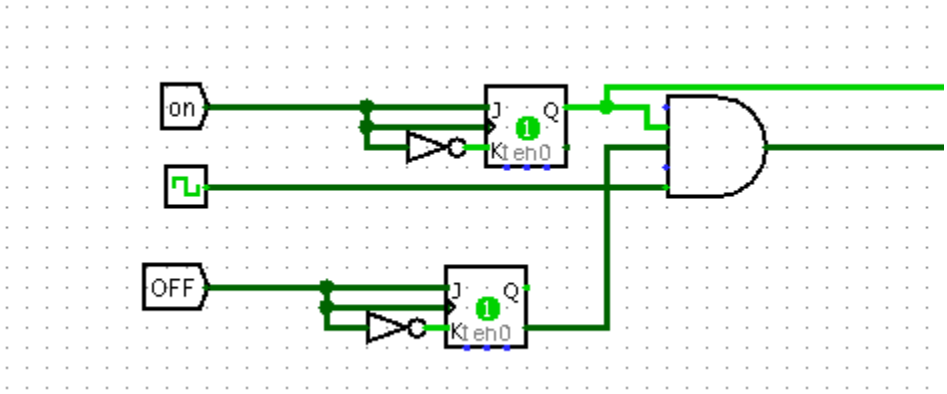
I had to take into account the leap counter modulating for every signal, not just from the clock or the adjustable setter, but both.

The Clock:



If I wanted to implement an adjustable timer, I had to rely on something besides the clock signal, due to having to freeze the time making it easier. Keep in mind, once you set the time, you'd have to reset the clock by resetting the simulation to do so again. The JK flip flop effectively lets the ON switch be pressed once to start the

clock, so it cannot be stopped unless the ON signal is off. One possible solution to this could be to implement an off switch, which simply hinders the AND gate.



The OFF switch successfully pauses the clock, but doesn't necessarily allow the user to set the time unless all carry signals are cut. Finally, you may notice at the beginning of every simulation the clock defaults all fields at 0. However, once the clock runs, they will skip over any 0s in carrying any hour, day, month, etc, so they work as intended.

Overall, I really enjoyed this project. It solidified my understanding of sequential and combinational logic with practical uses. I would very much look deeper and sharpen the features of my clock project from here on out.