Binary Clock - When 0's and 1's Tell Much More

Yevhenii Bevz

December 26, 2023

"**There are only 10 types of people: those who understand binary and those who don't.**"

1 Introduction

One always recognises the binary counting system, seeing a sequence of 0's and 1's, perhaps many struggle understanding it. Although the system itself is primitive in its nature, sometimes it seems to be challenging to explain how to count in binary, and why some specific sequence of two digits is a representation of a specific decimal number. Understanding how crucial the binary counting system is currently remains essential for anyone willing to dive deep into hardware, e.g. CPU's.

2 Project goal

The idea of the project is primitive–assist the students with understanding binaries. Having a real-life example of the binary counting system usage, might reduce the level of misunderstanding and provide enough for comprehending how to count using only zeros and ones.

3 Current status

As of December 26, 2023, both clocks' statuses may be described as **"almost finished"**. The *NodeMCU based* clock is capable of showing the actual time; DST is under question: requires additional attention and ESP8266 based alternative for Timezone.h library. The *GreenPAK based* clock is capable to counting from 0 to 12, perhaps the issue of fixed pin remains unresolved.

4 Known Issues

4.1 NodeMCU Based Clock

For the lack of practical skills, which were necessary for dealing with clock's hardware, the one known issues was never fixed: three LED-strips, responsible for showing 3 bits of minutes, continue blinking every time they receive an "active" signal. There is an assumption that the problem might be either in wires or in LED strips themselves.

However, in order not to spoil the project by breaking anything or tearing the taped wires, it has been decided to keep the current project status as it is.

4.2 GreenPAK Based Clock

After countless attempts, the solution for not-resettable pin has not been found yet. Currently the clock counts to 12 as expected, however after 12 the counting bounds change unwillingly to 5-12 instead of 1-12. Currently we maintain the experiments in order to fix the issue.

5 Detailed View

In this section we will take a deeper look in the principles of the binary clocks in order to fully comprehend their basics.

It is not difficult to get the general idea, as the clocks require the minimal knowledge of logic and electronics. It might be a little challenging for a complete newbie to organise the hardware, yet in general the project should not cause any trouble.

5.1 NodeMCU Based Clock

The main idea of this clock and also its most difficult part is to make the clock **show actual time without setting it manually**. Without this feature, the device is simply an Arduino counter which is of no great interest to us. With that said, the NTP (**N**etwork **T**ime **P**rotocol) came in handy, solving the problem almost completely. Speaking in general terms, the binary clock works in the following way:

1. The clock connects to the WiFi network using the **ESP8266** microchip;
2. The clock sends a request to NTP and receives current time according to the provided timezone;
3. The given time is converted into binary format and displayed using LED strips.

The only difficulty was to ensure that the DST (**D**aylight **S**aving **T**ime) was displayed correctly. Because the NTP call is what makes the clock show the

actual time, the request result does not take into account that some countries have the DST. That is why switching modes required to be handled in some specific way.

There was an idea of using the ***Timezone.h*** library, which works perfectly for most Arduino's, however it is not compatible with the ESP8266 we use in the clock.

5.2 GreenPAK Based Clock

Although we called this project a clock, the more appropriate name for this device would be "a clock-like counter", as it always starts counting from 00:00:00 to 12:59:59.

The idea of the project was to perform an experiment and find out whether it is possible to make the GreenPAK programmable device behave like an actual clock.

In this project we use only **GreenPAK SLG4662G** and Renesas' software **Go Configure Software Hub**, which we use for programming the device. Basically, all we need for the project is to know how to organise the logic in the way we would like our device to work.

The *"GreenPAK Clock"* is divided into two parts: *seconds* and *hour/minutes*. The *SLG46620G* has only 20 pins available, so in order to reduce the usage of pins, it was decided to program two separate devices instead of three (actually, it meant writing two different scripts instead of three).

On *Fig. 1* you can see a scheme for *SLG46620G* made via *Go Configure Software Hub* on the first matrix of the device, which counts from 0 to 9 and lights the corresponding LEDs on pins 7-10.

The main idea lies within the DFFs used in the design, which "remember" the signal until the needed time and then display it in the output. Also *Resets* have a crucial role in the design, as they prevent the clock-counter from counting further and reset everything to 0.

All the following designs have a similar nature, as only counting bounds differ, however an issue arose when it came to display hours and minutes *at the same time*.

The initial problem was the number of available pins on a single device. In general we need 19 pins for counting to 5, 9 and 24. It would be challenging to organise everything on a single device, so the goal has been changed to perform counting on separate devices.

With that decided, seconds were organised as it was demonstrated above. When it came to organising minutes and hours on a single device, another problem arose: as in seconds, two-digit numbers are displayed on separate matrices (that is the feature of *GreenPAK SLG46620G*), and that is why keeping things as they are might be inconvenient for users to spectate. With that said, it was decided to replace *24 hours* with *12 hours* which allowed us to avoid unnecessary matrices switching.

Here the last problem has been noticed, which significantly delayed the project. As you can see on figures, all DFFs we used have built-in **nRESETs**,

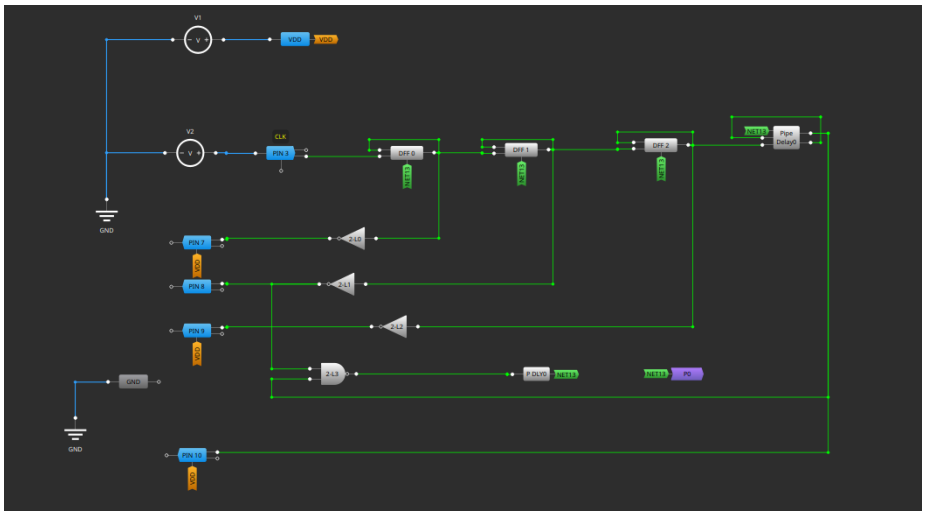

Figure 1: Seconds. Matrix 0

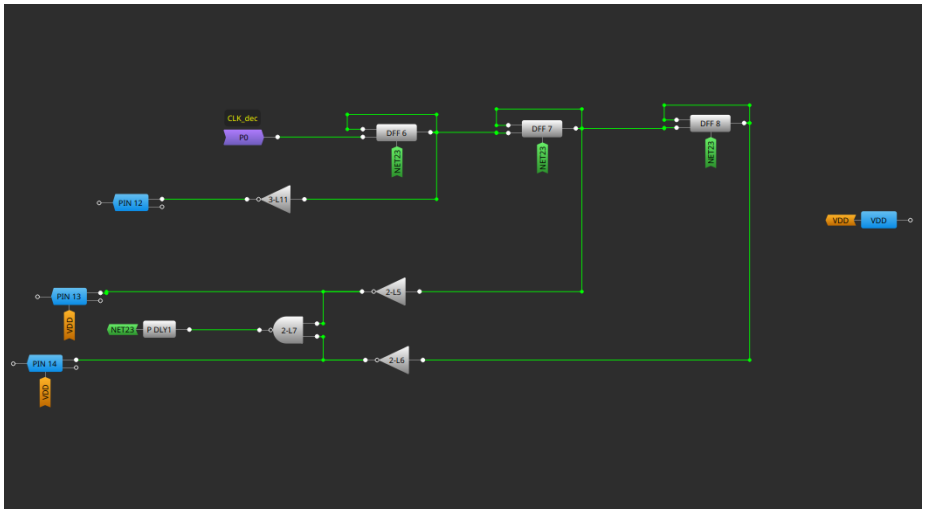

Figure 2: Seconds. Matrix 1

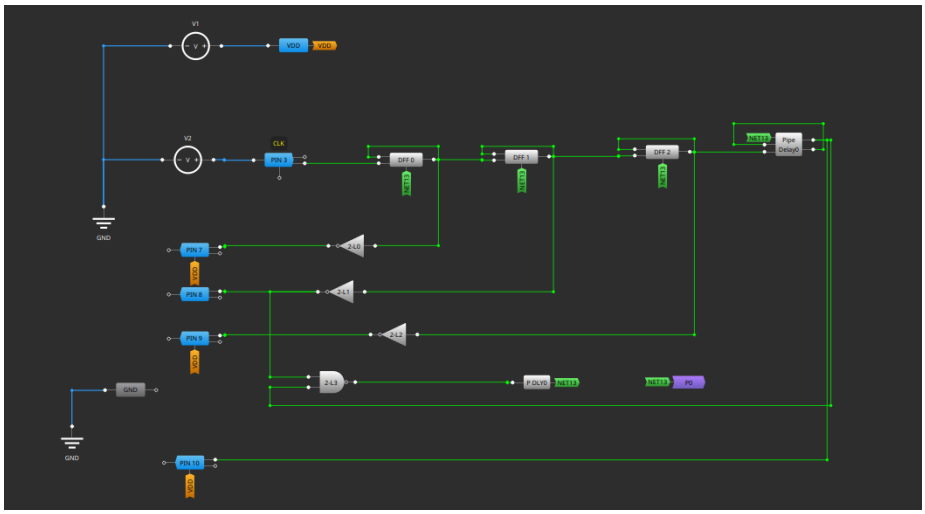

Figure 3: Minutes (0-5). Matrix 0

which we used to reset them when a specific combination of bits appeared. However, the *SLG46620G* has only 6 DFFs with nRESETs, and all of them had been used already. Below, you can see the design of minutes/hours when the problem has just been noticed (look at *hours* on *Matrix 1*).

Once you run the design, you might notice that **Pin18** does not reset after 12, while *Pin19* does. That is because the *DFF11* does not have the nRESET which we can use to directly reset the flip-flop, as we did with seconds or minutes. With that said, the only possible solution was to organise an nRESET for the given DFF ourselves.

6 Summary

The binary clock is indeed the best example to understand how the binary counting system works, and also we are able to make a conclusion that is not that difficult, if one pays more attention to it. Although it was a bit complex at first to understand everything, especially hardware, with enough time and practice even complete newbie may master at least basics.

Even though some issues currently remain unresolved, they are not impossible and might be fixed in the near future. This project has been a significant source of new worthy skills and experience, the results of which might be useful for people.

Both clocks–no need to deny the truth–have some flaws, perhaps they might bring benefits to APPS society, especially newcomers, who are unfamiliar with basic binaries.

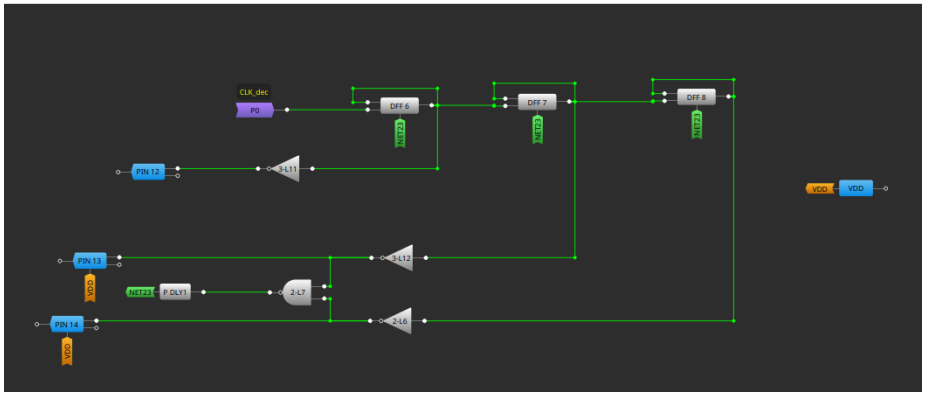

Figure 4: Minutes (0-9). Matrix 1

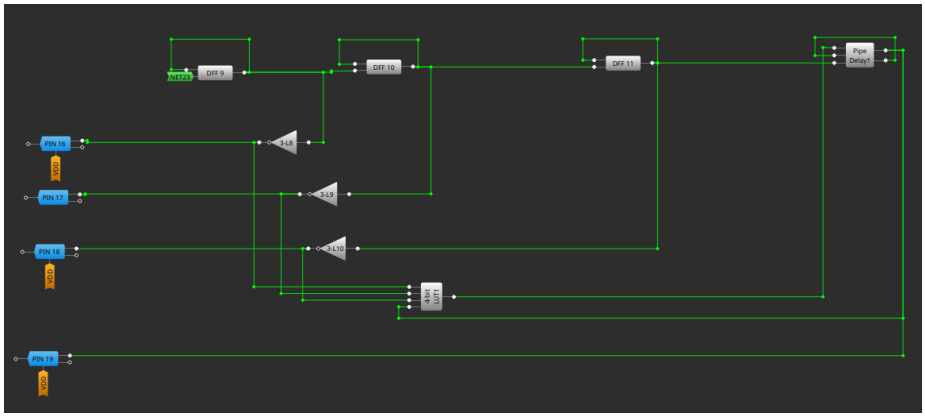

Figure 5: Hours. Matrix 1