LAB REPORT 1

1. The CLK input is not directly connected to the MINUTES\_COUNTER or HOURS\_COUNTER to ensure proper time synchronization. Instead, the seconds counter is driven by the clock, and only when it overflows (after 59 seconds) does it signal the minutes counter to increment. Similarly, the minutes counter only increments the hours counter after 59 minutes.
2. The setup logic works by controlling whether the clock operates in normal or manual time-setting mode. When SETUP\_TIME is low (0), the clock runs normally, with the CLK advancing the seconds, minutes, and hours. When SETUP\_TIME is high (1), the clock freezes, preventing the SECONDS\_COUNTER from advancing. In this mode, the MINUTES\_COUNTER and HOURS\_COUNTER can be manually adjusted using the MIN\_ADV and HR\_ADV signals, allowing the user to set the time without affecting the normal ticking of the clock. I added multiplexers to check the condition if the SETUP\_ALARM is on and to display the alarm time for that setting.
3. The CLK signal is ANDed with NOT SETUP\_TIME to allow the clock to tick normally when in time mode but stop in setup mode.

The clock’s carry-over logic ensures seconds flow into minutes and minutes into hours. The alarm system operates independently, starting at the minute counter without using the CLK, allowing the user to set the alarm time manual. There is no cary over in the alarm, as we’re only setting it one by one.

The clears work differently to clear different values.

The clock has seconds counter, but the alarm does not because it doesn’t matter as we only check the minutes and hour for alarm.

1. 1 decimal second equals 0.864 conventional seconds.

1 decimal minute equals 1.44 conventional minutes.

Regular Calendar:  
1 day = 24 hours × 60 minutes × 60 seconds = 86,400 seconds

French Revolutionary Calendar:  
1 day = 10 hours × 100 minutes × 100 seconds = 100,000 decimal seconds

1 decimal second = 86,400 / 100,000 = 0.864 seconds

1 decimal minute = 1,440 / 1,000 = 1.44 minutes

1. Days: Rolls over from 10 back to 1, incrementing the week.

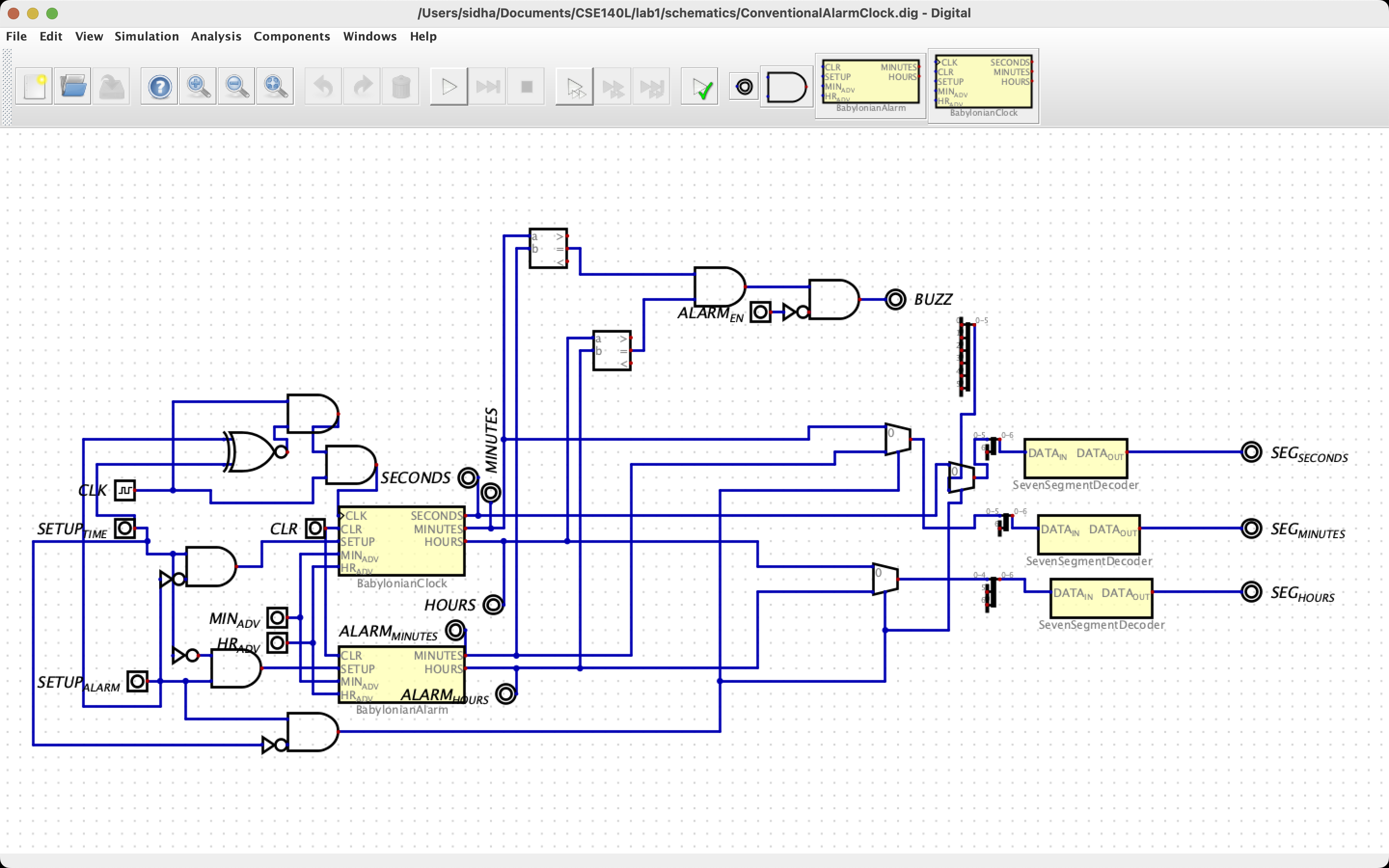
Weeks: Rolls over from 3 back to 1, incrementing the month.

Months: Rolls over from 12 back to 1, incrementing the year.

Years: Increment based on the month rollover, limited to 16 years.

So, I used the similar setup as in the Part 2 where I added the days, weeks, and months after the hours and extended the gate wires for them all along. I later added more equality comparators to check the alarm buzz condition with days, weeks, years as well.

ConventionalAlarmClock



FrenchRepublicAlarmClock

A screenshot of a computer

Description automatically generated

FrenchRepublicCalendarAlarmClock

A screenshot of a computer

Description automatically generated

To change from part one to two I changed the max bits from 59, 59, 23 to 99,99,10 for seconds, minutes, and hours respectively. I also had to change the number of bits for the gates/ splitter/mergers and outputs.