EE 113 DA\_ Digital Signal Processing Design

Lab 2: Fast Fourier Transform

Professor Mike Briggs, Fall 2018

Brian Tehrani UID: 604715464

Milad Nourian UID: 004854226

Attiano Purpura-Pontoniere UID:504847318

**Objective:**

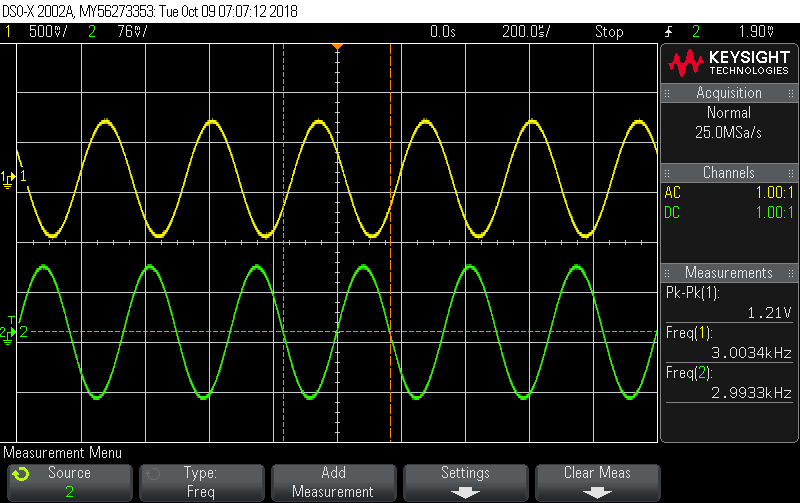
The purpose of Lab 2: Fast Fourier Transform(FFT) is to become familiar with the FFT function in Code Composer Studio(CCS) and creating input from the function generator. The lab requires us to read input from the function generator to an array, then take the FFT and IFFT of the array after performing computations in the frequency domain. We also take the FFT of an oddsized array, a square wave, and a 2-Dimensional FFT.

**Step 1:**

In step 1 of the lab we setup the function generator, LCDK, computer, and oscilloscope.

**Step 2:**

In step 2 of the lab we ran the TA’s code template in CCS to read from the Line Input (connected to the function generator) and write to the Line Output. We set the function generator to output a 3KHz .5Vpp sinusoid. The following picture shows the corresponding oscilloscope output:

Figure 1: 3KHz sinusoid output on oscilloscope from function generator and LCDK

**Step 3:**

In step 3 of the lab we modified the TA’s code to save samples from the function generator in a 1024-long array, we then plotted this array in CCS.

Relevant code:

*interrupt void interrupt4(void) // interrupt service routine*

*{*

*int16\_t left\_sample;*

*// Input from ADC (Line IN)*

*left\_sample = input\_left\_sample();*

*// Your code here*

*if(idx<N){*

*// Input is being read sample by sample real part in even indices, imaginary in odd.*

*x\_in[2\*idx]=left\_sample;*

*x\_in[2\*idx+1]=(float)0.0;*

*// Variable idx is global and its value is kept*

*idx++;*

*}*

*//flag*

*if (idx == 1024){*

*flag = 1;*

*}*

*// Output to DAC (Line OUT)*

*output\_left\_sample(left\_sample);*

*return;*

*}*

**Step 4:**

In step 4 of the lab we modified the TA’s code further to take the FFT of the samples of the 3KHz sinusoid we saved in an array in step 3 and calculated the magnitude of the FFT and then graphed it. We repeated these steps for a 4KHz sinusoid. Here are the following plots of the 3 and 4KHz sinusoids:

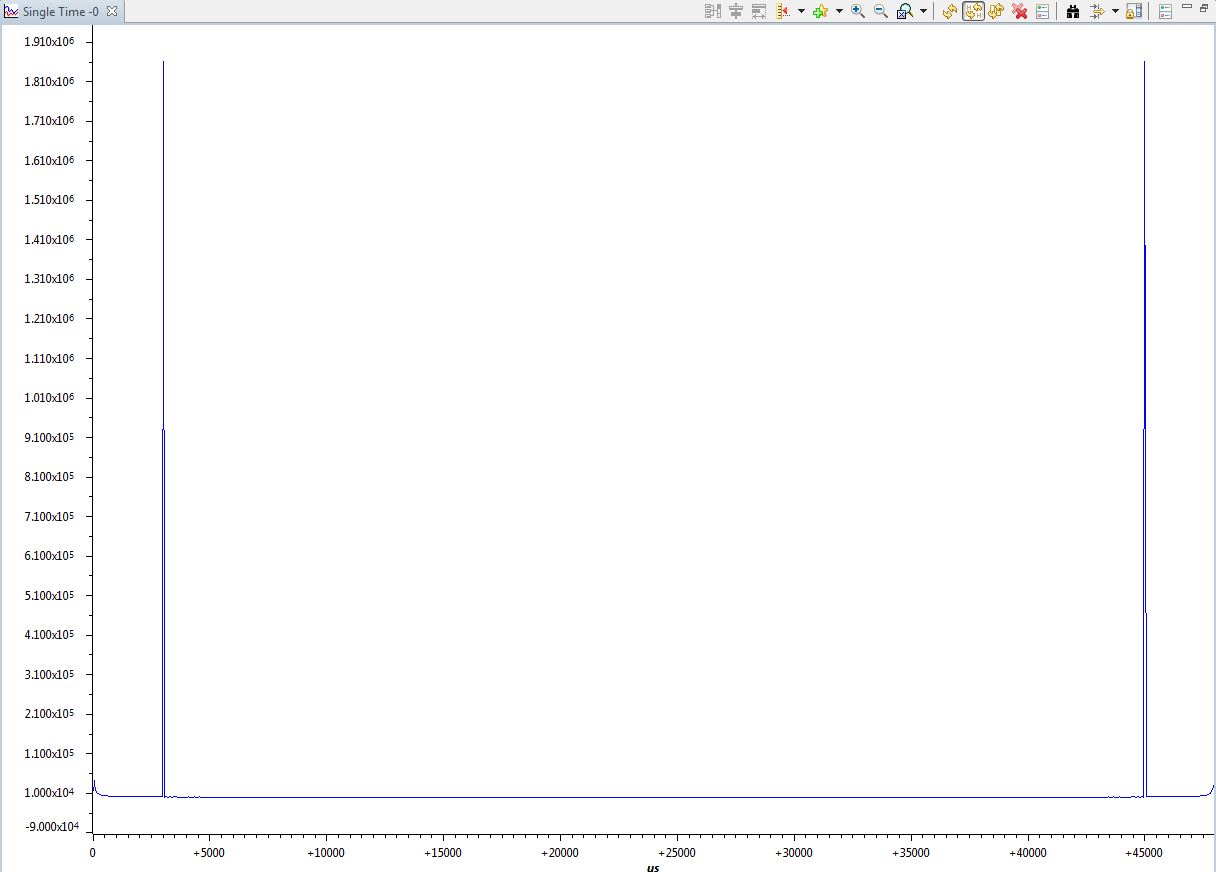


Figure 2: 3KHz sinusoid FFT Magnitude graph in CCS

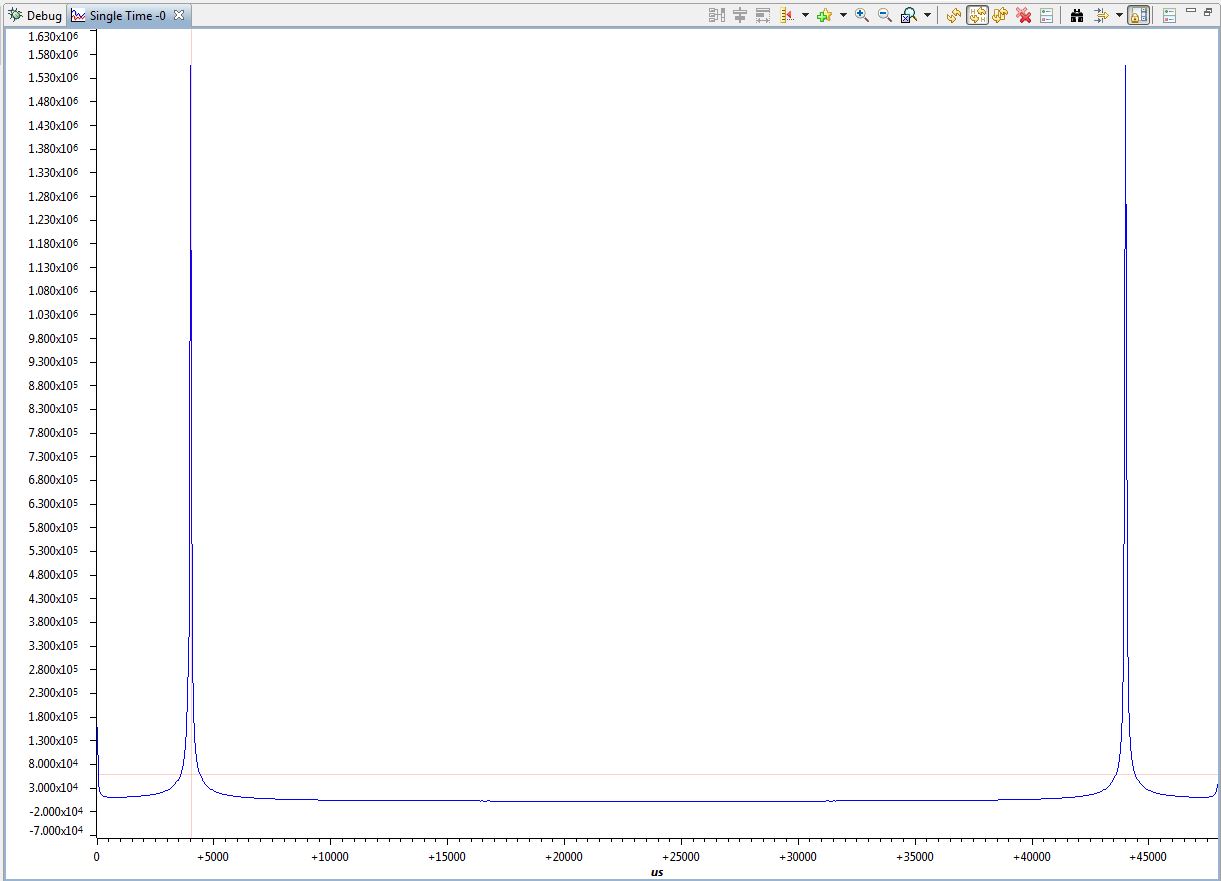
****

Figure 3: 4KHz sinusoid FFT Magnitude graph in CCS

Discussion:

The plot in figure 2 shows the FFT of a 3 kHz sinusoid and the plot for figure 3 shows the FFT of a 4 kHz sinusoid. We can see that the FFT both have peaks at 3 kHz and 4 kHz, and 45 and 44 kHz respectively. The 45 and 44 kHz peaks occur because of the sampling frequency of 48000Hz.

Relevant Code:

*while (1){*

*if(flag == 1){*

*int16\_t n;*

*for (n=0; n<N; n++)*

*{*

*x\_sp[2\*n] = x\_in[2\*n];*

*x\_sp[2\*n+1] = x\_in[2\*n+1];*

*x\_in\_new[n]= x\_in[2\*n];*

*}*

*// Call twiddle function to generate twiddle factors needed for FFT and IFFT functions*

*gen\_twiddle\_fft\_sp(w\_sp,N);*

*// Call FFT routine*

*DSPF\_sp\_fftSPxSP(N,x\_sp,w\_sp,y\_sp,brev,4,0,N);*

*// Call routine to separate the real and imaginary parts of data*

*// Results saved to floats y\_real\_sp and y\_imag\_sp*

*separateRealImg ();*

*int i = 0;*

*for (i = 0; i < N; i++){*

*magnitude [i]=*

*sqrt(y\_real\_sp[i]\*y\_real\_sp[i]+y\_imag\_sp[i]\*y\_imag\_sp[i]);*

*}*

**Step 5:**

In step 5 of the lab we used DSPF\_sp\_ifftSPxSP.c to take the IFFT of the the time domain sinusoidal waveform from its FFT. We then plotted separately the real and imaginary parts of the IFFT. We did this for the 4KHz sinusoids. For the 4000hz IFFT the peak-to-peak period is 250microseconds, which means the frequency is 1/period = The following screenshot details this further:

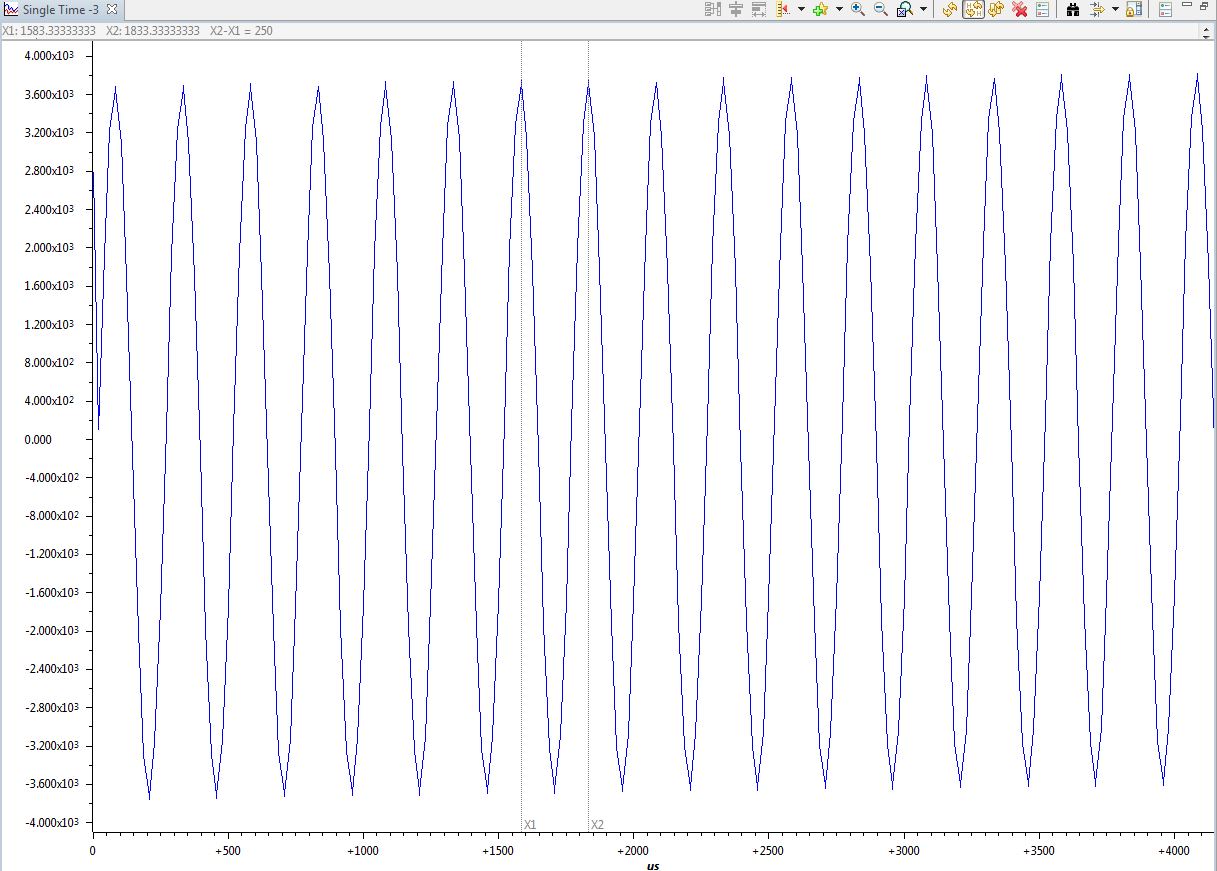


Figure 4: 4KHz sinusoid IFFT real part graph in CCS

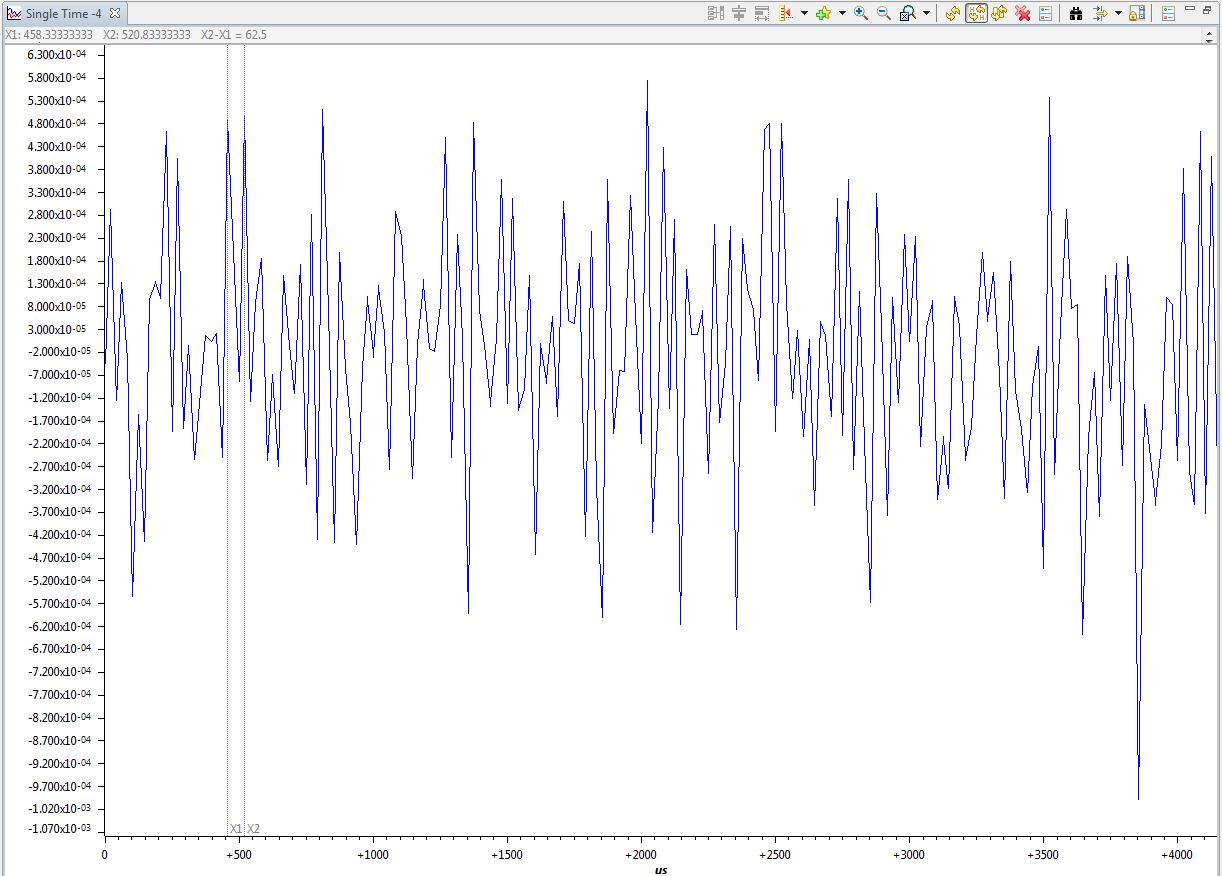


Figure 5: 4KHz sinusoid IFFT imaginary part graph in CCS

Relevant code:

*float x\_real\_sp[N];*

*float x\_imag\_sp[N];*

*separateRealImgForX () {*

*int i, j;*

*for (i = 0, j = 0; j < N; i+=2, j++) {*

*x\_real\_sp[j] = x\_sp[i];*

*x\_imag\_sp[j] = x\_sp[i + 1];*

*}*

*}*

*// Call the inverse FFT routine*

*DSPF\_sp\_ifftSPxSP(N,y\_sp,w\_sp,x\_sp,brev,4,0,N);*

*separateRealImgForX();*

**Step 6:**

In step 6 of the lab we zeroed out the last 124 elements of the input to see the effect on the FFT with zero padding on the first 900 values of the input sinusoid. We then plotted the FFT with padded zeros for the 3 and 4KHz sinusoids. We observed sinc like behavior around the peak which makes sense as the additional zeros makes the input mimic a pulse function. The following screenshots detail it further:

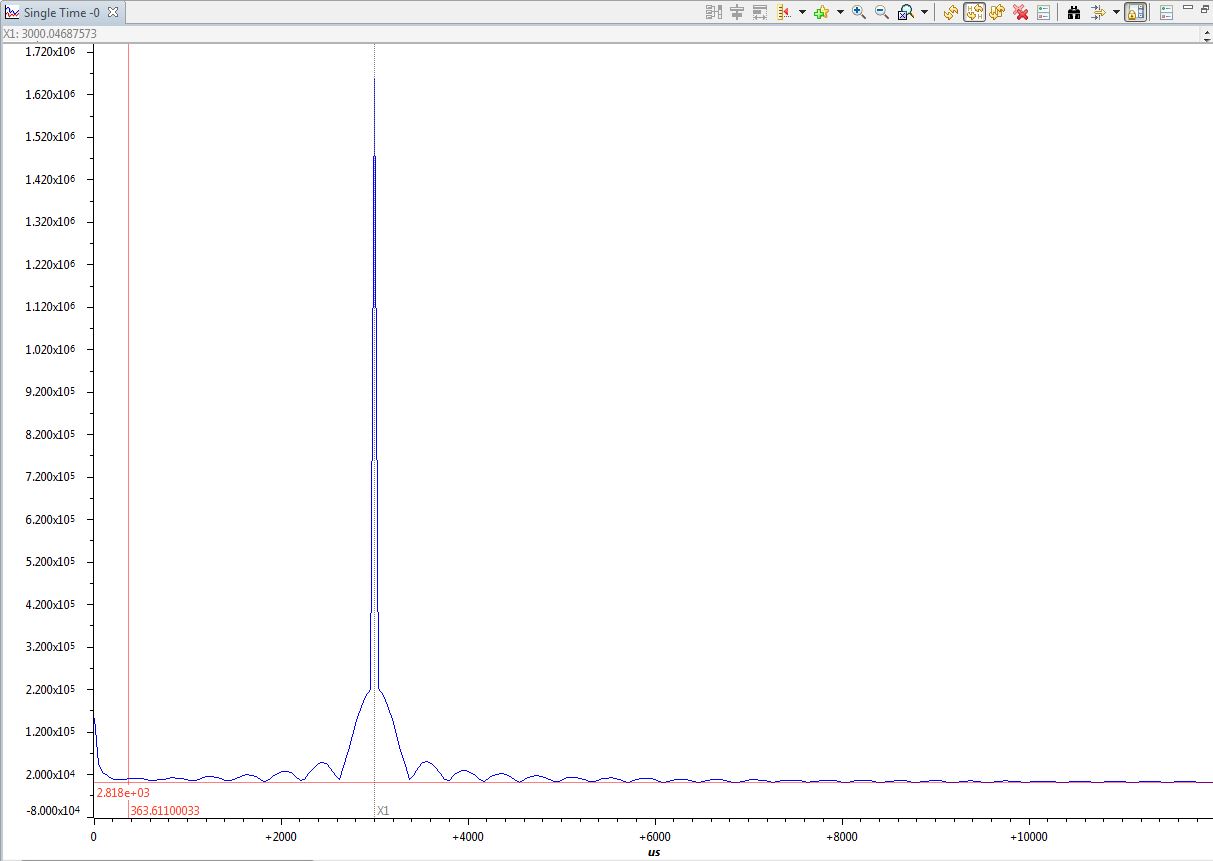
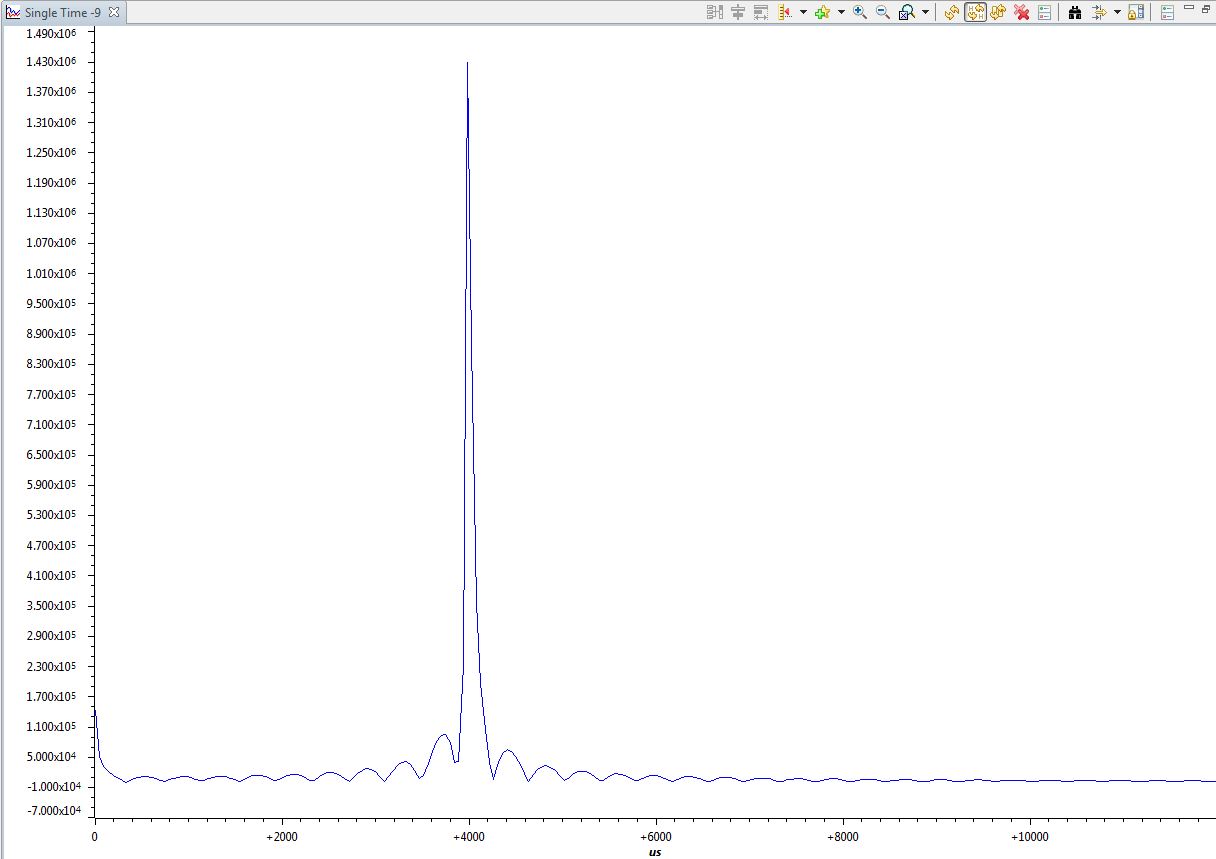
Figure 6: 3KHz sinusoid FFT magnitude graph with last 124 elements padded 0s in CCS

Figure 7: 4KHz sinusoid FFT magnitude graph with last 124 elements padded 0s in CCS

Discussion:

The differences between the FFT of the sinusoids in step 4 and step 6 is that in step 4 we had sharp peaks, in step 6 we have sinc-like behavior and fillets around the peaks. This is because we padded the array in step 6 with trailing 0s which makes the input act like a sinusoidal pulse and the fourier transform of a pulse is the sinc function, so we have a sinc peak at the frequency of the zero padded sinusoid.

The difference between the FFT of the zero-padded 4kHz and 3kHz sinusoids in step 6 besides the fact that the peak for the 3kHz is at 3000 and the peak for the 4kHz is at 4000, is that there is spectral leakage in the 4kHz signal so it isn't clear that the FFT is a sinc function overlaid on top of a delta at the frequency of the input sinusoid. The zero padded 3kHz FFT shows clearly that there is a sinc function overlaid on top of a delta function at 3kHz, but the 4kHz graph doesn't. The zero padded 4kHz sinusoid causes spectral leakage because at a sampling frequency of 48000samp/s, when the 4kHz sinusoid values are stored in an array containing 1024 samples, it contains a partial cycle. (1024samples(48 samples/s 4cycles/s))=85.333 cycles. Partial cycles create spectral leakage.

Relevant Code:

// added code from above

*int16\_t n;*

*for (n=0; n<N; n++)*

*{*

*x\_sp[2\*n] = x\_in[2\*n];*

*x\_sp[2\*n+1] = x\_in[2\*n+1];*

*x\_in\_new[n]= x\_in[2\*n];*

*}*

*int l;*

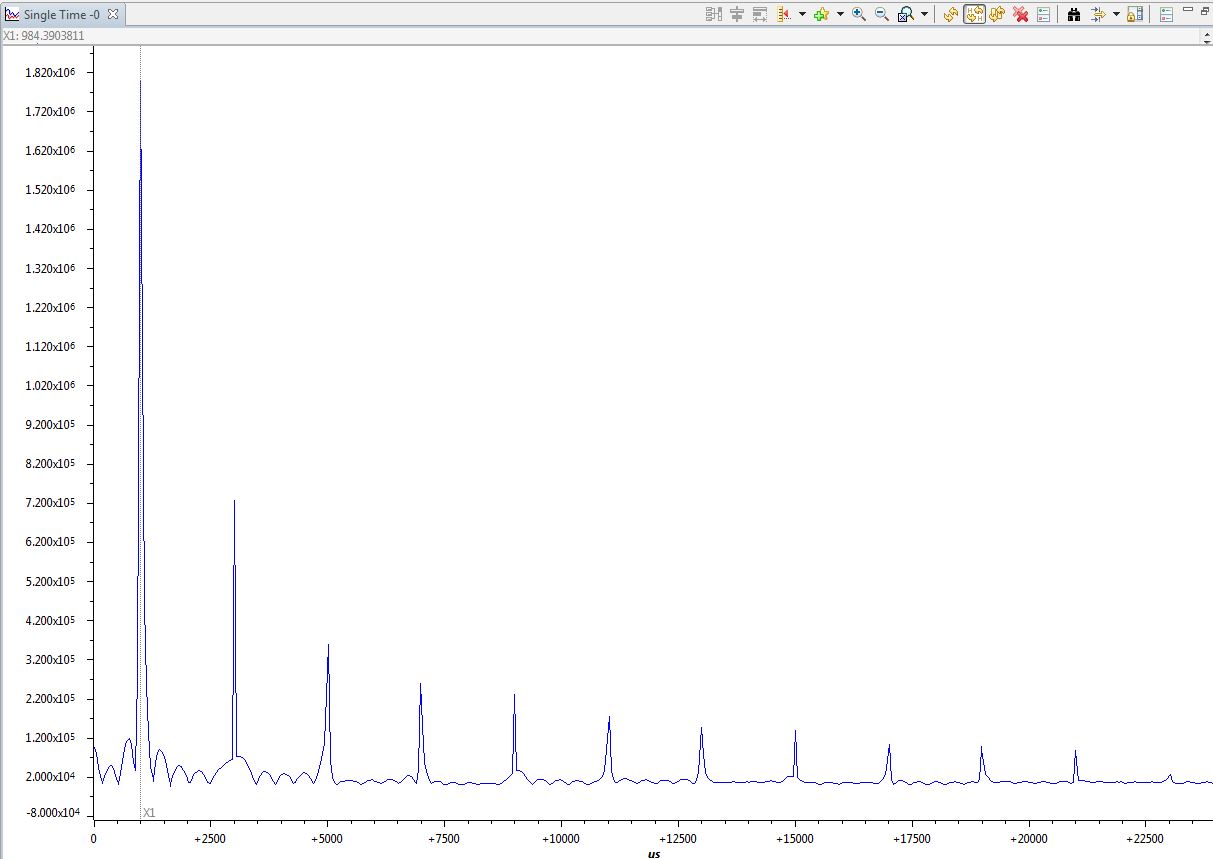
*for(l=1800;l<2048;l++){*

*x\_sp[l]=0;*

*}*

**Step 7:**

In step 7 of the lab we created a 1KHz square wave using the function generator, took the FFT of it, and then plotted it in both CCS and on the oscilloscope. We took the following screenshots of the plots:

Figure 8: 1KHz Square wave FFT magnitude graph in CCS

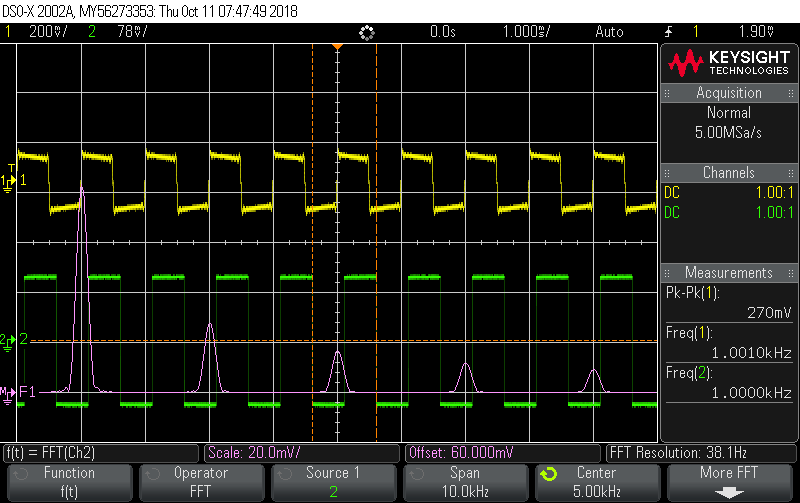


Figure 9: 1KHz Square wave & FFT magnitude graph on Oscilloscope

Discussion:

A square wave is a composition of many sinusoids, known as a linear combination of sinusoids. The FFT of a square wave shows the fundamental frequency at the frequency of the square wave and decaying peaks at every odd harmonic. Both the oscilloscope and CCS screenshots display this behavior. The full screenshot of the CCS actually displays a mirrored image of the harmonics, but the image was cropped to match the oscilloscope image. In reality the LCDK samples at 48 Ksps, therefore we can see the mirrored image. However, the oscilloscope has a sampling rate of 2 Gsps and the mirrored image will not show on the screen. The fillets appear in the CCS but not the oscilloscope because the 1024 array contains a partial cycle of the square wave.

**Step 8:**

In step 8 of the lab we are tasked with creating several variations of 2D FFT arrays. The first 2D array consists of 0’s and in the center we instantiate a square of 1’s. The size of this created array is 64x64. We then add 0’s to every even column to create a 2D array of both real and complex values. To get the magnitude of this 64x128 array we separate the array into a 64x64 real and a 64x64 complex. The graph of the magnitude is plotted in Matlab and shown in figure 10.

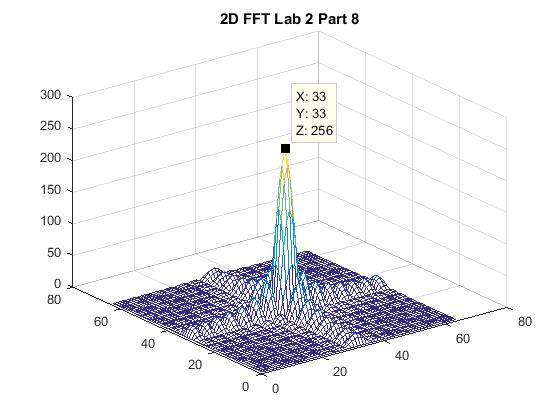
****

Figure 10: White square inside a black square array

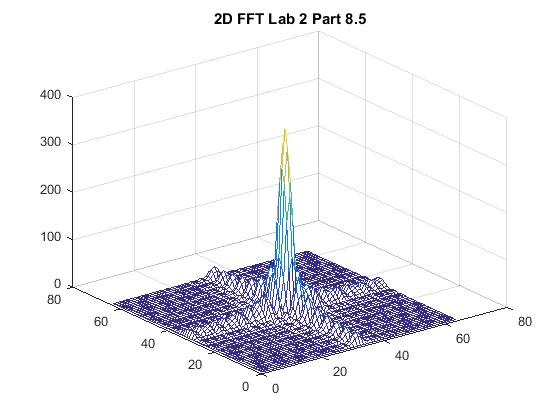


Figure 11: Rectangular 1’s array 24x16

For the next part we must be able to stretch the array in one dimension to create a rectangular 1’s array size 24x16. The magnitude plot is shown in figure 11.

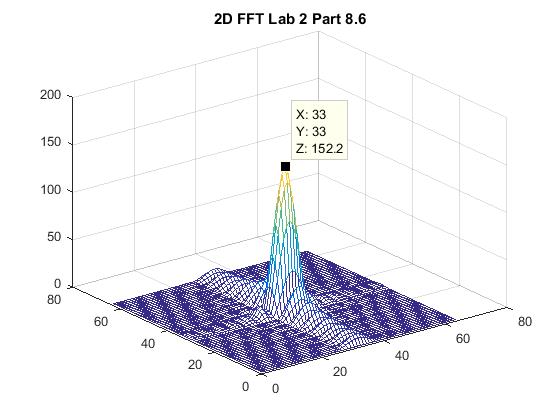


Figure 12: Sine array 16x16

For the last part of we need to create a 16x16 sine array at the center if the zeros array.

The results show the waveguide output translated form the xy plane in to the uv plane. From figure 10 to figure 11 we can see that by increasing the size of the aperture we get a higher beam directivity. This means that the peak of the wave is sharper and at a higher value. The waveguide in figure 12 shows a sine wave in one dimension and brick wall in another. We know that the multiplication in time domain for 2 signals would result in convolution in the frequency domain. It can be seen that ripples are lower in one direction than they are in the other direction.

Code:

Anastasios said it was ok not to show the code.