Research 1-71 EE Laboratory

CO-521-B Digital Signal Processing Lab

Spring 2023

Friday, 10^{th} of April 2023

LAB 5: Digital_Down_Converter

Report by: Prince Lee Muhera

Introduction

In this lab we use a simulation program developed by Xilinx to program an FPGA board, simulate hardware design, and obtain an understanding of how to program an FPGA board and how everything works behind the scenes. We are provided a .vhd and a .ucf file which contain the downloadable code for the FPGA board with specified instructions. However, the code in these files is incomplete. In lab, we load the files in Xilinx, correct the code in the .ucf file that we use for the Spartan 3 FPGA Board, download the code into the FPGA board and check the results. Then, we develop the simulation of the FPGA board and learn to switch things around in the circuit board simulation. We also learn to use this simulation to make changes into the physical FPGA hardware. Also, we take some time to understand how FPGAs implement logic functions.

Circuit Board Used

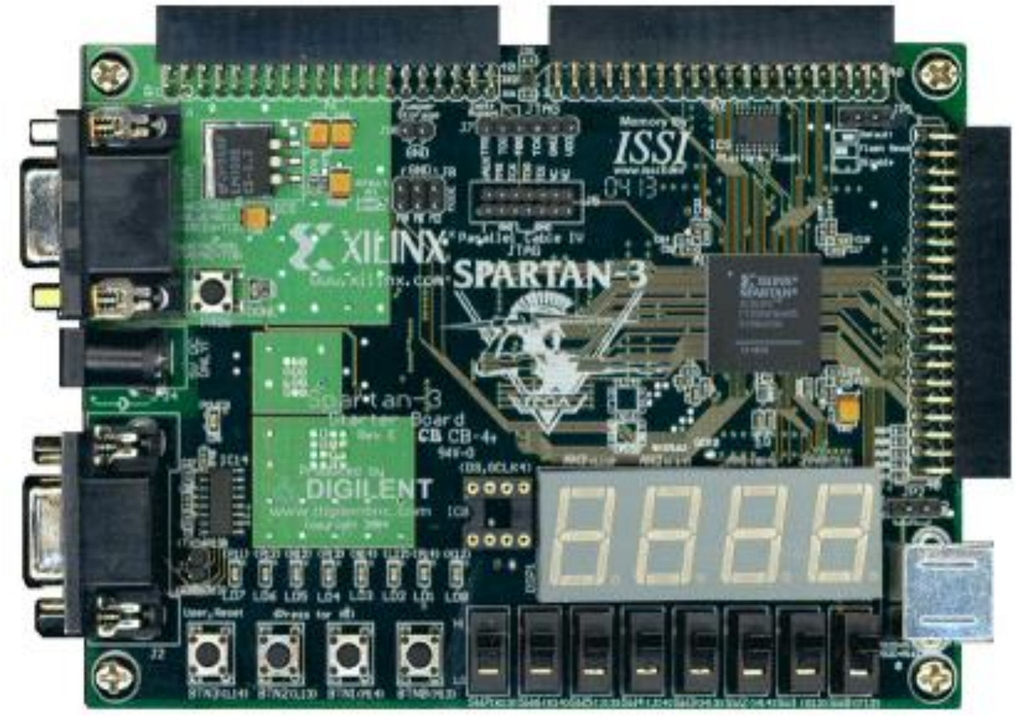


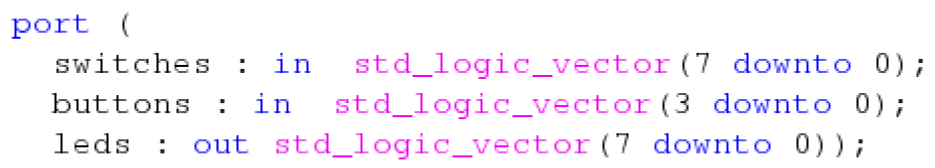
Figure 1: Spartan-3 FPGA Board

Design Files Used

Nr.	Name
1	XilinxIntro.ucf
2	Xilinx.vhd

Task Description

Firstly, we download the two files above from the course website. Then we create a project in the Project Navigator and load the downloaded files onto the workspace. When we open the .vhd file in the Xilinx editor, we find a code with a small error, in the following section:



```
port (
   switches : in  std_logic_vector(7 downto 0);
   buttons : in  std_logic_vector(3 downto 0);
   leds : out std_logic_vector(7 downto 0));
```

We change the third line on this section to the following:

“buttons : in std_logic_vector(1 downto 0);”

We do this because in our project we use only the buttons stated in the correction.

After editing, we are left with the following .vhd code:

```
library ieee;
use ieee.std_logic_1164.all;

entity XilinxIntro is
  port (
    switches : in  std_logic_vector(7 downto 0);
    buttons : in  std_logic_vector(1 downto 0);
    leds : out std_logic_vector(7 downto 0));

end XilinxIntro;

architecture simple of XilinxIntro is
begin  -- simple
  leds <= "00000000" when buttons(0)='1' else

          "11111111" when buttons(1)='1' else
          switches;

end simple;

architecture simple2 of XilinxIntro is
begin  -- simple2
  leds <= "11111111" when buttons(1)='1' else
          "00000000" when buttons(0)='1' else
          switches;
end simple2;
```

Next, we check out the constraints in the .ucf file. We find that this file is incomplete. After completing the code, we end up with the following:

```
#Connect ports to FPGA pins as specified in the Spartan3 starter board manual

NET "switches[0]" LOC = F12;
NET "switches[1]" LOC = G12;
NET "switches[2]" LOC = H14;
NET "switches[3]" LOC = H13;
NET "switches[4]" LOC = J14;
NET "switches[5]" LOC = J13;
NET "switches[6]" LOC = K14;
NET "switches[7]" LOC = K13;

NET "buttons[0]" LOC = M13;
NET "buttons[1]" LOC = M14;

NET "leds[0]" LOC = K12;
NET "leds[1]" LOC = P14;
NET "leds[2]" LOC = L12;
NET "leds[3]" LOC = N14;
NET "leds[4]" LOC = P13;
NET "leds[5]" LOC = N12;
NET "leds[6]" LOC = P12;
NET "leds[7]" LOC = P11;
```

We add the last 4 switches and the last 4 leds to the editor.

Now that the code is complete, we synthesize the code in the process window. Then, we run "Implement Design" and then "Generate Programming File". After that, we can run "Configure Target Device", which opens ISE iMPACT. In iMPACT, we do a boundary scan and obtain a map after it's done, provided the FPGA board is connected to the computer. Next, we right click on the left chip and click on "Program", and then select the generated .bit file from the window that appears, and select "BYPASS" in the next window. Now, the code is downloaded into the board. We can see the demonstration of this by flipping the switches. In the .ucf file, certain buttons and switches (input devices) are paired with the LEDs (output devices). As a result, when we flip the switches on or off on the board, the LEDs turn on and off. While the LEDs are turned on, if we press button <M13>, the LEDs turn off. While the LEDs are turned off, if we press button <M14>, they turn on. Pictures of demonstration can be found below:

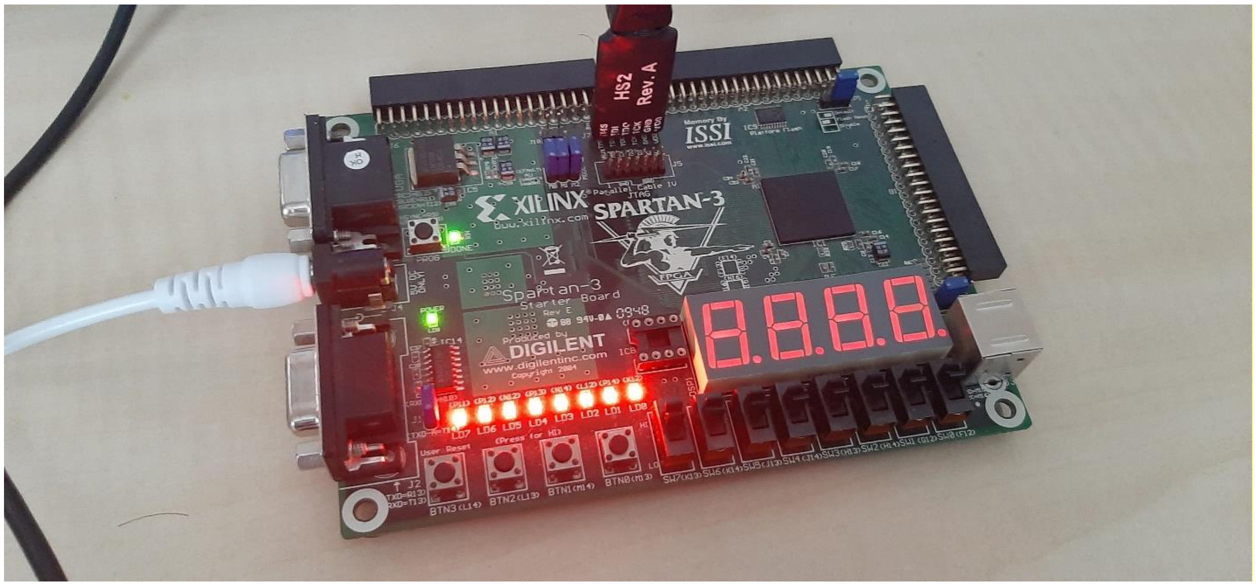


Figure 2: Programmed FPGA Board with all switches turned on.

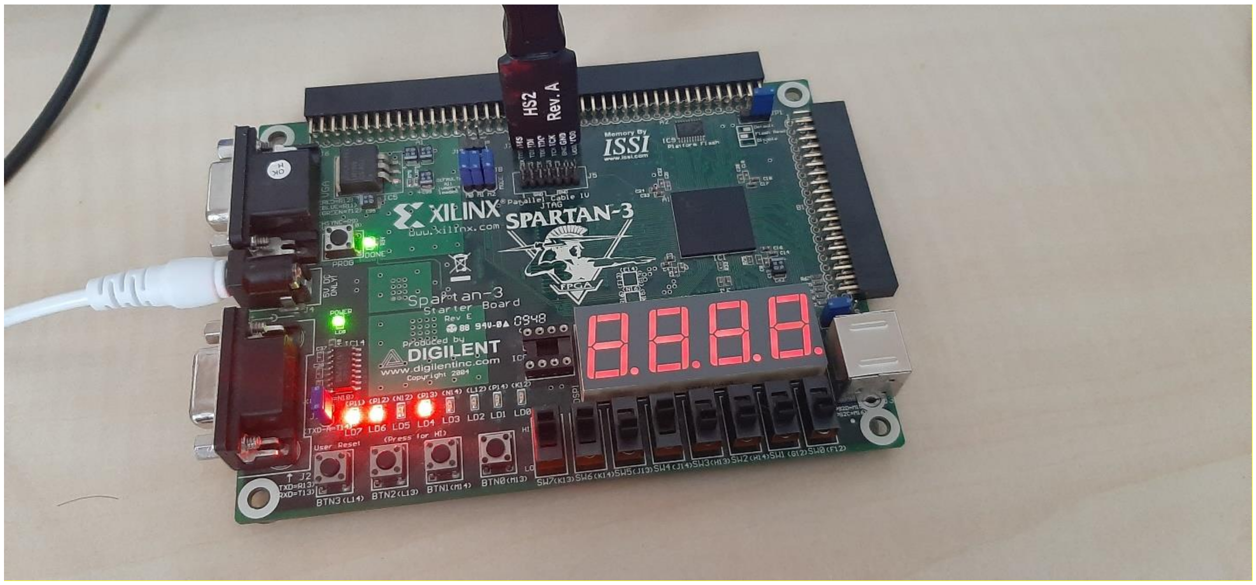


Figure 3: Programmed FPGA Board with some switches turned off.

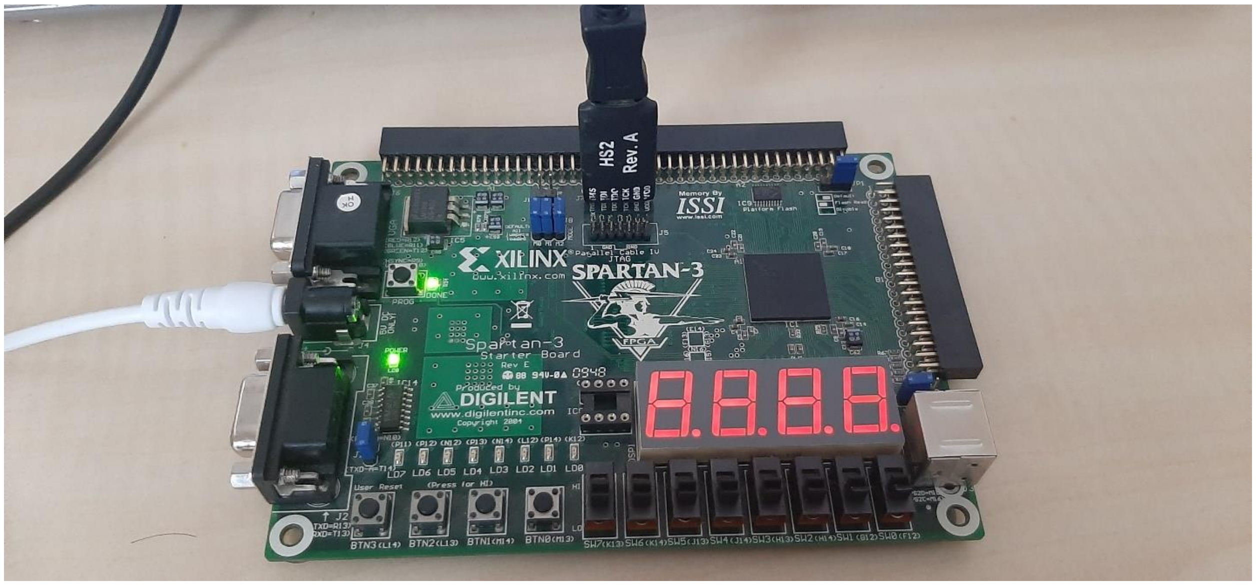


Figure 4: Programmed FPGA Board with all switches turned off.

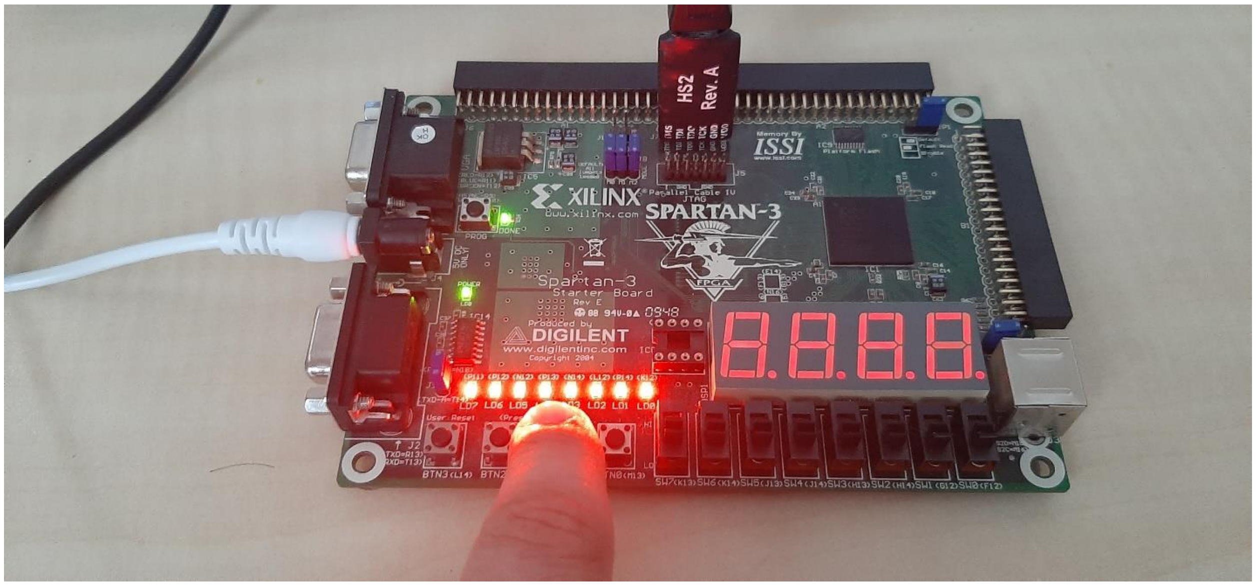


Figure 5: Programmed FPGA Board with all switches turned off and button <M14> pressed.

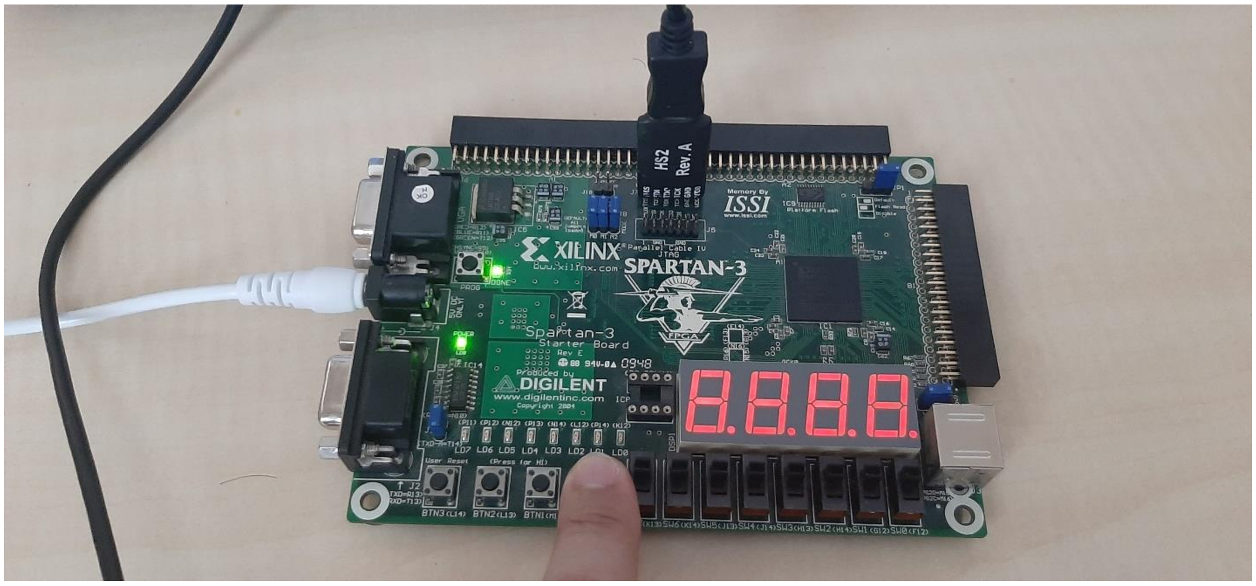
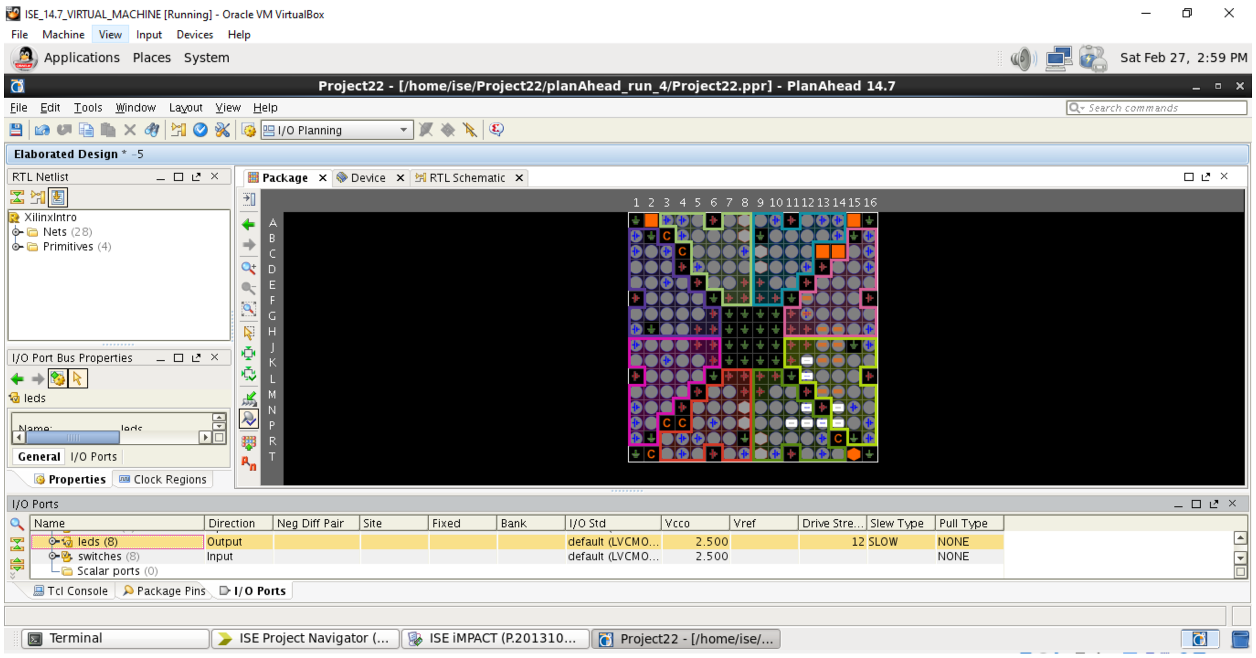


Figure 6: Programmed FPGA Board with all switches turned on and button <M14> pressed.

A link to a video demonstration can be found here: https://youtu.be/mkVpPtszFbk

When we expand "User Constraints" and run "I/O Pin Planning", we can open the planning of the circuit board design in "Plan Ahead". Here, we find the following design for the circuit board:


ISE_14.7_VIRTUAL_MACHINE [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Sat Feb 27, 2:59 PM
Project22 - [/home/ise/Project22/planAhead_run_4/Project22.ppr] - PlanAhead 14.7
File Edit Tools Window Layout View Help
I/O Planning
Elaborated Design * -5
RTL Netlist
XilinxIntro
Nets (28)
Primitives (4)
Package Device RTL Schematic
I/O Port Bus Properties
leds
Name: leds
General I/O Ports
Properties Clock Regions
I/O Ports
Tcl Console Package Pins I/O Ports
Terminal
ISE Project Navigator (...
ISE iMPACT (P.201310...
Project22 - [/home/ise/...

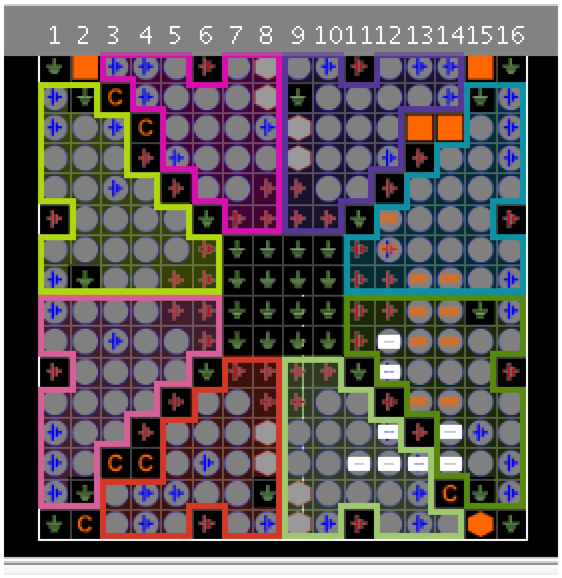
Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type
leds (8)	Output					default (LVCMO...	2.500		12	SLOW	NONE
switches (8)	Input					default (LVCMO...	2.500				NONE
Scalar ports (0)											

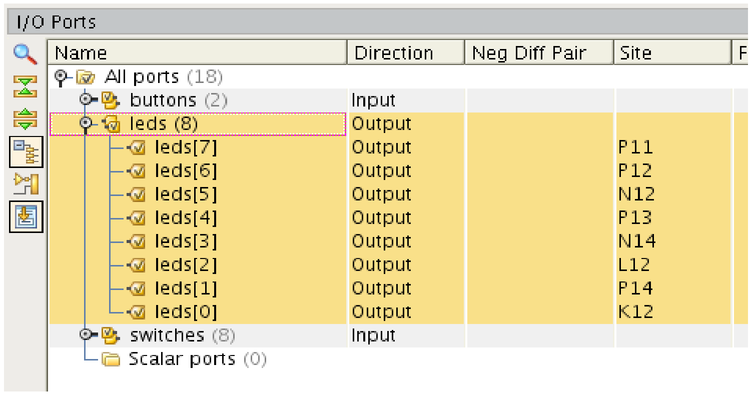
Figure 7: FPGA Board circuit design simulation.

Now we have a simulation of the circuit design. The selected pins are the LEDs on the circuit board. When we make changes to it and load them onto the board, we can see those changes on the actual physical board.

We can demonstrate this by switching the LED pins on both ends of the board, namely the <K12> and <P11> pins. To do this, we take pick up the <K12> pin in the simulation and put it in an empty space. Then, we put the <P11> pin in the place where the <K12> pin was, and replace the spot of the <P11> pin with the <K12> pin.

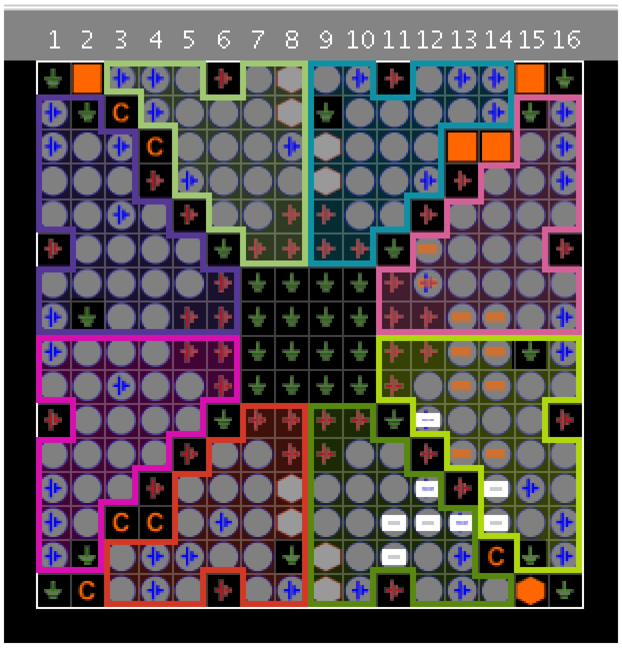
In the beginning, the board looks like this:




I/O Ports

Name	Direction	Neg Diff Pair	Site
All ports (18)			
buttons (2)	Input		
leds (8)	Output		
leds[7]	Output		P11
leds[6]	Output		P12
leds[5]	Output		N12
leds[4]	Output		P13
leds[3]	Output		N14
leds[2]	Output		L12
leds[1]	Output		P14
leds[0]	Output		K12
switches (8)	Input		
Scalar ports (0)			

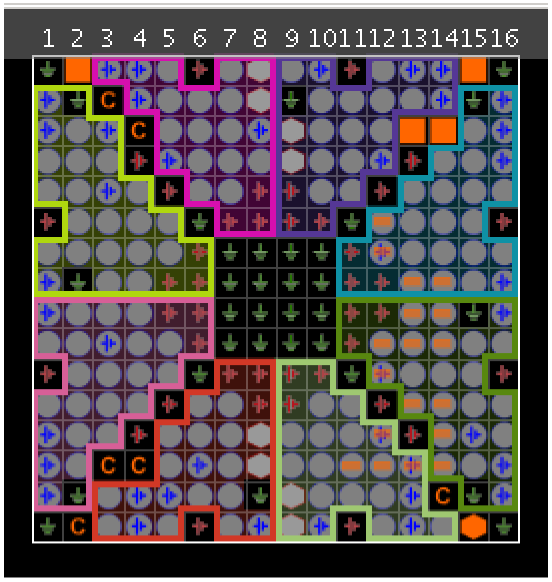
Figure 8: FPGA Board circuit design simulation without changes.

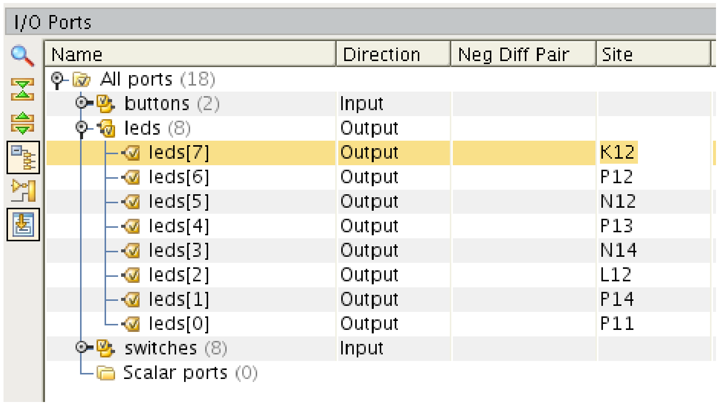


Picture Description: This picture is taken after the <P11> pin is removed and the <K12> pin is put in its place.

Figure 9: FPGA Board circuit design simulation with changes.

After making all the changes, we can see the following:




I/O Ports

Name	Direction	Neg Diff Pair	Site
All ports (18)			
buttons (2)	Input		
leds (8)	Output		
leds[7]	Output		K12
leds[6]	Output		P12
leds[5]	Output		N12
leds[4]	Output		P13
leds[3]	Output		N14
leds[2]	Output		L12
leds[1]	Output		P14
leds[0]	Output		P11
switches (8)	Input		
Scalar ports (0)			

Figure 10: FPGA Board circuit design simulation with <K12> and <P11> swapped.

After loading the changes onto the board, we can see that when we flip the <K13> switch, the <K12> LED lights up, and when the <F12> switch is flipped, the <P11> LED lights up, and vice versa. Pictorial demonstration is provided below:

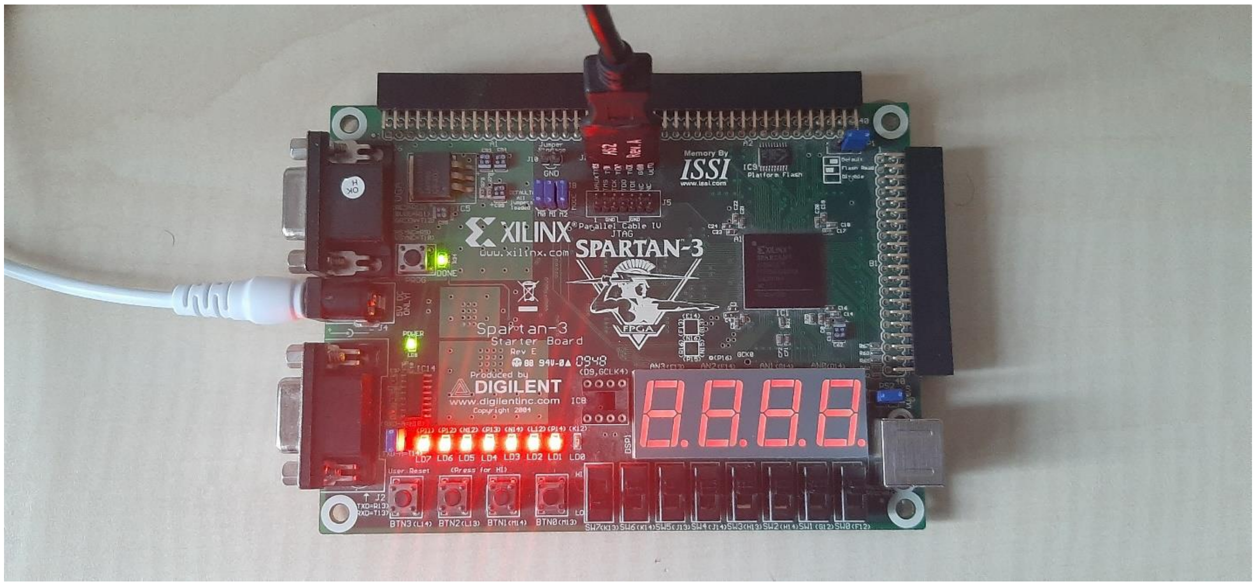


Figure 11: FPGA Board circuit with <K13> switch turned off.

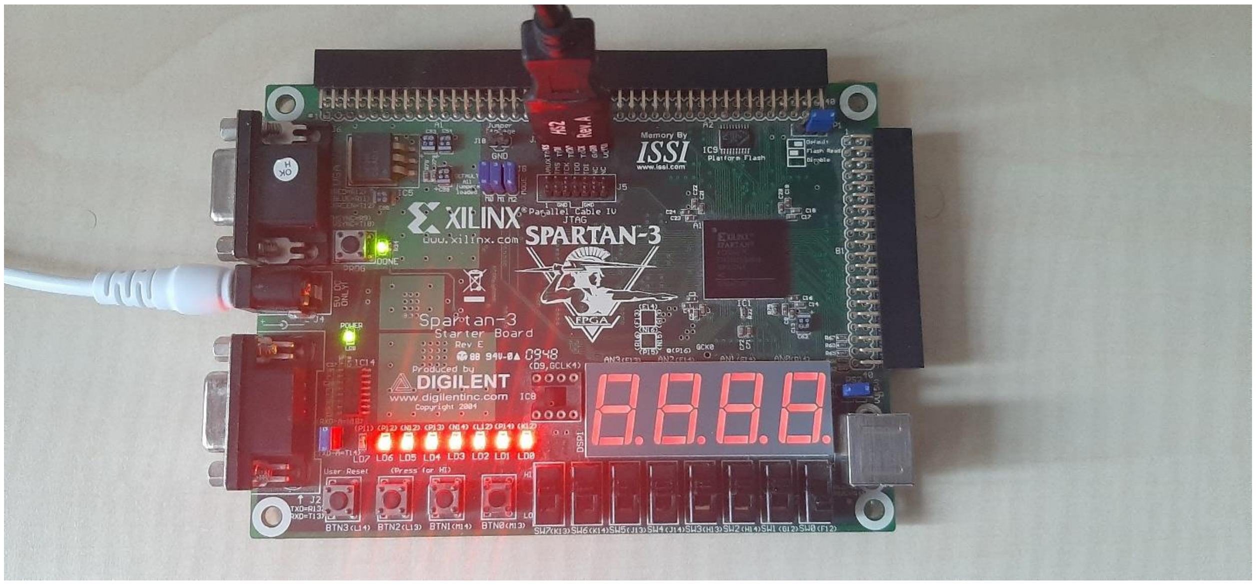
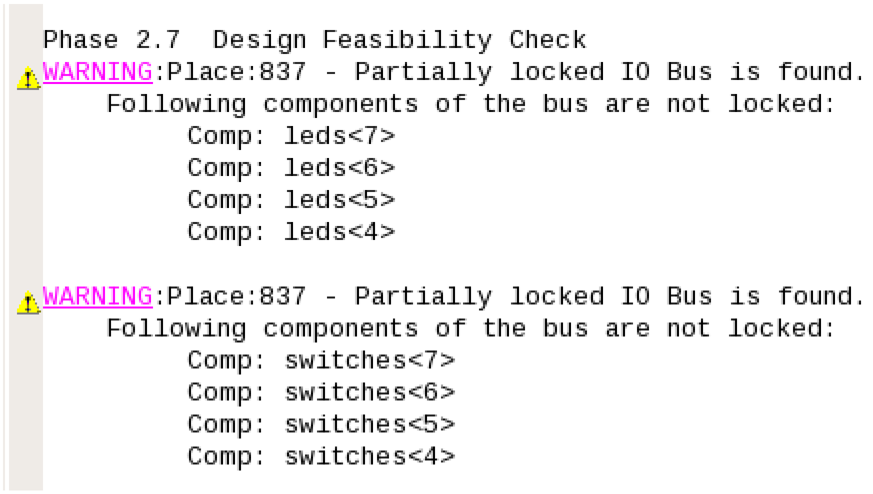


Figure 12: FPGA Board circuit with <F12> switch turned off.

A video has been provided in the following link: https://youtu.be/ce9vkTvAO3c

Problems/ Bugs

Some problems occurred when trying to upload the file, when the code was synthesized and program was implemented using the incomplete .ucf file. We received the following warning:



```
Phase 2.7  Design Feasibility Check
WARNING:Place:837 - Partially locked IO Bus is found.
   Following components of the bus are not locked:
        Comp: leds<7>
        Comp: leds<6>
        Comp: leds<5>
        Comp: leds<4>

WARNING:Place:837 - Partially locked IO Bus is found.
   Following components of the bus are not locked:
        Comp: switches<7>
        Comp: switches<6>
        Comp: switches<5>
        Comp: switches<4>
```

This warning was received because the .ucf file was incomplete, and the functions for the pins outlined in the warning message were not defined.

After the .ucf file was completed, the program ran smoothly.

Importance of .ucf file

The .ucf file is a constraints file format. In our project, it is used to apply various constraints on the design to the FPGA board. We used the .ucf file to map the switches and two of the buttons (input devices), that we used in our experiment, to the LEDs (output devices).

Problems encountered

Some problems were encountered in lab due to a faulty cable. The cable prevented the code to be downloaded into the FPGA board. This problem was rectified by changing the faulty cable.

Conclusion

In this lab, we received a basic introduction to Xilinx ISE, VHDL and the different ways of programming an FPGA board using Xilinx tools. We learned about different hurdles that we can come across when programming an FPGA board, and how to rectify these problems.

References

https://electronics.stackexchange.com/questions/108683/usage-of-ucf-file-and-clock-divider#:~:text=Xilinx%20UCF%20is%20a%20constraints,locations%2C%20timing%2C%20and%20area.&text=In%20this%20top%20file%2C%20then,that%20are%20in%20your%20project.

http://fpga-fhu.user.jacobs-university.de/?page_id=34

http://fpga-fhu.user.jacobs-university.de/wp-content/uploads/2014/10/Synthesis-Reference.pdf

http://fpga-fhu.user.jacobs-university.de/wp-content/uploads/2014/10/S3BOARD-rm.pdf