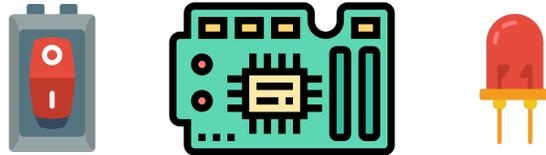


New York City College of Technology
Computer Engineering Technology



Exploring Switches & Lights Using the Altera DE-10 Lite

Lab 003

CET 4805 Component and Subsystem Design II
Section OL40
Professor Hakan Pekcan
Spring 2021
Report Written by Maria Pedroza & Galib F. Rahman

Table of Contents

Objective	3
Procedure & Results	3
Creating A Project	3
Task 1: Compile and Simulate	5
Task 2: Compile and Simulate	6
Task 1: Compile and Simulate	7
Task 2: Compile and Simulate	8
Conclusion	9

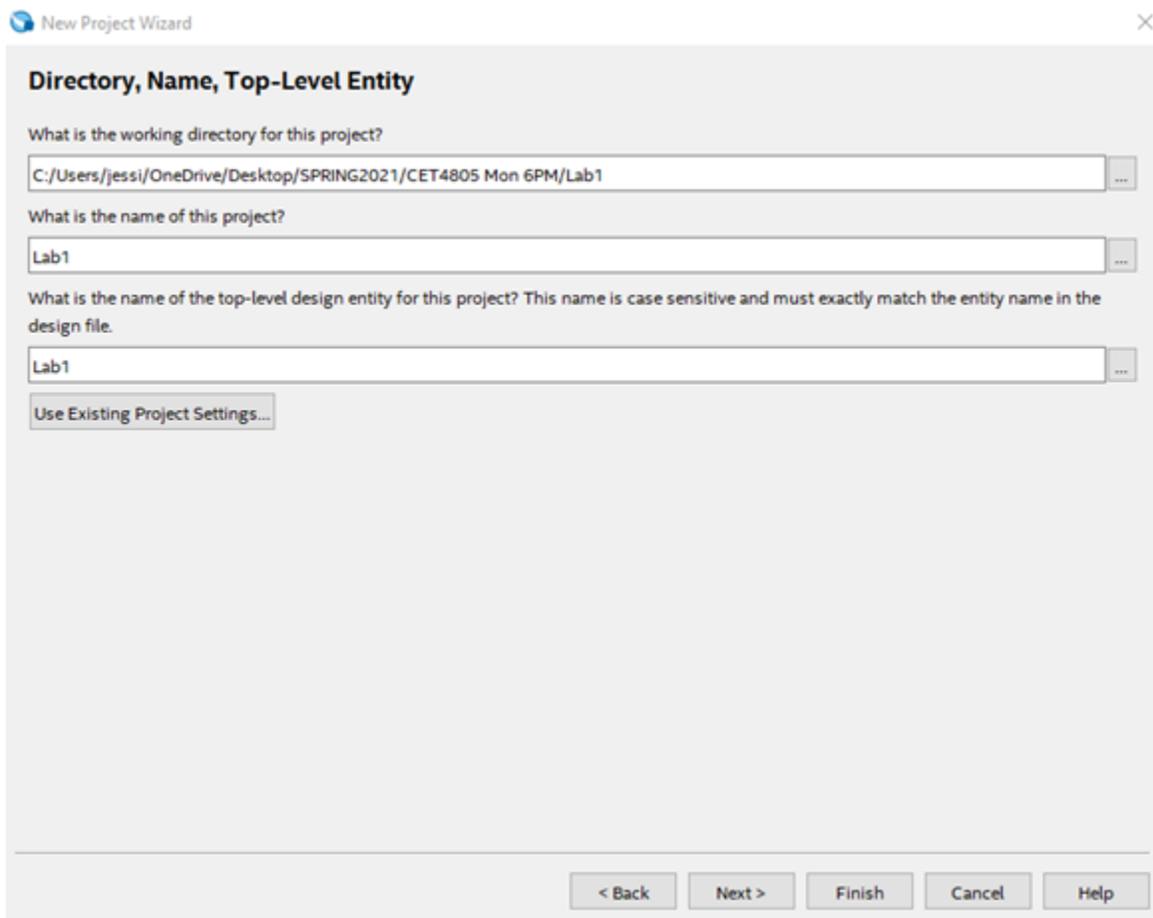
Objective

The objective of this laboratory exercise is to learn to interact with input and output devices using the FPGA chip and implement a circuit using said devices. The primary focus of this lab is to utilize the onboard switches (SW9-0) as input devices and control the onboard LEDs as output devices using VHDL.

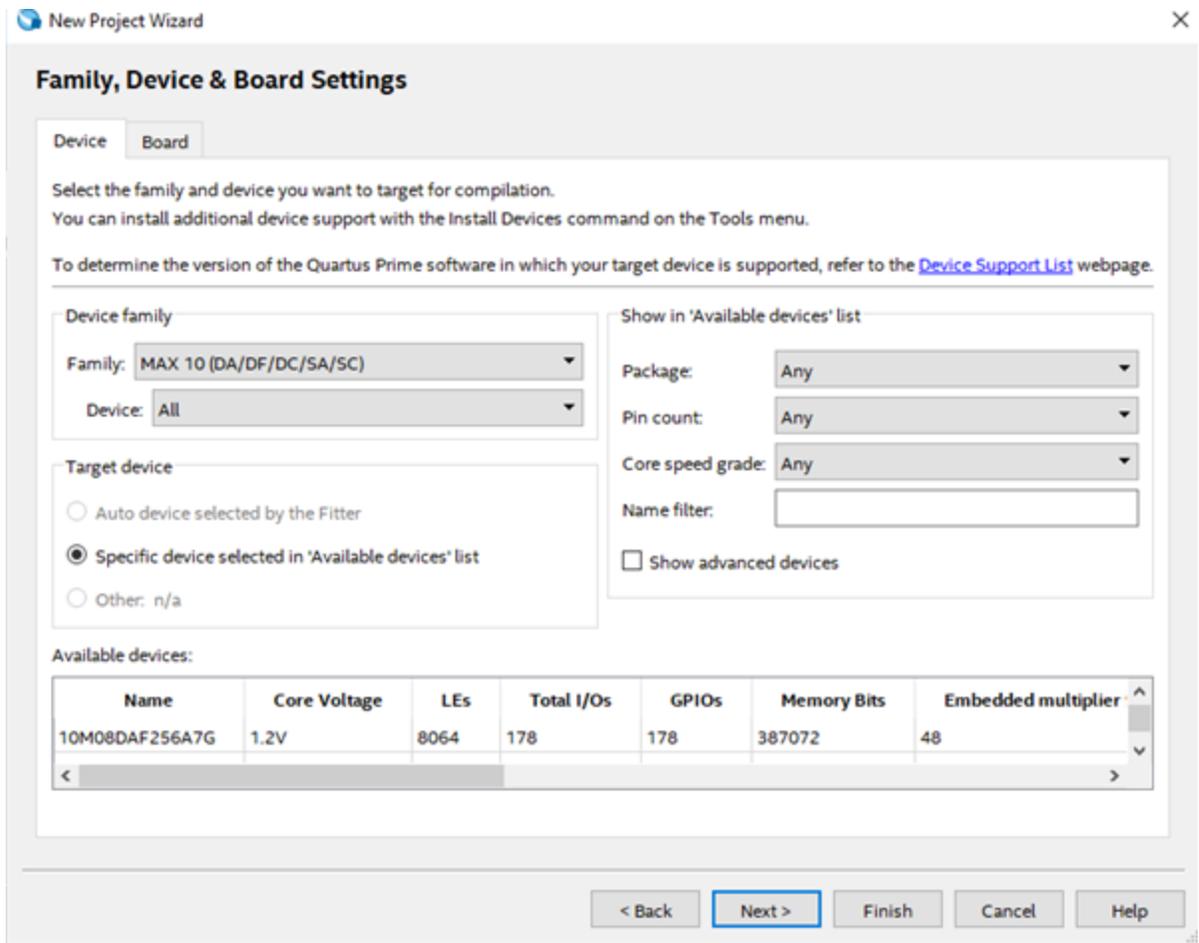
Procedure & Results

Creating A Project

1. To begin we will use the **New Project Wizard** to create a new project for our circuit.

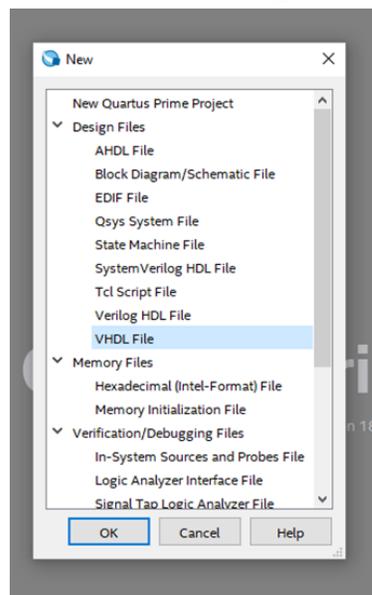


2. The Project Type can be: **Empty Project**
Add Files: Click on Next. You do not need to add any file to your project at this point.
3. You should then see “Family, Device, & Board Setting” screen



4. Click **Next** and **Finish**

5. Create a VHDL entity/architecture for the code in Figure 1 and include it in your project.



File > New > VHDL file > OK

6. Include in your project the required assignments for the DE10-Lite board, as discussed above. Compile the project. (You can skip pin assignments if you do not have the board with you at this moment.)
7. Note that you must save your file as **Lab1.vdh** where the TOP Level Entity name **MUST** match your VHDL File name.
8. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by toggling the switches and observing the LEDs. (If you do not have the board with you, you should pore the design by a Waveform Simulation.
9. Select **New > File > University Program VWF**
10. Click on Left pane, right-click on **Insert Node or Bus > Node Finder > List**.
11. Bring any signal you would like to observe to the Right pane by highlighting the name and pressing > arrow
12. On the Waveform Editor Window assign input **Values** to input signals and run Functional Simulation.

Task 1: Compile and Simulate

Modify the VHDL code to do the following:

Any switch that is going to be switched ON will turn on the next LEDR instead.

For example, if you switch on SW6, it should turn on LEDR7. (Switch, 9 might turn on LEDR0)

The VHDL code to this would look like

```
LEDR(7) <= SW(6);
```

Alternatively, you may write a couple of statement for all such as:

```
LEDR(9 downto 2) <= SW (8 downto 1);
```

```
LEDR(1) <= SW(9);
```

Task 2: Compile and Simulate

Modify the VHDL code to do the following:

Turning on any switch numbered between 0 and 4 itself would turn on upper 5 LEDRs:
LEDR(9:5)

Otherwise:

Turning on any switch numbered between 6 and 9 by itself would turn on first 5 LEDRs:
LEDR(4:0)

Any other combination of switches will turn on ALL LEDRs.

Example code for the architecture section of VHDL file is: Complete the file below.
Process (SW) begin

```
    if (SW(0) = '1' or SW(1) = '1' or SW(2) = '1' or SW(3) = '1' or SW(4) = '1') then
        LEDR (9 downto 0) <= "1111100000"; -- turn on upper LEDRs

    else if (SW(5) = '1' or SW(6) = '1' or SW(7) = '1' or SW(8) = '1' or SW(9) = '1')
    then
        LEDR(9 downto ) <= <fill here> --turn on lower LEDRs

    else
        LEDR <= <fill here> ; --turn on all

    end process;
```

Task 1: Compile and Simulate

Quartus Prime Lite Edition - C:/intelFPGA_lite/18.1/Lab001/Lab001 - Lab001

File Edit View Project Assignments Processing Tools Window Help

Lab001

Project Navigator Hierarchy Entity:Instance

MAX 10: 10M08DAF256A7G
Lab001

```
1 library ieee;  
2 use ieee.std_logic_1164.all;  
3  
4 entity Lab001 is  
5     port ( SW : IN STD_LOGIC_VECTOR(9 DOWNTO 0);  
6           LEDR : OUT STD_LOGIC_VECTOR(9 DOWNTO 0) --red LEDs  
7         );  
8  
9 end Lab001;  
10  
11 architecture Arc_Lab001 of Lab001 is  
12 begin  
13     LEDR (9 downto 2) <= SW (8 downto 1);  
14     LEDR(1) <= SW(9);  
15 end Arc_Lab001;
```

Tasks Compilation Task

- Compile Design
- Analysis & Synthesis
- Fitter (Place & Route)
- Assembler (Generate program)
- Timing Analysis

Messages

System (4) Processing (123)

Simulation Waveform Editor - C:/intelFPGA_lite/18.1/Lab001/Lab001 - Lab001 - [Lab001_20210301181859.sim.vwf (Read-Only)]

File Edit View Simulation Help

Master Time Bar: 0 ps Pointer: 285.17 ns Interval: 285.17 ns Start: 0 ps End: 0 ps

Name	Value
LEDR	0000000000
LEDR[9]	0
LEDR[8]	0
LEDR[7]	0
LEDR[6]	0
LEDR[5]	0
LEDR[4]	0
LEDR[3]	0
LEDR[2]	0
LEDR[1]	0
LEDR[0]	0
SW	0000000001
SW[9]	0
SW[8]	0
SW[7]	0
SW[6]	0
SW[5]	0
SW[4]	0
SW[3]	0
SW[2]	0
SW[1]	0
SW[0]	1

The timing diagram shows the relationship between the switch signals (SW) and the LED outputs (LEDR). The SW[0] signal is shown as a single pulse that goes high. Simultaneously, the LEDR[5] signal also goes high, while all other LEDR signals remain low. This demonstrates the logic where the state of SW[4] (which is 0 in this specific trace) would control LEDR[5] in a larger sequence.

This VHDL source code is designed such that the switch activates the succeeding LED with respect to the denoted number of said switch. In the simulated waveform we observe this output (e.g. when SW[4] is HIGH LED[5] is HIGH and all else is LOW)

Task 2: Compile and Simulate

The screenshot displays the Quartus Prime Lite Edition interface. The top window shows the VHDL source code for Lab001.vhd. The code defines an entity Lab001 with a port SW of type IN STD_LOGIC_VECTOR(9 DOWNTO 0) and an output LEDR of type OUT STD_LOGIC_VECTOR(9 DOWNTO 0). The architecture Arc_Lab001 contains a process that checks for active switches and sets the LEDR output accordingly.

```
1 library ieee;
2 USE ieee.std_logic_1164.all;
3
4 ENTITY Lab001 is
5     port ( SW : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
6           LEDR: OUT STD_LOGIC_VECTOR(9 DOWNTO 0) --red LEDs
7         );
8
9 END Lab001;
10
11 ARCHITECTURE Arc_Lab001 of Lab001 is
12 BEGIN
13
14 process (SW) begin
15     if (SW(0)='1' or SW(1)='1' or SW(2)='1' or SW(3)='1' or SW(4)='1' ) then
16         LEDR(9 downto 0) <= "1111100000";
17
18     elsif (SW(5)='1' or SW(6)='1' or SW(7)='1' or SW(8)='1' or SW(9)='1' ) then
19         LEDR(9 downto 0) <= "0000011111";
20
21     else LEDR <= "1111111111";
22
23     end if;
24
25 end process;
26
27 END Arc_Lab001;
```

The middle window shows the compilation messages, indicating that the compilation was successful with 0 errors and 13 warnings. The warnings include messages about missing clocks and timing constraints.

The bottom window shows a timing simulation waveform. The waveform displays the output of the LEDR (bits 9:0) and the input of the SW (bits 9:0) over time. The LEDR output is shown as a 10-bit vector, and the SW input is shown as a 10-bit vector. The waveform shows that when any one switch 0-4 is activated, the upper 5 LEDs (9:5) are HIGH, and when any one switch 6-9 is activated, the lower 5 LEDs (4:0) are HIGH. All other combinations result in the activation of all LEDs.

This VHDL source code is designed such that if any one switch 0-4 is activated the upper 5 LEDRs 9:5 are HIGH and if any one switch 6-9 is activated LEDRs 4:0 are HIGH and all other combinations result in the activation of all LEDRs. This behavior is demonstrated in our simulated waveform. The waveform activates each switch individually and then all switches are deactivated (or LOW). The observed output correctly matches the objective of the source code.

Conclusion

Upon completion of this laboratory exercise, my partner (Galib Rahman) and I (Maria Pedroza) have successfully utilized the Quartus® Prime software to control the onboard LEDs as output devices by modifying the VHDL code. We utilized the onboard switches (SW9-0) and modified assigned input values, shown as input signals, using the Waveform Editor Window and observed the output waveforms to verify the proper functionality of the VHDL source code.

