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 The DE2-115 FPGA Development Board

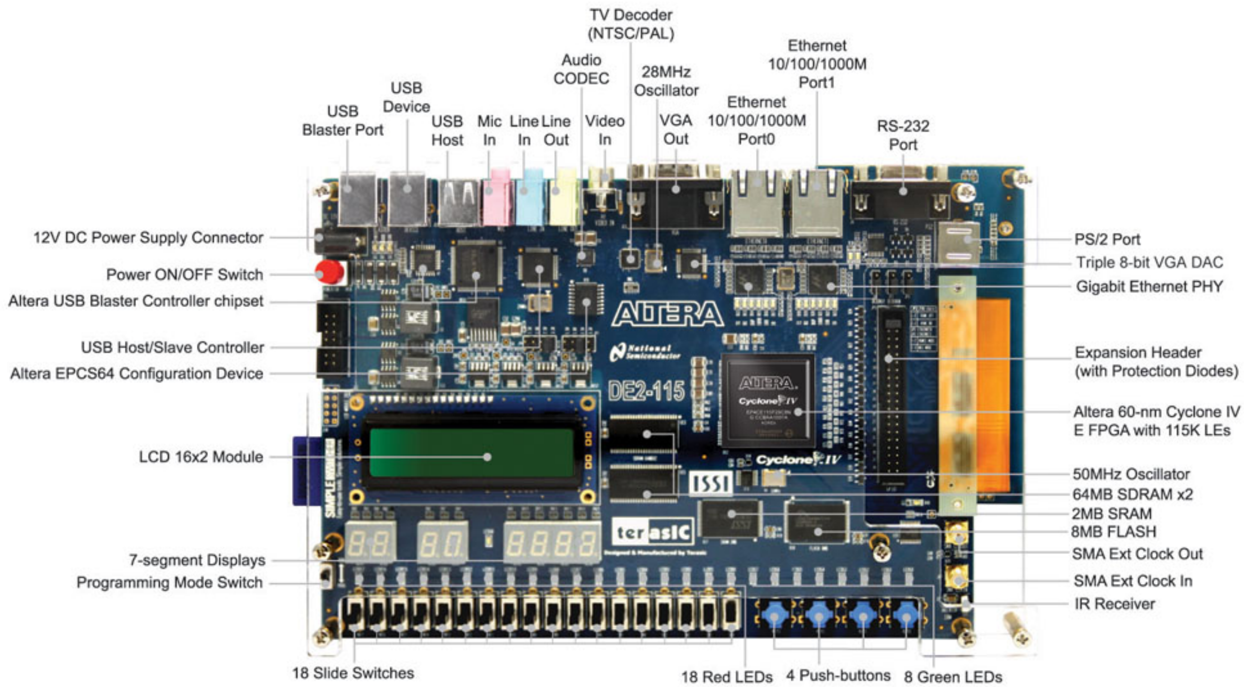


Figure 1: The DE2-115 FPGA Development Board

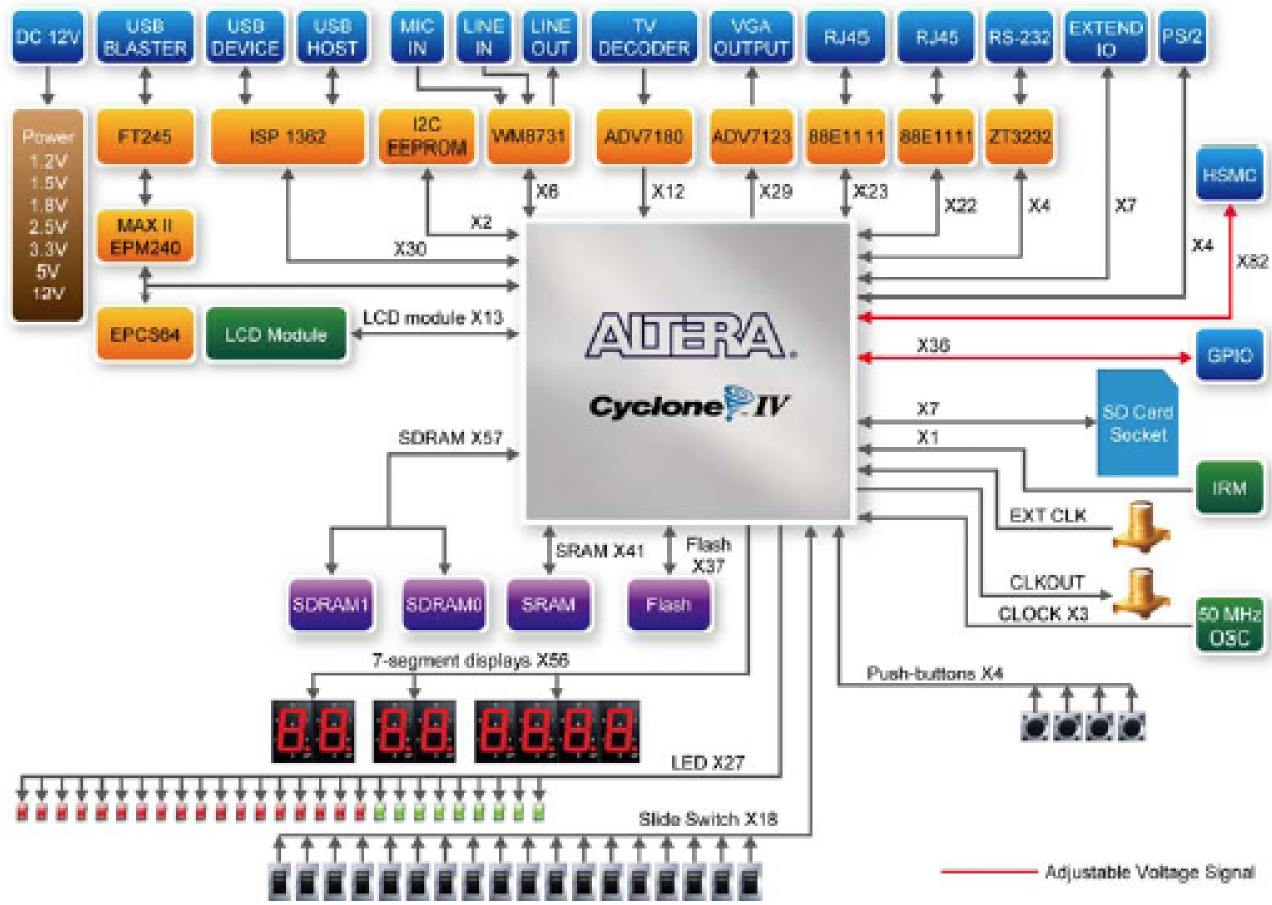


Figure 2: The DE2-115 Block Diagram

1.3 Pre-Lab

Before attempting this lab, you should complete the following:

- Download and install the *Quartus II 13.0 Web Edition Software* from the Altera website to your personal computer
- Download and read the following documents from the Sakai Resources pags; *DE2-115 User Manual, Quartus II Introduction*

1.4 Working with Quartus II

1.4.1 Creating a New Project

1. In order to begin working on your first FPGA project, you must open the program Altera Quartus II. To begin a new project goto **File → New Project Wizard** as shown in figure 3

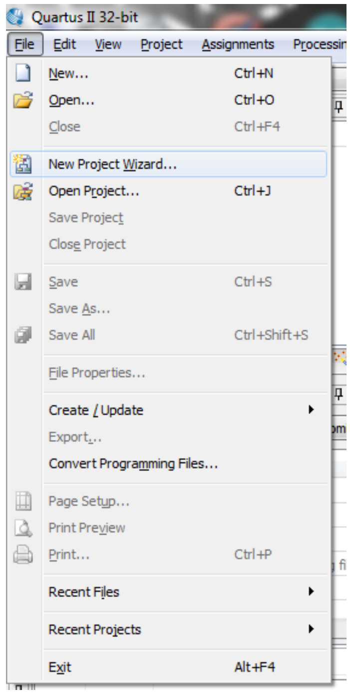
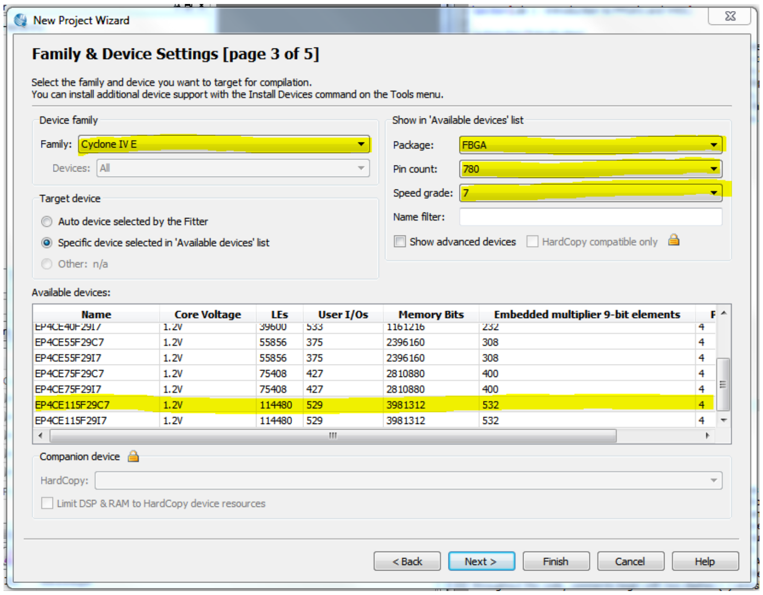


Figure 3: Create a new project menu

2. When the *New Project Wizard* window opens click **NEXT**
3. Create a folder in your Z-drive called *FPGA Lab*
4. Create a folder in FPGA Lab called *lab1*
5. Set the working directory to lab1
6. Name the project *lab1*
7. Click **NEXT** to proceed
8. Skip this step, click **NEXT** to proceed
9. Under Device Family select *Cyclone IV E*, set the package to *FBGA*, pin count to *780*, and speed grade to *7*. In the Available devices list, look for and select *EP4CE115F29C7* as shown in figure 4.



Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements	P
EP4CE40F29I7	1.2V	39600	533	1161216	232	4
EP4CE55F29C7	1.2V	55856	375	2396160	308	4
EP4CE55F29I7	1.2V	55856	375	2396160	308	4
EP4CE75F29C7	1.2V	75408	427	2810880	400	4
EP4CE75F29I7	1.2V	75408	427	2810880	400	4
EP4CE115F29C7	1.2V	114480	529	3981312	532	4
EP4CE115F29I7	1.2V	114480	529	3981312	532	4

Figure 4: Device selection menu

10. Click **FINISH** to begin your new project

1.4.2 Creating an Empty File

1. Goto **File** → **New** → **VHDL File** → **OK** as shown in figure 5

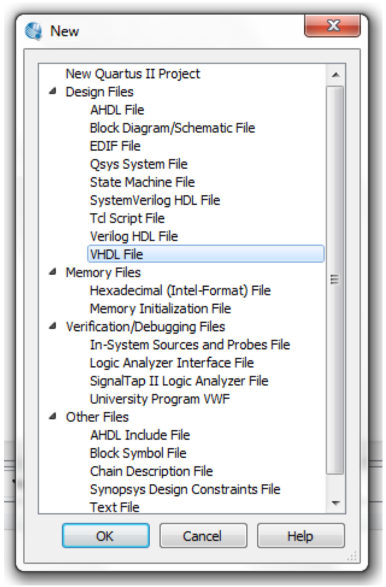


Figure 5: Create a new file menu

2. Save this new file by going to **File** → **Save As** → ***lab1.vhd*** → **SAVE** as shown in figure 6

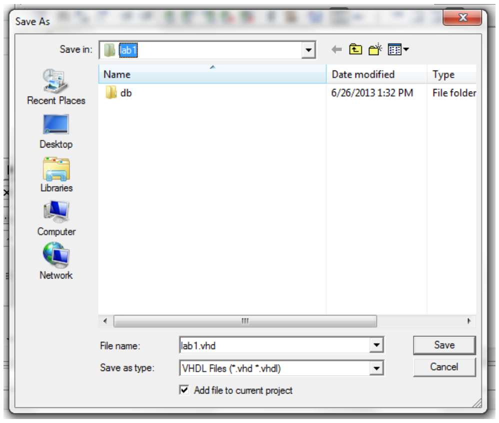


Figure 6: Save As dialog box

1.4.3 Writing VHDL Code

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.

```vhdl
-- Import logic primitives
LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Simple module that connects the SW switches to the LEDR lights
ENTITY lab1 IS
PORT ( SW: IN STD_LOGIC_VECTOR(17 DOWNTO 0); -- Initialize switches as an input
     LEDR: OUT STD_LOGIC_VECTOR(17 DOWNTO 0)); -- Initialize red LEDs as an output
END lab1;

-- Define characteristics of the entity lab1
ARCHITECTURE Behavior OF lab1 IS
BEGIN
     LEDR <= SW; -- Assign each switch to one red LED
END Behavior;
```

1.4.4 Setting Pin Assignments

The Cyclone IV chip on the DE2-115 contains 780 pins. The majoprity of these pins have copper traces to the hardware the control on the development board. As a result, Altera has provided a file that contains all of the pin assignments so that you can interface directly with the pins in your VHDL code. To set the pin assignments:

1. Goto **Assignments → Import Assignments → Select *D*E2-115.qsf** as shown in figure 7.This file can be dowloaded from the Altera website

2. Then click **ADVANCED → check Global Assignments → Ok** as shown in figure 8

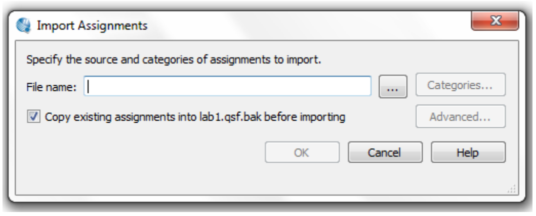


Figure 7: Import assignments dialog box

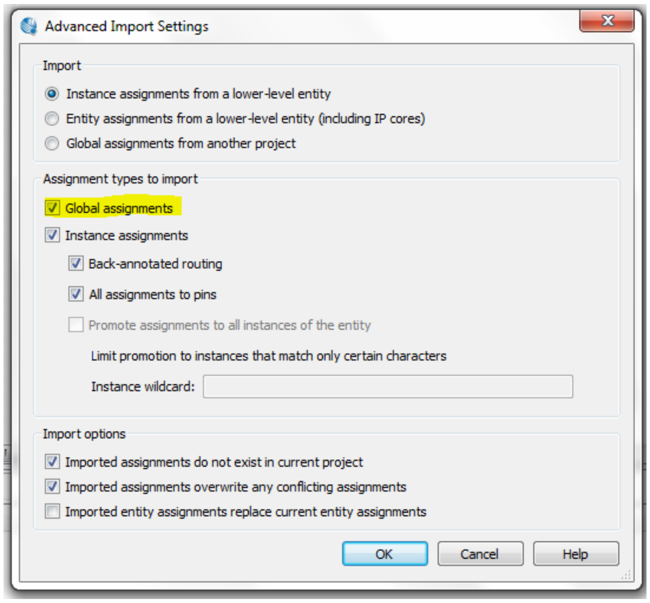


Figure 8: Import assignments advanced options menu

1.4.5 Compiling Hardware

1. to begin compiling your hardware, **Processing → Start Compilation** or you may press *Ctrl + L* on your keyboard

If the project compiles successfully, you may proceed to uploading the hardware. Otherwise if you have any errors you should debug your code. It is helpful to note that the first error should be solved first which will make it easier to solve the other errors. A succefull compilation will look like figure 9.

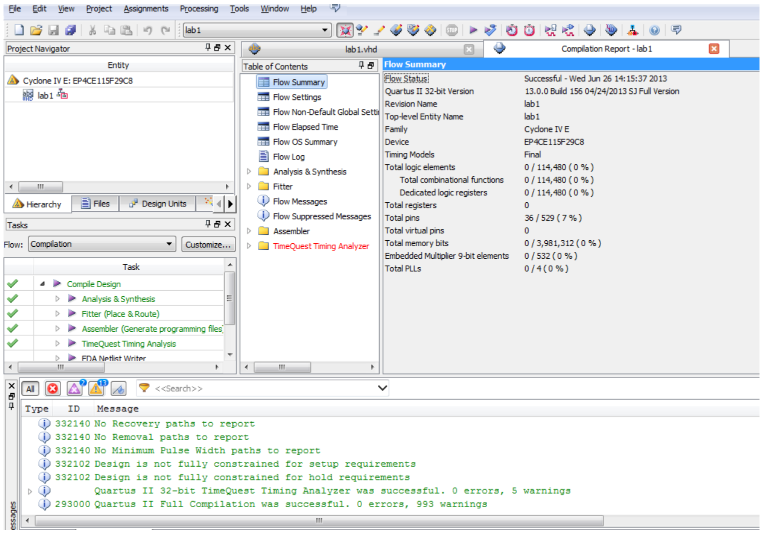


Figure 9: Successful completion of compilation

1.4.6 Testing Hardware with Waveforms

1.4.7 Uploading Hardware to Device

Once the hardware has compiled successfully, goto **Tools** → **Programmer** to open the hardware upload options. There are two typical modes for uploading hardware and it is important to understand when to use them. This can be seen in figure 10.

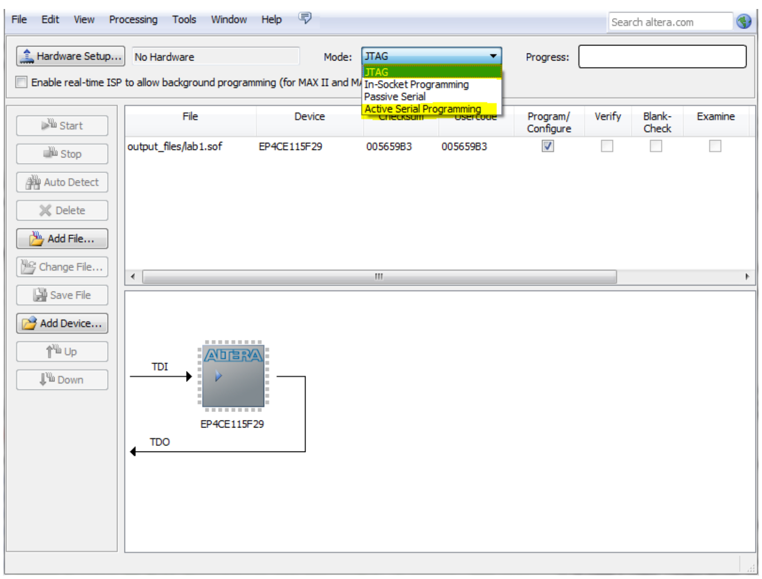


Figure 10: Modes for uploading hardware to the FPGA device

1. JTAG or Joint Test Action Group

   - This method loads the hardware directly to the FPGA chip
   - The FPGA is unable to save its current state so if the power is turned off the programmed hardware will disappear

- To program the FPGA with this method all you need to do is connect the USB cable to the development board and ensure that under **Hardware Setup** that USB-Blaster is selected. Then you must goto **Add Files** and add your compiled *lab1.sof* file
- Press **START** to upload your hardware, in a few moments you should see your development board behaving as instructed by your code

2. Active Serial

- This method loads the hardware on to the on-board configuration device. What this means is that the hardware description is saved into memory and is loaded onto the FPGA chip whenever the board is powered on. This method is more desirable because it allows the FPGA to work without being connected to the computer and only to an external power source.
- Before beginning this method you should first check that under **Assignments** → **Device** → **Device and Pin Options** are configured in the same way as figure 11, if not you must set the configuration scheme to *Active Serial* and also set the configuration device to *EPCS64*. If this was not set you must recompile your hardware.

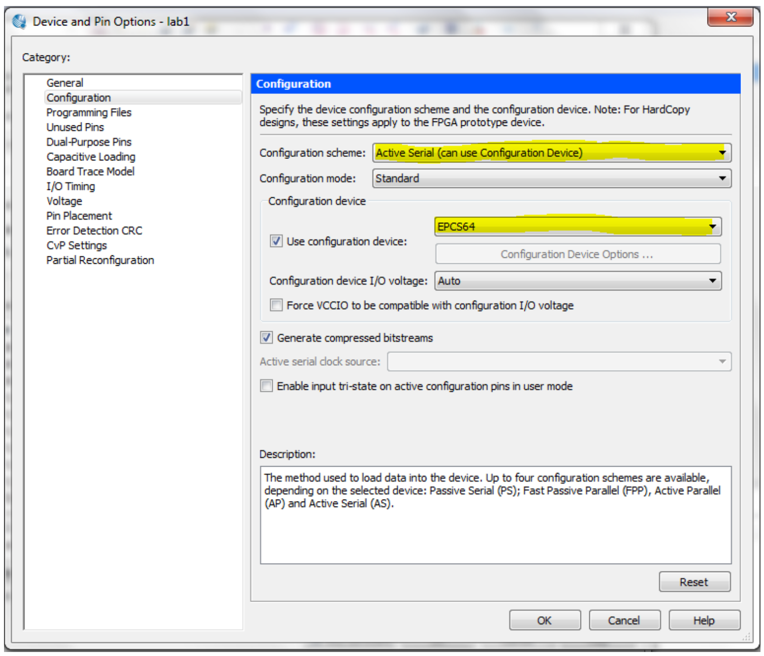


Figure 11: Active Serial Configuration Settings

- Next once again ensure that the hardware is set to *USB-Blaster* and that the mode is set to *Active Serial*
- Click on **Add Files**, and select *lab1.pof*
- Ensure that the development board is switched to *PROG*
- Click **START** to begin programming. This method takes slightly longer
- Switch the board back into the *RUN* position and verify that your logic is behaving properly

1.5 Activities

1.5.1 Implementing Logic

Implement the hardware from the circuit in Figure 12. 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.

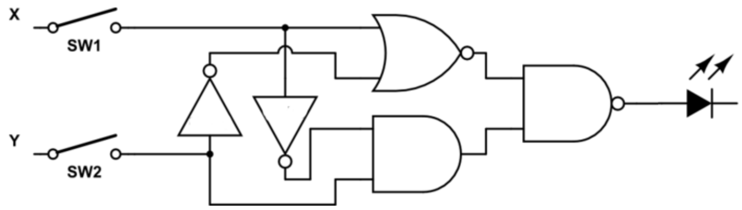
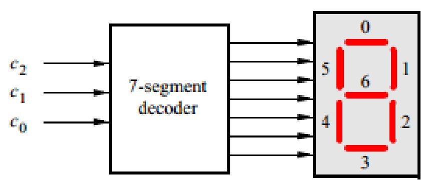


Figure 12: Circuit for activity 1

1.5.2 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 13 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 13: 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 Final Project

6.1 Introduction

6.2 Requirements

6.3 Project Ideas

6.4 Deliverables