Persistence of vision: LED Globe

Course: System On Chip DEsign [SOCLAB 1819]

Inleverdatum: [17/06/2019]

Student 1: **Jordi Jaspers**

Docent: **Ing. Luc Claesen** Groep: **EA – ICT 3**

Inhoudsopgave

[Verslag 2](#_Toc11437695)

[Introduction 2](#_Toc11437696)

[The build of the structure 3](#_Toc11437697)

[The Electrical Schematics 4](#_Toc11437698)

[Block Diagram & Verilog Code 6](#_Toc11437699)

[Parts List 7](#_Toc11437700)

[Hardware 7](#_Toc11437701)

[Software 7](#_Toc11437702)

# Verslag

## Introduction

The course “System on chip design” or SOCLAB gave us the option to create a project around a FPGA (=Field Programable Gate Array). Because it had to work with an FPGA. It was only natural we had to use the full parallel functionality it gives, unlike the Arduino that only uses serial functionality. So, for this project I chose to make a persistence of vision globe. Because LED’s are quite fun to work with and also because it is fancier than a persistence of vision fan.

Persistance of vision or POV is an optical illusion that occurs when visual perception of an object does not cease for some time after the rays of light proceeding from it have ceased to enter the eye. It is the same way a cartoon flipbook works or old projector in the year ca. 1900. Because our brain is to slow to process this information whenever light strikes the retina, the brain retains the impression of that light for about a 10th to a 15th of a second (depending on the brightness of the image, retinal field of view, and color) after the source of that light is removed from sight. This is due to a prolonged chemical reaction. As a result, the eye cannot clearly distinguish changes in light that occur faster than this retention period. The changes either go unnoticed or they appear to be one continuous picture to the human observer.

|  |
| --- |
| **Figure 1.1: Principle of POV** |

Applying this vast principle with LED’s on a fast-rotating arm, we could create images. But in our project instead of using an arm. We are using a circle that spins around its own axis. If it spins fast enough, we eventually will create a light emitting globe. This globe will then be capable of creating images.

|  |
| --- |
| **Figure 1.2: POV-Globe** |

## The build of the structure

Before we build the actual structure, we’ll need to make draw an overall plan, a blueprint of the structure. Also write down the steps of how you’re going to approach this build. This is very important, because if the structure is not stable enough the vibration will end up breaking your project.

|  |
| --- |
| **Figure 1.3: blueprint of the structure** |

The only thing left is to get all the part we need. I’ll refer to the parts-list at the end of this report. Now that we have gathered all the supplies we need, we need to cut the plates to the right size. When that’s done. Mark down and drill all the necessary holes. Afterwards clean and paint the metal part to prevent rust from ever showing up again. This is the first section. Next up, we have to create the two 3D-printed parts that are show in Figure 1.3.

I used the “Solidworks” software to draw the objects. The ring is going to holde the LED-strip and the Hall Sensor. So, when drawing the ring, I made it as big as possible. This will maximise the amount of LED’s that can stick to the ring while getting the hall sensor as close as possible to the magnet. The Cuboid on the other hand must only be capable of carrying the FPGA and the battery. Both have an 8mm hole to fit the rod that spins these objects.

The last step is mounting all the parts together and look for stability flaws. When found, make sure to finetune it. Like mentioned before the vibrations will otherwise break down the project.

|  |  |
| --- | --- |
| **Figure 1.4: 3D ring** | **Figure 1.5: 3D-Cuboid** |

## The Electrical Schematics

Before drawing the electrical schematics, we need to do some research about the FPGA. This is a required and necessary step. By looking at these datasheets we’ll see that GPIO\_00 and GPIO\_01 can both be used for input and output. So, for our input we’ll choose GPIO\_00 or Port 2. For our output we’ll use GPIO\_01 or port 4 on the schematic. The datasheet also tells us that port 12 is common ground on the FPGA. This means that our signals will need to use this ground pin in order to percept the signals.

|  |
| --- |
| **Figure 1.6: FPGA Datasheet** |

Second of all there is one more datasheet we need to look at before drawing the schematic. That is the datasheet of the hall sensor. We are using a 3144 switched Hall sensor. This means that every time the magnetic threshold is exceeded, the sensor switches on. Otherwise it will turn off. Our model is NC or normal closed-switched sensor. Its regular state is a high-output state and when the threshold exceeds it finds itself in a low-output state.

|  |
| --- |
| **Figure 1.7: Datasheet Hall sensor** |

Now we know everything about our components it’s time to draw the electrical scheme. As source we use a 3.7V 1200mah Li-ion battery, because they are rechargeable. To make them rechargeable we’ll connect it to a TP4056 circuit. This circuits helps as a BMS or battery management system. A system that prevents short-circuits and overloads. Plus, it had the ability to recharge a battery via micro USB. From here we connect the battery to a step-up converter that brings up the voltage to 5V. this is the minimum voltage required to power the FPGA and LED-strip together. Naturally the next thing we connect is the FPGA, the LED-strip and the hall-sensor in parallel. The only thing left is to connect the data-in port of the LEDs to the GPIO\_00 pin and the hall-sensor to the GPIO\_01 pin. This gets us the full schematic.

|  |
| --- |
| **Figure 1.9: Electrical scheme** |

## Block Diagram & Verilog Code

In figure 1.10 you’ll see the finished block diagram of the Verilog code that was put together for the FPGA. It really looks quite monstrous, but there actually just some key components that matter. The rest are easy counters and if statement for the loop. The 2 important blocks are the Infinite state machines and of course the memory where you can store all the colour and the image that you would like to display.

|  |
| --- |
| **Figure 1.10: Block Diagram** |

Furthermore,

## Demonstration (Users guide)

1. Open Quartus Prime
2. Choose a MODE 1-5
3. Compile
4. Upload the file in FPGA
5. Turn on battery
6. Take out the USB
7. Done!

# Parts List

## Hardware

* 1x DE0 FPGA NANO
* 1 x WS2812b LED Strip
* 1x 3144A Hall effect sensor
* 1x LiPo Battery (1200mAh)
* 1x TP4056 Protect/Charge circuit
* 1x Boost Converter
* 1x Ball Bearing
* 1x 12V 3000RPM DC Motor
* 1x 8mm Adapter
* 1x 200x200x20mm Steel base plate
* 3x 40x500x4mm flat steel
* M4, M5 bolts & nuts
* 8mm aluminum rod
* 3D-Printed Ring
* 3D-Printed Cuboid

## Software

* Quartus Prime
* System Builder for FPGA
* 3D printing software