ECSE436 – Signal Processing Hardware

Handout for Lab1: Intro to FPGAs

Prepared By: Dylan Watts & Mike Yuhas

Part 1: Introduction

An FPGA (Field Programmable Gate Array) allows us to implement a digital design in hardware. Unlike a DSP which has an architecture similar to that of a micro-controller, an FPGA allows implementation of combinational logic and memories in circuit-form. This gives FPGAs a greater capacity for parallelization and other problems that do not scale well on traditional computer architectures.

To achieve this, the FPGA is made up logical cells (also called logic elements for Altera FPGAs). Figure 1 shows a sample logic element for the Cyclone II family of FPGAs, the family we will be using. The diagram looks complex, but all we are interested in is the LUT (look up table) and register. The extraneous logic is simply used for routing between logic elements. With these two bocks we can implement combinational logic and memory. To give the FPGA a greater capacity, LEs are wired in a systolic array (which is where the routing logic comes into play). To help boost performance, RAMS and dedicated mutlipliers are scattered around the chip.

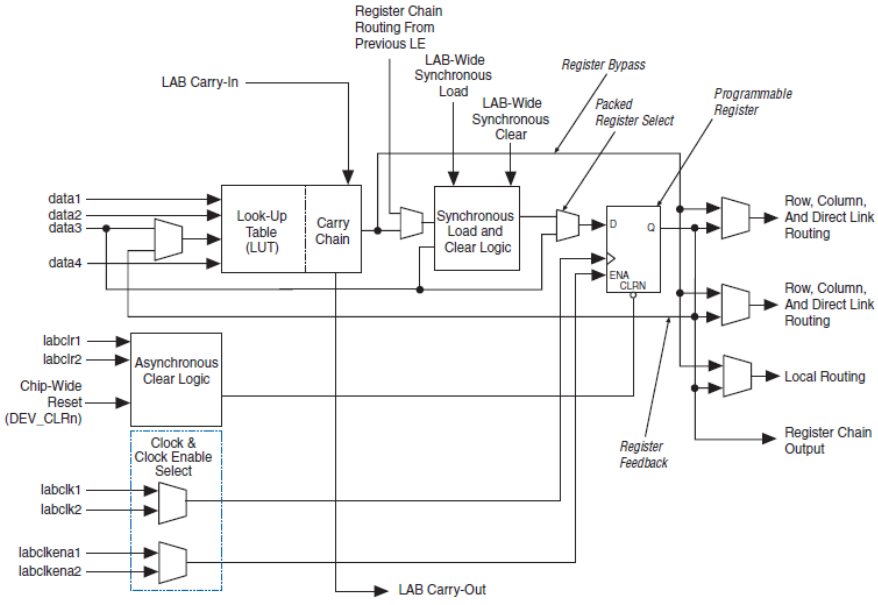


Figure 1: Block Diagram of Cyclone II LE

Part 2: Hello World

To get started, we will write a simple hello world program for the FPGA and then add more complex features. Our first project will allow you to control an LED using a switch on the development board.

1.) Check out the Altera DE2 board from the equipment room and ensure that the box contains all the necessary pieces. You should have: DE2 board (1x), Power Supply (1x), Usb A – USB B cable.

2.) Search for Quartus II in the programs list, we will be using Version 13.0 32-bit. Once the program loads click on file -> new project wizard. Here you will be presented with multiple steps to set up a new project.

3.) Click next to the Introduction, now enter the name and the path for your project. Here are a few tips to avoid headaches later on:

- Create a sub folder in the C:\altera\13.0sp1\(insert new folder here). It is an easy way to keep track of your project files and **the project will not compile unless it is in the C:\ drive.**
- Make sure the path is less than 32 characters
- Do not have spaces in the name of your project

(Note: since your programs are saved on the C:\ Drive, it is specific to **that computer**. In order to access it on a different computer, you must save it on an alternate drive or a USB stick, and load the files into the C:\altera\13.0sp1\ folder of that computer. It is way easier just to use the same computer for your labs, so choose wisely…)

4.) Hit next and next again. We must now configure the program to the particular board we are using, specifically the type of FPGA chip on your board (the big one that says "ALTERA" on it). This allows the compiler to accurately route the pins of the elements on your board:

- Select the **Cyclone II** family of devices.
- Find **EP2C35F672C6** from the scroll down list.

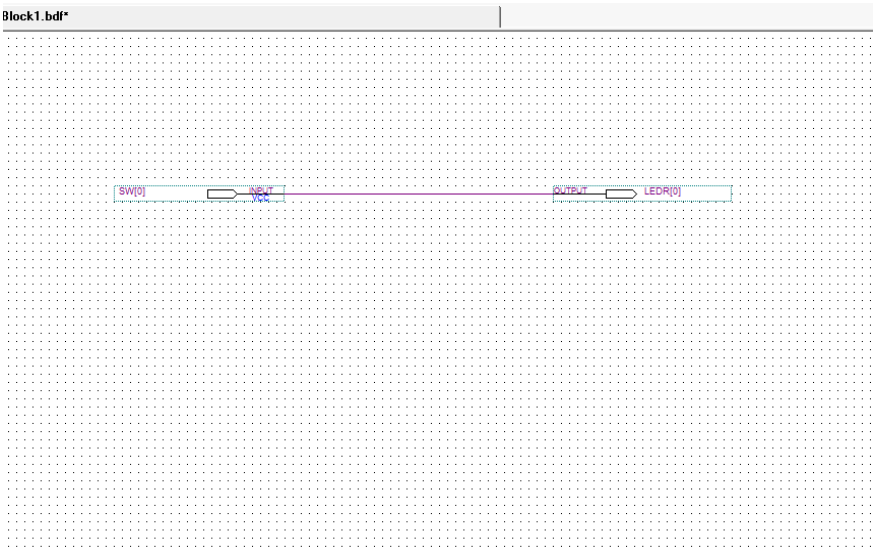
5.) Click next and then finish

6.) Now that our project is done, we will start to create our design. Press file->new and choose block Diagram/Schematic File

7.) Now let's add some inputs and outputs: click on the AND gate icon on the upper tool bar. This is where you will find standard logic components built into the program for your designs.

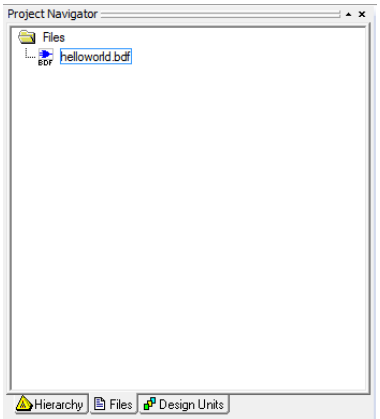
- A window will pop up, and in the Libraries file browser expand the C:/altera/13.0sp1/quartus/libraries folder. Expand primitives and then pin. Select input and press okay. Place the input pin icon on the grid. Press escape to exit place part mode. Double click the input pin and rename it to SW[0], leave the default value as VCC.
- Next add an output from the same symbol dialog box you used to place an input, this time select an output pin. Place it some distance to the right of your input and press escape to exit place pin mode. Double click on the pin and change its name to LEDR[0].

8.) Next select the mouse pointer icon on the upper tool bar and connect a wire from the input to the output. Can you guess what our design will do?



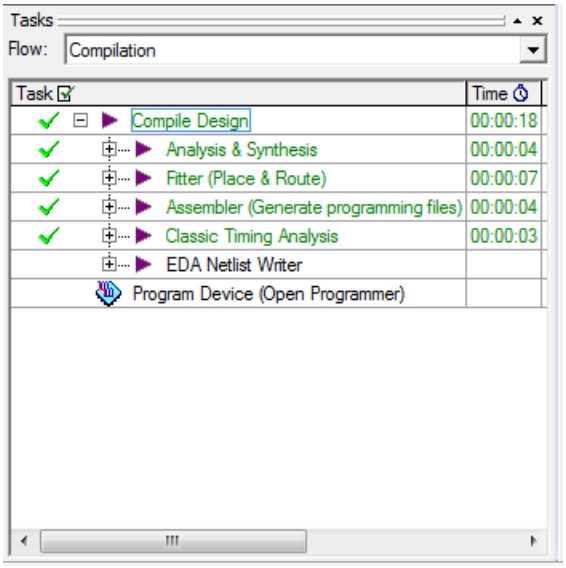
This is what your design should look like so far

9.) Hit CTRL+S to save, call it helloworld.bdf and save it in the directory with your project file. In the project navigator window select Files and right click on helloworld.bdf. Click Set as Top-Level Entity.



- Another tip – **make sure that the file name does not have spaces, it will not compile.**

10.) Now press the purple play arrow at the top of the screen to compile and synthesize the schematic. If you did everything right a green check mark should appear in front of Analysis & Synthesis, Fitter (Place & Route), Assembler (Generate programming files), and Classic Timing Analysis.



The ideal scenario

As you have seen in the Lab 0 demo, you will get lots of warnings. This is OK.

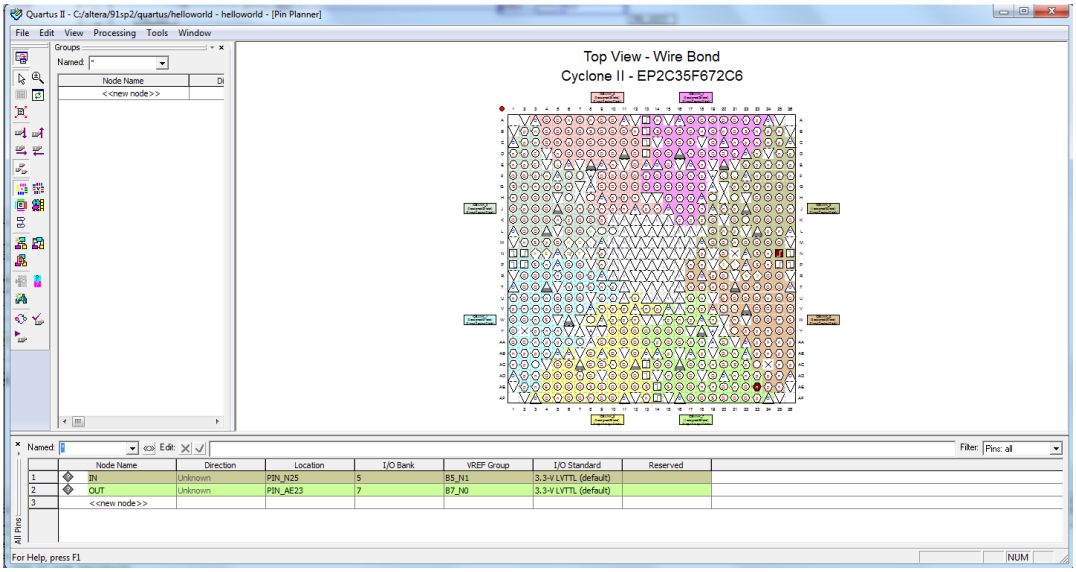
If you have a critical error then instead of a green checkmark a red X will appear. This is bad… but don't worry:

There could be many reasons for this happening, and it could be as small as putting a space in your design file name. See what the error message is, and if it's unfamiliar ask a classmate to see if they have had the same problem. Trust me, they will become invaluable resources later on. If you are still having trouble, come see me and we can sort it out.

11.) Now we need to assign which physical device pins will correspond to the input and output on our design. Click on the pin editor icon on the tool bar. A new screen should pop up

This is the pin planner, the compile button is the purple play button to the right.

12.) After the design has compiled, your input and output pins should show up in the bottom left hand corner. In the pin editor, you can click on the device pins in the picture to map them to a pin on the schematic. An easier way is to search for the Altera Cyclone II Starter Development Board Reference Manual, which lists which pins are associated with which components (this will be important later on when searching for Clocks). Double click the under the Location column of the pins, and in the drop down menu set the input to N25 and the output to AE23. You can then close the pin planner window.



The pin planner window, with the input and output pins shown in the bottom.

13.) After closing the pin planner window you should see that the input and outputs are now labelled with the pins you selected. If they are correct recompile the design.

14) If you haven't already, plug in the power adapter to your board at the upper left hand port, and plug in the USB cable to the port right next to it labelled "Blaster" (the other port is an input for devices with a serial connection like an Arduino).

After plugging the other end of the USB into the computer, the device driver should start to download. If it doesn't download properly, come see me.

There is a switch to the left of the LCD display with two options "RUN" and "PROG". **Make sure that the switch is set to "RUN"** or else the device will not load the program.

Finally push the big red button and watch your board light up (show it to your arts friends and say you made it do that, they'll be impressed)

14.) Under all of the compile checkmarks, double click on the program device button and select Hardware Setup. Under "Currently selected Hardware" you should see USB-Blaster [USB-0]. If you do not then your driver did not install correctly and you should come see me.

Once your device is loaded, select “Add File”. After compiling, a list of new folders should have appeared in your folder. Double click on the “output_files” folder, and select the **(projectname).sof**

You should now be able to click the “Start” button to load your design. The lights on the board should blink on and then off. Now try flipping the slide switch 0 on the board and see what happens!