



Lab #2: Combinational Circuit Design

2. Introduction

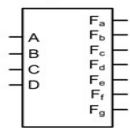
In the previous lab you built a circuit to add two 4-bit numbers, but the result was displayed in binary (easy for computers, but less convenient for humans). In this lab you will first design a '7-segment **decoder**' circuit which takes a 4-bit binary number and displays the corresponding human-readable number by lighting up the appropriate segments of a 7-segment display. For example, given the number **0010** the system would display . In the following parts, you will use your decoder to show the result of various circuits.

- In part 2.1 you will design a 7-segment decoder on paper. **Please do this before the start of class.**
- In part 2.2 you will **implement** your 7-segment decoder design as a schematic diagram
- In part 2.3 you will combine your 7-segment decoder with your work from **Lab1**, using three instances of your decoder to display the values of A, B, and Sum that your previous work computed.
- In parts 2.4, you are asked to design a logic circuit using the minimum number of **NAND** gates, and learn how to create a new Quartus project from scratch.

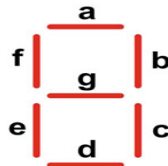
2.1 Designing a 7-Segment decoder (Pre-Lab)

1. First, complete the truth table below to define which 7-segment LED's (labeled **a..f**) to turn on in order to display the binary input number/code ABCD. Note that the LED's are '**Active Low**' which means you need to enter a **0** to turn an individual LED **On**. For example, in the row for the binary value **0010**, enter **0010010** to turn LEDs a,b,d,e,g **On**, thereby displaying the character .

7-Segment Display Decoder



7-Segment Display Layout



Note: Decimal values 0..15 are displayed as hex characters, 0..9,A..F

| A | B | C | D | F _a | F _b | F _c | F _d | F _e | F _f | F _g |
|---|---|---|---|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

2. For each of the segments in the display (i.e. for each column of the truth table, F_a , F_b , F_c , ..., F_g), complete one of the 4-input Karnaugh-maps below and use it to derive a minimal MSOP logic expression for when that particular LED (e.g. F_a etc) should be lit.

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_a =$ _____

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_b =$ _____

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_c =$ _____

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_d =$ _____

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_e =$ _____

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_f =$ _____

| CD \ AB | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

$F_g =$ _____

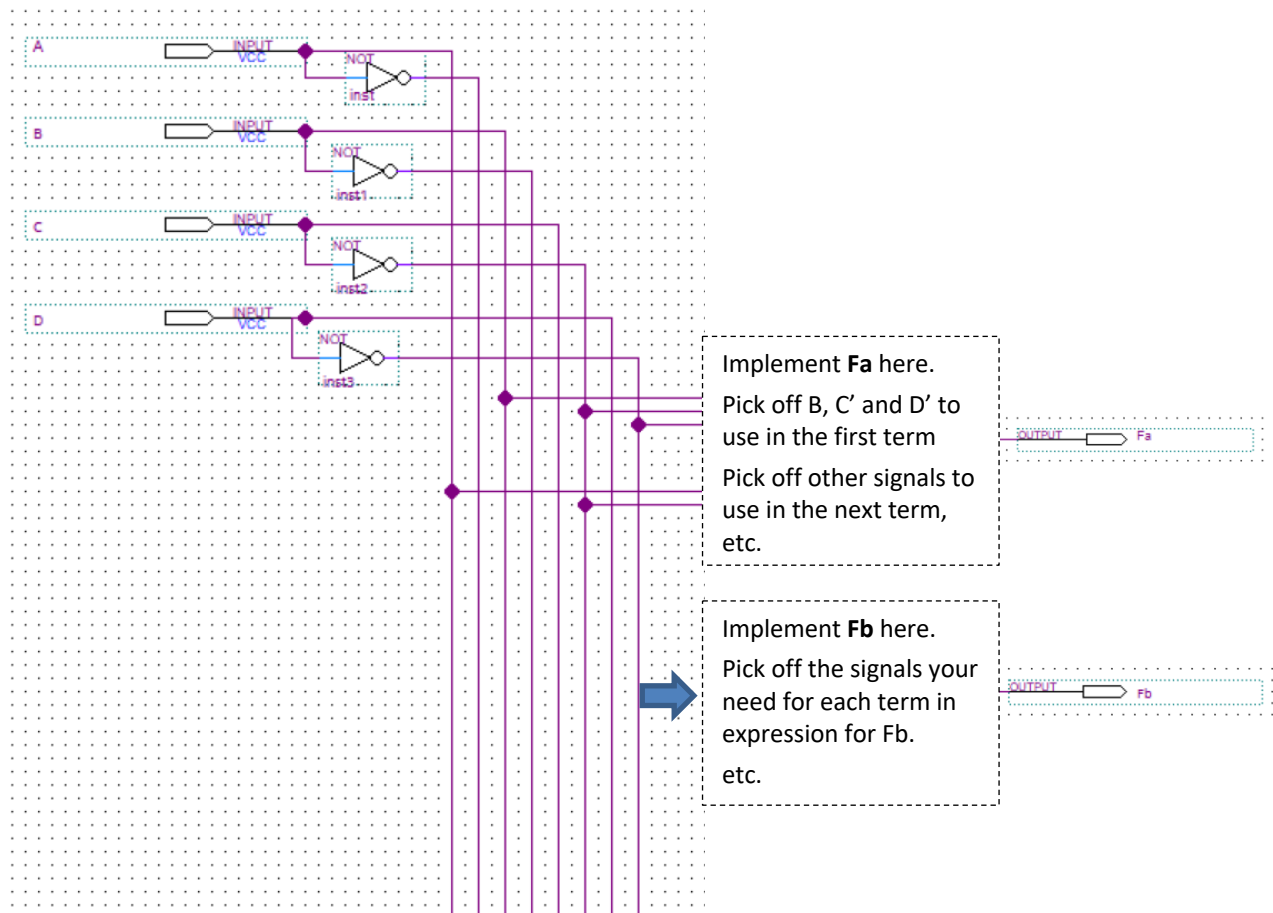
Show your work to the instructor/TA

2.2 Implement a 7-Segment decoder in Quartus

The aim of this section is to implement the 7-segment decoder you designed in section 2.1.

1. Since you will (later) be using components from your Lab1 design, start Quartus, **File > Open Project** your **Lab1.qpf** Project and then select **Project > Copy Project** to a new directory **Lab2P2** and new project Name **Lab2P2**. By default, Quartus will then open this new project.
2. Create a **File > New > Schematic** and immediately click **File > Save As** to a file called **Decode7Seg.bdf**. Double-click on the schematic to bring up the Component Library browser and add components to implement a 7-segment decoder using the logic functions you derived in section 2.1, with inputs **A,B,C,D**, and outputs **Fa...Fg**. This can rapidly get very messy, so you are strongly recommended to use a layout similar to the one below. It is then easy to pick off the specific inputs that you need to evaluate each term in the logic functions you designed.

You may also use the technique called **connecting by name**: if you give two wires on the same schematic the same name, they are connected even if they don't appear to be physically connected by a wire.



3. **File>Save** your work, then **File>Create/Update>Create...** a symbol file for your design as **Decode7Seg.bsf**, making sure it is saved in your **Lab2** working directory. We can now use this component in other designs!

Show your work to the instructor/TA

2.3 Adding 7-Segment displays to your Lab #1 TestAdder design

The aim of this section is to use the 7-segment decoder as a component to display the inputs and outputs of your Lab#1 adder circuit in more human-readable form.

1. Open your **TestAdder.bdf** design (which should already have been copied to your Lab2 directory) and double-click to add our newly created **Decode7Seg** component from the Component Library browser.
 - Connect the ABCD decoder inputs to the **SW[3..0]** input bus so we can display the first number that is being fed to the Adder. Remember to **R-click>Properties** and name each wire correctly. [Note: **A** is the most significant bit and **D** is the least significant bit].
 - Add an **Output** pin named **HEX0[0]** and hook it up to the Decoder output **Fa**. Repeat for **HEX0[1]** etc.
2. The pin assignments have not been changed, so **Compile** and **Program/Download** your design so we can check this single 7-segment decoder works before going any further. A reminder of how to program is:
 - Connect your board, Select **Tools>Programmer>HardwareSetup** and select **DE-SoC**.
 - Click **AutoDetect** and select **5CSXFC6D6**.
 - **Right-click** on the File field for the 5CSXFC6D6 and browse / select your **.sof** file.
 - Check the **Program Configure** check box in the fifth column, then click **Start**.
3. Flip the **SW[3..0]** switches starting at **0000** and then counting up to **1111**, and check the display shows the correct hex value in each case. If there are any **errors**, note which segment is incorrect and
 - Check your derivation of your Boolean expression for that particular LED (section 2.1)
 - Check your implementation of this expression in your Decode7Seg.bdf circuit diagram (section 2.2).
4. Once your single 7-segment display is working, return to your **TestAdder.bdf** design and add two more 7-segment decoders, one to display the second number being fed to the Adder on outputs **HEX1[.]**, and another to display the resulting sum on **HEX2[.]**. The **File>Save**.
5. Re-**Compile** and **Program** your design. Configure the input switches to add the two decimal numbers **11** and **6** (as you did in Lab1) and check if the HEXs show the correct input and output values.

Show your work to the instructor/TA

2.4 MSOP implementation with NAND gates only

The aim of this and next parts is to design and implement a MSOP (Minimum sum of product) expression for a function using only 2-input and 3-input NAND gates

1. **(Pre-Lab)** Obtain your personalized truth table from the instructor/TA. Please then write your name on the truth table sheet. Finish the following on paper before coming to class.
 - 1) Use a **Karnaugh** map to find an **MSOP** expression for the function.
 - 2) Then **Factor** the expression so it can be implemented using **only** 2-input and 3-input **NAND** gates. Draw your circuit on your notebook. You must do this without using dummy gates and must use the **minimum** number of NAND gates as specified on the sheet with the truth table.

Show your solution of the above step to the instructor/TA before going forward.

2. Start Quartus and follow the instructions on next page to create a new **Lab2P4** Project from scratch.
3. Now create a **File>New>Schematic File** in which to create your NAND-gate only design then **File>Save As NANDonly.bdf**. In addition to the 2-input and 3-input **NAND** gates, you may use **inverters** to complement the input variables where necessary. Name your four w,x,y,z input variables as **SW[3]..SW[0]**, and connect the output F to **LEDR[0]**.
4. This project has no Pin Assignments yet, so select **Assignments>Import Assignments** and browse to your downloaded **DE10-pins.qsf** file and hit **ok**. Check that the pin numbers appear.
5. **Compile** and **Program** your circuit on the DE10 board. Check its behavior matches your original Truth Table.

Show your work to the instructor/TA

Submission for Lab 2:

After you have been checked off all the four parts by the instructor/TA,

1. Scan your solution to the first step of Section 2.4 into a PDF file and include it in you Lab 2 folder.
2. Zip all the Lab2 project folders into one zip file, and name the zip file Lab2 <your last name>.zip, e.g. Lab1_Wang.zip, and submit it to Lab2 assignment in Blackboard.

Appendix: Creating a New Quartus Project from scratch

In some cases, you may be given a pre-configured Project to work on, or (if there is a lot of overlap) you might **Copy** an existing Project, but it is also useful to know how to create and configure a new Project from scratch:

1. Start Quartus and click on the New Project Wizard icon or select **File>New Project Wizard** to create a new project, e.g. **Lab2P4**. This brings up a succession of configuration windows
 - In the Introductory window, Click **Next**
 - Set the **Directory** to **Lab2P4** within your **ECE 2043** working directory. It will be created if necessary.
 - Also set Project **Name** and Top-Level **Entity** to **Lab2P4**, and click **Next**
 - Select an **Empty** project, and click **Next**
 - The **Add Files** window allows to add previous design files or files given by the Instructor to the list of Files within the Project. In this case, there are none so click **Next**.
 - In the Device & Board Settings window, the **Device family** pull-down menu should be **Cyclone V**.
 - In the **Available Devices** list (lower pane), scroll and select **5CSXFC6D6F31C6** (6th from the end!).
 - At this point you can click **Finish** rather than Next.
2. Go to **Assignments>Device** which brings up the Device & Board Settings window again, only this time with a button for **Device and Pin Options**. Click on it and select **Voltage** in the L-hand Category pane; then select **3.3-V LVTTTL** in the R-hand pull-down menu. Click **Ok**, then **Ok** again.
3. If you are going to do any simulations with **ModelSim** (eg. in Lab3) check this is configured correctly:
 - Go to **Tools > Options > EDA Tool Options**, and set the lowermost **ModelSim-Altera** field to:
C:\intelFPGA_lite\20.1\modelsim_ase\win32aloem