

Lab 3 – Four-bit, Seven-Segment Decoder

Introduction

As discussed in the textbook, seven segment displays are popular display elements used in many appliance and clock designs. In this lab, you will be designing a decoder that converts a 4-bit hexadecimal code to a corresponding 7-bit code to drive such displays. In addition, the design will have a *display_on* (enable) bit, that when active high turns on the display, and when low the display is off. In this design only one of the display digits will be driven at a time. The 4-bit hexadecimal code will be selected using four of the switches (0-3); the *display_on* will be from an additional switch (6); finally, two additional switches (4-5) will be used to select which one of the four display digits is active.

For your DE10-Lite board, the connection details are found in the User Manual. [The document is available on Canvas.] The following figure is taken from this document.

3. 4 Using the 7-segment Displays

The DE10-Lite board has six 7-segment displays to display numbers. [Figure 3-17](#) shows the connection of seven segments (common anode) to pins on MAX 10 FPGA. The segment can be turned on or off by applying a low logic level or high logic level from the FPGA, respectively.

Each segment in a display is indexed from 0 to 6 and DP (decimal point), with corresponding positions given in [Figure 3-17](#). [Table 3-6](#) shows the pin assignment of FPGA to the 7-segment displays.

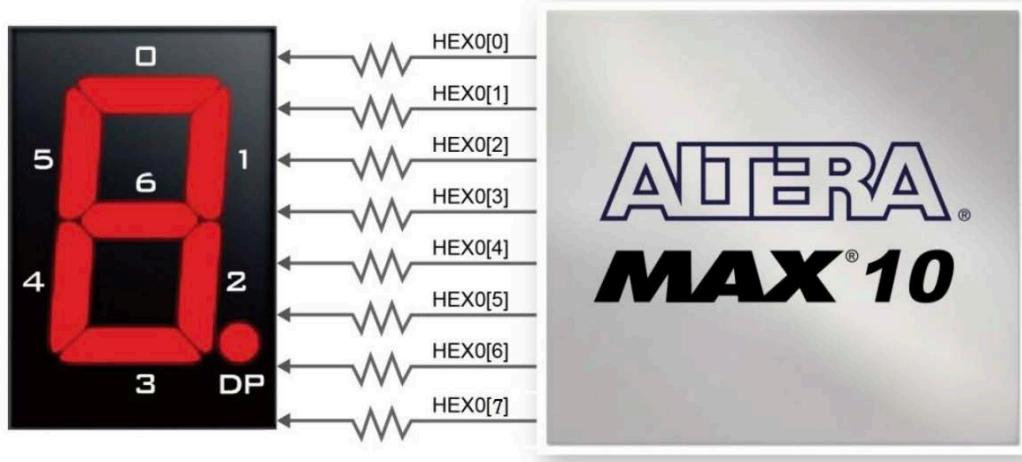


Figure 3-17 Connections between the 7-segment display HEX0 and the MAX 10 FPGA

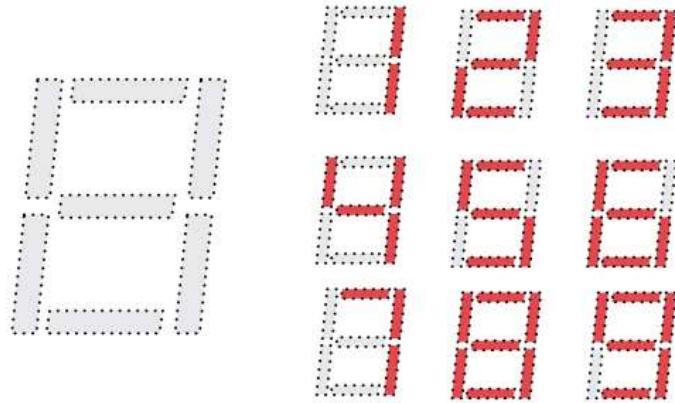


Figure 17. An un-illuminated seven-segment display and nine illumination patterns corresponding to decimal digits.

As described, each segment of each display has a dedicated signal. For instance, the topmost segment of the first display is **hex0[0]** whereas the topmost segment of the third display is **hex2[0]**. This was not the case previously with the Basys3 hardware. On that board, all four of the displays shared the same individual cathode lines. Thus, the topmost segments of all the displays were either on or off at the same time. To maintain consistency, you will be creating a decoder that would function on either board. The ***muxed_display*** module is created for you so that this design will work on the DE10-Lite board.

If you are interested in how the seven-segment displays worked on the Basys3 board, I highly encourage you to read the first few pages from the previous manual for this lab: **lab3_v14.pdf**. The Basys3 manual provides great detail on how seven-segment displays are wired and how current sinks are used in their design.

In addition to the seven segment decoder, you will be designing a second decoder that selects the display used based on two of the switches (the anode decoder).

Detailed Specification:

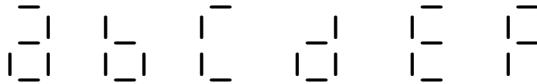
Seven input switches are to determine the output condition of the four seven-segment display digits. Switch six (***sw[6]***) is the display enable function. When off, the display is off; when on, the display is on. Switches five and four (***sw[5:4]***) control which digit is displayed as per the table below. The anode signals are active low, meaning that a signal of "0" must be supplied to turn on that specific display.

<i>sw[5]</i>	<i>sw[4]</i>	Digit Displayed	4-bit Anode code: <i>AN3-0</i> = <i>anode[3:0]</i>
0	0	0	
0	1	1	
1	0	2	
1	1	3	

Switches three through zero (***sw[3:0]***) control the digit that is to be displayed as per the table below. You will have to determine the bit pattern for each hexadecimal digit by completing the table below. Note the patterns in Figure 17 above for digits 0-9.

<i>sw[3:0]</i>	Digit Displayed	7-bit Segment code: <i>gfedcba</i> = <i>seg_out[6:0]</i>
0000	0	
0001	1	
0010	2	
0011	3	
0100	4	
0101	5	
0110	6	
0111	7	
1000	8	
1001	9	
1010	a	
1011	b	
1100	c	
1101	d	
1110	e	
1111	f	

For the six hexadecimal digits *a-f* use the segment patterns:

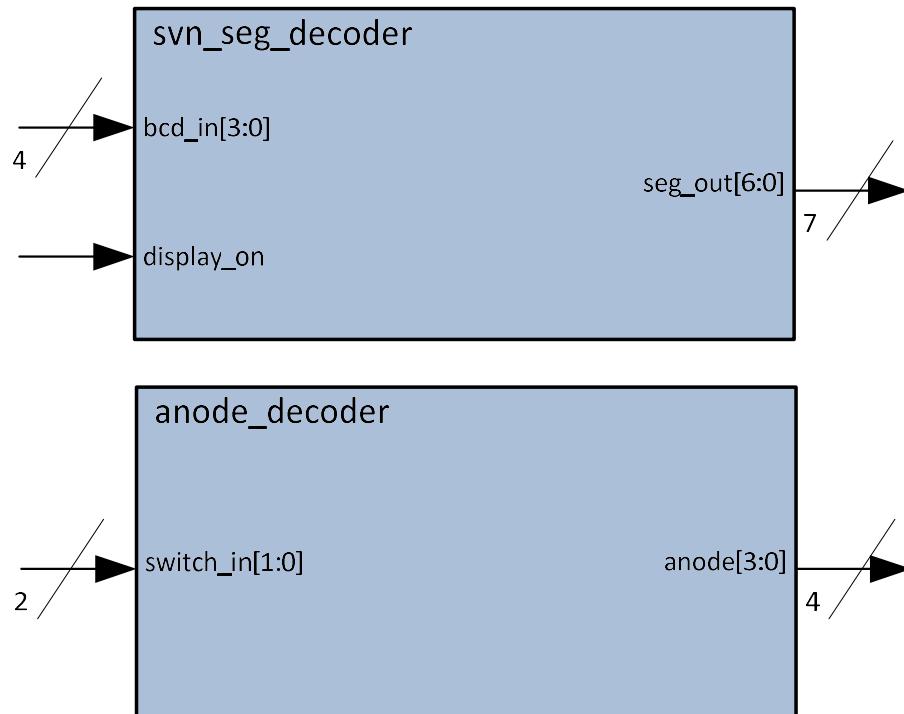


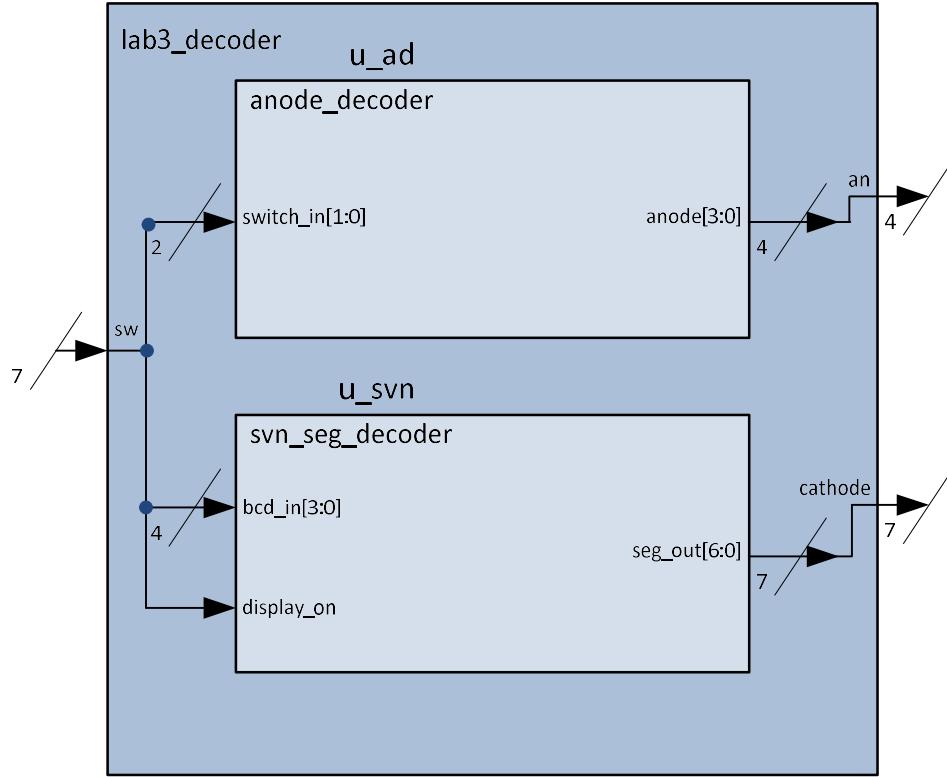
Design/Modules

In this design you will need to complete the design of three separate modules.

- The main module that contains the basic design for driving the seven cathodes is ***svn_seg_decoder***. This module decodes the 4 input bits (binary data represented by 4 switch selections) to 7 output bits (cathodes) of the seven-segment decoder.
- A second module called ***anode_decoder***, is for decoding the anode signals from the 2 input switches and selects one of the 4 anode signals of the seven-segment decoder.
- A third module, ***lab3_decoder***, instantiates these two modules into a single module that is connected to the i/o wrapper.

The block diagram for the each of the modules is shown below. Note the module names and the signal names. Skeleton files for these modules are provided in your design path.





Module **lab3_decoder** is then wrapped by **tb_lab3_decoder** for simulation, and **DE10_LITE_Temple_Top** for synthesis. No edits are needed for these modules. You can also open the Quartus settings file (**lab3_top.qsf**) and see the correspondence and signal naming between the pins on the FPGA and the signal names used at the top level i/o wrapper (**DE10_LITE_Temple_Top.sv**). You modified these files in the previous lab.

Summary of design steps:

- Complete the truth table for the *4-bit* anode signals for the 4 possible switch inputs: *00, 01, 10, 11*. Keep in mind that this 4-bit signal is *active low*. Edit the text file, **tb_anode_decoder.txt** based on your truth table.
- Complete the truth table for the *7-bit* segment (cathode) signals for the 16 possible data inputs: *0000 – 1111*. Keep in mind that this 7-bit signal is *active low*. Edit the text file, **tb_svn_seg_decoder.txt** based on this truth table. The format for the text files is provided in the comments at the head of the files. Complete these two files before doing the simulations.
- Complete the design for **svn_seg_decoder** based on this specification, including the *1-bit display_on* signal.
- Complete the design for **anode_decoder** for the anode signals for the two switch inputs.
- Complete the design for **lab3_decoder** that instantiates these two modules into a single module.

Simulation/Verification

There are *three* testbench wrappers: **one for each of the three above modules**:

tb_svn_seg_decoder.sv, **tb_anode_decoder.sv**, and **tb_lab3_decoder.sv**. The first two of these read the text files (**.txt**) containing your truth table specifications. A complete testbench is provided for the third module: **lab3_decoder**. This testbench applies 128 different test vectors to your design – exercising all combinations of the 7 switches.

There are three simulation command files (right click each of the ***.m_sim** files and Run) associated with each of the testbenches:

- **svn_seg_decoder.m_sim**, to simulate and test the seven-segment decoder module alone;
- **anode_decoder.m_sim**, to simulate and test the anode decoder alone; and
- **lab3_decoder.m_sim**, to simulate and test the integration of both modules into the lab3 decoder.

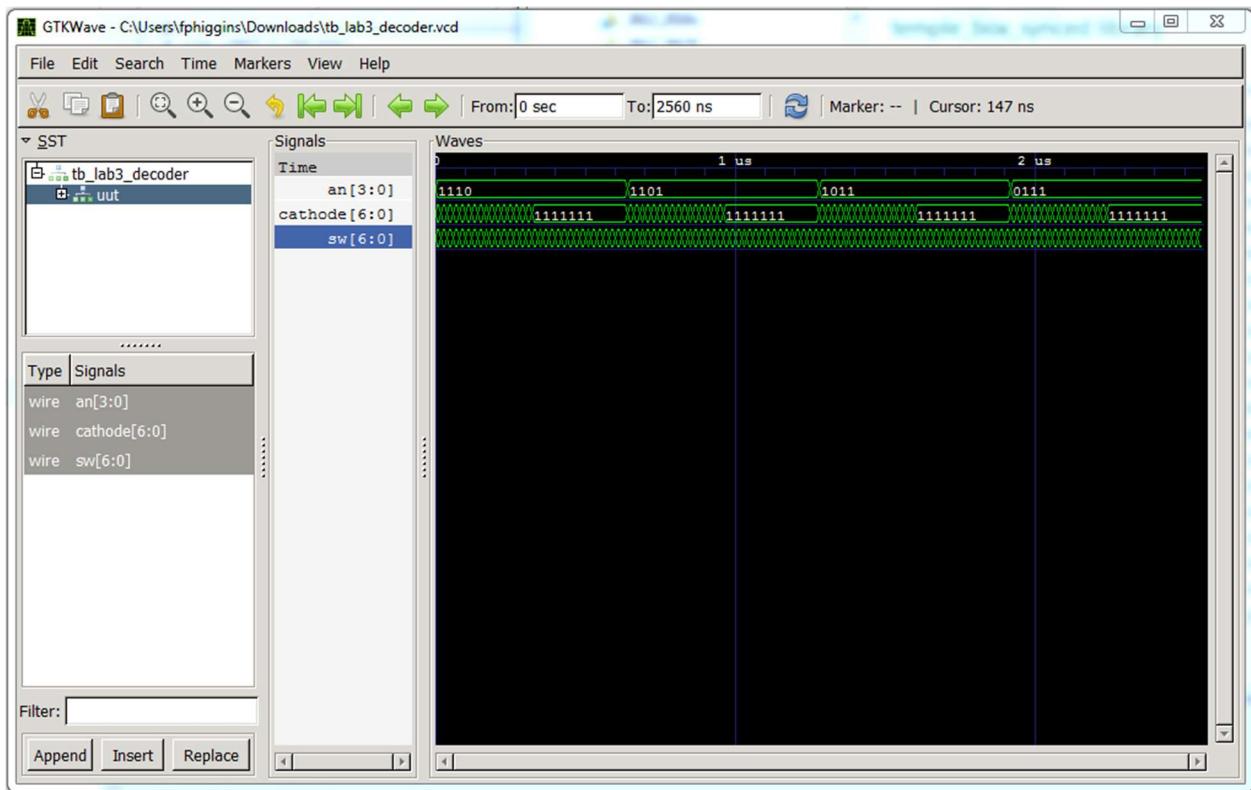
Remember to save a screen shot of each simulation log file for your reports. There will be three different simulation log files you will need.

There three different **vcf** waveform files that are generated from each simulation.

- **tb_svn_seg_decoder.vcd**
- **tb_anode_decoder.vcd**
- **tb_lab3_decoder.vcd**

Download and view these with **WaveTrace** as needed to debug your design.

As part of your lab report, show a timing diagram with all of your signals inside of the module **lab3_decoder** (**vcf** file) (instance name: **uut**) as shown below. Also demonstrate that you can change them from their default hexadecimal representation to binary as also shown below.



Be sure your simulation passes all three simulations before creating hardware.

Synthesis

Generate a binary file as you did in past labs (right-click and Run: **lab3_top.qsf**). When successful, you will generate the file: **lab3_top.sof**. Copy it to your shared folder to load and run on your DE10-Lite board. As you test your design in your hardware, remember: *Switch 7* is the *display on* switch – be sure to turn that switch on.