ECE 281 Lab 2 Report Guidance

2 March 2025| C3C Anders Enga and C3C Dexter James | USAF Academy, CO

Abstract: *This project implements VHDL code on a Basys3 board to interpret hexadecimal inputs from switches as the corresponding numeric or alphabetical character, and have one of the built in seven-segment displays show that character when a button is pushed. First, this project understood the theoretical underpinning of how seven segment displays functioned, and applied this knowledge to the hardware available with the understanding that the Basys3 boards have a slightly different functionality. Second, this project wrote the seven-segment display code to turn hexadecimal inputs into binary seven segment display inputs via a transformed truth table. Third, this project incorporated this seven-segment display entity into the larger Basys3 architecture, to transform physical switch inputs into hex, and mapping the binary outputs to the physical seven segment display. Success in this project proved an informative exercise in understanding the levels of abstraction associated with various levels of VHDL code, and understanding how to work around the constraints imposed by hardware that doesn't exactly match the theoretical model.*

Introduction

As per Digital Design and Computer Architececure RISC-V Edition, the truth table for turning hexadecimal inputs into seven segment display outputs is active high, such that a given hexadecimal input will turn on a corresponding set of seven segment display segments, thus lighting the corresponding character. For example, a hexadecimal '0' will turn on segments A-F of the display, and leave segment G off, to light up the '0' pattern in the display.

On the Basys3, there are four seven segment displays. A common anode is used for each, which should be driven high to illuminate that display. Each of the seven segments has an individual cathode, which must be driven low for the individual segment to light up. However, because the Basys3 uses transistors to drive current to the anode point, the anode enables are inverted. Therefore, both the anode and cathode must be driven low to illuminate the desired display and segment.

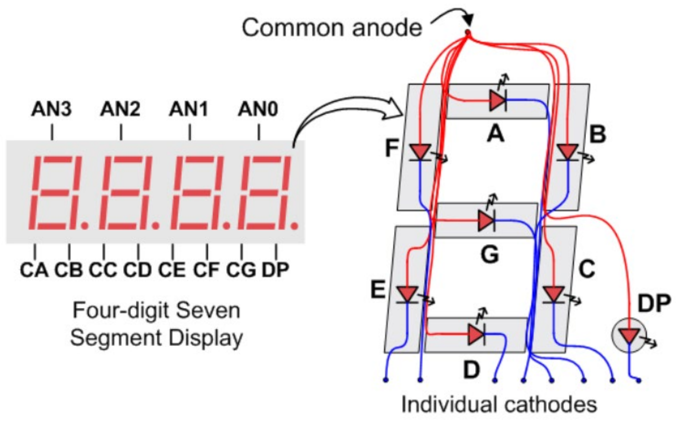

Figure 1: Basys3 Seven Segment Display Diagram

For this project to implement a design where switch inputs would be turned into hex, and a button pushed to light up the display, the truth table must be transformed. The pattern for a hexadecimal '0' input will have to turn off segments A-F of the display, and turn segment 'G' on, to light up the '0' pattern in the display. Moreover, the anode for this display must be driven low simultaneously for the '0' pattern to illuminate.

Moreover, on the Basys3, segment A is the least significant bit, compared to the *Digital Design*'s theoretical model which takes segment A as the most significant bit.

Methodology

This project took a hex value as an input in the form of 4 bits and displayed that hex value on a seven-segment display. The first step involved the creation of a truth table that showed which segments of the display should be turned on for every possible input. From there, that truth table had to be transformed by rotating and then flipping all the bits so that it would work with the display on a Basys3 board. Finally, an implementation in VHDL modeled the truth table using a lookup table on a Basys3 board. This implementation properly displayed the hex values using the seven-segment display built into the Basys3 board.

Design

Figure 2 shows the labels for each segment of the seven-segment display. In Digital Design and Computer Architecure RISC-V Edition, the segments are turned on and off using a 7-bit input with segment A corresponding to the most significant bit and segment G corresponding to the least significant bit. When a segment has an output of '1,' that means the segment is turned on for that input. An output of '0' would mean that the segment is turned off for that input. Table 1 gives examples of what the outputs should look like.

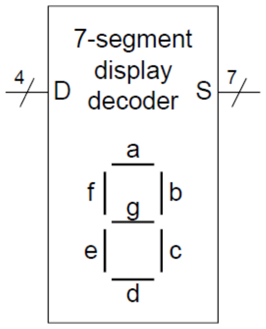

Figure 2: Shows the position of each segment on the display

Hex	Seg A	Seg B	Seg C	Seg D	Seg E	Seg F	Seg G
0	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1
F	1	0	0	0	1	1	1

Table 1: Untransformed Truth Table

The theoretical model of a seven-segment display shown by Table 1 does not consider the hardware constraints of the Basys3. Therefore, for each hex input, we manually rotated all the segment bits, A-G, around segment D, such that segment A would get segment G's spot and vice versa, segment B segment F's and vice versa, and so on. This reflects how the Basys3 board interprets the most to least significant bits opposite to the theoretical model, and the necessity to have our truth table for this device match that.

At the same time, we flipped every bit to reflect that for a segment to turn on, the cathodes need to be driven low due to the transistors within each segment. This means that any bit that was previously a '1' is replaced with a '0,' and vice versa. Table 2 shows how the truth table should look after rotating and flipping the bits.

Hex	Seg G	Seg F	Seg E	Seg D	Seg C	Seg B	Seg A
0	1	0	0	0	0	0	0
8	0	0	0	0	0	0	0
F	0	0	0	1	1	1	0

Table 2: Transformed Truth Table

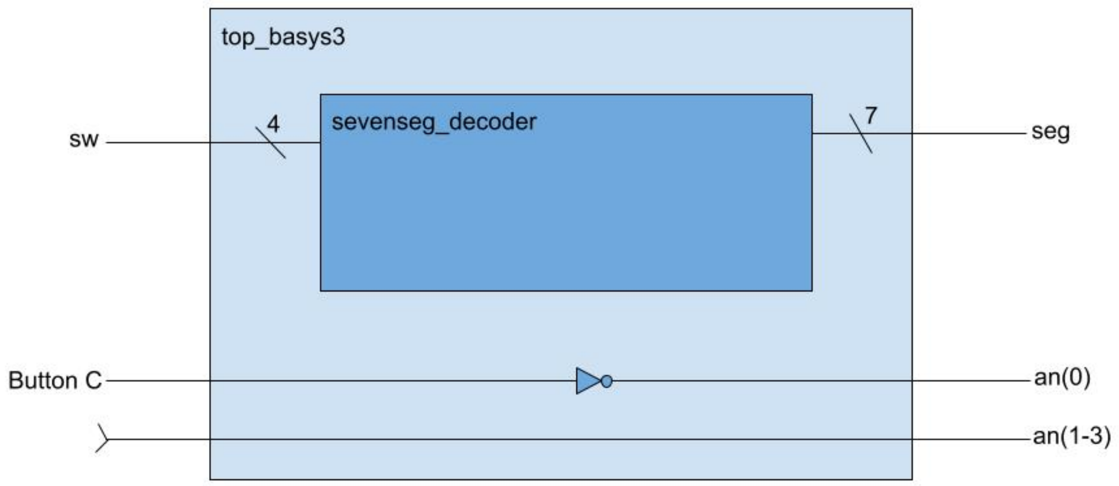

Figure 33: Top Level Design of the Basys3 Board, Incorporating the Seven Segment Decoder

Figure 3 shows the top-level plan of the VHDL code. Mapping the hex inputs into 4 binary inputs, and from that into the transformed outputs, would be the architecture of the 'sevenseg_decoder'. This would ensure that the hex inputs from the 4 switches, 'sw', would output the seven binary outputs 'seg' that match the specifications of the Basys3. These would drive the cathodes high or low, depending on whether the particular segment of G-A should be off or on.

Meanwhile, the 'Button C' input would provide the anode 'an(0)' signal to turn display 0 on or off, depending on whether it is pressed or not. To drive the anode low and thus allow the display to illuminate, the button C input would be inverted within the top_basys3 architecture. The other 3 bits of the 'an' output would be set to 0.

Therefore, a hexadecimal switch input would map to which segments to drive low to illuminate a certain segment. The button input would turn the entire display on or off, and allow those segments to be illuminated. Finally, all that was left was to implement this design in VHDL code.

Discussion

This project, in addition to the task of writing VHDL code that would work on our Basys3 board, had the additional challenge of necessitating the transformed truth table and incorporating that into VHDL. Recognizing that manually transforming 16 inputs wouldn't take too long, our VHDL implementation simply remapped each input to a new output, however, in the situation where are truth table grew to hundreds or thousands of entries, this wouldn't be feasible. Therefore, next time we would get the practice of doing a mathematical transformation of rotating and then flipping the bits in VHDL. We take away that while we may get away with manual transformations at this level, it is better to get the practice for those situations where we won't.

Moreover, we learned a lot about VHDL. The precise syntax of test benches, entities, and architecture has grown much clearer, especially the way that other entities are incorporated into the architecture of higher-level designs, will be very helpful in upcoming projects. In addition, the way the constraints file works has become more clear to us too.

Time log

All told, approximately 4 hours.

Documentation Statement

Capt. Yarbrough gave advice regarding what the test bench in VHDL should look like.