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 (7); finally, two additional switches (4-5) will be used to select which one of the four display digits is active.

For your Basys3 board, the connection details are found in the User Manual. [The document is available at the ***<Course web page> -> Basys3 Hardware -> Reference manual (pdf)***.] The following three figures are taken from this document.

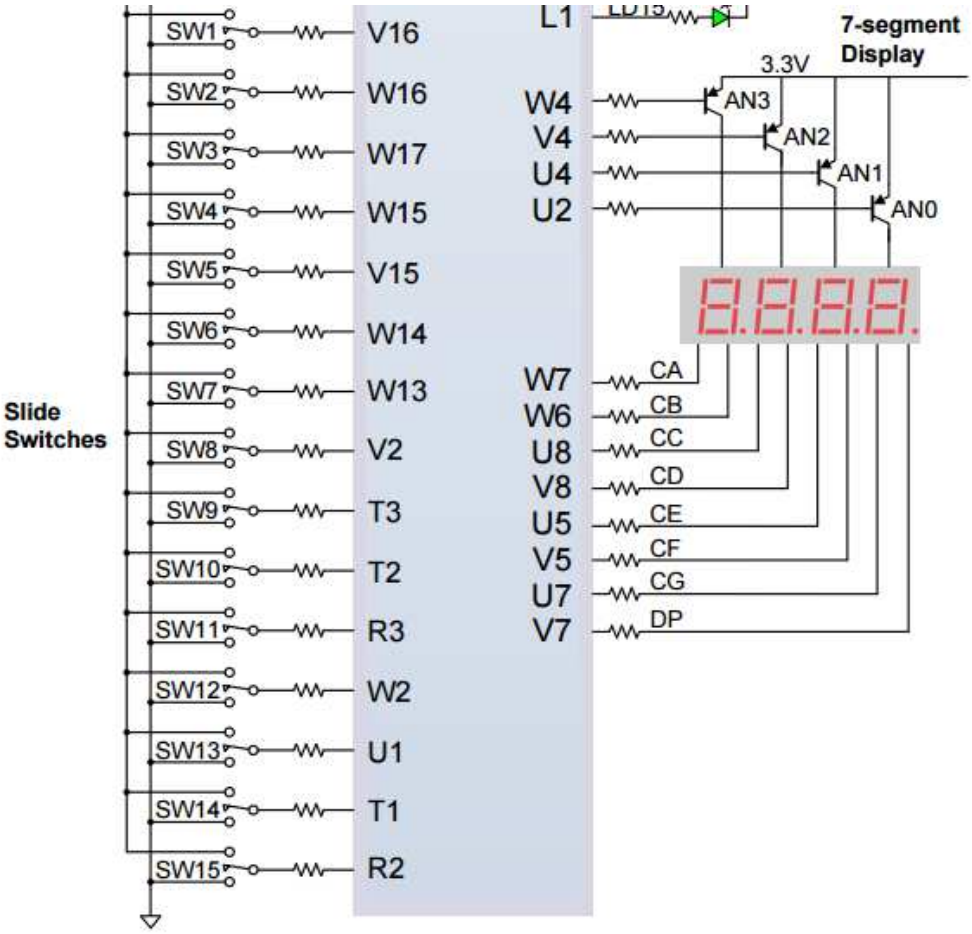


Figure 16. General purpose I/O devices on the Basys 3.

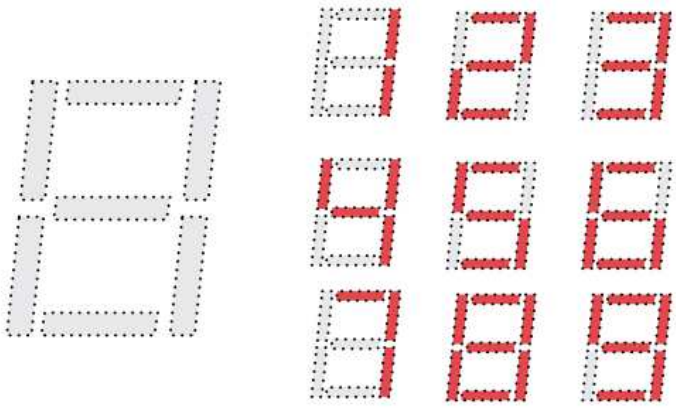


Figure 17. An un-illuminated seven-segment display and nine illumination patterns corresponding to decimal digits.

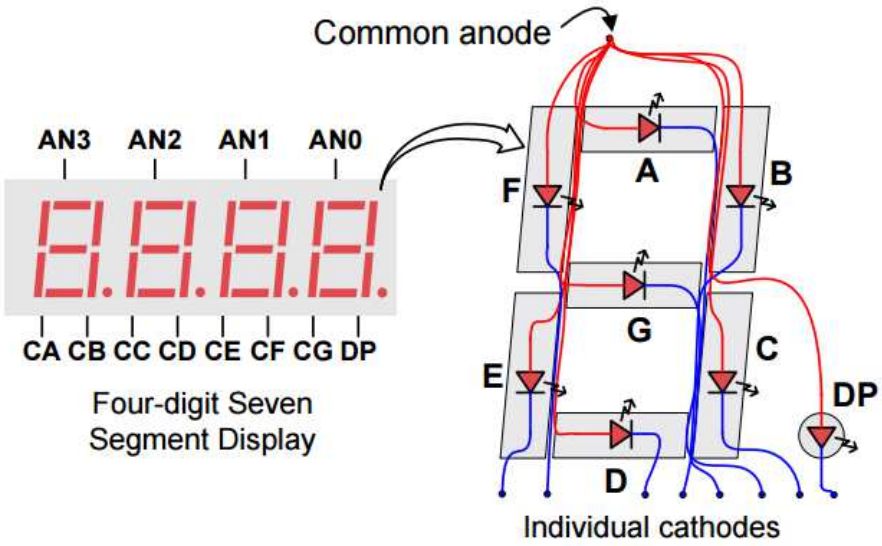


Figure 18. Common anode circuit node.

You can see from these figures that all four of the digits share the same individual cathode lines, i.e., cathode E (CD) is common to all four digits, cathode E (CE) is common to all four digits, etc. The anode is common for all of the segments *within* a digit – thus, AN0 is connected to the seven anodes of *digit 0*, AN1 is connected to the seven anodes of *digit 1*, etc.

For the next few labs you will activate these digits one at a time. Eventually, you will display all simultaneously by using time division multiplexing after you learn more about sequential design.

Note in Figure 16: the anodes are *active low* signals – for the transistor driver, its base must be pulled low to turn the transistor on; also the cathodes are *active low*. So to turn on all of the segments in *digit 0*, an[0] = 0, seg[6:0] = 7'b0000000.

This design also illustrates how to handle a *fan out* problem. Each i/o on the FPGA *is* capable of driving or sinking the current of one segment in the display. However, one i/o pin cannot drive all seven segments (the anode side). As a result, the designers added a transistor to switch and supply the larger current needed.

In addition to the seven segment decoder, you will be designing a second decoder that selects the digit displayed based on two of the switches.

Detailed Specification:

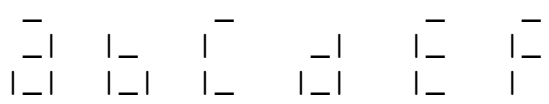
Seven input switches are to determine the output condition of the four seven segment display digits. Switch six (***sw[6]***) is the display enable function. When off, the display is off, when on the display is on. Switches five and four (***sw[5:4]***) control which digit is displayed as per the table below.

sw[5]	sw[4]	Digit Displayed	4-bit Anode code: *AN3-0 = anode[3:0]*
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*.

sw[3:0]	Digit Displayed	7-bit Segment code: *gfedcba = seg_out[6:0]*
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:

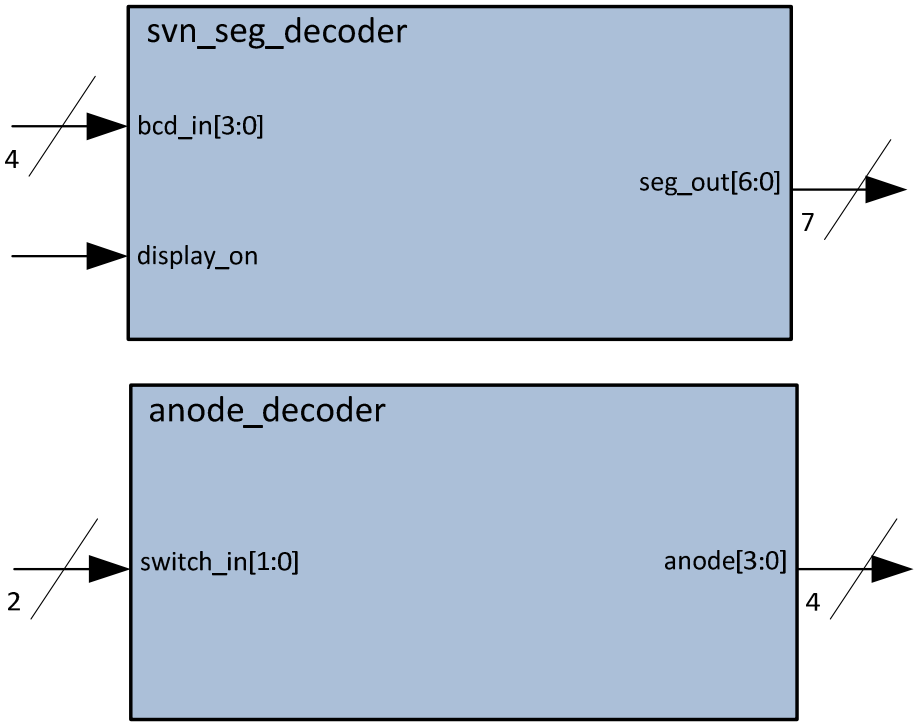


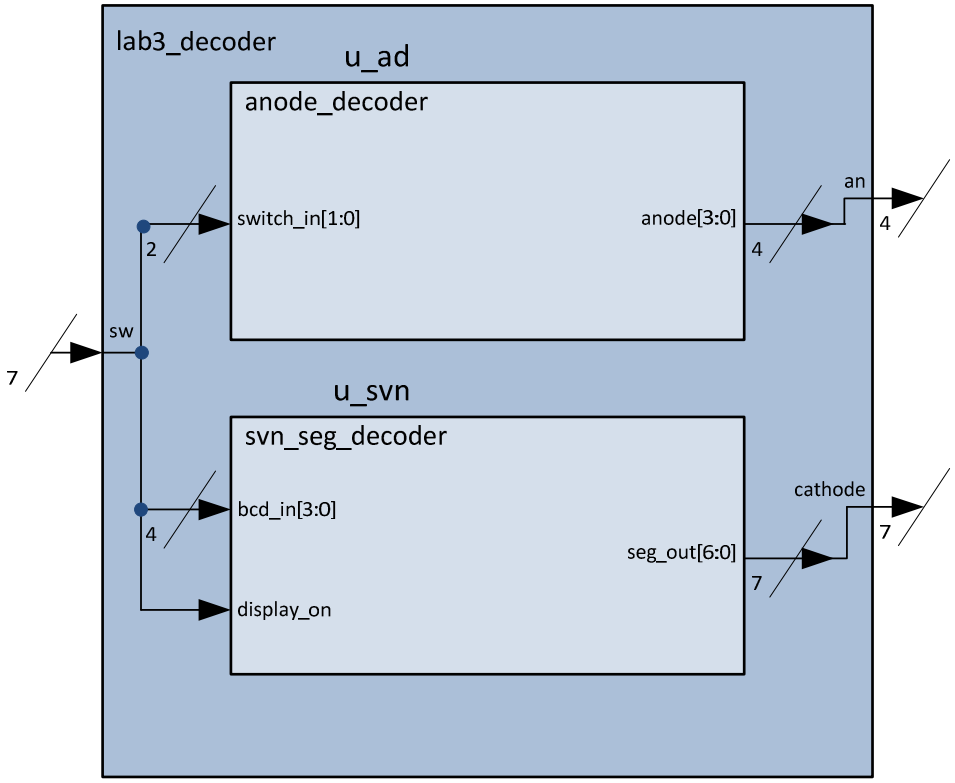
Design/Modules

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 ***lab3_top_io_wrapper*** for synthesis. No edits are needed for these modules. You can also open the user constraint file (***lab3.xdc***) 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 (***lab3_top_io_wrapper.v***). Do not modify these files for this lab.

Summary of design steps:

a) 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*.
b) 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*.
c) Complete the design for ***svn_seg_decoder*** based on this specification, including the *1-bit* ***display_on*** signal.
d) Complete the design for ***anode_decoder*** for the anode signals for the two switch inputs.
e) 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.v***, ***tb_anode_decoder.v*** and ***tb_lab3_decoder.v***. Each of these read a text file (***.txt***) to provide the test vector data.

You will need to edit two of these text files: ***tb_svn_seg_decoder.txt*** and ***tb_anode_decoder.txt*** based on your truth tables above. 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.

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 *.sim files and Run) associated with each of the testbenches:

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

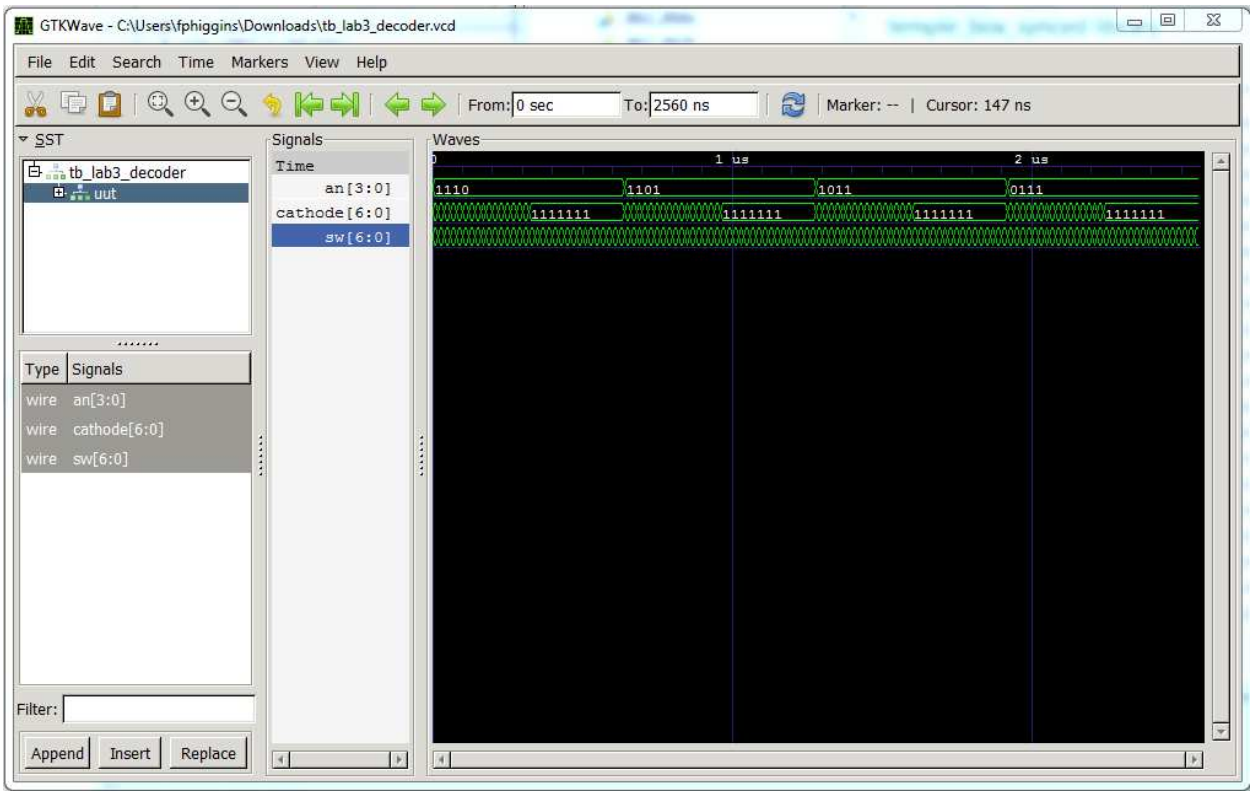
Remember to save a screen shot of each simulation log file for your reports. There will be three different simulation log files you will need.

There three different ***vcd*** 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 *gtkwave* 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*** (*vcd* 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 bit file as you did in past labs (right click and Run on: *lab3_top_io_wrapper.tcl*). When successful, you will generate the file: ***lab3_top_io_wrapper.bit***. Copy it to your memory stick to load and run on your Basys3 board. As you test your design in your hardware, remember: *Switch 7* is the *display on* switch – be sure to turn that switch on.