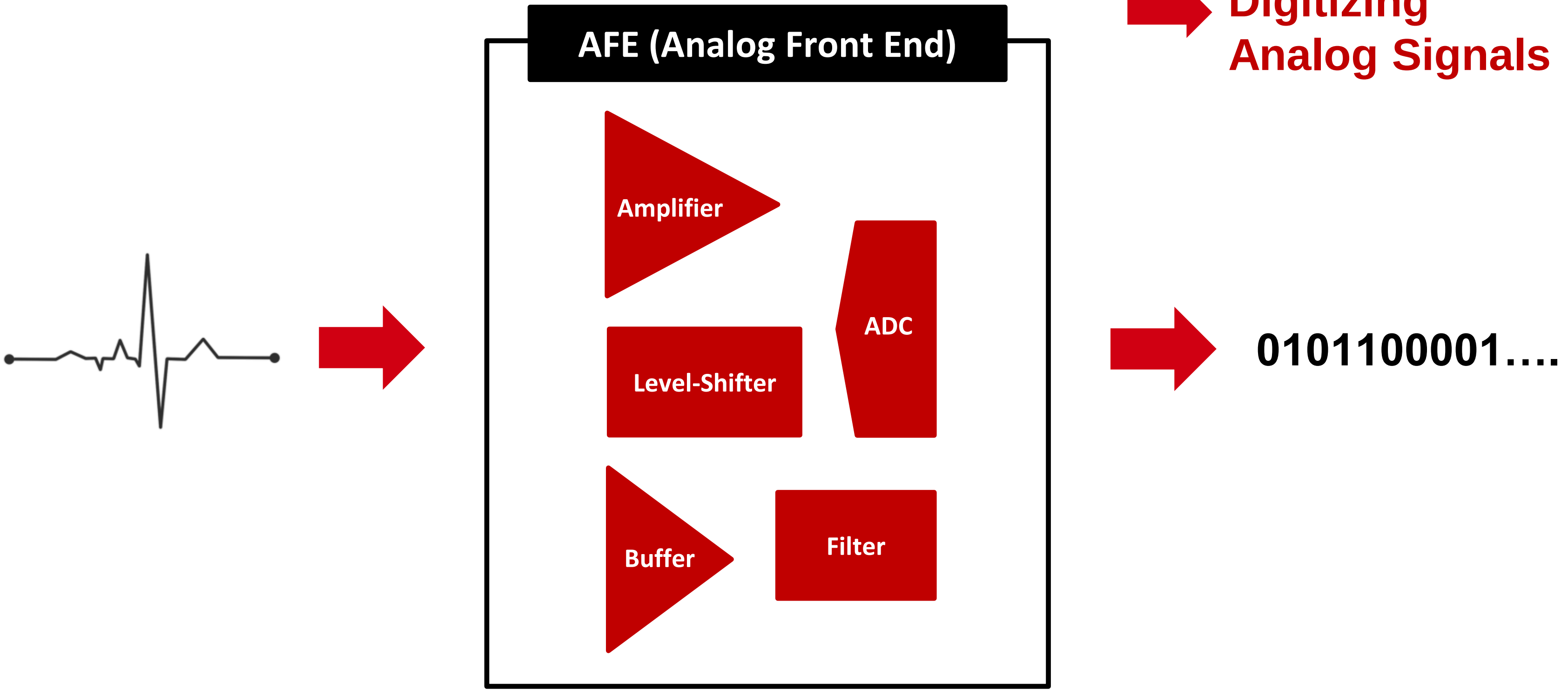
Chapter-4
Analog Blocks

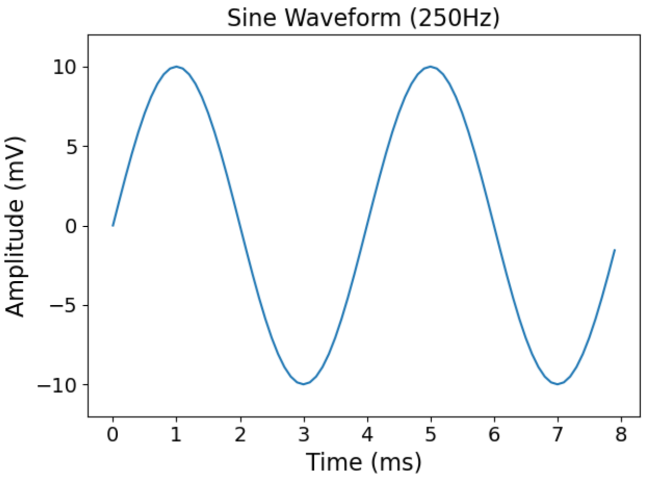
ADC, Amplifiers, Filters and Level-Shifters

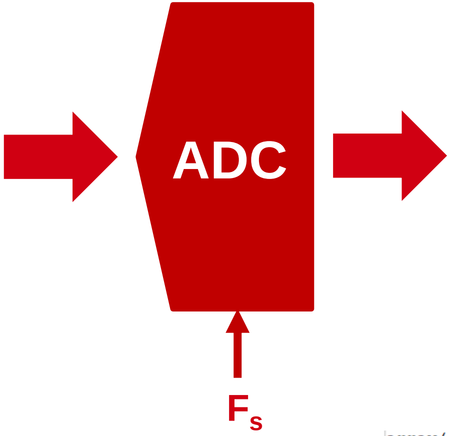
Analog Frontend

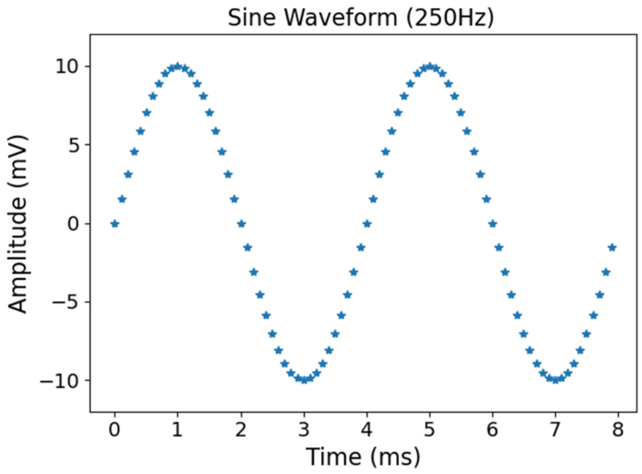
Digitizing
Analog Signals



Signal Sampling



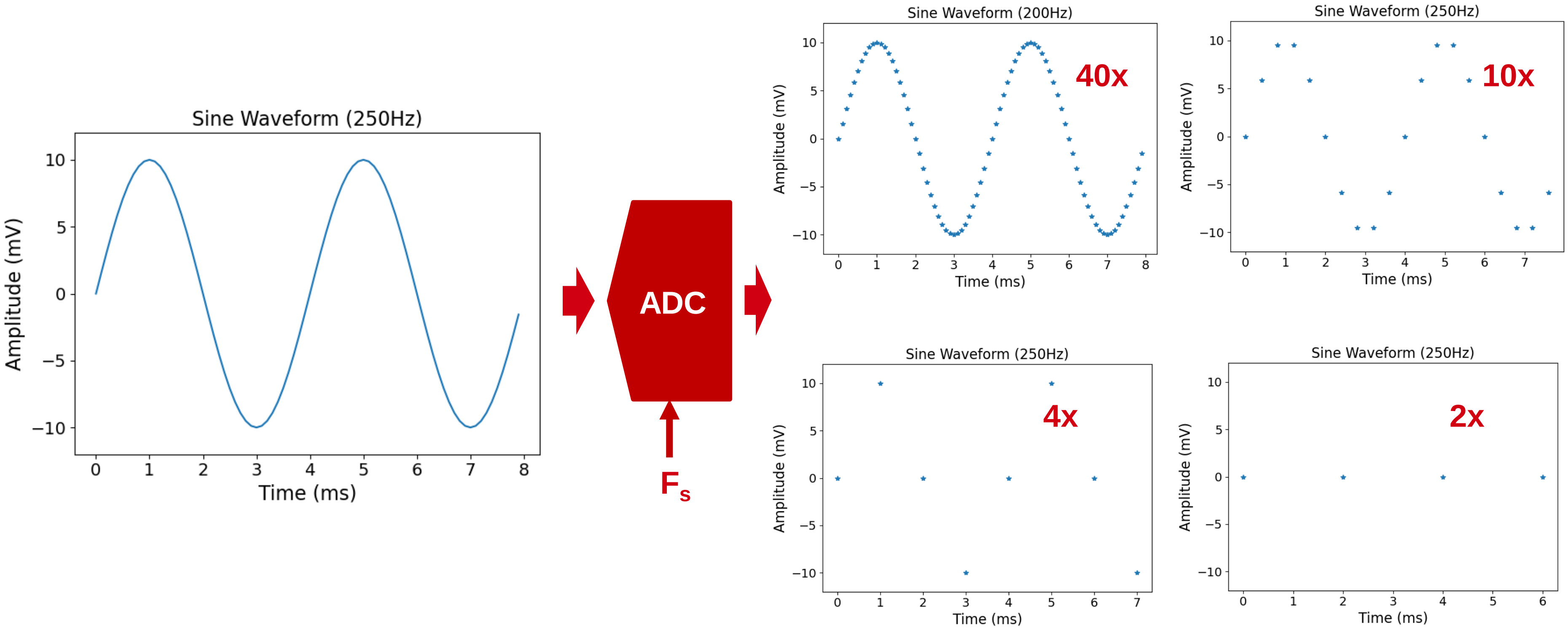




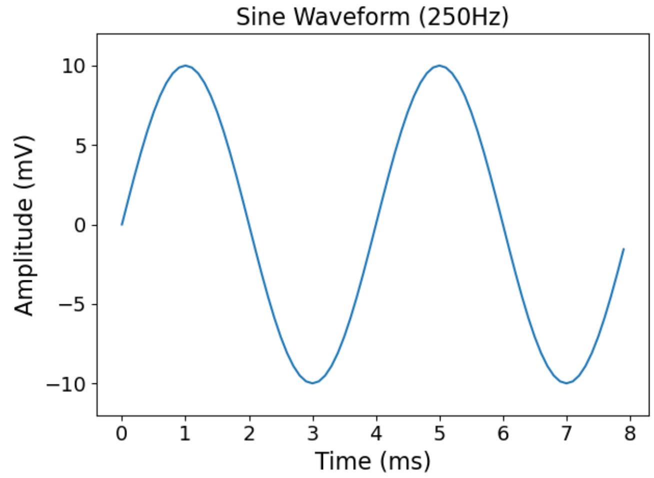

```
array([ 0.00000000e+00,  1.56434465e+00,  3.09016994e+00,  4.53990500e+00,
        5.87785252e+00,  7.07106781e+00,  8.09016994e+00,  8.91006524e+00,
        9.51056516e+00,  9.87688341e+00,  1.00000000e+01,  9.87688341e+00,
        9.51056516e+00,  8.91006524e+00,  8.09016994e+00,  7.07106781e+00,
        5.87785252e+00,  4.53990500e+00,  3.09016994e+00,  1.56434465e+00,
        1.22464680e-15, -1.56434465e+00, -3.09016994e+00, -4.53990500e+00,
       -5.87785252e+00, -7.07106781e+00, -8.09016994e+00, -8.91006524e+00,
       -9.51056516e+00, -9.87688341e+00, -1.00000000e+01, -9.87688341e+00,
       -9.51056516e+00, -8.91006524e+00, -8.09016994e+00, -7.07106781e+00,
       -5.87785252e+00, -4.53990500e+00, -3.09016994e+00, -1.56434465e+00,
       -2.44929360e-15,  1.56434465e+00,  3.09016994e+00,  4.53990500e+00,
        5.87785252e+00,  7.07106781e+00,  8.09016994e+00,  8.91006524e+00,
        9.51056516e+00,  9.87688341e+00,  1.00000000e+01,  9.87688341e+00,
        9.51056516e+00,  8.91006524e+00,  8.09016994e+00,  7.07106781e+00,
        5.87785252e+00,  ... ])
```

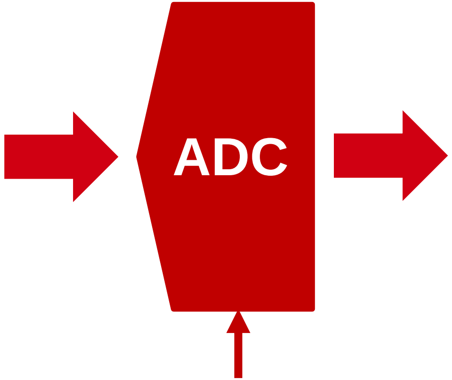
What should be the sampling frequency F_s ?

Sampling Freq

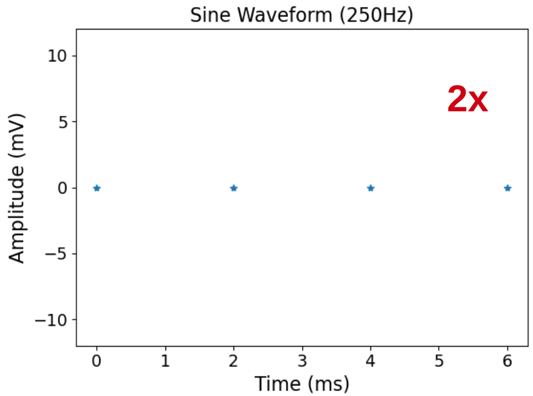


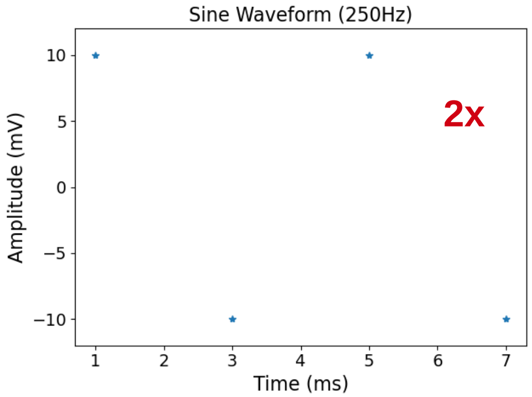
Required min(Fs)





F_s = 2x = 500Hz

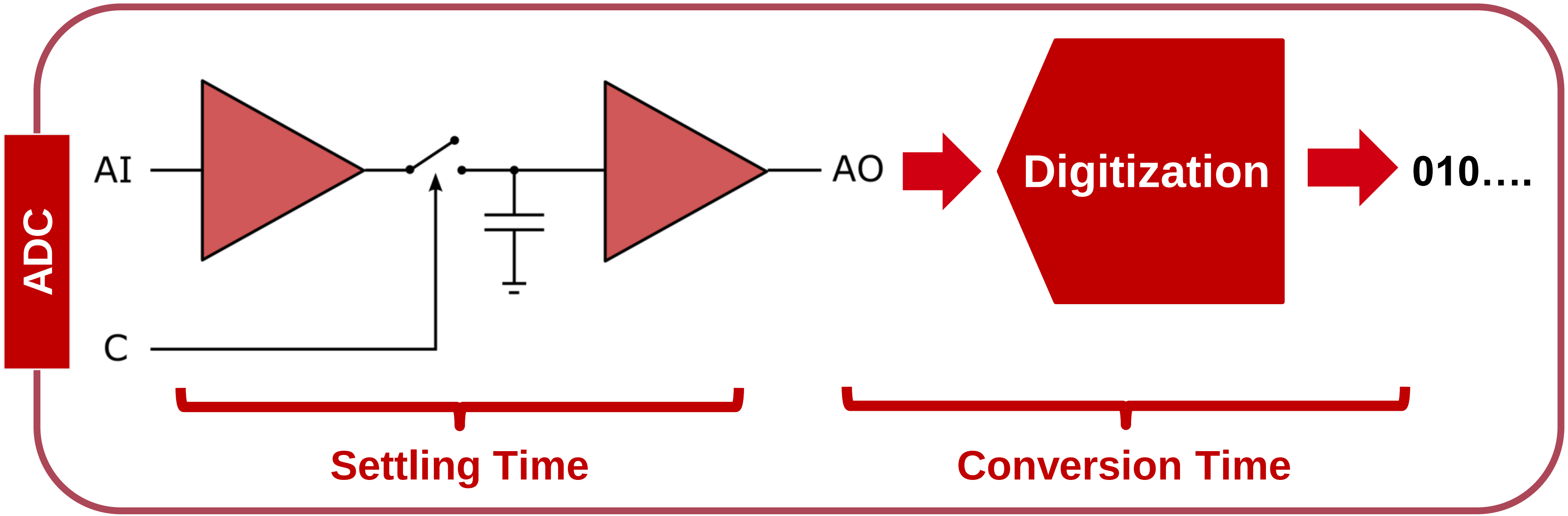




- **Required min(F_s) > 2x the Signal Frequency**
- **In Practice, F_s is selected >> 2x e.g., 10x**

Question – Why 10x, Why not 100x or 1000x ?

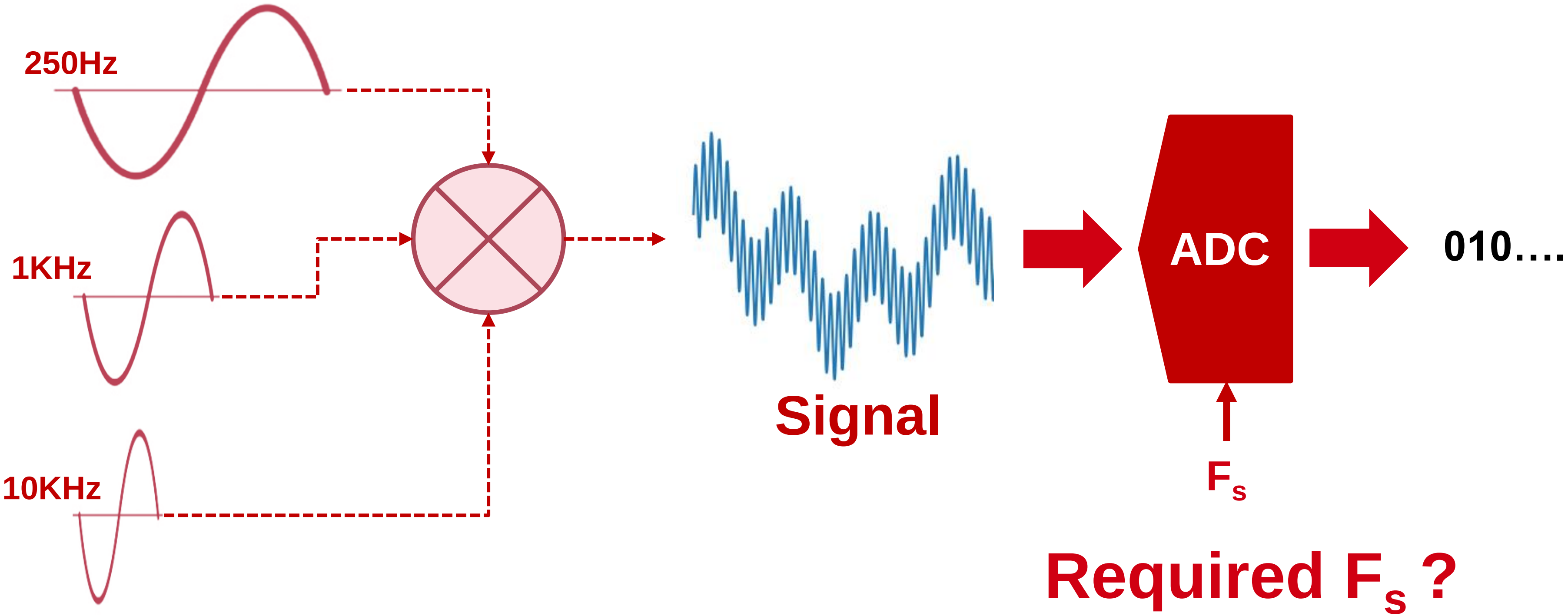
Max (Fs) of an ADC



A simplified **sample and hold circuit** diagram. AI is an analog input, AO — an analog output, C — a control signal. Image Courtesy - Wikipedia

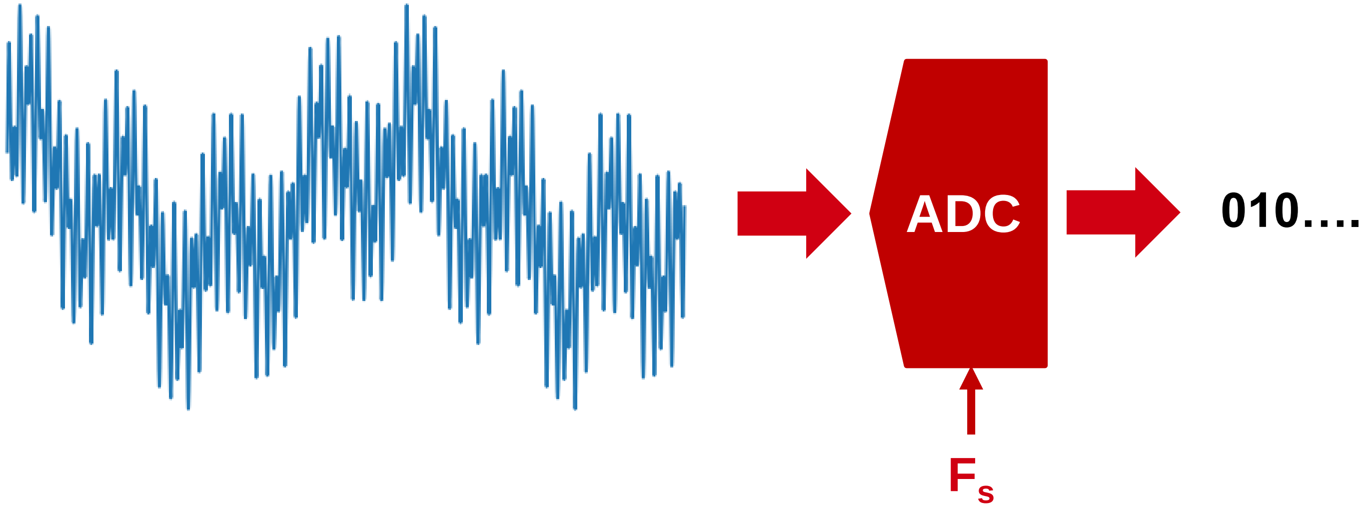
Conversion time is dictated by the architecture of the ADC.

Question #1



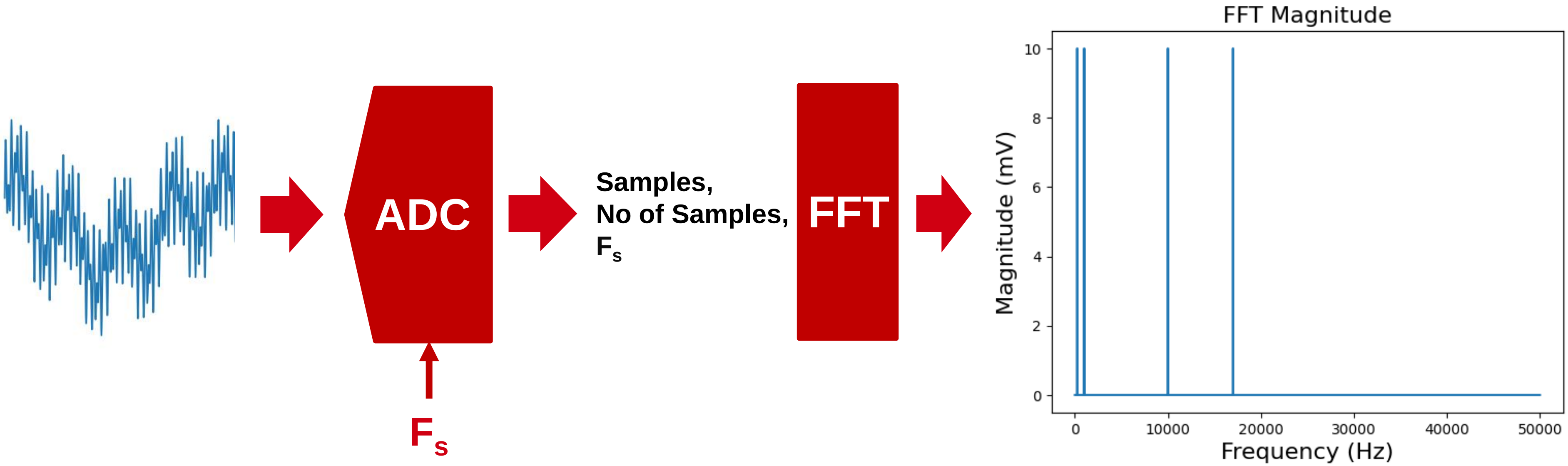
Required F_s ?

Question #2



Required F_s > 2x $\underbrace{\text{Highest Frequency Component in the Signal}}_{\text{How to Find ?}}$

Fast Fourier Transform

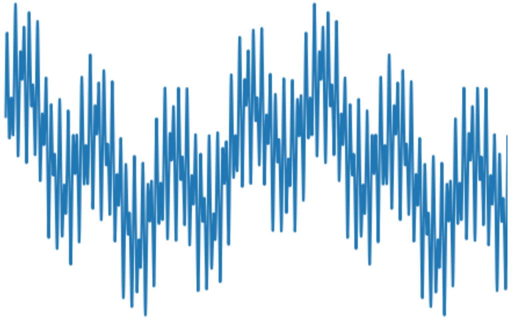


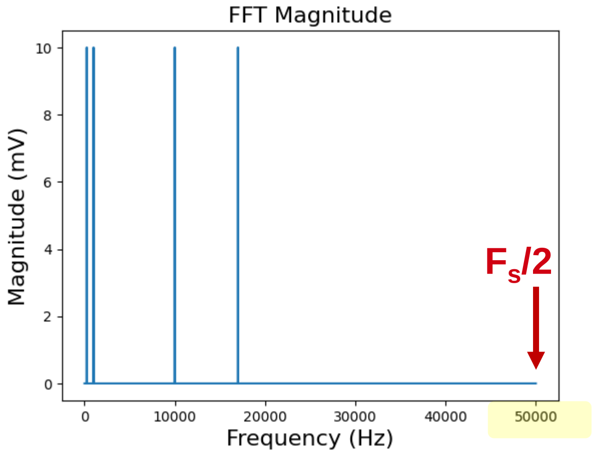
Note - Signal contains sine waves of amplitude 10mV and frequencies of 250Hz, 1KHz, 10KHz and 17KHz

Code Example

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

### Step-1: Creating Signal by Combining Sine Wave of 250Hz, 1KHz, 10KHz and 17KHz

# Define the time axis
fs = 100e3  # sampling frequency
Tstop = 1 # stop time
t = np.arange(0, 2*Tstop, 1/fs) * 1000

# Generate the sine wave
sine_wave = 10*np.sin(2*np.pi*250*t/1000) + \
    10*np.sin(2*np.pi*1000*t/1000) + \
        10*np.sin(2*np.pi*10000*t/1000) + \
            10*np.sin(2*np.pi*17000*t/1000)


### Step-2: Generate FFT and Plot its Magnitude

X = np.fft.fft(sine_wave)

# Calculate the frequency vector
freq = np.fft.fftfreq(len(X), 1/fs)

# Select only positive frequencies
pos_freq = freq[:len(freq)//2]
pos_X = 2*np.abs(X)[:len(X)//2]/len(X)

# Plot the magnitude of the FFT
plt.plot(pos_freq, pos_X)
plt.xlabel('Frequency (Hz)', fontsize=16)
plt.ylabel('Magnitude (mV)', fontsize=16)
plt.title('FFT Magnitude', fontsize=16)

plt.show()
```

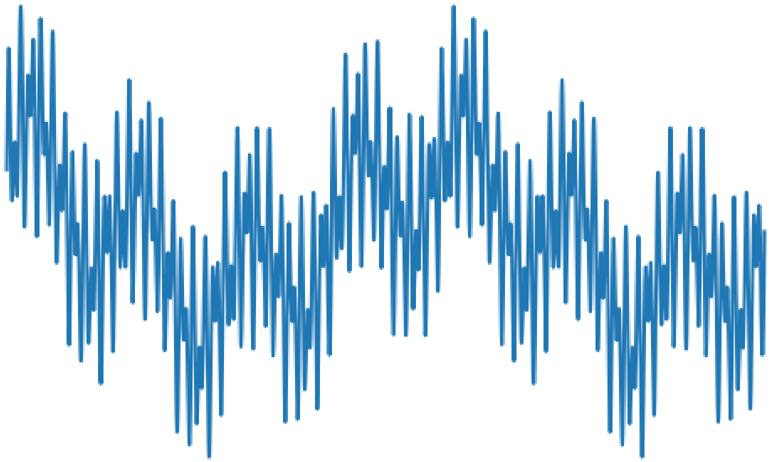
F_s for FFT ?



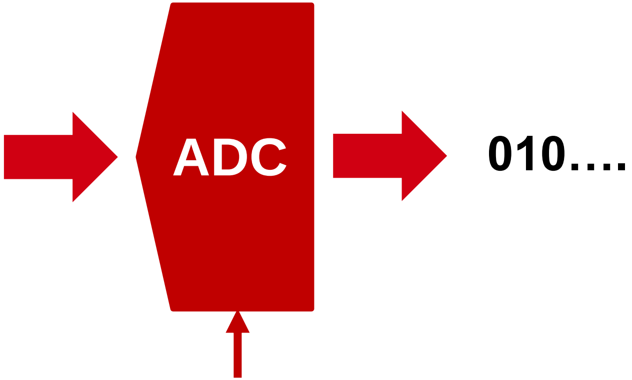


Prof. Kurian Polachan, IIIT-Bangalore

Answer to Question #2



Signal contains sine waves of amplitude 10mV and frequencies of 250Hz, 1KHz, 10KHz and 17KHz



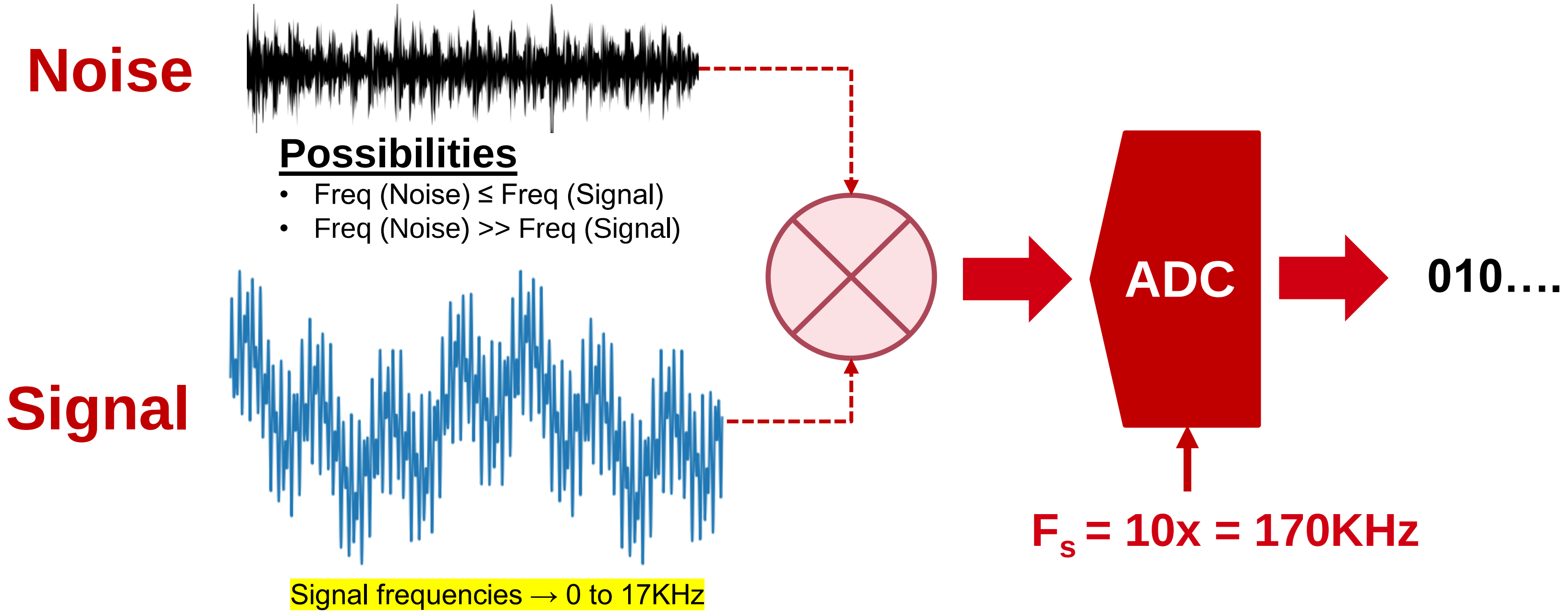
F_s = 10x = 10x17KHz = 170KHz

Summary

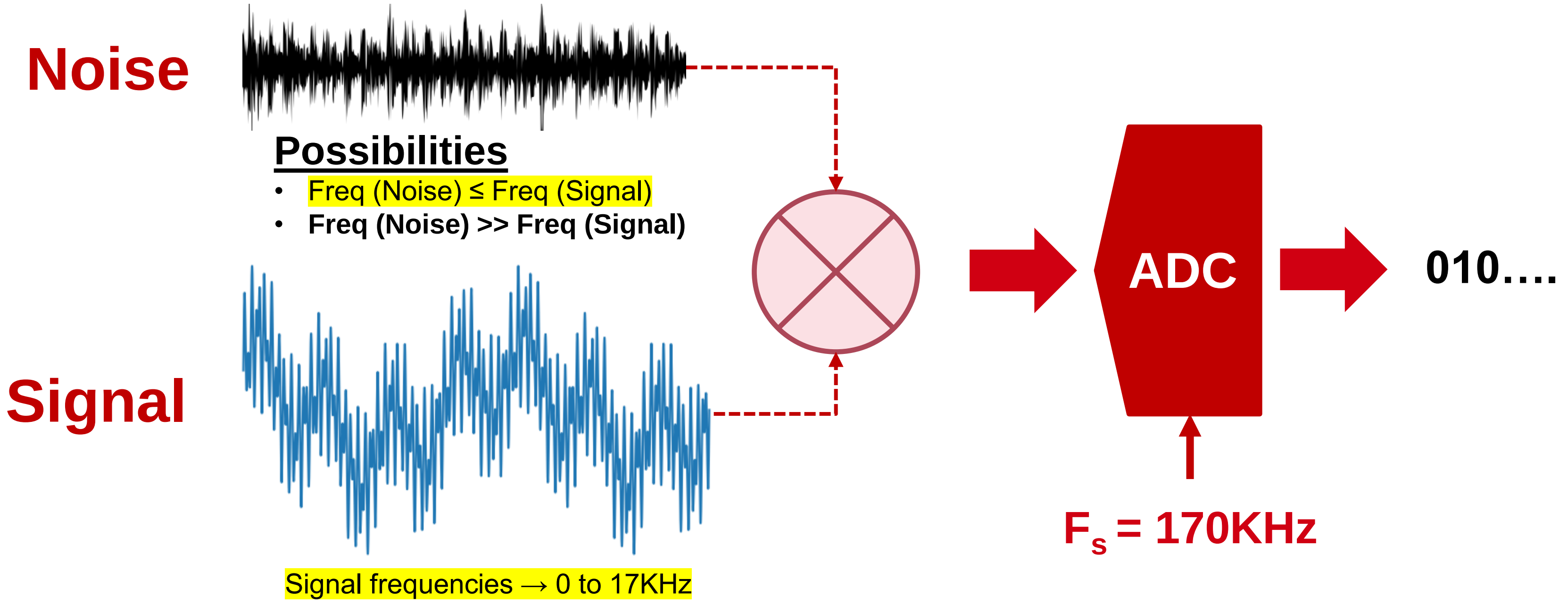
- For digitizing a signal, always use an ADC with sampling frequency, F_s > 2x the highest frequency component in the signal
- If the highest frequency component in the signal is not known, use FFT
- F_s of the FFT should be > 2x for the estimated highest frequency component of the signal.

Note – In practice, fix F_s >> 2x (e.g., 10x)

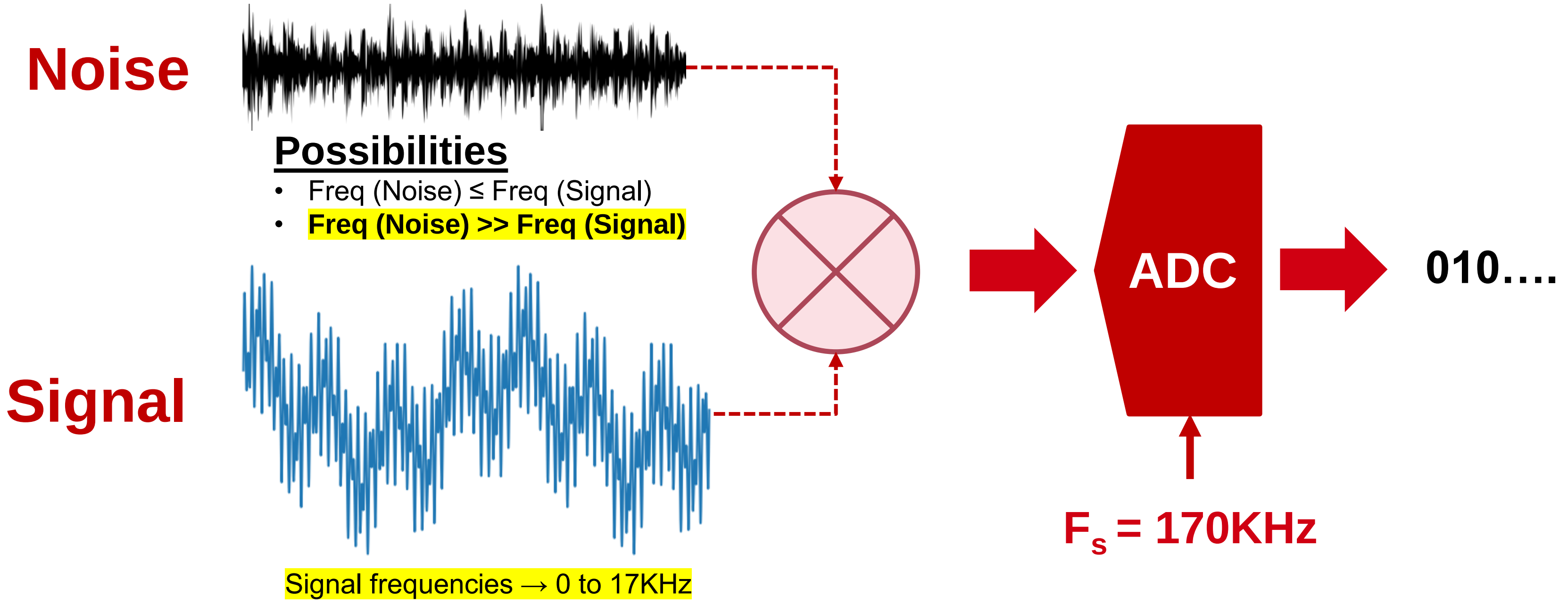
Effect of Noise ?



Case-1: $F_{Noise} \sim F_{Signal}$

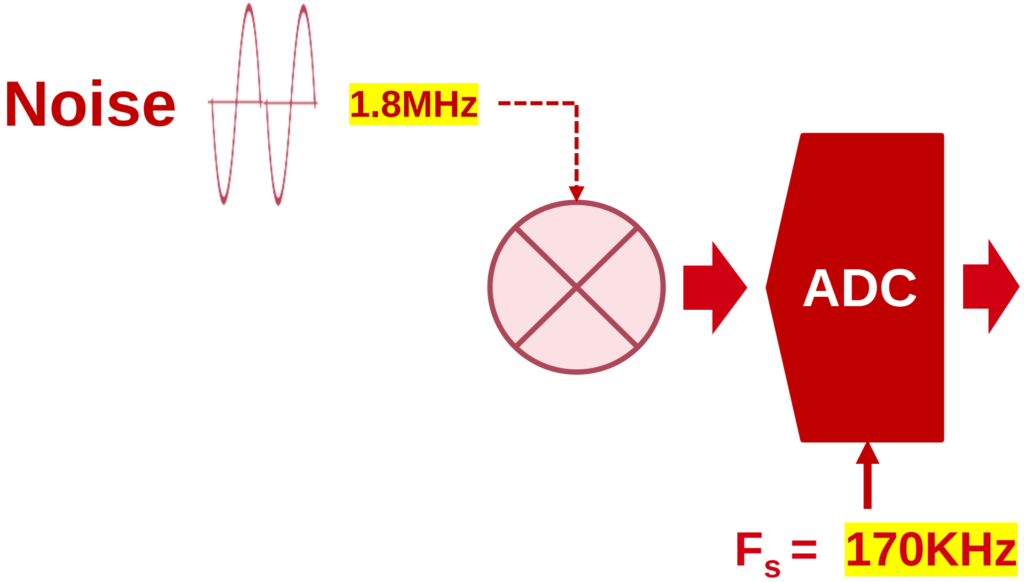


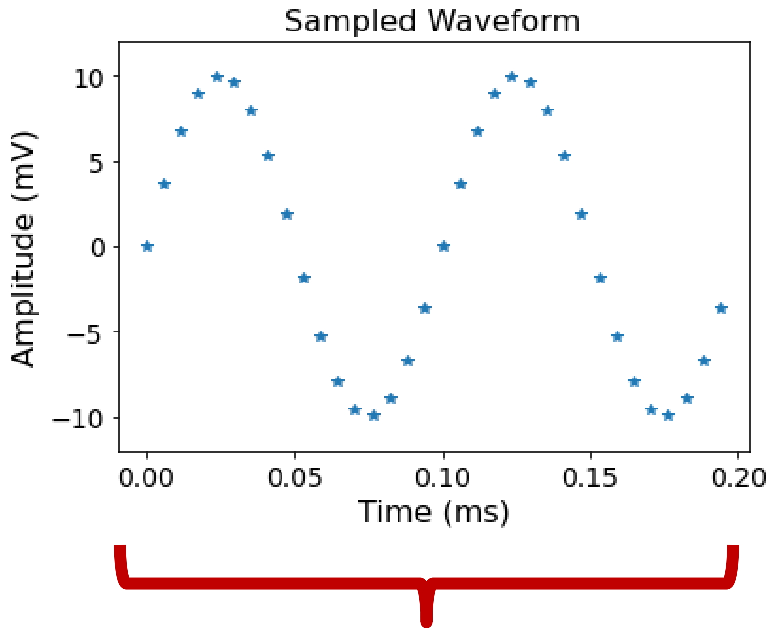
Case-2: $F_{Noise} >> F_{Signal}$



Possibilities
- Freq (Noise) ≤ Freq (Signal)
- **Freq (Noise) >> Freq (Signal)**

Case-2: Example



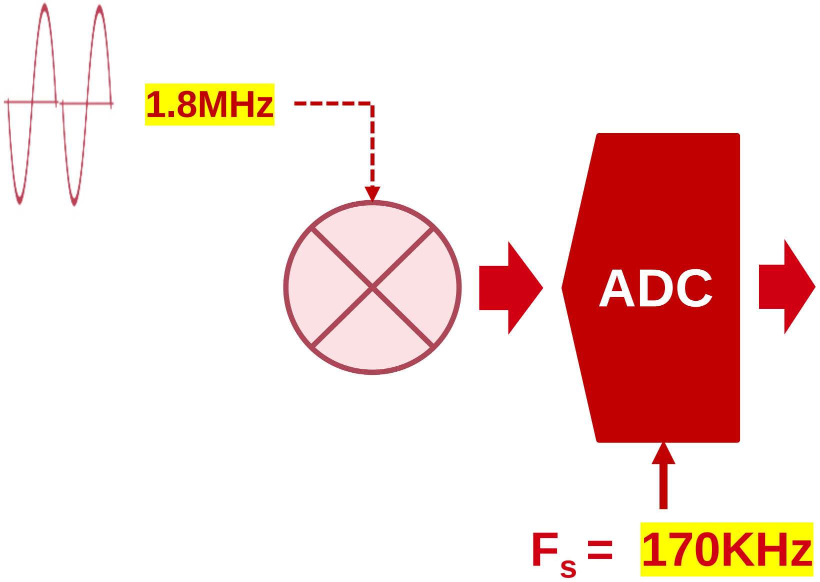


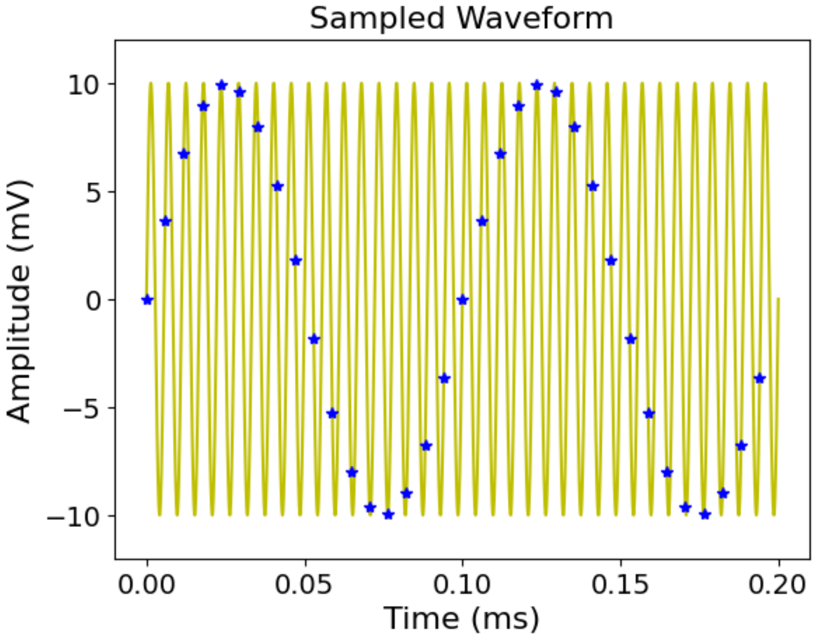
Signal Freq = 0-17KHz

Noise Appears as Signal

Case-2: Example

Noise

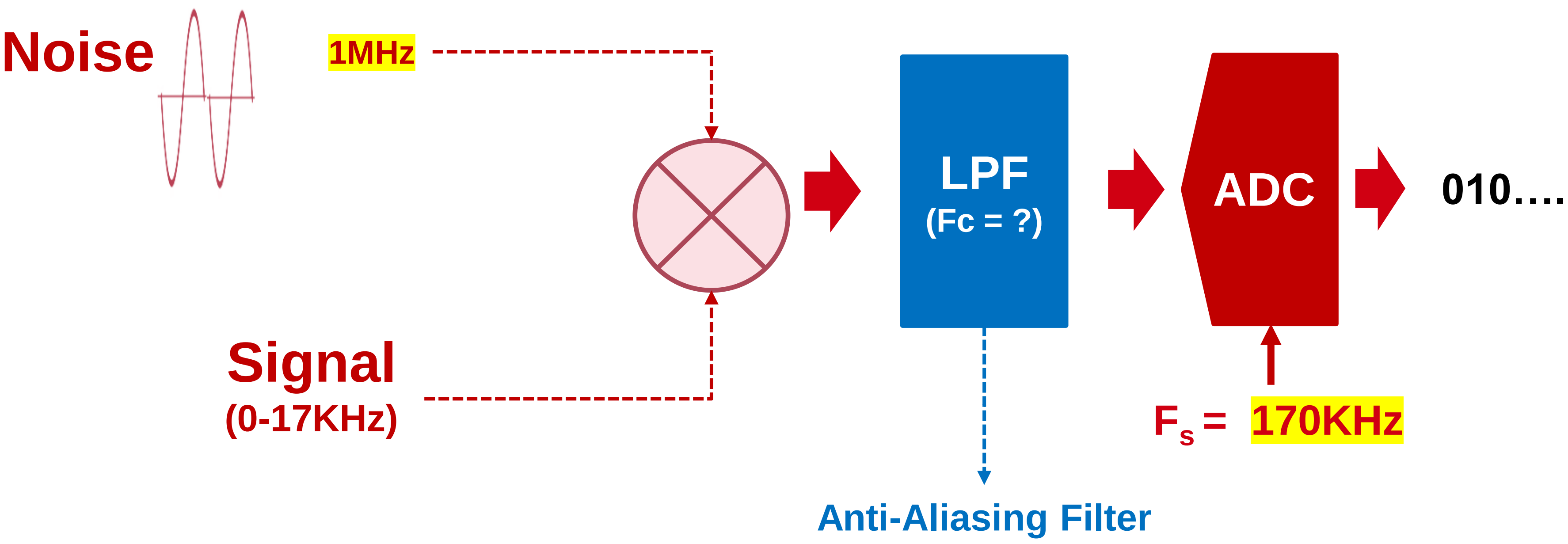




Signal Freq = 0-17KHz

Noise Appears as Signal

Sol. Antialiasing Filter



Filtering In-Band Noise

- **In-Band Noise: Any noise having frequency components that lies in the signal frequency range.**

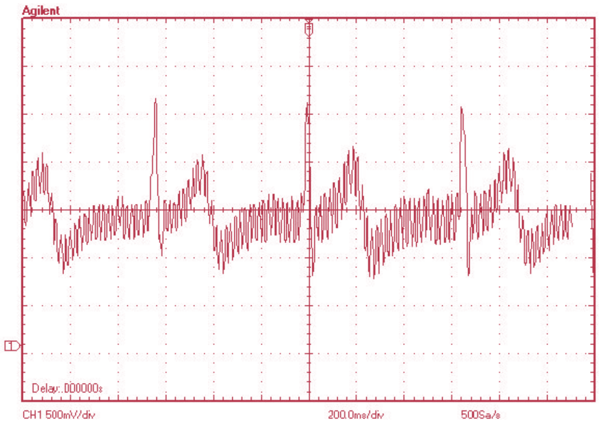
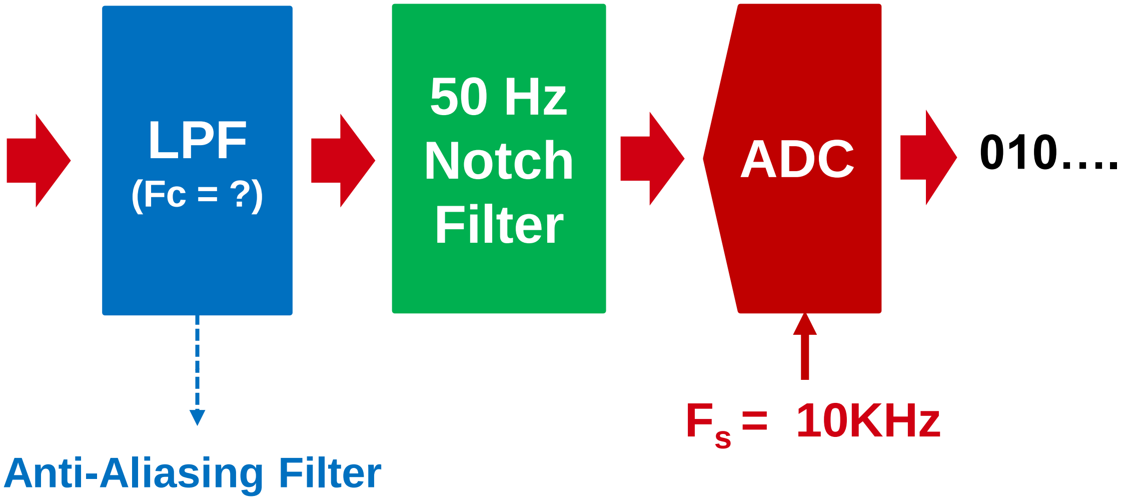
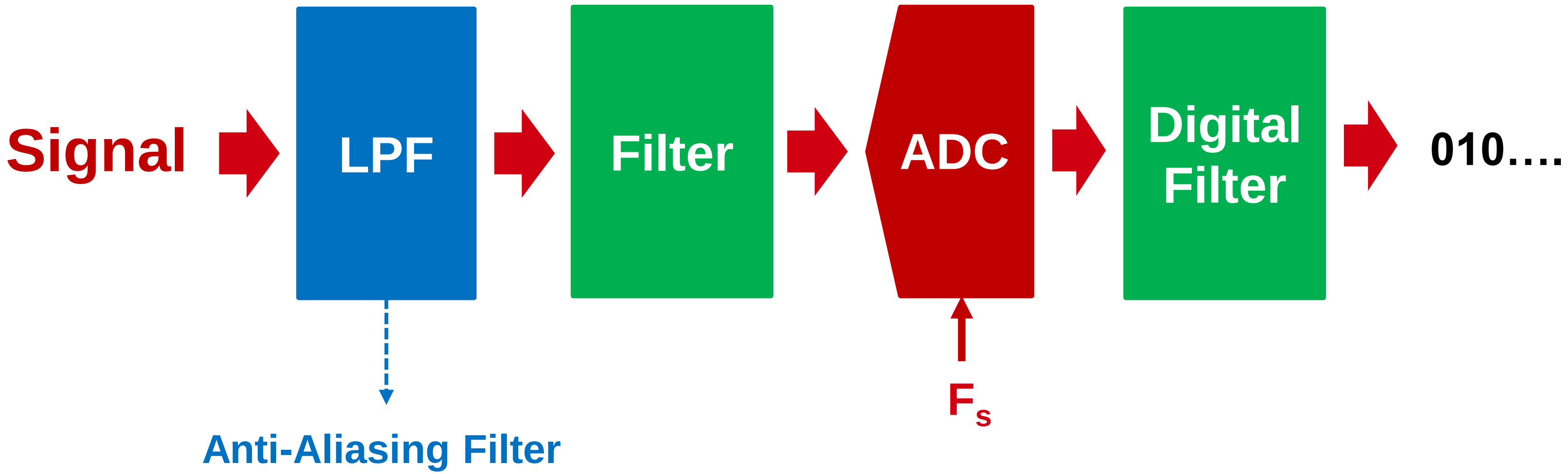


Image Courtesy - Gogoi, Mukti Nath, and Riku Chutia. "A Battery Operated Portable Wireless Electrocardiograph: A Bluetooth Wireless and PC Based Design." *International Journal of Engineering Research and Development* 5, no. 7 (2013): 55-60.



Mixed-Signal Filtering

- E.g., Filtering signal (500Hz-to-1KHz) from 50Hz power-line noise and from noise in the 0-to-500Hz and 1KHz-to-F_s/2 frequency bands.



Analog vs. Digital Filters

	Analog Filtering	Firmware Filtering	PSoC Digital Filtering
Method	Utilizing discrete passives (resistors and capacitors)	Utilizing MCU processor and specialized code	Dedicated digital filter co-processor and easy-to-use graphical configuration
Pros	Hardware only – no firmware Supports a wide-range of filter frequencies	Modifying filter design is as simple as updating the firmware Abundant availability of filter code examples	PSoC Creator provides visual indication of filter design and expected performance Simple to change and update Dedicated hardware enables high-performance filtering without impacting embedded MCU functions No firmware to write
Cons	Difficult to design unless you're an analog filter expert Design changes require hardware changes	Difficult to debug the filter code design Utilizes critical CPU resources May require expensive DSP-class MCUs	Digital only solution – does not preserve the original analog signal
Performance	Dependent on the quality of the discrete analog passives	Dependent on MCU performance and available clock cycles	Independent of MCU performance or load

Image Courtesy - PSOC® Digital Filter

Firmware

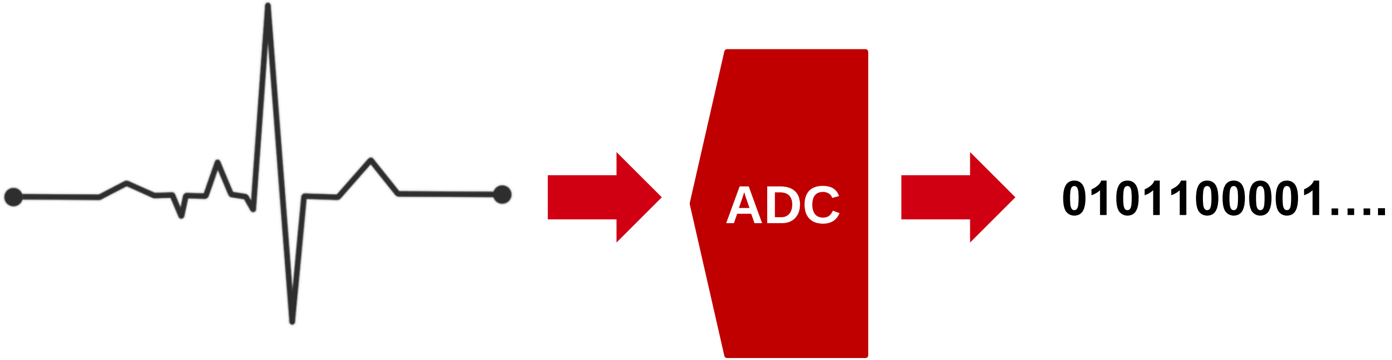
Digital Hardware

E.g., DSP

Summary

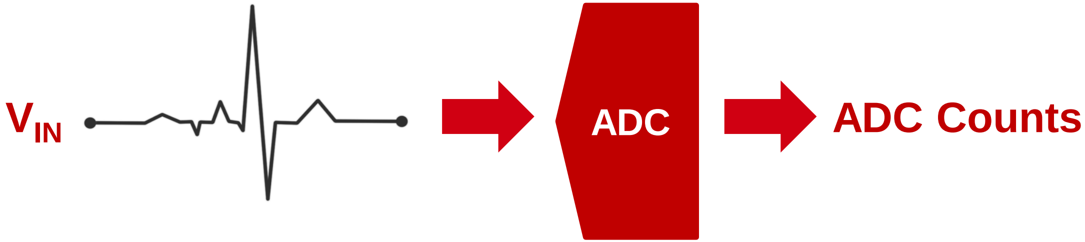
- Aliasing Effect - Frequencies above $\mathbf{F_s}$ /2 after digitization appears as frequencies below $\mathbf{F_s}$ /2
- For digitizing a signal, always use an Anti-Aliasing Filter (an LPF). The LPF should discard frequencies above $\mathbf{F_s}$ /2. The anti-aliasing filter should be implemented in analog domain before ADC. It cannot be implemented in the digital domain.
- Any filter that operates on signals within Fs/2 can be either implemented in analog or digital domain (e.g., filtering signal or in-band noise filtering)
- Digital Filters are of two types – Filters implemented in firmware or implemented using dedicated digital hardware blocks e.g., DSP

ADC Specifications



- **Input Signal Range, Supply Range and Type (unipolar or bipolar),**
- **Resolution (bits), Effective Resolution or ENOB,**
- **Sampling Frequency, Settling Time, Conversion Time**

Counts, Range, N

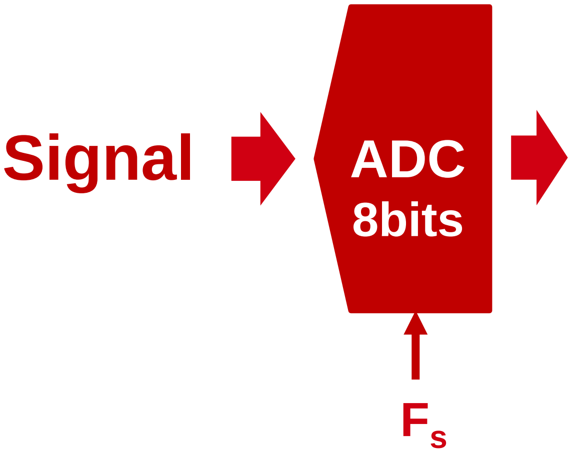


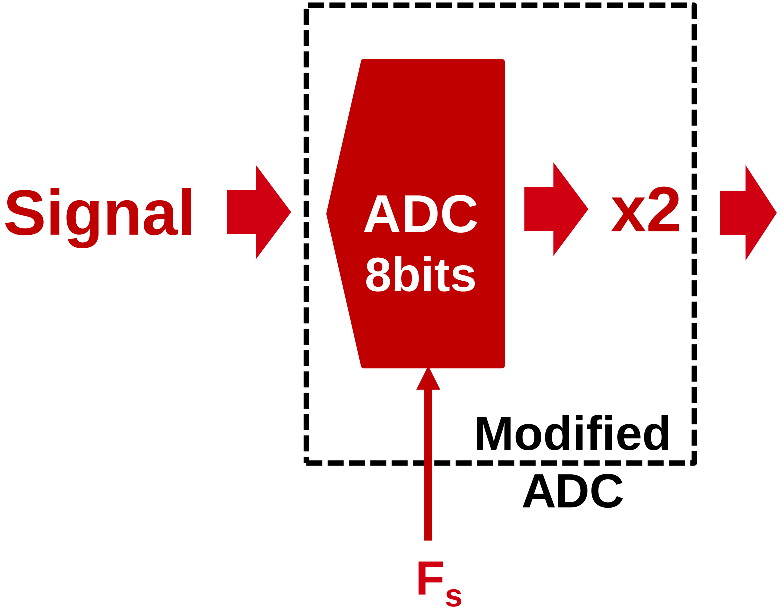
- **Resolution (bits) = N bits (e.g., N = 10 bits)**
- **Input Signal Range of the ADC (typical) = 0 to V_X (e.g., 0 – 1V)**
- **Minimum resolvable input, i.e., Resolution (mV) = V_X / (2^N) (e.g., 1mV)**
- **ADC Counts** $= \begin{cases} V_{IN} \text{ / Resolution (mV); if } 0 < V_{IN} < V_X \\ 2\text{^}N\text{; if } V_{IN} \geq V_X \\ 0\text{; if } V_{IN} \leq 0 \end{cases}$

Questions - ADC

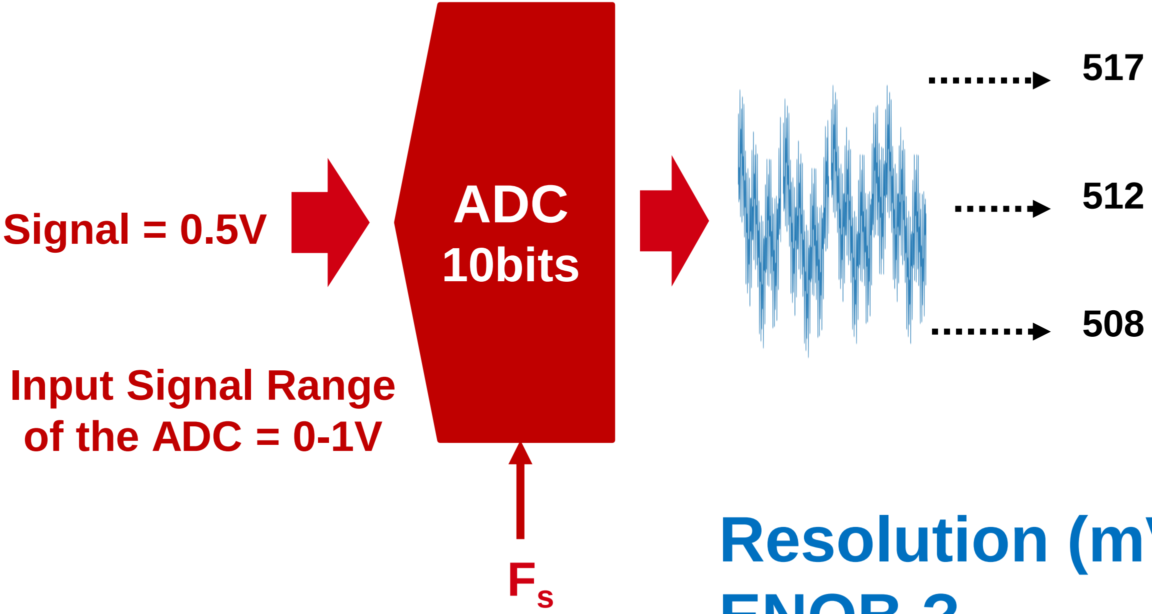
Q1. The input signal range of the ADCs are 0-1V, find the Resolution in mV ?

Q2. How to evaluate the resolution in mV of an ADC with unknown spec ?





Effective Resolution



Resolution (mV) ?

ENOB ?