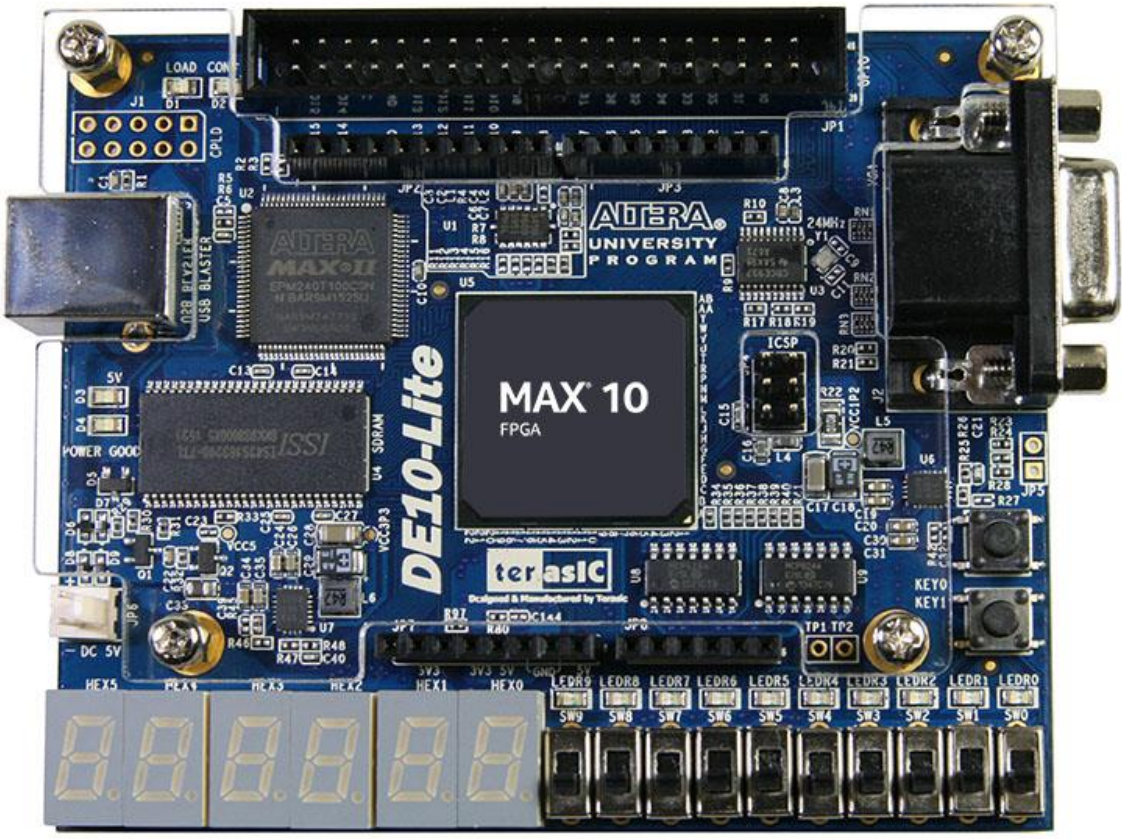
ENGR 272 Lab 1:
Basic Combinational Logic and the DE10-Lite



1.1 Background Information

The use of the logic gate integrated circuit (IC) has had several evolutions. The 7400-series logic chips were the most common digital logic devices around for many years. These chips typically contained 1 or 4 transistor-based logic gates (AND, OR, NOT, etc.), which designers would configure on a PCB to get the desired hardware equivalent to their logic design.

Most digital logic implementations have moved toward Programmable Logic Devices (PLDs). These newer devices are designed so that the user can reprogram the device configuration and/or logic by using a Hardware Description Language (HDL) instead of permanently configuring multiple chips on a circuit board.

PLDs have varying complexity and a wide range of applications:

1. Programmable Logic Arrays (PLA) have a fixed number of hardware-defined logic gates, with configurable connections between them.

2. Complex Programmable Logic Devices (CPLD) use an HDL to program both the connections between logic gates and the logic gates themselves, giving the user more flexibility in configuring the chip.
3. Field-Programmable Gate Arrays (FPGA) are similar to CPLDs in that both the connections and the logic gates are reprogrammable, but they differ in their hardware implementation.

One difference between CPLDs and FPGAs is that CPLDs use Electrically Erasable Programmable Read-Only Memory (EEPROM–Information about this can be found by referencing the textbook's index), which is less volatile than the RAM the FPGA uses.

1.2 Learning Objectives

In this section, the following items will be covered:

1. Using Quartus Prime to draft digital logic designs.
2. Programming the Max10 FPGA on the DE10-Lite.
3. Verifying combinational logic designs.
4. Simulating digital logic designs.

1.3 Materials

1. Quartus Prime Lite Edition V. 18.0/18.1
2. DE10-Lite kit with MAX10 10M50DAF484C7G FPGA
3. USB to USB-B cable

1.4 Procedure

There are 6 steps to successful digital logic design. This process flow is crucial, and you will see it repeated throughout this course.

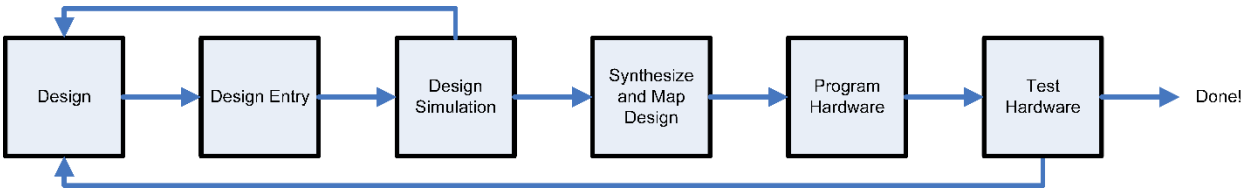


Figure 1.1: This six-step process is used for good digital logic design.

1. **Design:** The context of the design is established in this step. The context involves defining the inputs, desired outputs, and all the logic required in-between. In this step, all the minimizations and layout are planned for the design entry process. While this step is not always the longest, it should involve the most thought and effort. This typically requires a complete block diagram showing all the logic blocks and the connections between them, often with written explanations of specific functions.

2. **Design Entry:** The actual drafting of the digital logic design occurs in this step, translating the design from block diagrams and descriptions into the software. This can be accomplished directly by writing HDL code, or graphically by drawing a schematic that a software tool can convert into HDL code.
3. **Design Simulation:** Before committing to hardware, this step tests the design in a controlled computer simulation. If the design does not function as specified in the "Design" step, it is revised.
4. **Synthesize and Map Design:** When the design simulates correctly, the HDL and schematic source files are synthesized into a design file that can be written to the FPGA. This includes assigning the inputs and outputs of the design to IO pins.
5. **Program Hardware:** After the design file is created, it is used to configure the FPGA. Quartus Prime sends a bit stream over the USB-B cable to configure the DE10-Lite FPGA.
6. **Test Hardware:** Verify hardware operation once the FPGA has been programmed. The FPGA should operate exactly as the design predicted, which was verified by the simulation. Synthesis problems, timing violations, or incorrect assumptions about the hardware can require the designer to return to the "Design" step.

1.5 Design

1.5.1 General Design Concepts

There are a few design choices that are dictated by the hardware. Familiarize yourself with the following concepts before attempting to design your logic.

1. **Boolean Logic:** Boolean Logic is either True/HIGH/1 or False/LOW/0. In Digital Logic, HIGH generally refers to VCC (power supply +), and LOW refers to GND (ground).
2. **Active High/Low:** Something that is Active High will turn on when a HIGH signal is sent to it. Conversely, something that is Active Low will turn on with a LOW signal.
3. **Pull-up/Pull-down Resistors:** Inputs that are not constantly driven high or low by logic should never be left floating (not connected to anything), because signal fluctuations on floating pins can cause HIGH or LOW inputs to be mistakenly read, leading to undefined behavior. Pull-up/down resistors prevent this issue by tying inputs to either VCC or GND so that when the input isn't being driven externally, the input is at a fixed logic level. The resistance is large enough that it will be negligible when an input is connected. Figure 1.1 shows examples of Pull-Up and Pull-Down Resistors.

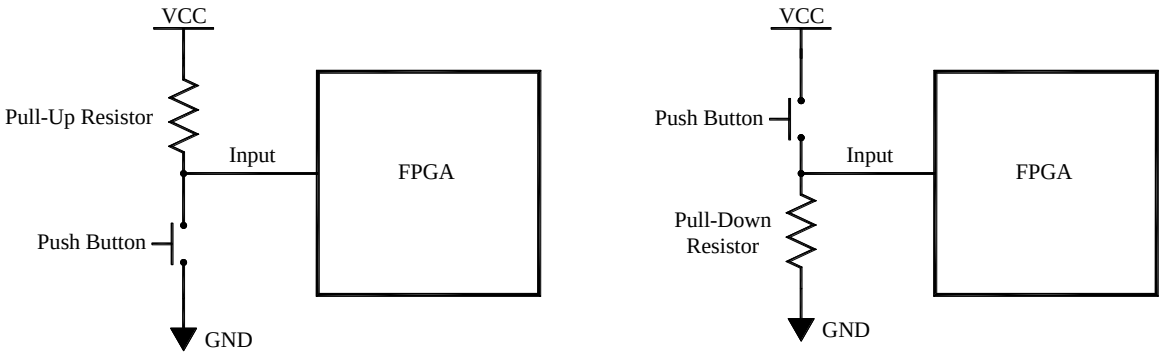


Figure 1.1: Pull-Up and Pull-Down Resistor examples

1.5.2 Block Diagrams

Block Diagrams are often used to describe an entire design at a high level. Block Diagrams are to be created near the start of a project and may evolve as the project progresses. Complex designs may require multiple block diagrams with varying levels of abstraction, with smaller and more detailed block diagrams used to describe larger blocks in the overall design.

At the beginning of a project, the block diagram should include:

• Labelled blocks representing each piece of hardware used in the project.

• All interfaces (connections) between blocks, represented with lines.

• Which signals are carried on each interface.

• How the interface connects to each block, for example, pin numbers, or other connector names.

Figure 1.2 shows a simple block diagram for this lab.

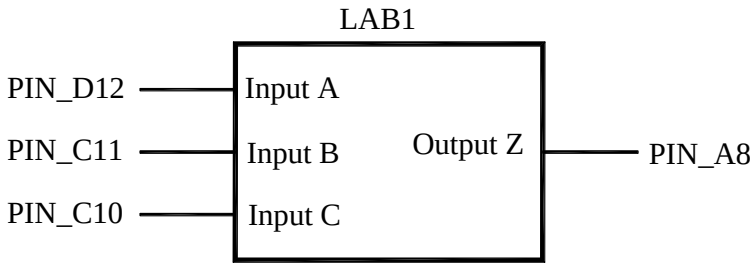


Figure 1.2: Possible block diagram for Lab 1

1.5.3 Schematics

Schematics show the logic gates required to implement each block in the block diagram. Proper design has inputs on the left or top, and outputs on the right or bottom. Figure 1.3 shows an example of a bad

layout, with disorganized inputs and outputs and non-sensical routing. Figure 1.4 shows an example of a good layout, with the 4 inputs separated at the top of the schematic, connected to a vertical bus that makes it easy to see which input each logic gate in the schematic is connected to.

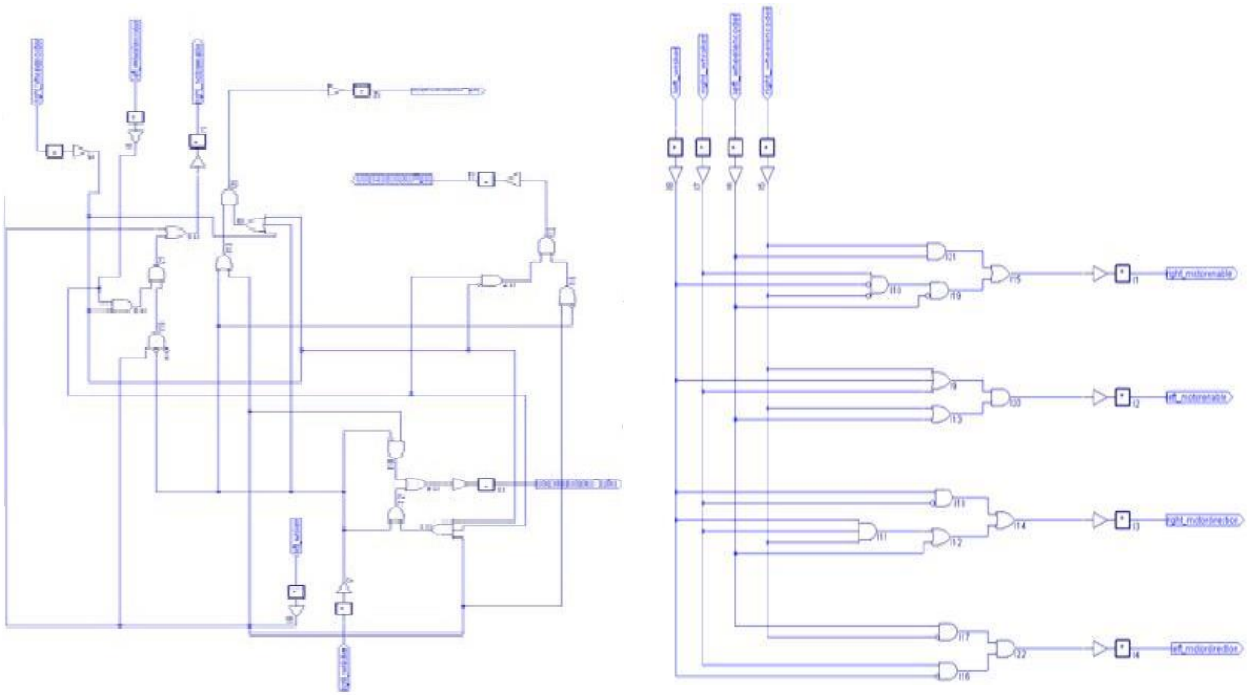


Figure 1.3 (Left): Bad Layout
Figure 1.4 (Right): Good Layout

Figure 1.5 shows the design that will be drafted in Quartus Prime.

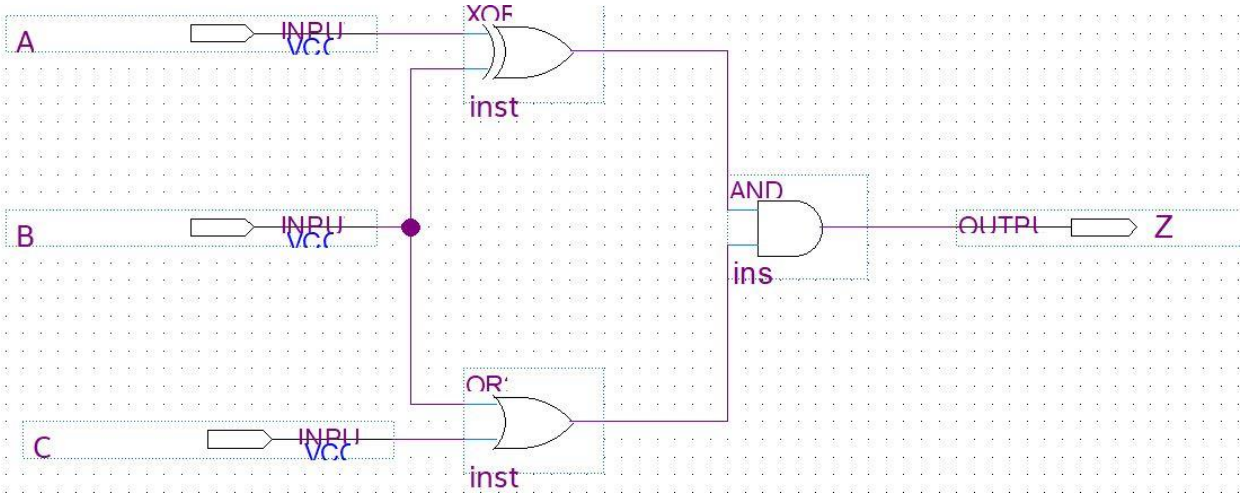


Figure 1.5: Section 1 Combinational Logic

1.6 Design Entry

1.6.1 Start Quartus Prime

Type Quartus into the Windows search bar and click on "Quartus (Quartus Prime 18.0)

Installation instructions for Quartus can be found on the ENGR 272 Moodle Page.

1.6.2 Create a New Project

A project in Quartus is a collection of sources (Schematic or HDL), test benches, and simulation outputs. The project information specifies to Quartus what FPGA model you are working with and which synthesis tool to use, as well as some other parameters. Each project will compile all of its contained source files together, so you will create a new project for every lab section.

Follow these steps to create the project for this section.

1. Click File → New Project Wizard
2. Click Next to advance.
3. Select a work directory where your project will be stored and name your project. You should create a new directory for each lab. Then click next.
4. In Project Type, select an Empty project and click next
5. For this lab, you do not need to add any files, so just click next.
6. The device Family is **Max 10**. Under Available devices, select **10M50DAF484C7G**, and click next.
7. For this lab you do not need to edit the EDA Tool settings, just click next.
8. Review the Project Information and click Finish.

1.6.3 Schematic Entry

Schematics are one type of source file in a project. A project may have several schematic source files, which will be explored in later labs.

Follow these steps to enter the design for this section shown in Figure 1.5 into Quartus.

Create a Blank Schematic Source File:

1. To add a source, Click File → New
2. In the New window under Design Files, select Block Diagram/Schematic File.
3. The Quartus project screen will now be showing the blank schematic file.

Add Symbols:

1. Click the Symbol button.
2. Select the installation directory → primitives →logic library.
3. Select and2.

4. Place one AND gate onto the schematic.
5. Add both other gates (XOR & OR2) from Figure 1.5 onto the schematic.

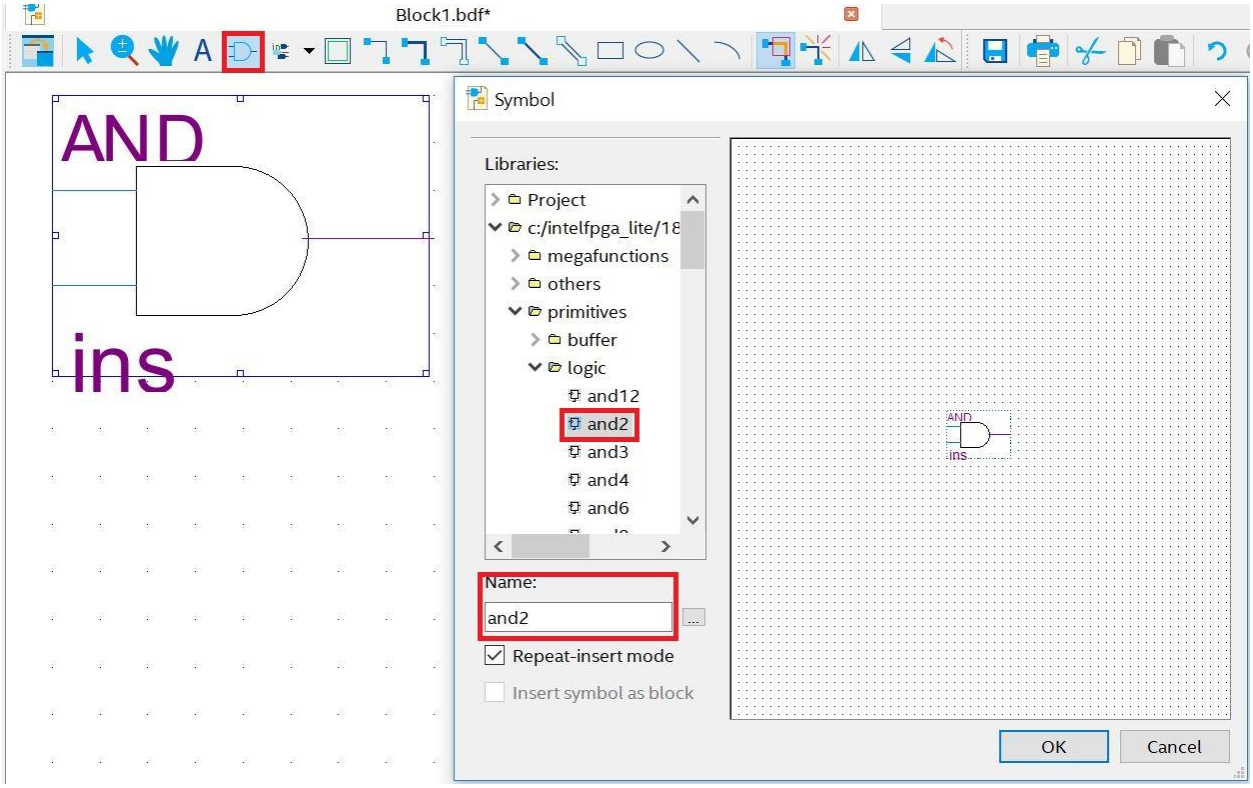


Figure 1.6: Add symbol process.

Connect the Symbols:

1. Click the Orthogonal Node Tool button.
2. Wire all the symbols together as shown in figure 1.5. Hold left click to draw wire connections.
3. Add hanging wires off the inputs and outputs so that they can be named and designated as inputs or outputs in the next step.

Nodes may sometimes appear to be connected when they aren't. An X signifies a floating end to a wire. Always check for broken connections in your schematics!

Define Inputs and Outputs:

Labeling the inputs and outputs on the schematic allows Quartus to create a 'netlist' (stands for a 'list of electrical networks') during the synthesis process. A netlist contains all inputs, outputs, logic gates, and the connections between them.

1. Click the Pin Tool and select Input.
2. Place the input marker on the wires leading into the gates, like in Figure 1.5
3. Following a similar process, add an Output marker named Z.

4. Save the schematic (Ctrl-S).

1.7 Design Simulation

Simulation is done with ModelSim for this course. See the links on the main Moodle page for information on the software. Your simulation results should match Figure 1.10.

To generate the Verilog Hardware Description Language file for use in ModelSim, go to File -> create/update -> create HDL Design File for current file -> Verilog

This is a hardware description language (HDL) file. Find and open the **.v** file in ModelSim and simulate your design. An Example of a ModelSim simulation is shown in figure 1.7. Note that this is not the simulation for this lab.

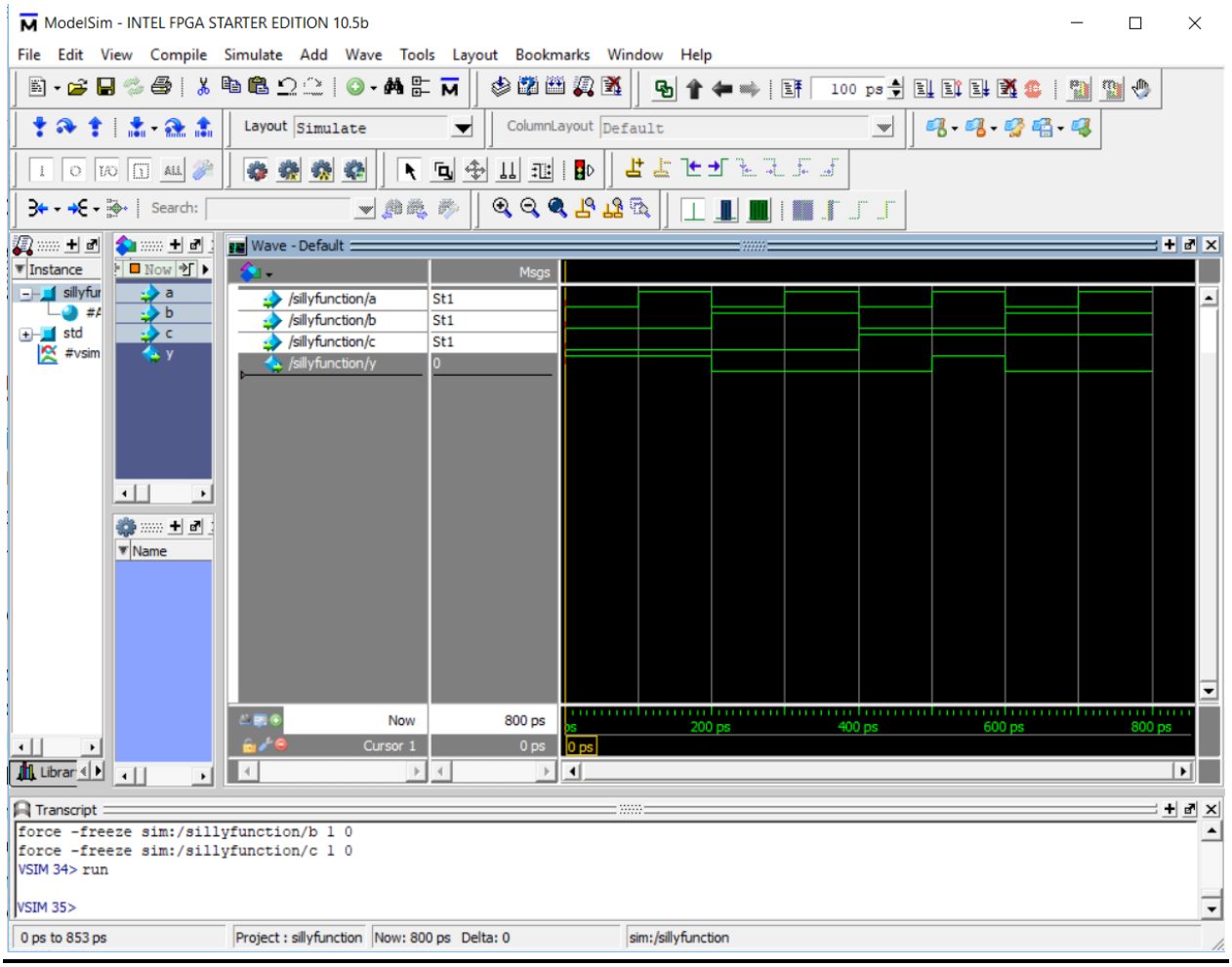


Figure 1.7: Example ModelSim simulation.

1.8 Synthesize and Map Design

During synthesis, the design is minimized and transformed into a netlist describing the hardware. Minimizations reduce the amount of hardware and space required for the design, improving speed and efficiency. After synthesizing, Quartus knows how many inputs and outputs are required for the design, and you can then map those inputs and outputs to pins.

You must re-compile your design any time you change source files for the project! This includes editing assignments.

1.8.1 Compile Design

Click the Blue Play button to Compile or go to the Processing tab and click Start compilation.

1.8.2 Assignment Editor

The Assignment Editor in Quartus is where you set which pins the inputs and outputs are connected to. Pin names can be found in the DE10-Lite user manual.

Fill out the table below in Table 1.1 to record which pins you selected. Assign Output Z to one of the LEDs on the FPGA, as shown on pg. 27 of the user manual. Assign the Inputs A, B, and C to switches.

	FPGA PIN
INPUT A	
INPUT B	
INPUT C	
OUTPUT Z	

Table 1.1: FPGA Pin table.

A similar table should be filled out for every project to help keep track of the pins, especially in more complex designs that can have dozens of inputs and outputs!

Without a pullup resistor, an input pin will float between logic high and logic low when the button is not pressed. Floating inputs cause erratic behavior. This is less of an issue when the input pin is constantly driven by logic to a value.

Follow the steps below to fill out the spreadsheet view for this project:

1. Click Assignments→ Assignment Editor.
2. Double Click <<new>> under To and either type in the name of a node (A) or use the Node Finder to select one.
3. Under Assignment Name type select Location (Accepts wildcards/groups). You can just type L and press enter.
4. Under Value, enter the Name of the Pin you wish to connect that node to. EG: PIN_A8
5. Repeat this process for the other two inputs and the output.
6. Save the pin assignments (Ctrl-S).

1.8.3 Creating the Programming File

The programmer needs a **.sof** (SRAM object file) file to program the Max10. Running the compiler will make the **.sof** file.

1.9 Program The FPGA

Use this process to program the .sof file onto the FPGA:

1. Plug the DE10-Lite into the computer with the provided USB cable.
2. Click Tools → Programmer.
3. Click Hardware Setup and select the USB Blaster. If it does not show up, make sure you correctly installed the USB drivers.
4. In the File column, output files/lab1.sof should already be there.
5. Make sure the correct device is selected.
6. Click Start

Communication between the Breakout Board and a PC via the USB connection cable requires the installation of the FTDI chip USB hardware drivers. Loading these drivers enables the computer to recognize and program the Breakout Board. Refer to the Quartus Installation instructions for USB driver installation instructions.

1.10 Test Hardware

Fill in the truth table below in Table 1.2 with the expected Z Output for every combination of the A, B, and C Inputs. Then, test each combination of inputs and record the output on the LED. This truth table should match your simulation results.

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

Table 1.2: Truth Table for Section 1 Testing

If you want to measure the actual voltage on the pins, you need to use a Digital Multimeter (DMM). A DMM has two probes, black (-) and red (+). To measure a voltage, first set the DMM to measure DC voltages in the 0 - 20V range (or equivalent, depending on your DMM). Place the black probe's metal tip

on one of the test points on your FPGA labeled GND and place the red probe's metal tip on the pin you want to measure. You will see a number around either 0, or 3.3V.

1.11 Checkoff

1. Verify that the schematic design matches the combinational logic circuit shown in Figure 1.5
2. Verify that the Inputs and outputs match the Pin & Pullmode Table in Table 1.1
3. Verify the truth table in Table 1.2 is completely filled out, and that the hardware logic matches the results.
4. Verify the simulation in ModelSim matches the expected Results.

1.12 Study Questions

The study questions go at the end of the Lab Report

1. Describe any problems encountered in this lab and your solutions to those problems.
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.
3. Give an example of when you should use an FPGA instead of a PLA and explain why.
4. Give an example of when you should use a PLA instead of an FPGA and explain why.
5. Summarize the main differences between FPGAs and CPLDs, **other than the difference described in section 1.1.**

1.13 Report Rubric

The reports for the labs are to be formal reports and are to be completed using a word processing application. Please turn in the report to Moodle in a PDF or MS Word format (No Google Docs share links).

Your report should contain at a minimum the following items:

1. **Title**
2. **Date**
3. **Name**
4. **Objectives**
5. **Equipment/Parts/Materials**
6. **Procedure**
    a. **Design**
    b. **Design Entry**
        i. **Block Diagrams (like the one shown in Figure 1.2)**
        ii. **Schematics you created in Quartus Prime (like the one shown in Figure 1.5)**
        iii. **Table 1.1 with the recorded pins you selected.**

        iv. **A screen capture of what pin assignments you made in the Assignment Editor in Quartus.**
    c. **Design Simulation**
        i. **Table 1.2 filled in with the expected Z Output for every combination of the A, B, and C Inputs.**
        ii. **A screen capture of the ModelSim simulation like the one shown in Figure 1.7**
7. **Complete the study questions in section 1.12**
8. **Observations**
9. **Conclusion**
10. **References**
11. **Appendix (Source code of Verilog files used for your simulations).**

All images need to have a title and a figure number. For more detailed information about what goes into each of the different sections, see "ENGR 272 Guidelines for Technical Lab Reports" in Moodle.

For an example of what your reports should look like see "ENGR 272 Example Report" in Moodle.

References:

1. Oregon State University ECE 272: Section 1: Basic Combinational Logic and the DE10-Lite (2019) https://eecs.oregonstate.edu/tekbots/courses/ece272/section1

2. Harris, S. L. & Harris, D. H. (2016) Digital Design and Computer Architecture: ARM Edition 1st Edition, Waltham, MA: Elsevier.