**Lab 2: Vivado Intro / Structural Decoder**

Author: Seth Ricks

Date: 9/22/2025

Class: ECEN 340

**Purpose:**

1. To review the Basys 3 board
2. To review Vivado usage
3. To implement a simple structural Verilog design

**Procedure:**

Part 1 – (First Lab Day) Download the Basys3 Vivado Decoder Tutorial. Follow the instructions in the tutorial until you have a functioning decoder.

Part 2 – Create a new decoder project with a different name and implement the same design using structural Verilog (gate-level) design techniques.

1. Open the schematic view in Vivado, and study/understand the implementation
2. Use “or, and” and other structural Verilog statements to implement the Decoder

**Results/Report:**

Part 1:

In this section of the lab, we were given some code to write on the Basys3 board. This code can be seen below in **Figure 1**.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Figure 1

When we synthesized this code, the Basys3 board created the schematic below in **Figure 2**.

**A diagram of a computer circuit

AI-generated content may be incorrect.**

Figure 2

This part of the lab allowed us to re-familiarize ourselves with how to set up Vivado, and how to work with Basys3 board. With this code we wrote to the board, we were able to control 8 on-board LEDs with 3 switches. This corresponded to a binary to decimal converter. For example, if the switch code was ‘0,1,1’, LED #3 turned on.

Part 2:

This part of the lab tasked us to re-write the aforementioned code in a more primitive manner. This meant that we needed to write the actual gates for each output logic. To do this, I created a logic table for the output we want. This can be seen in **Table 1** below.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **X1** | **X2** | **X3** |  | **L7** | **L6** | **L5** | **L4** | **L3** | **L2** | **L1** | **L0** |
| 0 | 0 | 0 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 |  | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 |  | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 |  | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 |  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 |  | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 |  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Table 1*

From this table, I was able to derive 8 logic statements:

I then implemented these logic statements into Verilog code that I uploaded into our development board. The code required 8 AND gate statements, which can be seen in our code in **Figure 3** below.

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 3

This code worked as intended. The three switches corresponded to 8 LEDs on the board, essentially creating the binary-to-decimal converter for 3 bits. Vivado also created another schematic with this code, which can be seen in **Figure 4** below.

**A diagram of a computer circuit

AI-generated content may be incorrect.**

Figure

**Conclusion:**

This lab was a great introduction/review for Vivado. I vaguely remembered how to set up a project, but this solidified it for me. It was also helpful to remember how to use the Basys3 board. Despite this, I am still struggling with some aspects of the Verilog language. For example, to implement my version of this logic I had to remove the ‘reg’ statement for the LED output. As far as I understand, ‘reg’ is for blocks like *always*, and ‘wire’ is for things that need to be continuously driven. I hope to understand this more in future lab and class experiences. I also hope to understand Verilog syntax better in general over the course of this class.