CPE/EEE 64

Lab Two: Introduction to Verilog

Instructor:
Dennis DHALQUIST

Abstract

Two input logic gates are synthesized for the Altera Cyclone IV FPGA using the Quartus IDE. The logic gates are verified using a System Verilog testbench and Mentor's Modelsim HDL simulator

COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. Introduction

THIS lab introduces the DE0-Nano and Altera's Quartus development environment. Quartus will be used to program the Altera FPGA on the DE0-Nano. We will start by using Quartus' schematic representation of logic modules. This will be very similar to the logic that you have created in Multisim. The schematic editor in Quartus can be used to graphically represent Verilog modules, like you will do in Lab Two, or logic primitives like will be done in this lab. This lab is created to introduce the student to the following concepts.

- Logic primitives on the FPGA
- Quartus development environment
- Synthesis of a block based design
- Assigning pins for a design
- Programming an Altera FPGA

A. *Included Screencasts*

A number of screencasts are included with this set of labs. They are available on Youtube and as zipped MP4s on the course website. They are intended to be short and to the point so they cover individual topics.

1) Installing Quartus
2) Opening, Compiling, programming with Quartus
3) Block editor in Quartus
4) Pin assignments in quartus.

II. Lab Procedure

THIS section is a guide for what must be accomplished in the lab. Keep in mind the some of the following material will need to be documented in your lab report. For this section you will need:

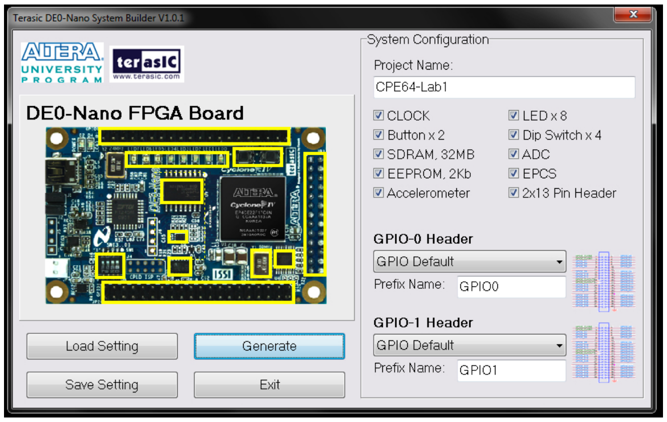
- DE0-Nano Development board
- A breadboard
- Four LEDs
- Two dip switches
- Windows or Linux based computer
- Internets

A. *Install Quartus*

Download most recent version of Quartus and Cyclone device drivers from Altera. Screencast 1 is a walk through for windows, but Quartus is also available for Linux.

B. *Expand Source.Zip*

The Quartus project file, logic gate modules, and test bench are contained within this archive. Expand it wherever is convenient for you, it will be accessed frequently. The Quartus project file included with the source code is generated with the Terasic DE0-Nano System Builder that is included with the Terasic System CD. You can use this if you want to generate a clean project for yourself.



Terasic DE0-Nano System Builder V1.0.1

System Configuration

Project Name: CPE64-Lab1

☑ CLOCK	☑ LED x 8
☑ Button x 2	☑ Dip Switch x 4
☑ SDRAM, 32MB	☑ ADC
☑ EEPROM, 2Kb	☑ EPCS
☑ Accelerometer	☑ 2x13 Pin Header

GPIO-0 Header: GPIO Default
Prefix Name: GPIO0

GPIO-1 Header: GPIO Default
Prefix Name: GPIO1

Fig. 1. TerasIC System Builder GUI

C. *Open Quartus project*

Navigate to where you expanded the Source.zip file. The Quartus project file you want to open is called "./Verilog/Quartus Project/64LabOne.qpf" When you open the project file open the "Top Level" .BDF file Inside this file there will be a number of logic gates attached to pins of the FPGA. A screen cast explains the intricacies of the Schematic editor.

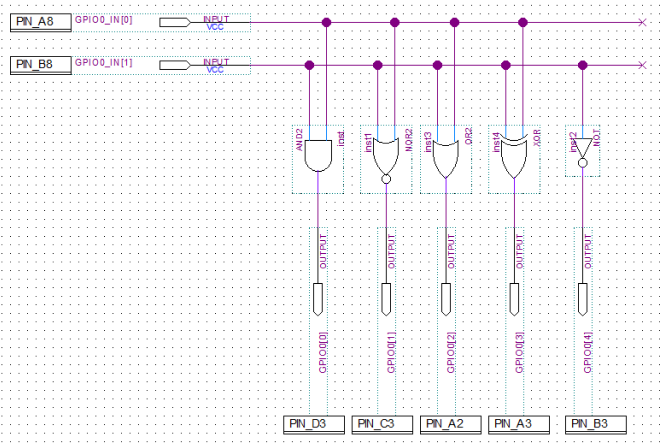


Fig. 2. Quartus logic schematic editor example

D. Prepare circuit to test Verilog gate modules with DE0-Nano

A switch and LED are going to be used to test the FPGA while it's operating. This switch will allow you to generate inputs for the FPGA. The LED circuit needs a current limiting resistor as shown in the Figure below.

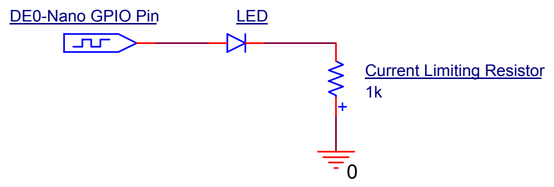


Fig. 3. LED with current limiting resistor

The LED will allow you to see the the output but we'll also need to be able to generate some input for the FPGA. We will do this with a dip switch and pulldown resistor. The pulldown resistor is needed to literally pull the charge off the wire so the input will read a solid zero. Otherwise the pin would "float" between 1 and 0 arbitrarily.

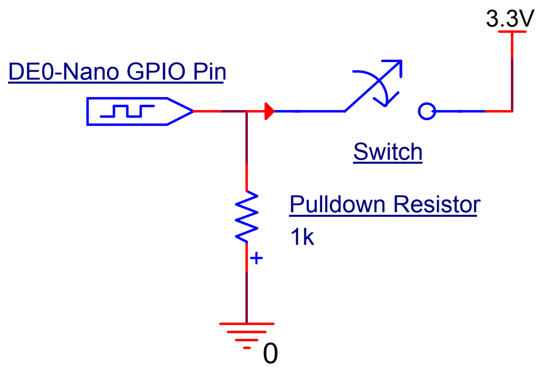


Fig. 4. Switch with pulldown resistor

1) GPIO headers on the DE0-Nano: Be careful when referencing the pin diagrams in the DE0-Nano user manual. It is easy to read it backwards and that can be a mistake that will cost you hours. It is easiest to match the Nano's orientation with the schematic and count from the nearest edge. Always check VCC_SYS, VCC3P3, and GND with a multimeter before attaching a circuit you have built.

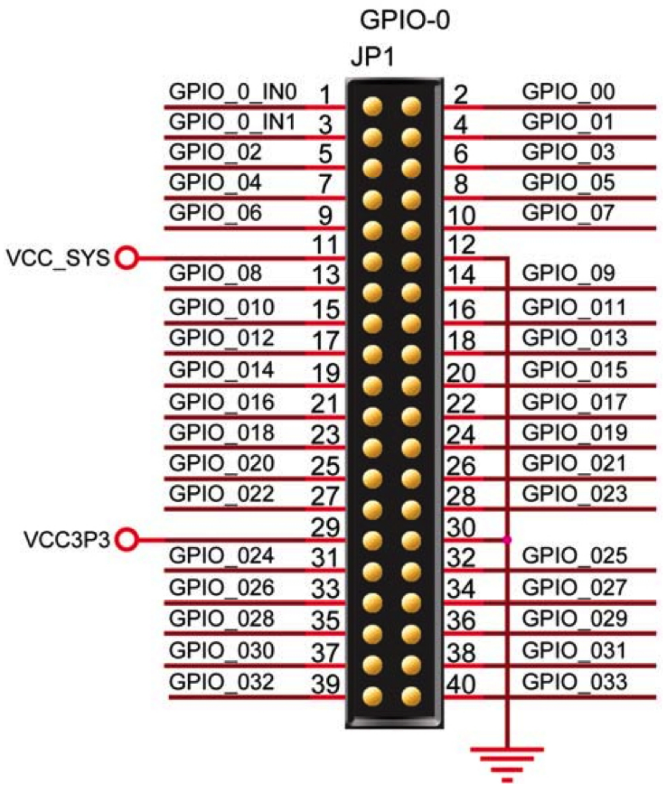


Fig. 5. Schematic of GPIO-0 header [1]

This is the header pin schematic from the DE0-Nano user manual. Inside the verilog code these pins follow a little different nomenclature. What is labeled as GPIO_0_IN[0] in figure 5 is GPIO0_IN[0] and GPIO_00 is GPIO0[0]. Refer to the DE0-Nano user guide for details on the orientation of the headers.

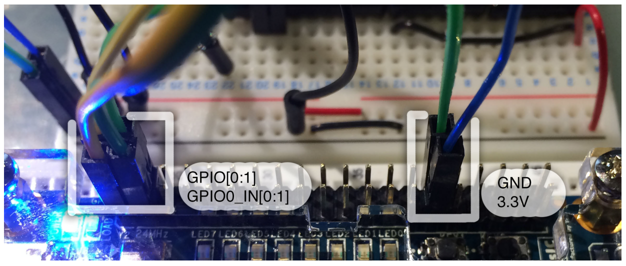


Fig. 6. Picture of loaded GPIO-0 header

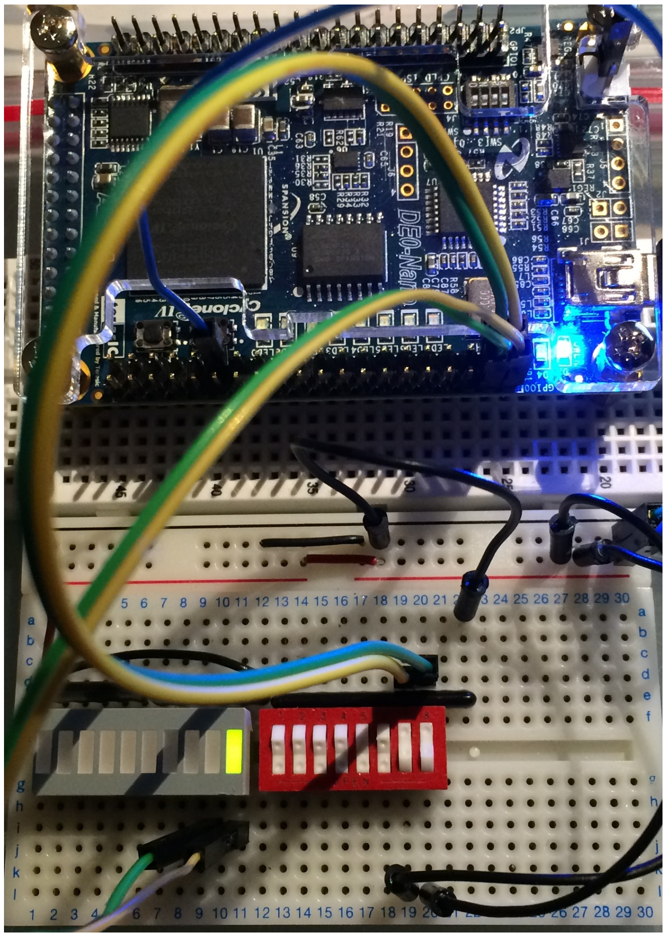


Fig. 7. Example switch and LED configuration with SIP resistors

2) Compile example with Quartus: Once you've created and tested your switch circuit we'll need to compile the example project. This can be done with the Quartus development environment. A walk through is provided in screencastX.

3) Use Quartus to program the Nano: refer to screencastX for a walk through. After synthesis Quartus will generate a .SOF[1] file that can be used to program the FPGA using Quartus' programmer. Follow the procedure outlined in screencastX.

E. Test behavior against expected truth table

Use the known behavior of the logic gates to test your design on the FPGA.

F. Design XOR module

Now that you have had the opportunity to experiment with the logic primitives in the schematic editor the time has come to create your own design. Lets say that we need a four input XOR gate for some reason. We know the truth table of a XOR gate, it must be high when any number of the inputs is high but all of them as shown in Table I. Quartus does have an embedded XOR primitive which the use of is disallowed. Create the circuit from the basic 7400series logic operators. Always remember, Google is your best friend.

[1]the .SOF stands for SRAM object file. This is an Altera standard for of their FPGAs.

TABLE I
TRUTH TABLE FOR XOR LOGIC FUNCTION

A	B	C	D	Output Q
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

G. Design AOI module from logical equation

The "And or Invert" function is common in digital design, you will implement it with logic primitives.

$$F = \overline{(A \wedge B) \vee (C \wedge D)}$$

III. Lab Report

THE lab report must be typed and submitted in a PDF format. Look to IEEE's guidelines for guidelines on format. The document should include the following items.

A. Figures to include

- Schematic of your XOR Gate
- Logic tables from theoretical prediction and experimental outcome
- Explanation and schematic of your XOR module.

- Explanation and Schematic of your AOI module.

B. *Questions to answer*

1) Notice the report that pops up when you compile your project. There are a number of statistics given by Quartus, the logic element usage ratio is your designs use of the total device capacity.
2) The synthesis engine in Quartus is very powerful and will optimize your design for the resources on the FPGA. Quartus generates the design using three logic gates and two inverters, which gates do you think it changed to inverters?

IV. Conclusion

THIS lab introduced Quartus and some of it's basic functionality. Quartus is very similar to many industry standard tools. Xilinx, another manufacturer of programmable logic, offers a tool suite very similar to Quartus called ISE. An understanding of Quartus will allow ISE to be learned very quickly.

The block editor introduced in theis section can also generate block diagrams for Verilog modules, which will be explored in Lab 3. While Verilog is a much more efficient way to create a digital design, the high level block diagram can serve as a great visualization tool when combined with the behavioral modeling of Verilog.

a) More Information: just like anything else, the Internet has an amazing amount of information available for the interested student.

- Altera Traning Curriculum for FPGA designers
- EEV Blog: What is a FPGA?

References

[1] TerasIC, *DE0-Nano User Manual*, 1st ed., 2012.