

**Department of Electrical Engineering**

**Lab Report 1: Xilinx ISE Tutorial**

Name: Socheath Sok, ID: 014470701

Partner: Ashley Tran

Group: 11

Professor: Dr. Ray Wang

Class: EE 301

Date Performed: September 13, 2017

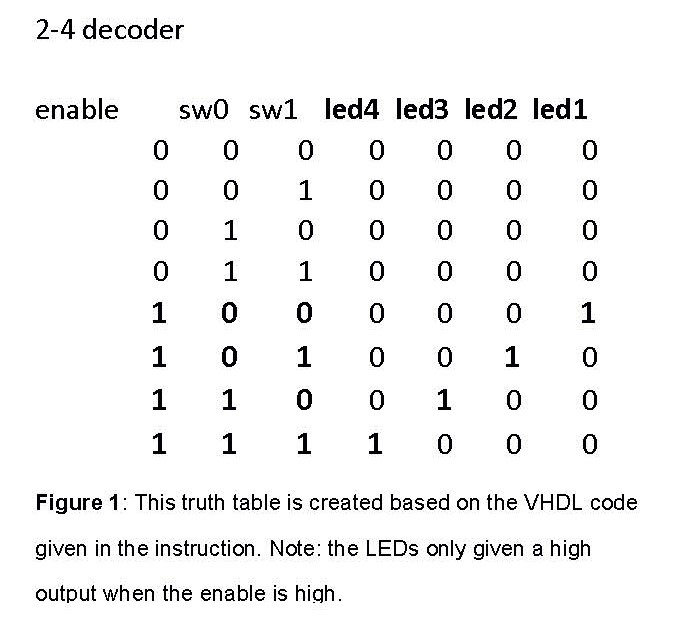
**Abstract**

The goal of this lab is to familiarize ourselves with how the Xilinx ISE Design software works. To do so, we used the program to create a 2 to 4 decoder VHSIC Hardware Description Language (VHDL) along with a simulation to prove that the code we used is right. With everything checked, we programmed our board, the Basys2, and used the Adept software to run our .bit file. Lastly, we manipulated the switches on our board and saw that the inputs resulted in the LED lighting up is in accordance with our truth table.

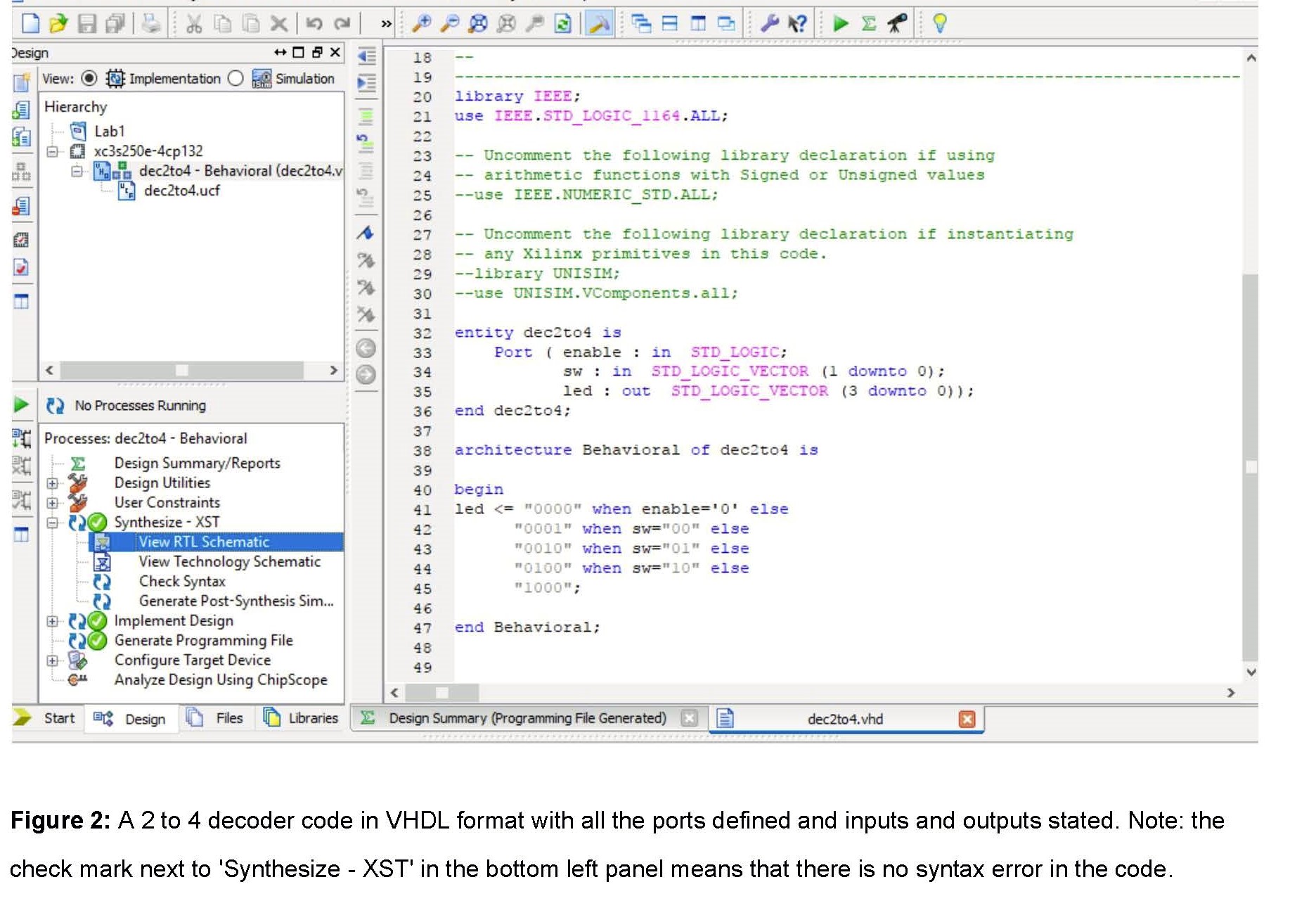
**Introduction**

1. **Creating the Code**

Before we began, we were given a VHDL code for the 2 to 4 decoder. From the code, we constructed a truth table shown in Figure 1 with three inputs (enable, sw0, sw1) and four outputs LEDs.

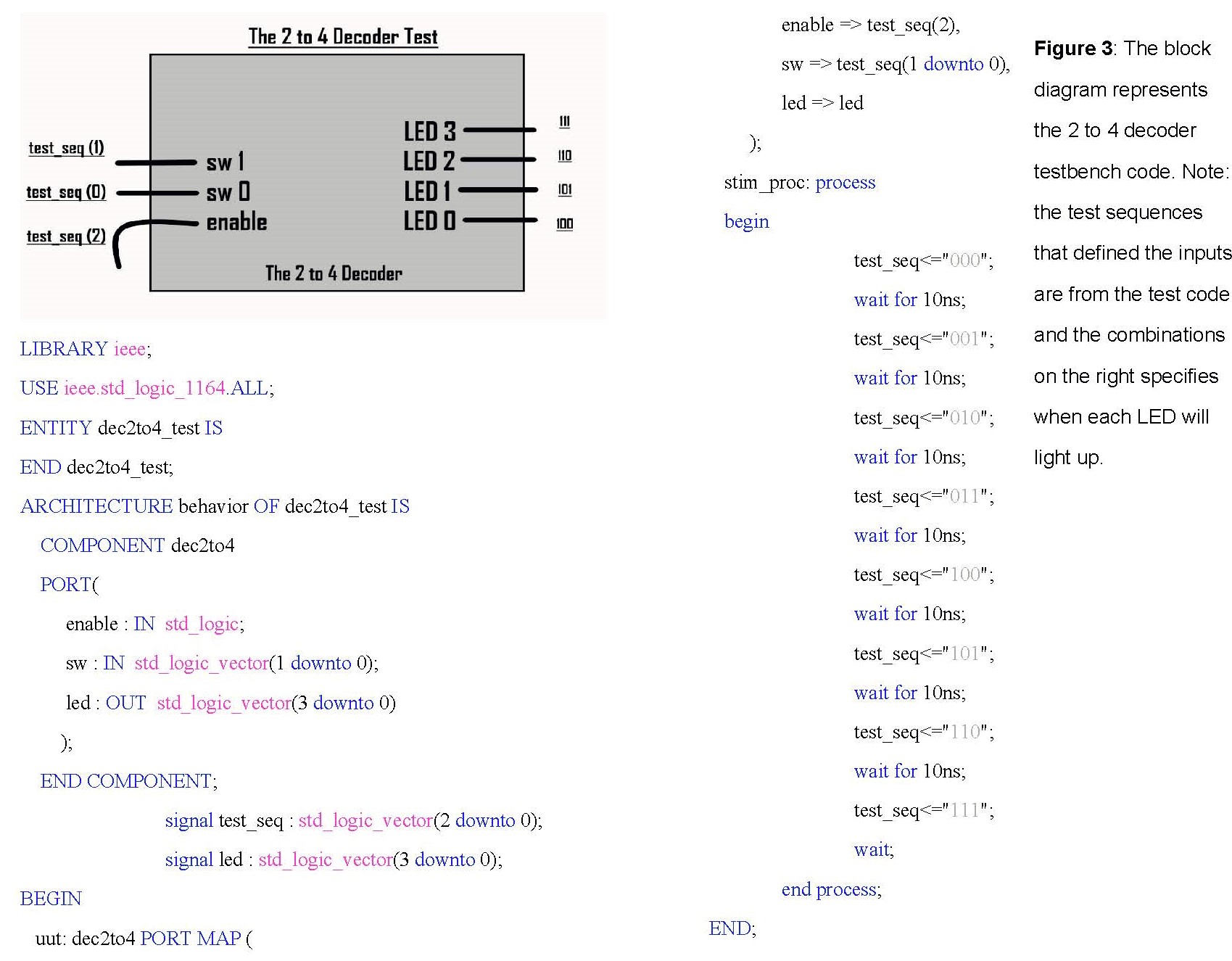


Next, we created a new project in HDL format in the Xilinx ISE software and inputting the properties of our Basys 2. Since our board is different from the one used in the tutorial, we had to change the device name to XC3S250E and the package to CP132, while keeping everything else the same. After that, we added a VHDL Module as our new source and named it “dec2to4”. For the port, we added the enable, the switch (sw), and the led, specified their directions, and their most and least significant bit (MSB and LSB). After we finished, a basic structure of our code was created and we just typed in the VHDL code given in the beginning in between the commands, ‘begin’ and ‘end Behavioral;’. The final code is shown in Figure 2 before it was saved and checked for any syntax errors.

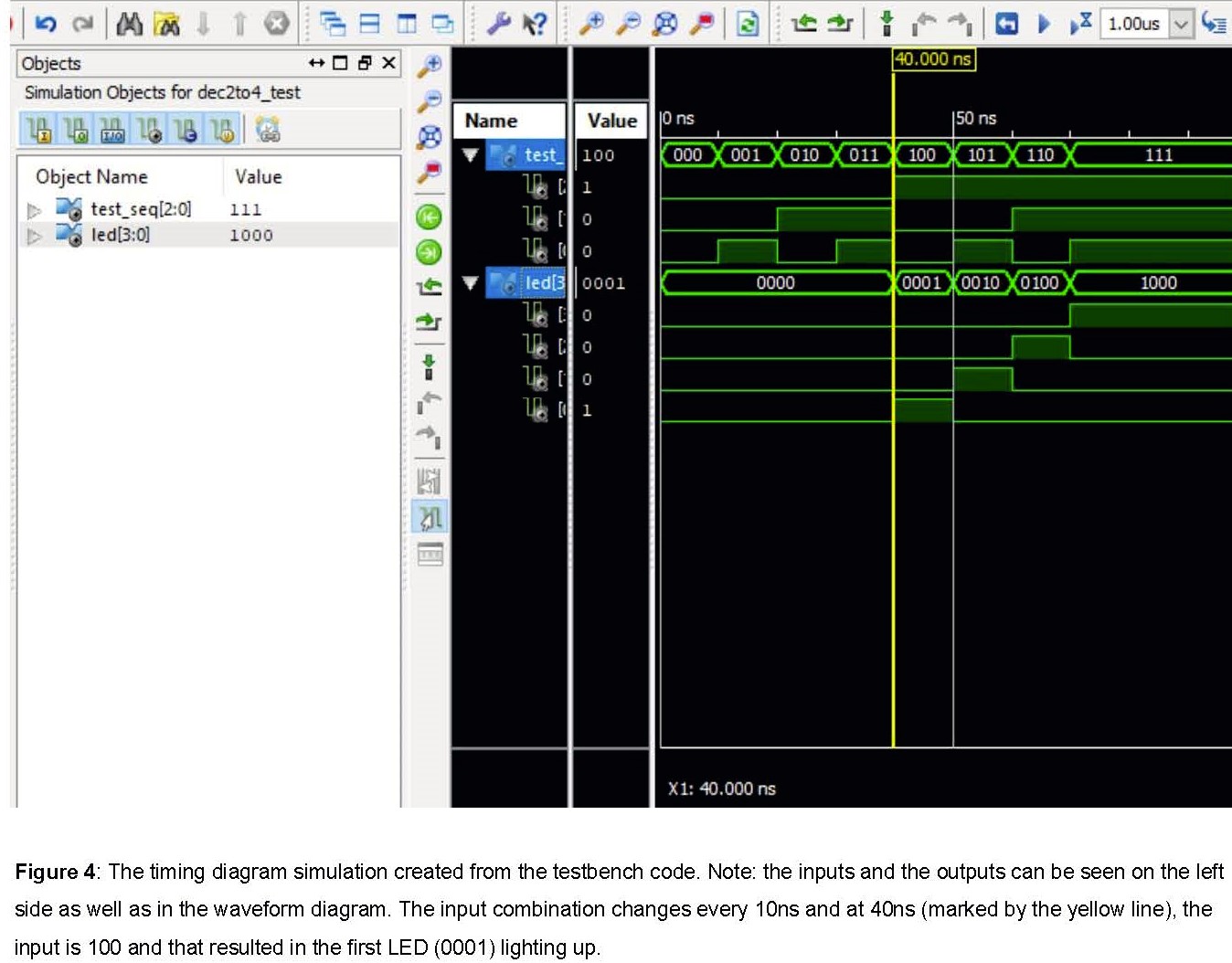


**B. Designing Simulation**

In order to test our code, we had to design a simulation for our 2 to 4 decoder. To do so, we created a VHDL testbench source and associated it with our VHDL module shown in Figure 2. After getting our skeleton code, we modified the inputs and outputs and assigned them as the units that will be tested instantaneously. Next, for the stimulus process, we assigned a test sequence to each input (000 to 111) and put in a 10ns delay between them after the output LED is displayed. A visual representation of how the signals of our testbench circuit are connected and mapped can be seen in the block diagram in Figure 3 along with our test code.

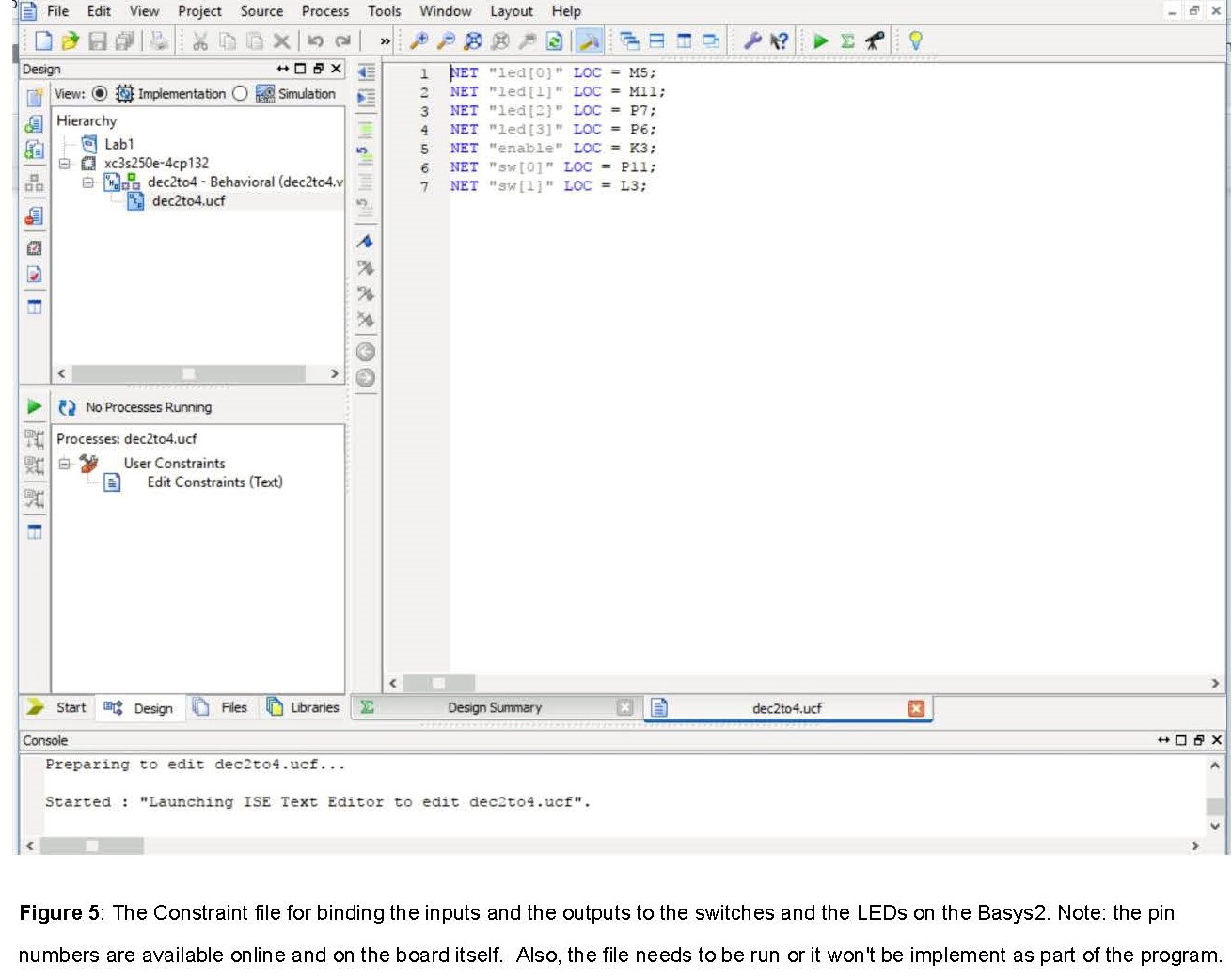


After the modification, we checked and found no syntax error in our testbench code. Since there are 8 combinations total and each one will take 10ns, we put in 100ns for the ‘Simulation Run Time’ and check the ‘Run for Specified Time’ box to ensure that our simulation will run the way we wanted it to. We, then, ran the simulation for and checked all the inputs and outputs to make sure it matches our truth table. The final timing diagram with the whole time extent as well as the all the combinations is shown in Figure 4 below.



**C. Design Implementation**

In order to define how the signals from our decoder code are mapped to the pins on our Basys2, we created an Implementation Constraints File (.urf), which is shown in Figure 5 below. Looking at our board, we wrote down all the pin numbers for the switches and the LEDs we wanted to use, so that we can include them in the constraints. After running the file, we checked for error in our implement design. When no more error is found, we generated our programming file, specifying the ‘JTAG Clock’ for the ‘FPGA Start-Up Clock’ section. Lastly, we connected our board to the computer and transferred our design, using the ‘Adept’ software. When we adjusted the switches to show different combinations, we saw that the outputs is in accordance with the truth table for our 2 to 4 decoder.



**Discussion**

After completing this lab, we know how to create codes to represent a simple circuit, and a timing diagram (simulation) that can help visualize how that circuit will work as well as checking for any error in the circuit code. In addition to that, we found out that our board, the Basys2, has different properties from the one used in the tutorial, so we had to make some changes before the program recognized some of the pin numbers in our constraint file. Another problem we encountered is when we transferred the design to the board and found out that the board does not recognize our program. The first reason was that we did not run our constraint file, and the second one is that we did not pick the ‘PC’ option on the board. After everything is fixed, the red LED next to the power switch on the board stop blinking and all the combinations produced the right outputs. In conclusion, this lab teaches us some of the functions of the Xilinx ISE Design software and the Adept software as well as showing us some common mistakes which we will be able to avoid in future labs.