Lab 3
FIR Filters
EE 445S

Enoc Balderas Daniel Diamont

February 18, 2019

Date Performed: February 4, 2019
Instructor: Professor Evans

1 Introduction

For this lab we focused on generating sinusoidal signals. We tested the performance and resolution of three different signal generation methods, math library function calls, difference equation, and look up table (LUT). Additionally, we compared two methods of outputting values: interrupts and DMA.

The math library function call uses an 11th order polynomial to approximate the sinusoid. We can use the math library function with a phase increment variable to output a sinusoid in an interrupt function.

1.1 Difference Equation

$$y[n] = sin(w_0)x[n-1] + 2cos(w_0)y[n-1] - y[n-2] \tag{1}$$

The difference equation is derived from the z-transform of a causal single sided sinusoid. Since this difference equation is derived from the transfer function of a causal single sided sinusoid, then the impulse response should be $y[n] = u[n]cos[w_0 n]$. The main advanteges of using this method are that it is not computationally intense nor does it need much memory.

1.2 EDMA: Look Up Table

The main benefit of having a LUT is that we can have very good resolution and not much computation. The only drawback is that we have to store the table. In this part of the lab we also familiarized ourselves with the EDMA hardware. Using the EDMA has the benefit of freeing the CPU from having to handle memory transfers.

2 Methods

2.1 Difference Equation

We created global variables for the desired sinusoid frequency and the initial conditions for the difference equation. Each time the ISR is called we calculate the current value $y[n]$ and update the buffers $y[n-1]$ and $y[n-2]$. Finally we scaled the output before writing it to the codec.

2.2 EDMA: Look Up Table

We calculated the $gcd(f, f_s)$ to find the number of samples needed to represent the sinosoid. Next we used the math library calls to calculate the LUT.

3 Results

3.1 Difference Equation

Code:

```
  /* variable initialization */
  if(flag == 1)
  {
    theta1 = (2*pi*6000.0/fs);
    theta2 = (2*pi*2000.0/fs);

    y1[1] = sinf(theta1);
    y2[1] = sinf(theta2);

    a1 = 2*cosf(theta1);
    a2 = 2*cosf(theta2);
    flag = 0;
  }

  .
  .
  .

  /* Difference Equation */
  y1[0] = a1*y1[1] - y1[2];
  y1[2] = y1[1];
  y1[1] = y1[0];

  y2[0] = a2*y2[1] - y2[2];
  y2[2] = y2[1];
  y2[1] = y2[0];
```

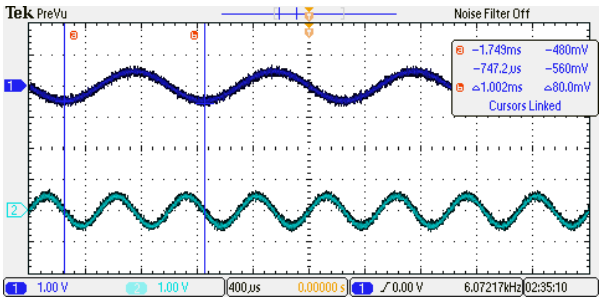

Figure 1: Blue: 1kHz Green: 2kHz.

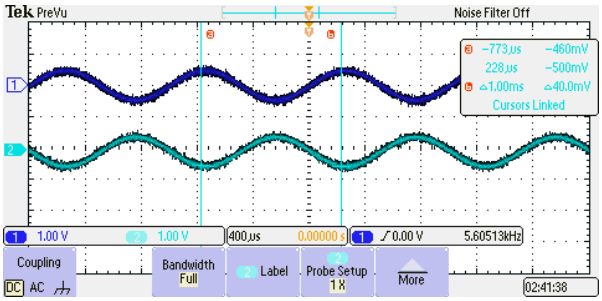

Figure 2: Blue: 1kHz Green: 7kHz.

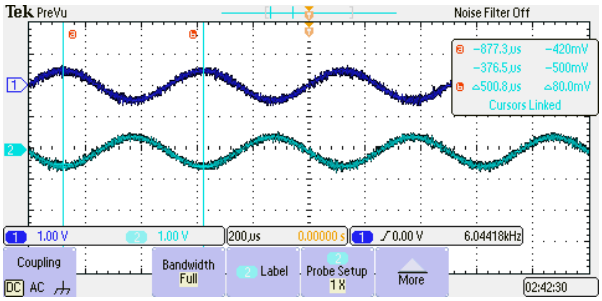

Figure 3: Blue: 2kHz Green: 6kHz.

3.2 EDMA: Look Up Table

Code:

```
%createtable
Uint32 A = 32000;
Uint32 i;
float temp;

float f = 1000;
float fs = 8000;

%LUT
for(i = 0; i < BUFFER_COUNT; i++)
{
  temp = sinf(2*M_PI*(f/fs)*i);
  table[i] = (Uint32) A*temp;
}
```

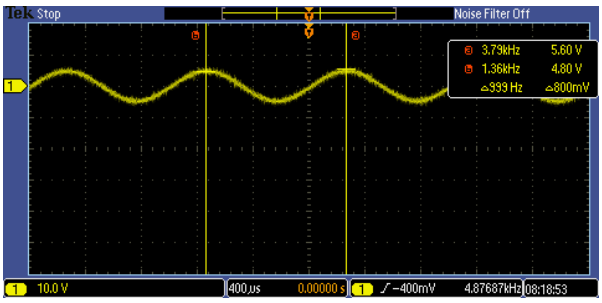

Figure 4: 1kHz sinusoid

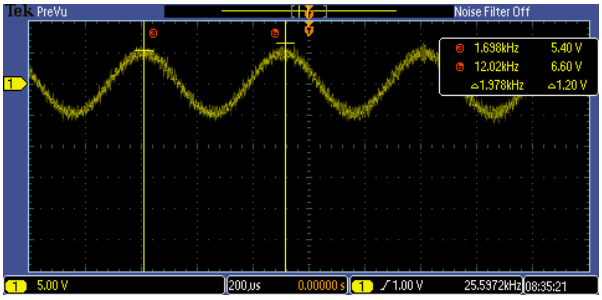

Figure 5: 2kHz sinusoid

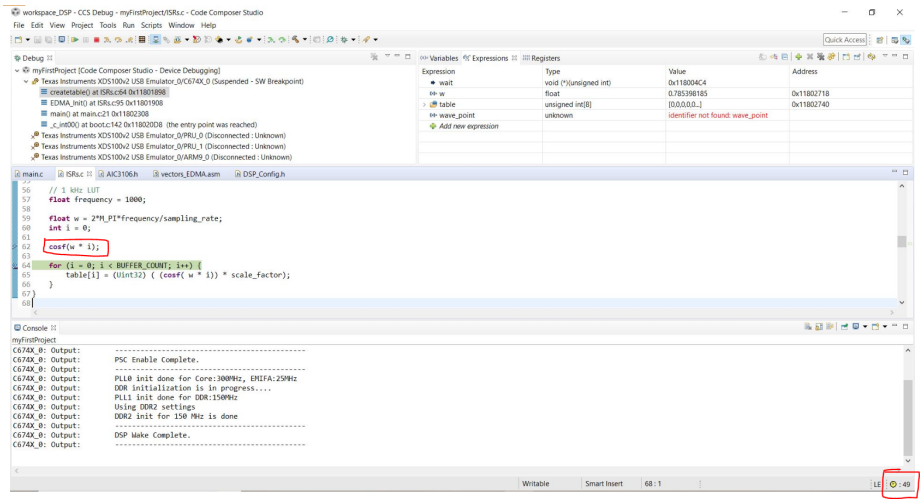

Figure 6: Clock cycles

4 Discussion

4.1 Difference Equation

The biggest issue we had implementing our difference equation was initializing the initial conditions. We copied the code from the book, but we overlooked a scaling factor that seemed inconsequential. To fix the problem we ended up changing our code a bit to make it clear what the initial conditions actually were.

From: `y1[3] = {0 1 0}`
To: `y1[3] = {0 sinf(w_0) 0}`

4.2 EDMA: Look Up Table

The LUT presented type casting challenges. We realized that we were typecasting before scaling which caused us to have significant roundoff error. We fixed our error by breaking up our statement into multiple statements.

5 Answers questions

5.1 Difference Equation

1. Explain what happened mathematically.

   Answer: Since we output at a rate of 8 kHz, and the Nyquist theorem states that we can only accurately represent signals with frequency content below half our output rate (4 kHz). This is also known as the folding

frequency. Because of this, everything above 4 kHz is projected symmetrically across the folding frequency. Therefore, 7 kHz maps to 1 kHz and 6 kHz maps to 2 kHz.

5.2 EDMA: Look Up Table

1. Compare and contrast the three methods.

   Answer:

   - Polling: The advantage of polling is that it is simple to implement. However, the processor will be busy waiting.
   - Interrupts: The advantages are that it frees the processor from busy waiting, and allows for easy implementation of sample by sample processing. On the other hand, they are more complex to implement than busy-wait. Also, there is the added overhead of the processor contex switching between threads, and performing memory accesses to store and retrieve data.
   - DMA: The advantage is that there is no contex switching between threads nor memory accesses, since the processor is not involved in the acquisition of data. The disadvantage is that the configuration of the DSP board is the most complex of the three methods, since it must be configured for EDMA.

2. Why is scaling necessary?

   Answer:
   Scaling is necessary for two reasons:

   - To make full use of the DAC's range
   - If the scaling was left out we would have a lot of roundoff error when the cosf floating point result gets casted to an integer.

3. How many cycles does a sin take? How many cycles between samples?

   Answer: A sin function call takes 49 cycles. There are 46,875 ($\frac{375MHz}{8kHz}$) cycles between samples.