Chapter 3

Seven Segment Decoder

The purpose of this laboratory is to design a seven segment display decoder in VHDL, simulate the design in ISim, synthesize the design using XST, and demonstrate the seven segment display decoder operating on your FPGA prototyping board. This is the first part of a two-part seven-segment decoder laboratory.

Learning Objectives

After completing this laboratory, you should:

- Understand how the seven segment display circuit operates
- Create a circuit that properly displays hexadecimal digits

Exercises

Exercise 1 – Seven-Segment Decoder Review

Before designing a seven-segment controller circuit, it is necessary to understand how the seven-segment display operates. Figure 3.1 summarizes the twelve signals that are used to drive the seven-segment display. This display contains four anode signals (labeled AN0, AN2, AN2, and AN3), and eight cathode signals (CA, CB, CC, CD, CE, CF, CG, and DP). Each digit of the four-digit display has its own anode (left-most digit is controlled by AN3, the second to the left digit is controlled by AN2, the second to the right digit is controlled by AN1, and the right-most digit is controlled by AN0). In order to light up any of the segments of a digit, its corresponding anode signal must be asserted (i.e., provide current for individual light-emitting diode (LED) segments of the digit). For example, to enable the right two digits of the four digit display, AN1 and AN0 must be asserted.

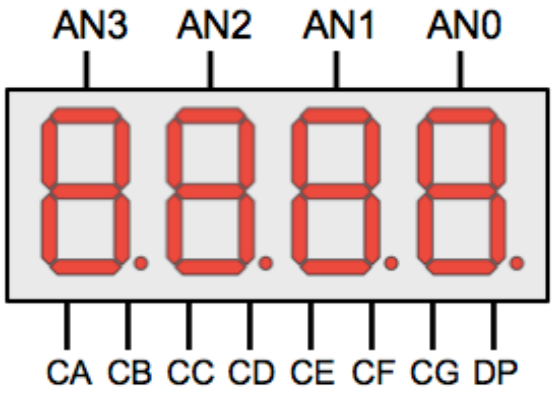

Figure 3.1: Four Digit Seven Segment Display.

The eight cathode signals correspond to individual light emitting diode (LED) segments within a digit. A schematic view of these eight LEDs is shown in Figure 3.2. To light (i.e. turn on) any segment of the display, there must be a voltage drop between the anode and the cathode. You can provide this voltage drop by asserting the anode signal and deasserting the cathode signal. The specific LED segments that are turned on can be controlled by individually specifying the value of each cathode signal.

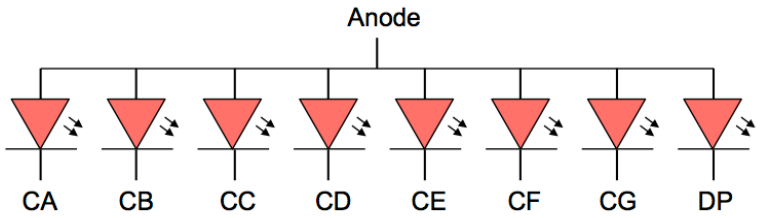

Figure 3.2: Schematic View of Seven Segment Digit.

The LED segments are placed stratigically so that specific digits and characters can be displayed (see Figure 3.3). For example the cathode signal CA corresponds to the top horizontal segment within the seven-segment display, cathode signal CC corresponds to the bottom right vertical segment, and so on. The cathode signal DP refers to the small "Digit Point" at the lower right of the digit. Custom characters can be displayed on the digit by turning on specific segments within the digit display. For example, the digit '3' can be displayed by turning ON segments A, B, G, C, and D. A font of 16 different hexadecimal characters is specified in Table 3.1. This table describes the cathode values for 16 different hex digits.

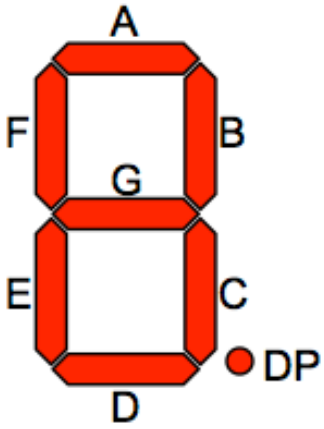

Figure 3.3: Four Digit Display.

This four-digit, seven-segment display configuration is known as a "common cathode" arrangement because the cathode signals are shared among all four digits. If more than one anode signal is asserted, each of the digits whose anode signals are asserted will display the same value (based on the value of the cathode signals). This sharing of cathode signals between all of the digits is a way to reduce the pin count of the four-digit seven-segment display. Different values can be displayed on the four digits by time-multiplexing the cathode signals. You will create a time-multiplexing seven-segment display controller during the next laboratory assignment.

The schematic of the seven-segment display for the Nexsys2 board is shown in Figure 3.4 (this figure is copied from page 9 of the Nexys2 schematic). This circuit is slightly different from the circuit used to drive the LEDs. Notice the use of a bipolar junction transistor (BJT) to drive the four anode signals of the display. A BJT is used to drive the anode because the FPGA output pins cannot provide sufficient current to drive all seven segments. While an FPGA pin can provide enough current for a single LED, it cannot provide enough current for eight LEDs (i.e., all seven segments and the digit point). The FPGA output pin for each digit anode signal (labeled AN3, AN2, AN1, and AN0) drives the base input of a PNP BJT. When the base of the transistor is 'low', the voltage between the emitter (tied to 3.3V)

Character	CG	CF	CE	CD	CC	CB	CA
0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	1
2	0	1	0	0	1	0	0
3	0	1	1	0	0	0	0
4	0	0	1	1	0	0	1
5	0	0	1	0	0	1	0
6	0	0	0	0	0	1	0
7	1	1	1	1	0	0	0
8	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0
A	0	0	0	1	0	0	0
B	0	0	0	0	0	1	1
C	1	0	0	0	1	1	0
D	0	1	0	0	0	0	1
E	0	0	0	0	1	1	0
F	0	0	0	1	1	1	0

Table 3.1: Seven Segment Display Cathode Values for 16 Characters.

and the base (driven to 0V by the FPGA) turns the transistor is 'on' and the transistor can provide sufficient current between the emitter and the collector to display all seven segments.

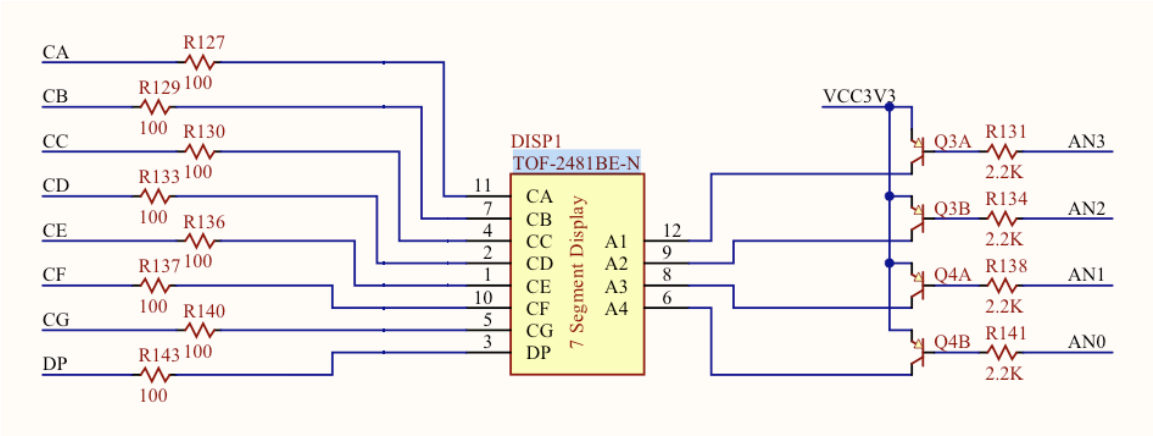

Figure 3.4: Single Digit Display.

In order to light up a specific segment within the four-digit display, the anode signal corresponding to the digit to display must be low asserted. As described above, the anode signal from the FPGA is low asserted. In other words, the transistor will provide current to the display when the corresponding FPGA ANx signal is low. In addition, the cathode signals corresponding to each segment are low-asserted. These signal must be set low so that current can flow from the anode to the cathode. If the cathode is high, there will be no voltage drop between the anode and the cathode and no current will flow to light the LED. For example, to display the number '3' on the left-most digit of the display, the AN3 signal must be asserted (set to a logic low) and the following cathode signals must be set low: CA, CB, CG, CC, and CD. The other cathode signals must be set high to prevent their corresponding segments from being displayed: CE, CF, and DP.

Note the resisters are placed between the cathode signals and the FPGA pins in Figure 3.4. Like the LEDs, these resistors are used to limit the amount of current through the segments. More information about the seven-segment display can be found by reading pages 5-6 of the Nexys2 Reference Manual.

After reading this introduction and the reference manual, answer the following questions. Do not proceed with the

next exercise until you have answered these questions.

Question: What is the purpose of the anode signals (AN3 to AN0)?

Question: What is the purpose of the segment cathode signals (CA, CB, CC, CD, CE, CF, CG, and DP)?

Question: Are the anode control signals low asserted or high asserted?

Question: Are the segment cathode signals low asserted or high asserted?

Question: The seven-segment display interface has 12 input pins: 7 segment pins, 1 DP pin, and 4 anode pins. Why does the display manufacturer time-multiplex the display, instead of providing segment and DP pins for each digit?

Question: How many pins would an 8-digit display require if you have unique segment pins and DP pins for each digit?

Question: How many pins would an 8-digit display require if you have seven segment pins and one DP pin shared among all 8-digits?

Question: What would happen if all four anode control signals were asserted simultaneously?

Question: If the voltage drop between the collector and emitter of the BJT is 0.7V and the forward bias voltage of the LED used by the segment is 1.7V, how much current will flow through the segment LED when it is turned on?

After answering these questions, you should have sufficient understanding of the seven-segment display to proceed with the design.

Exercise 2 – Seven Segment Decoder Design

In this exercise, you will create a seven-segment display decoder design in VHDL. For this laboratory, we will not display different values on all four digits at the same time (this task will be completed in next week's lab). Instead, you will create a circuit that displays one 4-bit hexadecimal value on one or more of the four digits.

Create a new VHDL file that contains an entity named "seven_seg_decode" and an architecture for your seven-segment controller. Create an entity with the following input and output ports (if you use the port names listed below, it will be easier for you to use the testbench in the next exercise):

Port Name	Direction	Width	Purpose
sw	Input	8	Switches (value to display)
btn	Input	4	Buttons (determines which segment to display)
seg	Output	7	Segment cathode signals (seg(0)=CA, seg(1) = CB, seg(2) = CC, seg(3) = CD, seg(4) = CE, seg(5) = CF, and seg(6) = C)
dp	Output	1	Decimal point cathode signal
an	Output	4	Segment anode control signals (i.e., AN0, AN1, AN2, and AN3)

To help you conceptualize the hardware needed for this lab, a block diagram of this design is shown below in Figure 3.5. In addition, a screencast is available to help you understand how to implement the logic for this lab in VHDL.

Screencast: Seven Segment Display Architecture

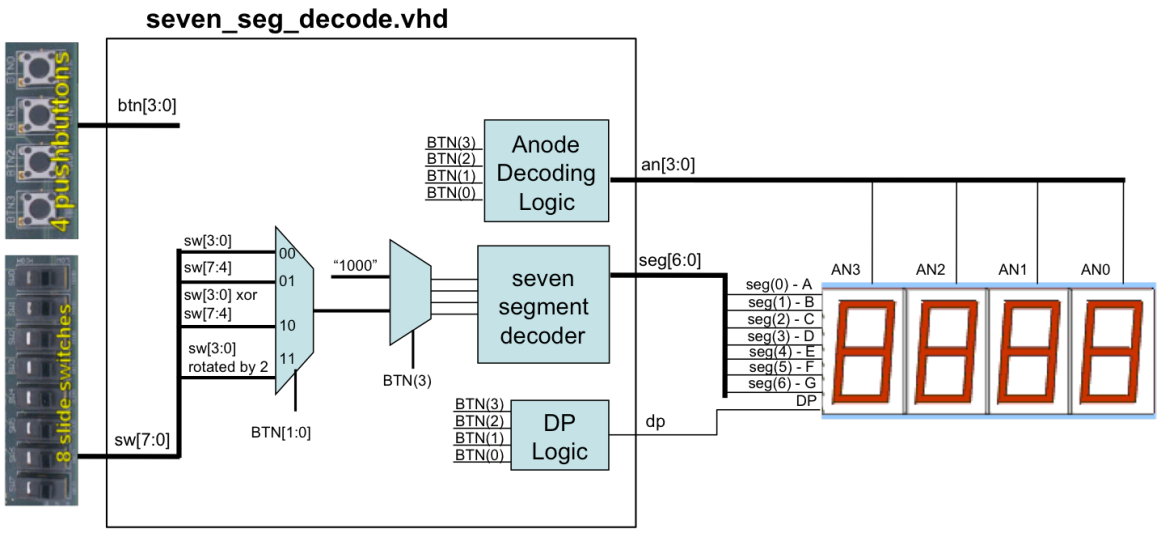

Figure 3.5: Seven Segment Decoder Architecture.

After creating the entity and empty architecture, write a concurrent statement that "decodes" a four-bit value and generates a seven-bit signal for the segments that displays the correct hexadecimal digit on the seven-segment display.

Question: Which concurrent statement is most appropriate for the decoding function that converts a 4-bit signal into the seven cathode display signals? (i.e., a simple signal assignment statement, a conditional signal assignment statement, or a selected signal assignment statement)

After creating your seven-segment decoder concurrent statement, create additional logic in your VHDL architecture that meets the following specifications:

- If BTN(3) is asserted, display the value "8" (assert all segments) on all four digits and assert the DP signal on each digit. This provides a "test" mode to see if all of the segments and decimal points are working. This has a higher priority than any other condition - all other inputs should be ignored when BTN3 is asserted.
- If BTN(2) is asserted, blank all segments and decimal points - nothing should be displayed on the seven-segment display. This has higher priority than the conditions listed below but a lower priority than the previous statement (i.e., if BTN3 is asserted and BTN2 asserted, BTN(3) has priority and the test mode should be displayed).
- If BTN(2) and BTN(3) are both de-asserted, display the following on the seven-segment display depending on the values of BTN(0) and BTN(1). Note that the DP signal should NOT be asserted in any of these cases.
  - BTN(1)='0' and BTN(0)='0': Display the hexadecimal value of the lower four switches (sw(3 downto 0)) on segment 0 (i.e., AN0)
  - BTN(1)='0' and BTN(0)='1': Display the hexadecimal value of the upper four switches (sw(7 downto 4)) on segment 1 (i.e., AN1)
  - BTN(1)='1' and BTN(0)='0': Display the hexadecimal value created by XOR'ing the lower four bits of the switches with the upper four bits of the switches on segment 2 (i.e., AN2)
  - BTN(1)='1' and BTN(0)='1': Display the hexadecimal value of the lower four switches (sw(3 downto 0)) "rotated right" by 2 on segment 3 (i.e., AN3). In other words, the four bit signal should be created by organizing the switches in the following order (MSB first): sw(1), sw(0), sw(3), sw(2).

Create a single VHDL entity and architecture that implements the seven-segment decoder described above. Continue editing your VHDL until it successfully compiles. Once your VHDL compiles, perform some initial testing of your VHDL in the ISim simulator to identify any obvious problems. Create a .tcl file to perform some basic testing.

Upload: Upload the .tcl file you used to test your seven-segment display code.

When you are confident your seven-segment decoder is working as described in this document, proceed to the next exercise.

Exercise 3 – Simulation with a Testbench

Before synthesizing a circuit from your HDL, test your circuit carefully. To aid you in your simulation, a detailed testbench has been created for you. This testbench instances your design and provides stimulus for many different test cases. These test cases have been chosen to check for common errors. Download this testbench file and add it to your project. Compile both your design and this testbench. If you designed your entity with the port names as described above, you should be able to compile the testbench without any problems. If you have different names on your top-level ports, you will need to make some small changes to the testbench. Once the testbench and your design compile together, you are ready to begin simulation.

Download: seven_seg_decode_testbench.vhd

The testbench is designed to provide you feedback on the test conditions and the presence of errors in your circuit. The testbench will print messages on the console throughout the test to describe the test condition. If there is an error with your circuit, the testbench will provide an error message indicating the expected value of the circuit and the actual value. The following example indicates an error in the tested circuit:

```
ERROR: expecting SEGMENTS=1000110 actual=1100110 at time=8000 ns
```

This message indicates that the testbench expected the value "1000110" on the segment output port but that the tested circuit produced a value of "1100110". A time stamp is provided by the message so you can go back to your simulation and zoom into the simulation to find the error condition. If you receive a message like this, make sure you spend some time looking at the situation to resolve the bug before moving forward in the lab.

Continue testing your circuit until you get the "Simulation Done" message. You may need to issue several `run xx us` commands to move the simulation forward far enough to reach the simulation end time. The following message demonstrates what this message may look like.

```
Simuation Done at time: xxxxx ns
```

Continue this process of simulation and debuggign until your architecture passes all of the tests in the testbench.

Question: What time does the simulation end (i.e., when is the Simulation Done message printed)?

Upload: Copy and paste your working seven-segment decoder design after it passes the testbench.

Exercise 4 – Circuit Synthesis and Download

At this point in the lab you should feel confident that your VHDL design is working properly. After completing your extensive testing using the testbench, most, if not all, logic errors have been fixed. Hopefully the testbench that was given to you helped you to identify any logic errors. It is much easier to identify and fix these errors during simulation than when you have downloaded the design to the board.

You are now ready to synthesize your design and download it to the board. You will need to assign pin locations for each output and input used in the design. You have already used the buttons and switches in a previous lab. For this lab, you will need to create a new .ucf file that contains the buttons, switches, segment signals, anode signals, and the DP signal. A copy of your .ucf file will need to be submitted in blackboard. Add the .ucf file to your project and generate a programming .bit file from your design using the procedure you followed during the previous labs. Download the bitfile onto the board and verify that the design works as you anticipated.

Upload: Copy and paste your working .ucf file

Question: Review the synthesis report and determine the number of "slices" of your design.

Personal Exploration

Spend some additional time in this lab by exploring FPGA editor or by creating a new design with a different behavior from your seven-segment display (make sure you save a copy of your seven segment display). Ideas for personal exploration include:

- Create a different seven-segment decoder that implements a different font (upside down, letters in the alphabet, your own unique character set, etc.)
- Create a different implementation of the lab using sequential statements (i.e., a process) instead of concurrent assignment statements
- Change the behavior of the buttons
- Use the switches to perform a simple function (for example, add the value of sw(7 downto 4) to the value of sw(3 downto 0))

Note that you do not need to submit any of the code you developed as part of your personal exploration. Provide a simple summary of the ideas you explored.

Pass Off

- Show your segment display decoder testbench simulation (it should pass all tests).
- Demonstrate to the TA a working circuit on your board.