Final Project

Introduction:
The overall system uses a main controller. The controller takes in a command from the UART and interprets the input. Depending on the command the SPI will send data from the block memory or the switches. The AD2 then takes the data from the DA2. The I2c module then takes the voltage and outputs 12 data bits. This then goes into our encoder. The encoder splits the 12 bits into two 6 bit packets if a packet is ‘111111’ then it will concatenate ‘11’ if the packet is not all 1’s then we concatenate a ‘01’to the front. Th UART then sends the data back to the PC. Our python code is then able to interpret the output signal and plot a voltage depending on if we choose the block memory or the switches.

Methods

UART:

takes in data at a baud rate of 115200. The speed of the UART is required. If the UART is required. If the UART is slower than the I2C then we may lose data. The UART receives a command and transmits output back to the PC.

SPI:

The SPI has the option to switch between block memory or the switches on the board. The SPI sends data at 10Mhz. We chose this speed due to the previous design specifications.

I2C:

The I2C takes data that is input by the DA2. The clock speed on this is 100kHz. We chose this speed because it is slower than the existing UART speed.

Encoder:

The encoder takes 6 bits of data output by the I2C and turns it into a byte.
If the bits are ‘111111’then it will concatenate ‘11’ to the front, if anything else ‘01’ concatenates to the front. This relies on the I2done and UART done signals.

I2C : 100kHZ

SPI : 10 MHZ

UART : 115200 baud rate

FIFO Depth

Write frequency = 100kHz

Reading frequency = 10 MHZ

Burst length = 3000

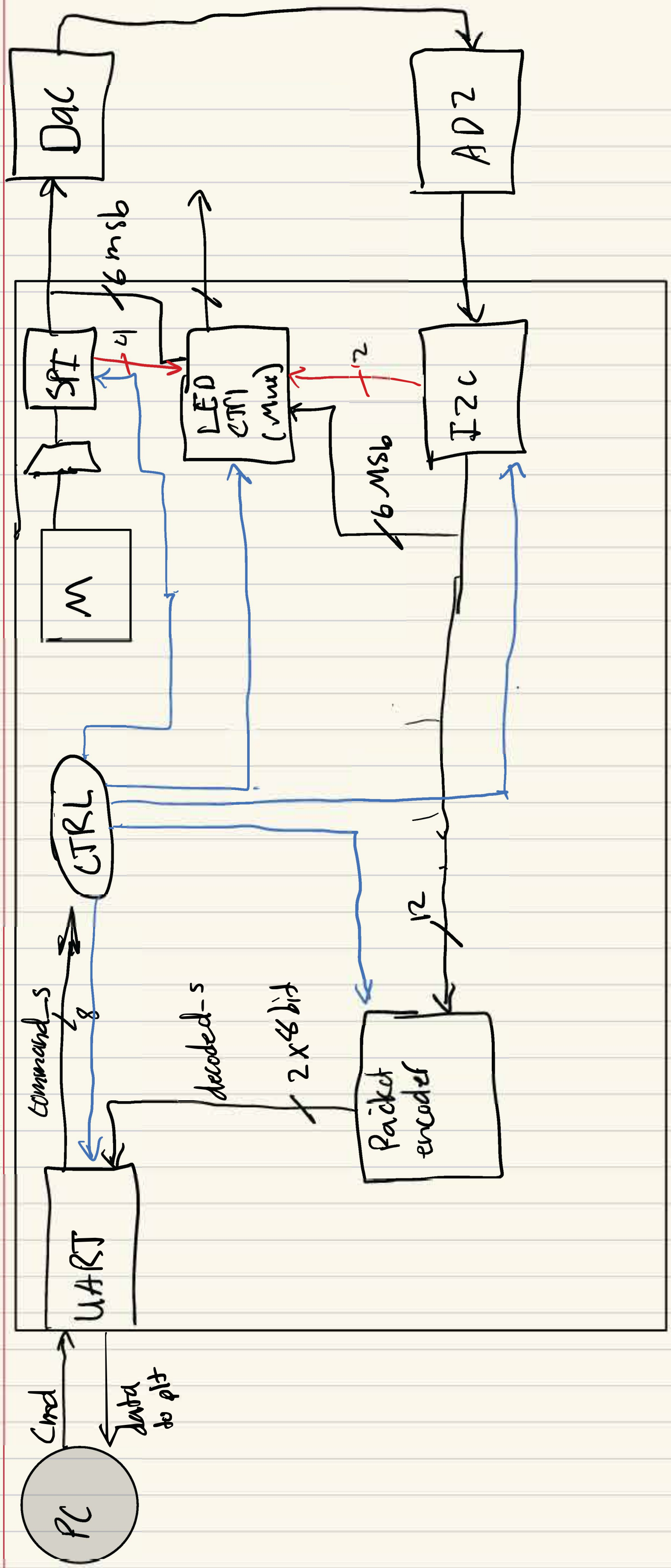
time to write $= \frac{1}{100kHz} = 10\mu s$

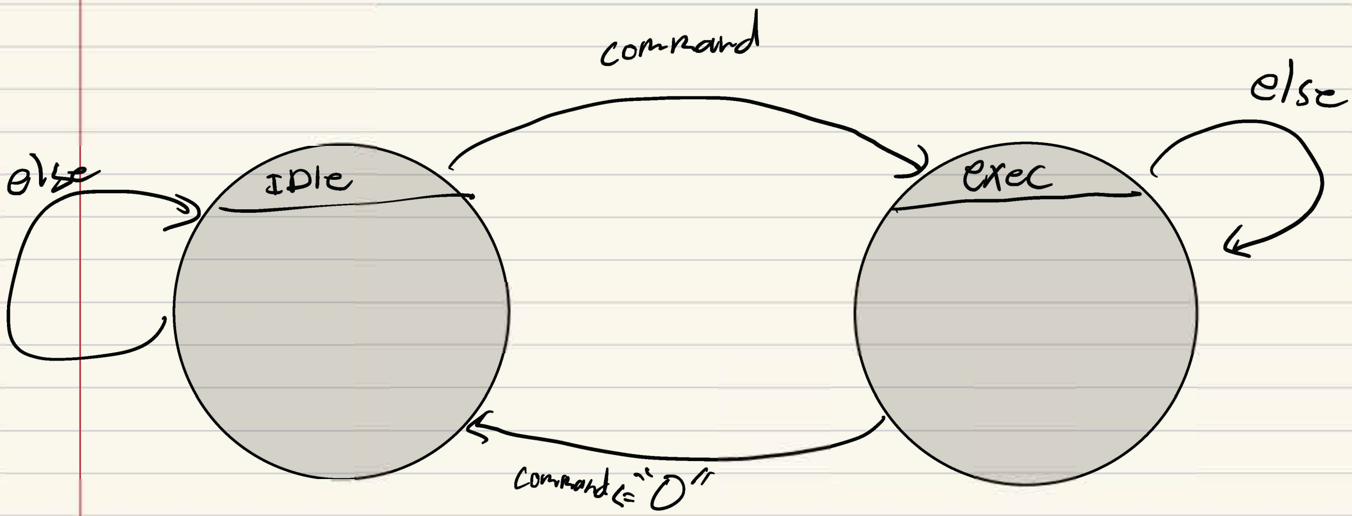
time to read $= \frac{1}{10MHz} = 1 \times 10^{-7} s$

$\frac{10\mu s}{1 \times 10^{-7} s} = 100$

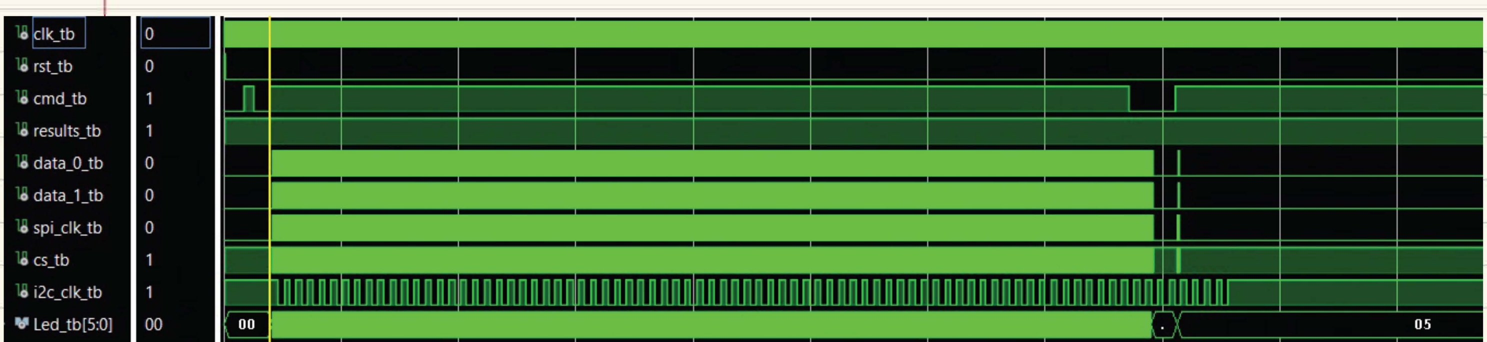
FIFO = 3000 − 100

Fifo Depth = 2900


PC
Cmd
data to plt
UART
command_s
8
CTRL
decoded_s
2x8 bit
Packet encoder
12
M
SPI
4
6 msb
Dac
LED Ctrl (Mux)
6 MSb
12
I2C
AD2



command
else
IDle
exec
else
command <= "0"



clk_tb 0
rst_tb 0
cmd_tb 1
results_tb 1
data_0_tb 0
data_1_tb 0
spi_clk_tb 0
cs_tb 1
i2c_clk_tb 1
Led_tb[5:0] 00
00
05

SPI

00 Idle
01 transmit 1/transmit 2

I2C

000 waiting
001 config/conack
010 idle
011 transmit 1/transmit 2/ack
100 receive/master ack/stop

Commands

X	X	X	X	state	SPI	ADC	sw/rem

Key	bin	lights	in
0	0000	X	X
2	0010	ADC	sw
3	0011	ADC	mem/cont
4	0100	SPI	sw
5	0101	SPI	mem/cont
8	1000	state	sw
9	1001	state	mem/cont

state lights

0000XX	waiting	I2C 4MSB	
0001XX	configuration		
0010XX	idle		
0100XX	transmitting		
1000XX	receiving		
XXXX01	Idle	SPI	
XXXX10	transmit		

FIFO Size

write with 12 read width is 6

Uart full count - 1736.11

Uart half cout - 868

I2C clk speed to 3ksPS

$$\frac{100e6}{3000} \approx 33\,333.3$$

Block mem

16 bit · 2000 = 3200

SPI clk

$$\frac{100e6}{50e3} = 2000$$

FIFO Depth

Write frequency = 100kHz

Reading frequency = 10 MHZ

Burst length = 3000

$$\text{time to write} = \frac{1}{100\text{kHz}} = 10\mu s$$

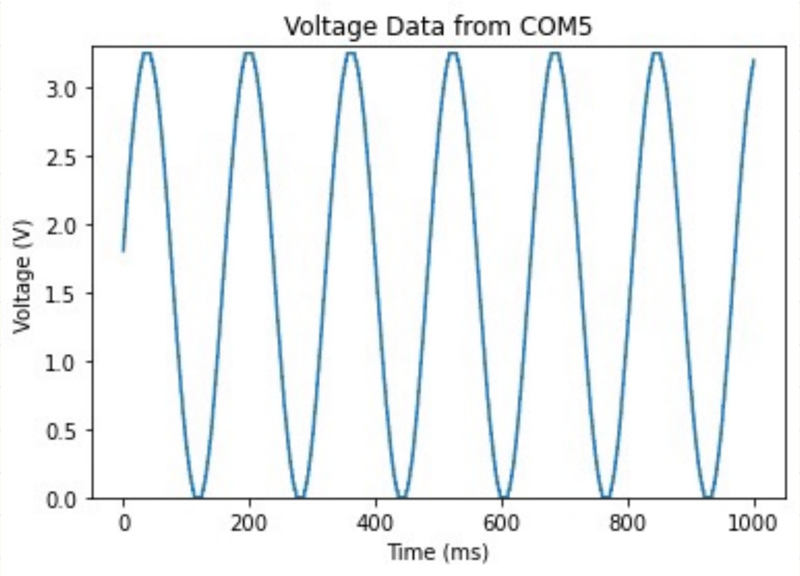
$$\text{time to read} = \frac{1}{10\text{MHz}} = 1\times10^{-7}s$$

$$\frac{10\mu s}{1\times10^{-7}s} = 100$$

FIFO = 3000 − 100

$\boxed{\text{Fifo Depth} = 2900}$

Conclusion



We were able to successfully implement the final design. We learned that our I2C's speed was causing our sine wave to be warped. Once we corrected our clock speed this caused our sine wave to correctly show up. We were not able to implement the FIFO. In order to successfully implement the FIFO. We need to connect the output of the I2C to the FIFO then we take the output of the FIFO and connect it to the encoder.