ECE 272 Lab 1
Fall 2018

Basic Combinational Logic and the DE10-Lite

Phi Luu

October 3th, 2018

Grading TA: Edgar Perez

Lab Partner: Benjamin Geyer

1 Introduction

The purpose of this lab is to learn how to use Quartus Prime and the MAX 10 DE10-Lite. With these software and hardware tools, we can verify the properties of combinational logic design. Throughout this lab, we will learn how to construct a combinational logic schematic and program this schematic into the DE10-Lite using Quartus Prime.

Most digital electronic implementations have moved towards Programmable Logic Devices (PLD). PLD can be classified into two following categories:

- *Programmable Logic Arrays* (PLA) have fixed hardware-defined logic gates whose connections are configurable.
- *Complex Programmable Logic Devices* (CPLD) use a Hardware Description Language (HDL) to program both the connections of the logic gates and the logic gates themselves.
- *Field-Programmable Gate Arrays* (FPGA) is a subset of CPLD.

The MAX 10 DE10-Lite is an FPGA, which means it will allow us to build and program highly customizable circuits throughout this course without using excessive electronic hardware.

2 Design

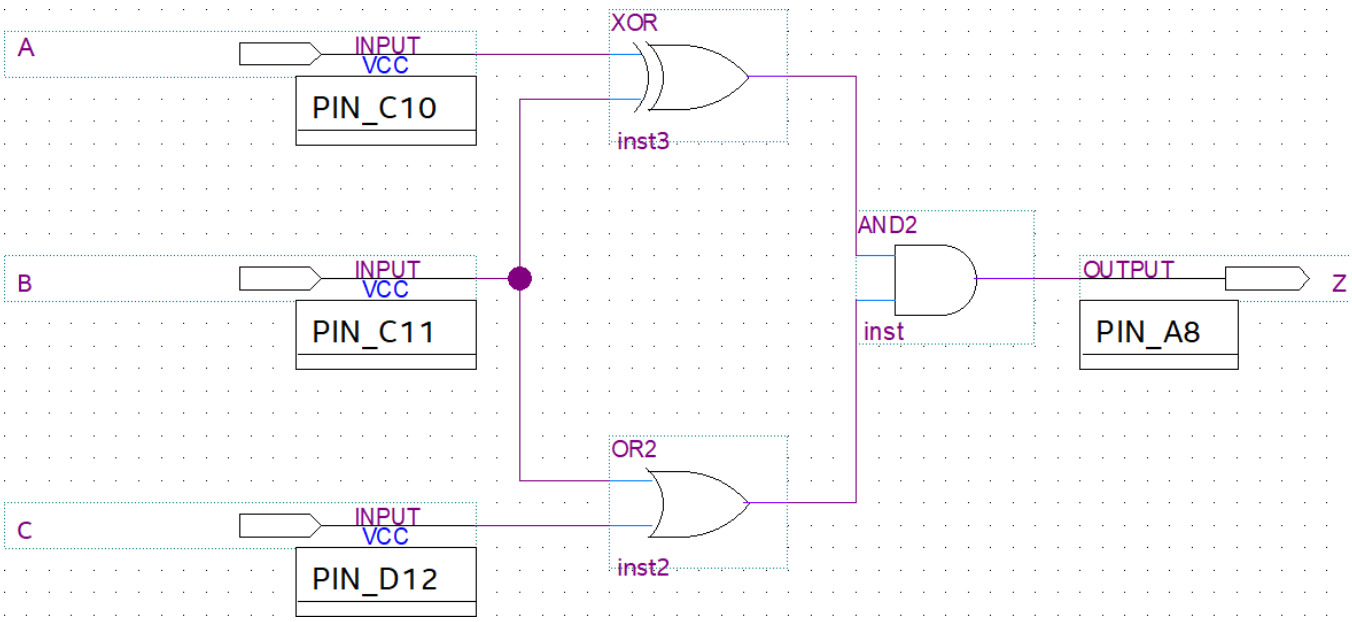

Figure 1: The schematic design of lab 1

Figure 1 above shows a schematic constructed in Quartus Prime. The digital circuit includes a two-input XOR gate, a two-input OR gate, and a two-input AND gate. The output of the circuit is determined by the combination of these three gates, and the truth table of each gate is as follows:

A	B	A XOR B	A	B	A OR B	A	B	A AND B
0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	0
1	0	1	1	0	1	1	0	0
1	1	0	1	1	1	1	1	1

Table 1: Truth tables of two-input gates XOR, OR, and AND, respectively

Figure 2 below shows the block diagram based on which the switches and the LEDs are mapped.

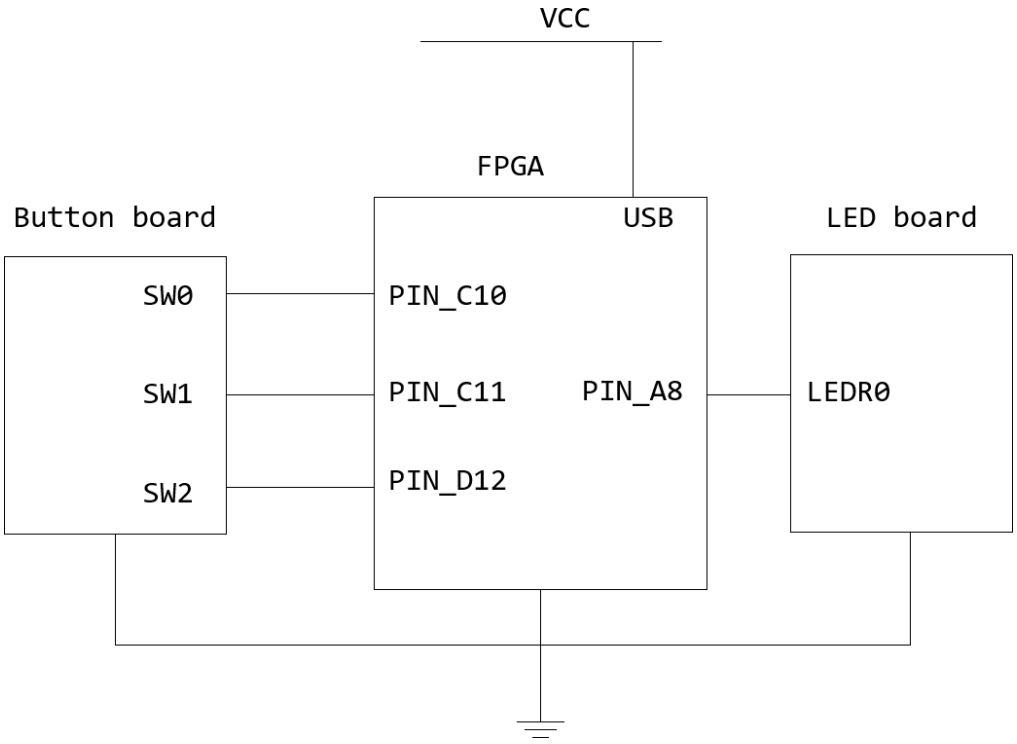

Figure 2: The block diagram designed for this lab, showing the pin mapping

We assign the inputs A, B, and C and the output Z in the Assignment Editor of Quartus Prime, similarly to Table 2 below.

	FPGA PIN
INPUT A	PIN_C10
INPUT B	PIN_C11
INPUT C	PIN_D12
OUTPUT Z	PIN_A8

Table 2: Pin assignments, connecting the schematic with the DE10-Lite

3 Results

A	B	C	EXPECTED Z	ACTUAL Z
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

Table 3: Truth table of the final results

Shown above in Table 3 is the outcome of this lab. A **0** in an input value means the corresponding switch is **not pressed** (or is **toggled off**). Conversely, a **1** in input value means the corresponding switch is **pressed** (or is **toggled on**). This binary concept is also similar with the output: the corresponding LED will turn **off** if the output is **0** and will turn **on** if the output is **1**.

Since the actual outputs of Z matched their expected values, the logic circuit was successfully implemented.

4 Experiment Notes

Reflection

This lab was quite easy since it is the first lab session of the course. The lab showed step-by-step process of getting started with Quartus Prime. I carefully followed the instructions and was able to design a schematic, compiling the program, and upload the code to my DE10-Lite FPGA.

Study Questions

1. Describe any problems encountered in this lab and your solutions to those problems.

   During this lab, I encountered a USB driver problem. However, I was able to solve it quickly by install the pre-downloaded driver located in *C:\intelFPGA_lite\18.0\quartus\drivers\usb-blaster*.

2. Give an example of where discrete logic ICs (such as the 7400 series logic chips mentioned in section 1.1) are used in industry and why.

   Discrete logic ICs are commonly implemented on many instances of technology, one of which is the smartphone. A discrete logic ICs can contains billions of circuits inside and provide a variety of logic gates, such as NOT, AND, OR, NAND, NOR, and XOR while still maintain very small sizes and light weights. Due to the absence of soldered connection, ICs have higher reliability. Therefore, discrete logic ICs are widely use, in this case, in the smartphone industry.

3. Give an example of when you should use an FPGA instead of a PLA and explain why.

   An FPGA is different from a PLA because a PLA has configuration options only for the connections between its fixed hardware-defined logic gates, whereas an FPGA allows programming on both the logic gates' connections and the logic gates themselves. A thought example (not necessarily a real-world example) of favoring FPGA over PLA is when we need the system to perform parallel tasks—as parallel execution is the nature of HDL—and when we need to do extensive configurations on hardware levels. Referenced materials.

4. Give an example of when you should use an PLA instead of a FPGA and explain why.

   This answer is based on the comparison between FPGA and PLA made in the answer to question 3. FPGA seems to be better at multitasking and allows much more flexibility in hardware design than PLA. However, FPGA costs more money and consumes more power than PLA. Additionally, due to its parallel nature, FPGA is not very good at sequential execution compared to PLA. Referenced materials.

5. Summarize the main differences between FPGAs and CPLDs, other than the difference described in the note in section 1.1.

   Section 1.1 of the lab manual notes that FPGAs are different from CPLDs in hardware implementation. After referring to some materials, it seems that FPGAs consist of "logic blocks" in a network of programmable interconnect. These "logic blocks" are made up of gates array, as opposed to Electrically Erasable Programmable Read-Only Memory (EEPROM) cells in CPLDs. CPLDs' architecture has switch-like interconnect, which allows the connections to travel through great distances and reduces the amount of routing between blocks. The main differences between FPGAs and CPLDs can also be seen in the following figure:

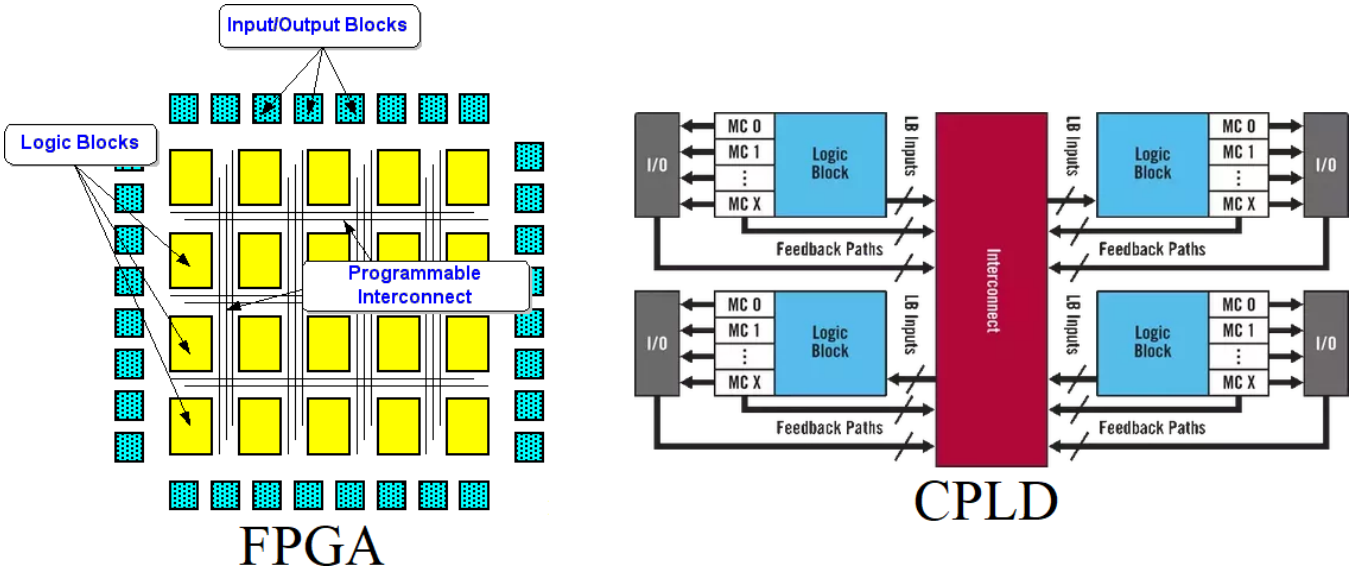

Figure 3: FPGA's architecture (left) vs. CPLD's architecture (right). Source

Appendix

No appendix is available in this lab.