UNIVERSITETI I EVROPËS JUGLINDORE
УНИВЕРЗИТЕТ НА ЈУГОИСТОЧНА ЕВРОПА
SOUTH EAST EUROPEAN UNIVERSITY

**Faculty of Contemporary Sciences and Technologies**
**Tetovo**

# Digital Logic Design

**(Sem. 4, 2022/2023)**

**Project:**

***Binary counter 0 to 7 and 7 to 0***

**Student**:
Ron Ismaili (130050)

**Mentor**:
Prof. Dr. Mentor Hamiti

May 2023, Tetovo

# Table of Contents

## Contents

# 1. Problem definition

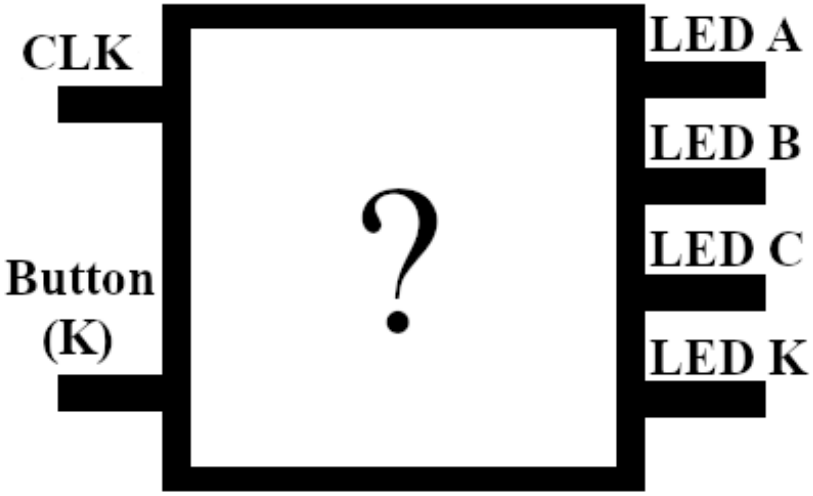
For the purposes of this project, I need to design and implement a binary counter which can count from 0 to 7 and from 7 to 0. The way I plan to implement this circuit is to have a clock (CLK), 1 external input (K) which will be implemented using a button and 4 external outputs (A, B, C and K) which will be implemented using LEDs.

The idea is that for every time interval (positive edge) of the clock where the LED K is off (K = 0) the counter counts up by one, whereas every time the LED K is turned on (K = 1) the counter counts down by one.

There are a total of four LEDs in the circuit. LEDs A, B and C are for demonstrating the 3-digit binary number, and LED K is for showing if the counter will count up or down on the next positive clock edge.

LED A is the most significant bit whereas LED C is the least significant bit. The counter's initial state will be 000 (meaning LED A = LED B = LED C = 0) when it is connected to a power source.

The following is a black box diagram of what the final circuit should look like:




*The black box diagram*[1]

[1] I created this diagram using GIMP.

# 2. Logic design

Now that we have clearly defined the problem at hand, we may continue by working towards a working solution. Since the goal of this project is to implement a counter, I have decided to use D-flip-flops because they are very easy to work with.

The first step I devised in my plan to completing this circuit was to create the state table of the circuit.

| K | A | B | C | $A^+$ | $B^+$ | $C^+$ | $D_A$ | $D_B$ | $D_C$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

*Table for K = 0 (counting up)*

| K | A | B | C | $A^+$ | $B^+$ | $C^+$ | $D_A$ | $D_B$ | $D_C$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

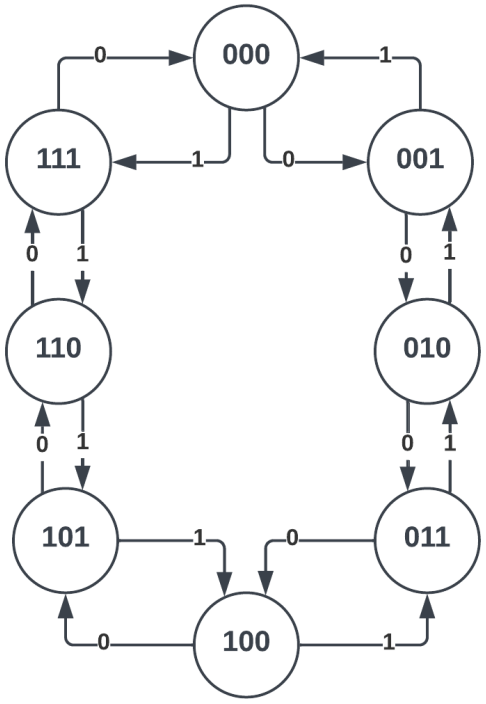
*Table for K = 1 (counting down)*

Since we have 4 external inputs, we need to have 16 entries (rows) in our combinational table. The first column is K which is reserved for representing if we are counting up/down, A, B and C are reserved for the binary representation of the numbers 0 through 7.

If instead of setting K as the first column I had set it as the last one, the table would have looked much more confusing. In that case the table would alternate between counting up and down for all numbers from 0 to 15. In the representation I used (K as the first column), numbers 0 – 7 are reserved for counting up and numbers 8 – 15 are reserved for counting down.

Even though I have created 2 different tables, logically speaking**, we only have a single table with 16 entries**. The reason why I split it up like this is to make it easier to parse the information for the case when K = 0 (counting up) and K = 1 (counting down).

After filling out the first four columns of our table, where A, B and C represent the current states for the D-flip-flops, we now have to determine what the next states for these flip flops are. The way we do this is by taking a look at the following state diagram:



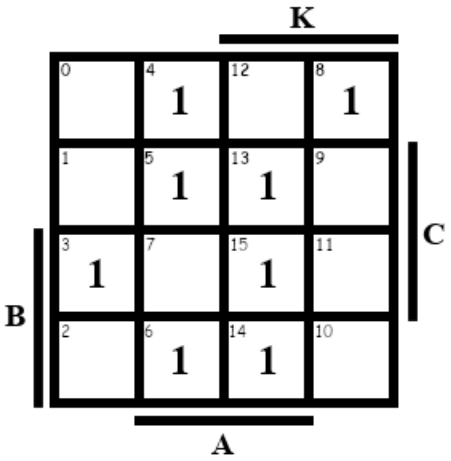

*The state diagram*[2]

The state diagram tells us that for the external input 0 the state is increment by one until it reaches state 111 at which point it goes back to its initial state meaning 000, and if the external input is 1 the state is decremented by one until it reaches the state 000 at which point it goes to 111.

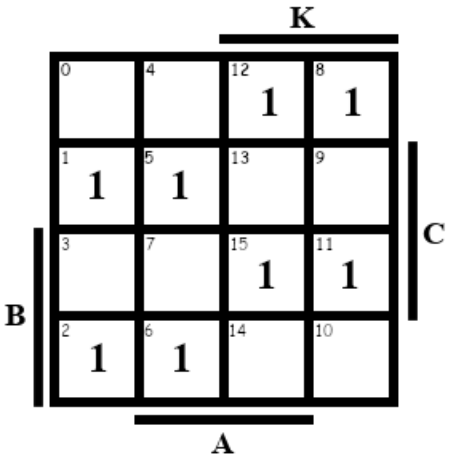
From this information we can easily find the next states i.e. $A^+$, $B^+$ and $C^+$ by looking at where the arrows take us in the state diagram. An example would be that for the current state 000 with external output 0 the next state would be 001.

Finding $D_A$, $D_B$ and $D_C$ is very easy now as we know that the D-flip-flop's output is determined by what is in its input, in our case we just carry over the information from the $A^+$, $B^+$ and $C^+$ columns. Now that the combinational table is filled out, we need to find the functions for $D_A$, $D_B$ and $D_C$, we can do this by examining their columns using Karnaugh maps.

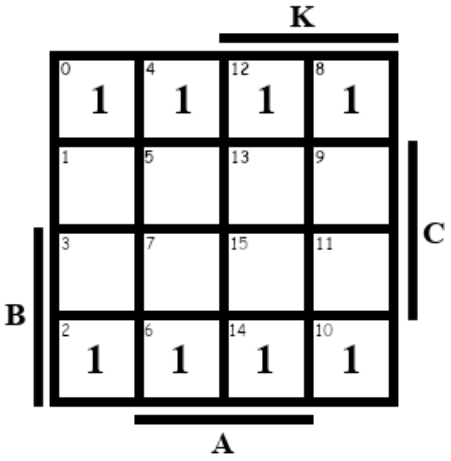
[2] I created this state diagram using Lucidchart.




*K-map for $D_A$*[3]




*K-map for $D_B$*




*K-map for $D_C$*

[3] I created these K-maps using GIMP.

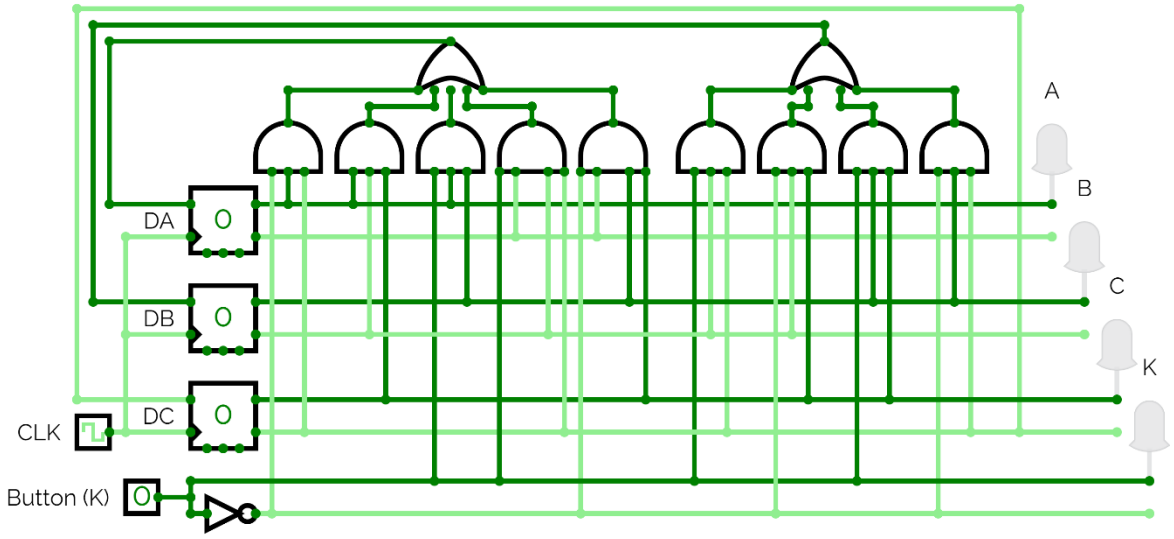
The functions that I derived from the aforementioned K-maps are the following:

$$D_A = \overline{K}A\overline{C} + A\overline{B}C + KAB + K\overline{ABC} + \overline{KA}BC$$

$$D_B = K\overline{BC} + \overline{KB}C + KBC + \overline{K}B\overline{C}$$

$$D_C = \overline{C}$$

Now that we have these functions, we have everything we need to create the first design of our circuit:

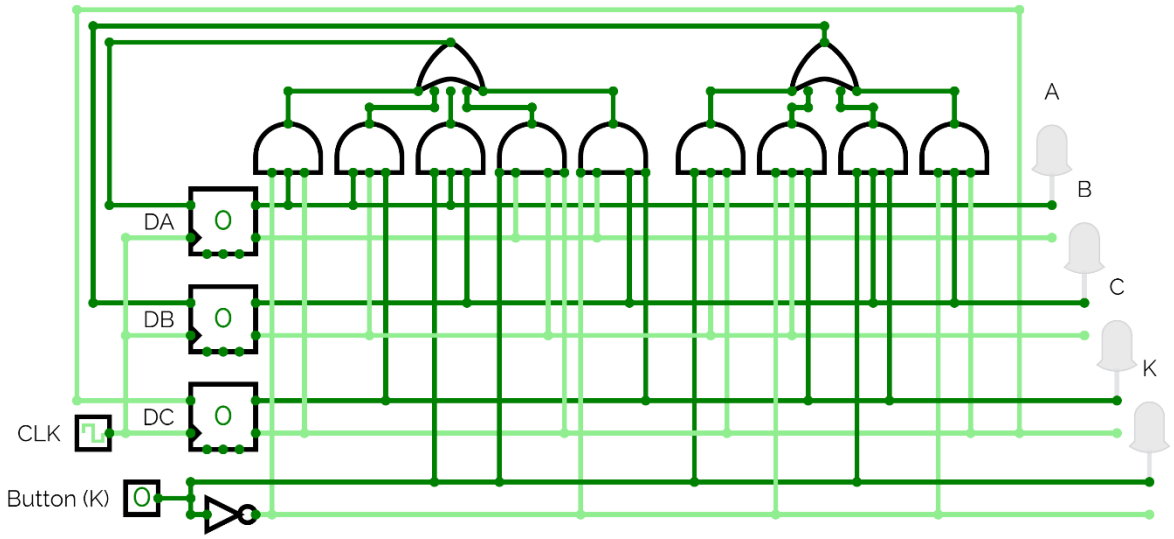



*The first design of the digital circuit*[4]

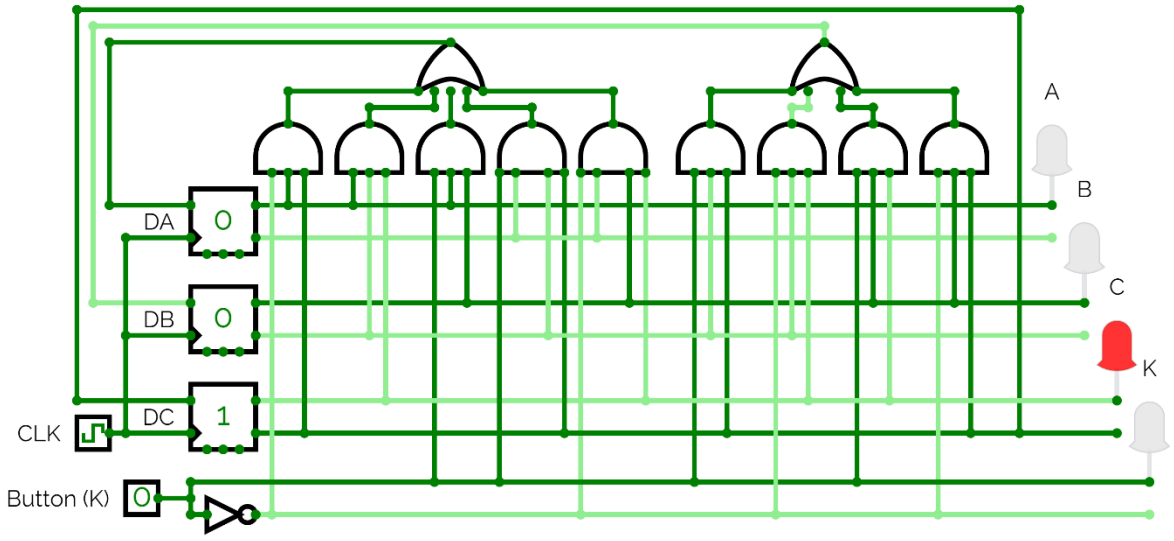
[4] I created this design using CircuitVerse.

# 3. Simulation and testing

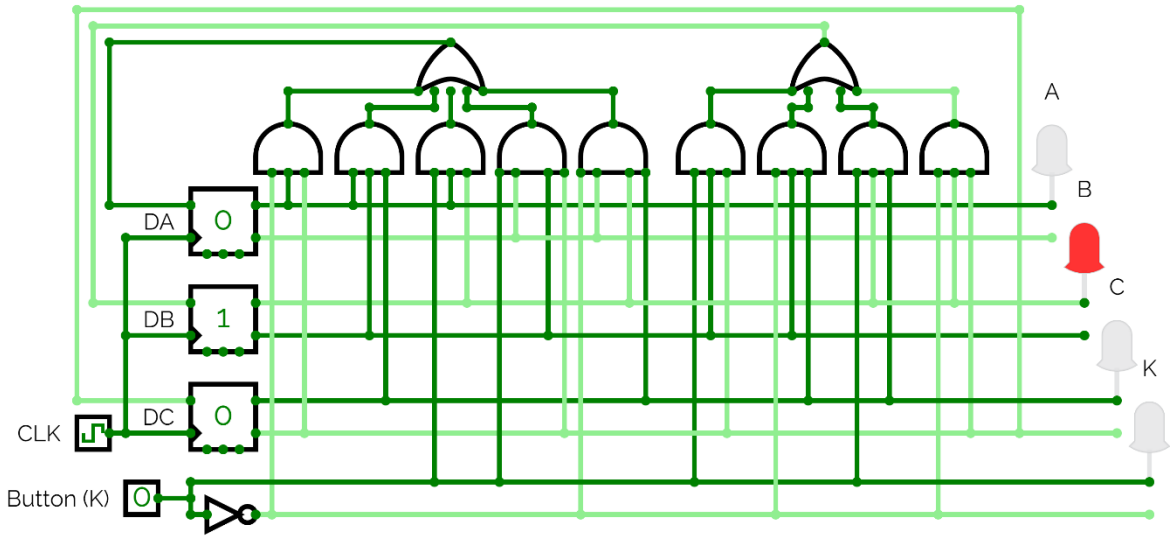
Now that we have the design of our digital circuit it is time to simulate it using software and test its functionality.



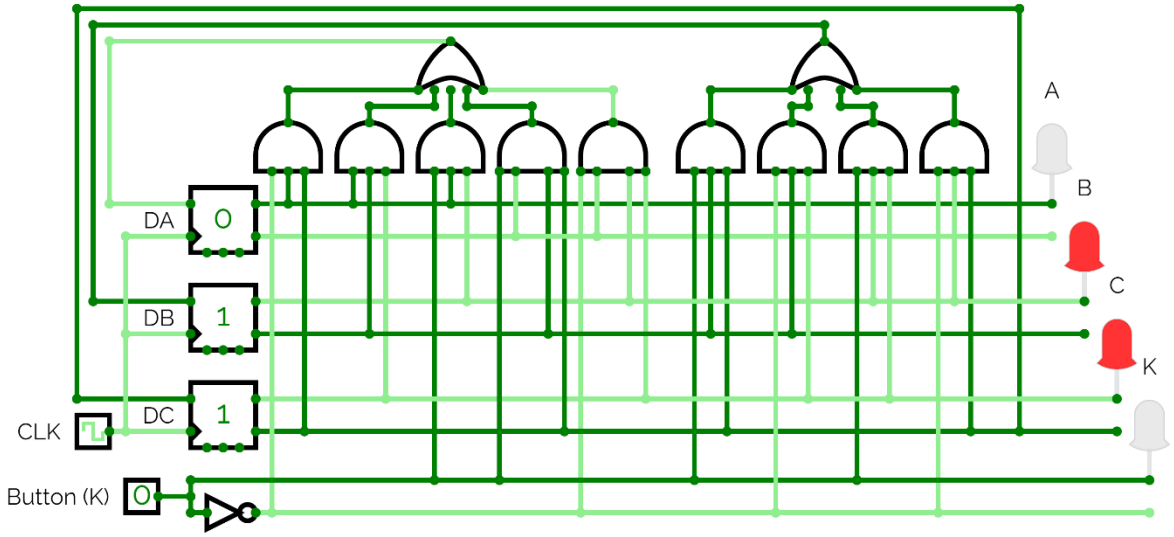

*State 000 K = 0*



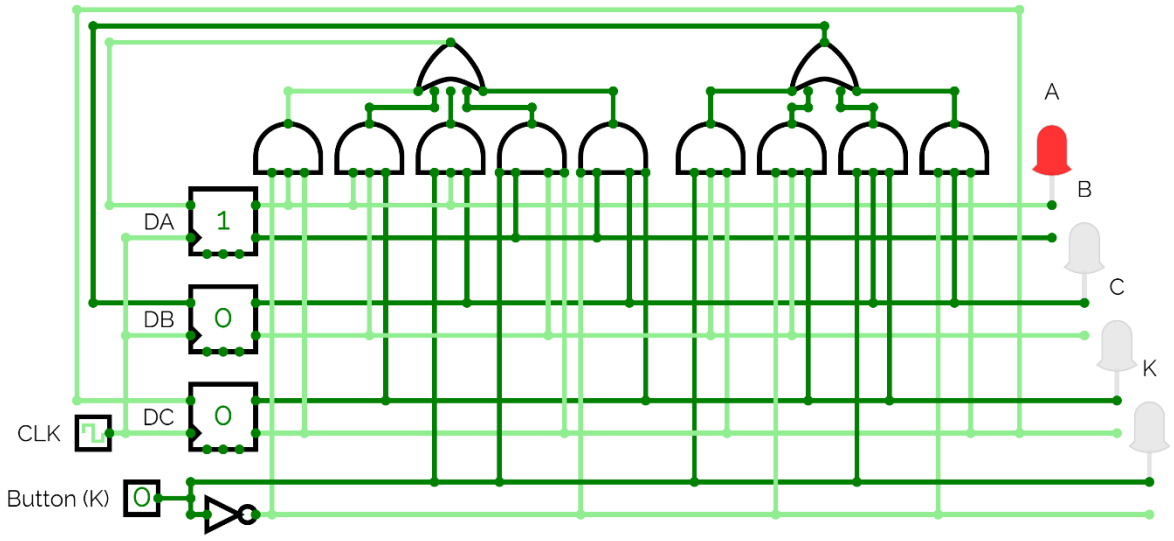

*State 001 K = 0*



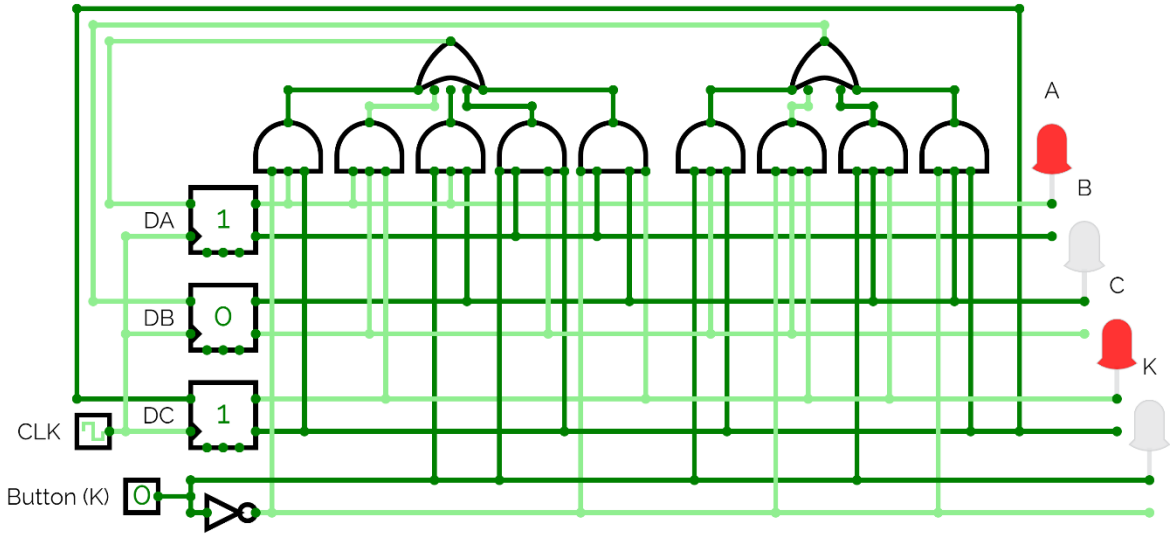

*State 010 K = 0*



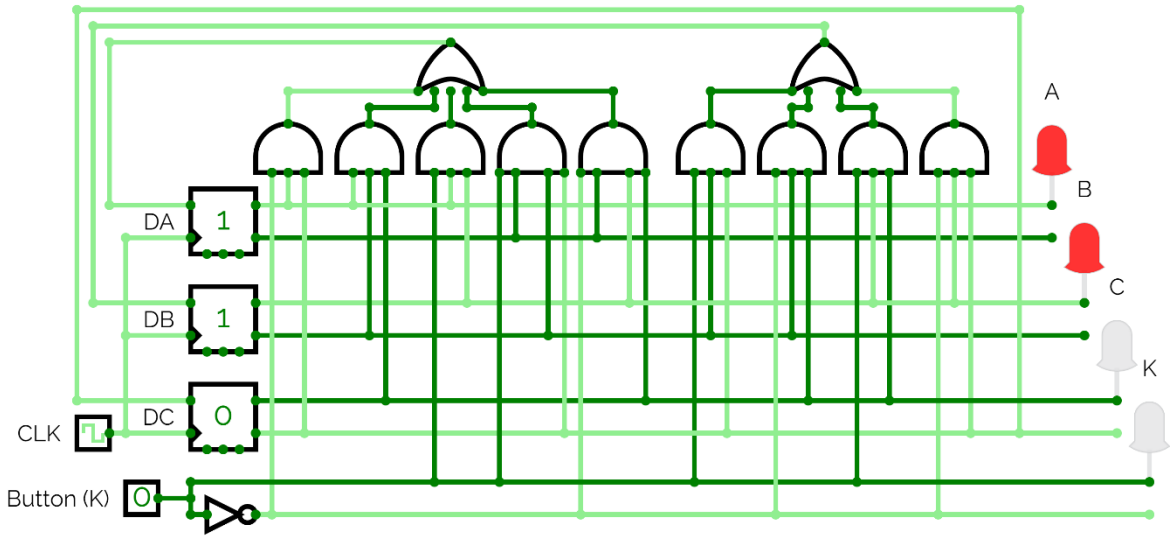

*State 011 K = 0*



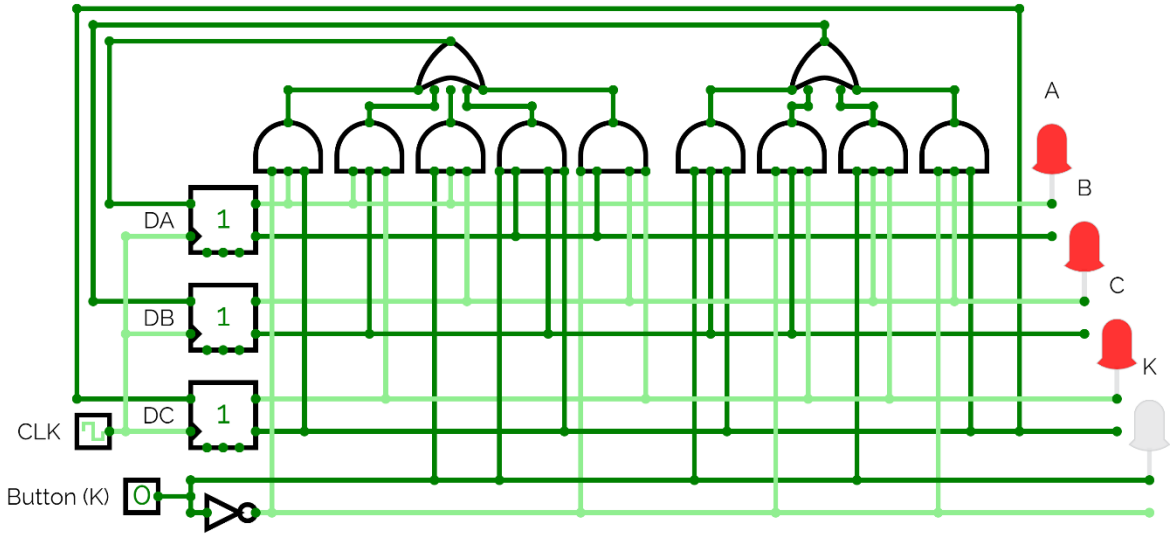

*State 100 K = 0*



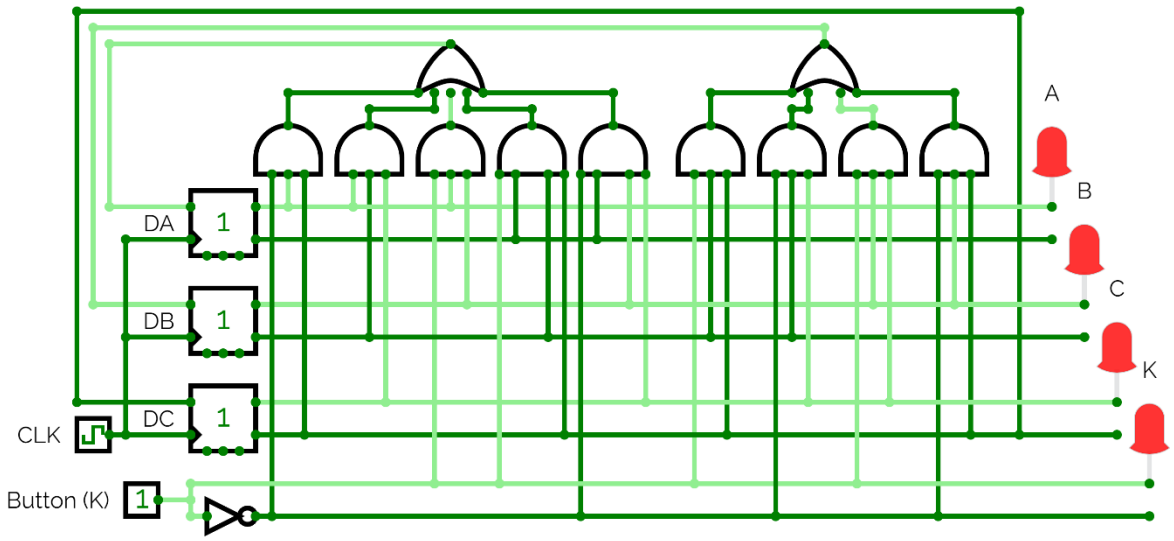

*State 101 K = 0*



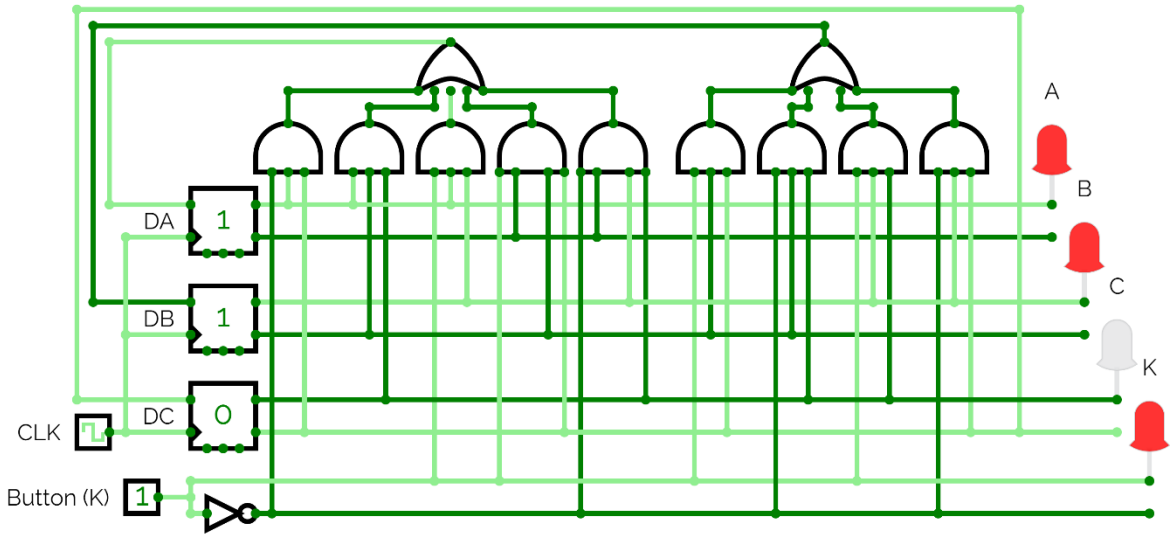

*State 110 K = 0*



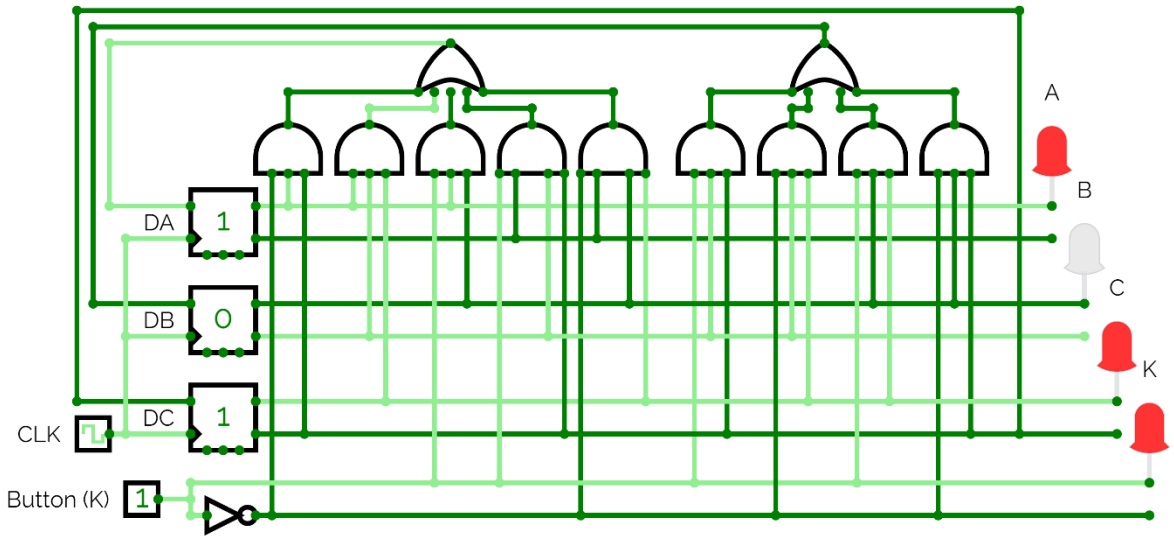

*State 111 K = 0*



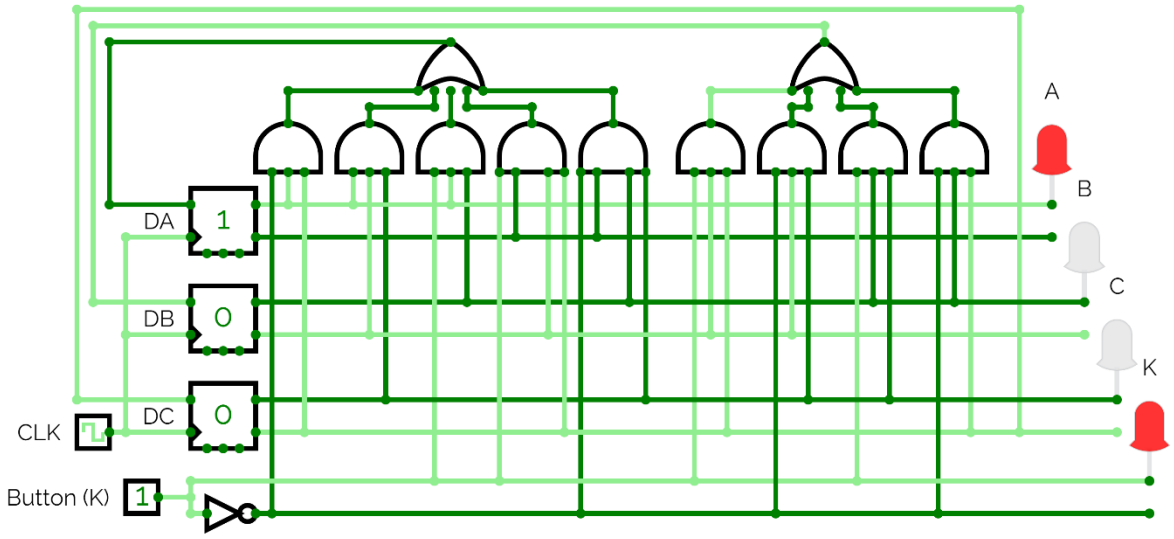

*State 111 K = 1*



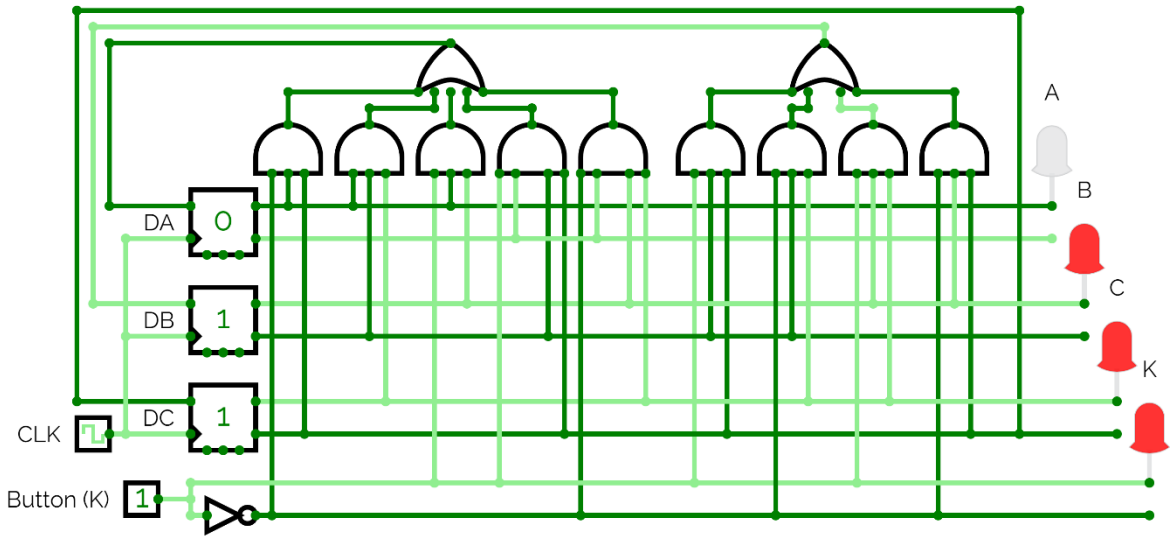

*State 110 K = 1*



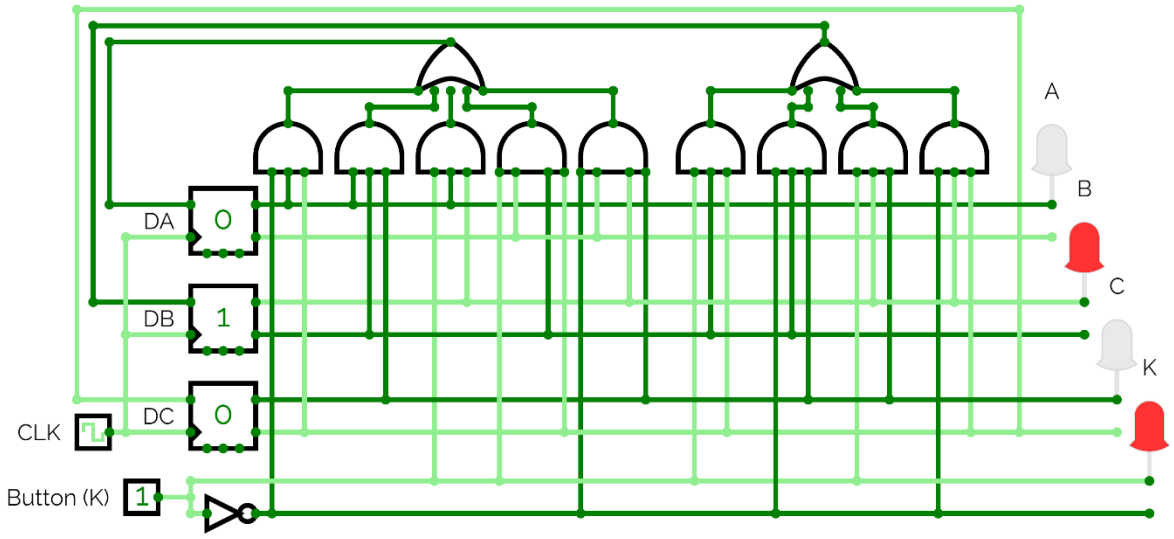

*State 101 K = 1*



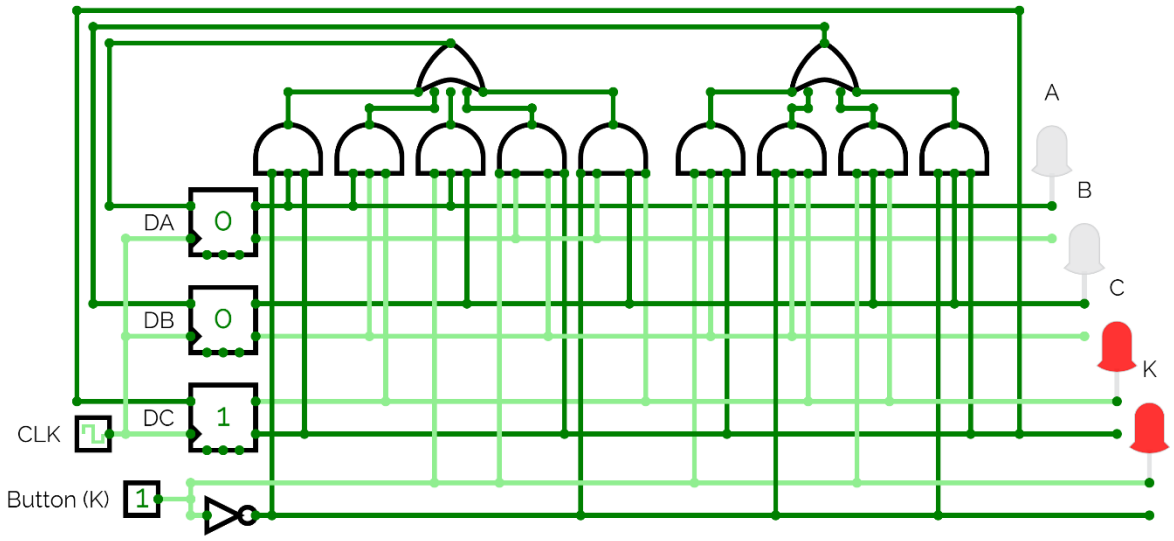

*State 100 K = 1*



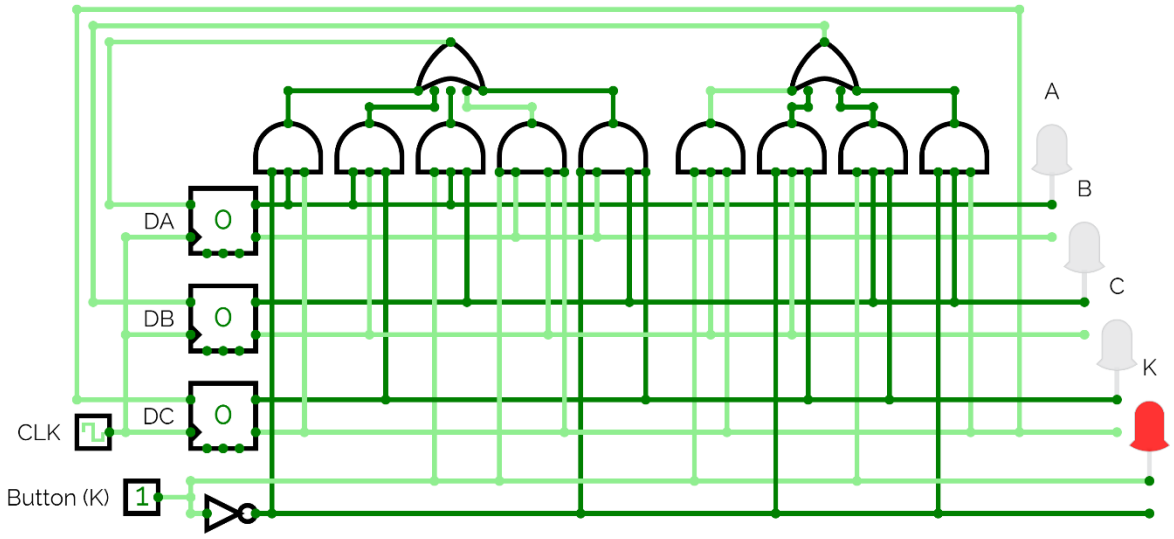

*State 011 K = 1*




*State 010 K = 1*




*State 001 K = 1*




*State 000 K = 1*

We have successfully tested all different possible combinations of the circuit and we conclude that the circuit is fully functional.

# 4. Specification

Now that we have a fully functional circuit it is time to prepare the specification for it, more specifically, what items are needed to practically implement this circuit.

The following is a list of items that are required to create this circuit:

Gates:

- 3-input AND gates x7
- 4-input AND gates x2
- 4-input OR gate x1
- 5-input OR gate x1
- **2-input OR gate x7 (when simplified)**
- NOT gate x1

Flip flops:

- D-flip-flop x3

Inputs / outputs:

- Timer x1
- Button x1
- LEDs x4

Others:

- Wires
- Power Supply
- Resistors
- Capacitors

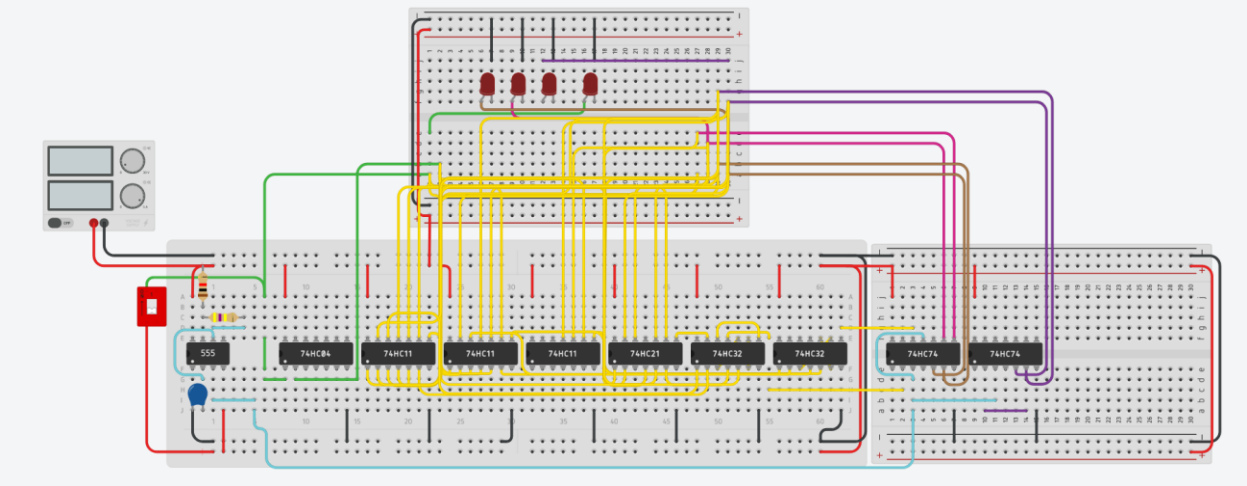
http://raven1.magix.net/List%20of%207400%20series%20integrated%20circuits.pdf
After taking a look at the website above, I found the ICs below:

**1x** Hex inverting buffer – 74HC04 / 744049
**3x** 3-input AND – 73HC11 / 7411
**1x** 4-input AND – 74HC21 / 7421
**2x** 2-input OR – 74HC32 /7432
**2x** D flip-flop – 74HC74 / 7474

# 5. Practical realization

The final step of this project is to do the practical realization which will be important for practically implementing this circuit.

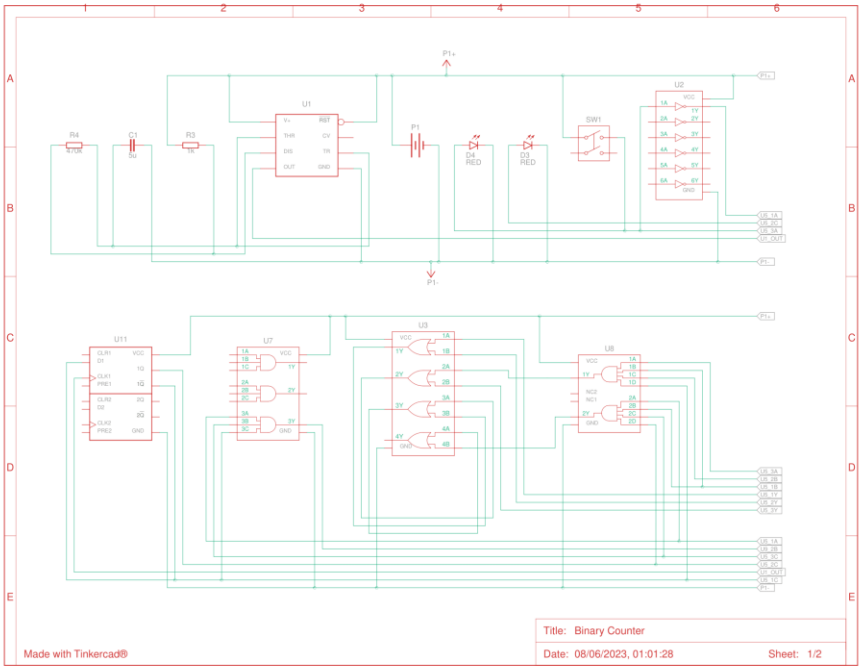
The following is a picture of the final circuit:

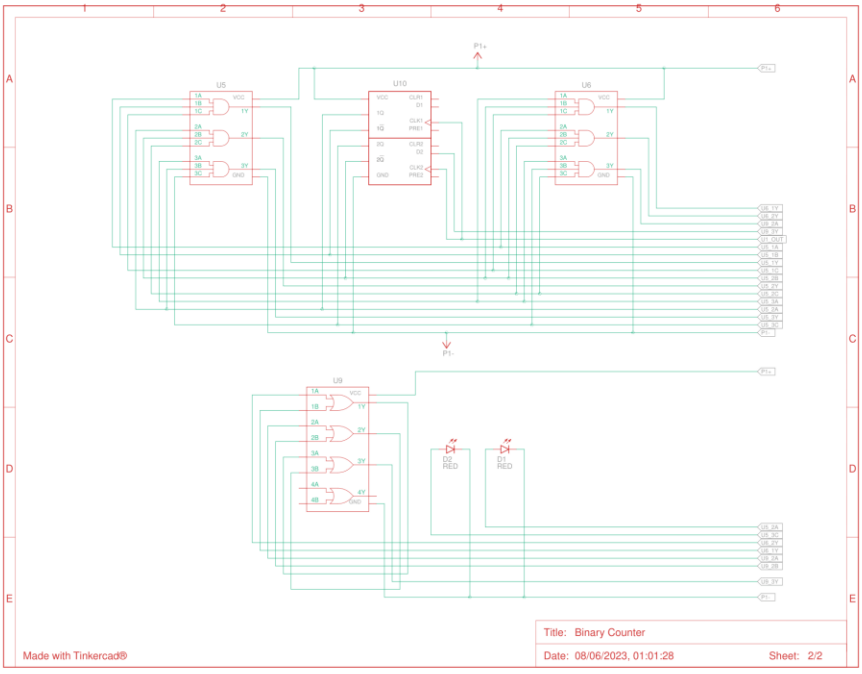



*The final circuit design*[5]

Last and perhaps most importantly, I have also added the schematics of this circuit on the last page of the documentation.

[5] I created this circuit using Tinkercad.


Title: Binary Counter
Date: 08/06/2023, 01:01:28
Sheet: 1/2
Made with Tinkercad®


Title: Binary Counter
Date: 08/06/2023, 01:01:28
Sheet: 2/2
Made with Tinkercad®