**ECE 2504 – Introduction to Computer Engineering**

**Design Project 1: Design and Implementation of Combinational Circuits**

***Read this project specification completely before beginning any design or wiring!***

1. **Preliminaries**

# *Honor Code Requirements*

You must complete this project and the associated report *individually*. Do not discuss *any aspect of your solution or approach* with *anyone* except for your instructor or an ECE 2504 GTA. Consider all information that you derive from your design process to be proprietary. This includes – *but is not limited to* – the manner in which you implement your driver circuits. Copying, using, or discussing any element of any other person’s design is a violation of the Virginia Tech Honor Code, and will be prosecuted as such.

You may discuss general features of Quartus and your Parts Kit with other students. Direct all other questions to your GTA or to your instructor.

# *Objectives*

After completing this project, you should be able to design, simulate, and implement a multiple-output combinational logic circuit from a specification. You will also gain experience in writing a project report to describe a design process and its results.

# *Project Description*

In Section 1 of the course, we observed that binary codes need not correspond to numeric values. As long as a one-to-one correspondence exists between the elements of a binary code and the elements of some other data set – such as the decimal digits – we can encode the elements of the set using the binary code.

In this project, we wish to design a digital circuit that receives four-bit values as inputs and supplies seven-bit values as outputs. The four-bit input values represent decimal digits encoded in a designed binary code. The seven-bit output values represent signals whose purpose is to drive a seven-segment LED display. The values will cause the LED display to show *decimal digits*. After designing and simulating the driver circuits for the seven-segment display, you will implement the driver circuits on your DE0 Nano Board.

# *Preparation*

You will need access to a computer that can run the Quartus Prime Lite Edition software package. You will also need the DE0 Nano Board and the ECE 2504 Parts Kit.

Completing Project 1 will require you to rely upon skills that you have developed while doing other assignments. It will also involve developing new skills:

* Consult Chapters 2 and 3 of the textbook to review Karnaugh mapping and combinational logic design.
* Review the ECE Lab Manual section “Creating and Simulating Schematic Designs in Quartus” and the Project 0 Specification for information on how to create and simulate gate-level schematic designs in Quartus.
* Review the Project 0 Specification for information on how to compile your schematic designs and program your DE0 Nano Board.
* Consult Section 3.5 of the DE0 Nano Board Manual (included with the project materials) for information on using the GPIO Expansion Headers.

If you have any questions on the use of the lab kit or your parts, see a CEL GTA or your instructor.

1. **Getting Started**

Before proceeding to the specific technical requirements of Project 1, let’s walk through the Quartus archive for Project 1.

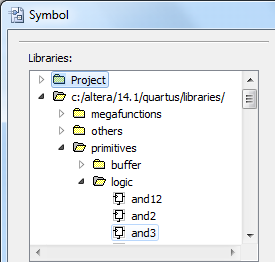
* Create a working folder for Project 1 (c:\ece2504\project1). Copy P1Implementation.qar to that folder.
* Open Quartus. Choose **File** > **Open Project**. Choose P1Implementation.qar and open the archive.
* After the archive unpacks, double-click P1Implementation in the Entity window. This will open P1Implementation.bdf.

P1Implementation.bdf shows the top-level design. It contains a component called SevenSegmentDrivers, which takes a four-bit standard binary-coded decimal number as the input and supply outputs that can drive a seven-segment LED display. The inputs and outputs of the component are mapped to pins on your DE0 Nano Board. *Note that the pin connected to A is SW3, the pin connected to D is SW0*. If you have trouble reading the pin names, you can right-click to see the pin properties.

* Right-click **SevenSegmentDrivers** and choose **Open Design File** to open the gate-level schematic. If you receive a list of file options, choose the file **SevenSegmentDrivers.bdf**. Another way to do this is to choose **Files** in the **Project Navigator**, then double-click **SevenSegmentDrivers.bdf**.

The existing gate-level schematic does *not* perform the task required by your specification, but it does provide an example of how you should lay out the circuit that you must design. To create your gate-level circuit, you will need to alter the reference circuit by deleting the existing gates and replacing them with the gates that make up the circuit you have designed. Preserve the section of the circuit that provides the input values A, B, C, D (the input pins on the left) and their complements (the inverters on the right). This will provide an easy starting point for you to lay out your own circuit diagram.

Remember that you can add logic gates to a schematic by pressing  on the toolbar. This opens the dialogue box shown in the figure below.



Primitive gates

Figure 1.1. Selecting primitive gates from the Symbol Tool

1. **Doing Your Project**

# *Specification*

Let a four-bit binary input code ABCD represent a decimal digit encoded in a (5, 3, 2, -1) format. Only ten of the sixteen four-bit codes correspond to decimal digits. The remaining six combinations represent don’t care conditions. Use the following table to represent the correspondence between decimal digits and codes in this format.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decimal Digit | 0 | - | 2 | 1 | 3 | - | - | 4 | 5 | - | - | 6 | 8 | 7 | - | 9 |
| (5, 3, 2, -1) | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| Minterm | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Figure 2.1. Correspondence between input codes, decimal digits, and minterm values

Note that the correspondence between the four-bit codes and the minterm values is the same as it would be if the four-bit values represented standard binary numbers. This is because the minterm representation for a binary combination is the same irrespective of what the bits of the binary combination represent.

Let (t, u, v, w, x, y, z) represent the set of outputs that you will use to drive the pins of a seven-segment LED display. The outputs are such that when the user applies a valid (5, 3, 2, -1) code as the input, a particular number should appear on the seven-segment display. The characters should appear as in Figure 2.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0000 → 0 | 0011 → 1 | 0010 → 2 | 0100 → 3 | 0111 → 4 |
|  |  |  |  |  |
| 1000 → 5 | 1011 → 6 | 1101 → 7 | 1100 → 8 | 1111 → 9 |

Figure 2.2. LED Display appearances for the ten valid input combinations.

The diagram at the left shows the locations of the segments on the display.

# To complete Project 1, you must:

* Model a circuit that accepts decimal inputs encoded in the (5, 3, 2, -1) format and outputs the values that will drive a seven-segment LED display to show the correct decimal digit.
* Derive minimal logic equations from your model.
* Create and simulate a *gate-level circuit* in Quartus that maps your logic equations to 2-input NAND gates and inverters.
* Implement the circuit in your schematic on your DE0 Nano Board.

# *Modeling the Circuit*

Design all seven driver circuits to correctly light the segments of the seven-segment LED display for each valid four-bit input. An input of *logic-0 lights a segment*, while an input of *logic-1 causes a segment to remain unlit*. For example, to display the number 1, segments u and v must be lit, and must therefore receive inputs of 0. Segments t, w, x, y, and z must remain unlit, and must therefore receive inputs of 1. The input that each segment receives is the same as the output of the corresponding driver circuit.

* Derive a truth table that shows all input combinations and the corresponding output for each driver circuit. *Remember to indicate don’t-care conditions when appropriate.*
* Use the information in the truth table to create your Karnaugh maps, and use your Karnaugh maps to derive logic equations for each output.

# *Creating a Gate-level Circuit*

After you derive the logic equations for your driver circuits, create your design in the **SevenSegmentDrivers** component of the P1Implementation schematic. All seven driver circuits should appear in the same schematic. **Use only 2-input NAND gates and inverters in your driver circuits.** Show a discrete inverter whenever you need to use one. If you need the complemented version of an input variable (A, B, C, D), you must use an inverter to derive it. (If you preserve the input signal wires as they exist in the original schematic, you will have the input signals and their complements already wired.)

Even though a correct model for the seven-segment display driver circuit will only have one correct representation, there are many ways that you could choose to implement your logic function using only 2-input NAND gates and inverters. You should determine the manner in which you implement each segment in a way that optimizes the overall circuit. You must explain in your report which criteria you used to make this selection.

Hint: Consider propagation delay and gate count.

The original circuit provided in the model **does not** correctly implement the requirements of your project. Do not use the circuits in this schematic as elements of your design. Use the original gate-level circuit as a model for creating your gate-level circuit. Modify the schematic to create one that implements the logic equations you derived. *Include your final gate-level schematic in your report*.

Hint: To obtain a better quality schematic image, select the circuit from your bdf file that you want in your report and copy it. Go to your Word document and use Paste > Paste Special, then select an image format such as bitmap.

When you make changes to a schematic, you must update the symbol file. On the toolbar, choose   
**File** > **Create/Update** > **Create Symbol Files for Current File**. Follow the instructions described in previous assignments for placing new or updated symbols in the schematic window. You may need to rewire the connections and revise the pins in the P1Implementation schematic.

*Simulating the Gate-level Circuit*

To simulate your gate-level circuit, follow the instructions you received in Project 0 for compiling your design and creating a waveform file. Remember that you can group input signals together and then use the Count Value function to easily generate all of the valid input combinations for your circuit.

Hint: Be sure to group your signals in the correct order; e.g. to make the group ABCD, A (SW3) should be first (on the top) and D (SW0) should be last (on the bottom).

Your circuit’s behavior should be consistent with the table in the Project Requirements section and with the model you created. For example, if you apply the simulated input ABCD = 0000 (which corresponds to the decimal digit 0 in our decimal Gray code), you should obtain the value TUVWXYZ = 0000001 as the output of SevenSegmentDrivers. Your simulation should reveal that each input case produces the correct output. Use differences between the desired behavior of the circuit and its actual behavior (as represented by the simulation) to pinpoint and fix the bugs in your circuit. *Provide a screenshot of your final simulation waveforms in your report.*

* *Each valid input and corresponding output should be labeled.*

Hint: Most students use some combination of highlighting, text boxes, and/or drawing tool shapes such as lines or boxes.

* Be sure the signal names are readable.

Hint: Make sure the column is wide enough to display the signal name and that the image is clear.

# *Building the Circuit*

After you simulate your gate-level schematic and verify that it works, implement the gate-level schematic on your DE0 Nano Board. Follow the instructions you received in Project 0 for compiling your design and programming your board.

For this implementation, you will have to connect pins from the GPIO Expansion Header to the pins of your seven-segment LED display. Place your LED display on the breadboard from your Parts Kit so that it straddles the channel, as shown in Figure 2.3.

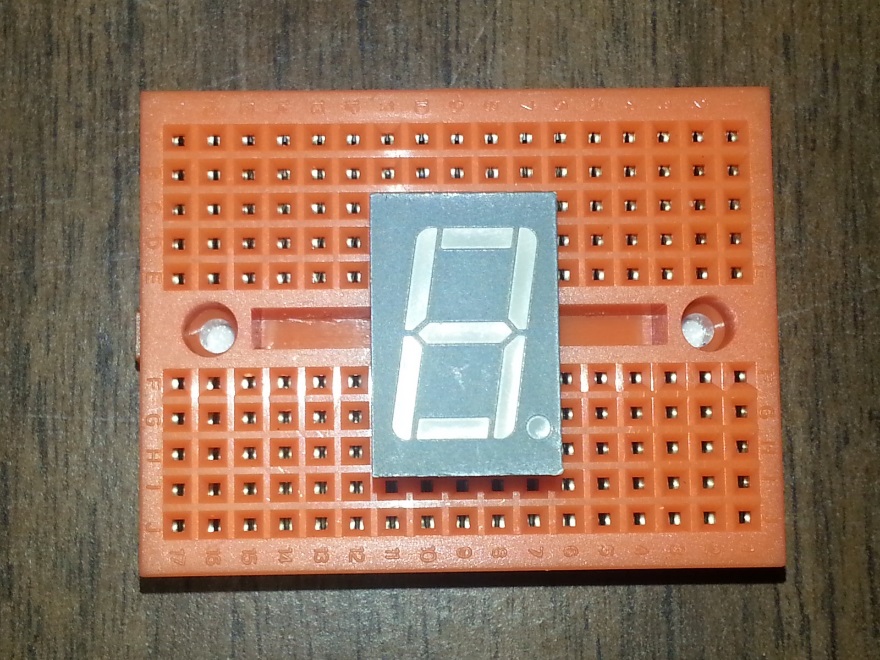


Figure 2.3. Placing your LED Display on the breadboard.

We will be using pins from GPIO Expansion Header 1 (GPIO1) to provide the outputs of your driver circuits to the pins of the display. Figure 2.4 shows a suggested way for orienting the Breadboard to make connections with the DE0 Nano Board easier to make.

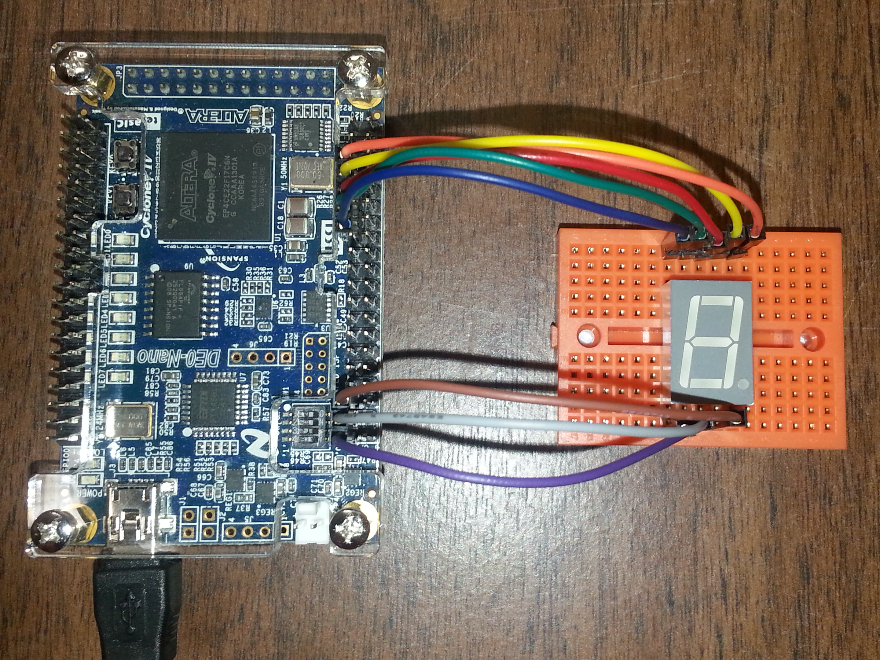


Figure 2.4. Connecting your Nano Board to your LED Display.

GPIO1 is on the right when you orient the Nano Board as shown

Use your jumper wires to connect the appropriate pins of the Nano Board to the LED Display. The pins have already been set in the top-level Quartus schematic. The correspondence between the pins of GPIO1 and the pins of the LED Display is summarized in Figure 2.6. Consult Figure 3-8 on page 18 for a diagram of the pin arrangement on GPIO1.

|  |  |  |
| --- | --- | --- |
| LED Display Pin | GPIO1 Pin Name (Pin Number) | Z Y +5 T U  X W +5 V P  Z  T  U  V  W  X  Y  P |
| Z | GPIO\_110 (15) |
| Y | GPIO\_18 (13) |
| +5 (top) | VCC\_SYS (11) |
| T | GPIO\_16 (9) |
| U | GPIO\_14 (7) |
| X | GPIO\_132 (39) |
| W | GPIO\_130 (37) |
| +5 (bottom) | Do not connect. |
| V | GPIO\_128 (35) |
| P | Do not connect. |

Figure 2.6: LED Display and Nano Board connections

The schematic shows the pins on the LED Display, as seen from the top.

Test your circuit by using the switches on the Nano Board to provide valid input combinations to the driver circuits that you implemented on the FPGA. For each valid input combination, the LED display should take on the value that corresponds to the input combination you applied to the switches. *Note that SW3 corresponds to A; SW0 corresponds to D.*

If your circuit simulated correctly in Quartus but does not work when you build it, the problem is in your wiring. Check your connectors and verify that you have connected the correct GPIO pins on the DE0 Nano Board to the correct pins of the LED display.

**Completing Your Project**

# *Circuit Validation*

This project does not require CEL validation. Instead, you will provide the source files that will allow the GTA to compile the same files that you used to implement your LED display drivers on the DE0 Nano Board. Create an archive of your work by choosing **Project > Archive Project** after you complete the implementation. When prompted for a name for your archive, the default archive name will be the same original archive. *Append your Virginia Tech PID to the end of the filename.*

When you create the archive, it should appear in the same folder that was created when you opened the original archive. Upload the archive to Canvas. *Make certain that you upload the completed archive that you created, and not the one that was provided to you.*

# *Project Report*

Prepare and submit a written report that presents a detailed discussion of the project. It should include the design approach you followed, the equations you derived, a discussion of how you mapped your equations to 2-input NAND gates and inverters, the kinds of design decisions you made and the alternatives you considered, your simulation results, your evaluation of the implementation, your observations, and your conclusions.

Prepare your report using a word processor. *Do not include handwritten items*. Proofread your report to make sure that it is free of spelling and grammar errors. Submit your report as a PDF file on Canvas.

# *Grading*

The design project will be graded on a 100 point basis, as shown on the project report cover sheet.