EGCP 1010
Digital Logic Design (DLD)
Laboratory #8

Creative Design Using Altera CAD

Tools and UP1 Board

Prepared By:
Alex Laird

on

November 15, 2007

Lab Partner:
Ryan Morehart

Objective:

The goal of this laboratory is to provide the student with a complete and manageable design problem, from idea to design to implementation using Altera Tools and a UP1 Board. My partner and I decided to make morse code like machine.

Procedure and Results:

My partner and I decided to do a circuit similar to a Morse Code interpreter. Using the two buttons on the UP1 Board, the user would press them in such a combination, one representing short and one representing long, and after three combinations, the Seven Segment Display would show what number (in hex) was inputed.

We decided that as the user pressed one of the push buttons, the Display would show how many button presses had occurred up to that point. Then, on the third button press, the Display would display the number.

The left button will represent a short signal, and the right button will represent long. For combinations, they are shown below in Table 1.

Combination	**Display**
...	**A**
.._	**B**
._.	**C**
.__	**D**
_..	**E**
_._	**F**
__.	**7**
___	**8**

Table 1: Morse Code Combinations

After much trial and error with a poorly built project, we scrapped our project and started a new one, this one built off of the template for the Altera Lab 7.

We added to Seven_Seg_Display symbols to the schematic, each of which would display control either the left or the right Seven Segment Display on the UP1 Board. For the left Seven Segment Display we simply grounded all inputs to display a constant zero. This display was not used in our program. We tied the output for the right Seven Segment Display to our actual implementation.

For the push buttons we made a simple de-bouncing circuit with D Flip-Flops so electrical charges wouldn't fire multiple times as the button got closer and closer to a connection when it was being pressed. To do this, we tied the actual button press as the clock trigger for a constantly high D Flip Flop. The output for the first D Flip-Flop became the input for the second. The output for the second was inverted and brought back around as a low reset for the first Flip-Flop. This made it so, even if several hundred charges jumped back and forth in the instant button was pressed, only one actual button press would register. The button press was outputted. This de-bouncing circuit was used for both the left and the right button.

An LPM_Counter was added to interpret the count of the clock. It is output as an array with clock values from twenty-two (22) to zero (0), each value (in incrementing order) being half as fast as the previous value.

To keep track of what number would be displayed from the keypad, the inputs were kept track of in a LPM_ShiftReg. On the third button click, the output is passed to the big_mux component. big_mux had previously been incrementing, keeping track what the current key press was (one (1) or two (2)). Select, which comes from an LPM_Counter which determines the current number of key presses in the series, is the array used to decide which input of the mux to accept. Seven_Seg_Driver. We rewrote the driver to handle what to do with given inputs and how to display them as outputs.

The final circuit is shown below in Figure 1. The rewritten Seven_Seg_Driver is shown in Figure 2.

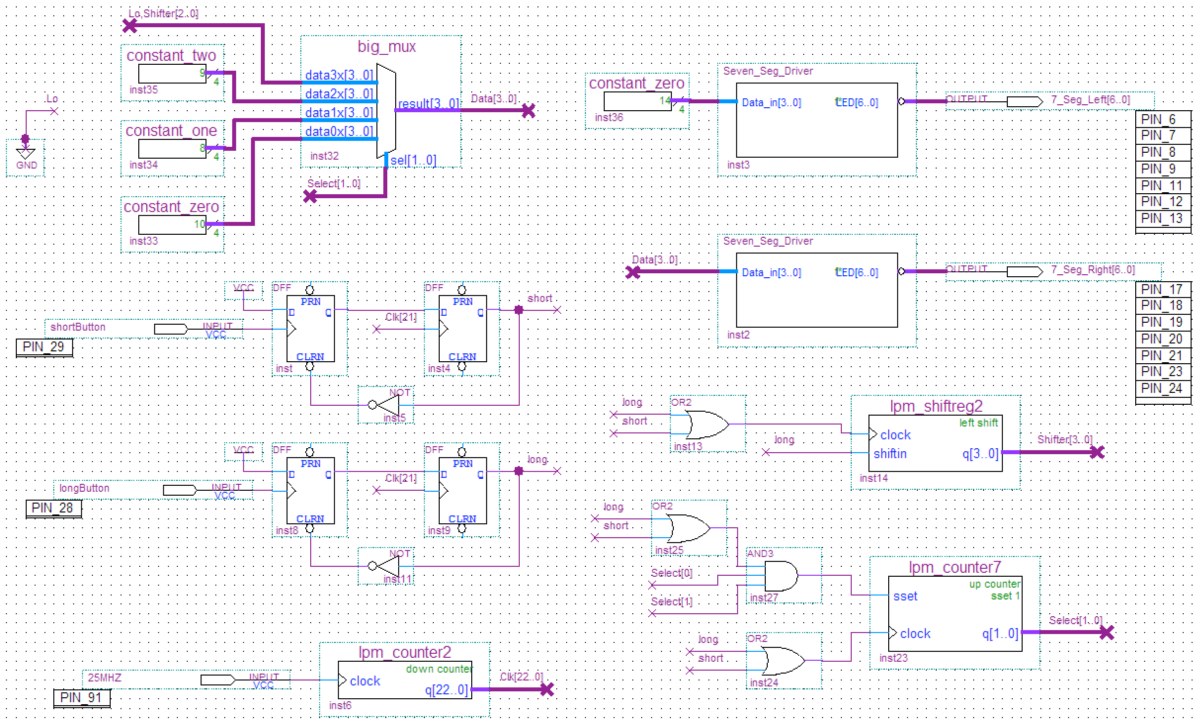

Figure 1: Final Schematic for Morse Code Generator

```
--****** VHDL Design file for a Binary to Seven Segment Driver ***
library IEEE;
use IEEE.std_logic_1164.all;

Entity Seven_Seg_Driver is
        port(Data_in: in  STD_LOGIC_VECTOR (3 downto 0);
            LED: out STD_LOGIC_VECTOR (6 downto 0));
End Seven_Seg_Driver;

--****** This Case statement is very close to a truth table ***

Architecture Structure of Seven_Seg_Driver is
Begin
        Process(Data_in)
        Begin
                Case Data_in is
                --   comment       Segment"ABCDEFG"
                        when "0000" => LED <= "1110111";
                        when "0001" => LED <= "0011111";
                        when "0010" => LED <= "1001110";
                        when "0011" => LED <= "0111101";
                        when "0100" => LED <= "1001111";
                        when "0101" => LED <= "1000111";
                        when "0110" => LED <= "1110000";
                        when "0111" => LED <= "1111111";

                        when "1000" => LED <= "0110000";
                        when "1001" => LED <= "1101101";
                        when "1010" => LED <= "1111110";

                        when others => LED <= "1001001";--this should never occur
                End Case;
        End process;
End Structure;
```

Figure 2: Seven_Seg_Driver.vhd Rewritten File

Conclusion and Suggestions:

This lab gave me a good idea of what it’s like to start with a large scale project from scratch; to imagine an idea and follow through with it. It illustrated that even some ideas, which seem to be very big in our head, can be broken down into simpler, individual solutions to work together towards your final project. I enjoyed this lab very much after we finally got our circuit working!