Introduction to Digital Systems

Project: Clock

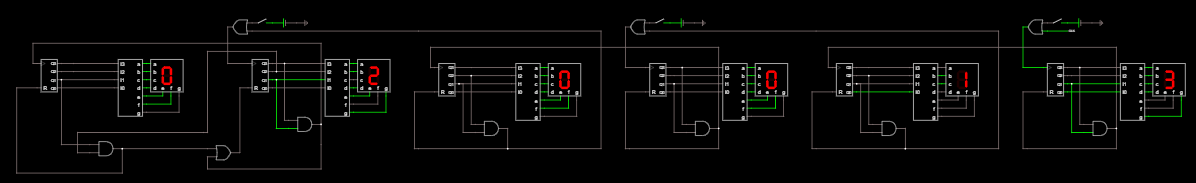
Name: Yusupov Yuldashbek

ID: 317304

Description

In this project, we are going to discuss how to make simple clock using falstad. Project, itself, represents six 7-LED displays, that will show time and can be adjusted, so briefly saying, we will make a true working clock. Using plugged in features and instruments of falstad, we are going to look through the implementation of a clock, which will use a 100Hz clock, which is heart of our clock, as an input and 7-LED displays, to show time and monitor several cases, as outputs. Simple algorithms behind our clock implementation, as well as its construction, will be described with truth tables and other graphical representations.

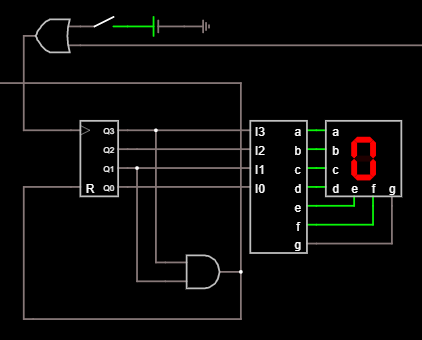
Clock picture



Close look up

of one

7-LED display



Detect 10

In this part, we will try to understand and implement algorithm that will count up to 10. It is used, because 7-LED display shows hexadecimal values, so in order for our clock not to show letters, we need to write some algorithm for it to stop counting after 9 and reset the 7 segment decoder.

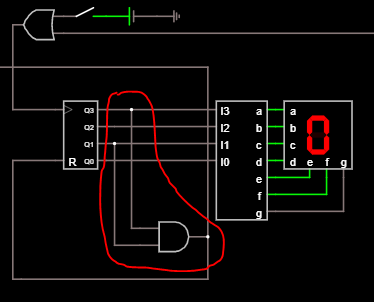
We have this truth table and now we need to make Karnaugh map for it

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | A | B | C | D | RES |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 0 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 1 |
| 11 | 1 | 0 | 1 | 1 | - |
| 12 | 1 | 1 | 0 | 0 | - |
| 13 | 1 | 1 | 0 | 1 | - |
| 14 | 1 | 1 | 1 | 0 | - |
| 15 | 1 | 1 | 1 | 1 | - |

Karnaugh map for above truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | - | - | - | - |
| 10 | 0 | 0 | - | 1 |

Using Karnaugh map, we may notice that there are 4 cells, painted red, which may be used for writing an algorithm for detecting number 10. Function is A\*C, or in digital systems, A and C.



Our detect 10 algorithm implemented in clock.

(Marked red)

Detect 6

After implementing our detect 10 algorithm, we may now proceed for writing another algorithm for detecting 6. As we know, seconds and minutes are counted to 60. Since, our first algorithm counts up to 10, not 60, we need to write new one, to detect 60. Fortunately, to represent seconds, minutes and hours, we need two 7-LED displays, which means we can only consider first 7-LED display and write function to it. Simply saying, we only need to detect number 6.

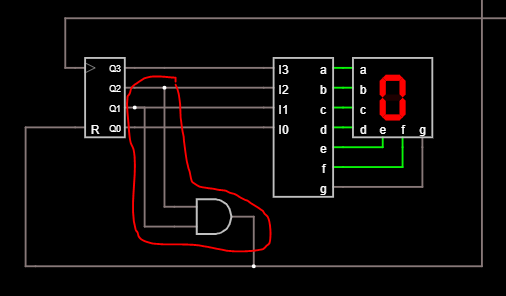
We have this truth table and now we need to make Karnaugh map for it

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| i | A | B | C | D | RES |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | - |
| 8 | 1 | 0 | 0 | 0 | - |
| 9 | 1 | 0 | 0 | 1 | - |
| 10 | 1 | 0 | 1 | 0 | - |
| 11 | 1 | 0 | 1 | 1 | - |
| 12 | 1 | 1 | 0 | 0 | - |
| 13 | 1 | 1 | 0 | 1 | - |
| 14 | 1 | 1 | 1 | 0 | - |
| 15 | 1 | 1 | 1 | 1 | - |

Karnaugh map for above truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | - | 1 |
| 11 | - | - | - | - |
| 10 | - | - | - | - |

Using Karnaugh map, we may notice that there are 2 cells, painted red, which may be used for writing an algorithm for detecting number 6. Function is B\*C, or in digital systems, B and C.



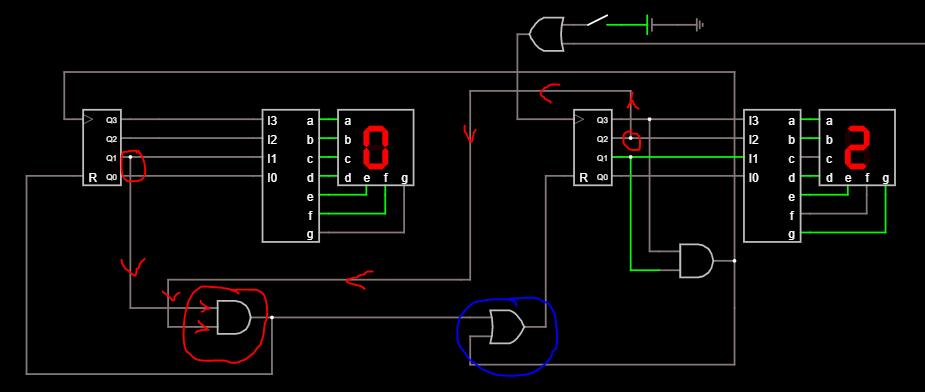
Our detect 6 algorithm implemented in clock.

(Marked red)

Detect 24

Now, after writing algorithms for detecting 10 and 6 and implementing them in seconds and minutes of our clock, we may now proceed to an algorithm for detecting 24 hours. We know that there are 24 hours in a day, however, our algorithms, written above, do not detect it, so we have to write one more.

This one is the easiest, since we know that 7 segment decoder uses 4 bits as outputs, and we know that each bit is 2 to some power. We can use that fact here. Two is 2 to the power 1 and four is 2 to the power 2, which means, that we only need these 2 bits to detect number 24. Finally, our algorithm will be B\*C, or in digital systems, B and C. However, we need to take C of the first 7 segment decoder and B of the second one of hours.



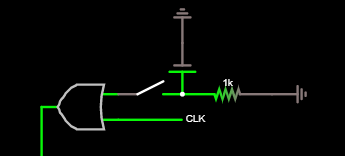
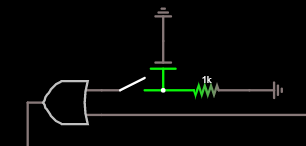
Here, we can see our detect 24 algorithm (marked red) with arrows of how the high state goes and activates AND gate. Moreover, you also may notice OR gate marked blue. It is used for our second 7-LED display, because this display should be reset in two cases:

1. When it detects 10
2. When it detects 24

Thus, we need OR gate there. Also our AND gate on the right is connected with input of the first 7 segment decoder, which means that every time it detects 10, the left decoder should assign high state and the number on display should be increased by 1. This kind of schematic is used all over the circuit and you can see it on the second page of this documentation.

Push Switch setter

In this part, we will discuss how to set time using push switchers as below.

Here, you may notice two types of push switchers. The idea behind it is very easy. To increment number on our display we need to make rising edge, which means that we should go from low state to hogh state. It can be done by switch or by clock. Since we need to set time with switch, we will use it.

We will use three things:

1. Ground
2. Switch
3. Voltage source

Voltage source is used for providing voltage, which is needed to go from low state to high state and vice versa. We connect one end of voltage source with ground and second one with switch. Another end of switch we connect with OR gate.

OR gate is used because we need to have rising edge rather from clock, previous decoder or switch. Input of the last decoder, which is used to represent tenth of seconds, we connect with switch and clock, because counting starts from seconds. Other inputs of decoders we connect with switch and algorithms of previous decoders. Example is on the second page.

At the end, we have fully and correctly working clock.

Extra task

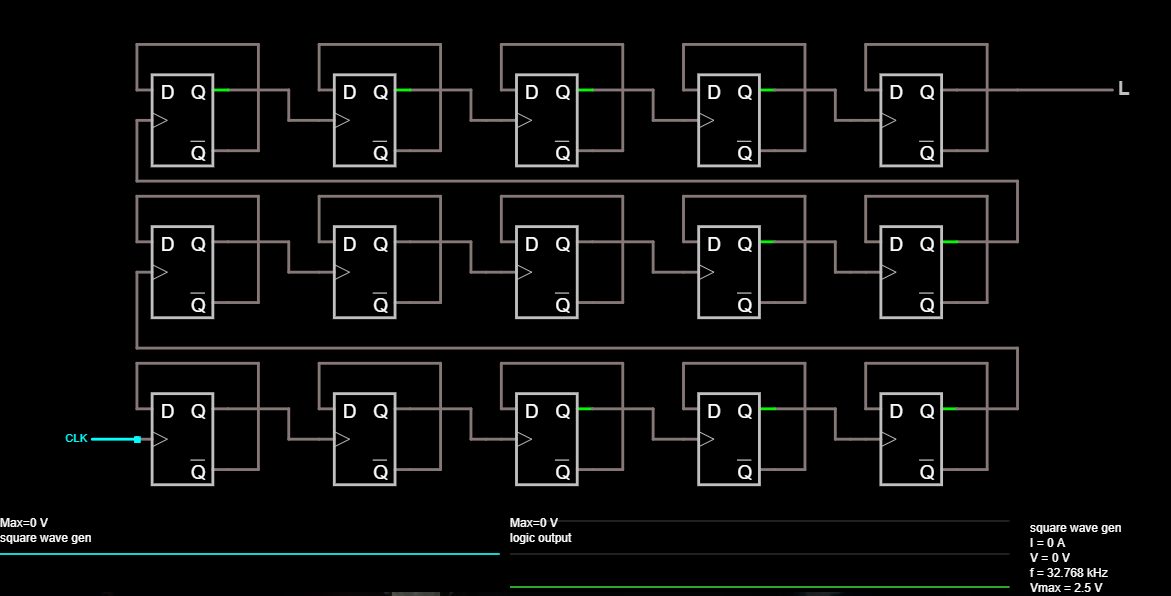
Designing circuit that will take 32kHz input and give 1Hz output.

First of all, I used clock and set it to 32.768kHz input, which is a lot. We need to create a circuit that will convert this 32kHz to 1 Hz. I used D-type flip-flops to solve this task.

As we know, D-type flip-flop decreases the frequency by 2. This is perfect for our case. Now, we only need to connect 15 D-type flip-flops in series and connect output.

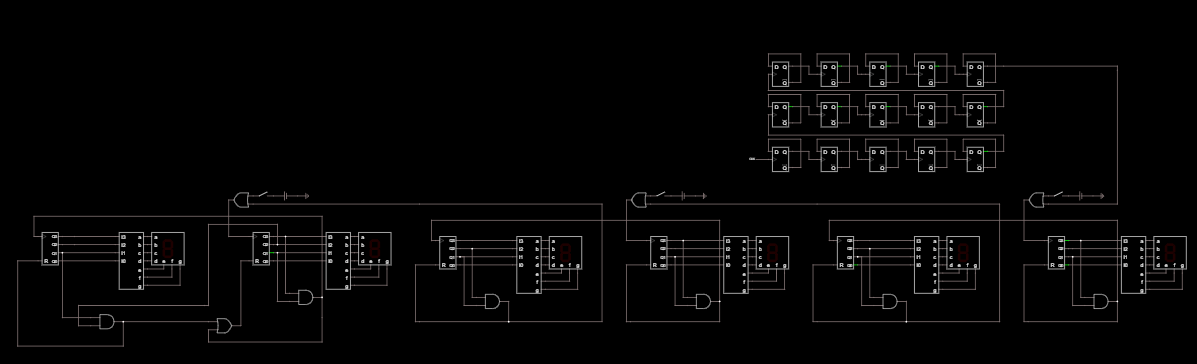
15 D-type flip-flops is used, because 32678Hz is 2 to the power 15, which means that we have to divide our input by two 15 times.

Final result looks like this:



As you see, each flip-flop is connected in series and output is 1Hz.

Now, we have to implement this solution to our clock:



Conclusion

At the end, I would like to say some words and leave credits. Project clock was very interesting and exciting laboratory task. Schematics and algorithms were nice to implement and, overall, I think that I managed to solve this task.

I would like to thank our professors, Porakowski Wiktor, Kalisiak Michał, Bartłomiej Kola and Tomasz Owczarek. Thank you!