Assignment 3: Lab Report

Digital Design Lab

Sarah Brown

I. Introduction

Clock signals are very important to digital circuits to synchronize outputs. Since these signals are so important, it is also important to know how to generate clock signals and things related to them. This assignment focuses on generating a square wave with a frequency of 11 MHz and a 50% duty cycle. This is implemented with hardware and software techniques. The hardware technique uses an oscillation circuit centered on a 4 MHz crystal and the software technique uses the LPC1769.

II. Hardware Solution

The hardware design uses several elements. The first element establishes an initial oscillation from the 4 MHz crystal. This element is based on this design:

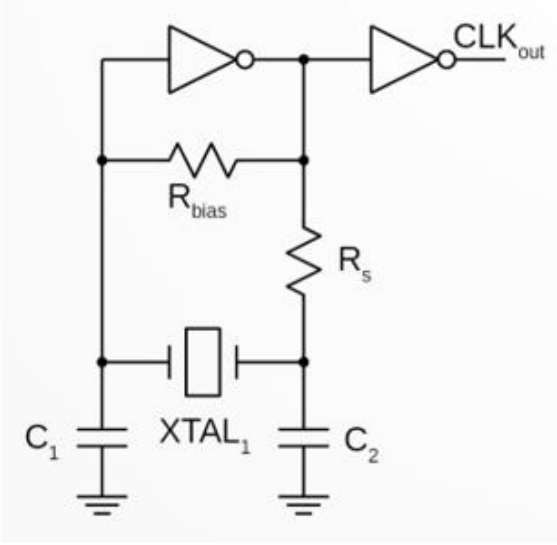

Figure 1: An oscillator circuit with a crystal.

This circuit has 4 configured components: Rs, Rbias, C1, and C2. The equation for C1 and C2 is as follows:

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

However, the smallest capacitor values that I had were 20pF. So instead I calculated CL to ensure that I stayed within the specifications for CL. The chip that is used in this lab has a max of 20pF for CL. Ci is given in the inverter datasheet to be 10pF. Cstray is assumed to be 5pf.

$$C_L = \frac{(20 + 10)C_2}{20 + 10 + 20} + 5 = 12pF$$

12pF is within the CL range so C1=C2=20pF.

Rs is calculated to match the reactance of C2.

$$R_s = \frac{1}{2\pi f C_2}$$

$$R_s = \frac{1}{2\pi(4 * 10^6)(20 * 10^{-12})} = 1989.4 \cong 2k\ \Omega$$

So Rs is rounded to 2k Ω. Rbias is then selected to be 1M Ω. After assembling this, the circuit generated the following output:

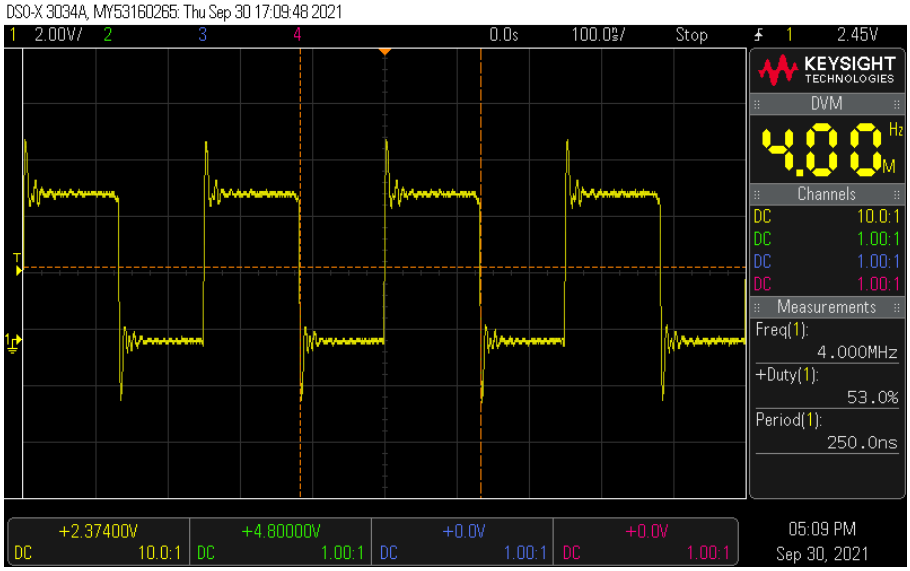

Figure 2: Oscilloscope output of 4 MHz crystal.

The final output frequency goal is 11 MHz so this needs to be divided by 4 and then multiplied by 11. This is done with a divider circuit with a synchronous 4-bit counter and then will a PLL circuit. Since it needs to be divided by 4 to achieve 1 MHz, this can be done by using Qb on the 4-bit counter.

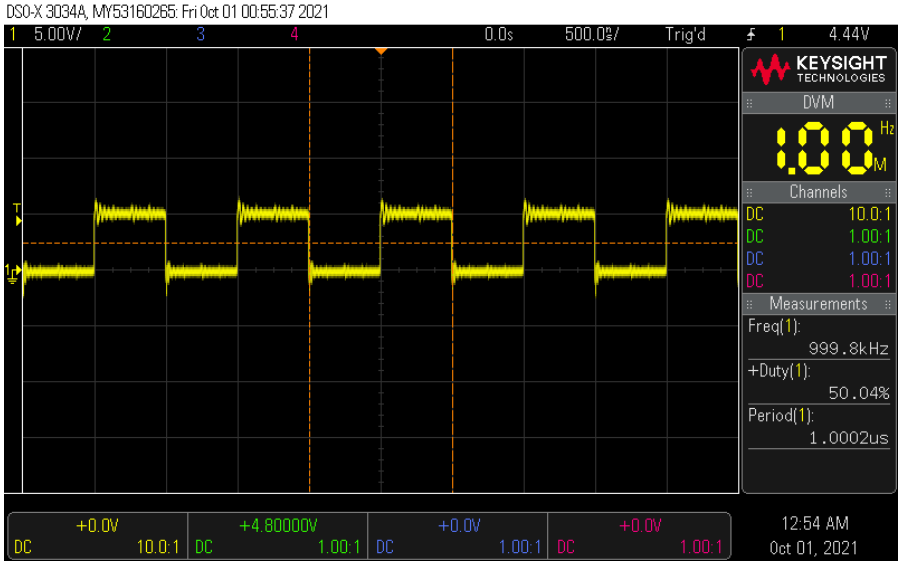

Figure 3: Oscilloscope output of signal after being divided by 4 resulting in 1 MHz.

Afterwards, this needs to be multiplied by 11 or 9 to achieve the 11 MHz or the 9 MHz. This is done within a PLL, this has three main parts as seen here:

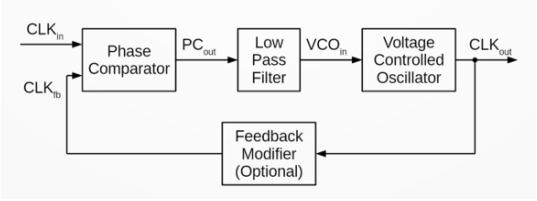

Figure 4: PLL.

R1 and C are selected from this chart with a goal of 10 MHz. This results in a value of R1 = 2.2k and C = 100 pF.

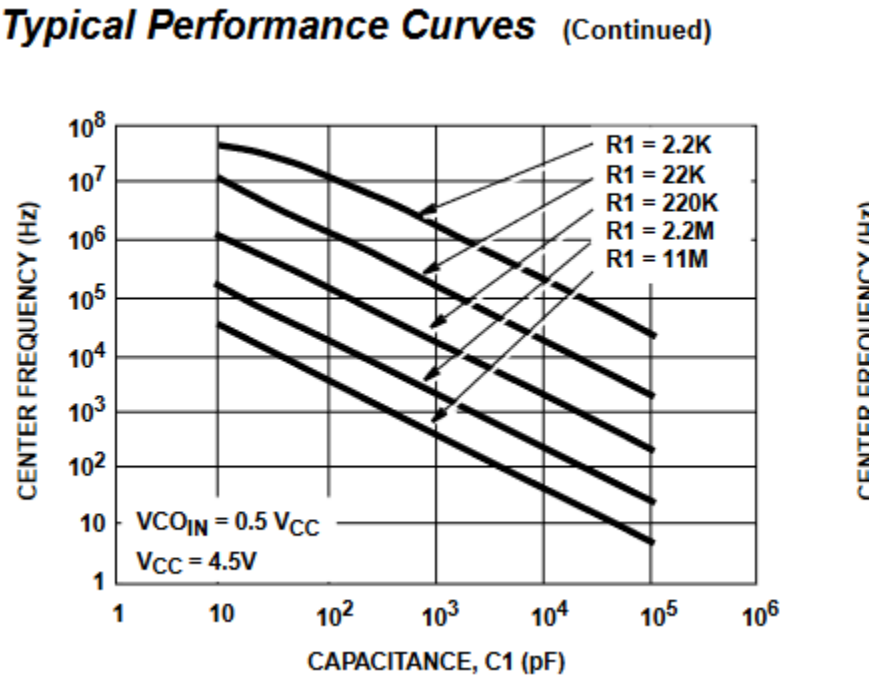

Figure 5: Typical performance curve for PLL.

A standard low pass of a voltage divider of 100 kilo-ohms and 10 kilo-ohms to a 1 uF capacitor is used to set the nominal voltage-controlled oscillator frequency. This sets how quickly it stabilizes.

The feedback is set with another 4-bit counter. However, this is used to divided by starting at 5 for 11MHz and 7 for 9MHz and triggering the load function when the ripple is set high. This allows for easier implementation of the switch. Below are the scope outputs for when the switch is released and pressed. The outputs had some noise due to the nature of building such a high frequency circuit on a breadboard. The final schematic is shown in the appendix.

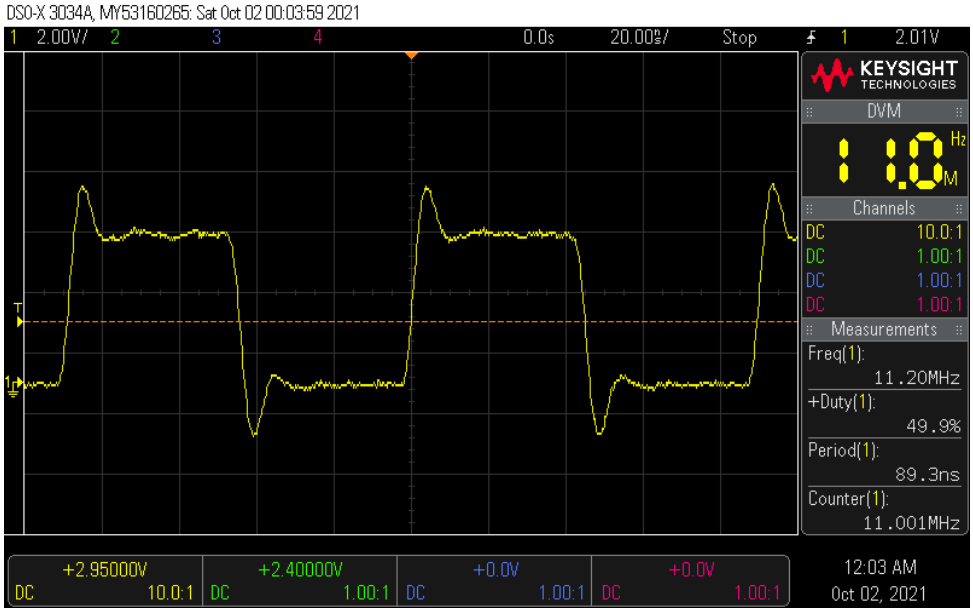

Figure 6: Scope output for 11 MHz, button not pressed.

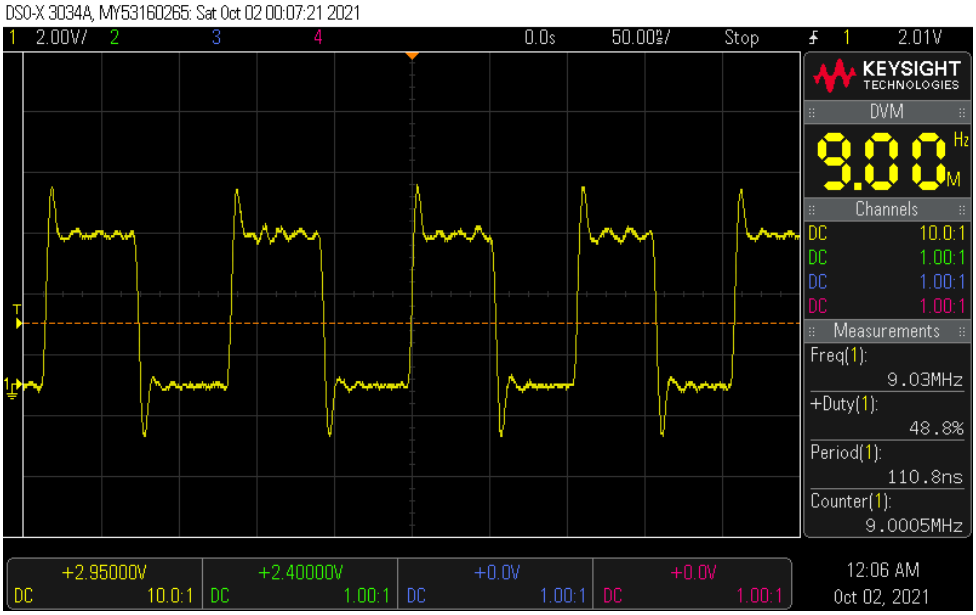

Figure 7: Scope output for 9 MHz, button pressed.

To verify that this output fits within the specification (1% error on frequency and 5% error on duty cycle), the following calculations were done:

$$\left|\frac{Actual - Expected}{Expected}\right| \cdot 100\%$$

11 MHz duty cycle:

$$\left|\frac{49.9-50}{50}\right| \cdot 100\% = 0.2\%$$

11 MHz frequency:

$$\left|\frac{11.2-11}{11}\right| \cdot 100\% = 1.82\%$$

9 MHz duty cycle:

$$\left|\frac{48.8-50}{50}\right| \cdot 100\% = 2.4\%$$

9 MHz frequency:

$$\left|\frac{9.03-9}{9}\right| \cdot 100\% = 0.333\%$$

Therefore, all of these values are within the error range.

III. Software Solution

By using the RC oscillator between into the LPC1769, it is possible to start with a square signal of 4 MHz. There is also an internal PLL feature that can be used to divide and multiply the signal. The 4 MHz signal is used as a base and the final signal is output on pin 1.27 on PAD10 so this must be appropriately configured as an output.

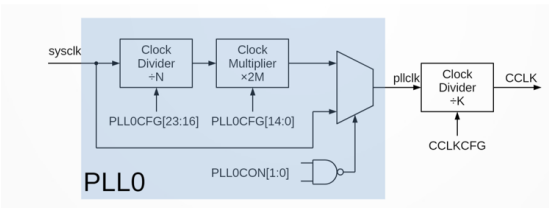

Figure 8: Details about the internal PLL in the LPC1769.

Examples in the LPC1769 documentation can be followed to calculate appropriate values to plug into the code and the registers.

M can be found as follows:

$$M = \frac{(F_{cco})(N)}{2F_{in}}$$

$$M = \frac{(288 * 10^6)(1)}{2(4 * 10^6)}$$

$$M = 36$$

K can be found as follows:

$$K = \frac{(F_{cco})}{(F_{out})}$$

$$K = \frac{(288 * 10^6)}{(11 * 10^6)}$$

$$K = 26.182 \cong 26$$

However, since the code is 0 indexed each parameter should have an one subtracted from it leaving the values at: M = 35, K = 25, and N = 0. The steps laid out in the LPC1769 were followed with these parameters with a satisfactory result. A button was not used in the software portion. The code is shown in the appendix.

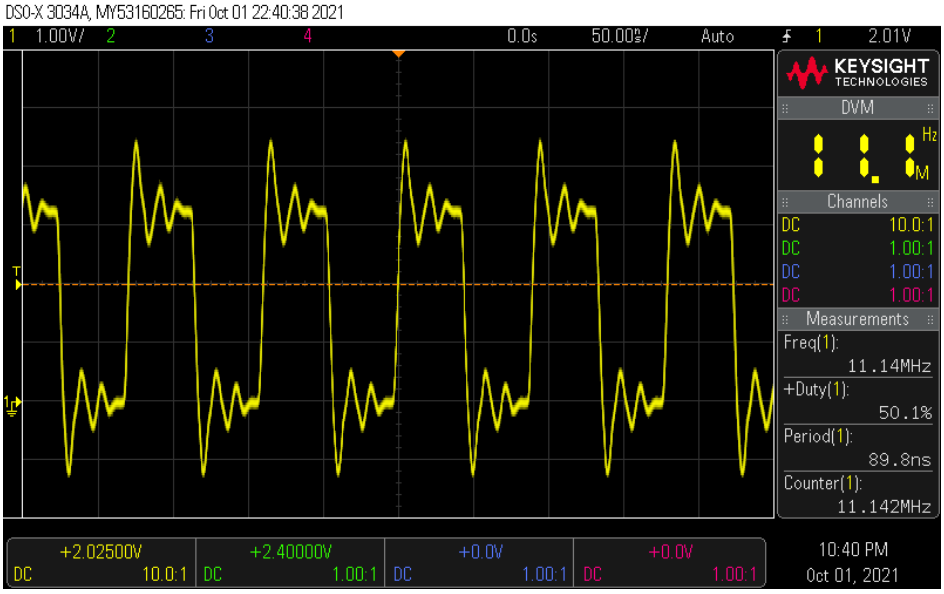

Figure 9: Scope output for 11 MHz (no button)

To verify that this output fits within the specification (1% error on frequency and 5% error on duty cycle), the following calculations were done:

$$\left|\frac{Actual - Expected}{Expected}\right| \cdot 100\%$$

11 MHz duty cycle:

$$\left|\frac{50.1-50}{50}\right| \cdot 100\% = 0.2\%$$

11 MHz frequency:

$$\left|\frac{11.14-11}{11}\right| \cdot 100\% = 1.27\%$$

Here, the duty cycle is within range, but the frequency is slightly outside of the 1% error range. This is due to the calculation for K not providing a clean result and rounding occurring. This could be fixed by using one of the different internal oscillators instead of the RC one. I would have tried this other approach by did not notice that it was out of bounds due to accidentally truncating the frequency to 11.1 MHz instead of 11.14 MHz.

IV. Appendix

A. *Schematic*

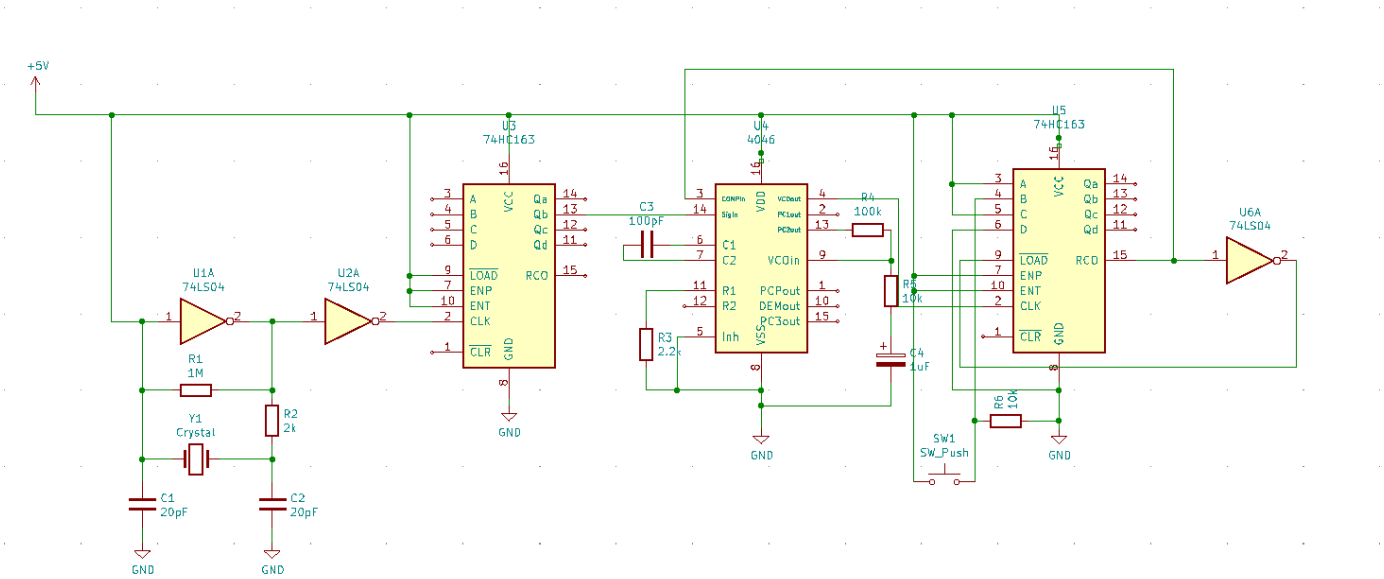

Figure 1: Schematic of hardware design

B. *Code*

```
// define clock/pll registers
#define CCLKCFG (*(volatile unsigned int *) 0x400FC104)
#define CLKSRCSEL (*(volatile unsigned int *) 0x400FC10C)
#define PLL0CFG (*(volatile unsigned int *) 0x400FC084)
#define PLL0CON (*(volatile unsigned int *) 0x400FC080)
#define PLL0FEED (*(volatile unsigned int *) 0x400FC08C)
#define PLL0STAT (*(volatile unsigned int *) 0x400FC088)
#define CLKOUTCFG (*(volatile unsigned int *) 0x400FC1C8)
// define pin registers
#define PINSEL3 (*(volatile unsigned int *) 0x4002C00C)

// macros
#define PLL_FEED() PLL0FEED = 0xAA; PLL0FEED = 0x55

#define K 25
#define M 144

void setup() {
        PINSEL3 |= (01<<22); // configure clkout as output
}

int main(void) {
        // step 1 - disconnect PLL0 with one feed sequence
        PLL0CON &= ~(1 << 1);
        PLL_FEED();

        // step 2 - disable PLL0 with one feed sequence
        PLL0CON &= ~(1);
        PLL_FEED();
```

```
        // step 3 - change cpu clock divider to speed up operation without PLL0, if desired
        // step 4 - write to the clock source selection control register to change the clock source if needed
        CLKSRCSEL = 0;

        // step 5 - write to PLL0CFG and make it effective with one feed sequence (can only be updated when PLL0 disabled)
        PLL0CFG = 35; // bit 15 and bits 23:16 will be 0 by nature of decimal 35. Writing n at same time as m, but it is n=1-1=0
        PLL_FEED();

        // step 6 - enable PLL0 with one feed sequence
        PLL0CON |= (1 << 0);
        PLL_FEED();

        // step 8 - wait for PLL0 to achieve lock
        // step 8 is swapped with 7 per rec of powerpoint slides
        while (!(PLL0STAT >> 26)) {
        }

        // step 7 - change CPU clock divider setting for operation with PLL0
        CCLKCFG = K;

        // step 9 - connect PLL0 with one feed sequence
        PLL0CON |= (1 << 1);
        PLL_FEED();

        CLKOUTCFG = 0b0100000000;
        setup(); // setup pins


    while(1) {
    }
}
```

Appendix