PROJECT 1

LAB 1 REPORT

June 13, 2022

Tim Driscoll
Clemson University
Department of Electrical and Computer Engineering
tdrisco@clemson.edu

Abstract

The main goal of lab one centered around using the switches, keys and 7-segment displays on the lab FPGA chip. These I/O devices where implemented on the device using basic logic functionality. Upon completion of the final circuit the 7-segment displays on the FPGA chip were used to display the word HELLO and other sequences of the word. Through this project it was determined that by combing smaller sub-circuits of basic logic components into a larger final circuit basic I/O functionality can be obtained on the FPGA chip.

1 Introduction

Lab one consisted of five different parts that all combined together in order to develop a larger circuit which was implemented on the FPGA chip. Part one included the basic design of a D flip-flop which was generically designed then designed to be implemented on the board. In the final circuit the D flip-flop served as a way to store the input values that represented different characters to be outputted to the 7-segment displays. The next component that was designed was a multiplexer, in this part of the lab we went through a series of progressions when creating the multiplexer. At first the multiplexer was designed as a simple 2-to-1 multiplexer with one bit inputs. The design was then adjusted to account for eight bit inputs and outputs. The last production of the multiplexer involved creating a 5-to-1 multiplexer with three bit inputs and outputs. This last progression of the multiplexer is what was used in the final circuit and it was implemented to select which letter code would be sent to the decoder. The last component that was designed for lab one was simple decoder the decoder was used to accept a three bit input from the output of the multiplexer and convert this code to the desired seven bit output to display the letter on the 7-segment display. These components were all combined in the final circuit to perform there designed task and obtain the desired output.

2 Design

2.1 D flip-flop

A D flip-flop is a relatively simple logic component that is used to update data from the input to the output based off of the rising or falling edge from a clock. When the desired clock edge is detected the D flip-flop latches, setting the current input to the output. If the input is changed it will not update the output until the desired clock edge is received. In the lab the D flip-flop was implemented to latch on a rising edge. Using a simple if statement that checked for a rising edge. If the rising edge was detected then the input would be assigned to the output. When this design was mapped to the board, the clock was mapped to a KEY. This resulted in the D flip-flop to latch when the specified KEY was pressed in the FPGA. The inputs were mapped to SW (switches) on the FPGA, specifically switches 2,1 and 0 in the final deisgn. The switches could then be used to represent any three bit binary number. Lastly the output was mapped to a signal defined in the final circuit architecture, so it could be used as an input to the multiplexer. When specifically testing the implementation of only the D flip-flop the output was mapped to LEDR 2 through 0 in order to test functionality.

2.1.1 Test of D flip-flop

In order to test the functionality of the D flip-flop the separate VHDL file where the D flip-flop was mapped to the board was used. After creating this file where the input data was mapped to SW 2 through 0 on the FPGA, the clock was mapped to KEY 0 and the output was mapped to LEDR 2 through 0. The design was then uploaded to the board where it could be manually tested for proof of correctness. The tests included setting the input and pressing the KEY to ensure that the LEDRs updated after the KEY was pressed. The other test that was done was to change the input SWs and ensure that there were no changes to the LEDRs. It was ensured that the input was reflected by the output only after the KEY was pressed.

2.2 Multiplexer

A multiplexer is another relatively simple logic device that was implemented in this lab. A multiplexer is used to output a single piece of data from many inputs. The input is determined by selector where each input has a corresponding selection code. As described in Section 1 the development of the multiplexer in this lab went through multiple stages. The stage that was later used in the final report was the 5-to-1 multiplexer. This multiplexer used five three bit inputs and a three bit selector to have a single three bit output. The implementation of all the progressions was the same with some minor changes and additions. The 5-to-1 multiplexer included five three bit data input, a single three bit selector input and a single three bit output. There was a series of 4 when else statements to assign the output to one of the input values based off of the value of the selector. This implementation made it very easy to make adjustments to the number of inputs, because all that was needed were more when else statements. When the design of the multiplexer was mapped to the board four of the multiplexer inputs corresponded to the outputs from the d flip-flops. The last input corresponded to the SW 2 through, which replicated the inputs to the d flip-flops. The three selector bits needed for the multiplexer were mapped to SW 9 through 7 on the FPGA chip. Lastly the output from the multiplexer was mapped to a signal defined in the final lab architecture and was used as the input to the decoder. The test bench for the multiplexer showing its functionality can be seen in the appendix below.

2.2.1 Test of Multiplexer

In order to properly test the functionality of the multiplexer two different test benches were created. Two test benches were used to test two different versions of the multiplexer. In figure 1 the test bench for the 2-to-1 multiplexer is shown. This test bench shows the

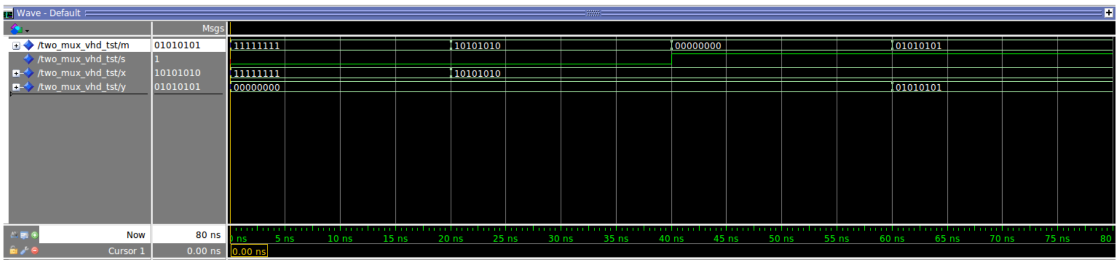


Figure 1: 2-to-1 Multiplexer Test Bench

functionality by showing that the output changes from x to y when s changes from 0 to 1. It also shows that if x changes while it is selected the output will also update. This same test is shown with y. Below is the test bench for the 5-to-1 multiplexer which involves a

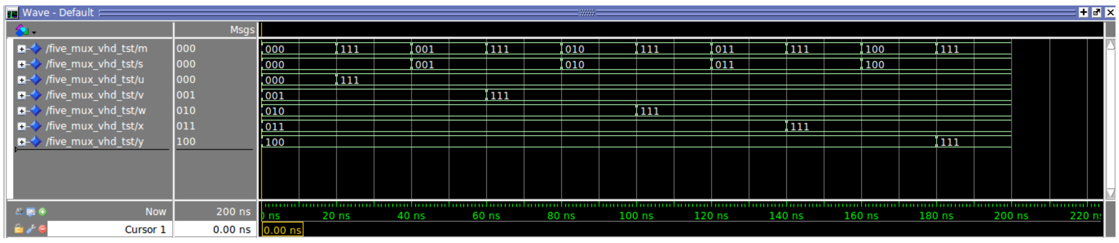


Figure 2: 5-to-1 Multiplexer Test Bench

lot more transitioning. The test bench was setup the exact same way as the 2-to-1 test bench, but all the inputs u through y were checked and verified.

2.3 Decoder

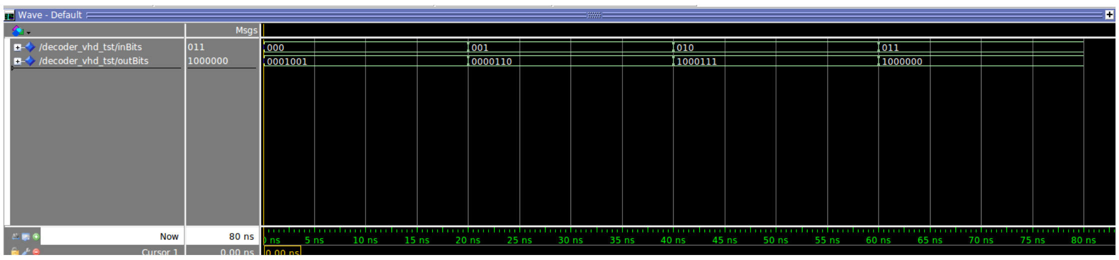
A decoder was the last logic element that was needed to complete this lab. The decoder is another simple logic element that receives a code which then corresponds to a specific value that is to be outputted. For the implementation of the decoder a simple case statement was used to evaluate the code or input bits and determine which corresponding output data should be assigned. Since each code corresponded to a unique output the case statement implementation of the design to be adjustable and easy to read. When the design of the decoder was mapped to the board it was very simple. The input of the decoder was mapped to the output signal from the multiplexer and the output of the decoder was mapped directly to the 7-segment displays on the FPGAs. Within the architecture of the decoder the specific inputs (codes) and outputs corresponded to the letters that were to be displayed on the 7-segment displays. These inputs, outputs and display representations can be seen below in table 1. The test bench for the decoder showing its functionality can be seen in appendix below.

Table 1: Decoder I/O Representation

Input	Output	7-Segment Output
000	0001001	H
001	0000110	E
010	1000111	L
011	1000000	O
Any Other Input	1111111	Blank

2.3.1 Test of Decoder

The decoder was tested both with a test bench and with a board implementation. The test bench was first used to check the general functionality of the decoder. The functionality was tested by going through all the possible input three bit input code and ensuring that the correct outputs were produced. The test bench can be observed by figure three below. The implementation of the decoder on the board was specifically done to ensure that the



```
/decoder_vhd_tst/inBits   011      000      001      010      011
/decoder_vhd_tst/outBits  1000000  0001001  0000110  1000111  1000000
```

Figure 3: Decoder Test Bench

outputs from the decoder corresponded to the correct letters (H,E,L,O,Blank).

2.4 Final Circuit

The last part of the lab involved creating a final circuit composed of all the previous sub-circuits and implementing it onto the FPGA. The final circuit's inputs included a clock, three bit data input, and a three bit selector input. The outputs of the final circuit included five seven bit outputs which were connected directly to the 7-Segment displays. The data input was represented as SW_ff and was the same three bits that were used as inputs to each one of the four D flip-flops. SW_ff also represented a direct input into the multiplexer. The clock input was then used as the clk input to each one of the d flip-flops allowing them to latch on the desired rising edge. The last input to the final circuit was SW_mux which was used as the three selector bits for the 5-to-1 multiplexer that was implemented in the final circuit design. The five, seven bit outputs were represented as HEX0-HEX4.

Figure 4 below is a representation of the final circuit that is only designed to display single letter on a single 7-segment display. In the full representation of the final circuit the outputs from the above D flip-flops are the inputs to an additional four multiplexers. These four additional multiplexers have a corresponding for additional decoders and 7-segment displays connected to them. In total the final circuit is comprised of four D flip-flops, five multiplexers, 5 decoders and five 7-Segment displays. The inputs and outputs to the final circuit still remain the same as described above. This circuit functioned by having four different available D flip-flops to store four different three bit numbers used to represent one of the letters as shown in table 1. The five multiplexers then received these four different three bit numbers as inputs as well as the value currently stored by SW_ff. The SW_mux was then used to select which one of the five inputs would be outputted by each one of the five multiplexers. The five decoders then received the three bit input from their respective multiplexer. The decoder then used the three bit input code to determine the output which can be observed in table 1 above. Finally the outputs from the decoders were assigned to by HEX0-HEX4. When the final circuit was mapped to the board the clock input was mapped to the four different KEYs on the FPGA. This allowed for a button push of an individual key to latch and individual D flip-flop. The SW_ff data input was mapped to SWs 2-0 which allowed the switches to represent any three bit binary number. The multiplexer select bits were represented by SWs 9-7 which allowed for the same functionality as just described. Lastly all the outputs HEX0-HEX4 were mapped directly to HEX0-HEX4 which corresponded to the 7-segment displays 0 through 4.

2.4.1 Test of Final Circuit

The final circuit was tested in two ways like the decoder. First to test the final circuit the a test bench was created in order to test the general functionality. The test bench was modeled like the desired simulation which was to produce the character patterns shown in table 2. This was modeled in the test bench by using the clock and SW_ff to set the word HELLO on HEX4-HEX0. After this was set the value of SW_mux was cycled from 1 to 4 and the sequences were checked. The test bench can be observed below by figure 5. After the test bench showed correct results the final circuit was uploaded to the board and the functionality was checked and confirmed.

3 Conclusion

The main goal of this lab was to gain familiarity with and learn how to connect basic I/O devices to the FPGA chip used in lab. The other main goal was to implement a circuit that used the I/O functionality from the FPGA. Through the steps included in the lab basic logic components were designed and tested on the board, these components included the D flip-flop, multiplexer and decoder. The combinations of these logic components created a final circuit that could be implemented on the FPGA and utilize the I/O. This circuit had the basic functionality of displaying a series of character patterns that can be observed in the table below.

Table 2: Final Circuit Output Results

$SW_9SW_8SW_7$	Character Patterns				
000	H	E	L	L	O
001	O	H	E	L	L
010	L	O	H	E	L
011	L	L	O	H	E
100	E	L	L	O	H

The lab provided a great learning experience and allowed for an application of many of the topics that were discussed during the beginning of lecture. Some of the main lessons that were learned throughout this lab included how to design generic logic components and then port map them to the functional I/O on the FPGA. Another major lesson that was learned was how to implement a final circuit design that included the interconnection of multiple other sub-circuits. This lab showed the power of being able to use a hierarchical design where basic logic functions were built within a single project and then later combined to create a larger circuit, with a specific functionality. Throughout the lab there were minimal functional errors that occurred but a decent amount of time was spent understanding the flow of VHDL and how the components were implemented. This biggest source of error occurred when creating the final circuit in part five, and understanding how to implement all the previously made circuits. Overall the lab provided a great learning experience where the final goal was obtained and the desired results shown in table 2 could be displayed.

References

M.Smith "Digital Computer Design" Clemson University Holcombe Department of Electrical and Computer Engineering, January 2020

M.Smith "ECE3270 Digital System Design Lab 1: Design of basic gates and I/O" Clemson University Holcombe Department of Electrical and Computer Engineering, January 2020

W. Daily, R. Curtis Harting and T. Aamodt, Digital Design Using VHDL a systems approach. Cambridge University Press

4 Appendix

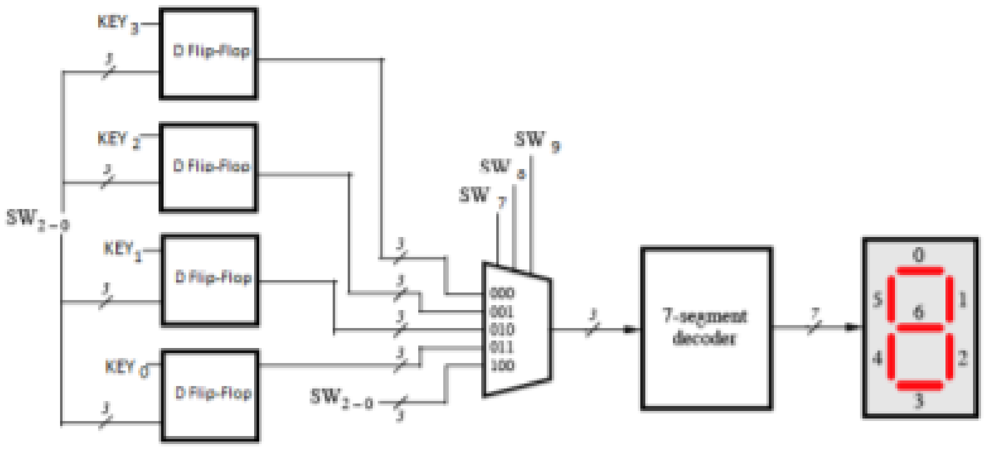
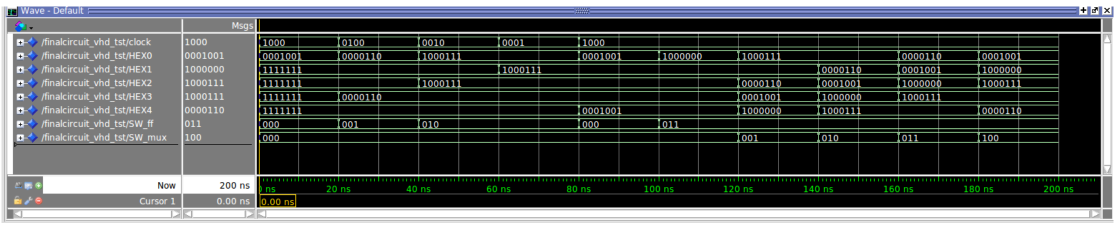


Figure 4: Final Circuit Design



```
Wave - Default
                                  Msgs
/finalcircuit_vhd_tst/clock   1000      1000      0100      0010      0001      1000
/finalcircuit_vhd_tst/HEX0    0001001   0001001   0000110   1000111             0001001   1000000   1000111             0000110   0001001
/finalcircuit_vhd_tst/HEX1    1000000   1111111                       1000111                                 0000110   0001001   1000000
/finalcircuit_vhd_tst/HEX2    1000111   1111111             1000111                                 0000110   0001001   1000000   1000111
/finalcircuit_vhd_tst/HEX3    1000111   1111111   0000110                                           0001001   1000000   1000111
/finalcircuit_vhd_tst/HEX4    0000110   1111111                                 0001001             1000000   1000111             0000110
/finalcircuit_vhd_tst/SW_ff   011       000       001       010                 000       011
/finalcircuit_vhd_tst/SW_mux  100       000                                                         001       010       011       100

                         Now  200 ns    0 ns      20 ns     40 ns     60 ns     80 ns     100 ns    120 ns    140 ns    160 ns    180 ns    200 ns
                    Cursor 1  0.00 ns   0.00 ns
```

Figure 5: Final Circuit Test Bench