ENSC 254

Summer 2018

Lab 4

Shine a light...

Deadline:June 18th, 2018
Last revision: June 5, 2018

Karol Swietlicki
Simon Fraser University

1 Hello, hardware!

Starting from this lab we finally get our hands on the hardware. We will also study how humans perceive light.

The deliverable for this lab is a written report.

2 Board verification

Take out the FPGA board, the power supply and one USB cable from their box. Plug in one end of the power supply into a wall socket and the other end into the power jack of the FPGA. Flip the power switch to the left. A light should turn on. If the light does not go on, let a TA know.

3 Programming the board

Download the zipped project from Canvas. Turn on the Xilinx XSDK. Import the project into the default workspace. You need to import it as "project from an folder or archive". Importing an "archive" won't work. You only need to edit ASM_LED/src/ASM.S

Insert one end of the USB cable into your computer and the other end into the "PROG" connector on the FPGA. There are three connectors on the FPGA into which the USB cable is physically able to fit. Make sure you got the right one, it is right by the power jack.

Build the imported project and then run it in debug mode, using either GDB or the system debugger on hardware. (Avoid the QEMU option.) The OLED screen on the bottom of the board should be blank for a long time and then report a version number: v4.6. If this is not the case, your board is not being configured properly. You should also be thrown into a debugger, debugging the main function. This will happen by default, even if you haven't set a breakpoint in main. You need to go into the debug configurations options menu and arrange that the board is reset and "programmed" when you start to debug. This is a matter of ticking two checkboxes with the relevant functionality.

Note that the XSDK is a bit... temperamental at times. Make sure your files are getting saved properly. If debugging stops working, power cycle the board. If the problem persists, power cycle your desktop.

Make sure you can step through your code. Once you can, proceed to the next section.

4 Simple blinking

LEDs are controlled by 32-bit register writes to the address 0x41210000. Experiment with different written values to figure out which LEDs correspond to which bit in the memory location.

Write code that toggles the leftmost LED every second. If the LED is on, switch it off. If the LED is off, switch it on. Do this once every second, forever.

This will likely take form of a loop which runs very many iterations. It is up to you to construct this delay loop. Use a stopwatch and try several programs to get this timing as close to a second as possible.

To help you fine-tune the loop without a need to recompile, you can use the buttons and the sliding switches to your benefit.

The state of the buttons is visible through reads from the address 0x41200000. Each button has a dedicated bit in the result of such a read.

And finally, the eight sliding switches below the LEDs are available via reads from address 0x41220000. This is analogous to the buttons, just at a different location.

In both cases, you can use the debugger to see what the obtained values are, given a certain state of the machine.

Please note that the buttons exhibit a behavior known as bouncing. The phenomenon is widely documented. Feel free to research it, if you wish to use buttons. We will deal with it in a future lab.

5 Duty cycles

The duty cycle is the percentage of time the LED is lit.

In the previous section you have lit the LEDs with a signal which has a duty cycle of 50% and which has a period of two seconds.

Create a signal which has the LED be lit for four seconds, following by one second of darkness. This signal has a duty cycle of 80% and a period of five seconds.

Also create a signal which has a duty cycle of 5% and a period of 10 seconds. In other words, the LED should spend 500ms lit and 9500ms unlit.

6 Persistence of vision

Using a duty cycle of 50%, find the frequency at which the LED appears steadily lit due to it blinking too fast for the human eye to notice.

How does changing the duty cycle affect this frequency? Create a graph which plots the effect of the changing duty cycle on the frequency required for the blinking to be too fast to see.

Take note of the fact that with the reduction in duty cycle the LED becomes visually dimmer.

7 Intensity of light

Use a signal which has a frequency high enough not to cause visible blinking at any duty cycle.

Assume that a fully lit LED has a brightness of 1. Assume that a completely dark LED has a brightness of 0.

Take a look at how various duty cycles affect the apparent/visible brightness of the LED.

Which is the first duty cycle that causes the LED to be visibly completely lit?

Which duty cycle produces brightness visually half-way between the two extremes?

Which duty cycle range is impossible to tell apart from the LED being fully lit, if any?

Create a comprehensive plot which demonstrates how the visible brightness changes across the entire range of the duty cycles possible.

Write a report which demonstrates all your findings. The length is yours to pick, though you are expected to be very detailed in your explanations. Submit your report to CourSys in any modern office format, as well as a plain text file.

Careful: Please use your unaided eye for this series of experiments. A camera will likely distort the results.