

Digital Design CSCE 2114

Lab 4 Report

Zack Fravel

October 16th, 2015

zpfravel@uark.edu

Abstract

Lab 4 consisted of the continuing familiarization with the FPGA board and VHDL programming. This lab's purpose was to create a display that shows the corresponding hexadecimal values to the 4-input switch orientation.












Introduction

Before learning how to implement a solution to the problem at hand, creating a display for hexadecimal values, the lab had a walkthrough on correctly setting up a new project in Quartus for our specific FPGA board configuration. Once we had a new project, we were able to start a new VHDL file.

Once we have our Quartus file all set up and ready to go, the FPGA board is ready to program. We were given partially complete code by the lab instructor, the thing to do now is to add the correct hex values by instructing the FPGA which lights to specifically emit or turn off.

Design and Implementation

Before programming however, part of the set up in Quartus involves making sure we have the correct pins assigned for the LEDs and switches on the FPGA.

	Node Name	Direction	Location
1	 bcd[3]	Input	PIN_AE14
2	 bcd[2]	Input	PIN_P25
3	 bcd[1]	Input	PIN_N26
4	 bcd[0]	Input	PIN_N25
5	 leds[1]	Output	PIN_AF10
6	 leds[2]	Output	PIN_AB12
7	 leds[3]	Output	PIN_AC12
8	 leds[4]	Output	PIN_AD11
9	 leds[5]	Output	PIN_AE11
10	 leds[6]	Output	PIN_V14
11	 leds[7]	Output	PIN_V13

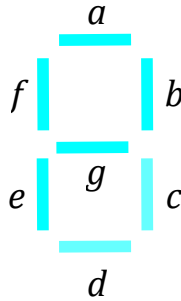
After assigning the pins to their correct location, we have to use the compiler tool to initialize and run the program on the FPGA board. We were given this code segment to start with:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY seg7 IS
    PORT ( bcd : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          leds : OUT STD_LOGIC_VECTOR(1 TO 7) ) ;
END seg7 ;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
    PROCESS ( bcd )
    BEGIN
        CASE bcd IS
            ----- abcdefg
            WHEN "0000" => leds <= "0000001" ;
            WHEN "0001" => leds <= "1001111" ;
            WHEN "0010" => leds <= "0010010" ;
            WHEN "0011" => leds <= "0000110" ;
            WHEN "0100" => leds <= "1001100" ;
            WHEN "0101" => leds <= "0100100" ;
            WHEN "0110" => leds <= "0100000" ;
            WHEN "0111" => leds <= "0001111" ;
            WHEN "1000" => leds <= "0000000" ;
            WHEN "1001" => leds <= "0001100" ;
            WHEN OTHERS => leds <= "1111111" ;
        END CASE ;
    END PROCESS ;
END Behavior ;
```

While this code is correct, it only displays hex values from 0-9. Here's the whole list of values we're shooting for:

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Now that we know the values for A-F, we're able to simply implement them directly into the code. We're given this diagram to help specify which LEDs to turn off and on in order to create the correct shape for each letter.



Once this information is given, it's as simple as looking at the shape of the letter and writing the code (format abcdefg):

```

LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY seg7 IS
    PORT ( bcd : IN  STD_LOGIC_VECTOR(3 DOWNT0 0) ;
          leds : OUT STD_LOGIC_VECTOR(1 TO 7) ) ;
END seg7 ;
ARCHITECTURE Behavior OF seg7 IS
BEGIN
    PROCESS ( bcd )
    BEGIN
        CASE bcd IS
            ----- abcdefg
            WHEN "0000" => leds <= "0000001" ;
            WHEN "0001" => leds <= "1001111" ;
            WHEN "0010" => leds <= "0010010" ;
            WHEN "0011" => leds <= "0000110" ;
            WHEN "0100" => leds <= "1001100" ;
            WHEN "0101" => leds <= "0100100" ;
            WHEN "0110" => leds <= "0100000" ;
            WHEN "0111" => leds <= "0001111" ;
            WHEN "1000" => leds <= "0000000" ;
            WHEN "1001" => leds <= "0001100" ;
            WHEN "1010" => leds <= "0001000" ;
            WHEN "1011" => leds <= "1100000" ;
            WHEN "1100" => leds <= "0110001" ;
            WHEN "1101" => leds <= "1000010" ;
            WHEN "1110" => leds <= "0110000" ;
            WHEN "1111" => leds <= "0111000" ;
            WHEN OTHERS => leds <= "1111111" ;
        END CASE ;
    END PROCESS ;
END Behavior ;

```

Results

This lab was relatively straight forward. The only problem I encountered was in the determination of which lights to turn off or on in making the shapes of the different letters, I was confused because a '1' turns the light off while a '0' keeps the light on. Once I had figured that out, it was very simple to implement. Because of the nature of the display, some letters have to be represented in lower case in order to differentiate between, for example, the number 0 and the letter D.

Conclusion

The original problem of this lab was to implement VHDL code in order to program a display on the FPGA board that uses the 4 input switches to display 16 different hexadecimal values (0-9 A-F). An extension that could be made to the project would be to simply expand the truth table to allow even more values to be displayed on the FPGA.