Indian Institute of Technology Hyderabad

Course: Electronic Devices & Circuits Lab

Course Code : EE2301

Lab Assignment 3

Digital Clock

Submitted By :
Aroshish Pradhan
EE23BTECH11009
Venkatesh Bandawar
EE23BTECH11010

Submitted To :
Gajendranath Chaudhury
Dept.of EE
IIT Hyderabad

Contents

1 Introduction

This project involves creating an 8 hour digital clock from basic digital components. The clock has a seconds display, a minutes display and an hours display. It is created using asynchronous counter circuits which is made by cascading D Flip Flops. The output of the counters is connected to BCD to Seven Segment Decoder to display the output in decimal numbers. The seconds and minutes have to reset when they reach 60 and the hours has to reset when it reaches 8 for the clock to function correctly. The reset logic for the counters is implemented accordingly, described further in the report. The input clock signal is provided using an Arduino Board which generates a square wave of frequency 1 Hz. The Logic High and Logic Low of the circuit are also provided through the Arduino. In the next project, we shall create a 1Hz oscillator circuit using op-amps that can be used to give the clock signal and then the circuit will work from basic, without any microcontrollers.

2 Aim of the Experiment

The aim of the experiment is to construct the circuit of an 8 hour digital clock and demonstrate its working.

3 Experimental Setup

The clock contains a seconds display (0 to 59, reset at 60), a minutes display (0 to 59, reset at 60) and an hours display (0 to 7, reset at 8).

For the seconds display, two counters were used, one for the units digit (mod 10) and one for the tens digit (mod 6).

The following components were used:

- IC 7474 - D Flip FLop
- IC 7447 - BCD to Seven-Segment Decoder
- IC 7400 - NAND Gate
- IC 7408 - AND Gate
- Seven-Segment Display
- Arduino UNO (for clock signal)

The units digit counter was created by cascading 4 D Flip Flops (mod 10 $\implies$ 4 bits required) in asynchronous way ($\overline{Q}$ of preceding Flip Flop connected to CLK of

next). The flip flop should reset when the output become 1010, so the appropriate clear signal is given to the flip flops. The circuit is as shown below (Fig. 1).

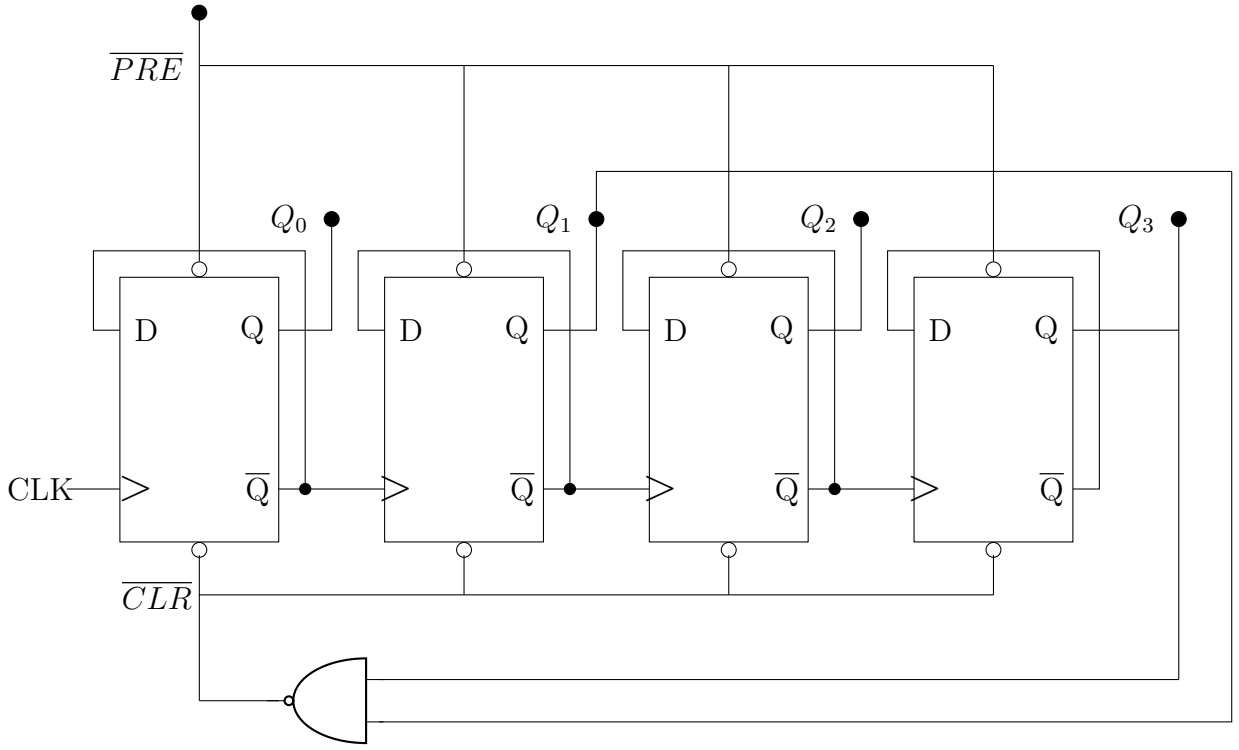


Figure 1: MOD 10 Asynchronous Counter

The tens digit counter requires 3 flip flops since it has to count up to 6 (110 in binary). Here the reset should happen at 110. The circuit is given in the next page (Fig. 2).

For the hours display we created the asynchronous counter using 4 D Flip Flops to count up to 7 (0111) and reset at 8 (1000). The MSB of the counter becomes 1 for the first time at 8 so that bit is inverted and given to $\overline{CLR}$. The circuit in the next page (Fig. 3).

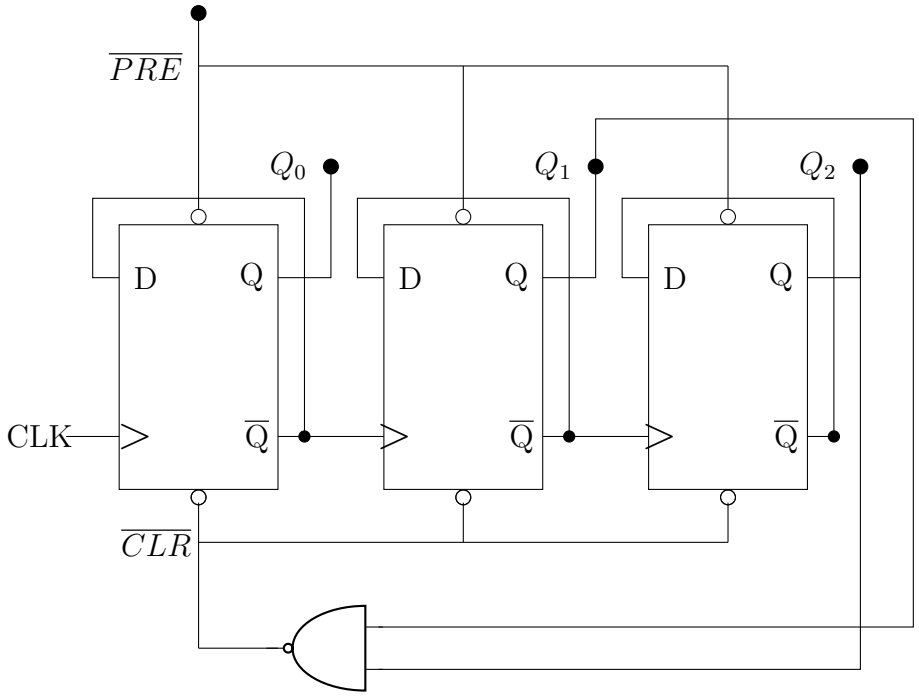


Figure 2: MOD 6 Asynchronous Counter

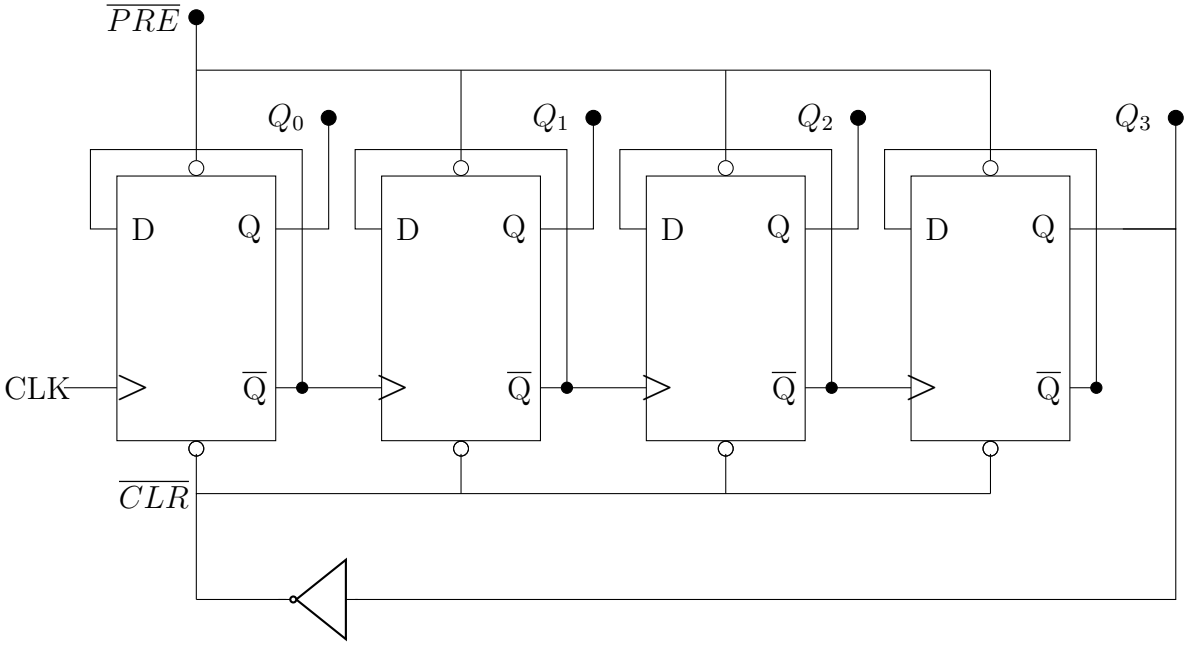


Figure 3: MOD 8 Asynchronous Counter

These counters (two MOD-10, two MOD-6 and one MOD-8) are put one after the other to generate the complete clock output in binary, with the $\overline{CLR}$ of previous counter connected as CLK of next counter as shown in the block diagram.

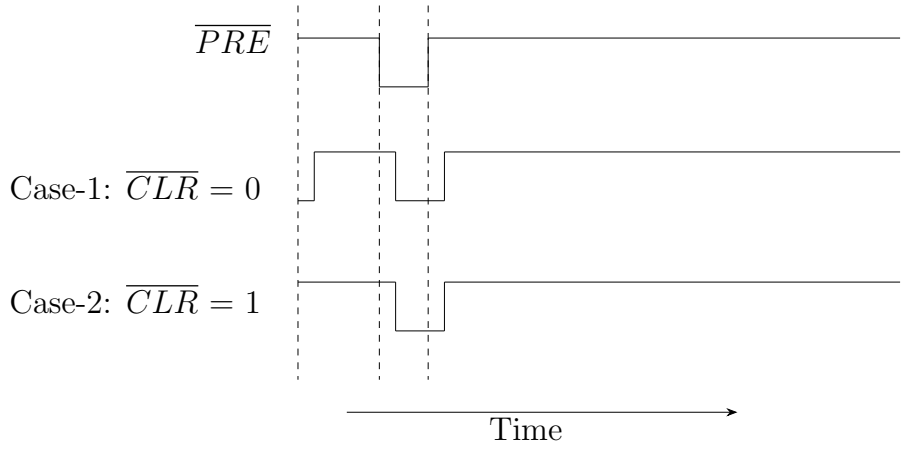
The outputs of the counters are connected to BCD to 7 segment decoders that convert the binary input into outputs that are fed into a 7-segment display that displays the corresponding number in decimal system. The output is then observed on the displays.

4 Working Principle

The circuit consists of multiple counters one after the other, connected in asynchronous way. The CLK of a counter is the $\overline{CLR}$ of the previous counter. The way this works is described in detail below:

- Consider the first mod-10 counter (for the units digit of the seconds display). Its clock signal is given by the Arduino, which generates a clock pulse of time period 1 sec (1Hz frequency). The $\overline{CLR}$ pin is connected to NAND of the FF outputs depending on the reset condition.
- Initially, $\overline{PRE}$ is set to 1. Depending on the previous state of the D Flip FLop, there may be some random state in the FF. Depending on the state, $\overline{CLR}$ may be 0 or 1.
  - **Case-1**: $\overline{CLR} = 0$ initially - The outputs of FFs will be set to 0 (refer table below). Then after a small delay, $\overline{CLR}$ becomes 1 (because outputs are 0) and FFs operate normally. $\overline{PRE}$ momentarily becomes 0 and then 1 so FFs continue to operate normally starting at 0000.
  - **Case-2**: $\overline{CLR} = 1$ initially - As $\overline{CLR}$ and $\overline{PRE}$ are 1, the garbage value is retained for a while (normal FF operation). Then $\overline{PRE}$ is made 0 momentarily which pulls all outputs to 1 (refer table below), which in turn pulls $\overline{CLR}$ to 0. $\overline{PRE}$ returns to 1 and $\overline{CLR}$ remains at zero, pulling all outputs back to zero ($\overline{CLR}$ after a delay becomes 1 now). After this process both $\overline{CLR}$ and $\overline{PRE}$ are at 1, so FFs continue operating normally strarting from 0000.
  - The delays here are due to the gates, typically in order of nanoseconds.

  This reset mechanism works for all the counters simultaneously, ensuring the clock starts from 00 : 00 : 00.



TRUTH TABLE					
INPUTS				OUTPUTS	
$\overline{PR}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
0	0	X	X	X	X
1	1	↑	1	1	0
1	1	↑	0	0	1
1	1	0	X	Q_0	$\overline{Q_0}$

Working of the Counter

- Now the counter starts at 0. Before the first positive edge of the clock, $\overline{Q}$ was at 1. During the rising edge, this $\overline{Q}$ gets transferred to output and $\overline{Q}$ becomes 0. Since $\overline{Q}$ fell from 1 to 0, it does not trigger the next FF (CLK is positive edge triggered). Therefore, output is 0001.
- During the next clock cycle, similarly, the output of first FF becomes zero, but $\overline{Q}$ changes from 0 to 1, triggering the next FF. Therefore, the earlier $\overline{Q}$ of the next FF becomes output so now output is 0010. in the next cycle, output become 0011, third FF is not triggered yet. Then both outputs become 0 simultaneously, causing output of third FF to become 1 (0100). This way, the keeps counting upward.
- The $\overline{CLR}$ signal is the NAND of the 2^{nd} and 4^{th} bits of the counter. So before the output becomes 1010, $\overline{CLR}$ is at 1 ($\overline{PRE}$ was at 1) so normal operation continues. When output becomes 1010, $\overline{CLR}$ becomes 0 for an instant, causing all outputs to become 0, which again makes $\overline{CLR}$ 1.

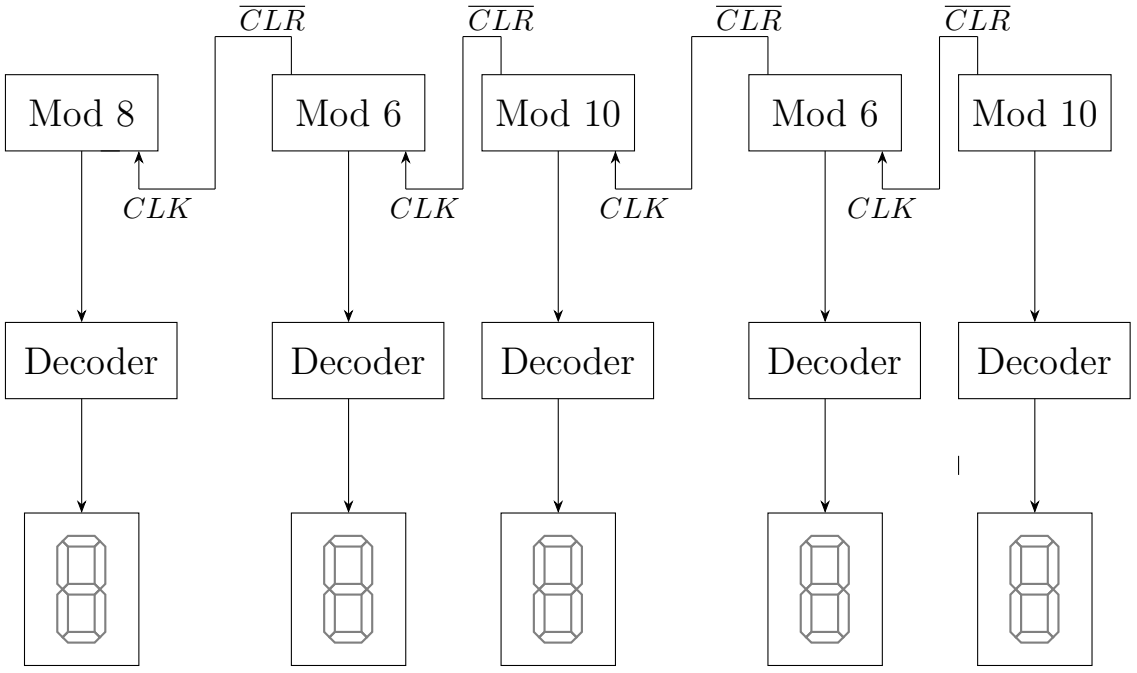
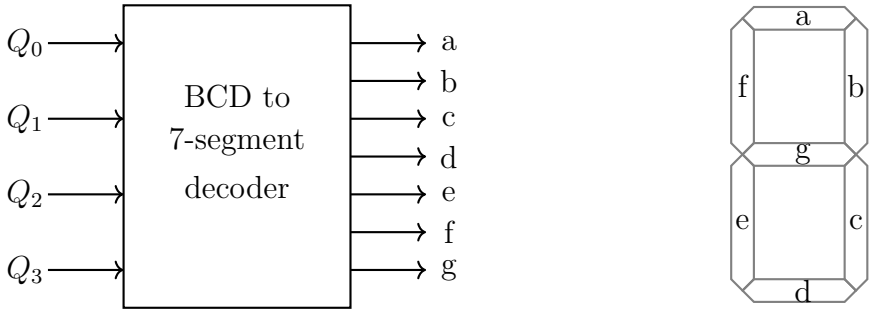


Figure 4: Block Diagram of Circuit

- The tens digit of the seconds counter is built similarly. its CLK is the $\overline{CLR}$ of the previous counter. When that counter resets, the $\overline{CLR}$ becomes momentarily 0 then 1, triggering this FF. It works similarly except that it resets at 110 (6). The $\overline{CLR}$ of this is given to the units digit of the minutes as CLK.
- The minutes work exactly like the seconds but with lower frequency. The $\overline{CLR}$ of the tens digit is given as CLK of the hours FF.
- The hours FF resets at 8 (1000) similarly. The tens digits of the hours display is forced to show 0 (by grounding) as it is not needed for an 8-hour clock.

BCD to Seven Segment Decoder

- The outputs of all FFs (shown as Q_0, Q_1, Q_2 etc.) are in binary but we need to display them as decimal numbers on the 7-segment display. To do the conversion, we use BCD to 7-segment decoder that takes these binary inputs and gives 7 outputs that can be connected to the display.



Digit	Q_0	Q_1	Q_2	Q_3	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Table 1: Truth table of BCD to 7-segment decoder

- From truth table we get:

$$
\begin{aligned}
a &= Q_0 + Q_2 + Q_1Q_3 + \overline{Q_1}\,\overline{Q_3} \\
b &= \overline{Q_1} + \overline{Q_2}\,\overline{Q_3} + Q_2Q_3 \\
c &= Q_1 + \overline{Q_2} + Q_3 \\
d &= \overline{Q_1}\,\overline{Q_3} + Q_2\overline{Q_3} + Q_1\overline{Q_2}Q_3 + \overline{Q_1}Q_2 + Q_0 \\
e &= \overline{Q_1}\,\overline{Q_3} + Q_2\overline{Q_3} \\
f &= Q_0 + \overline{Q_2}\,\overline{Q_3} + Q_1\overline{Q_2} + Q_1\overline{Q_3} \\
g &= Q_0 + Q_1\overline{Q_2} + \overline{Q_1}Q_2 + Q_2\overline{Q_3}
\end{aligned}
$$

- The above functions are the outputs of the decoder that are fed into the display to show decimal numbers as output.
- The ICs were connected in the way shown in Fig.4 to complete the circuit and the V_{cc}, ground, and initial clock were provided by the Arduino board.

5 Observations

The clock was found to be working properly. The seconds, minutes and hours displays were incrementing and resetting correctly. The clock started from a random state, reset itself to 00:00:00 and then showed up to 07:59:59. For testing purposes we increased the clock frequency and checked that all counters were working properly.

6 Extension to 24 Hour Clock

The same circuit was extended to include 24-hour functionality. The clock can be converted back to 8-hour with bery simple changes. The $\overline{CLR}$ of the units digit was given as CLK of tens digit. The units digit's $\overline{CLR}$ was as follows:

- Take NAND of 2^{nd} bit of tens digit and 3^{rd} bit of ones digit. This output is 1 when 24 does not occur and is 0 when it occurs.
- Take NAND of 2^{nd} bit and 4^{th} bits of ones digit. This output is 1 when 10 does not occur and is 0 when it occurs.
- Take AND of above two and give it to $\overline{CLR}$ of units digit. This way when either of 10 or 24 occurs, units digit resets to zero.
- The first output (NAND of 2^{nd} bit of tens digit and 3^{rd} bit of ones digit) is directly given as $\overline{CLR}$ of tens digit so it resets at 24.

This way the clock will work upto 24 hours.

Conversion to 8-hour

To convert the 24-hour clock to 8-hour, the following steps were followed:

- We took NAND of 2^{nd} bit and 4^{th} bits of ones digit. Now, just detach the wire of 2^{nd} bit and use it to short the NAND gate terminals, effectively converting into a NOT gate. Now $\overline{CLR}$ is the NOT of the 4^{th} bit, meaning it will now reset at 8 instead of 10. We can leave the 'reset at 24' part as it is because 24 won't occur.
- The second change is that remove the clock signal of the FFs that correspond to the tens digit of the hours and reset it to zero by grounding $\overline{CLR}$.

A picture of the circuit is attached below:

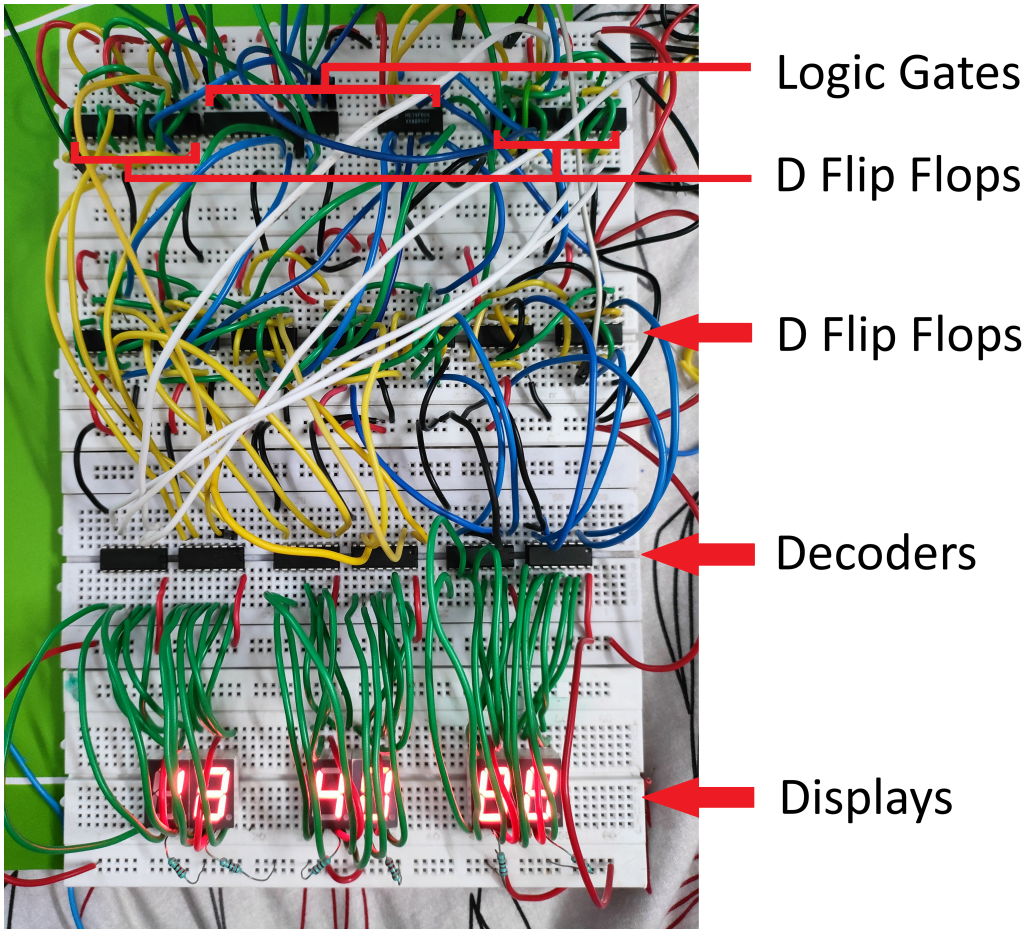


Figure 5: Image of Circuit

7 Conclusion

Both the 8-hour and 24-hour clocks were working as intended. This experiment demonstrated the use of counter circuits in generating numbers in binary, BCD conversions and display on 7-segment display. The use of logic gates in the circuit for resetting ensured proper functioning of the clock. The principle behind the functioning was relatively simple though assembly and debugging the circuit took time. This project paved the way for creating more digital circuits of similar logic. In the next project, we shall build an oscillator for providing the clock input.