Lab Two: Introduction to logic on the FPGA

Ben Smith

***Abstract*—This document is an introduction to the DE0-Nano development board, Altera's Cyclone IV FPGA and the Quartus IDE. The schematic editor feature of Quartus is used to synthesize logic gate primitives and more complex logic functions from these primitives. The synthesis of these simple structures allows the student to gain experience with the development tools used by the DE0-Nano Development board.**

I. Introduction

THE first lab in this introductory series introduces the DE0-Nano development board and Altera's Quartus development environment. Quartus will be used to program the Altera Cyclone IV FPGA on the Terasic DE0-Nano development board.[1] The schematic entry method used in this lab will be very similar to the design entry method that you have experienced with in fiwrst lab with Multisim. Now, instead of discrete logic, the FPGA will implement the logic. The Cyclone IV could implement the equivalent of 3300 2-input nand gates. Imagine hand wiring and the amount of circuit board space all that logic would require if using Discrete ICs. This lab is intended to introduce the student to the following concepts:

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

A. DE0-Nano Development board

The DE0-Nano has a number of peripheral devices built into the board to expand the capabilities of the FPGA. Interacting with most of these devices will be beyond the scope of this course but represent real world design challenges and are worth experimenting with after this course is completed. Altera and Terasic offer a number of tutorials for the periphrial devices though in the DE0-Nano user manual and the Altera University Program. These labs will mostly use one of the 40-pin GPIO headers to interact with the outside world. Let's take a moment to recognise the potential of this development board should you choose to pursue it. Figure 1 shows a high level diagram of the development board. There is much more information available in the DE0-Nano manual but these are some system highlights. The SDRAM can be used to store large amounts of information for fast access, great for a *soft processor* [2]. The EEPROM allows program memory that does not clear with power reset, also very useful for a soft processor. The A/D converter allows analog voltages to be read which allows the development board to interact with the outside world. It also features an accelerometer that will refresh at rates up to 1600Hz. The A/D converter and G-sensor will be great tools to interact with sensors in your later classes. If you choose to go down a path of controls engineering FPGAs will be a great way

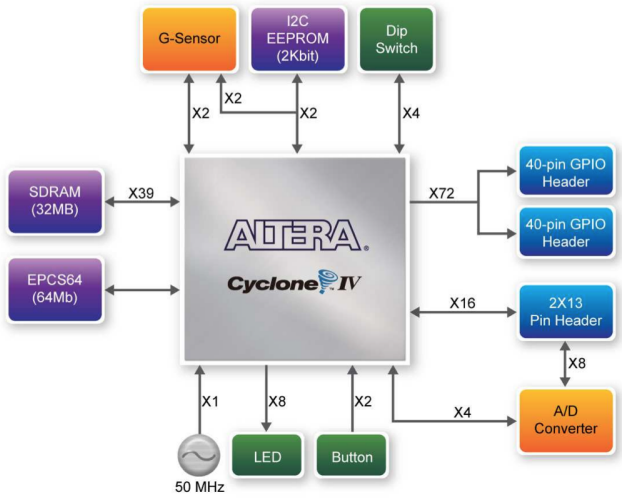


Fig. 1: Block Diagram of DE0-Nano [1]

to implement *deterministic*[3] Control systems. Interfacing with these periphrials will require the implementation of a bus protocol like SPI or I2C. The De0-Nano system CD comes with some example code that uses these devices that would get the interested student off the ground in no time. Take the time to read through the DE0-Nano board's manual.

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. It would behoove you to take a look at the lab report section before starting the procedure. Recording results of experiments will be stressed because the documentation of what you do is the most important part. To begin these labs you will need:

- De0-Nano Development board
- A breadboard
- Four LEDs with 200Ω current limiting resistors
- Two dip switches with 200Ω pulldown resistors
- Windows or Linux based computer
- Internets

A. Getting start with your DE0-Nano

Altera's University program provides excellent tutorials for Quartus. This first tutorial covers installation of drivers for the DE-Series board. This is information will only be pertenent to those installing quartus on their own computer. Installing Quartus

[1] FPGAs remained a mystery to me for quite some time, the video does a great job of discussing the benefits and drawbacks of the devices EEV Blog: What is a FPGA?

[2] A soft processor is a microprocessor implemented inside of the FPGA. These softprocessors can be programmed in languages like C for more standard development processes that typically are much faster than HDL languages. The advantage of this method is the ability to build the microprocessor to suit a particular design and build hardware peripherals with the FPGA

[3] a deterministic system is a system in which no randomness is involved in the development of future states of the system. A deterministic model will thus always produce the same output from a given starting condition or initial state. [2]

B. Lab Part One: LED controller with onboard devices.

The included document Quartus II Introduction shows how to use the graphical editor feature of quartus. Dont worry about section 8 of the Quartus Introduction, we'll cover simulators later. The product of this tutorial will be the first lab demo. Notice the Boxed section below, demos will be presented as a specification and deliverable format. This first Example box describes what the three sections mean and how to use this information to help you successfully demo the lab.

Laboratory Demo: Light Controller

Specification:	Use onboard periphrials to generate inputs for the FPGA to control an LED according to Figure 11 of the Altera tutorial *Quartus II Introduction.*
Deliverable:	Labeled schematic of circuit with I/O pins labeled with FPGA pin number and Verilog net name, truth table from your experimental result.
Process :	The student will be asked the result of a few random numbers to verify the correct operation of their circuit.

C. Lab Part Two: Interacting with external devices though GPIO

A switch and LED are going to be used to test the FPGA while it's operating. This switch will allow you to generate input for the FPGA. Figure 2 LED circuit needs a current limiting resistor. The LED will

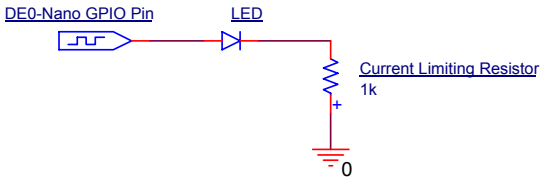


Fig. 2: LED with current limiting resistor

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. CMOS devices tend to float to the high state, so if you ever have a signal stuck high, it's worth looking into the pulldown network if you're using one.

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, a simple mistake like this can cost a sub stantial amount of time. It is easiest to match the Nano's orientation with the schematic and count from the nearest edge. Figure 4 shows GPIO 0 next to a schematic of the header. Always check VCC_SYS, VCC3P3, and GND with a multimeter before attaching a circuit you have built. This is the header pin schematic from the DE0-Nano user manual. Inside the Verilog code, these pins follow a 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.

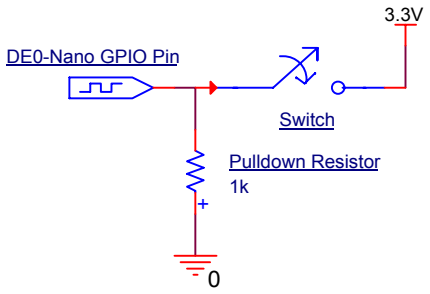


Fig. 3: Switch with pulldown resistor

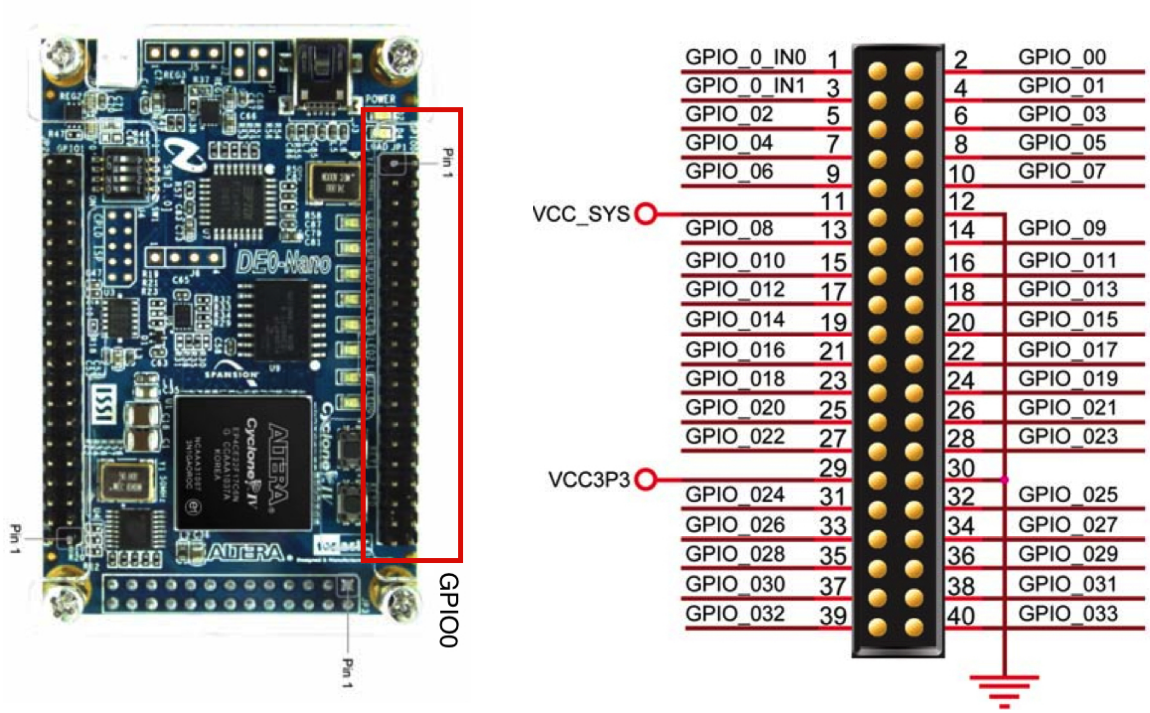
D. Design AOI module from logical equation

The *And or Invert* function is common in digital design. You will implement it with logic primitives in the schematic editor. There will be a wealth of information available online about how to implement the function on the Internet, but try and fail at least once before heading to Google.

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

Laboratory Demo: AOI Gate

Specification:	4 dipswitch inputs and 1 LED outputs with appropriate pulldown and current limiting resistors for the DE0-Nano Development board.
Deliverable:	Labeled schematic of circuit with I/O pins labeled with FPGA pin number and Verilog net name, truth table from your experimental result.
Process :	The student will be asked the result of a few random numbers to verify the correct operation of their circuit.



(a) Orientation of GPIO Header [1]

(b) Schematic of GPIO-0 header [1]

Fig. 4: GPIO0 and its orientation on the development board

III. Lab Report

THE lab report must be typed and submitted in a PDF format to the course's Moodle site. Look to IEEE's guidelines on format. I would reccomend using the word template they provide, it will ensure professional looking documents without much effort. The document should include the following items.

A. Figures to include

- Quartus schematic of your Light Controller module.
- Quartus Schematic of your AOI module.
- Truth tables for both designs.

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. What was the logic usage of your FPGA?
2) The FPGA is based on 3.3 volt logic, what compatability issues do you anticipate with more common 5V devices like the 7400 series logic ICs you were using.
3) This lab uses a graphical design entry method. What advantages do you see in this method, what drawbacks do you anticipate compared to a text based system?

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 devices, offers a tool suite very similar to Quartus called ISE. Although we will not discuss ISE, an understanding of Quartus will allow ISE to be learned very quickly. The tools used in this lab are the what you can expect to see in Industry as a CPE or EEE.

References

[1] TerasIC, *DE0-Nano User Manual*, 1st ed., 2012.
[2] S. Foundation. (2013) Dynamic systems. [Online]. Available: http://www.scholarpedia.org/article/Dynamical_systems