University of California, Davis
Department of Electrical and Computer Engineering

EEC180   **DIGITAL SYSTEMS II**   **Spring Quarter 2023**

Lab 1: Questa/Modelsim/Quartus Tutorial

Objective:
This tutorial/lab covers the complete design flow for implementing a high-level Verilog design on the DE10-Lite board. The tools covered in this tutorial include System Builder, Quartus Prime and Questa/ModelSim. Part 0 uses a simple design example to illustrate the design flow. In Part I, you will then practice implementing simple logic gates on the DE10-Lite board. In Part II, you will design and implement a combinational logic circuit to display a 4-bit input in decimal on 7-segment displays.

Resources:
The Terasic webpage for the DE10-Lite board contains links to information and resources for the board. These resources include:
- DE-10-Lite User Manual, Version 1.4, Release Date 01-24-2017
- DE10-Lite CD-ROM, Version 2.0.0, Release Date 01-24-2017

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=218&No=1021&PartNo=4

Prelab:
We strongly recommend you install the tools on your local machine. While there are options to use the ECE department machines, you will not be able to program your hardware outside of the lab. If you are a Mac installing the tools may be difficult, but it can be done using a virtual machine (ideally running Windows). The setup instructions below include directions for Mac, Windows, and ECE department machine users.

DE-10 Lite Boards:
Your DE10-Lite kit will be distributed during the first lab session. Your kit will include a DE10-Lite board, and USB-Blaster cable. If you notice any inactive LEDs, or other components that fail throughout the quarter, please swap your board by notifying a TA.

Installing the Quartus and Questa/ModelSim Tools on your Personal Computer:
The preferred configuration this quarter is Intel® Quartus® Prime 22.1.1 for design and Questa* - Intel® FPGA Edition (a full-featured version of ModelSim) for simulation. The following guide describes how to install the tools on your local machine. We strongly recommend you locally install the tools. The next subsection discusses an alternative if you are not able to do so.

Setup guide: https://canvas.ucdavis.edu/files/20129280/download?download_frd=1

Kemper 2110 Lab Machines:
The machines in the lab run Windows and include Quartus 19.1 with Modelsim. These machines can be used during the lab session to run your labs. However, they cannot be accessed remotely.

Remotely Accessing Quartus and Questa/ModelSim Tools from ECE Department Machines:
If you can't install the tools on a local machine due to performance issues or an unsupported OS, you can use the lab machines to work on your labs remotely. **You will not be able to program your development kit remotely**. To access the department machines, you can use this guide:

Setup guide: https://canvas.ucdavis.edu/files/20129281/download?download_frd=1

Note: Questa is the full-featured version of ModelSim that includes many additional features for advanced verification. Questa was recently included in the free release of Quartus Prime Lite. For the purposes of this course, there is no difference between ModelSim and Questa, so installing either tool will work.

Part 0 – Design Flow Using System Builder, Quartus Prime and Questa/ModelSim

A. Creating Project Files Using System Builder (Windows only)

System Builder is a Windows-based utility that is included in the DE10-Lite CD-ROM and is described in Chapter 4 of the User Manual. This tool generates Quartus Prime project files and does the pin assignment mapping for the I/O pins on the DE10-Lite board. If you are using Linux machines from the ECE department, the setup script described in the above tutorial is a substitute for this program.

- Run **DE10_Lite_SystemBuilder.exe**

- Specify the project name and the I/O devices that you will use in your project.

For example, for this tutorial specify the Project Name as **majority** and select the LED, Button, Switch, and 7-Segment I/O devices, as shown in Figure 1. Then click the Generate button to save the project files to your project folder. Once you have successfully generated your Quartus Prime project files, exit System Builder.
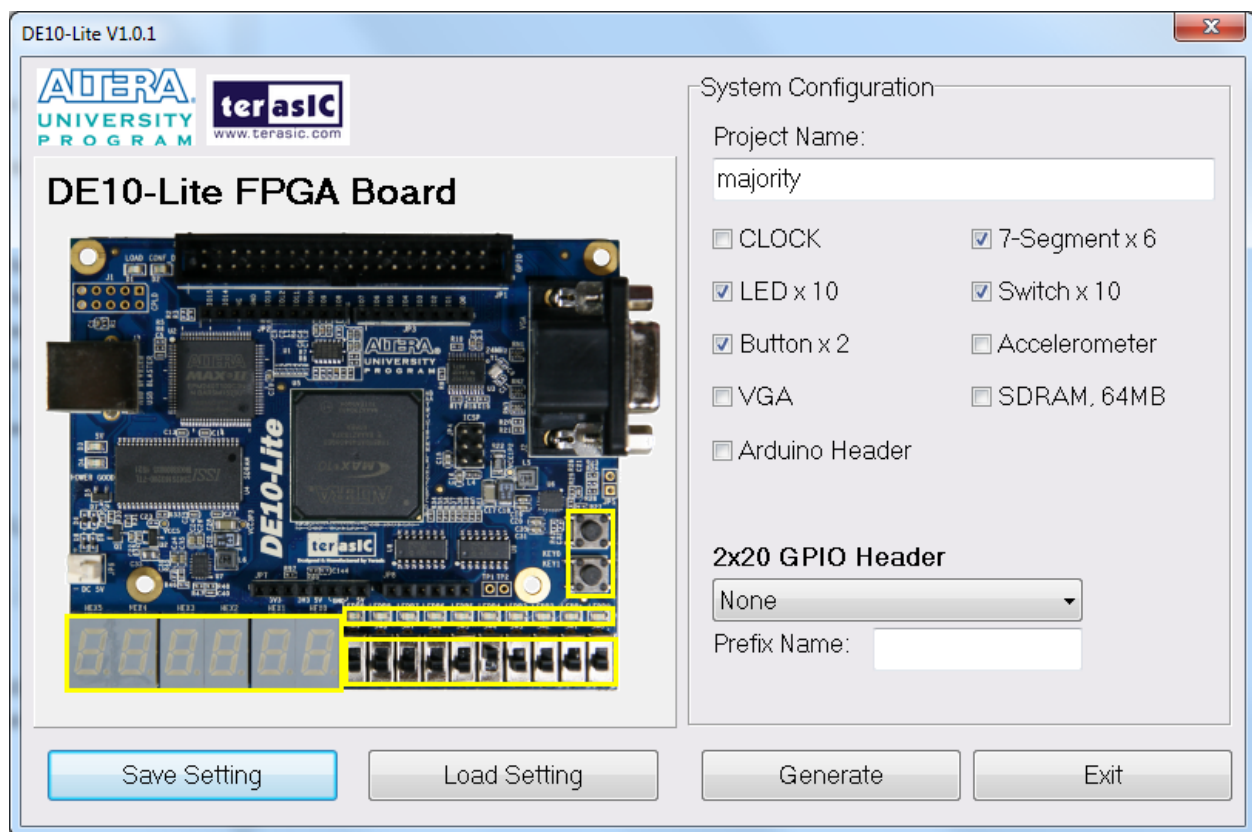
Figure 1. System Builder Configuration

System Builder will create a set of files in your project directory, as shown in Figure 2. You will need to be concerned with only a few of these files in this lab.

Name	Date modified	Type	Size
majority.htm	4/6/2017 10:15 AM	Chrome HTML Do...	12 KB
majority.qpf	4/6/2017 10:15 AM	QPF File	1 KB
majority.qsf	4/6/2017 10:15 AM	QSF File	10 KB
majority.sdc	4/6/2017 10:15 AM	SDC File	3 KB
majority.v	4/6/2017 10:15 AM	V File	1 KB

Figure 2. Files Generated by System Builder

The majority.v file is the template for your top-level Verilog design file. The template generated by System Builder is shown in Figure 3. System Builder maps the I/O devices with the proper pin assignments that are hardwired on the DE10-Lite board. This is a great convenience so that you don't need to manually do the pin mapping. This is one of the biggest advantages of using System Builder to create your project file and Verilog template file. Notice that the declarations and structural coding sections are left blank for you to complete.

```
//=========================================================
//  This code is generated by Terasic System Builder
//=========================================================

module majority(

    ///////////// SEG7 //////////
    output              [7:0]       HEX0,
    output              [7:0]       HEX1,
    output              [7:0]       HEX2,
    output              [7:0]       HEX3,
    output              [7:0]       HEX4,
    output              [7:0]       HEX5,

    ///////////// KEY //////////
    input               [1:0]       KEY,

    ///////////// LED //////////
    output              [9:0]       LEDR,

    ///////////// SW //////////
    input               [9:0]       SW
);



//=========================================================
//  REG/WIRE declarations
//=========================================================




//=========================================================
//  Structural coding
//=========================================================



endmodule
```

Figure 3. System Builder Verilog Template (majority.v)

In this example, we will illustrate the design of a majority gate, as described in Section 3.6 of your text, Digital Design – A Systems Approach by Dally. We will use switches for the inputs and an LED for the output.

B. Compiling and Programming Using Quartus Prime

- Run Quartus Prime and open the project file that you just created using System Builder.

- Select File > Open Project and navigate to the folder where you stored your files. Select the majority.qpf file.

- Double-click the majority entity instance in the Project Navigator pane in order to open the template created by System Builder for your top-level design.

- Add the following line of code in the Structural coding section of the majority.v file:

assign LEDR[1] = (SW[0]&SW[1]) | (SW[0]&SW[2]) | (SW[1]&SW[2]);

As described in lecture, posted notes, and your text (p. 50), Verilog uses the symbols &, |, ^, and ~ to represent the logic operations AND, OR, XOR, and NOT, respectively. The keyword assign is used to describe a combinational logic function. In this case, LEDR[1] will be asserted (logic high) when at least 2 of the 3 switch inputs, SW[0]-SW[2], are high.

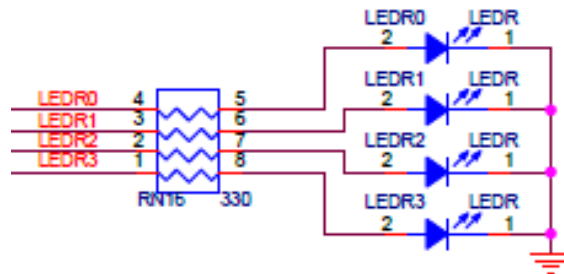Note that the LEDs are active high, as shown in Figure 4.

Figure 4. Excerpt from DE10-Lite Schematic Showing Active-High LEDs

- Once you have entered your code, click Processing > Start Compilation. The design should compile with 0 errors.

- Select Tools > Programmer. Connect your DE10-Lite board to your host computer with the USB cable. The Hardware Setup should be set to USB-Blaster[USB-0], as shown in Figure 5. Make sure the majority.sof file is listed and the Program/Configure box is checked as shown in Figure 5. Then click Start.
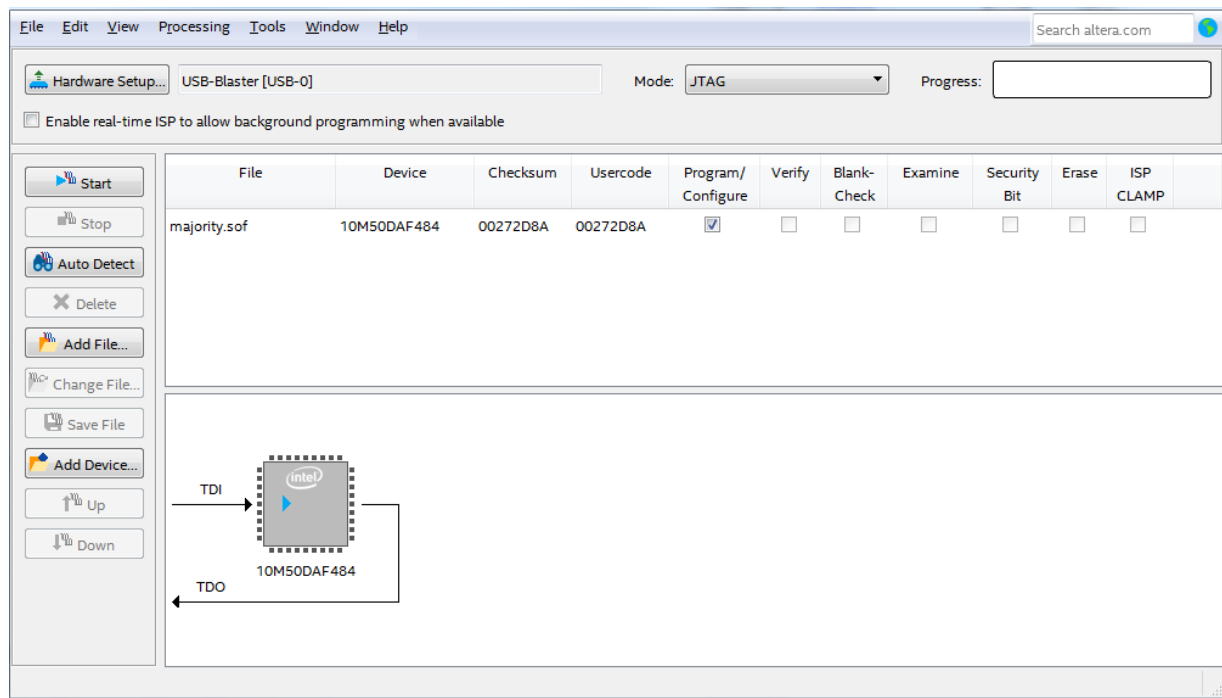
Figure 5. Programmer Tool Configuration

You can now test the majority gate design on the DE10-Lite board using SW[0], SW[1] and SW[2] as inputs. When 2 of the 3 switch inputs are high, then the output, LEDR[1] should be on; otherwise, it should be off.

Verify that the majority gate design works as expected.

C. Simulating in Questa/ModelSim-Intel Using a Testbench

A very powerful debugging technique is to simulate a design using either functional or timing simulation. This is generally a better way to debug a design than simply downloading and testing because in a functional simulation you can see internal signals as well as I/O signals. We will start by doing a functional simulation of the majority gate design using a testbench. Although it is possible to manually give stimulation commands or draw waveforms, the best approach is to write a testbench program in Verilog to generate the stimulation inputs and monitor the outputs.

An example testbench for the majority gate module is show in Figure 6. This testbench code has been adapted slightly from the code given in Figure 3.8 (p. 50) in your text. The changes were needed only because the names of the I/O signals are different for the DE10-Lite board than the signals used in the text.

```
module tb_majority;

    reg [2:0] count;

    wire  [7:0] HEX0;
    wire  [7:0] HEX1;
    wire  [7:0] HEX2;
    wire  [7:0] HEX3;
    wire  [7:0] HEX4;
    wire  [7:0] HEX5;
    wire  [1:0] KEY;
    wire  [9:0] LEDR;
    wire  [9:0] SW;

    assign SW[2:0] = count;

    majority UUT (.HEX0(HEX0), .HEX1(HEX1), .HEX2(HEX2),
              .HEX3(HEX3), .HEX4(HEX4), .HEX5(HEX5),
              .KEY(KEY), .LEDR(LEDR), .SW(SW));


    initial begin
       count = 3'b000;
       repeat (8) begin
          #100
          $display("in = %b, out = %b", count, LEDR[1]);
          count = count + 3'b001;
       end
    end
endmodule
```

Figure 6. Testbench Module – tb_majority.v

Note that the testbench Verilog code is **not** intended to be synthesized; it is for simulation only. Therefore, it can use non-synthesizable Verilog constructs such as time delay (#) and the initial and $display statements. The testbench instantiates the module to be tested, in this case the majority gate design, as a component named UUT (Unit Under Test) and provides stimuli for the inputs and monitors the outputs, In Questa/ModelSim, the Waveform viewer is a convenient tool for checking the proper operation of a circuit.

To perform a functional simulation using Questa/ModelSim Intel, do the following.

- Enter the tb_majority.v into a file in the same project directory as majority.v.

- In your Quartus Prime project directory, create a new subdirectory for your simulation project. As an example, you could call the subdirectory **simulation**.

- Run Questa/ModelSim -Intel FPGA from the start menu or a desktop icon. (Do not launch it from within Quartus Prime since the testbench will not be part of such a project.)

- Select File > New> Project to open the Create Project dialog box. Give the project a name and specify the project location as the subdirectory that you created above. Leave the other settings in the default state. (i.e. Default Library Name = work, Copy Settings From = Copy Library Mappings). Click OK.

- Next an Add Items to Project Window opens. Add both your tb_majority.v and your majority.v. Leave the open set to "Reference from current location".

- Select Compile > Compile All. Your project should compile without errors.

- Click Simulate > Start Simulation. Select Design > work > tb_majority

  - **If using Questa** you need to tell the simulator to save the internal signals of the design (so we can look at them in the Wave viewer). To do so, before clicking OK, select "Optimization Options…" then check "Apply full visibility to all modules". Then click OK.

- To view a simulation waveform, click View > Wave. Then in the Objects window, select count, right click on it and select Add Wave. This will add the count signal to your Wave window. Next expand the LEDR signal, select LEDR[1] and add it to the Wave window. This way you can see the waveform for the input and output signals of the majority gate.

- You can run the simulation by selecting Simulate > Run > Run 100 or by simply typing run in the command window. This will run your simulation for 100 time units, which is 100 ps by default. See Figure 7 for a screenshot of the Wave window after simulating for 800 ps.
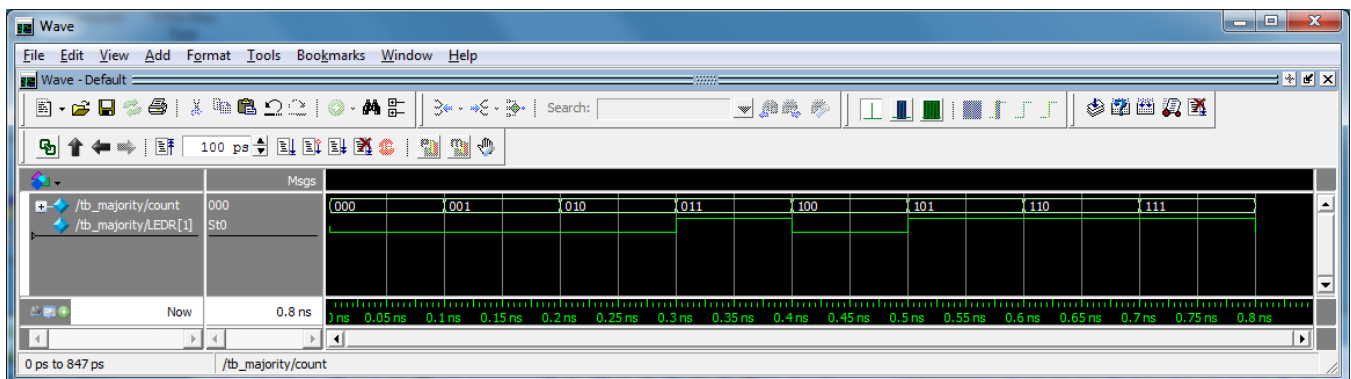

Figure 7. Functional Simulation Waveform for Majority Gate Design

You should also see output in the Questa/ModelSim console window from the $display command in the textbench. The console output should be a text version of the waveform output and should be the same as in Figure 3.9 in your text and shown in Figure 8.

in = 000, out = 0
in = 001, out = 0
in = 010, out = 0
in = 011, out = 1

Figure 8. Questa/ModelSim Console Output

Part I – Implementing Basic Combinational Logic Gates

Following the example give in Part 0, create a new project and implement some logic gates of your choice using the switches, buttons, LEDs and 7-segment displays on the DE10-Lite board. Download your design and verify proper operation. There are many possible design options. A few examples are 2-, 3- or 4-input AND, NAND, OR, NOR gates. You could also use a pushbutton to directly control an LED or segment of a 7-segment display. Note that the segments of the 7-segment display are active-low, as shown in Figure 9. Also note that bit 7 corresponds to the decimal point. For example, in order to permanently turn off all the segments of the HEX0 7-segment display, you could use the following assign statement:

```
assign HEX0 = 8'b11111111;
```

You can verify experimentally or from the DE10-Lite board schematic that the push-button switches are active-low.

Design a "testbench" verilog module which instantiates a copy of your module, exercises the circuit through all possible input combinations, and prints inputs and outputs alongside each other in a format that is easy to read.
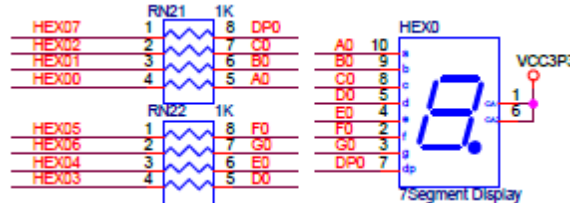

Figure 9. Excerpt from DE10-Lite Schematic Showing Active-Low 7-Segment Display. Active low operation is visible because the common nodes on pins 1 and 6 are tied to Vcc.

Once your design works on the DE10-Lite board, demonstrate it to your TA for checkoff.

Part II – Implementing a Decimal Display for 4-bit Switch Inputs

In this part, design a simple combinational circuit so that the value of the 4-bit switch input, SW[3] – SW[0], is displayed in decimal on HEX[1] and HEX[0]. Thus, your display will show the values 00, 01, 02, … 09, 10, 11, 12, 13, 14, 15 depending on the positions of SW[3]-SW[0]. The other 7-segment displays (HEX5-HEX2) should be off.

Go through the following steps:

- Write a truth table for the outputs (i.e. the segments on the HEX1 and HEX0 displays) based on the 4-bit input (SW[3]-SW[0]).

- Draw a Karnaugh map for each output function.

- Use the Karnaugh map to generate logic equations for each output.

- Implement each logic equation using a Verilog **assign** statement.

- Compile your Verilog design and test it in the DE10-Lite board.

Later, you will learn other ways to describe combinational logic in Verilog, such as using a **case** statement to directly implement a truth table without solving for the logic equations.

Demonstrate your working design on the DE10-Lite board to your TA for verification.

Lab Report (100 pts total)
Submit the following items for verification and grading:

1) Lab cover sheet with TA verifications for:
   - Part I implementation of basic logic gates using switches and LEDs
   - Part II implementation of decimal display of numbers from 00 - 15
2) Complete Verilog source code for both lab exercises (Part I and Part II).
3) Truth table, Karnaugh maps and output equations for your Part II design.
4) Printed outputs from Testbenches showing inputs alongside outputs for Parts I and II.