Assignment 3

Carson Bolinger, Roberto Solis, Carson Wagner

February 23, 2024

1 Introduction

In Assignment 3, we were asked to generate a 10 MHz square wave with a 50% duty cycle. Since we were a group of three, we had to add a button so that when pressed, it would go to 8 MHz. On the hardware side, we used a 6MHz crystal, which was then followed by a clock multiplier/divider to be able to get a frequency of 10MHz. Again, being a group of three, we had to include a button that when pressed would give us an 8MHz wave.

2 Final Code

```
/*
===============================================================================
 Name        : Carson Wagner, Carson Bolinger, Roberto Solis
 Author      :
 Version     :
 Copyright   : 2/13/2024
 Description : main definition
===============================================================================
*/


#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>
#include <stdio.h>

// Main PLL and PLL0
#define CLKSRCSEL (*(volatile unsigned int *)0x400FC10C)
#define PLL0CFG (*(volatile unsigned int *)0x400FC084)
#define PLL0CON (*(volatile unsigned int *)0x400FC080)
#define PLL0STAT (*(volatile unsigned int *)0x400Fc088)
#define PLL0FEED (*(volatile unsigned int *)0x400FC08C)
#define CLKSRCSEL (*(volatile unsigned int *)0x400FC10C)



// Clock Divider
#define CCLKCFG (*(volatile unsigned int *)0x400FC104)
#define PINSEL3 (*(volatile unsigned int *)0x4002C00C)
#define CLKOUTCFG (*(volatile unsigned int *)0x400FC1C8)


// Initialize Switch to change Frequency from 10 MHz to 8 MHz
#define FIO2DIR (*(volatile unsigned int *)0x2009c040) // Switch; P2.0, PIN 42
#define FIO2PIN (*(volatile unsigned int *)0x2009c054) //


void pllSetup10MHz() {

// Disconnecting the PLL
```

```
43 PLL0CON &= ~(1 << 1);
44 PLL0FEED = 0xAA;
45 PLL0FEED = 0x55;
46
47 // Disabling PLL
48 PLL0CON &= ~(1 << 0);
49 PLL0FEED = 0xAA;
50 PLL0FEED = 0x55;
51
52 // Setup the Clock Divider
53 CCLKCFG = 1 - 1;
54
55 // Select OSC_CLK (External Crystal)
56 CLKSRCSEL = 0;
57
58 // M = 12, NEW M = 45
59 PLL0CFG = 44; // (45 - 1) = 44
60 PLL0FEED = 0xAA;
61 PLL0FEED = 0x55;
62
63 // Enable the PLL
64
65 PLL0CON |= (1 << 0);
66 PLL0FEED = 0xAA;
67 PLL0FEED = 0x55;
68
69 // Wait for PLL0 to check if it is Locked
70
71 while(((PLL0STAT >> 26) & 1) == 0) {
72
73 // do nothing until it locks
74 }
75
76 // Dividing 360 MHz/36 = 10 MHz square wave, K = 36
77 CCLKCFG = 35; //(36 - 1)
78
79
80 // Connecting PLL0
81 PLL0CON |= (1 << 1);
82 PLL0FEED = 0xAA;
83 PLL0FEED = 0x55;
84
85 }
86
87
88 void pllSetup8MHz() {
89
90 // Disconnecting the PLL
91 PLL0CON &= ~(1 << 1);
92 PLL0FEED = 0xAA;
93 PLL0FEED = 0x55;
94
95 // Disabling PLL
96 PLL0CON &= ~(1 << 0);
97 PLL0FEED = 0xAA;
98 PLL0FEED = 0x55;
99
100 // Setup the Clock Divider
101 CCLKCFG = 1 - 1;
102
103 // Select OSC_CLK (External Crystal)
104 CLKSRCSEL = 0;
105
106 // M = 45
107 PLL0CFG = 44; // (45 - 1) = 44
108 PLL0FEED = 0xAA;
109 PLL0FEED = 0x55;
110
111 // Enable the PLL
112 PLL0CON |= (1 << 0);
113 PLL0FEED = 0xAA;
```

```
PLL0FEED = 0x55;

// Wait for PLL0 to check if it is Locked

while(((PLL0STAT >> 26) & 1) == 0) {

// do nothing until it locks
}


// Dividing 360 MHz/8 = 8 MHz square wave, K = 45
CCLKCFG = 44; //(45 - 1)


// Connecting PLL0
PLL0CON |= (1 << 1);
PLL0FEED = 0xAA;
PLL0FEED = 0x55;

}


// Function to create a delay in milliseconds

void wait_ms(int as) {

float a = 0.002715;
float b = 0.1;

    volatile int i;
    as = (as-b)/a;

    for (i = 0; i < as; i++) {

    //do nothing

    }

}


int main(void) {

    // Bits of PINSEL3 set to 10
PINSEL3 &= ~(1 << 23);
PINSEL3 |= (1 << 22);
CLKOUTCFG |= (1 << 8);

FIO2DIR &= (0);



// Call PLL Setup Method for 10MHz
pllSetup10MHz();

    while(1) {

    // When Switch 1 (P2.0) is pressed display 8MHz frequency,
    if((~(FIO2PIN >> 0) & 1)) {

    pllSetup8MHz();
    wait_ms(500);
while ((~(FIO2PIN >> 0) & 1)){
//do nothing
}
    }
    else
    {
    // Display 10 MHz frequency
    pllSetup10MHz();
while (((FIO2PIN >> 0) & 1)){
```

```
185 //do nothing
186 }
187     }
188
189    }
190
191     return 0;
192  }
```

2.1 Calculations for Software

To determine the requirements for generating a 10 MHz wave, we used some of the formulas from the Modules. We used $CCLK = \frac{2M}{N} \times \frac{sysclk}{K}$ and we had to follow that the clocks multiplier oscillator was between 275 MHz $\leq \frac{2M}{N} \times \frac{1}{\text{sysclk}} \leq 550$ MHz. Now to find what we needed for the 10 MHz wave, we used 360MHz for CCLK, and sysclk we used the 4MHz clock. So then we found that $360 = \frac{2M}{1} * (4MHz)$ which got us $M = 45$ Now to find our K value, we did this: $(2)(45)\frac{4MHz}{K} = 10MHz$ after solving for K, we got $K = 36$.

Now doing the same to find an 8MHz wave, we did this $360 = \frac{2M}{1} * (4MHz)$ which got us $M = 45$. Now to find K we did this: $(2)(45)\frac{4MHz}{K} = 8MHz$ after solving for K, we got $K = 36$.

2.2 Schematic for Software

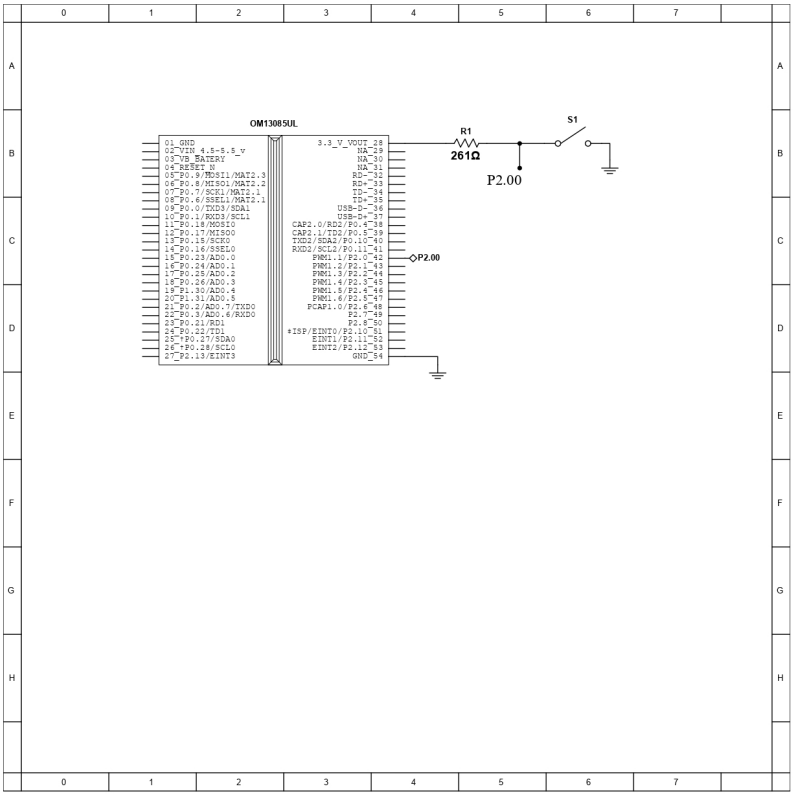


Figure 1: Software Schematic.

2.3 Oscilloscope snapshots for Software

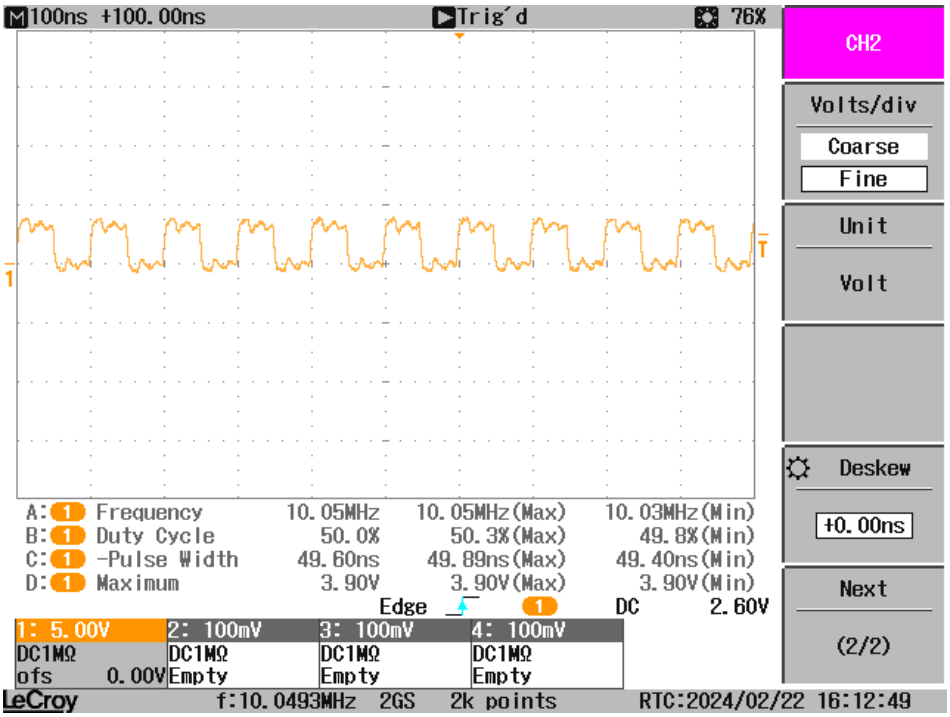


Figure 2: Here we have our 10 MHz Square wave.

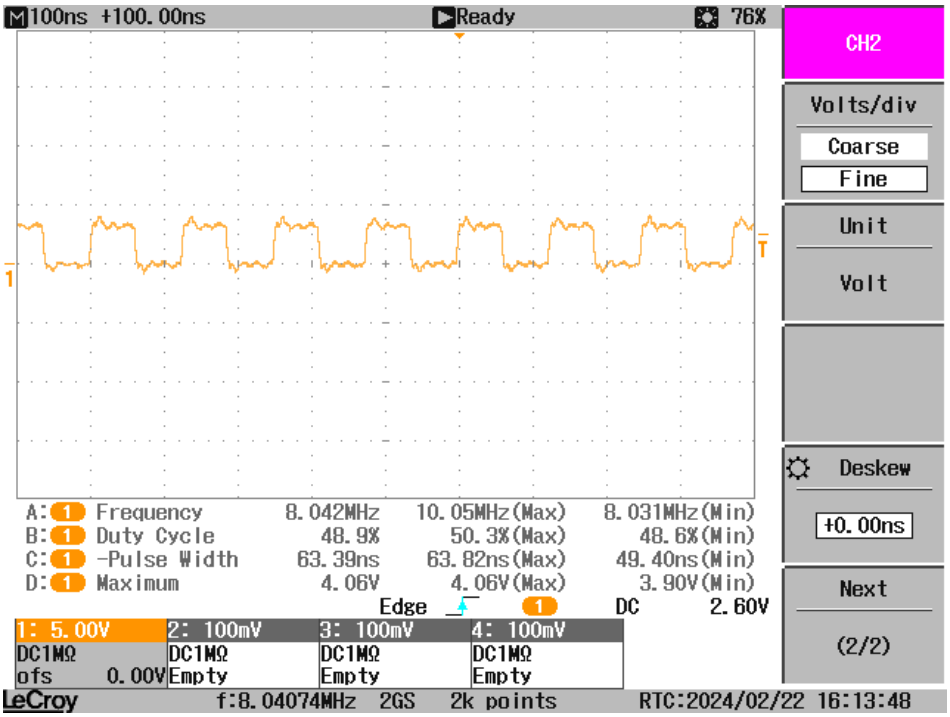


Figure 3: Here we have our 8 MHz Square wave.

3 Hardware

3.1 Hardware Schematic

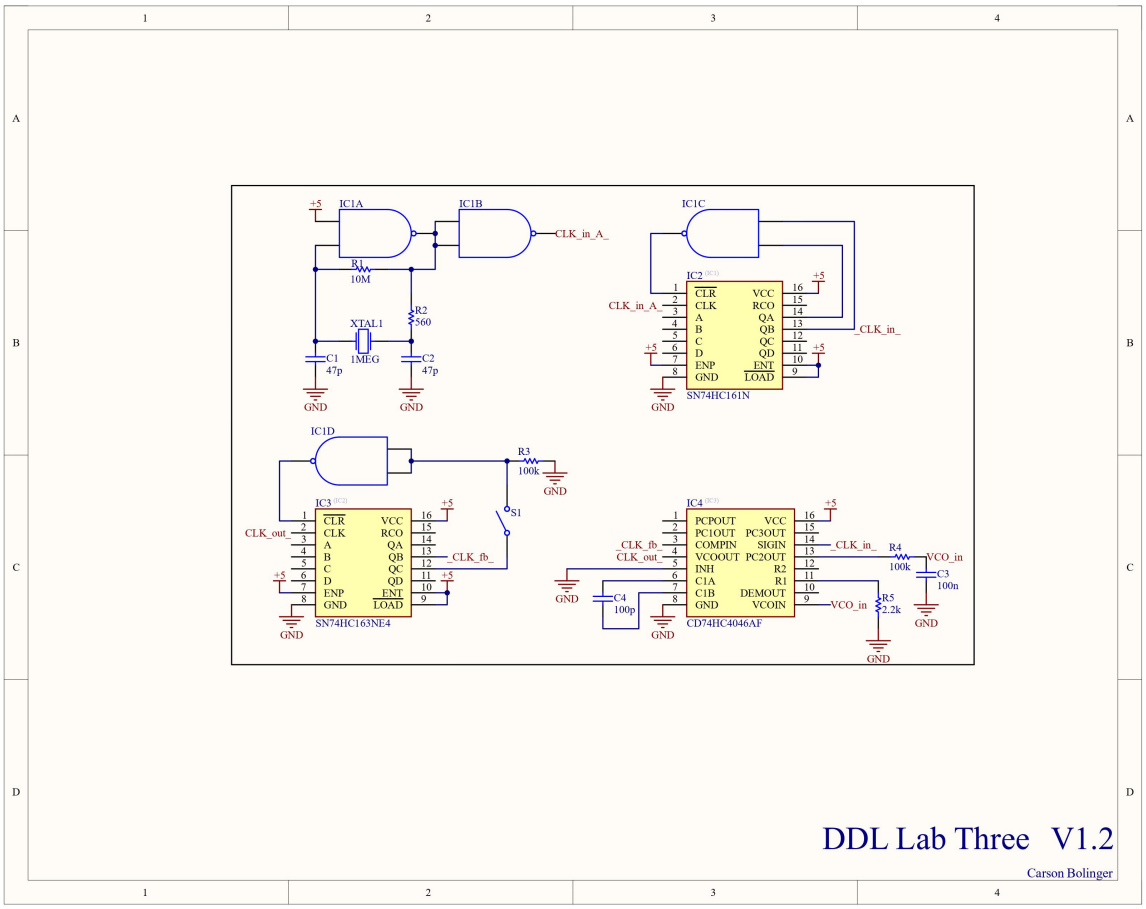


Figure 4: Hardware Schematic.

3.2 Calculations

This schematic was designed by finding the recommended load capacitance of 30pF from the data sheet for the crystal oscillator. Then the equation below was used to calculate the capacitance needed while assuming that $C_1 = C_2$ and $C_{stray} = 5pF$.

$$C_L = \frac{(C_1 + C_i)\, C_2}{C_1 + C_i + C_2} + C_{stray}$$

From the crystal oscillator, we get a 6MHz square wave that needs to be adapted to a 10MHz and 8MHz signal. This is done by calculating the multiplication factor in fraction form for both output frequencies respectively as $\frac{5}{3}$ and $\frac{4}{3}$ respectively. To achieve this we first needed a clock divider that would divide our frequency by the denominator, then we needed to use the PLL chip to multiply our signal by the numerator.

This was achieved by using counter chips and reset logic to create the dividers. The multiplication was done by placing the numerator divider in the feedback look of the PLL such that it will cause a multiplication of the output.

The low pass filter of the PLL was designed to have a cutoff frequency of 15Hz using the formula;

$$R_4 = \frac{1}{2 * pi * C_3 * f_c}$$

Next, for the values of R_5 and C_4 we looked through the data sheet for the PLL and used the provided tables to determine the necessary values.

3.3 Hardware Oscilloscope Snapshots

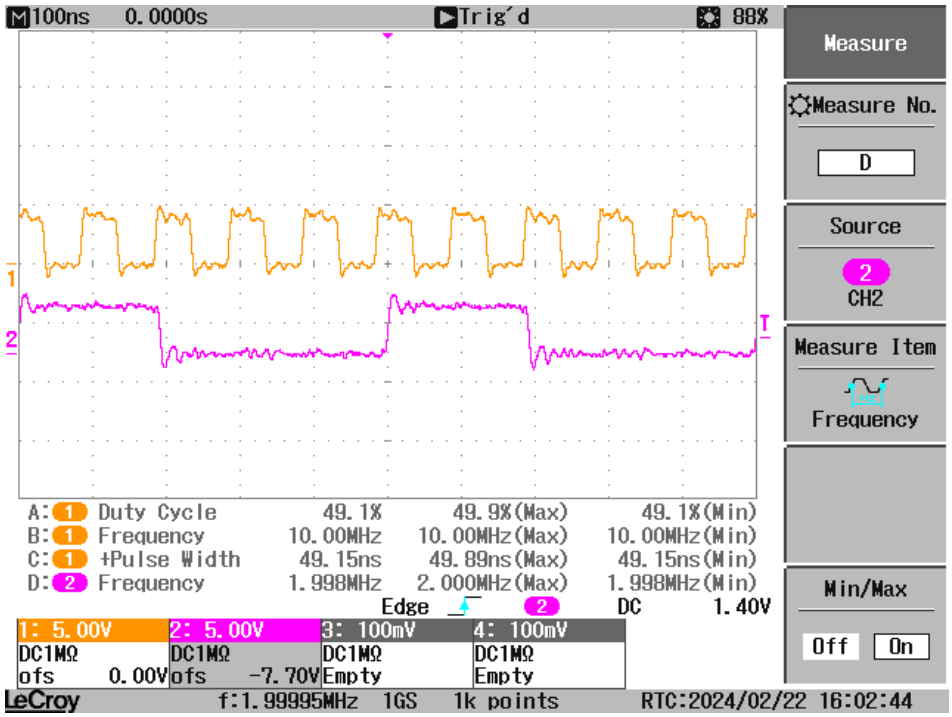


Figure 5: This is our 10 MHz wave with Hardware.

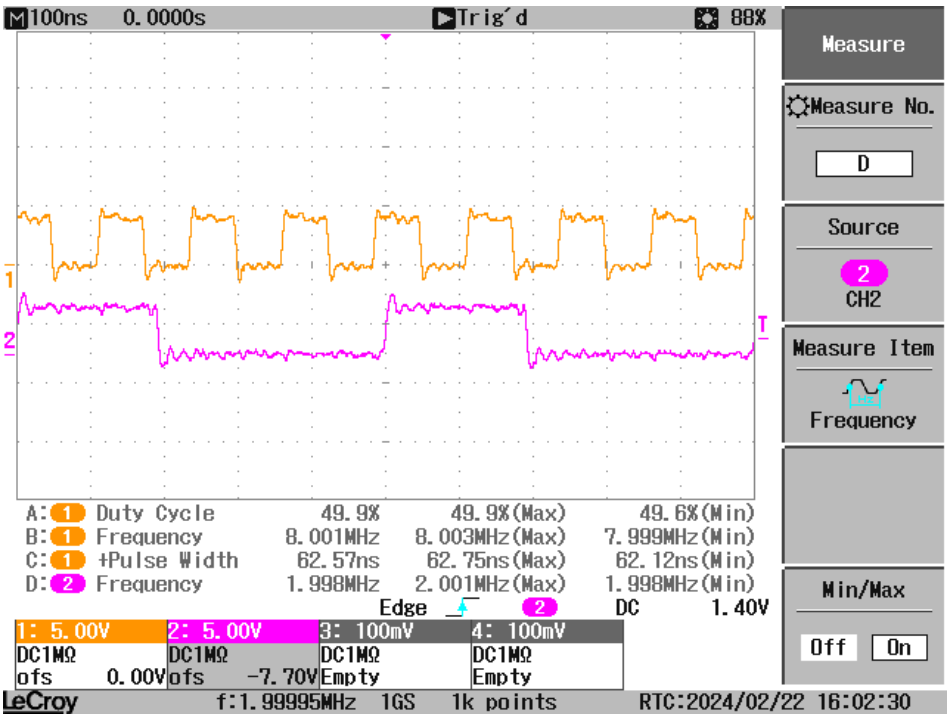
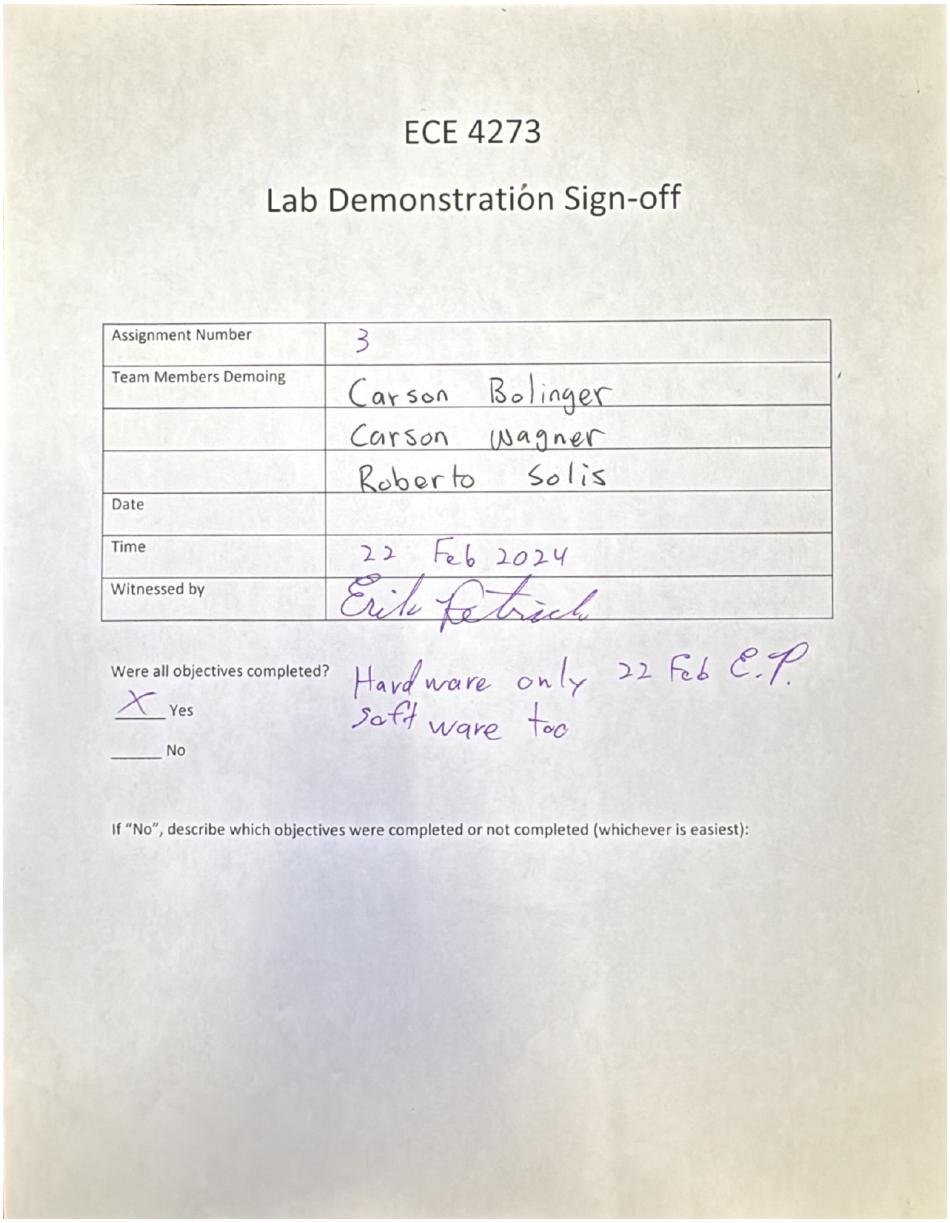


Figure 6: This is our 8 MHz wave with Hardware.

4 Lab Demonstration Sheet



ECE 4273

Lab Demonstration Sign-off

Assignment Number	3
Team Members Demoing	Carson Bolinger
	Carson Wagner
	Roberto Solis
Date	
Time	22 Feb 2024
Witnessed by	Eric Fetrick

Were all objectives completed?

X Yes

____ No

Hardware only 22 Feb E.P.
Software too

If "No", describe which objectives were completed or not completed (whichever is easiest):

Figure 7: Lab Demonstration Sign-off.

Contributions

Carson Bolinger (Electrical Engineering)

Designed the hardware clock including the PLL and the clock dividers before building and troubleshooting them to complete the hardware component of the lab. After completing the hardware component, I worked to correct critical errors in the software.

Roberto Solis (Electrical Engineering)

Contributed to developing the software responsible for generating the 10MHz and 8MHz square waves. Also calculated values to get the 8MHz square wave.

Carson Wagner (Computer Engineering)

Read through the user manual and calculated M, N, and K for the PLL0, and worked on the software. Calculated values to get the 10MHz square wave.