EE 3420 Lab Guide: Digital to Analog Converter

Written by: Grant Seligman, and Gabe Garves

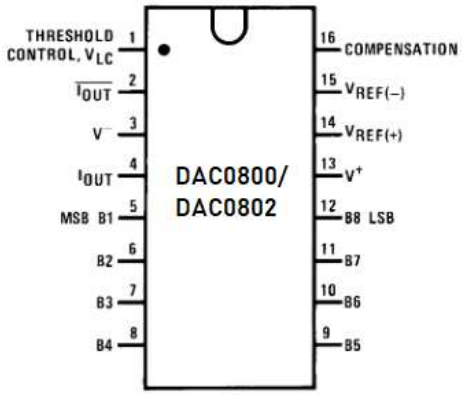

Figure 1

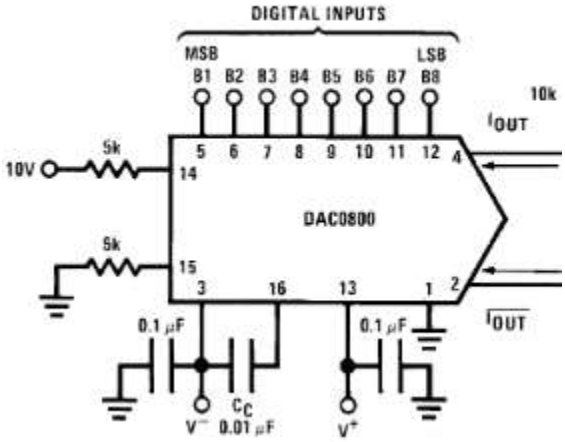

Figure 2

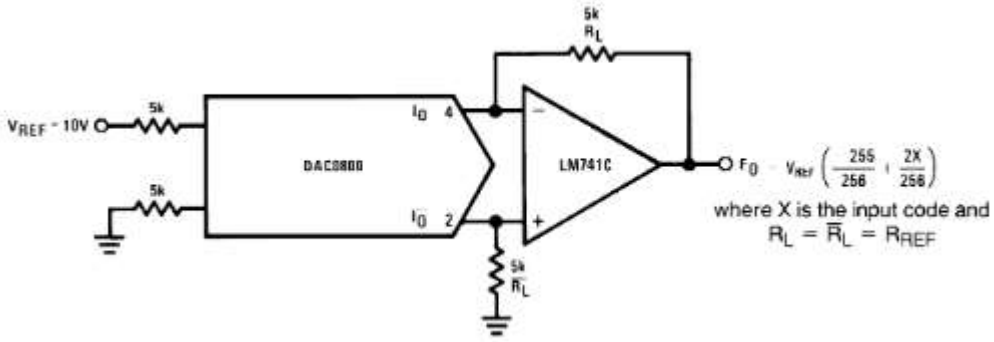

Figure 3

All figures come from the DAC0800 NI datasheet [1]

Example Overview:

In the world of signals, it's important to be able to convert between analog and digital signals. In this case, we will be converting a digital signal into an analog signal. Unfortunately, the Arrow MAX1000 does not have an on-board digital to analog converter (DAC). Instead we will be utilizing an external DAC0800 chip along with an external LM741CN op-amp. A verilog module will be created to have the Max10 input a digital signal to our DAC which will then convert it to an analog signal. The sinusoidal output waveform can be measured on an oscilloscope.

Verilog Breakdown:

The verilog module in **Figure 4** takes in a clock signal and outputs a sequence of 16 8-bit values. Each 8 bit sample represents a point on the signal over a single period. Samples with values from 128 to 255 represent a point between 0 and 1. Samples with values from 0 to 127 represent points below the x-axis or 0 to -1. For every positive edge of the clock input, an 8-bit sample is fed to the DAC and a counter is incremented so that the next clock cycle will send the next sample in sequence. Adjusting the analog signal output from the DAC can be done by changing the size of the sample set sent and how fast we send them.

```
1  //DAC Module Testing
2  /*
3  AUTHOR: GRANT SELIGMAN
4  DATE: 3/6/2020
5  FROM: TXST SENIOR DESIGN PROJECT FALL 2019-SPRING 2020
6  FOR: TEXAS STATE UNIVERSITY STUDENT AND INSTRUCTOR USE
7  */
8  module DAC_MS_Module(s_out,clk);
9  //the clock needs to be at 1MHz
10 output reg [7:0]s_out;
11 input wire clk;
12 reg [3:0] counter;
13 //reg with each sample bit; 8 bits per word, 16 words in register array
14 reg [7:0]sample_set[15:0];
15
16 //s_out[0] is the LSB, s[7] is the MSB
17 //could add a clock converter and a counter to switch pins to diff values
18 initial
19 begin
20     s_out = 8'b10000000; //output starts at is 0 volts
21     counter = 4'd0;
22     //setting sample points into the sample_set register
23     sample_set[0]  = 8'b10000000; //128
24     sample_set[1]  = 8'b10110001; //177
25     sample_set[2]  = 8'b11011010; //218
26     sample_set[3]  = 8'b11110110; //246
27     sample_set[4]  = 8'b11111111; //255
28     sample_set[5]  = 8'b11110110; //246
29     sample_set[6]  = 8'b11011010; //218
30     sample_set[7]  = 8'b10110001; //177
31     sample_set[8]  = 8'b10000000; //128
32     sample_set[9]  = 8'b01001111; //79
33     sample_set[10] = 8'b00100110; //38
34     sample_set[11] = 8'b00001010; //10
35     sample_set[12] = 8'b00000000; //0
36     sample_set[13] = 8'b00001010; //10
37     sample_set[14] = 8'b00100110; //38
38     sample_set[15] = 8'b01001111; //79
39 end
40
41 always@(posedge clk)
42     begin
43     //since counter is upto 4 bits, once it reaches 1111(15 in binary)
44     //counter should reset back to 0 and begin increasing again.
45         if(counter <= 4'd15)
46         begin
47             s_out = sample_set[counter];
48             counter = counter + 1;
49         end
50         else
51         begin
52             counter = 0;
53         end
54     end
55
56 endmodule
```

Figure 4

In order to calculate the 16 samples you first have to choose a frequency to start with. For this lab, 10kHz was chosen. To find the period:

$$Period = \frac{1}{Frequency} = seconds$$

$$Period = T = \frac{1}{10000} = 0.0001\ seconds$$

Now that we have the period, we need to find the Δt or the spacing between each of the 16 samples:

$$\Delta \quad = \frac{Period}{Samples} = seconds/sample$$

$$\Delta \quad = \frac{0.0001}{16} = 6.25\ \mu s/sample$$

Once we have the Δt, we can calculate each y(t) value of the sine wave at each sample over a single period (in radian mode):

$$y(\quad) = \sin\left(2\frac{*}{T}\right) for\ 0 \le \quad \le T, -1 \le y \le 1$$

$$y(6.25\ \mu s) = \sin\left(2\frac{*6.25}{.0001}\right) = 0.3826$$

We need to take this sample value and convert it to a number between 0 and 255 since we are dealing with 8-bits data samples.

$$Sample¿ = \frac{(y(\quad)+1)*255}{2}$$

$$Sample¿ = \frac{(0.3826+1)*255}{2} \approx 177 = 10110001$$

Make sure to round up the sample because we are not converting floats, only integers into binary for this lab.

Note: You can throw these equations in excel and have it do the math and binary conversion for you. The function for decimal to binary conversion is:
=DEC2BIN(number, # of bits).

When sampling any signal, you have to take Nyquist's Theorem into account. You have to sample at least 2 x *frequency* in order to get a visible waveform. The higher the sampling rate along with the number of samples taken, the more accurate the final waveform will be. For the best resolution, you want the sample rate to be 10 x *frequency*

at least. So we will run this verilog module at 1Mhz. Depending on the DAC, it will have a set frequency range it can operate in. The DAC0800's digital side has a settling time of 100ns which means it can handle a frequencies up to 10MHz.[1]

FPGA Implementation:

Go ahead and make a new **Quartus Project** and create a **symbol file** of the above module. NIOS isn't necessary for this lab and it is easier to program the DAC without it, Next you will need to create a **PLL** for the 1MHz signal clock input into the DAC module. Revert back to the DTMF lab guide if you need assistance with creating a PLL. Then take the DAC module and the PLL and connect them in a **Block Diagram file.** Your circuit should look similar to **Figure 5.** Then use **Figure 6** to set your pins.

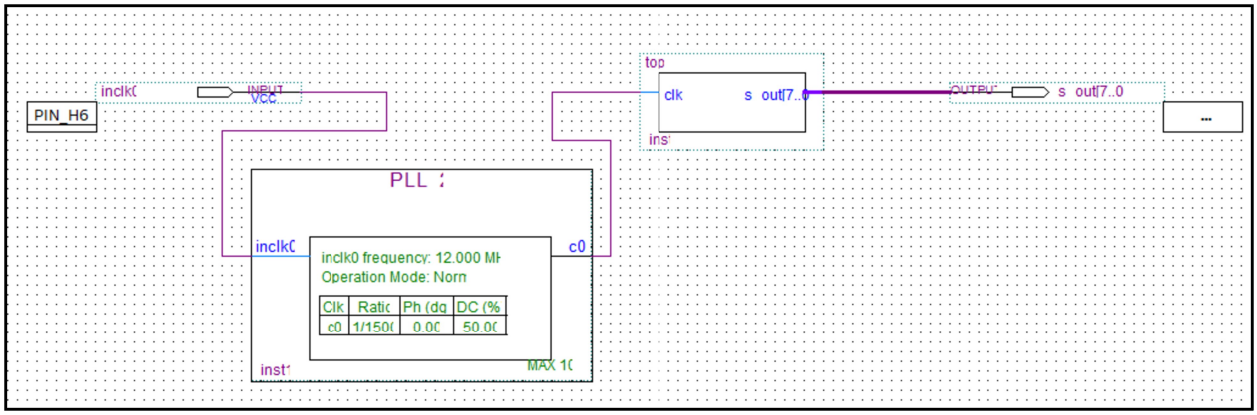

Figure 5

Wiring Up the Circuit

When connecting the DAC to the FPGA on a breadboard, make sure that the s_out[0] output pin from the Max1000 is connected to the LSB pin on the DAC and the s_out[7] is connected to the MSB DAC pin. **Figure 2** and **Figure 3** show how the DAC and the op-amp should be wired together. The Oscilloscope will be connected to ground and the output pin of the op-amp. You should get a result similar to **Figure 7 or Figure 8.** **Figure 8** is close to what you should observe.

As a disclaimer, due to the 2020 quarantine this lab has not been fully tested so the oscilloscope output may vary. The verilog module may need to be adjusted if stepping like in Figure 7 occurs. Ask your instructor for help if you need assistance getting this setup working.

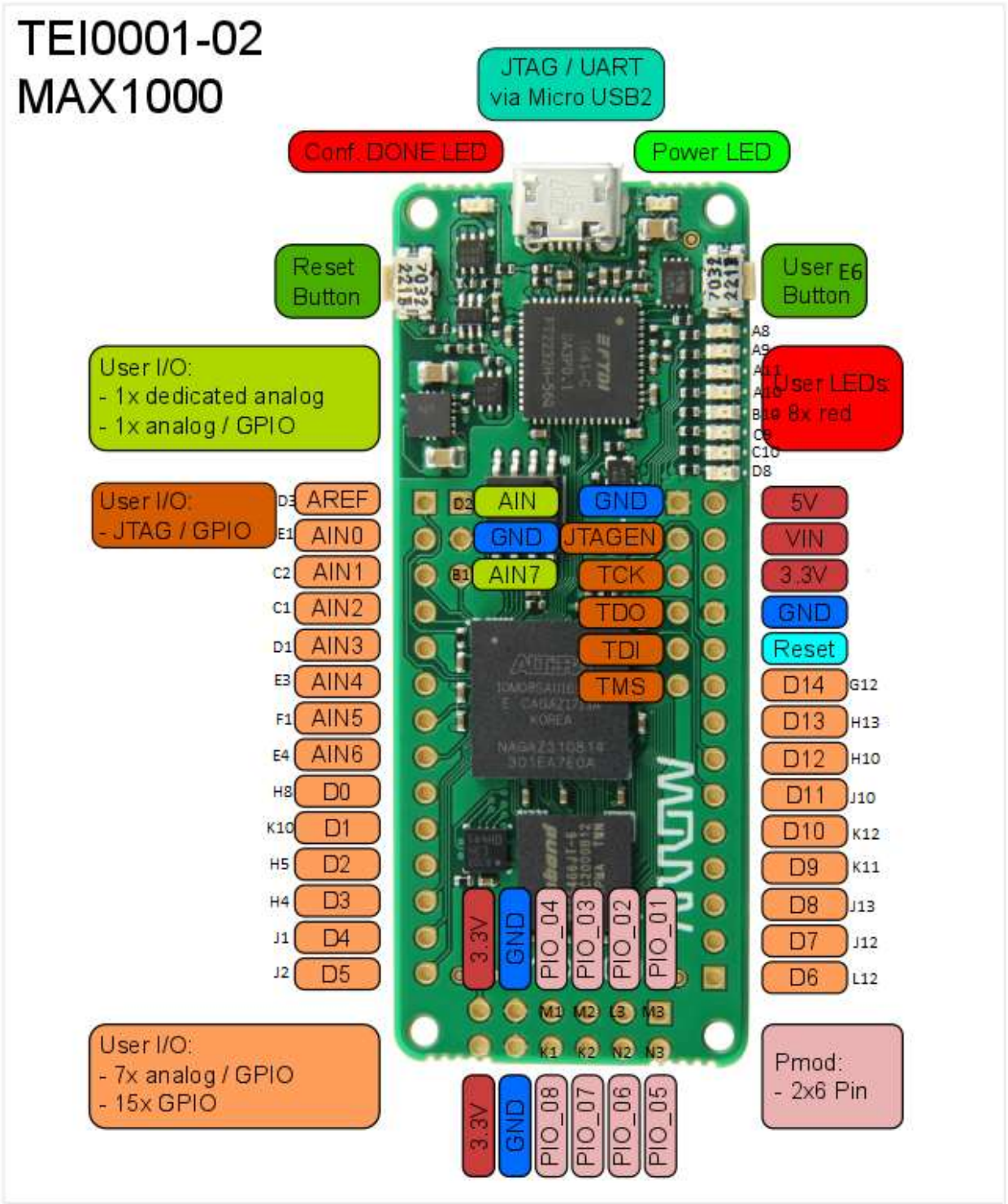
TEI0001-02
MAX1000
JTAG / UART
via Micro USB2
Conf. DONE LED
Power LED
Reset
Button
User E6
Button
User I/O:
- 1x dedicated analog
- 1x analog / GPIO
User LEDs:
8x red
User I/O:
- JTAG / GPIO
AREF
AIN0
AIN1
AIN2
AIN3
AIN4
AIN5
AIN6
D0
D1
D2
D3
D4
D5
AIN
GND
JTAGEN
AIN7
TCK
TDO
TDI
TMS
5V
VIN
3.3V
GND
Reset
D14
D13
D12
D11
D10
D9
D8
D7
D6
3.3V
GND
PIO_04
PIO_03
PIO_02
PIO_01
3.3V
GND
PIO_08
PIO_07
PIO_06
PIO_05
User I/O:
- 7x analog / GPIO
- 15x GPIO
Pmod:
- 2x6 Pin

Figure 6

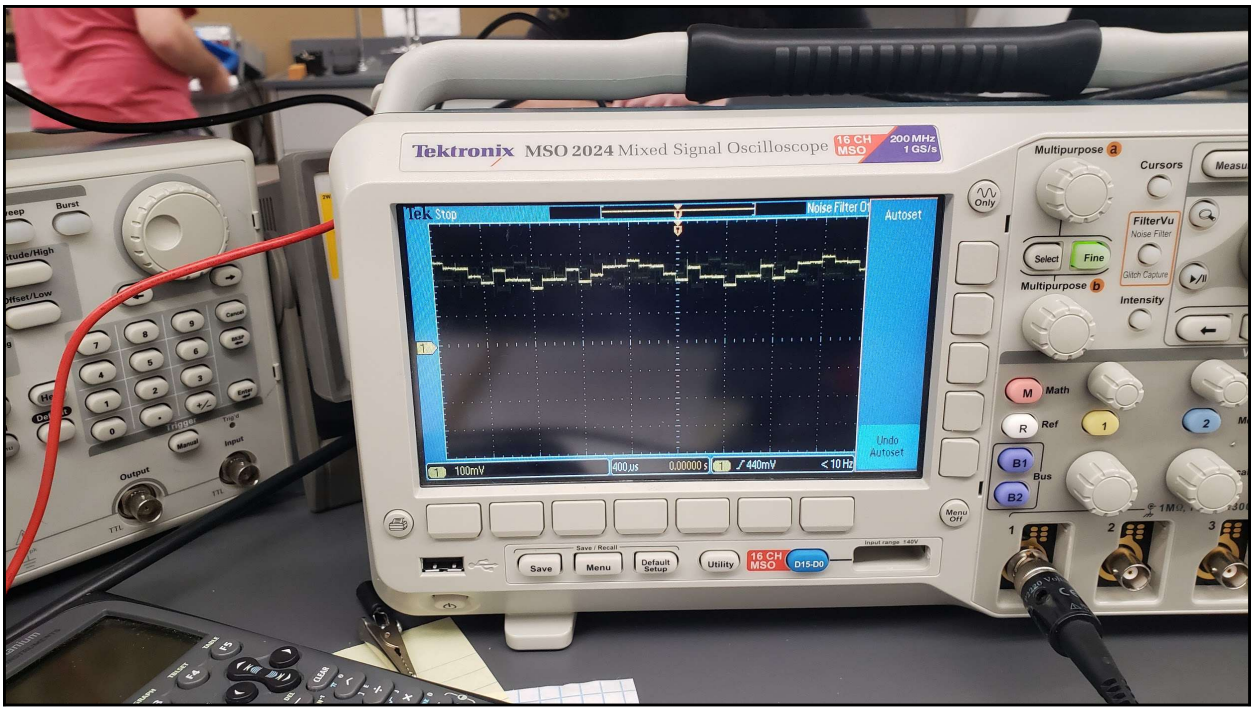

Figure 7

This waveform has some stepping issues and this could either be the DAC module or you set the PLL to too low of a frequency. Both these figures had the PLL drop down the 12MHz clock at 10kHz instead of 1Mhz.

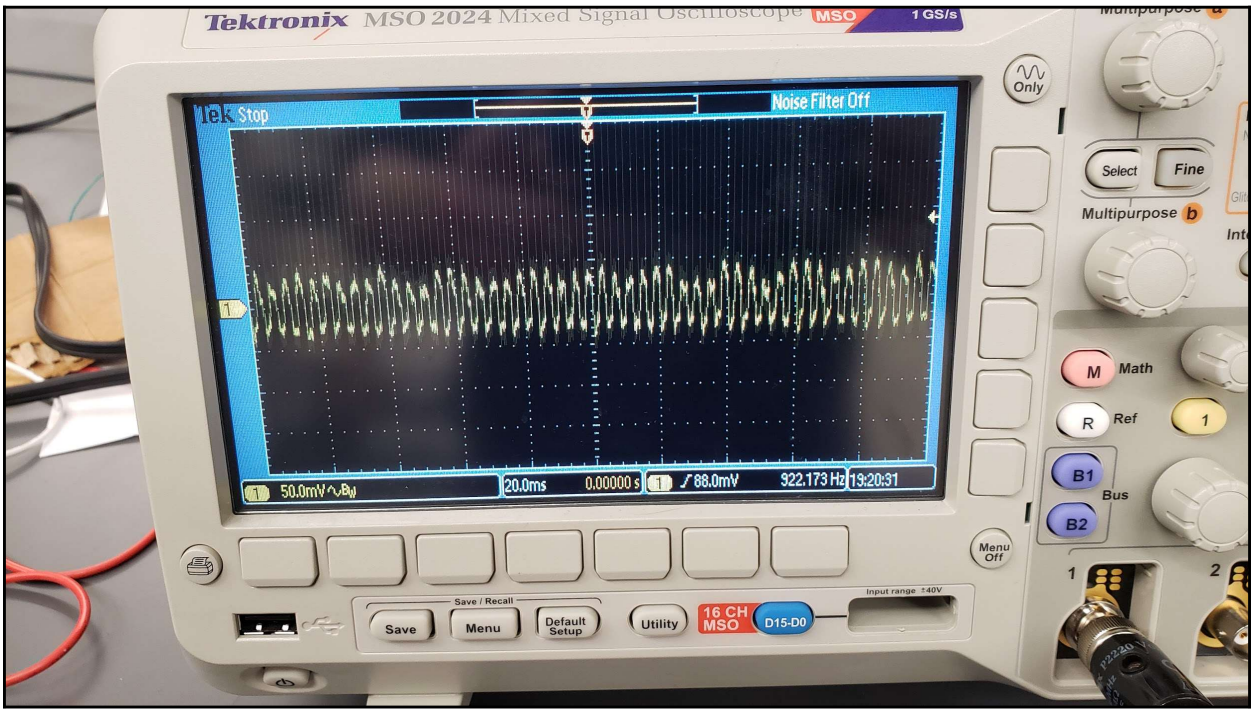

Figure 8

References

[1] “dac0800.pdf.” Accessed: Apr. 28, 2020. [Online]. Available: http://www.ti.com/lit/ds/symlink/dac0800.pdf.