Lab 3 - Si5351 Clock Generator

Si5351 Clock Generator

This lab involves testing the Si5351 (which has an i2c interface).

1. Introduction

1.1 I2C Interface to Si5351

The i2c interface is simple, ensure that the SDA and SCL lines of the Si5351 are connected to an appropriate port on the Beaglebone. In the example below, I used /dev/i2c-2.

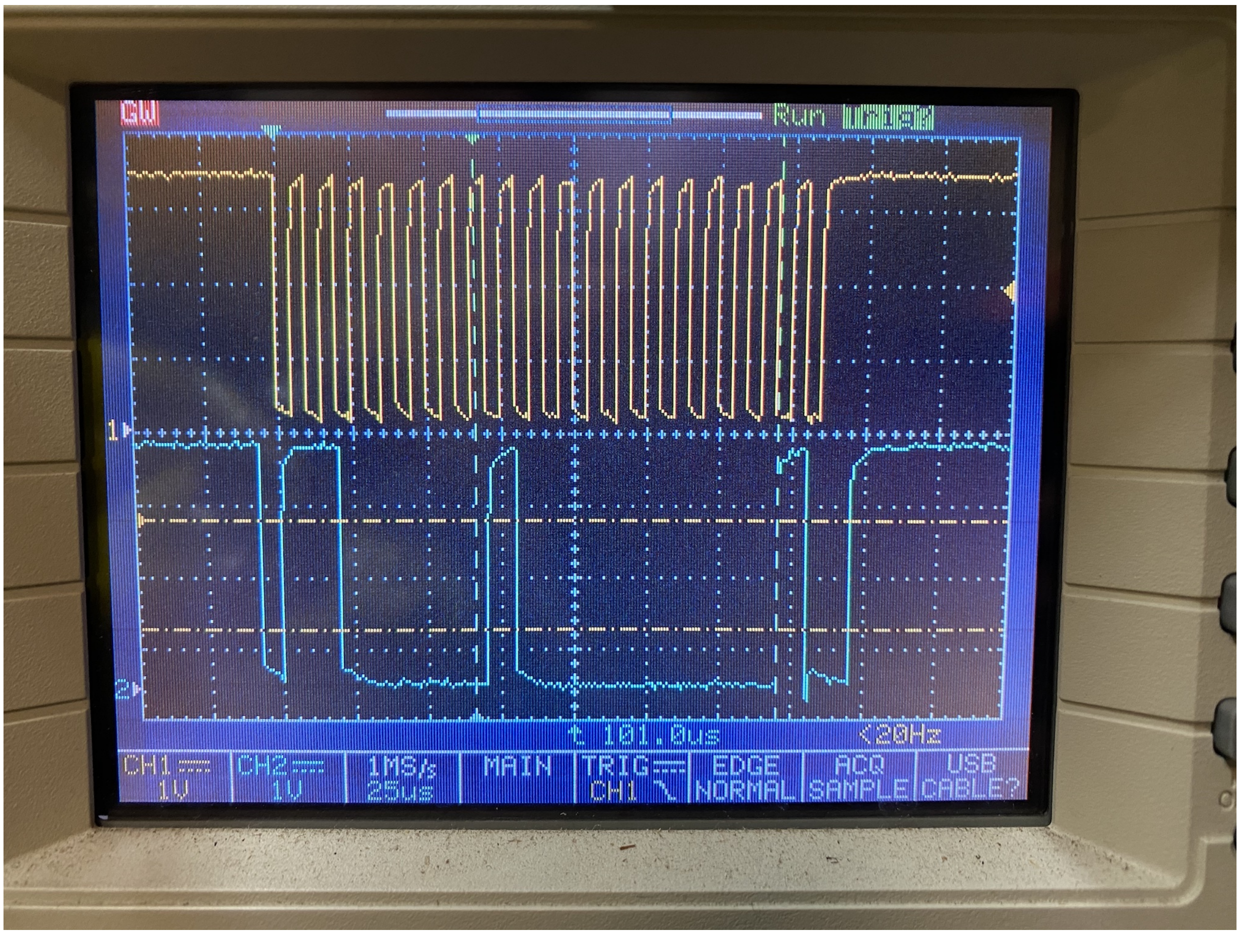
Then make sure all the software required is installed on the BBG.

```
sudo apt update
sudo apt-get install libi2c-dev
```

The commands below show all the i2c devices and that the Si5351 at address 0x60 is responding to the probe on I2C2 (i2c chip 2):

```
debian@beaglebone:~$ i2cdetect -l
i2c-1   i2c             OMAP I2C adapter                        I2C adapter
i2c-2   i2c             OMAP I2C adapter                        I2C adapter
i2c-0   i2c             OMAP I2C adapter                        I2C adapter
debian@beaglebone:~$ i2cdetect -y -r 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Here is the i2c transaction for `i2cdetect -y -r 2 0x60 0x60`.



(http://localhost:4000/assets/images/2021/03/si5351-i2c.jpg)

1.2. Linux userspace driver

There are several ways that this interface can be made. We are going to create a userspace driver, which is the most straightforward. The following program, derived from the Linux Kernel **userspace driver documentation (https://www.kernel.org/doc/html/latest/i2c/dev-interface.html)** reads and prints register 0 of device /dev/i2c-2, address 0x60, i.e. register 0 of the Si5351. Note that the reason we need to specify address 0x60 is that there could be multiple devices connected to a single i2c chip.

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/i2c-dev.h>
#include <i2c/smbus.h>

#define I2C_FNAME       "/dev/i2c-2"
#define SI5351_ADDR     0x60

int     i2c_file;

void i2c_init()
{
        i2c_file = open(I2C_FNAME, O_RDWR);
        if (i2c_file < 0)
                exit(1);
}

int i2c_read(unsigned char reg)
{
        if (ioctl(i2c_file, I2C_SLAVE, SI5351_ADDR) < 0)
                exit(1);

        int res;

        /* Using SMBus commands */
        res = i2c_smbus_read_byte_data(i2c_file, reg);
        if (res < 0)
                exit(1);
        else
                printf("r dev(0x%x) reg(0x%x)=0x%x\n", SI5351_ADDR, reg, res
);
        return res;
}

int
main()
{
        i2c_init();
        i2c_read(0);
}
```

It can be compiled and executed as follows:

```
debian@beaglebone:~$ gcc -o si5351 si5351.c -li2c
debian@beaglebone:~$ ./si5351
r dev(0x60) reg(0x0)=0x11
```

1.3. Programming the Clock Generator

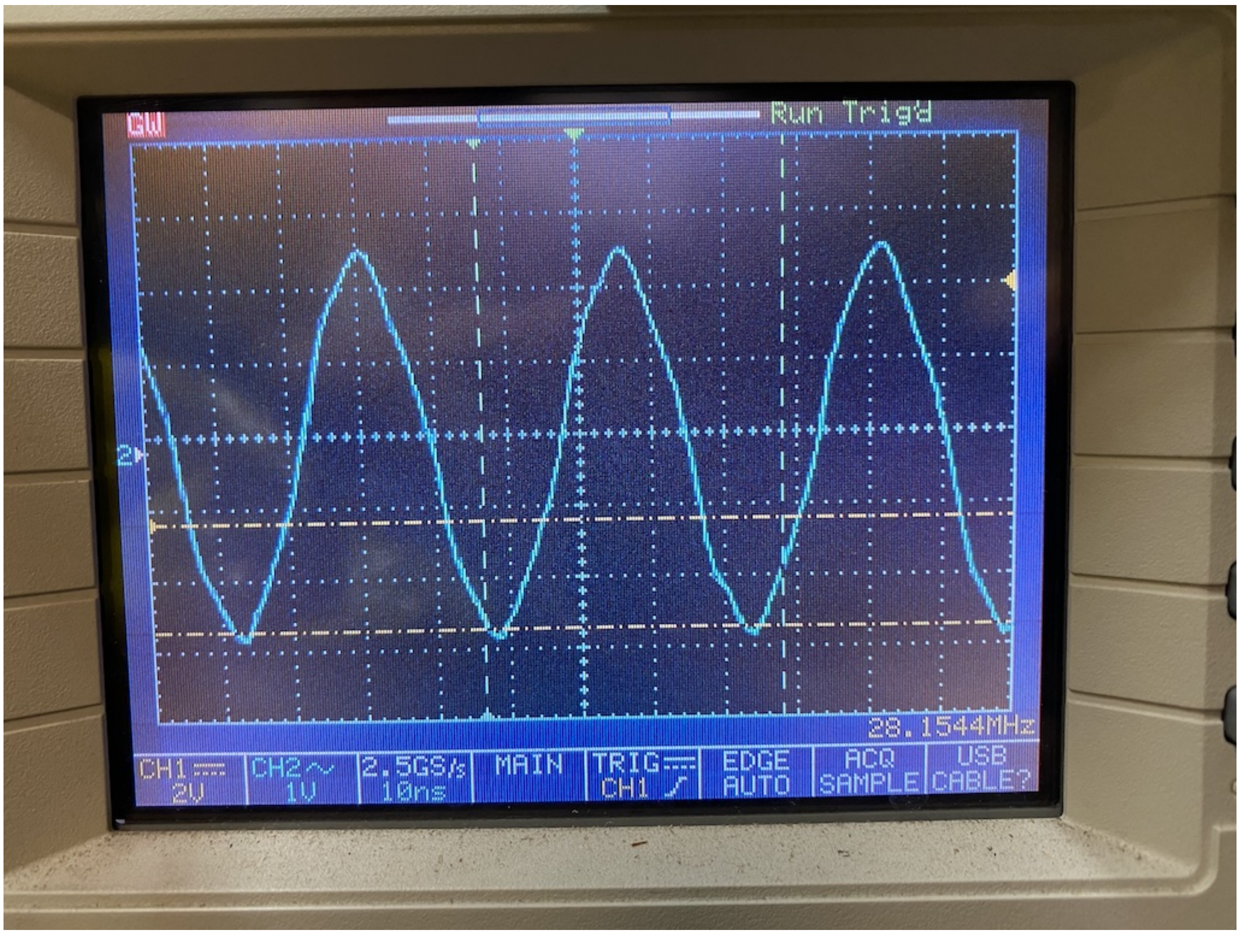
The **data sheet (https://www.silabs.com/documents/public/data-sheets/Si5351-B.pdf)** for the Si5351 refers to the **Clock builder (https://www.silabs.com/developers/clockbuilder-pro-software)** software. This is only available for Windows but in embedded systems it is a fact of life that if you use operating system X, there will be a piece of software that is only supported on operating system Y that you need. Here is a header file that I generated for CLK0 and CLK1 outputs of 28.1544 MHz.

```
/*
 * Si5351A Rev B Configuration Register Export Header File
 *
 * This file represents a series of Silicon Labs Si5351A Rev B
 * register writes that can be performed to load a single configuration
 * on a device. It was created by a Silicon Labs ClockBuilder Pro
 * export tool.
 *
 * Part:                                               Si5351A Rev B
 * Design ID:
 * Includes Pre/Post Download Control Register Writes: Yes
 * Created By:                                         ClockBuilder Pro v3.1
[2021-01-18]
 * Timestamp:                                          2021-03-12 15:00:15 G
MT-08:00
 *
 * A complete design report corresponding to this export is included at the
 end
 * of this header file.
 *
 */

#ifndef SI5351A_REVB_REG_CONFIG_HEADER
#define SI5351A_REVB_REG_CONFIG_HEADER

#define SI5351A_REVB_REG_CONFIG_NUM_REGS                                  52

typedef struct
{
        unsigned int address; /* 16-bit register address */
        unsigned char value; /* 8-bit register data */

} si5351a_revb_register_t;

si5351a_revb_register_t const si5351a_revb_registers[SI5351A_REVB_REG_CONFIG
_NUM_REGS] =
{
        { 0x0002, 0x53 },
        { 0x0003, 0x00 },
        { 0x0004, 0x20 },
        { 0x0007, 0x00 },
        { 0x000F, 0x00 },
        { 0x0010, 0x0F },
        { 0x0011, 0x0F },
        { 0x0012, 0x8C },
        { 0x0013, 0x8C },
        { 0x0014, 0x8C },
        { 0x0015, 0x8C },
        { 0x0016, 0x8C },
        { 0x0017, 0x8C },
        { 0x001A, 0xAF },
        { 0x001B, 0xC8 },
        { 0x001C, 0x00 },
        { 0x001D, 0x0E },
        { 0x001E, 0x8D },
        { 0x001F, 0x00 },
        { 0x0020, 0x85 },
        { 0x0021, 0x58 },
        { 0x002A, 0x00 },
        { 0x002B, 0x04 },
        { 0x002C, 0x00 },
        { 0x002D, 0x0D },
        { 0x002E, 0xE0 },
        { 0x002F, 0x00 },
        { 0x0030, 0x00 },
        { 0x0031, 0x00 },
        { 0x0032, 0x00 },
        { 0x0033, 0x04 },
        { 0x0034, 0x00 },
        { 0x0035, 0x0D },
        { 0x0036, 0xE0 },
        { 0x0037, 0x00 },
        { 0x0038, 0x00 },
        { 0x0039, 0x00 },
        { 0x005A, 0x00 },
        { 0x005B, 0x00 },
        { 0x0095, 0x00 },
        { 0x0096, 0x00 },
        { 0x0097, 0x00 },
        { 0x0098, 0x00 },
        { 0x0099, 0x00 },
        { 0x009A, 0x00 },
        { 0x009B, 0x00 },
        { 0x00A2, 0x00 },
        { 0x00A3, 0x00 },
        { 0x00A4, 0x00 },
        { 0x00A5, 0x00 },
        { 0x00A6, 0x00 },
        { 0x00B7, 0x92 },

};

/*
 * Design Report
 *
 * Overview
 * ========
 * Part:               Si5351A
 * Created By:         ClockBuilder Pro v3.1 [2021-01-18]
 * Timestamp:          2021-03-12 15:00:15 GMT-08:00
 *
 * Design Notes
 * ============
 * Si5351A 10-pin MSOP
 *
 * Design Rule Check
 * =================
 * Errors:
 * - No errors
 *
 * Warnings:
 * - No warnings
 *
 * Design
 * ======
 * Inputs:
 *     IN0: 27 MHz
 *
 * Outputs:
 *    OUT0: 28.1544 MHz
 *          Enabled LVCMOS 8 mA
 *          Offset 0.000 s
 *    OUT1: 28.1544 MHz
 *          Enabled LVCMOS 8 mA
 *          Offset 0.000 s
 *    OUT2: Unused
 *
 * Frequency Plan
 * ==============
 * PLL_A:
 *    Enabled Features = None
 *    Fvco             = 893.9022 MHz
 *    M                = 33.1074888888888888... [ 33 + 4837/45000 ]
 *    Input0:
 *       Source           = Crystal
 *       Source Frequency = 27 MHz
 *       Fpfd             = 27 MHz
 *       Load Capacitance = Load_08pF
 *    Output0:
 *       Features       = None
 *       Disabled State = StopLow
 *       R              = 1  (2^0)
 *       Fout           = 28.1544 MHz
 *       N              = 31.75
 *    Output1:
 *       Features       = None
 *       Disabled State = StopLow
 *       R              = 1  (2^0)
 *       Fout           = 28.1544 MHz
 *       N              = 31.75
 *
 * Settings
 * ========
 *
 * Location      Setting Name    Decimal Value      Hex Value
 * ------------  --------------  -----------------  -----------------
 * 0x0002[3]     XO_LOS_MASK     0                  0x0
 * 0x0002[4]     CLK_LOS_MASK    1                  0x1
 * 0x0002[5]     LOL_A_MASK      0                  0x0
 * 0x0002[6]     LOL_B_MASK      1                  0x1
 * 0x0002[7]     SYS_INIT_MASK   0                  0x0
 * 0x0003[7:0]   CLK_OEB         0                  0x00
 * 0x0004[4]     DIS_RESET_LOLA  0                  0x0
 * 0x0004[5]     DIS_RESET_LOLB  1                  0x1
 * 0x0007[7:4]   I2C_ADDR_CTRL   0                  0x0
 * 0x000F[2]     PLLA_SRC        0                  0x0
 * 0x000F[3]     PLLB_SRC        0                  0x0
 * 0x000F[4]     PLLA_INSELB     0                  0x0
 * 0x000F[5]     PLLB_INSELB     0                  0x0
 * 0x000F[7:6]   CLKIN_DIV       0                  0x0
 * 0x0010[1:0]   CLK0_IDRV       3                  0x3
 * 0x0010[3:2]   CLK0_SRC        3                  0x3
 * 0x0010[4]     CLK0_INV        0                  0x0
 * 0x0010[5]     MS0_SRC         0                  0x0
 * 0x0010[6]     MS0_INT         0                  0x0
 * 0x0010[7]     CLK0_PDN        0                  0x0
 * 0x0011[1:0]   CLK1_IDRV       3                  0x3
 * 0x0011[3:2]   CLK1_SRC        3                  0x3
 * 0x0011[4]     CLK1_INV        0                  0x0
 * 0x0011[5]     MS1_SRC         0                  0x0
 * 0x0011[6]     MS1_INT         0                  0x0
 * 0x0011[7]     CLK1_PDN        0                  0x0
 * 0x0012[1:0]   CLK2_IDRV       0                  0x0
 * 0x0012[3:2]   CLK2_SRC        3                  0x3
 * 0x0012[4]     CLK2_INV        0                  0x0
 * 0x0012[5]     MS2_SRC         0                  0x0
 * 0x0012[6]     MS2_INT         0                  0x0
 * 0x0012[7]     CLK2_PDN        1                  0x1
 * 0x0013[1:0]   CLK3_IDRV       0                  0x0
 * 0x0013[3:2]   CLK3_SRC        3                  0x3
 * 0x0013[4]     CLK3_INV        0                  0x0
 * 0x0013[5]     MS3_SRC         0                  0x0
 * 0x0013[6]     MS3_INT         0                  0x0
 * 0x0013[7]     CLK3_PDN        1                  0x1
 * 0x0014[1:0]   CLK4_IDRV       0                  0x0
 * 0x0014[3:2]   CLK4_SRC        3                  0x3
 * 0x0014[4]     CLK4_INV        0                  0x0
 * 0x0014[5]     MS4_SRC         0                  0x0
 * 0x0014[6]     MS4_INT         0                  0x0
 * 0x0014[7]     CLK4_PDN        1                  0x1
 * 0x0015[1:0]   CLK5_IDRV       0                  0x0
 * 0x0015[3:2]   CLK5_SRC        3                  0x3
 * 0x0015[4]     CLK5_INV        0                  0x0
 * 0x0015[5]     MS5_SRC         0                  0x0
 * 0x0015[6]     MS5_INT         0                  0x0
 * 0x0015[7]     CLK5_PDN        1                  0x1
 * 0x0016[1:0]   CLK6_IDRV       0                  0x0
 * 0x0016[3:2]   CLK6_SRC        3                  0x3
 * 0x0016[4]     CLK6_INV        0                  0x0
 * 0x0016[5]     MS6_SRC         0                  0x0
 * 0x0016[6]     FBA_INT         0                  0x0
 * 0x0016[7]     CLK6_PDN        1                  0x1
 * 0x0017[1:0]   CLK7_IDRV       0                  0x0
 * 0x0017[3:2]   CLK7_SRC        3                  0x3
 * 0x0017[4]     CLK7_INV        0                  0x0
 * 0x0017[5]     MS7_SRC         0                  0x0
 * 0x0017[6]     FBB_INT         0                  0x0
 * 0x0017[7]     CLK7_PDN        1                  0x1
 * 0x001C[17:0]  MSNA_P1         3725               0x00E8D
 * 0x001F[19:0]  MSNA_P2         34136              0x08558
 * 0x001F[23:4]  MSNA_P3         45000              0x0AFC8
 * 0x002C[17:0]  MS0_P1          3552               0x00DE0
 * 0x002F[19:0]  MS0_P2          0                  0x00000
 * 0x002F[23:4]  MS0_P4          4                  0x00004
 * 0x0034[17:0]  MS1_P1          3552               0x00DE0
 * 0x0037[19:0]  MS1_P2          0                  0x00000
 * 0x0037[23:4]  MS1_P4          4                  0x00004
 * 0x005A[7:0]   MS6_P2          0                  0x00
 * 0x005B[7:0]   MS7_P2          0                  0x00
 * 0x0095[14:0]  SSDN_P2         0                  0x0000
 * 0x0095[7]     SSC_EN          0                  0x0
 * 0x0097[14:0]  SSDN_P3         0                  0x0000
 * 0x0097[7]     SSC_MODE        0                  0x0
 * 0x0099[11:0]  SSDN_P1         0                  0x000
 * 0x009A[15:4]  SSUDP           0                  0x000
 * 0x00A2[21:0]  VCXO_PARAM      0                  0x000000
 * 0x00A5[7:0]   CLK0_PHOFF      0                  0x00
 * 0x00A6[7:0]   CLK1_PHOFF      0                  0x00
 * 0x00B7[7:6]   XTAL_CL         2                  0x2
 *
 *
 */

#endif
```

Our approach will be to use the header file above, **Application Note AN619 Manually Generating an Si5351 Register Map for 10-MSOP and 20-QFN Devices** (https://www.silabs.com/documents/public/application-notes/AN619.pdf) and a **Silicon Labs Knowledge Base entry** (https://www.silabs.com/community/timing/knowledge-base.entry.html/2011/02/18/programming_the_si53-IrM8) which says

```
ClockBuilder Desktop allows a user to generate RAM configuration files to program the Si5351
 with custom frequency plans via I2C.  Once the register map has been generated, use the proc
edure below to program the device.

1. Disable all outputs.
       reg3 = 0xFF
2. Write reg187 = 0xC0
3. Power down all output drivers
       reg 16 = 0x80*
       reg 17 = 0x80*
       reg 18 = 0x80*
       reg 19 = 0x80*
       reg 20 = 0x80*
       reg 21 = 0x80*
       reg 22 = 0x80*
       reg 23 = 0x80*
       * If using the Si5351C with no crystal present on XA/XB, set reg16-23 = 0x84.
4. Set interrupt maks register (see Register 2 description in datasheet)
5. If needed, set crystal load capacitance, XTAL_CL in reg183[7:6].  See datasheet for regist
er description.
6. Write registers 15-92 and 149-170 using the contents of register map generated by ClockBui
lder Desktop.
7. Apply PLL A and PLL B soft reset.
       reg177 = 0xAC
8. Enable outputs with OEB control in register 3.
```

If these steps are followed, the output as specified in the include file should appear. It doesn’t look much like a square wave on my 150 MHz bandwidth oscilloscope. Why?



[(http://localhost:4000/assets/images/2021/03/si5351-osc.jpg)](http://localhost:4000/assets/images/2021/03/si5351-osc.jpg)

Finally, we wish to have the inphase (I) clock (CLK0) lagging the quadrature clock (CLK1) by 90 degrees (or 1/4 cycle). We can do this by setting the CLK1_PHOFF register to the appropriate value.

2. Laboratory Experiment

Assuming that you already have the elec3607-labquestions directory on your Beaglebone from Lab 1, update the files using:

`debian@beaglebone:~$ cd elec3607-labquestions; git pull`

`...`

`debian@beaglebone:~$ cd`

Part 1 - I2C Interface (30%)

Connect up your BBG to the Si5351. Verify that you can obtain the following output.

```
debian@beaglebone:~$ i2cdetect -y -r 2
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Part 2 - I2C Transaction (30%)

Use the userspace i2c driver (`si5351.c`) shown above to read register 0. Execute the program and capture the activity of the SCL and SDA pins on an oscilloscope. Make a printout of the oscilloscope display and annotate all parts of the i2c transaction (start, data, r/w, ack, etc). What is the period of the entire transaction?

Part 3 - Si5351 Configuration (40%)

Modify the userspace driver for the Si5351 to generate a 7 MHz square wave output on CLK0 and CLK1, with CLK1 being delayed by 90 degrees from CLK0.