Lab 2: Multiplexed Display

Peter Johnson[1]

E155 Fall 2018 Section 2, September 20, 2018

***Abstract*— This paper describes the, circuit design, programming and testing of the MuddPi Mark IV such that it can control two seperate off-board 7-segment hexadecimal displays based on inputs from switches while only using a single 7-bit decoder.**

I. Introduction

In this lab, the MuddPi Mark IV utility board was wired to control a second 7-segment display and receive another set of 4 inputs from a second set of switches. Once wiring was complete, SystemVerilog was written to control the 7-segment displays which would display the hexadecimal number conveyed by the four switches and the on board LEDs which would display the sum of the inputs. To achieve the dual display with a single 7 segment decoder, time multiplexing was used.

II. Design and Testing Methodology

I approached the design of this system by first considering the requirements of the 7-segment displays. Since I had a single 7-segment decoder I would need to time multiplex the signal it was processing. This means that it would switch back and forth; reading one set of input and turning on the appropriate segments on one display before switching to the other inputs and the other display at a speed not visible to the human eye (60 Hz). This also meant that a pin on the FPGA would correspond to a segment on both displays, but one display would be off when the other was on, ensuring the numbers would not bleed together. For this toggling, I referred to the E85 Ch.5 lecture slides which went over a counter used to toggle the most significant bit of a variable at a desired rate. Given the relationship:

$$f_{Out} = f_{Clk} * p \; / \; 2^N$$

As well as the know values: the desired frequency was 60 Hz, Clock was 40 MHz and word size was 32 bits I then solved for p:

$$60\ Hz = 40\ MHz * p \; / \; 2^{32}$$

$$p \approx 6442$$

This counter went into its own module since it seemed appropriate to reduce the complexity at my top level module. I used a HDSP-5721 common anode dual 7-segment display. Following this idea of only one set of inputs being used at a time, only one display would be turned on at a time. To do this, I hooked up a 2N3906 PNP transistor between the anode of each display and the 3.3V pin. The transistors would be turned on by a separate pin from the FPGA, toggling at the same rate 60 Hz rate. This was done because the max current from a 3.3 V I/o pin is 4 mA which is insufficient for the current draw of the display (10mA per segment). To turn on a display, a 0 would need to be passed by the FPGA to the base of the transistor (property of PNP). This would result in the base being a lower voltage than the emitter allowing flow through the diode between them. The drop of the diode would result in the base being at 2.6V. The resistor for this signal was calculated to be 3.7kΩ (see attached). Next was the analog requirements to turn on each of the LED diodes. To turn on a diode segment, a 0 not a 1 needed to be passed to the pin of that segment since the pins were the cathodes. By passing a 0 to the cathode, the cathode would be pulled to 0 producing a voltage drop across the diode resulting in it producing light. The desired current flowing through an LED is roughly 5-20mA. The drop across an LED is roughly 1.7-2.7 V and we know that the base would be at 2.6V. This allowed me to solve for an appropriate resistor using Ohm's Law:

$$V=IR$$

I chose I=10 mA and V=2.6-2.5V (close to max brightness), so R=10Ω. I then programmed the FPGA via JTAG thinking it was quicker to test by actually flipping the switches. I then went through all 16 positions ensuring the correct value was displayed.

Last was the logic for the onboard LEDs. Instead of writing a full adder module, I just used the behavioral "+" which implied a full adder in the hardware.

III. Code and Schematics

Refer to attached schematics and code.

IV. Results and Discussions

All of the prescribed tasks were accomplished: the LEDs show the sum of the switches, the two 7-segment displays correctly display two independent hexadecimal values from the switches and when reset, the board is able to return to the program since PROM was correctly programmed.

V. Conclusions

The lab was successfully completed. The Mudd Pi Mark IV was fully assembled and tested. The on-board LEDs display the sum of the inputs, and the 7-segment displays are able to correctly display hexadecimal values based on inputs from the switches. Once the board is reset or turned on and off, the LEDs continue to function, showing the PROM was successfully programed. I gained experience with time multiplexing, sharing hardware and some transistor theory. In total I spent 7 hours.

[1]Peter Johnson is with Department of Engineering Harvey Mudd College, Claremont, CA 91711 pjohnson@hmc.edu