
RUTGERS, THE STATE UNIVERSITY OF NEW JERSEY

ECE493 SPECIAL TOPICS

# Hardware/Software Design of Embedded Systems Laboratory

Fall 2013

# Contents

# 1 Lab 1 - Introduction to FPGA's and VHDL

## 1.1 Introduction

This lab will introduce you to the Altera DE2-115 FPGA Development Board. The DE2-115 contains all of the hardware necessary to prototype and create various hardware configurations on the Altera Cyclone IV FPGA chip that will be used throughout the course of this lab. By completing this lab, you will have an understanding of all the hardware contained on the FPGA development board, along with an understanding of how to connect peripherals to the development board. Lastly, this lab will go over the standard template for designing hardware in the VHDL programming language. All this will be accomplished by following the Quartus II introductory packet along with the following activities.

## 1.2 Pre-Lab

## 1.3 Working with Quartus II

### 1.3.1 Creating a New Project

File > New Project Wizard

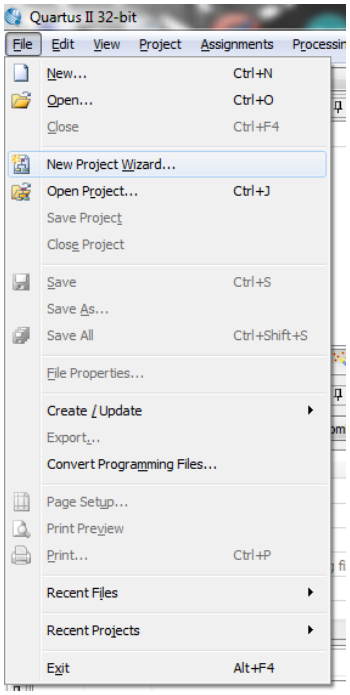



Figure 1: Create a new project menu

NEXT
Create a folder in your Z-drive called FPGA Lab
Create a folder in FPGA Lab called lab1
Set the working directory to lab1
Name the project to lab1
NEXT
Skip this step, NEXT
Under Device Family select Cyclone IV E, set package to FBGA, pin count to 780, and speed grade to 7. in the Available devices list, look for and select EP4CE115F29C7

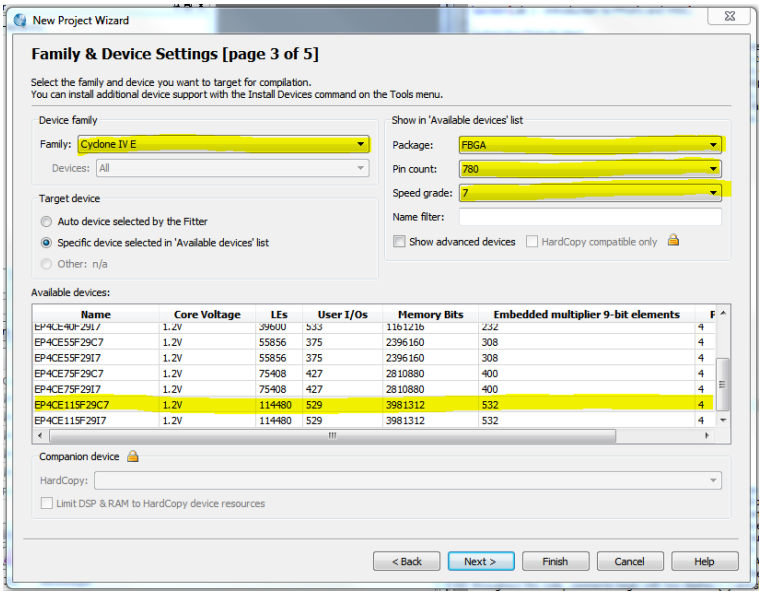



Figure 2: Device selection menu

FINISH
File > New > VHDL File > OK

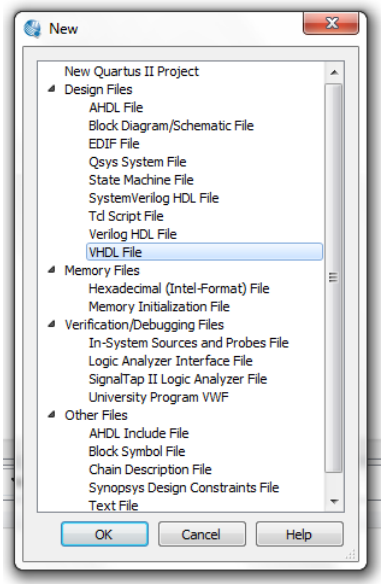



Figure 3: Create a new file menu

File > Save As > *lab1.vhd* > SAVE

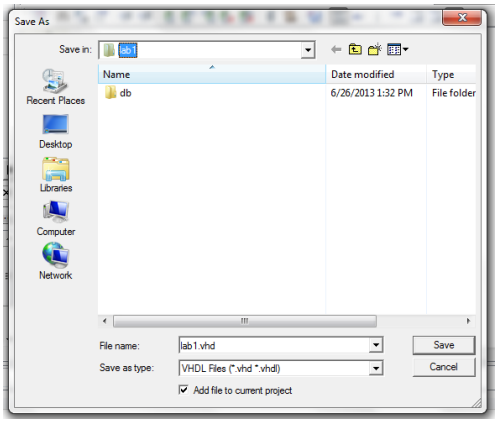



Figure 4: Save As dialog box

### 1.3.2 VHDL Basics

The following code block shows how to interact with the switches and LEDs on the DE2-115. Notice how the program begins with importing the ieee library which contains all of the basic logic primitives as established within the IEEE standard 1164. When working in industry it is common for large companies to create their own libraries as well. Every VHDL file should contain at least one entity (module) that is the same as the name of the file. An entity contains information about the structure of the module such as how many inputs/outputs (I/O) and what type of logic to expect at the I/O. Finally we define the entity in an architecture block, this section does the work on the hardware. As can be seen, this code is setting the red LEDs as defined in the array to the accompanying switches on the board. Take note on the use of comments throughout the code, comments begin with two dashes (–) and should always be used to describe what you are trying to accomplish, this way someone else who reads your code will understand it easily and your code will look more professional.

```
1  -- Import logic primitives
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4
5  -- Simple module that connects the SW switches to the LEDR lights
6  ENTITY lab1 IS
7  PORT ( SW: IN STD_LOGIC_VECTOR(17 DOWNTO 0); -- Initialize switches as an input
8       LEDR: OUT STD_LOGIC_VECTOR(17 DOWNTO 0)); -- Initialize red LEDs as an output
9  END lab1;
10
11 -- Define characteristics of the entity lab1
12 ARCHITECTURE Behavior OF lab1 IS
13 BEGIN
14       LEDR <= SW; -- Assign each switch to one red LED
15 END Behavior;
```

### 1.3.3 Setting Pin Assignments

Assignments > Import Assignments > Select ... *.qsf* ADVANCED > Check Global Assignments > Ok

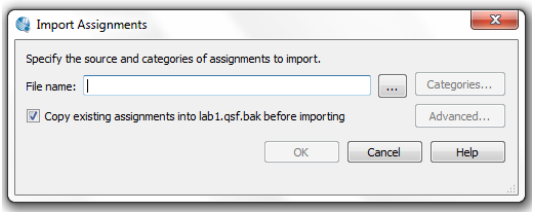



Figure 5: Import assignments dialog box

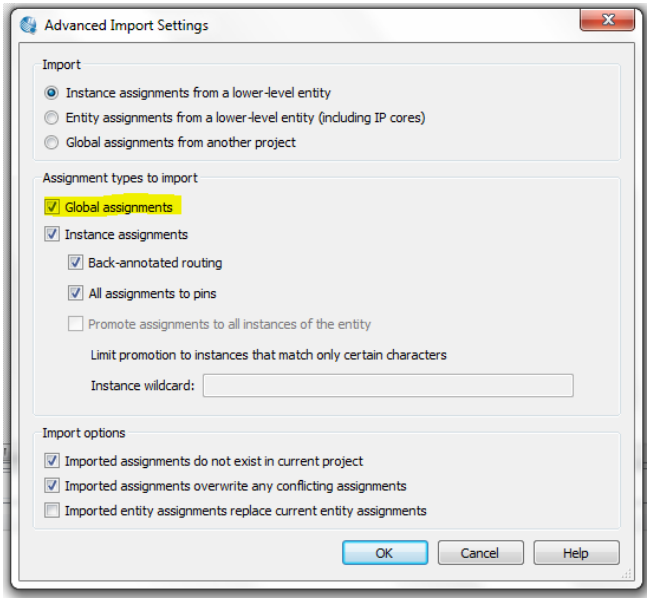



Figure 6: Import assignments advanced options menu

### 1.3.4 Compiling Hardware

Processing > Start Compilation or Ctrl + L
If the project compiles succefully, you may proceed to uploading the hardware. Otherwise if you have any errors you should debug your code. It's helpful to note that the first error should be solved first which will make it easier to solve the other errors.

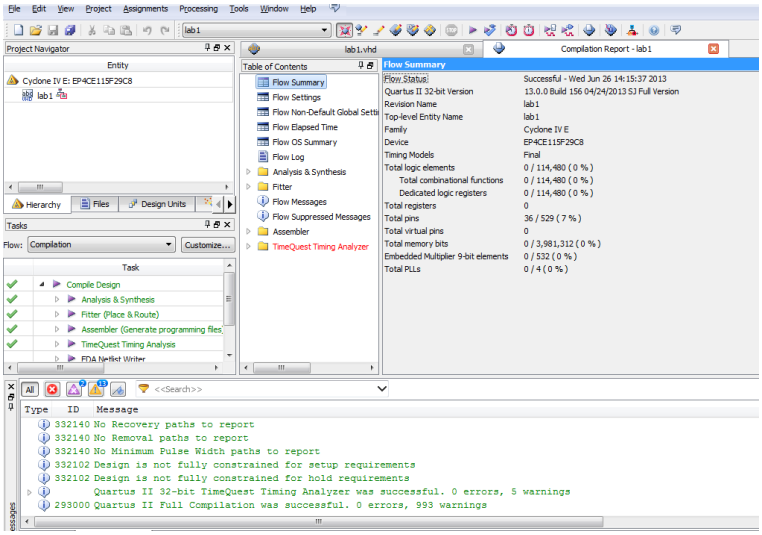



Figure 7: Succeful completion of compilation

### 1.3.5 Uploading Hardware to Device

## 1.4 Activities

### 1.4.1 Implementing Logic

Implement the hardware from the circuit in Figure 8. The inputs should come from SW(1) and SW(2) and the output should be shown on any of the available LEDs. Use the implemented circuit to test and create a truth table with your results and place it within a comment in the program file.

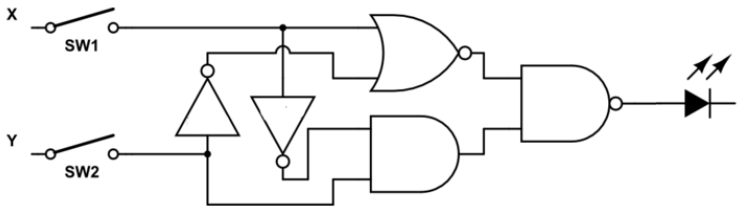
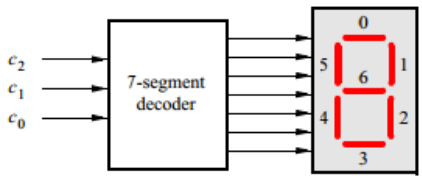



Figure 8: Circuit for activity 1

### 1.4.2 7 Segment Display Decoder

The 7-segment display is comprised of 7 LEDs that are arranged in such a way that allows for the creation of the numbers 0-9 and a select few characters with some clever use. Figure 9 shows the block diagram and output table. Your task is to create a 3 input, 7 output decoder that will display a number from 0-6. To accomplish this task, you should program the switches SW(0) - SW(6) to make the first 7 displays show the numbers 0-6 when its switch is turned on.




| $c_2c_1c_0$ | Character |
|---|---|
| 000 | 0 |
| 001 | 1 |
| 010 | 2 |
| 011 | 3 |
| 100 | 4 |
| 101 | 5 |
| 110 | 6 |
| 111 | |

Figure 9: 7 segment display and decoder

**Tips:**

- The eight 7 segment displays can be accessed with the 7-bit signal vectors HEX0...HEX7. For example, to output to the first display (HEX0) you can either set each bit individually (HEX0(5) <= '1';) or set the whole vector with (HEX0 <= '1111111') which would display the number 8.
- more to come...

# 2 Lab 2 - Latches, Flip-Flops, and Counters

## 2.1 Introduction

## 2.2 Lab Activities

### 2.2.1 Latches

Activity on SR Latch

### 2.2.2 Flip-Flop

Activity on JK

### 2.2.3 Counters

Create a counter that can increment and decrement with the use of a switch and pushbuttons.....

# 3 Lab 3 - Adders, Subtractors, and Multipliers

## 3.1 Introduction

## 3.2 Lab Activities

### 3.2.1 Half Adder

Build the following Half adder circuit

### 3.2.2 Full Adder / Subtractor

Expand on the half adder to create a full 8 bit carry adder. How would you implement a subtractor?

### 3.2.3 Multiplier

Follow this multiplier circuit.....

# 4 Lab 4 - Finite State Machines

## 4.1 Introduction

## 4.2 Lab Activities

### 4.2.1 FSM

### 4.2.2

# 5 Lab 5 - A Simple Computer

## 5.1 Introduction

16bit simple processor

## 5.2 Lab Activities

### 5.2.1 Design an ALU

Add, Sub, Mult, Shift, XOR, AND, NAND,

### 5.2.2 Design a RAM

512 bit memory

### 5.2.3 Design the Program Counter

simple counter

### 5.2.4 Create a VGA Driver

Display the output onto the screen

# 6 Lab 6 - Audio Synthesizer

## 6.1 Introduction

## 6.2 Lab Activities

### 6.2.1 Pitch Control

### 6.2.2 Frequency Modulation

### 6.2.3 Custom 8bit Music

# 7 Lab 7 - Ethernet Exploration

## 7.1 Introduction

## 7.2 Lab Activities

### 7.2.1

### 7.2.2

# 8 Final Project

## 8.1 Introduction

## 8.2 Requirements

## 8.3 Project Ideas

## 8.4 Deliverables