National University of Computer & Emerging Sciences Islamabad

EL-1005 Digital Logic Design

SPRING 2024

SEMESTER PROJECT

WHACK-A-MOLE GAME

Lab Instructor:

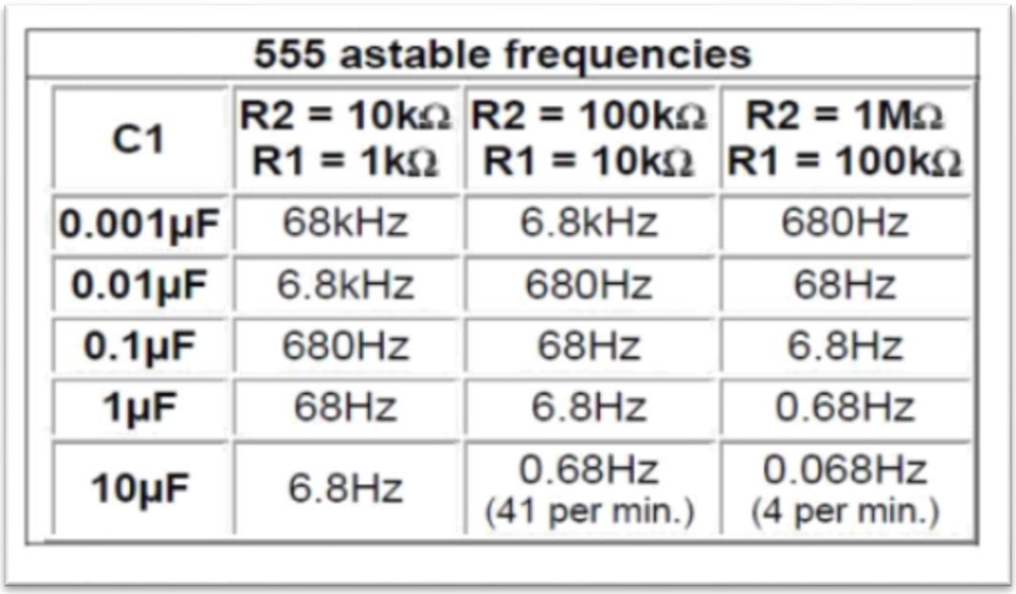
ENGR. AYESHA QAISER HASHMI

Presented by:

Hafiz Muhammad Ibrahim {23i-2013}
Abdul Ahad {23i-2014}
BS CYS - B

Use of 555 – Timer IC for Pulse generation:

The first step was to solve the problem of generating clock pulse without the use of external circuit elements like microcontroller or the trainer board on which the clock pulse was easily available. Therefore, we used 555 timer IC for generating clock pulse. We used two of them, one for a delay of approximately a second which we used in the count down clock (30-0) and the other one was used in prompting the user to press button within specified time (1.5 sec) approximately.
For the timer with delay of one second me and my group member adjusted the resistance and capacitance for maintaining the charging and discharging of current up to the interval we desired. We used this table as a reference.

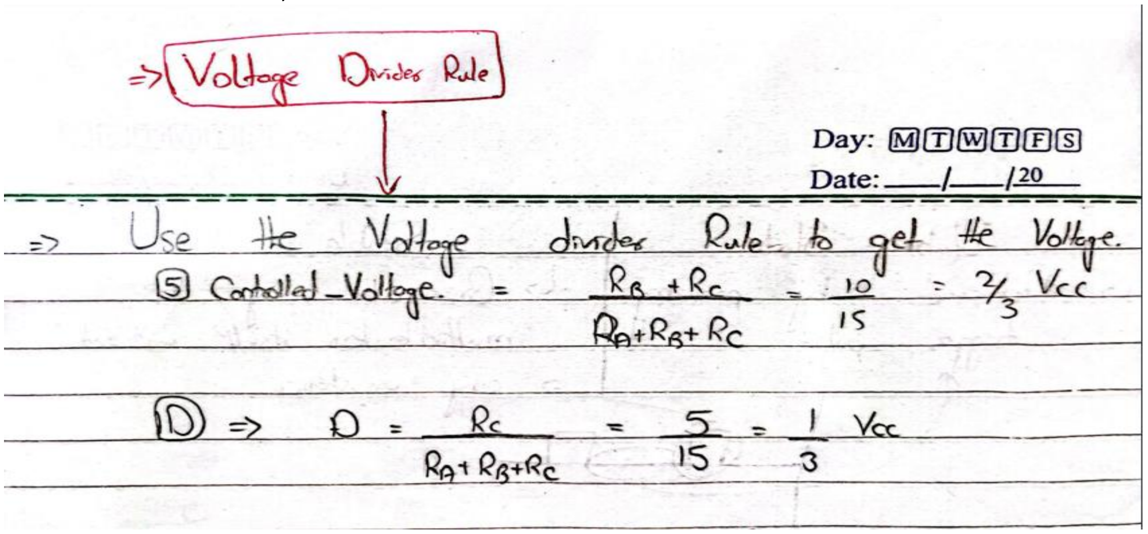


555 astable frequencies

C1	R2 = 10kΩ
	R1 = 1kΩ
		R2 = 100kΩ
		R1 = 10kΩ
			R2 = 1MΩ
			R1 = 100kΩ

0.001µF	68kHz	6.8kHz	680Hz
0.01µF	6.8kHz	680Hz	68Hz
0.1µF	680Hz	68Hz	6.8Hz
1µF	68Hz	6.8Hz	0.68Hz
10µF	6.8Hz	0.68Hz (41 per min.)	0.068Hz (4 per min.)

The calculations behind it are,



=> $\boxed{\text{Voltage Divider Rule}}$

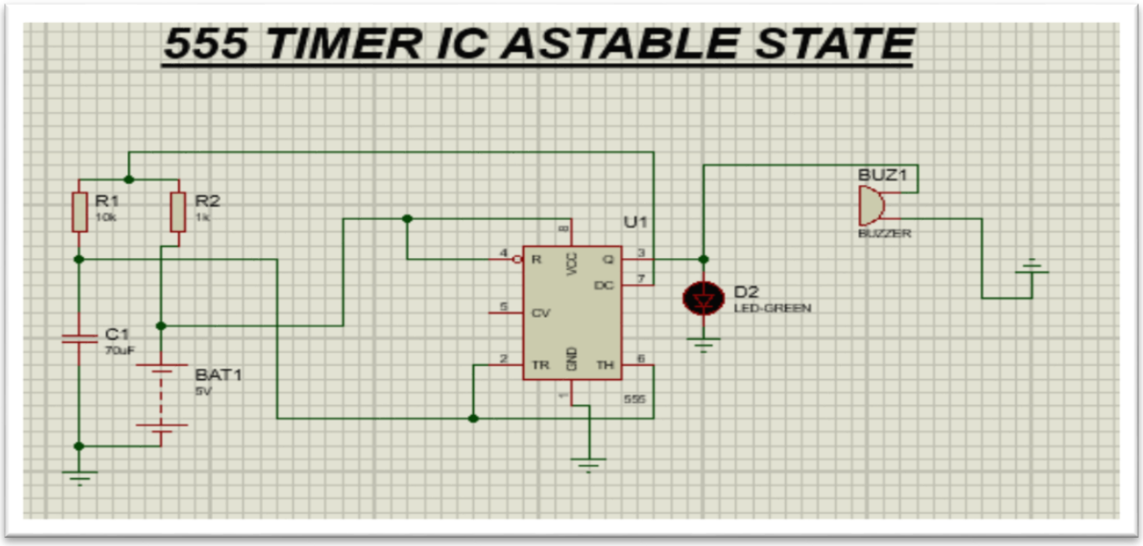
Day: M T W T F S
Date: ___/___/20___

=> Use the Voltage divider Rule to get the Voltage.

$\boxed{5}$ Controlled Voltage $= \dfrac{R_B + R_C}{R_A+R_B+R_C} = \dfrac{10}{15} = \dfrac{2}{3}\,V_{cc}$

$\boxed{D} \Rightarrow D = \dfrac{R_C}{R_A+R_B+R_C} = \dfrac{5}{15} = \dfrac{1}{3}\,V_{cc}$

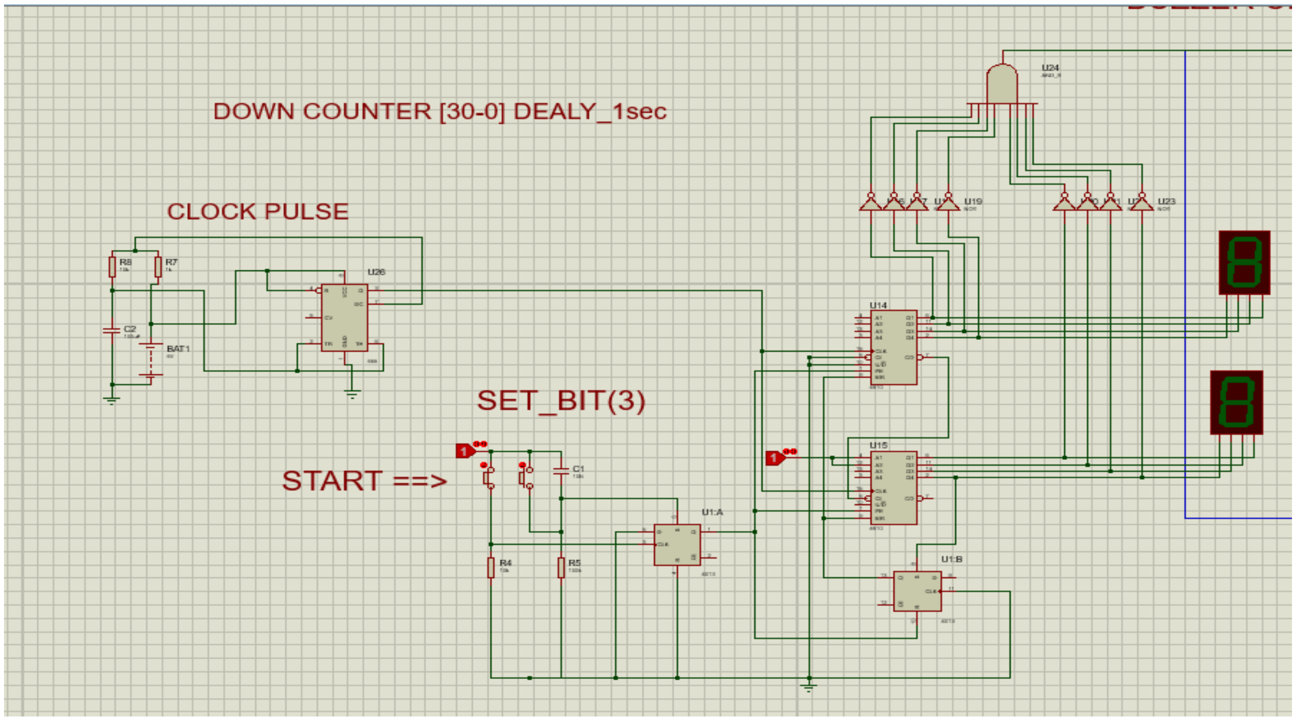
The breadboard implementation of this circuit is as per mentioned,



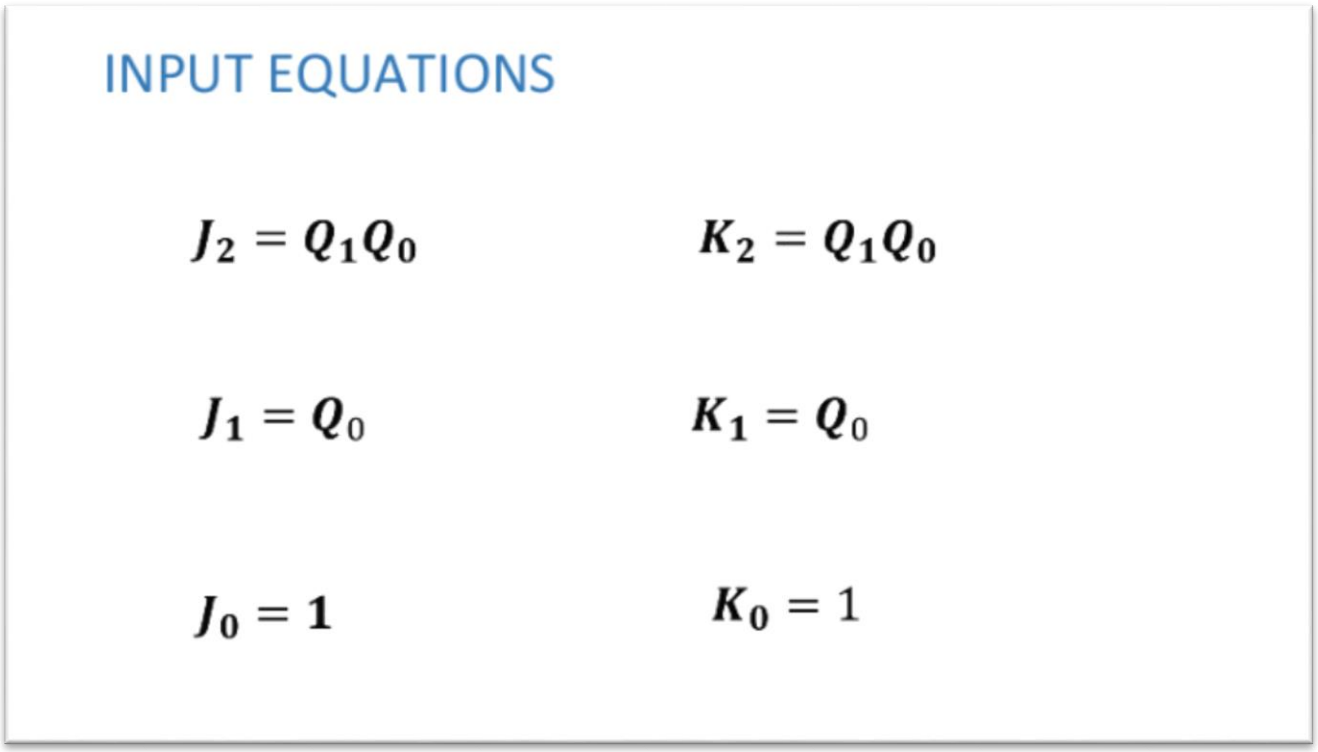
Then we implemented a down counter using **an up/down counter IC-4510** along with **2 BCD-7seg displays**. We used **2 D-Flip Flops along with capacitors and resistors and set the bit of 2nd Display to 3** as most of the down counters work from 99-0 if they are cascaded. For this purpose, we approached different techniques that we find helpful. After trying JK flip flops we used D flip flops and they worked. The only issue lies where we must set the bit to 3 manually otherwise it's all good.
Here is the circuit diagram of this circuit.

Set bit_3 = PRESS IT AT THE START OF THE GAME ONCE. (Set display to 3)

Start Press the start button to start the timer.



Then the main and the easiest step was to generate random numbers, basically to make the LED's glow at random. For this purpose, we used 3-bit JK Flip Flop and then implemented an up counter that helped us in achieving the randomness not true randomness but up to some extent random number generation.
Here are the equations that we derived from the mathematical work,



INPUT EQUATIONS

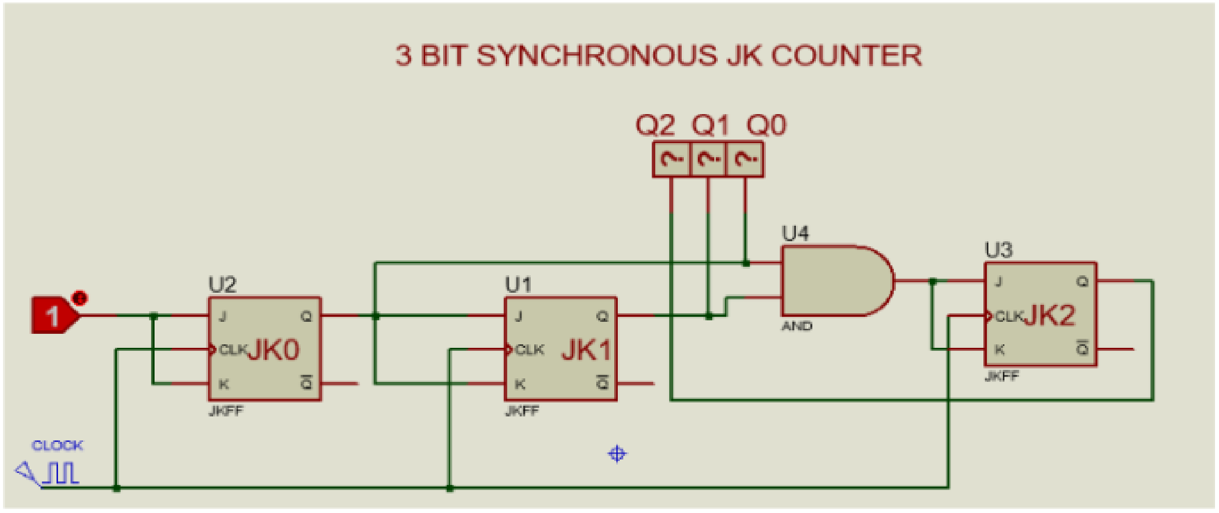
$J_2 = Q_1Q_0$ $K_2 = Q_1Q_0$

$J_1 = Q_0$ $K_1 = Q_0$

$J_0 = 1$ $K_0 = 1$

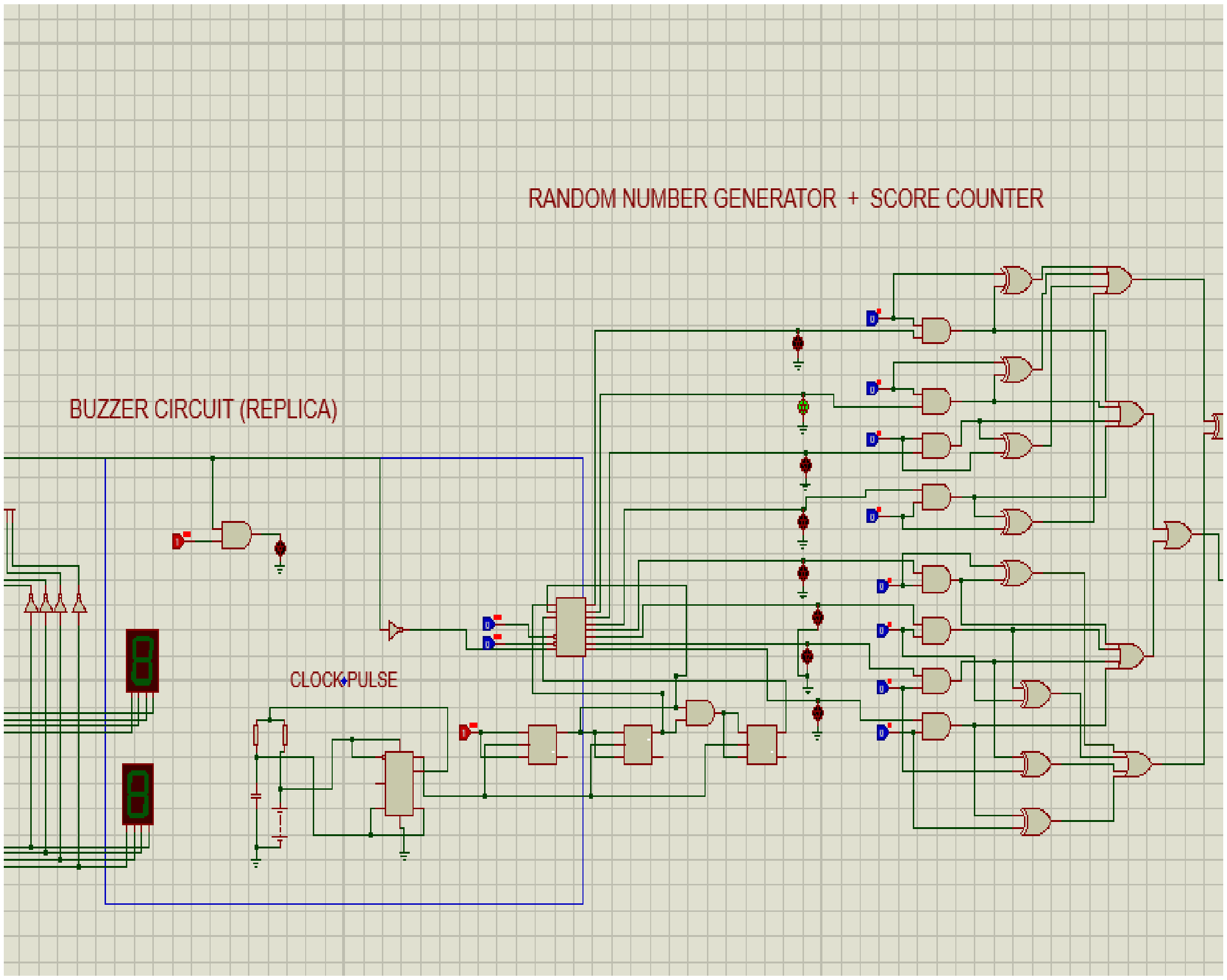
The proteus diagram of this part is:

IMPLEMENTATIONS OF EQUATION



Then we used a decoder/demultiplexer **IC-74HC238** which we also used once in our lab task. The selection of this IC was basically linked with its functionality as other decoder IC's give outputs based on the input of seven segment in simple words multiple outputs for a specific state which will make the circuit less efficient as there was a junk of gates to be applied then. Anyhow, the whole circuit diagram is as mentioned.

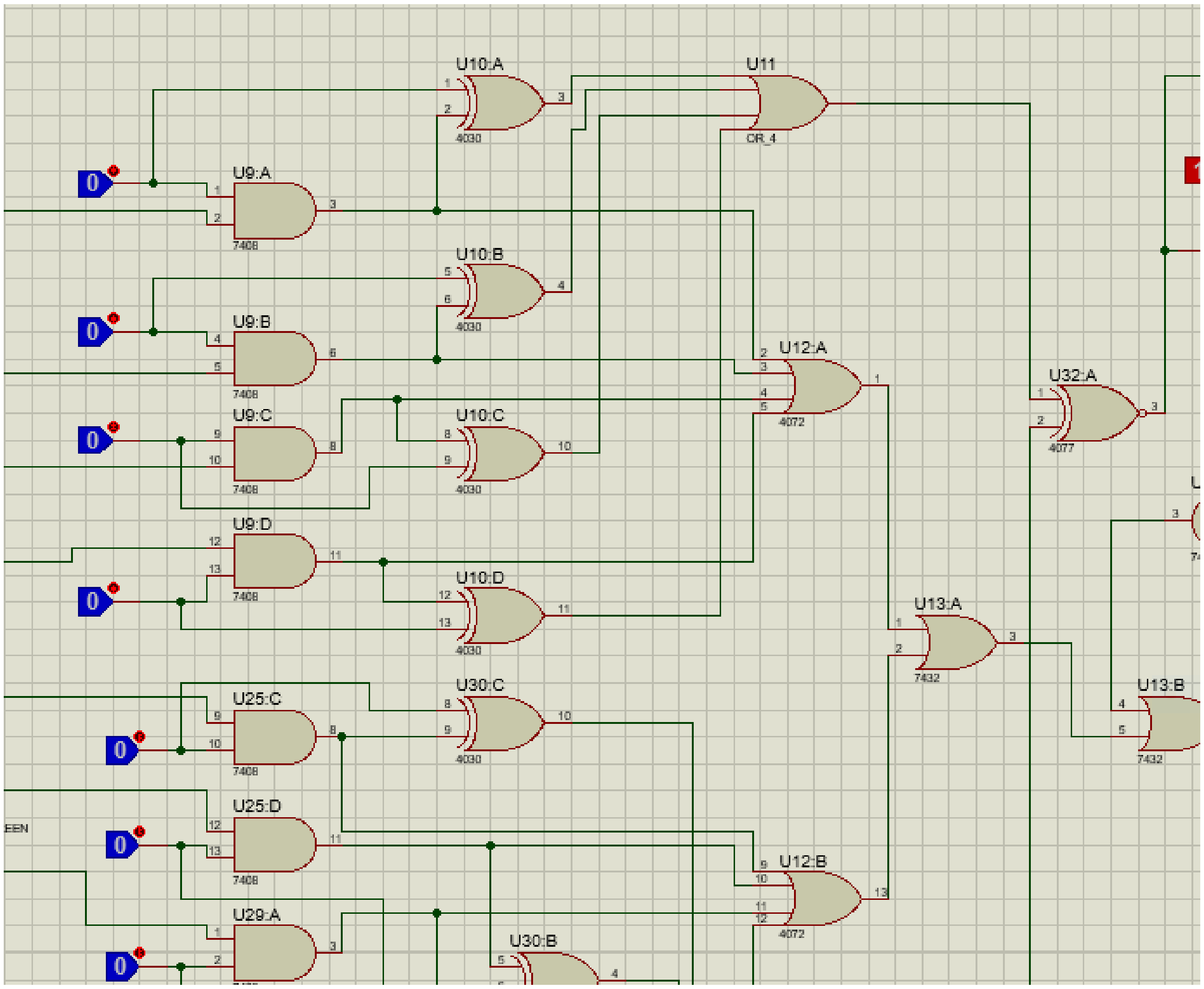
<u>IC-74HC238 since it has 3 enables that should be set to 0,0 & 1.</u>
<u>The one depends upon the clock. It will remain 1 until count down timer is not 00.</u>



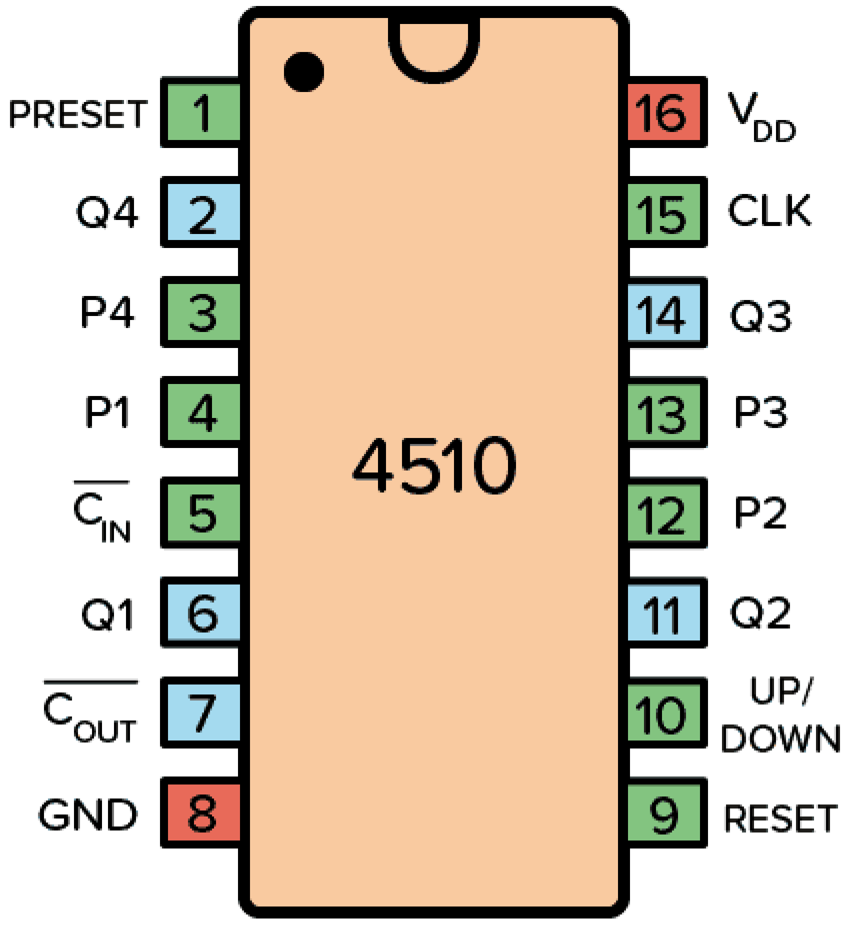
Things might not be clear well enough, for which we apologize as it was totally impossible for us to get the clear picture or snapshots, but we have attached all the equipment details that we have used in this project. Moreover, the proteus file is also attached.
The buzzer circuit will work when the clock is set to 00. This will be basically at the start before setting the clock to 3 bit and at the end when the clock turns to 00 and stops the supply to all other circuit elements that come after the buzzer. Since the buzzer was unable to produce sound so I used LED to help clearly determine the working.

Then comes the final part where we struggled a lot. It was basically a two-digit score counter with the ability to set/reset, increment, and decrement the scores. For this purpose, we used and gates to check if both the buttons are pressed simultaneously for the increment part.
And for the decrement part we used XOR gates which will give an output of 1 if both the input bits are different. We linked it with the button/logic toggle and other with the output of the AND gate.

So, if the button was pressed and the output was still 0 we get input combinations of 1/0 and output 0. This will result in set bit (1). Similarly, we applied this technique with all the other 7 buttons.
Then we used OR gates to minimize the output stream. Here is the circuit overview,

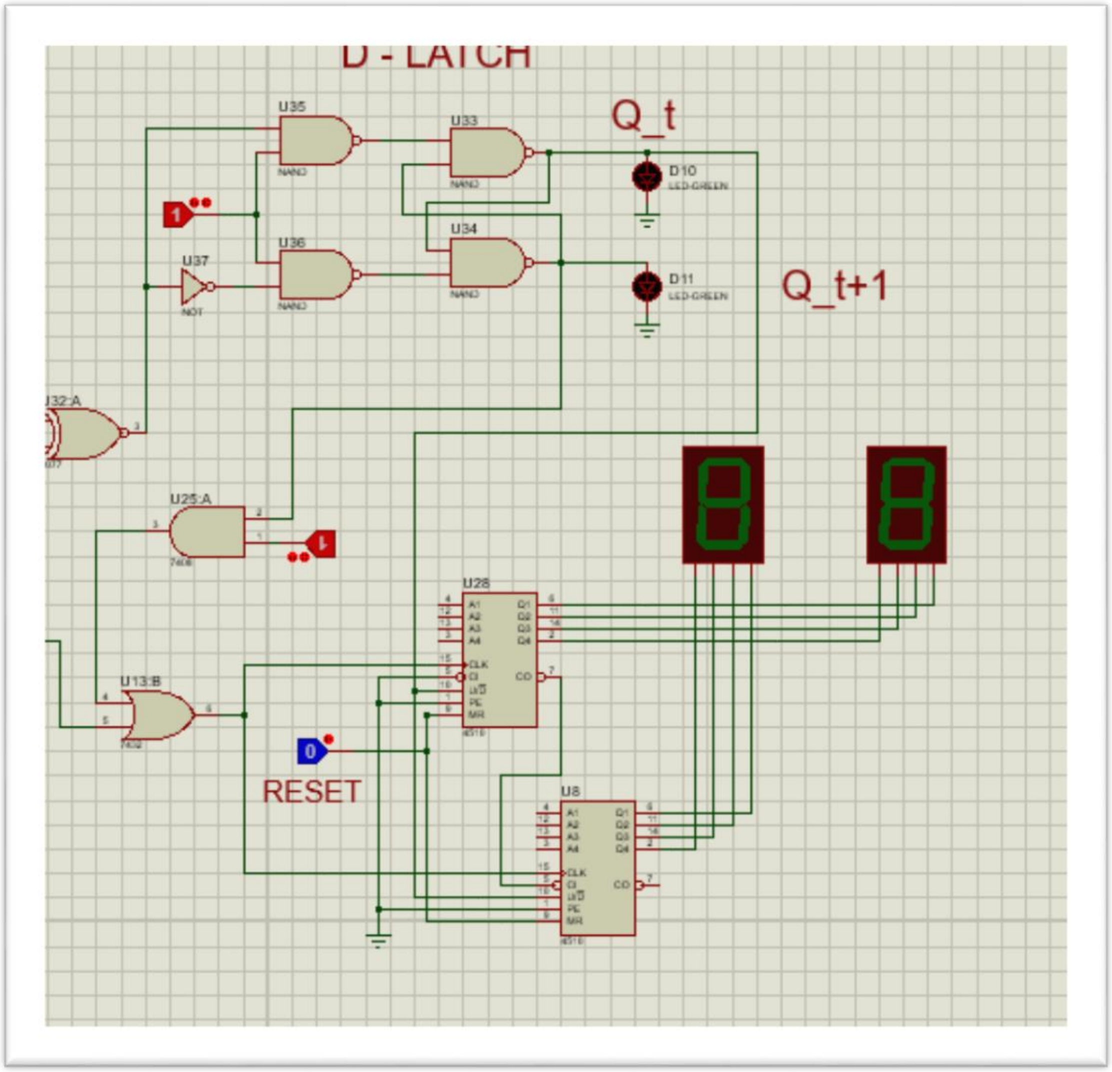


Then we used XNOR. Which was a tricky choice. Let me explain why it is so?
When we started the circuit, it was automatically adjusted to 0 and 0. So for some inputs we must get one at the output at any cost to set the counter to add up the score.
The counter **IC-4510** will add in count when the input at pin 5 is 1 and will count down when the input at pin 5 is 0. Also, we used different combinations of AND gates therefore pressing the buttons was a step must perform.
The pin configuration of **IC-4510** is as per mentioned,



Then we used a memory element D-latch NAND based to preserve if any of the false button was pressed. Since we used clock pulse as the input of random number generator circuit it is important to use a memory element to store the previous state as it would be washed out even before the next buttons pressed. That was also a logical part.

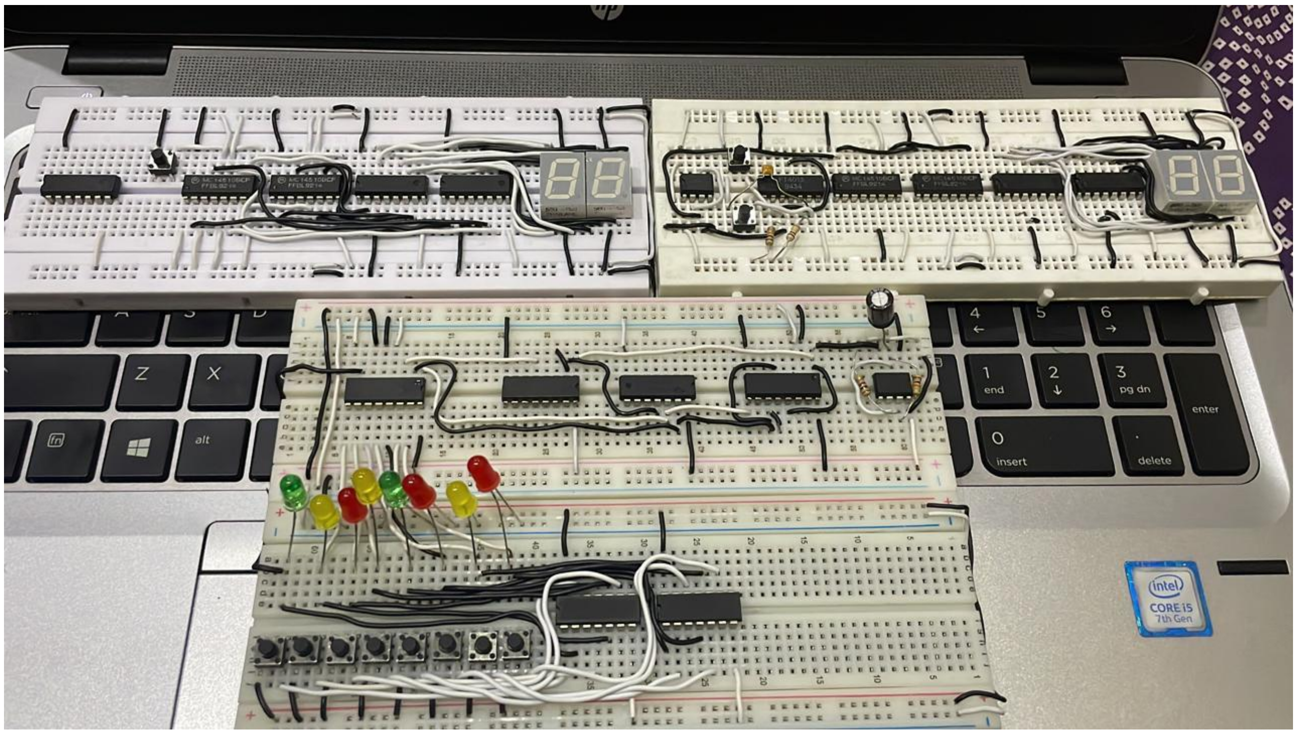
Here is the close overview of the circuit,


D - LATCH
Q_t
Q_t+1
U35
U33
U34
U36
U37
NAND
NOT
D10
D11
LED-GREEN
U32:A
U25:A
U13:B
U28
U8
4510
CLK
CI
U/D
PE
MR
CO
A1
A2
A3
A4
Q1
Q2
Q3
Q4
1
0
RESET

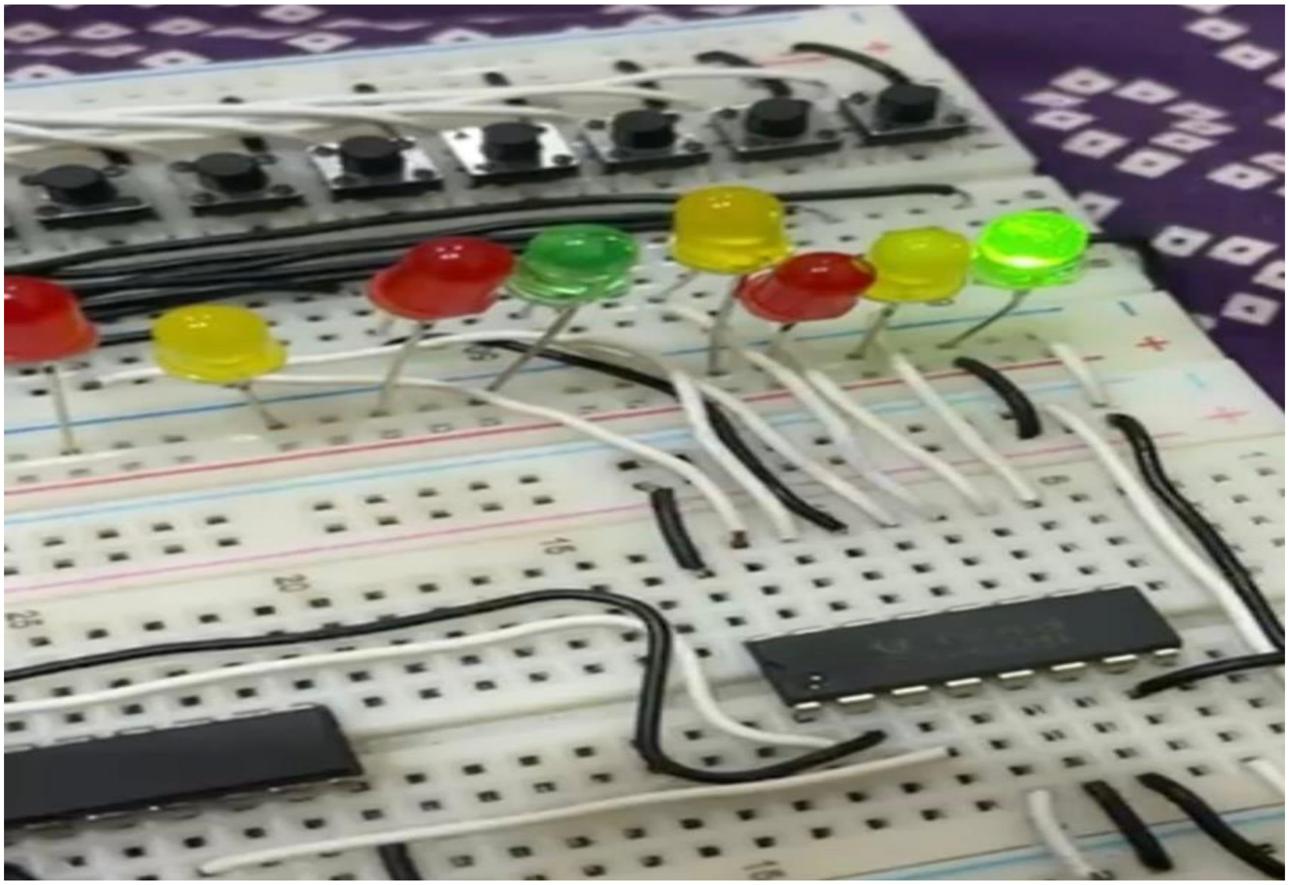
MATERIAL USED:

1. **555 timer IC** (2 units).
2. Resistors: 10k ohm (+3), 1k ohm (+2), and 100k ohm (+1).
3. Capacitors: 1nF (+1), 100mF (+1), and 80mF (+1).
4. **NAND based D-Latch**. (1 unit) basically a NAND gate required.
5. 4013 D-FF (2 units).
6. **Up/Down Counter IC's 4510** (4 units).
7. **Displays {BCD TO 7-SEG anode [8 input pins]}** (4 units).
8. **Decoder/Display Driver {7447}**
9. NOT GATES (4 units).
10. Push Buttons. (12 units). In proteus we used logic toggle.
11. LED's (11 units).
12. **74LS112 JK-FF (3 units).**
13. **Decoder IC-74HC238** (1 unit).
14. AND GATES (3 units).
15. NOR GATES (2 units).
16. XNOR GATE (1 unit).
17. OR GATES (4 units).
18. BATTERY. (5v output).
19. Buzzer.
20. Breadboard (5 pcs).

Practical Implementation:



BEFORE:



AFTER: