Lab 4 – Programming Combinational Logic on the Basys FPGA Board

Thomas Chin
861290572
EE120A, Section 021

Overview

Part 1

In part A, I used a given schematic and constraint file to test out a call button on the BASYS2 board. An LED on the board would turn on when the button was pressed, and turn off when pressed again. Part B implemented the same system in part A through Verilog.

Part 2

In part A, I was given a rising edge detector FSM to analyze. Part B implemented the FST from part A. We filled in the ZERO, CHANGE, and ONE states using a given document in the comments of the code. The output was brief one-clock tick that occurred when a button on the BASYS2 board was pressed.

Part 3

In part A, we analyzed a schematic overview and schematic utility used in displaying numbers and letters on the BASYS2 7 segment decoder, based on binary and hexadecimal form.
In part B, we designed the components of part A's schematic overview in verilog. I learned how the LEDs for each symbol are connected and how to trace each path to create a pattern.

New Concepts

One new concept was learning to generate symbols from Verilog functions

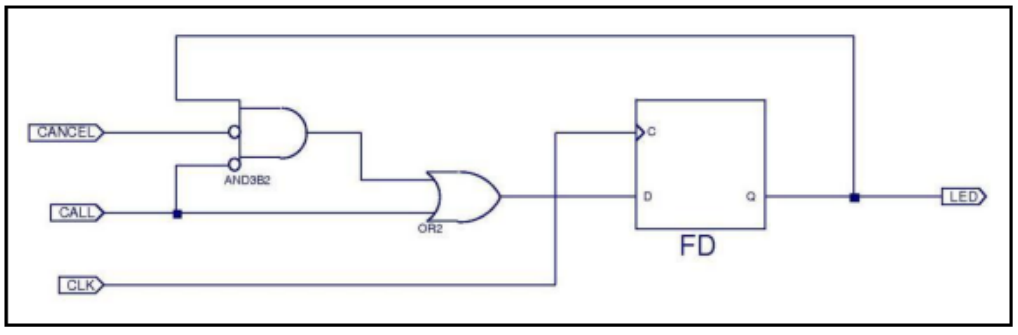
Analysis

Procedure:

1. Xinlinx ISE Design and Synthesis Environment
2. Created configuration files
3. Used Adept to download files onto Basys

Records

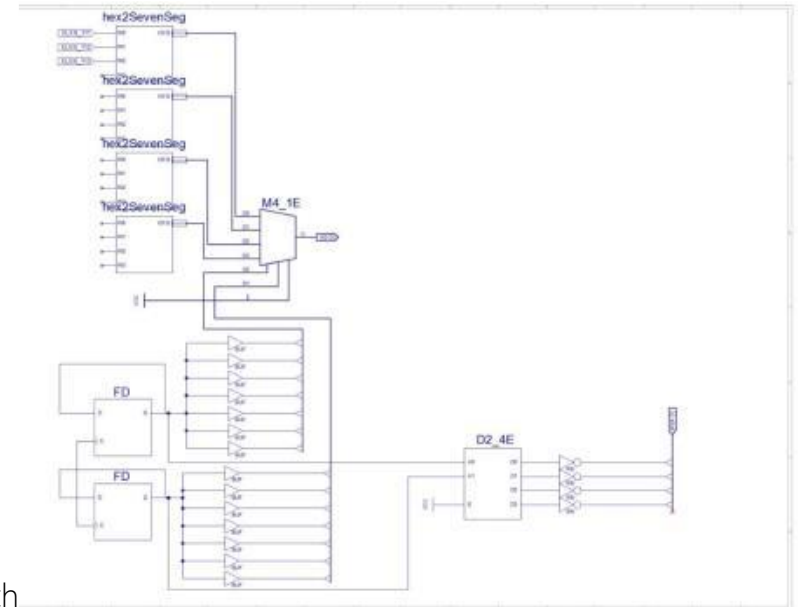
Part 1:



Discussion

MY system works in accordance with the lab manual. The most trouble I had was figuring out how to create each symbol using Verilog.

Conclusion



The purpose of th e lab was to understand the concept of clocks using verilog code in parts A and B, and to improve upon lab 3's objective by using multiple switches to output numbers and letters on all four 7-segment LEDs, using verilog black boxes on our schematic.

Questions

1. If the clock signal is of low frequency, there will be some lag in between the time it takes for the light to actually turn on and off after we hit the switch.

2.

```
module testBench(
        input wire CLK,
        input wire CALL ,
        input wire CANCEL ,
```

```
        output reg LED );

        reg c_state ;

        // Combinatorial block
        always @(*) begin
        case ({CANCEL, CALL})
                2'b00: c_state = LED;
                2'b01: c_state = 1;
                2'b10: c_state = 0;
                2'b11: c_state = 0;
        default : c_state = 'd0 ;
        endcase
        end

        // Sequential block
        always @( posedge CLK ) begin
        LED <= c_state ;
        end
endmodule
```