ENGG2410: Digital Design
Lab 4: Design of 7-Segment Decoder for "7-Segment Display (NEXYS-3 Board)" via *VHDL*

Shawki Areibi
School of Engineering, University of Guelph
Fall 2023

Start Date: Week #7 2023
Duration: 1 Week, DEMO: Same Day
Report Due Date: Week #8 2023, In the Drop Box

1 Objectives:

- Understand the advantages of using **Hardware Descriptive Languages** for design entry.
- Design and build Decoders including a 7-segment decoder.
- Use "When Else Statements" to design the 7-segment decoder.
- Show how to describe logic modules and interconnect them in a hierarchical fashion using VHDL.

2 Introduction

By now, you have built combinatorial logic circuits and tested them by using the simulator and by downloading them into the NEXYS 3 Board. It has all been so simple!. So simple, in fact, that you might be asking the question, "Since the computer can figure out the logic design from just the truth table, what do you need me for?" Well, maybe we don't!. On the other hand, we have calculators but we still have accountants. The important fact is that the tedious calculations have been automated, leaving us with the more challenging tasks to perform that computers do not yet do so well. For example, accountants now spend less time adding numbers and more time figuring out neat ways to reduce tax obligations. Likewise, you can spend more time thinking up neat circuits and less time mechanically grinding out the details.

CAD Limitations

However, the computer-aided design programs will always have limitations, so you cannot completely ignore the details of how the gates are connected to build a design. Having knowledge of how logic is transformed and minimized allows you to check the output from the programs and determine if some terrible mistake is being made.

More detailed specifications are also needed if you are pushing the limits of your devices. For example, you may be trying to build a circuit that performs some operation very quickly, so you need to reduce the number of gates the signals must pass through on their way from the inputs to the outputs. The CAD programs you are using may not handle this situation, so you may be required to do more work. So, for now, you have not been completely automated out of the picture.

Design Entry Using VHDL

In the 1980's rapid advances in integrated circuit technology lead to efforts to develop standard design practices for digital circuits. VHDL was developed as a part of that effort. VHDL was originally intended to serve two main purposes. First, it was used as documentation language for describing the structure of complex digital circuits. As an official IEEE standard, VHDL provided a common way of documenting circuits designed by numerous designers. Second, VHDL provided features for modeling the behavior of a digital circuit, which allowed it use as input to software programs that were then used to simulate the circuit's operation. In recent years, in addition to its use for documentation and simulation, VHDL has also become popular for use in **design entry** in CAD systems. The CAD tools are used to synthesize the VHDL code into a hardware implementation of the described circuit. In this lab our main use of VHDL will be for synthesis. The tutorials and technical report provided to you with this lab will help you synthesize the required design.

3 NEXYS-3 Board I/O Resources

The Nexys3 board includes eight slide switches, five push buttons, eight individual LEDs, and a four digit seven-segment display as shown in Figure 1.

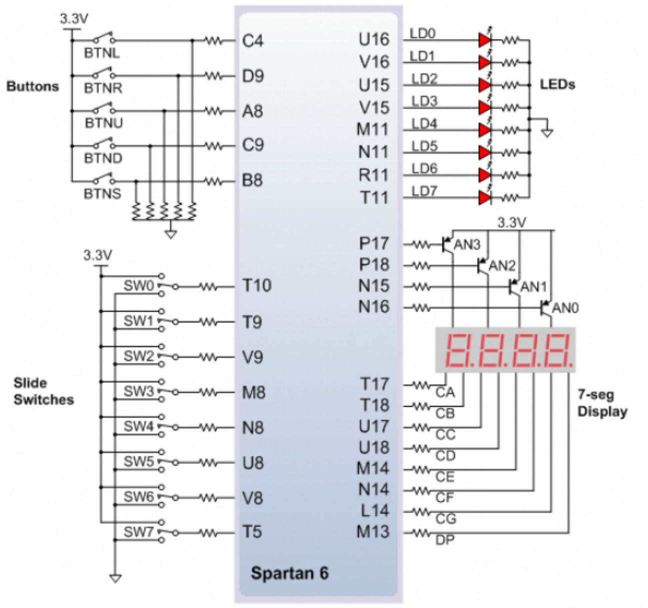

Figure 1: NEXYS-3 I/O Interface

The push-buttons and slide switches are connected to the FPGA via series resistors to prevent damage from inadvertent short circuits (a short circuit could occur if an FPGA pin assigned to a pushbutton or slide switch was inadvertently defined as an output). The push-buttons are ”momentary” switches that normally generate a low output when they are at rest, and a high output only when they are pressed. Slide switches generate constant high or low inputs depending on their position.

The eight individual high-efficiency LEDs are anode-connected to the FPGA via 390-ohm resistors, so they will turn on when a logic high voltage is applied to their respective I/O pin. Additional LEDs that are not user-accessible indicate power-on, FPGA programming status, and USB and Ethernet port status.

3.1 7-Segment Display

The Nexys3 board contains a four-digit common anode seven-segment LED display as shown in Figure 2. Each of the four digits is composed of seven segments arranged in a figure 8 pattern, with an LED embedded in each segment. Segment LEDs can be individually illuminated, so any one of 128 patterns can be displayed on a digit by illuminating certain LED segments and leaving the others dark.

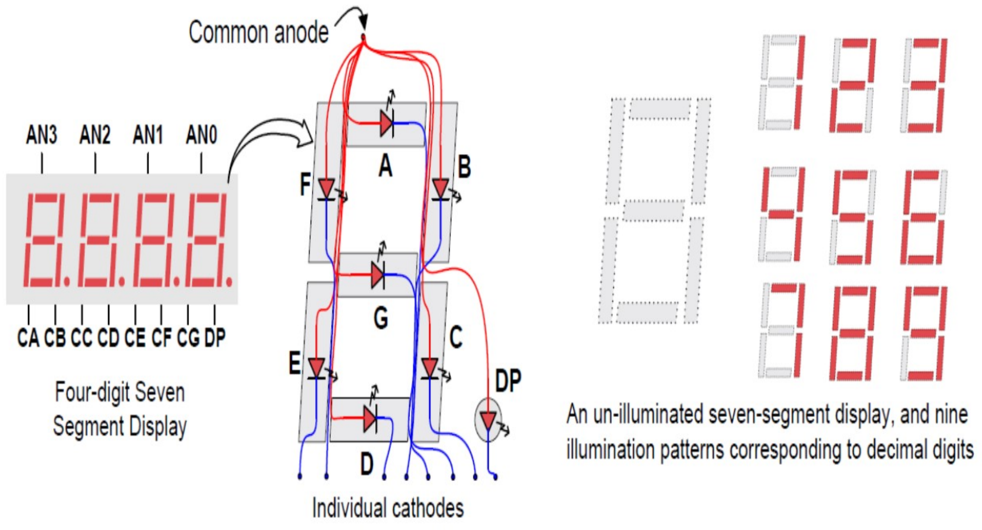
Common anode
AN3
AN2
AN1
AN0
CA CB CC CD CE CF CG DP
Four-digit Seven Segment Display
A
B
F
G
C
E
D
DP
Individual cathodes
An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

Figure 2: NEXYS-3 7-Segment Display

Of these 128 possible patterns, the ten corresponding to the decimal digits are the most useful. The anodes of the seven LEDs forming each digit are tied together into one common anode circuit node, but the LED cathodes remain separate. The common anode signals are available as four digit enable input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four D cathodes from the four digits are grouped together into a single circuit node called CD). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted.

3.2 More Resources on 7-Segment Displays

Here are some useful resources on 7-Segment Displays:

- 7-Segment Displays and decoders: What are they? (YouTube)
  Introduction to 7-Segment Displays and Decoders
- VHDL Seven Segment Display Interfacing (Nexys-3 FPGA Board) (YouTube)
  NEXYS-3 FPGA Board 7-Segment Display
- NEXYS3 -3 Reference Manual (Document)
  NEXYS-3 FPGA Board Reference Manual

4 Preparation

In the section of lab #4 on the web page you will find the following links:

1. VHDL Tutorial Tech Report "VHDL For Digital Design".
2. Design Entry Using VHDL (SOE).
3. ISE 13.3 VHDL Tutorial (By Digilent).
4. ISE 13.3 In Depth Tutorial (By Xilinx).
5. Controlling NEXYS 3 Display (VHDL CODE).
6. Controlling NEXYS 3 Display (UCF File).
7. VHDL OnLine (Tutorials, Courses, News)

You will use the **VHDL Tutorial "Tech Report"** as a reference to VHDL programming. I assume that you have already read the document **"Design Entry Using VHDL"** in the previous lab.

You may also want to use items (5) and (6) above to explore how the LEDs and 7-segment displays are controlled using a VHDL code (i.e complete VHDL code to test the switches, LEDs and Seven Segment Displays).

4.1 Displaying Switch Settings on NEXYS 3 Board LEDs/7-Seg Display

This example creates a circuit that displays the settings of the DIP switches on the LEDs and LED digits of the Digilent NEXYS 3 board. The particular set of LEDs, which is activated, is selected by the Push Buttons. The VHDL code for this example is given in the Student Resources of this lab. The steps for compiling and testing the design using the Digilent NEXYS 3 board are as follows:

- Open a new project with Foundation tools and choose VHDL entry.
- Synthesize the VHDL code (item 5 above) in the nexys3_slideswitch_vhdl.txt file for a NEXYS 3 (Spartan 6 FPGA) board. (Rename to nexys3_slideswitch.vhd)
- Compile the synthesized netlist using the nexys3_slideswitch_ucf.txt constraint file (item 6 above). (Rename it to nexys3_slideswitch.ucf)
- Down-load the nexys3_slideswitch.bit file into the NEXYS 3 board either with the iMPACT utility or Digilent Adept tool.
- Set the DIP switches and press the push-buttons. Depending on the push buttons pressed observe the results on the LEDs or 7-Segment Display (Please read the VHDL Code carefully to understand the functionality!).

$X_3X_2X_1X_0$	**Display (note Capitalization)**
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

Table 1: Truth Table for 7-Segment Display

4.2 Binary to 7-Segment Decoder

A seven-segment display is often used on computers, watches, VCRs and many electronic devices to display numbers and some characters. It consists of seven independent lights (light emitting diodes (LEDs)) in an "8" configuration as shown earlier in Figure 2. By turning on different segments, you can display different numbers and some letters.

You are to create a logic circuit to drive one of the seven-segment displays on the NEXYS 3 board. Design a circuit that takes a four bit (X_3, X_2, X_1, X_0) Binary input from the DIP switches and drives any 7-Segment Digit Display on the NEXYS 3 board as described in Table 1. Note that for the letters, some are capitalized and some are not. (The reason is that a capital **B**, for example, would come out the same as an 8 on a 7-segment display, so we will display a lower-case b instead). Determine the equations for the 7-segment display segments, with no minimization. Write VHDL code to represent the logic function for each segment as a Boolean equation (with AND, OR, NOT, etc.) or using when-else statements. Simulate and test your equation[1] using the Xilinx Foundation functional simulator.

5 Requirements

Before you start the lab you need to be well prepared since this lab is quite lengthy and requires lots of time.

1. First, check (make sure) that the 7-segment display on the NEXYS-3 board is working, by copying the VHDL file mentioned in Section 4.1.
2. You can then go through the **VHDL Tutorial using Xilinx tools** to get a better understanding of using the VHDL editor and wizard for entering your designs, synthesizing and compiling them.

[1]See section 3-1 in the text book and in particular example 3-2 for designing a BCD-to-Seven-Segment decoder.

3. **Design** the 7-segment decoder as explained in Section 4.2.
    - Use "When else" statements to implement the 7-segment decoder.
    - Write the VHDL code for the 7-segment decoder, build it and test it.
    - This VHDL code will be used as a component and integrated with the adder/subtractor circuit in your final design.
4. Demonstrate your lab to the TA.

6 Deliverables

1. **Lab Preparation File**:
   - Name your file as follows: ENG2410_F23_LAB4_Section(Mon,Tue,Thu,Fri)_Group#_Preparation.pdf
   - The preparation file can be hand written or typed. But has to be neat.
   - The preparation file should consist of the Truth Table, Boolean Equations derived, Circuit Diagram and partial VHDL Code.
   - Only one submission is allowed. Multiple submissions will not be marked.
2. **Lab Final Report File**:
   - Hand in a Hard Copy of your report to the Teaching Assistant.
   - Name your file as follows: ENG2410_F23_LAB4_Section(Mon,Tue,Thu,Fri)_Group#_FinalReport.pdf
   - Check the format of the final report below in the next section.
   - Keep the first page only for the title page.
   - Only one submission is allowed. Multiple submissions will not be marked.
3. **Zipped Project File**:
   - Name your file as follows: ENG2410_F23_LAB4_Section(Mon,Tue,Thu,Fri)_Group#_Project.zip
   - The project file should have a README file that explains the switches and LEDs you used in your design.
4. **DEMO**:
   - You will need to demonstrate to the Teaching Assistant that your design fulfills the requirements.
   - The DEMO will take place during the LAB hours and not outside these hours.

7 Report

Below is the general format of the report required:

1. Title Page:
   - Course #, and Date
   - Lab # and name of experiment
   - Your Group #, and Names
2. Start a new page and explain how you implemented your design by providing the following:
   (a) **Problem Statement**,
      i. Briefly describe the problem solved in the lab.

   (b) **Assumptions and Constraints**.
      i. Constraints could be for example using only NAND gates or NOR gates in your design.

   (c) **How you solved the Problem**,
      i. Truth Table.
      ii. Boolean Equations.
      iii. Circuit Diagram.

   (d) **VHDL Code**,
      i. List the VHDL code that you have written.
      ii. Make sure that the code has comments and documented.

   (e) **System Overview & Justification of Design**
      i. Give an overview of the system to be designed.
      ii. Briefly explain how the system works and reasons behind the design.
3. **Circuit Diagram**
   (a) Brief explanation of the hardware.
   (b) The schematic you produced using Xilinx ISE tools.
4. **Error Analysis**
   (a) Printed **Simulation** waveform data, showing that the simulation coincides with the expected truth table of your function.
   (b) Include the VHDL **Test Bench** in an Appendix.
   (c) Describe any problems with the system.
   (d) If no problems in the final system, describe problems/errors encountered during the development and how they were resolved.