Lab 3

12pm-2pm

Calvin Liu

804182525

Darin Minamoto

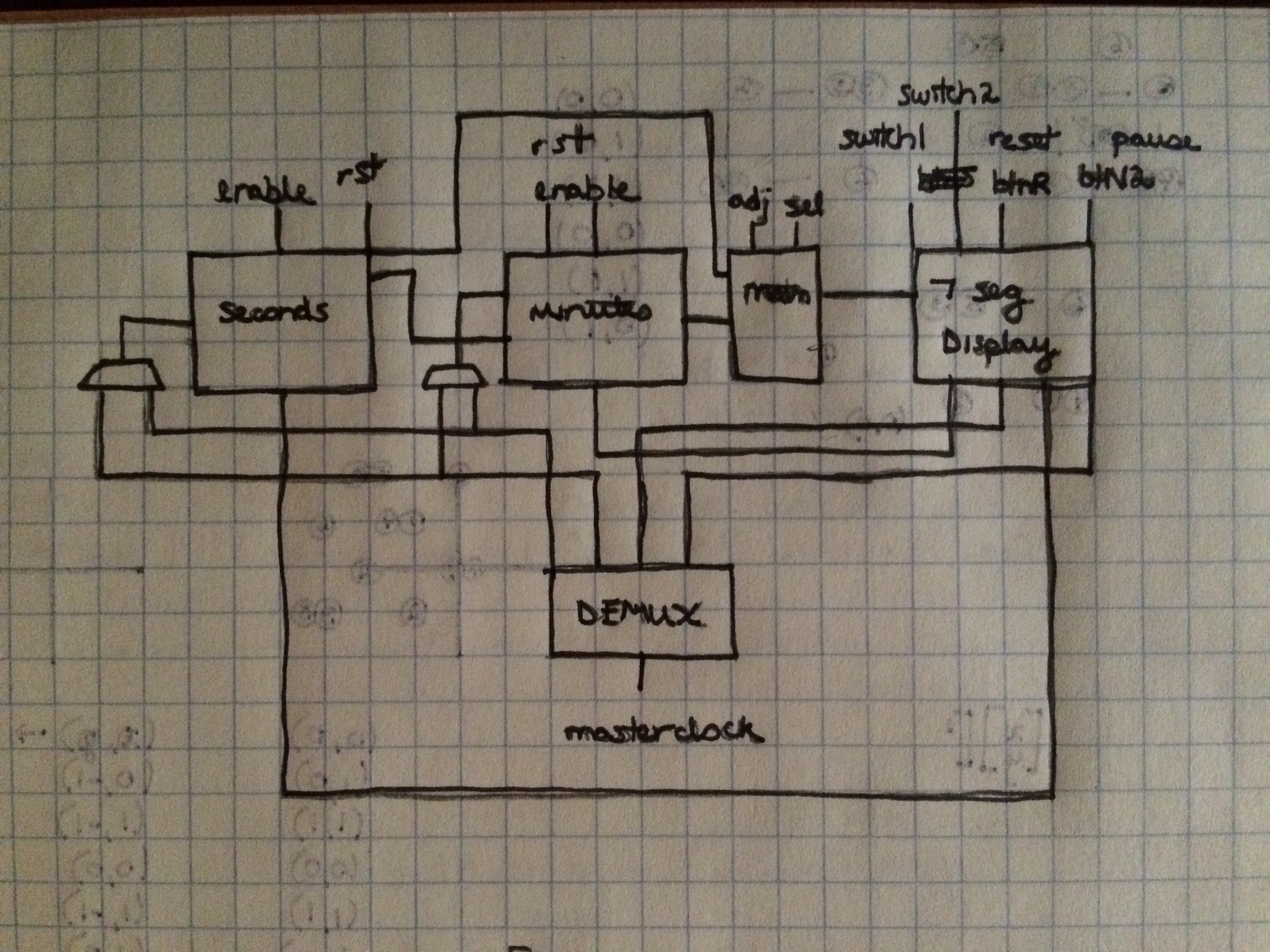
704140102

**Introduction:**

The purpose of the lab was to create a stopwatch through the FPGA circuit board using the four LED seven segment displays. The left two digits display the minutes while the right two seven segment displays show the seconds. The stopwatch is supposed to have different ways of counting as well. The different ways of counting are at a 1Hz and 2Hz. These are produced via a clock. There is a setting on the stopwatch that lets it reset the timer, pause the timer, select whether you want to use the 2Hz counter and apply it to which denomination: either seconds or minutes.

**Design:**

Our design includes making multiple modules in order to create the stopwatch and then linking then together. We made a seconds counter module, a minutes counter module, a clock module, a main module, and a seven segment display module. Below is the high level design:



For the seconds counter, we just made a simple counter that counts based on the clock signal being fed into the seconds counter and had to account for the overflow at 60 seconds.

For the minutes counter, we just made a simple counter, but the minutes counter must know whether or not the seconds counter has overflowed past 60 seconds.

module minutes(clkmin, rstmin, enablemin, countmin, overflow

);

input clkmin, rstmin, enablemin, overflow;

output [6:0] countmin;

reg [6:0] countmin = 0;

wire overflow;

always @(posedge clkmin, posedge rstmin)

begin

if(rstmin || countmin == 100)

countmin <= 7'b0000000;

else if(enablemin || overflow == 1)

countmin <= countmin + 1;

end

endmodule

The clock module is a type of selector that takes in a master clock signal and produces one of four signals and outputs it to the modules that needs that frequency. If the clock signal is multiplexed based on the different clocks signals.

The main module is a module we made to connect the different clock frequencies and the minutes module and the seconds module together. This is then outputted to the seven-segment display which contains the values of the minutes and seconds we need to display. The main also controls the adj, sel, and rst options that the fpga board takes.

always @(posedge masterclock)

begin

//////////////Clock Adjustments//////////////////

if(Hz1 == 50000000)

// if (Hz1 == 100)

begin

Hz1 <= 0;

clock1 <= ~clock1; // toggle clock

end

else

begin

Hz1 <= Hz1 + add1;

end

///////////////2 Hz////////////////////////////

if(Hz2 == 25000000)

// if (Hz2 == 50)

begin

Hz2 <= 0;

clock2 <= ~clock2;

end

else

begin

Hz2 <= Hz2 + add1;

end

///////////////////500Hz//////////////////////

if(Hz3 == 100000)

// if(Hz3 == 1)

begin

Hz3 <= 0;

clock3 <= ~clock3;

end

else

begin

Hz3 <= Hz3 + add1;

end

//////////////////Blinking 5Hz/////////////////

if(Hz4 == 10000000)

begin

Hz4 <= 0;

clock4 <= ~clock4;

end

else

begin

Hz4 <= Hz4 + add1;

end

end

The seven-segment display is then used to take in the clock signal we want to use and send it to the FPGA. The clock signal is set at a large frequency since we need to multiplex the signals and change them throughout the cycles. After parsing the value of the individual digits from the minutes and seconds, each cycle will output the value of the parsed values to the corresponding register on the FPGA.

always @(posedge clk)

begin

secondsCountReg = secondsCount;

minutesCountReg = minutesCount;

seconds1 = secondsCountReg % ten;

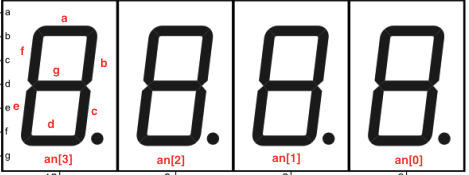
seconds2 = secondsCountReg / ten;

minutes1 = minutesCountReg % ten;

minutes2 = minutesCountReg / ten;

end

Since the clock signal is going at a fast speed, it will look like all the digits on the FPGA are being lit up all at the same time while another frequency increases the minutes and the seconds.



The seven-segment dsplay is chosen through a case statement:

always @(\*)

begin

if(enableBlink)

begin

case(caseStatements)

0:

begin

sseg\_num = seconds1;

an = 4'b1110;

end

1:

begin

sseg\_num = seconds2;

an = 4'b1101;

end

2:

begin

sseg\_num = minutes1;

an = 4'b1011;

end

3:

begin

sseg\_num = minutes2;

an = 4'b0111;

end

endcase

end

end

always @(\*)

begin

case (sseg\_num)

0: seg = 8'b11000000;

1: seg = 8'b11111001;

2: seg = 8'b10100100;

3: seg = 8'b10110000;

4: seg = 8'b10011001;

5: seg = 8'b10010010;

6: seg = 8'b10000010;

7: seg = 8'b11111000;

8: seg = 8'b10000000;

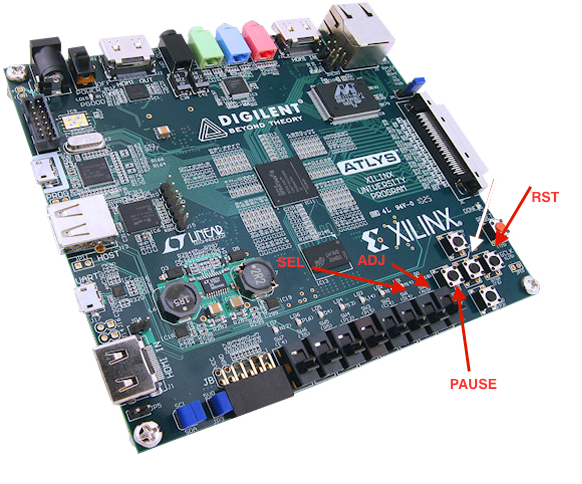
9: seg = 8'b10010000;

default: seg = 8'b10111111;

endcase

end

Below is the intended design of the FPGA that a user can use for reference in using the stopwatch:



During the design of it, we were not able to implement the pause button correctly, but everything else was working just fine. The pause was most likely not working because we did not implement the debouncing correctly.

**Simulation and Testbench:**

We tested each module with their separate testbenches for each module. Most of testbenches dealt with just setting the clock to go for a certain time and then toggling it back and forth after a certain amount of time since the code in the module ran by itself with just a clock.

Most of the testbench files look like this for all the modules since the master clock is only thing that needs to be set in order to run the clock and demultiplex the output into the corresponding modules:

Example of the testClock.v

initial begin

// Initialize Inputs

masterclock = 0;

adj = 0;

sel = 1;

rst = 1;

#10;

rst = 0;

// Wait 100 ns for global reset to finish

#500000000;

// Add stimulus here

end

always begin

#1 masterclock = ~masterclock;

end

The most important thing to check when running the testbench was running the seven segment display module because that module used all the other previous modules and thus if we ran the seven segment display then we could still see the output and waveforms of the modules that were used

Example of the sevenSeg.v

initial begin

// Initialize Inputs

clk = 0;

btnR = 0;

btnS = 0;

btn2 = 0;

btnMid = 0;

// Wait 100 ns for global reset to finish

#100;

// // test for blinking lights

//

// // turn on ADJ

// btn2 = 1;

//

// #1000;

//

// // turn off ADJ

// btn2 = 0;

// // test for pause button

//

// #10000;

//

// btnMid = 1;

// #10;

// btnMid = 0;

//

// #10000;

//

// btnMid = 1;

// Add stimulus here

end

always begin

#1 clk = ~clk;

end

During the simulation, the pause button was not working correctly some of the time. Although it did work, sometimes the pause button would not stop the timer/renable it. We believe this error is caused by the lack of success in implementing a debouncer to make sure the singal is not “bouncing.” Everything else in the simulation worked accordingly besides this tiny aspect.

**Conclusion:**

This lab turned out to be significantly more difficult than the past lab we did since we needed to design up to 5 modules and had to demultiplex the clock from a 100MHz clock and reduce the frequency into a 1Hz, 2Hz, 500Hz, and 5Hz clock. In addition to designing 5 different modules we needed to find a way to connect them and be able to output the results to the seven-segment FGPA board and be able to reset the timer and select a certain time to adjust. We managed to do it successfully and implemented the sel and adj based on the two switches and rst and pause based on the buttons and LED seven-segment display output correctly. The seven-segment display shows correctly at the start of compilation after running the program.

The problem that we had to face was that the pause button does not behave correctly 100% of the time. I believe this is because we did not implement debouncing correctly and because of this, the signal of the clock while the button is pressed fluctuates back and forth and may not trigger what the button was meant to do.

**Individual Efforts:**

Darin did a lot of work for the clock, minutes, and seconds testing and fixed some of the code while out of the lab while Calvin worked on the lab report. Together, most of the code was written during the lab and during lab office hours together. This includes the seconds module, the minutes module, the clock module, and the seven segment display. The main function was first implemented by Darin and then it was fixed and modified by Darin and Calvin as we continued with the lab. The implementation of the FPGA was worked on by Calvin and Darin together.