Digital Logic II

Design with Verilog

CST 133

Lab 4 – Design of a BCD Adder

Objective: Design structured circuit using structural modeling, datapath modeling, and behavioral modeling techniques.

# Part 1: BCD addition

Design a module that can perform a binary-coded decimal (BCD) addition. You have two 4-bit BCD (0-9) inputs “a” and “b” and an 8-bit output “x” which represents a two digit BCD number.

X[7:4] represents the upper BCD digits

X[3:0] represents the lower BCD digits

The block diagram of this module is shown below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Binary value | | | | | Digits | X[7:4] | X[3:0] |
| 0 | 0 | 0 | 0 | 0 | 00 | 0000 | 0000 |
| 0 | 0 | 0 | 0 | 1 | 01 | 0000 | 0001 |
| 0 | 0 | 0 | 1 | 0 | 02 | 0000 | 0010 |
| 0 | 0 | 0 | 1 | 1 | 03 | 0000 | 0011 |
| … | | | | | … | … | … |
| 0 | 1 | 0 | 0 | 1 | 09 | 0000 | 1001 |
| 0 | 1 | 0 | 1 | 0 | 10 | 0001 | 0000 |
| 0 | 1 | 0 | 1 | 1 | 11 | 0001 | 0001 |
| 0 | 1 | 1 | 0 | 0 | 12 | 0001 | 0010 |
| 0 | 1 | 1 | 0 | 1 | 13 | 0001 | 0011 |
| 0 | 1 | 1 | 1 | 0 | 14 | 0001 | 0100 |
| 0 | 1 | 1 | 1 | 1 | 15 | 0001 | 0101 |
| 1 | 0 | 0 | 0 | 0 | 16 | 0001 | 0110 |
| 1 | 0 | 0 | 0 | 1 | 17 | 0001 | 0111 |
| 1 | 0 | 0 | 1 | 0 | 18 | 0001 | 1000 |

BCD Adder

a

[

3

:

0

]

b

[

3

:

0

]

X

[

3

:

0

]

X

[

7

:

4

]

**2**

**–**

**BCD**

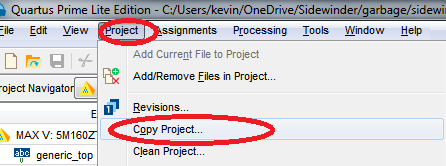
**Digits**

**BCD**

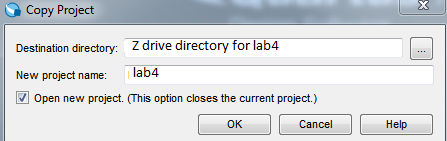
**Inputs**

# Part 1 Instructions

1. This time let’s use the default template file provided by your instructor. It is generally located on the course website.
2. Unzip the template file to your Z drive.
3. Launch the project from your Z drive by double-clicking on the de10\_generic.qpf file.
4. Go to PROJECT | COPY PROJECT



1. Enter lab4 as the New Project Name. Go to your lab4 directory and create a lab4 folder for this project. Also enter the new project name as lab4.



1. New project launches. Close the old project.
2. Create a new Verilog file called bcd\_adder.v
3. In the Verilog file, please code a BCD adder. It should follow the following format.

**module bcd\_adder(**

**a,b,x**

**);**

**input wire [3:0] a;**

**input wire [3:0] b;**

**output reg [7:0] x;**

**//Write your code here.**

**endmodule**

9) Test your BCD adder using ModelSim and provide a screenshot of the simulation in your submission. Please make sure your simulation shows all binary inputs and outputs.

# Part 2: BCD to 7-segment display decoder

1. Design a module using Verilog’s behavioral design method that can perform a BCD to seven segment display decoder. This module receives a 4-bit binary input and generates the seven output signals (a-g) plus DP (dot pixel) for the display.

**module bcd\_to\_seven( bin\_in, sseg\_out**

**);**

**input wire [3:0] bin\_in;**

**output reg [7:0] sseg\_out;**

**//write your code here**

**endmodule**

## Example of inputs and outputs:

The inputs of bin\_in should be binary 0 to 15. The outputs of sseg\_out should drive a seven segment display to indicate a number from 0-F hexadecimal.

For instance, if a binary number of 1111 is placed into the decoder, then the output sseg\_out should drive a 7-segment display to indicate F.

Input: 4’b1000 is 8 in hexadecimal. All segments should be lit.

Output: 8'b00000000

Input: 4’b1110 is E in hexadecimal. Should light segments A,D,E,F,G.

Output: 8'b00001100

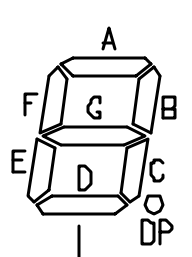
Input: 4’b0001 is 1 in hexadecimal. Should light segments B and C.

Output: 8'b11110011

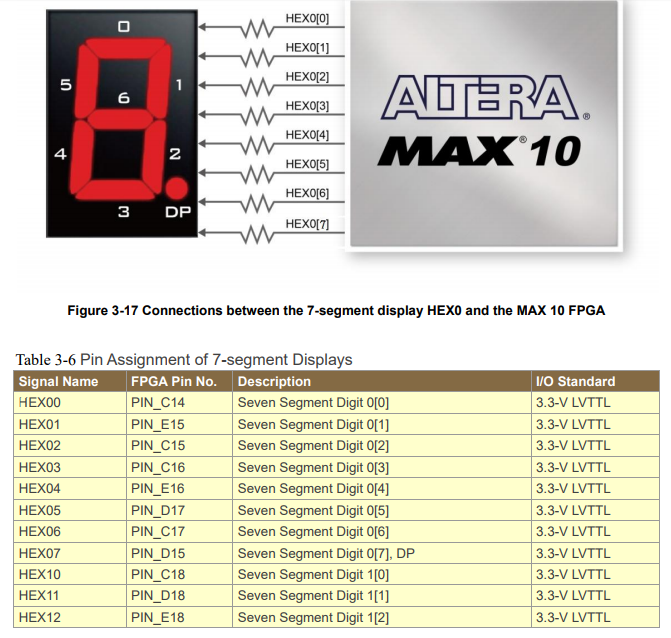
Question to think about: If we want to display the number 1, why are B and C turned off? Shouldn’t they be turned on?

|  |  |
| --- | --- |
| Segment | Bit number |
| A | 0 |
| B | 1 |
| C | 2 |
| D | 3 |
| E | 4 |
| F | 5 |
| G | 6 |
| DP | 7 |

The 7-segment signal should be setup as follows:



Refer to this mapping from the DE10-Lite datasheet regarding HEX displays 0 to 7.



(From DE10-Lite Terasic datasheet)

1. Test your BCD decoder using ModelSim or University Waveform Simulator and provide a screenshot of the simulation in your submission. Please make sure your simulation shows all binary inputs and outputs.

# Part 3

1. Use structural top level code to connect the two BCD decoder modules to two HEX displays on the DE10-Lite (Use HEX0 and HEX1) as shown in the diagram below.
2. Do not create a new Verilog file. Start instantiating a copy of your Part 1 and Part 2 modules (bcd\_adder.v and bcd\_to\_seven.v) in your generic\_top file as shown in the diagram below by using wires. Add your code before the endmodule section of the de10\_generic file (This is your top level file).

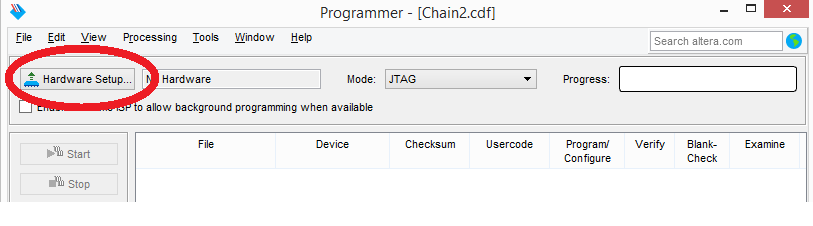
As an example, you will need a copy of the bcd\_adder.

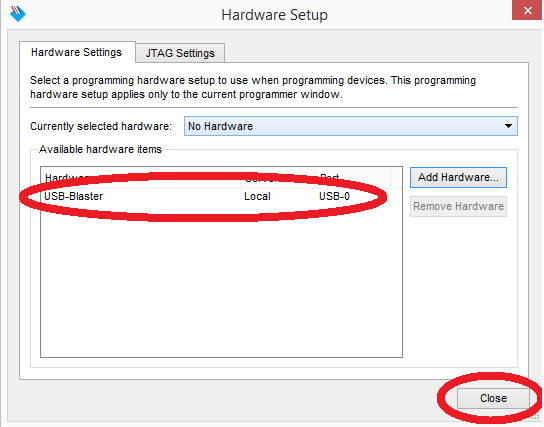
**bcd\_adder b1(.a(<wire this yourself>),.b(<wire this yourself>),.x(<wire this yourself>));**

You’ll also need to make wires to connect things together. In order to make a multi-bit wire, you can write the following to create a 4-bit wire named w1.

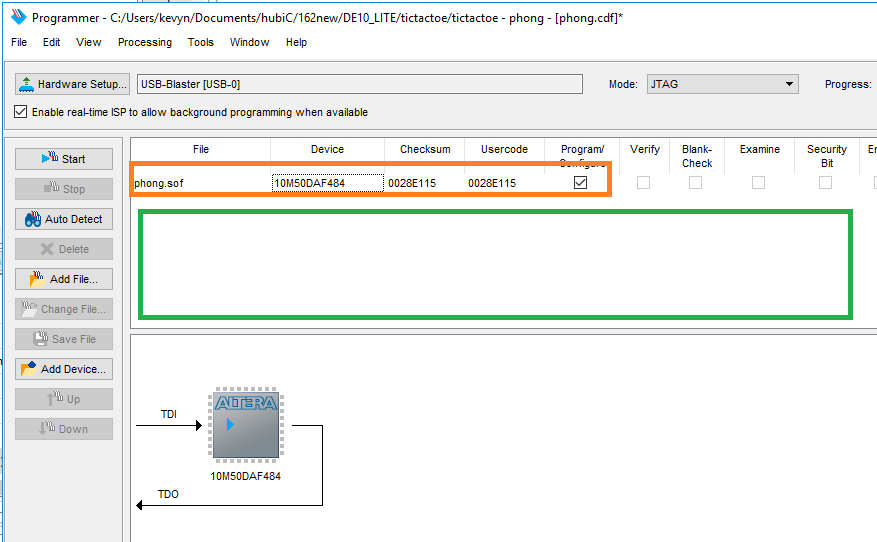
**wire [3:0] w1;**



1. Test your design by downloading it to the DE10-Lite board as shown below.
2. Make sure your design is compiled.
3. Click TOOLS|PROGRAMMER, then the Hardware Setup button. 
4. Double click on USB-Blaster until it shows up as ‘Currently selected hardware’. Then click close. If you can’t see the USB-Blaster, make sure your DE10-Lite board is plugged into the USB port. Otherwise, contact your instructor.



1. Usually, the default file that populates is correct. In this case, my project name was phong. Your .sof file will have the name of your project. Verify that the 10M50DAF484 device is selected.



1. If you find that the file is incorrect, right click in the green area and navigate to your project on the student file server (the Z drive) or wherever you decided to put the project and find the \*.sof, which is the SRAM Object File from your project. Once you have located it, click **Open**. In the programmer window, you should now see your File name, the Device (10M50DAF484). There will be several checkboxes to the right of this information. Make sure the checkbox labeled Program/Configure is checked. If there are any other devices, click on them and delete them as they will cause your programming to fail.
2. Click on STARTin the left toolbar. Wait until the configuration of the FPGA successfully completes.

**It is now configured and ready to test. Demonstrate your design to your instructor before the due date.**

# Submissions

The instructor does not guarantee last second availability. It is your responsibility to submit the lab in a timely manner. Note that the instructor’s mailbox may not be open after 5 PM.

A penalty of 20% per day up to 4 days will apply. After 4 days, no submissions will be accepted and you will receive a score of 0 for the lab.

For this lab you must submit the following to your instructor’s mailbox by 9 PM Pacific Standard Time on the day before your next scheduled lab.

1. This lab document completed in its entirety, with all supporting documents attached as well as required signatures.