EE288 Data Conversions/Analog Mixed-Signal ICs
Spring 2018

Lecture 5: Spectrum Analysis

Prof. Sang-Soo Lee
sang-soo.lee@sjsu.edu
ENG-259

Course Schedule – Subject to Change

Date	Topics
24-Jan	Course introduction and ADC architectures
29-Jan	Converter basics: AAF, Sampling, Quantization, Reconstruction
31-Jan	ADC dynamic performance metrics, Spectrum analysis using FFT
5-Feb	ADC & DAC static performance metrics, INL and DNL
7-Feb	~~OPAMP and bias circuits review~~
12-Feb	SC circuits review
14-Feb	~~Sample and Hold Amplifier - Reading materials~~
19-Feb	Flash ADC and Comparator: Regenerative Latch
21-Feb	Comparators: Latch offset, preamp, auto-zero
26-Feb	Finish Flash ADC
28-Feb	DAC Architectures - Resistor, R-2R
5-Mar	DAC Architectures - Current steering, Segmented
7-Mar	DAC Architectures - Capacitor-based
12-Mar	SAR ADC with bottom plate sampling
14-Mar	SAR ADC with top plate sampling
19-Mar	Midterm Review
21-Mar	$\boxed{\text{Midterm exam}}$
26-Mar	Spring break
28-Mar	Spring break
2-Apr	Pipelined ADC stage - comparator, MDAC, x2 gain
4-Apr	Pipelined ADC bit sync and alignment using Full adders
9-Apr	Pipelined ADC 1.5bit vs multi-bit structures
11-Apr	Fully-differential OPAMP and Switched-capacitor CMFB
16-Apr	Single-slope ADC
18-Apr	Oversampling & Delta-Sigma ADCs
23-Apr	Second- and higher-order Delta-Sigma Modulator.
25-Apr	Hybrid ADC - Pipelined SAR
30-Apr	Hybrid ADC - Time-Interleaving
2-May	ADC testing and FoM
7-May	Project presentation 1
8-May	Project presentation 2
14-May	Final Review
20-May	Project Report Due by 6 PM

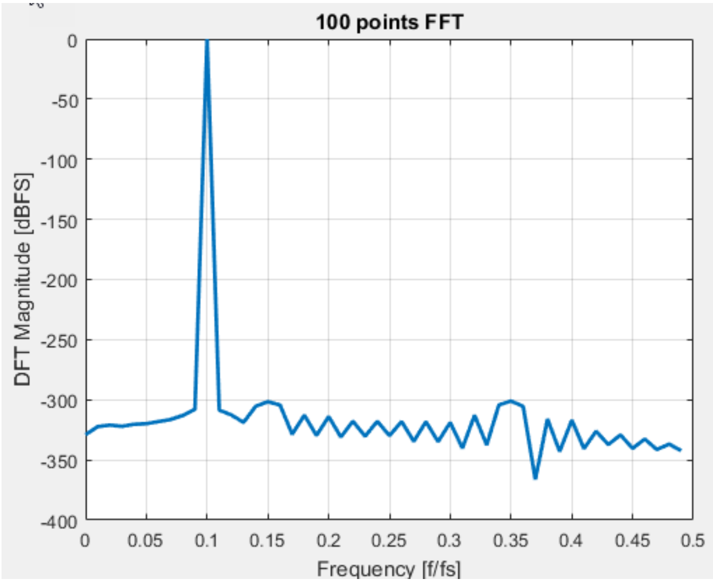
No class on Feb 14, next Wed.

Spectrum Analysis Using MATLAB

***Midterm Exam dates are approximate and subject to change with reasonable notice.**

FFT MATLAB Example 1

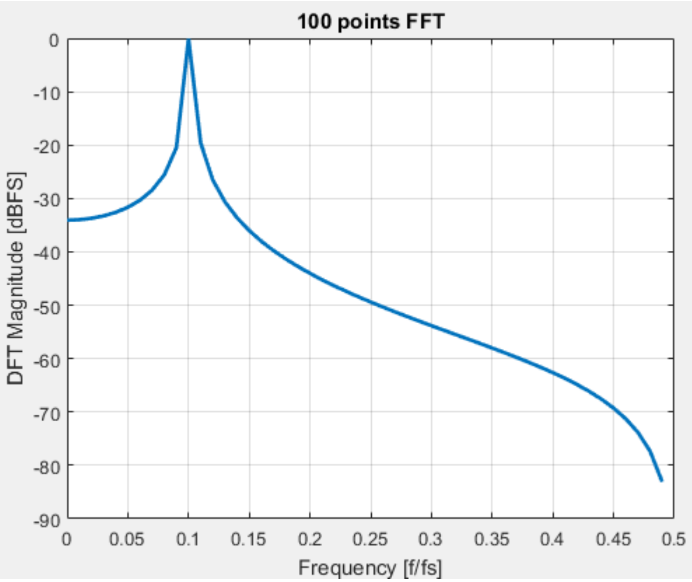
```
clear all; close all; clc;
N = 100;                        % FFT size
fs = 1000;                      % Sampling rate
fx = 100;                       % Input signal tone
FS = 1;                         % Full Scale (actually half)
t = 0:N-1;
x = FS*cos(2*pi*fx/fs*t);
s = abs(fft(x));                % FFT and take absolute
s = s(1:end/2);                 % remove redundant half of spectrum
s = 20*log10(2*s/N/FS);         % dB relative to full-scale
f = [0:N/2-1]/N;                % frequency vector
plot(f, s, 'linewidth', 2);
xlabel('Frequency [f/fs]');
ylabel('DFT Magnitude [dBFS]');
title(strcat(num2str(N),' points FFT'));
grid on;
```



fft_p51.m

FFT MATLAB Example 2

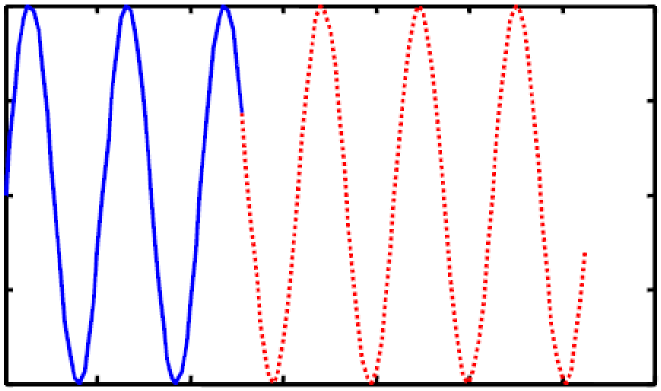
```
clear all; close all; clc;
N = 100;                        % FFT size
fs = 1000;                      % Sampling rate
fx = 101;                       % Input signal tone
FS = 1;                         % Full Scale (actually half)
t = 0:N-1;
x = FS*cos(2*pi*fx/fs*t);
s = abs(fft(x));                % FFT and take absolute
s = s(1:end/2);                 % remove redundant half of spectrum
s = 20*log10(2*s/N/FS);         % dB relative to full-scale
f = [0:N/2-1]/N;                % frequency vector
plot(f, s, 'linewidth', 2);
xlabel('Frequency [f/fs]');
ylabel('DFT Magnitude [dBFS]');
title(strcat(num2str(N),' points FFT'));
grid on;
```

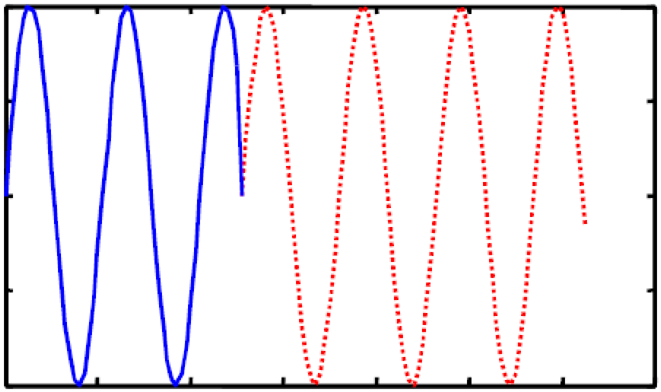


fft_p51.m

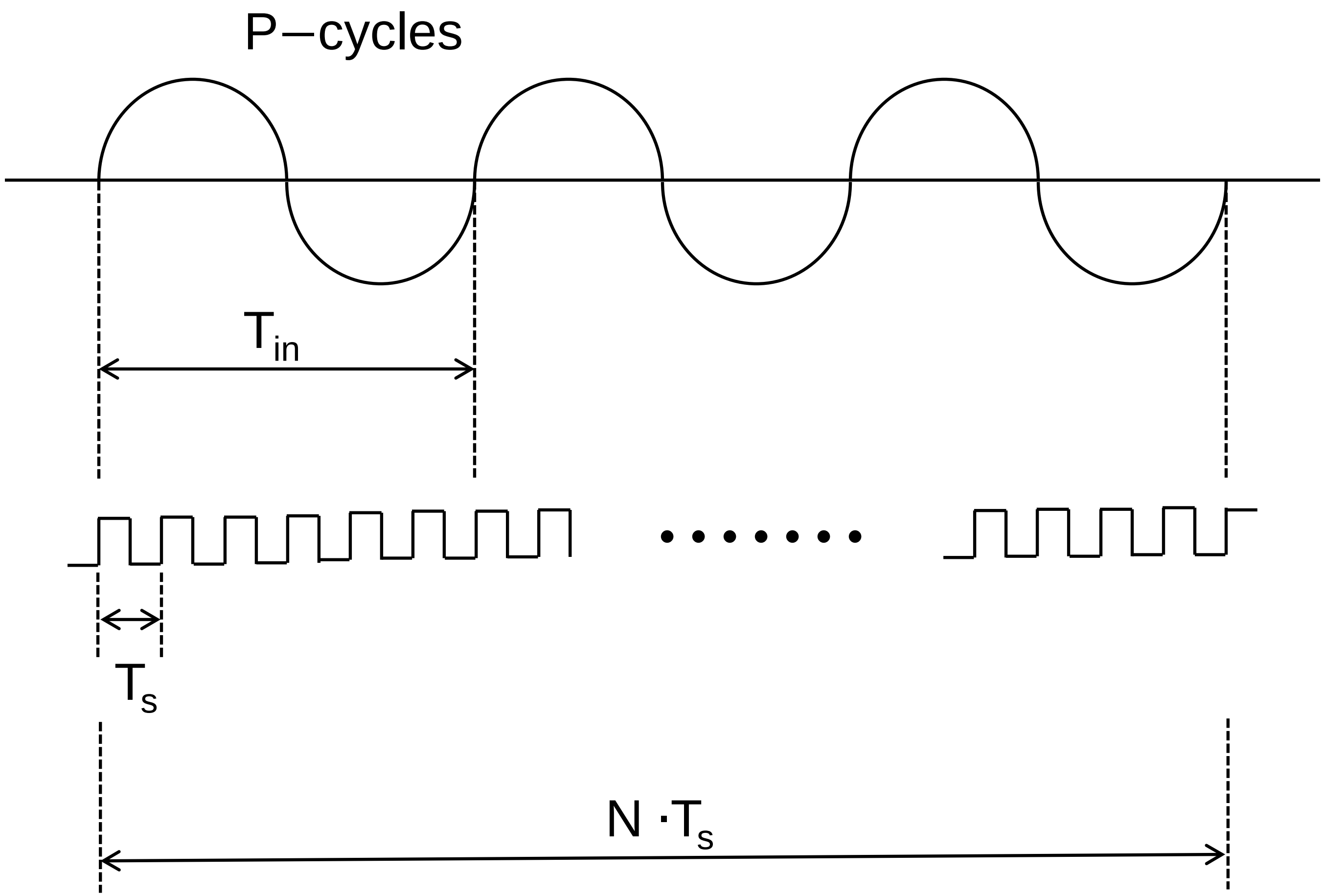
Spectral Leakage

- DFT implicitly assumes that data repeats every N samples
- A sequence that contains a non-integer number of sine wave cycles has discontinuities in its periodic repetition
  - Discontinuity looks like a high frequency signal component
  - Power spreads across spectrum
- Two ways to deal with this
  - Ensure integer number of periods
  - Windowing





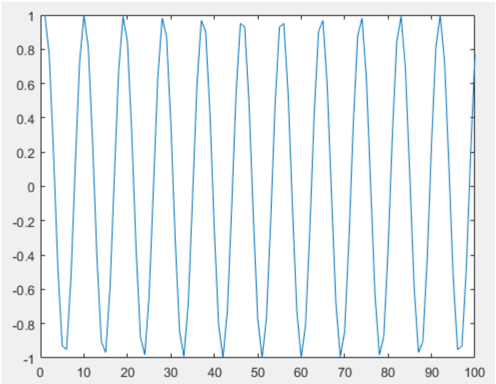
Integer Number of Cycles for N-point FFT

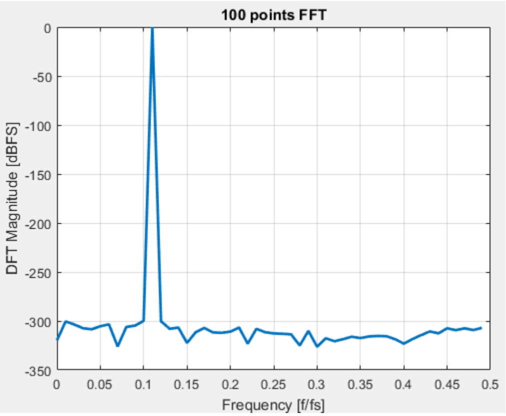


$$P \cdot T_{in} = N \cdot T_s \Rightarrow \frac{P}{f_{in}} = \frac{N}{f_s} \Rightarrow \boxed{f_{in} = P \cdot \frac{f_s}{N}}$$

Integer Number of Cycles : Coherent Sampling

```
clear all; close all; clc;
N = 100;                        % FFT size
cycles = 11;                    % Signal tone
fs = 1000;                      % Sampling rate
fx = cycles*fs/N;               % Input signal tone
FS = 1;                         % Full Scale (actually half)
t = 0:N-1;                      % Time vector
x = FS*cos(2*pi*fx/fs*t);       % Cosine signal
figure; plot(x);
s = abs(fft(x));                % FFT and take absolute
s = s(1:end/2);                 % remove redundant half of spectrum
s = 20*log10(2*s/N/FS);         % dB relative to full-scale
f = [0:N/2-1]/N;                % frequency vector
figure; plot(f, s, 'linewidth', 2);
xlabel('Frequency [f/fs]');
ylabel('DFT Magnitude [dBFS]');
title(strcat(num2str(N),' points FFT'));
grid on;
```





fft_p52.m

Windowing

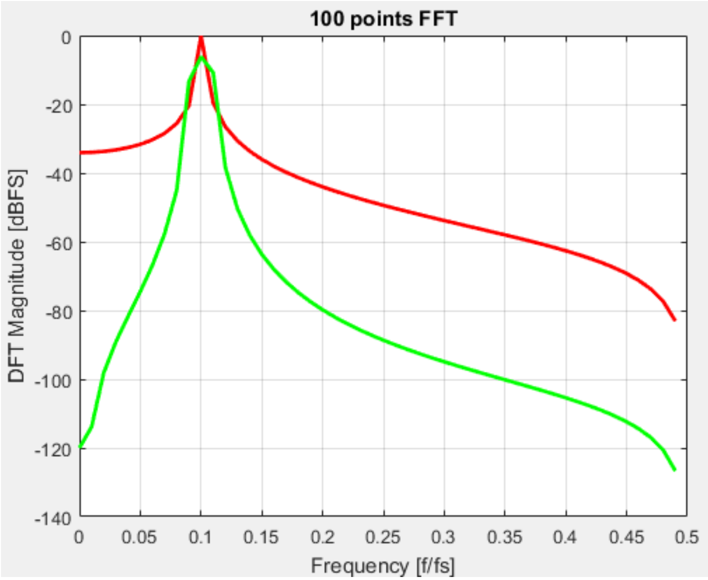
- Spectral leakage can be attenuated by windowing the time samples prior to the DFT
- Windows taper smoothly down to zero at the beginning and the end of the observation window
- Time domain samples are multiplied by window coefficients on a sample-by-sample basis
  - Means convolution in frequency
  - Sine wave tone and other spectral components smear out over several bins
- Lots of window functions to chose from
  - Tradeoff: attenuation versus smearing
- Example: Hann Window

Windowing

```
clear all; clc; close all;
N = 100;
fs = 1000;
fx = 101;
A=1;

x = A*cos(2*pi*fx/fs*[0:N-1]);
s = abs(fft(x));
s = 20*log10(2*s/N/A); % dB relative to full-scale
s = s(1:end/2);
f = [0:N/2-1]/N; % frequency vector
plot(f, s, 'r', 'linewidth', 2);
grid on; hold on;
w = hann(length(x));
x1 = x'.*w;
s1 = abs(fft(x1));
s1 = 20*log10(2*s1/N/A); % dB relative to full-scale
s1 = s1(1:end/2);
plot(f, s1, 'g', 'linewidth', 2);

xlabel('Frequency [f/fs]');
ylabel('DFT Magnitude [dBFS]');
title(strcat(num2str(N),' points FFT'));
```



fft_p54.m

HW#1 MATLAB Code

```
% i_10bit_ADC_tbl simulation
clear all; clc; close all;
signal = load('64pt_data1.txt');     % load the data samples
N = length(signal);                  % N-point FFT
fs = 100e6;                          % Sampling frequency
% fx = fs*(cycles/N)
cycles = 31;                         % Signal tone_bin
FS = 1                               % Full scale range
% prettyFFT(signal);

s = abs(fft(signal));
s = s/N*2;
f = [0:N-1];                         % full frequency vector

sigbin = cycles + 1;
noise = [s(2:sigbin-1), s(sigbin+1:end/2)]; % remove DC component

s = s(1:end/2);                      % remove redundant half of spectrum
f = [0:N/2-1];                       % frequency vector
noise = [s(2:sigbin-1), s(sigbin+1:end)]; % remove DC component

snr = 10*log10( s(sigbin)^2/sum(noise.^2) );
s = 20*log10(s/FS);                  % dB relative to full-scale

figure;plot(f, s, 'linewidth', 2);
% xlabel('Frequency [f/fs]');
xlabel('Frequency [bin]');
ylabel('DFT Magnitude [dBFS]');
grid on;
axis tight;
title(strcat('Ideal 10-bit ADC, SNR=',num2str(snr,4),' dB'))
```

fft_HW1.m