Analysis of Coherent Sampling, FFT, ZOH and Aliasing

Supriya Anand Mishra
Roll No.: **24b1312**
B.Tech, Electrical Engineering
Indian Institute of Technology Bombay

Course: EE 719 - Mixed Signal VLSI Design
Instructor: Prof. Maryam Shojaei Baghini
Semester: Spring Semester

Given Parameters (Common to Q1, Q2, Q3)

- FFT length: $N = 4096$
- Sampling frequency: $f_s = 64$ MHz
- Signal frequency: f_{in}

Question 1

In a sampling case,

$$\frac{f_s}{f_{in}} = 3.2 = \frac{16}{5}$$

This ratio is rational, hence an integer number of signal cycles can be captured within a finite number of samples. However, since the numerator and denominator share common factors, the sampled sequence repeats every 16 samples.

Observation:

- Early repetition of the sample sequence
- Severe redundancy in the FFT window
- Poor spectral behavior in practice

Conclusion: Although the ratio is rational, the sampling is not practically coherent due to excessive redundancy.

Question 2

(a) Selection of Input Frequency for Coherent Sampling

FFT bin spacing:

$$\Delta f = \frac{f_s}{N} = \frac{64 \times 10^6}{4096} = 15.625 \text{ kHz}$$

Choosing an integer FFT bin:

$$k = 960$$

$$\boxed{f_{in} = \frac{k}{N} f_s = 15 \text{ MHz}}$$

This satisfies 10 MHz $< f_{in} <$ 20 MHz and ensures coherent sampling.

(b) Length of Sampling Window

$$T = \frac{N}{f_s} = \frac{4096}{64 \times 10^6} = 64 \ \mu s$$

(c) FFT of Coherently Sampled Signal

The signal is generated in bin-aligned form:

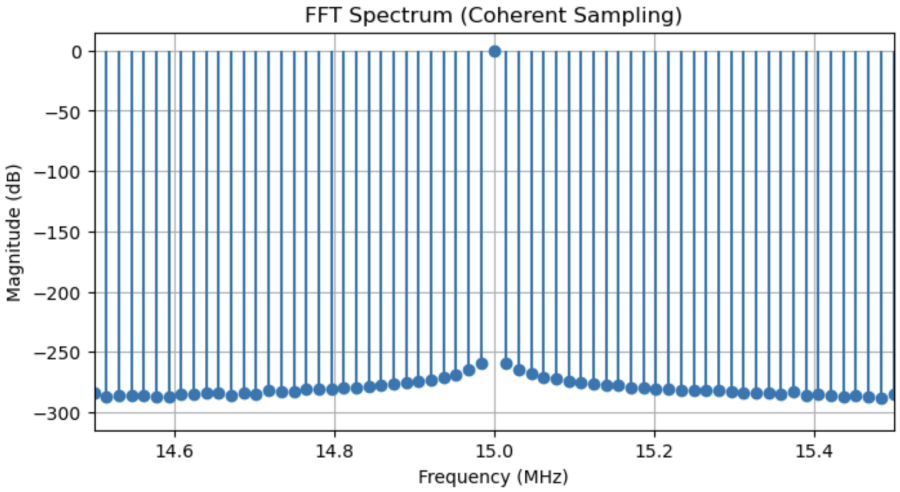
$$x[n] = \cos\left(2\pi \frac{k}{N} n\right)$$

Python Code

```
import numpy as np
import matplotlib.pyplot as plt

fs = 64e6
N = 4096
k = 960

n = np.arange(N)
x = np.cos(2*np.pi*k*n/N)

X = np.fft.fft(x)
X_mag = np.abs(X)/(N/2)

f = np.arange(N//2)*fs/N
X_mag = X_mag[:N//2]
X_dB = 20*np.log10(X_mag + 1e-15)

plt.figure(figsize=(8,4))
plt.stem(f/1e6, X_dB, basefmt=" ")
plt.xlabel("Frequency (MHz)")
plt.ylabel("Magnitude (dB)")
```

```
plt.title("FFT Spectrum (Coherent Sampling)")
plt.grid(True)
plt.xlim(14.5,15.5)
plt.show()
```

Plot




```
Peak bin index: 960
Peak frequency (MHz): 15.0
```

Figure 1: FFT spectrum showing single-bin energy (coherent sampling)

Observation: All energy lies in a single FFT bin, confirming coherent sampling without spectral leakage.

Question 3

(a) ZOH Sampled Signal

A more general coherent signal is chosen:

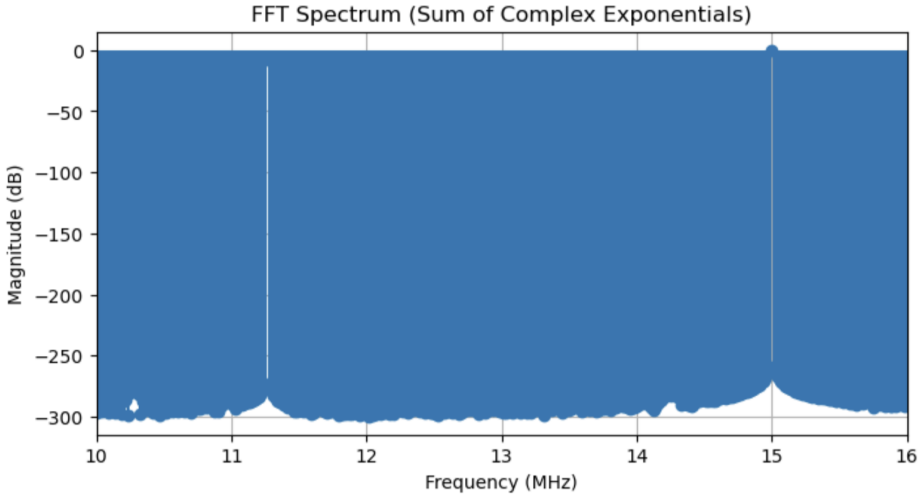
$$x[n] = e^{j2\pi \frac{960}{4096} n} + 0.4\, e^{j2\pi \frac{721}{4096} n}$$

Python Code (ZOH Time Domain)

```
k1 = 960
k2 = 721

x = np.exp(1j*2*np.pi*k1*n/N) + 0.4*np.exp(1j*2*np.pi*k2*n/N)

L = 8
zoh = np.repeat(x, L)
fs_zoh = fs * L

```

```
t_zoh = np.arange(len(zoh))/fs_zoh

plt.figure(figsize=(8,4))
plt.plot(t_zoh[:400]*1e6, np.real(zoh[:400]))
plt.xlabel("Time ( s )")
plt.ylabel("Amplitude")
plt.title("ZOH Signal (Real Part)")
plt.grid(True)
plt.show()
```

Plot




```
Peak bins: [962 959 961 721 960]
```

Figure 2: Zero-Order Hold signal in time domain

(b) Fourier Transform of ZOH Signal

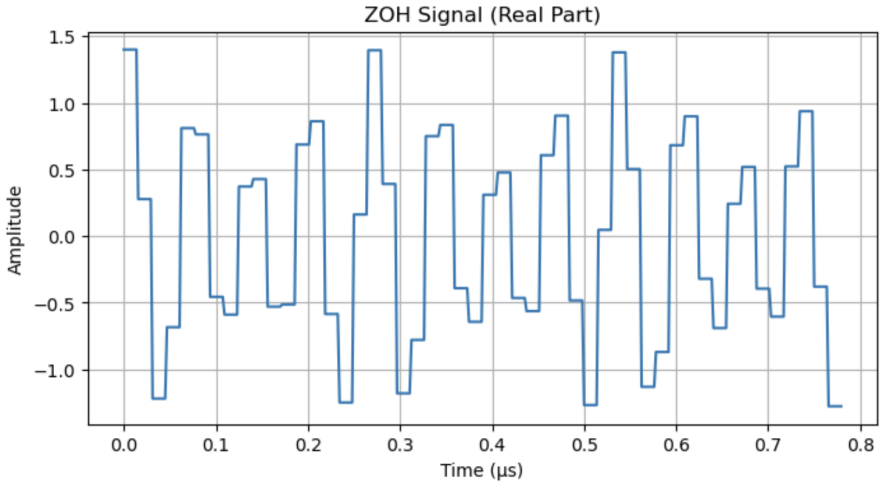
The ZOH frequency response is:

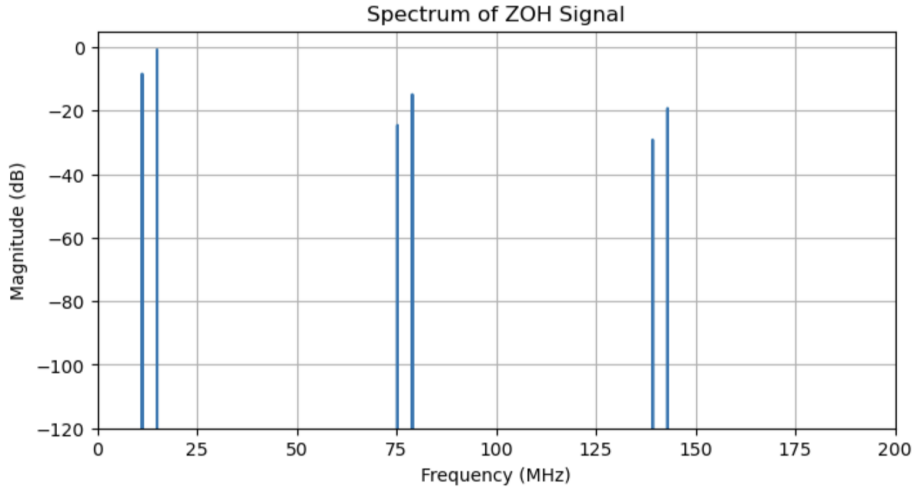
$$H_{\text{ZOH}}(f) = T_s \,\text{sinc}(fT_s)\, e^{-j\pi fT_s}$$

Python Code (ZOH Spectrum)

```
Z = np.fft.fft(zoh)
Z_mag = np.abs(Z)/len(zoh)

f_zoh = np.arange(len(zoh))*fs_zoh/len(zoh)
Z_dB = 20*np.log10(Z_mag + 1e-15)

plt.figure(figsize=(8,4))
plt.plot(f_zoh/1e6, Z_dB)
plt.xlabel("Frequency (MHz)")
```

```
plt.ylabel("Magnitude (dB)")
plt.title("Spectrum of ZOH Signal")
plt.grid(True)
plt.xlim(0,200)
plt.ylim(-120,5)
plt.show()
```





(c) Comparison

The FFT of discrete-time samples shows ideal discrete spectral lines, whereas the ZOH spectrum contains spectral images and sinc-shaped attenuation due to the hold operation.

Question 4

Given:

$$x(t) = 10\cos(20000\pi t) + 20\sin(30000\pi t)$$

(a) Nyquist Sampling Rate

$$f_1 = 10 \text{ kHz}, \quad f_2 = 15 \text{ kHz}$$

$$\boxed{f_{s,\text{Nyquist}} = 30 \text{ kHz}}$$

(b) Impulse-Train Sampling

Sampling frequency is chosen such that:

$$16 \text{ kHz} \leq f_s \leq 24 \text{ kHz}$$

which is below the Nyquist rate.

(i) Frequency Components at Sampler Output

Spectral replicas appear at:

$$|kf_s \pm 10 \text{ kHz}|, \quad |kf_s \pm 15 \text{ kHz}|$$

causing overlap and aliasing.

Python Code (Aliasing Demonstration)

```
fs_alias = 18e3
t = np.arange(0,5e-3,1/fs_alias)

x = 10*np.cos(2*np.pi*1e4*t) + 20*np.sin(2*np.pi*1.5e4*t)

X = np.fft.fft(x)
f = np.arange(len(X))*fs_alias/len(X)

plt.figure(figsize=(7,4))
plt.plot(f/1e3, np.abs(X))
plt.xlabel("Frequency␣(kHz)")
plt.ylabel("Magnitude")
plt.title("Aliased␣Spectrum␣(Undersampling)")
plt.grid(True)
plt.show()
```

Plot

(ii) Band-Pass Filter (7–18 kHz)

Both original and aliased components lying within the passband appear at the filter output.

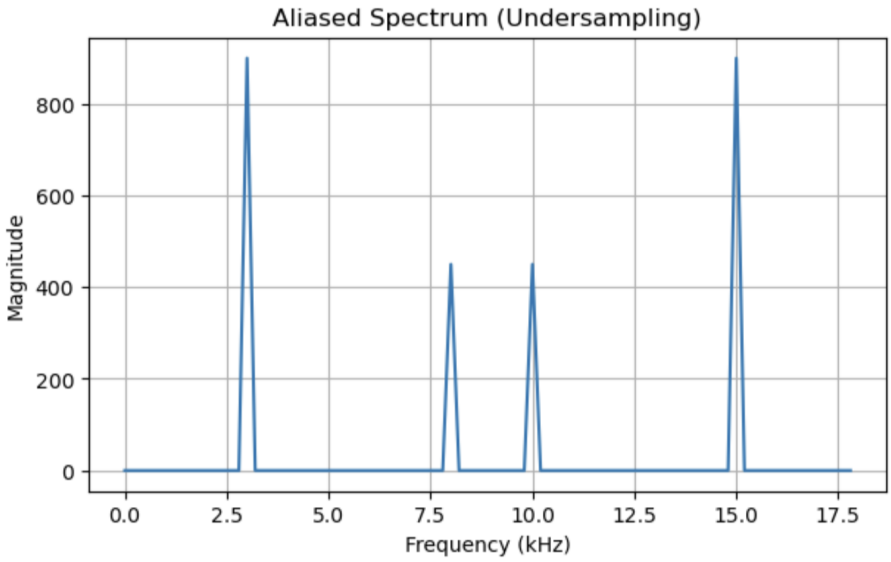


Figure 3: Aliased spectrum due to undersampling

(iii) Low-Pass Filter (Cutoff = 18 kHz)

Aliased components below the cutoff frequency pass through the filter. Once aliasing occurs, it cannot be removed by filtering.

Conclusion

This report demonstrates coherent sampling, FFT analysis, the effects of Zero-Order Hold, and irreversible aliasing using analytical reasoning and numerical simulations.