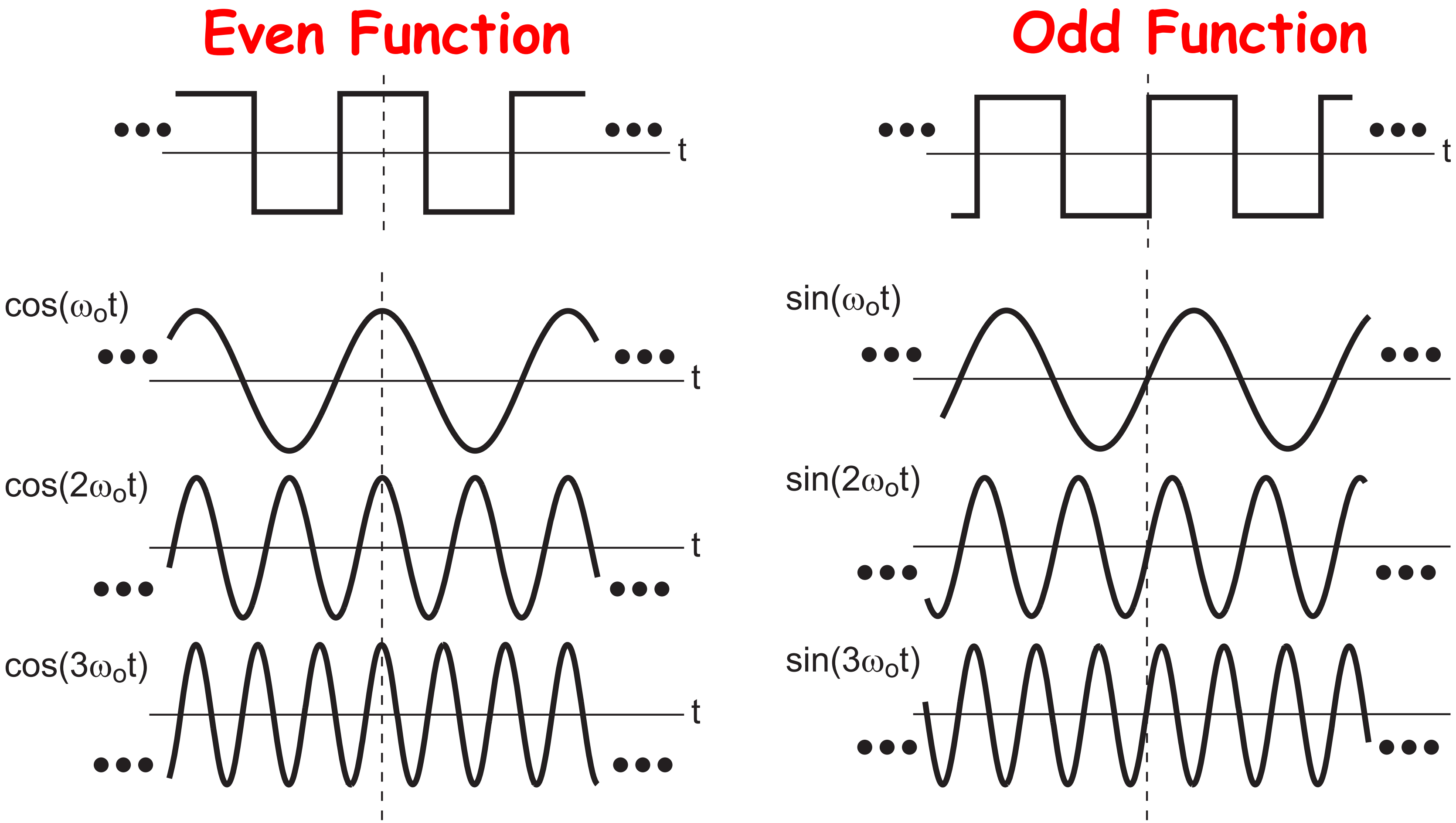
Wrap Up

- Fourier Transform
- Sampling, Modulation, Filtering
- Noise and the Digital Abstraction
- Binary signaling model and Shannon Capacity

Cosines and Sines as Basis Functions

- Periodic functions can be approximated by the addition of weighted cosine and sine waveforms with progressively increasing frequency



Fourier Series and Fourier Transform

- **The Fourier Series deals with *periodic* signals**

$$x(t) = \sum_{n=-\infty}^{\infty} \hat{X}_n e^{jnw_o t}$$

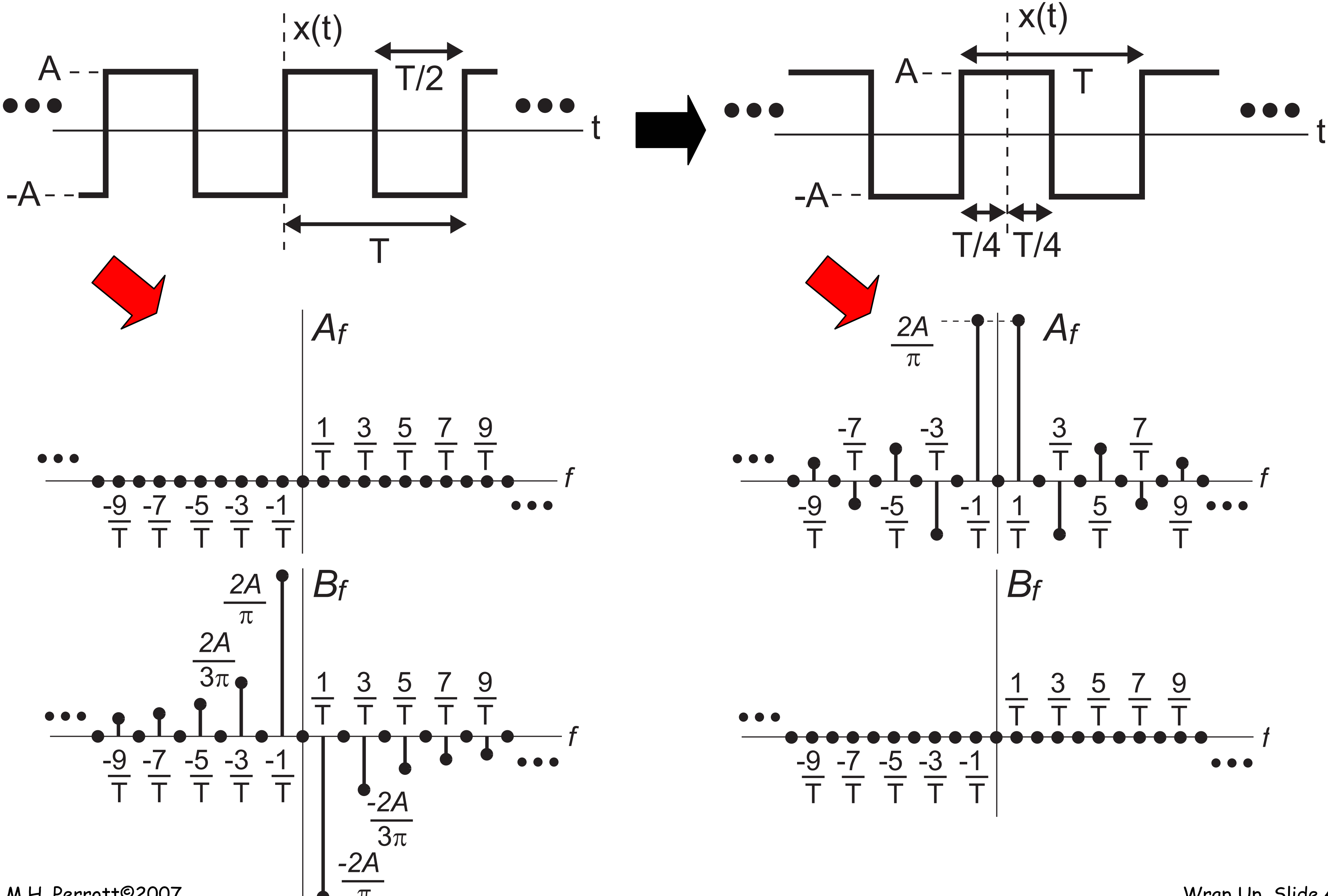
$$\hat{X}_n = \frac{1}{T}\int_{t_o}^{t_o+T} x(t) e^{-jnw_o t} dt$$

- **The Fourier Transform deals with *non-periodic* signals**

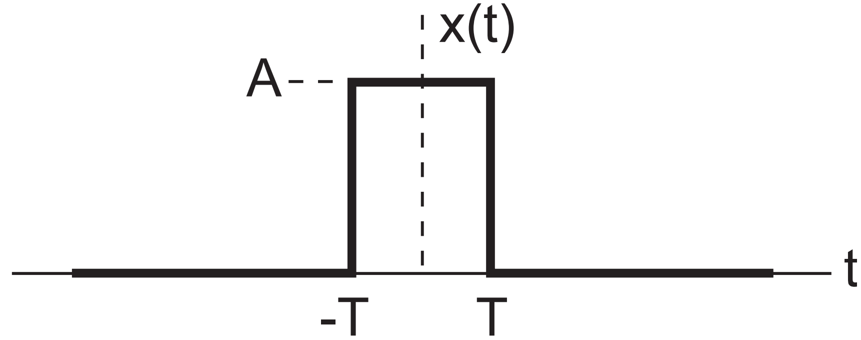
$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi f t} df$$

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt$$

Graphical View of Fourier Series

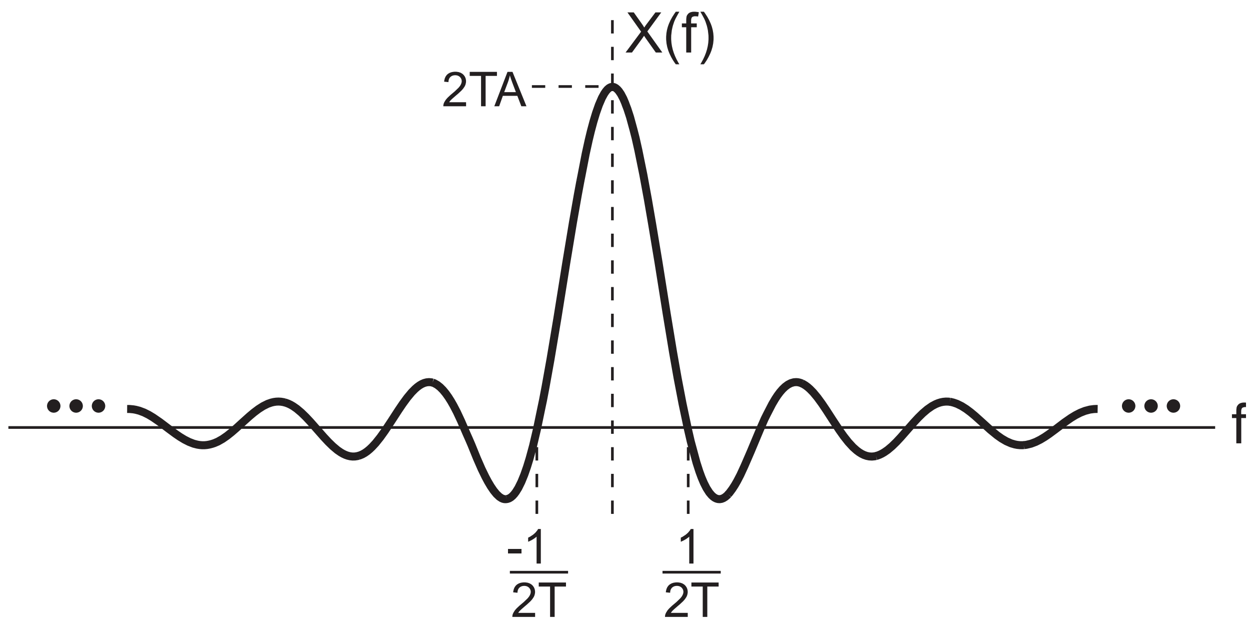


Graphical View of Fourier Transform



$$X(f) = \frac{A \sin(2\pi f T)}{\pi f}$$

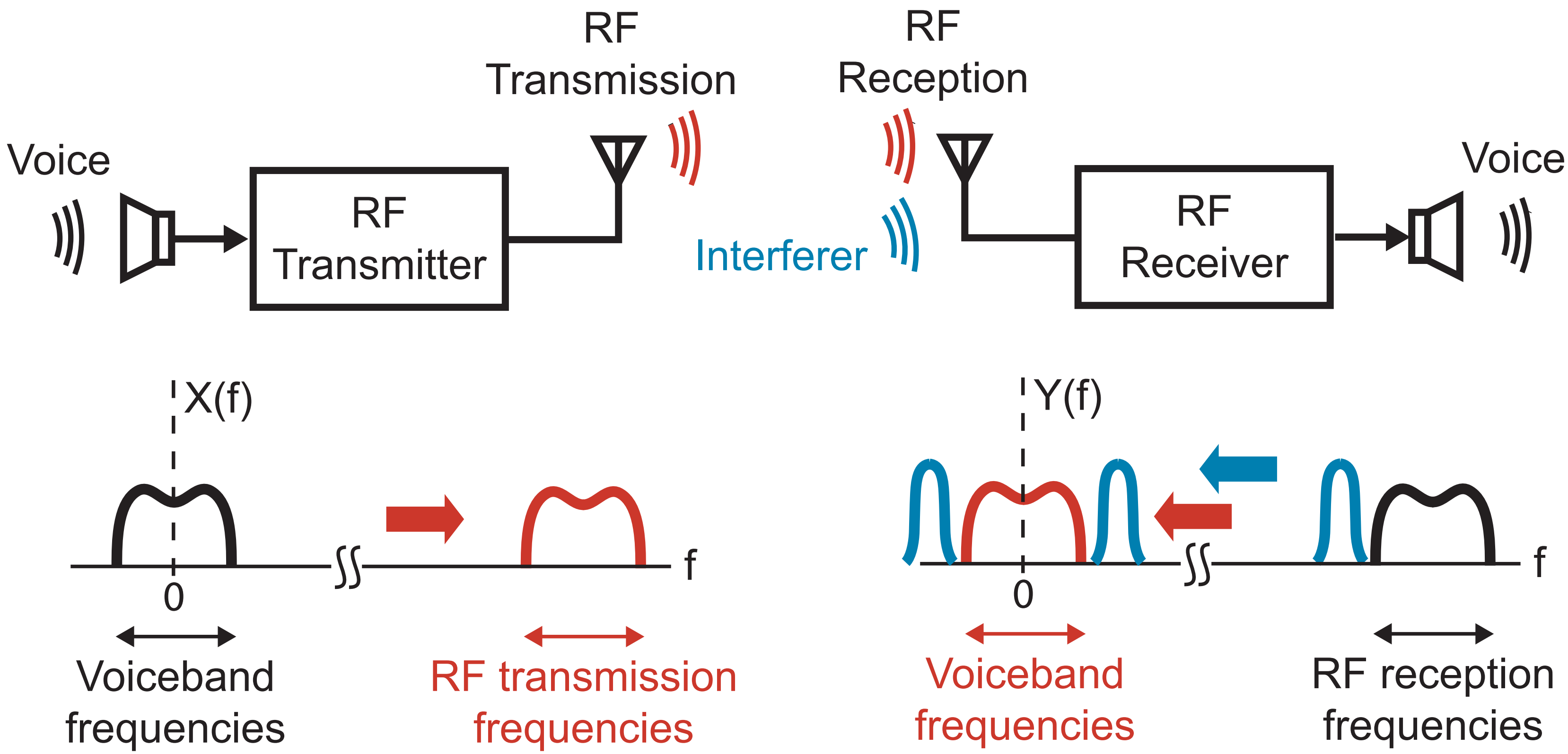
This is called a *sinc* function



Filtering in Continuous and Discrete Time

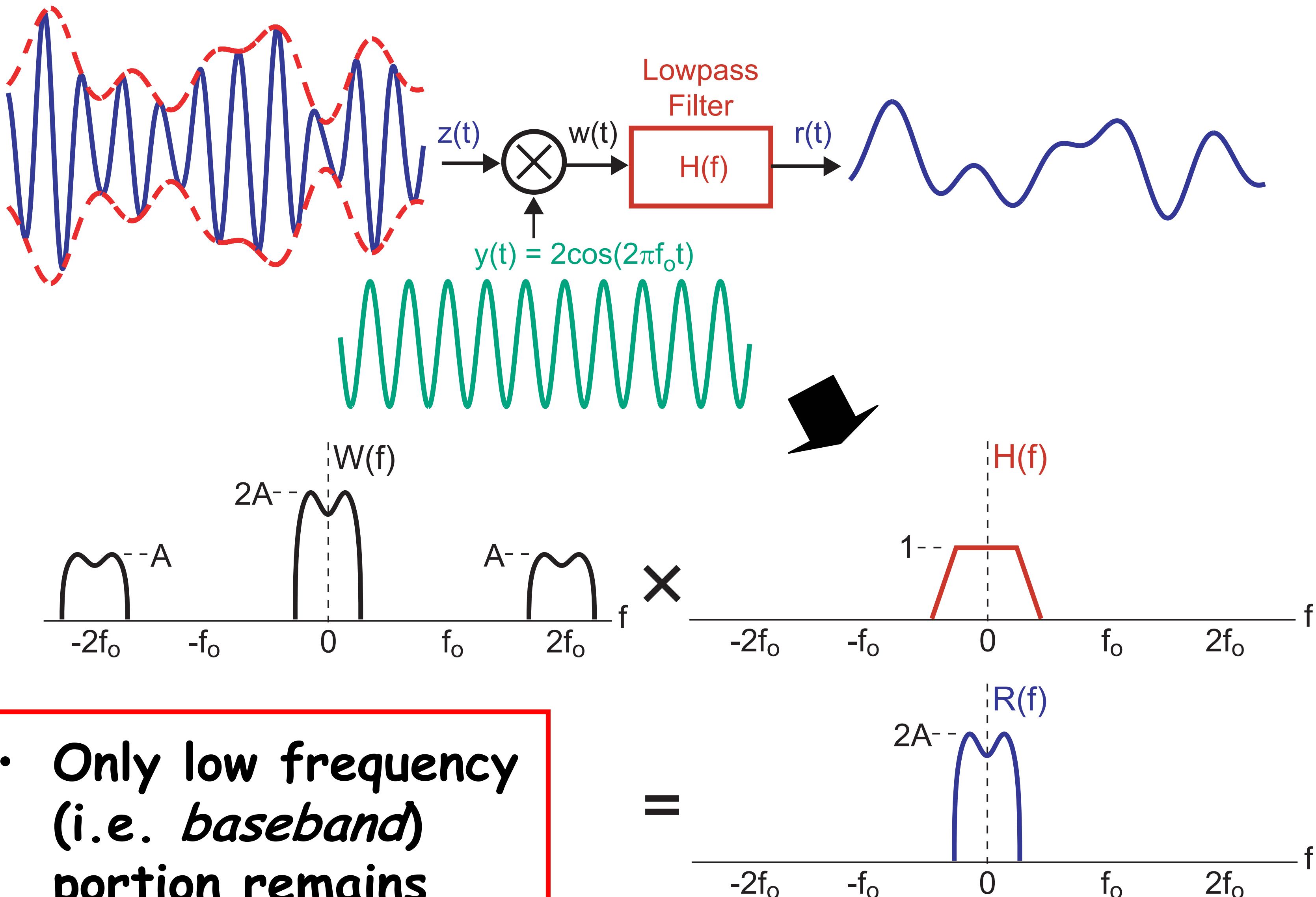
- Lowpass, highpass, bandpass filtering
- Filter response to cosine wave inputs
- Discrete-Time Fourier Transform
- Filtering based on difference equations

Motivation for Filtering



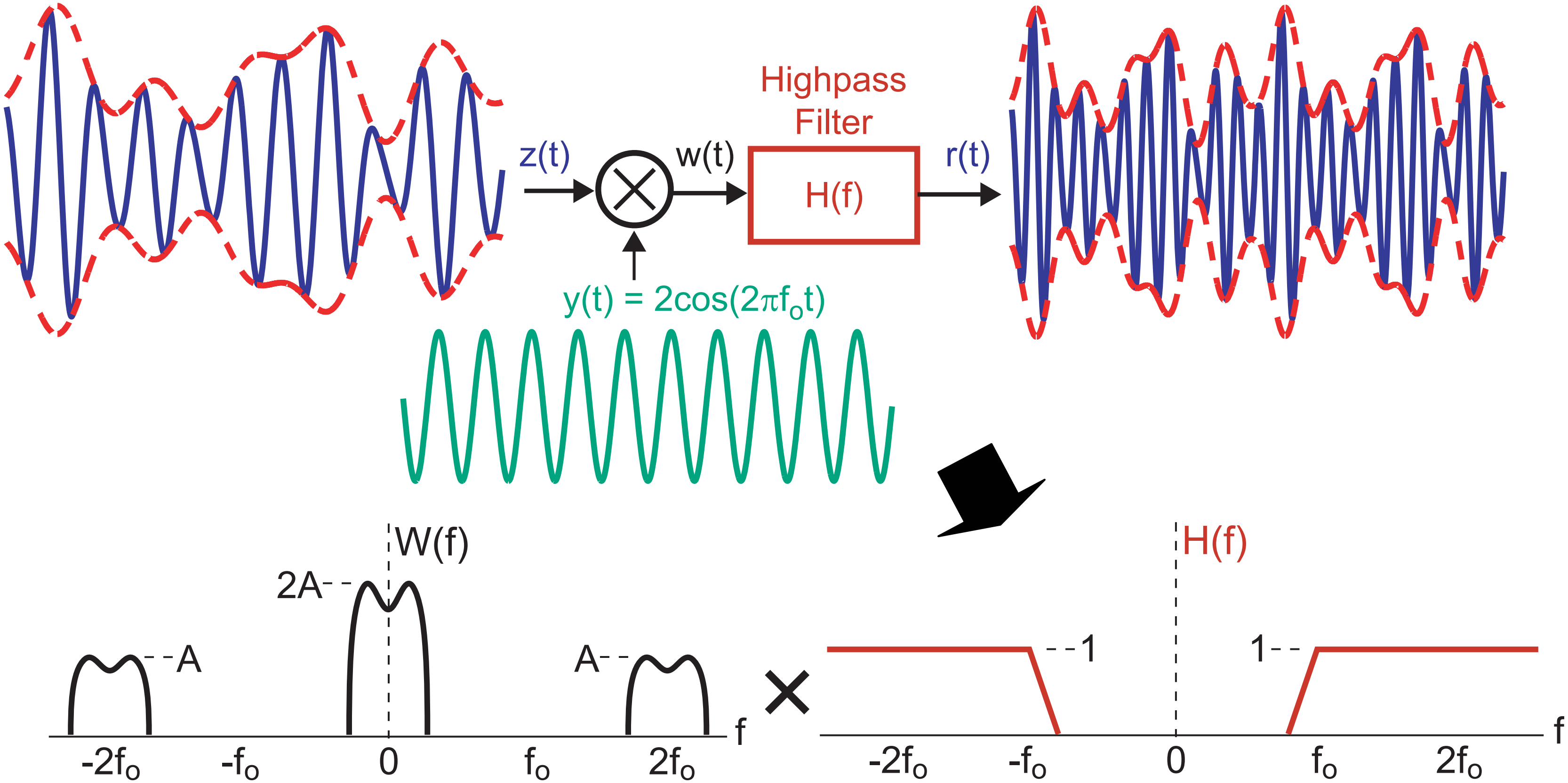
- **Filtering is used to remove undesired signals outside of the frequency band of interest**
  - **Enables selection of a specific radio, TV, WLAN, cell phone, cable TV *channel* …**
  - **Undesired channels are often called interferers**

Lowpass Filter

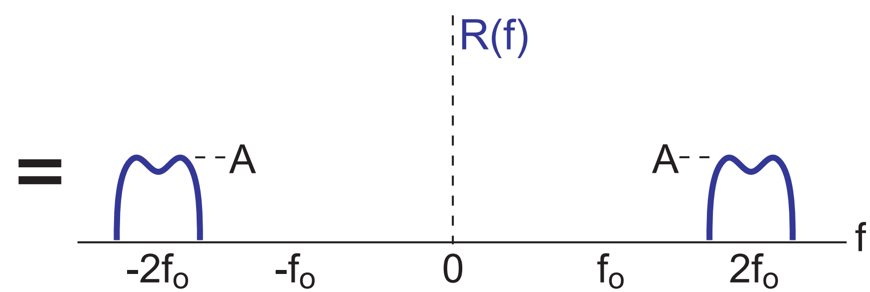


- Only low frequency (i.e. *baseband*) portion remains

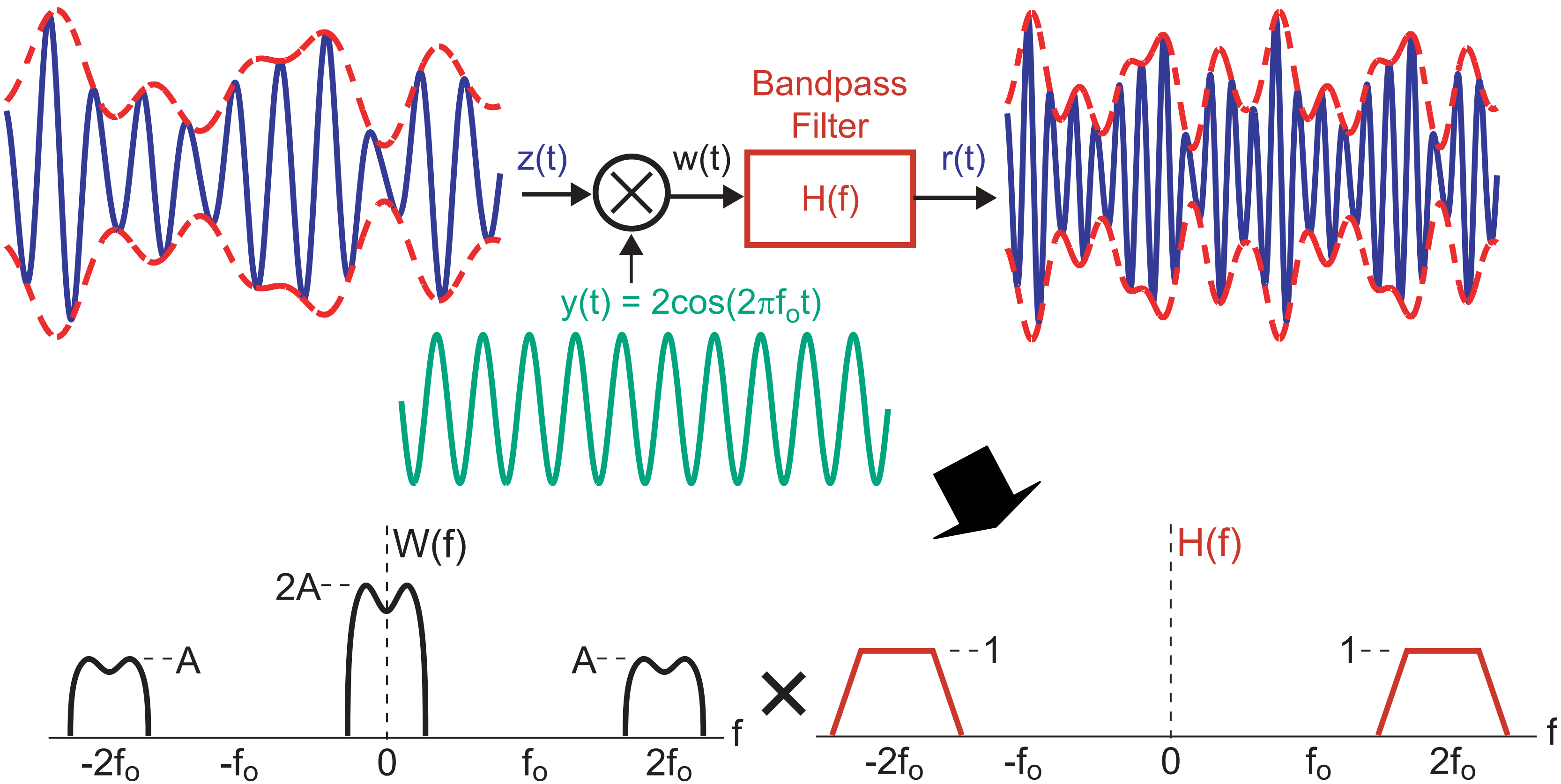
Highpass Filter



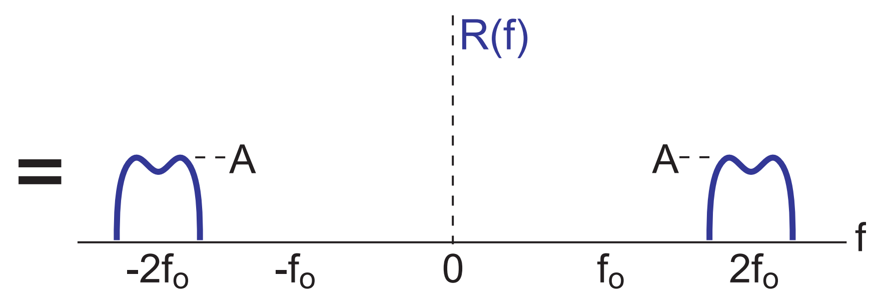
- **Only high frequency (i.e. *RF*) portion remains**



Bandpass Filter

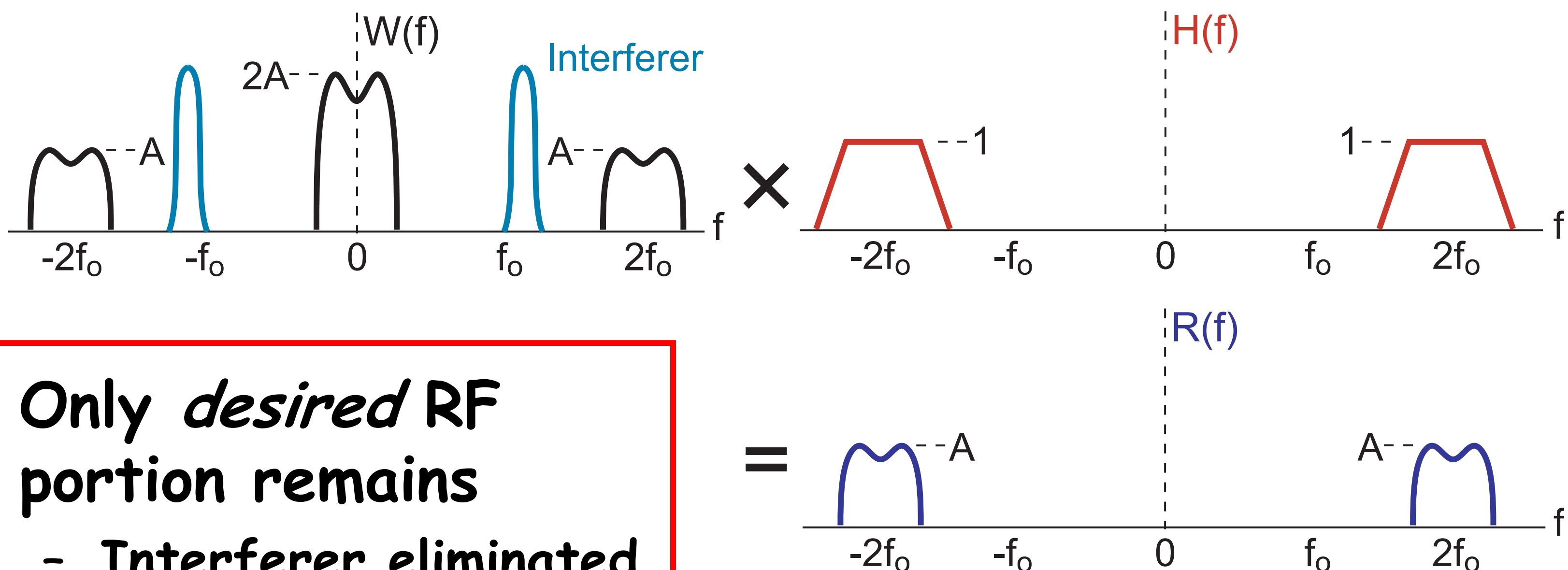


- **Only high frequency (i.e. *RF*) portion remains**



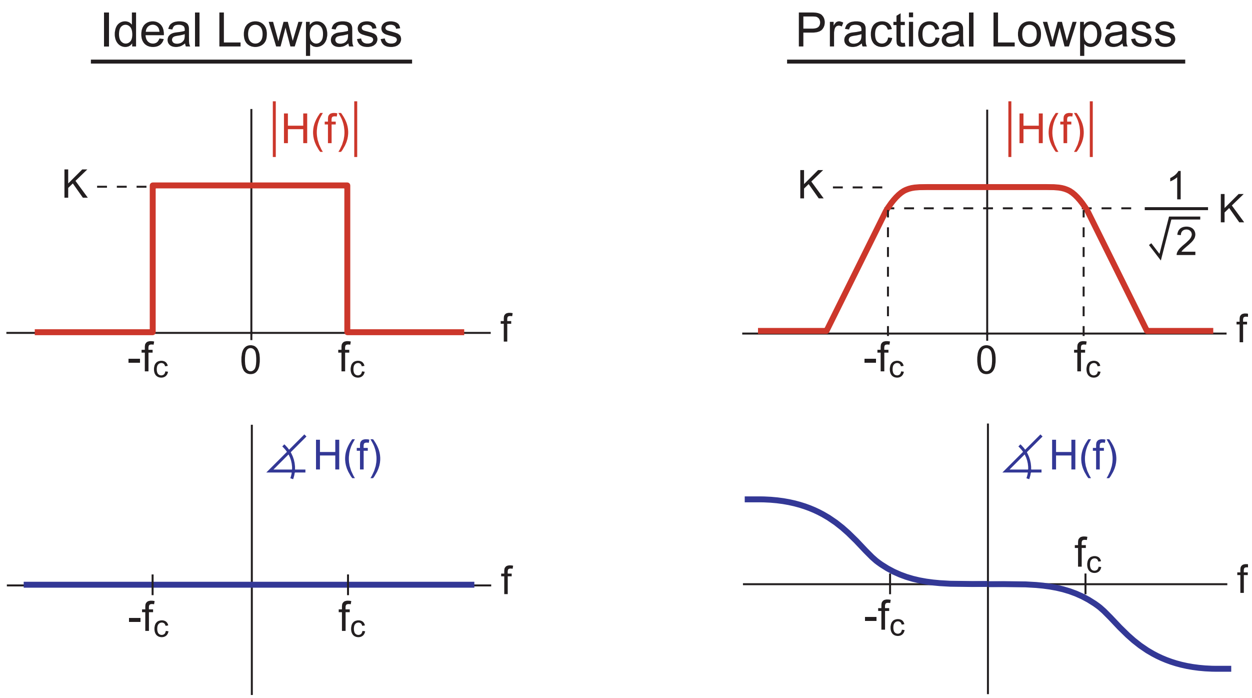
Why is Bandpass Filtering Useful?

- Allows removal of interfering signals
  - Highpass filtering would be of limited use here
- Typically higher complexity implementation than with lowpass or highpass filters
  - Many RF systems such as cell phones use specialized components called *SAW filters* to achieve bandpass filtering



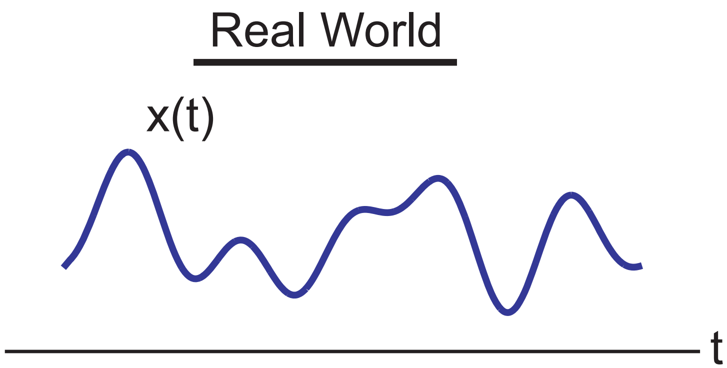
- Only *desired* RF portion remains
  - Interferer eliminated

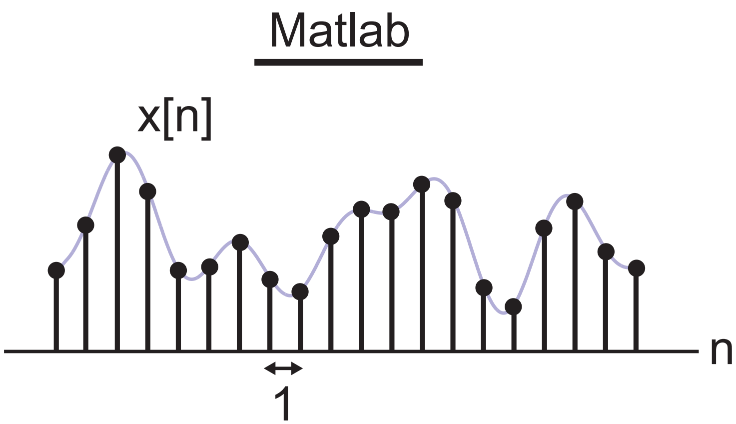
A More Formal Treatment of Filters



- **An ideal filter would have a "brickwall" magnitude response and zero phase response**
  - Practical filters have a more gradual magnitude *rolloff* and a non-zero phase response
- **Design of the filter usually focuses on getting a reasonable magnitude rolloff with a specified cutoff frequency f_c (i.e., filter *bandwidth*)**

Designing and Using Filters Within Matlab





- **Our lab exercises will have you design and use filters in Matlab**
  - Matlab will interface to the USRP board in order to receive "real world" signals from the antenna
- **Matlab framework is based on *discrete-time sequences* (which are indexed on integer values)**
  - Correspond to *samples* of corresponding real world signals (which are *continuous-time* in nature)

We need another Fourier analysis tool

The Discrete-Time Fourier Transform

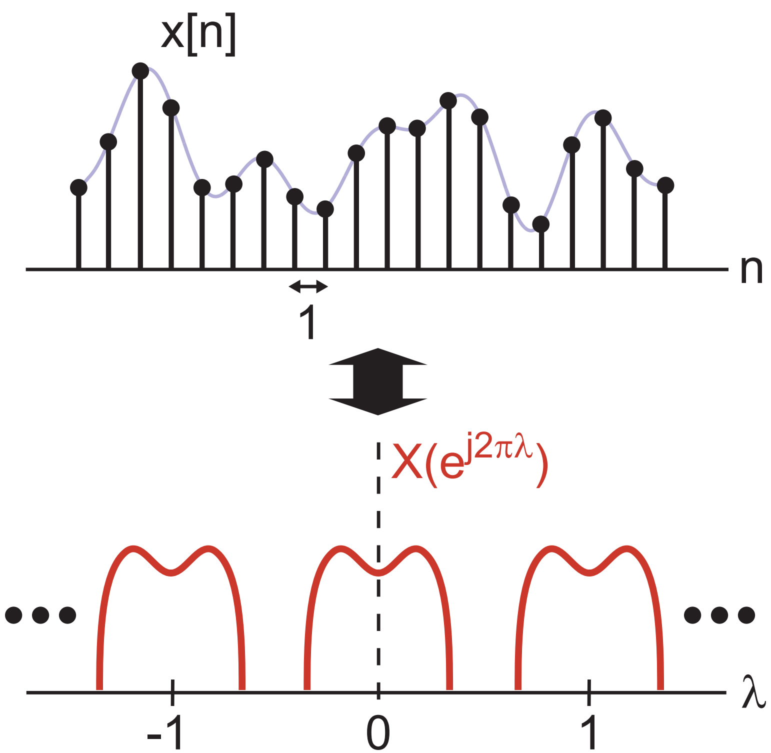
- Allows us to deal with *non-periodic, discrete-time* signals
- Frequency domain signal is *periodic* in this case

$$x[n] \Leftrightarrow X(e^{j2\pi\lambda})$$

- **Where:**

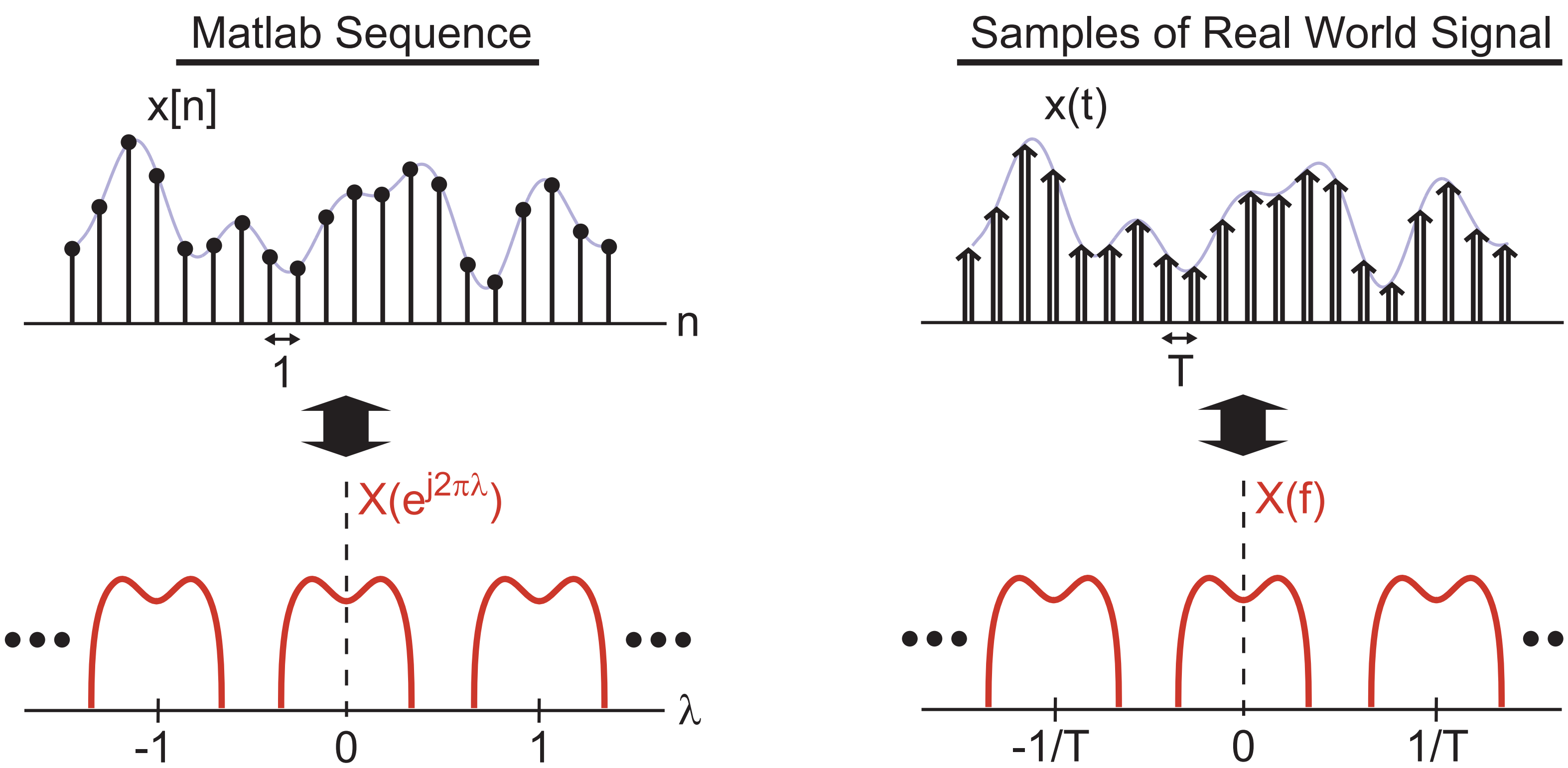
$$x[n] = \int_{-1/2}^{1/2} X(e^{j2\pi\lambda}) e^{j2\pi\lambda n} d\lambda$$

$$X(e^{j2\pi\lambda}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j2\pi\lambda n}$$



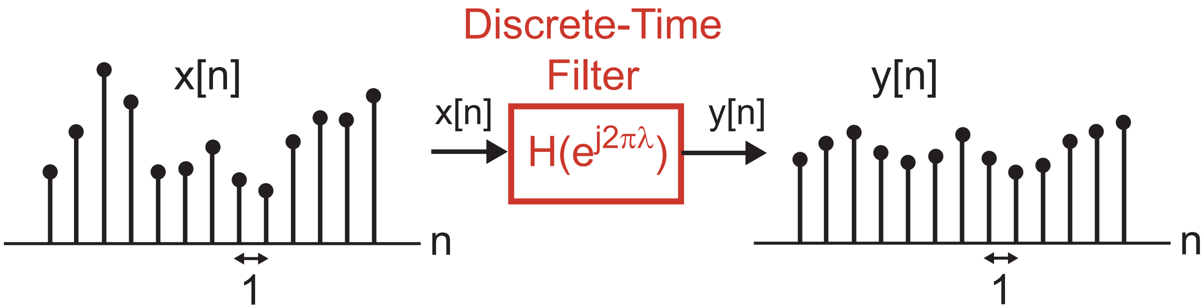
Note: *fft* function in Matlab used to compute *DTFT*

Relating to Samples of `Real World' Signals



- Samples of a continuous-time signal with sample period *T* leads to frequency domain signal with period *1/T*
  - We simply scale frequency axis of *fft* in Matlab
- We will say much more about *sampling* later …

Filters Within Matlab



- Implemented as *difference equations*
  - Current output, y[n], depends on weighted values of previous output samples and current and previous input samples, x[n]

$$y[n] = \sum_{k=1}^{M} a_k y[n-k] + \sum_{k=0}^{N} b_k x[n-k]$$

- Group *a* and *b* coefficients as vectors:

$$\mathbf{a} = [a_0 \; a_1 \; \cdots \; a_M], \quad \mathbf{b} = [b_0 \; b_1 \; \cdots \; b_N]$$

- Execute filter using the *filter* command:

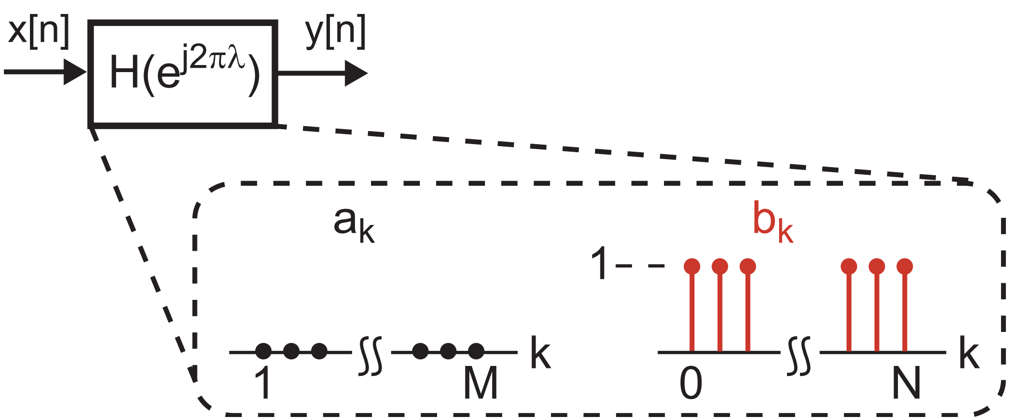
$$y = \mathrm{filter}(b, a, x);$$

FIR Filters

- **Finite Impulse Response (FIR) filters use only *b* coefficients in their implementation**

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] \quad \Rightarrow \quad \boxed{H(e^{j2\pi\lambda}) = \sum_{k=0}^{N} b_k e^{-j2\pi\lambda k}}$$

- **Example:**

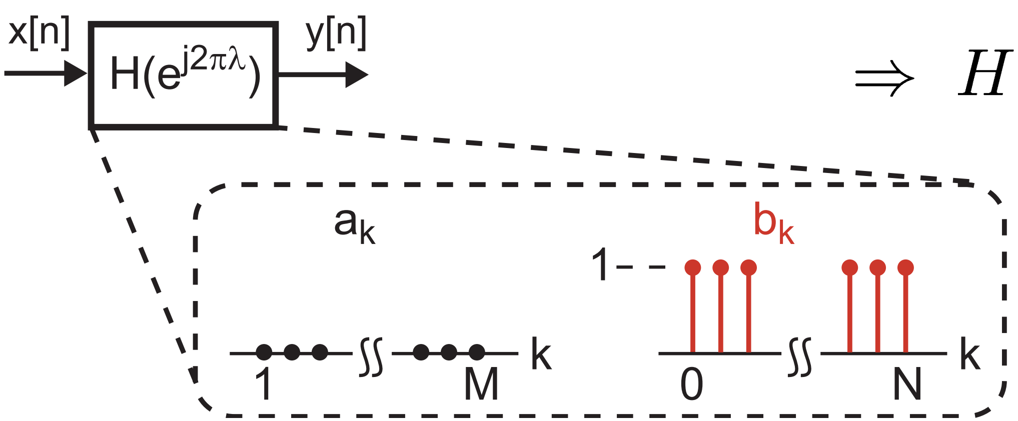


Note:

$$\sum_{k=0}^{N} r^k = \frac{1-r^{N+1}}{1-r}$$

$$H(e^{j2\pi\lambda}) = \sum_{k=0}^{N} 1 e^{-j2\pi\lambda k} = \boxed{\frac{1-e^{-j2\pi\lambda(N+1)}}{1-e^{-j2\pi\lambda}}}$$

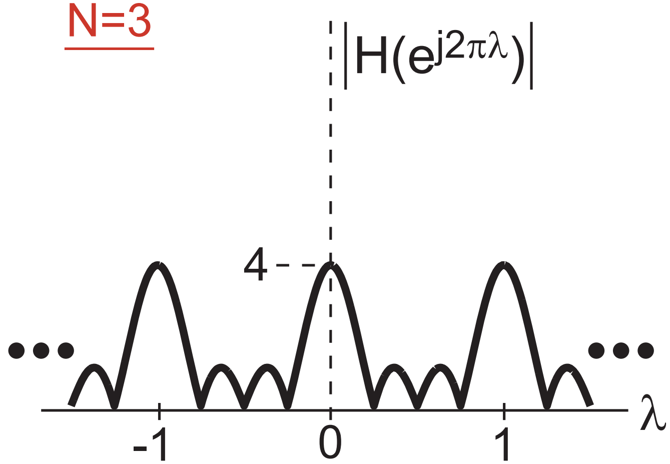
Filter Order for FIR Filters



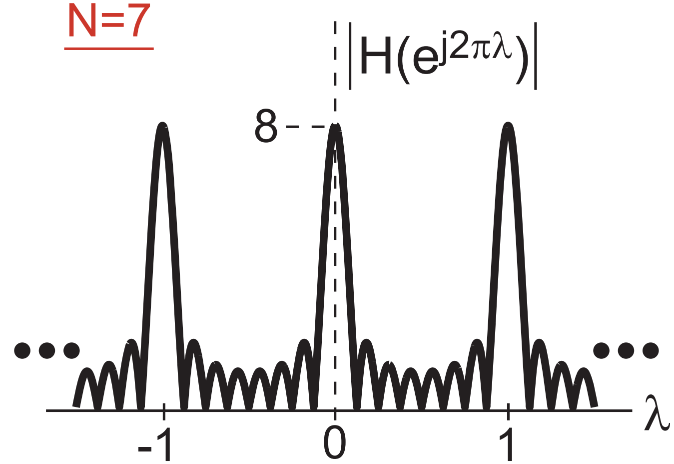
$$\Rightarrow \; H(e^{j2\pi\lambda}) = \frac{1 - e^{-j2\pi\lambda(N+1)}}{1 - e^{-j2\pi\lambda}}$$

- Consider two different values for *N*

N=3

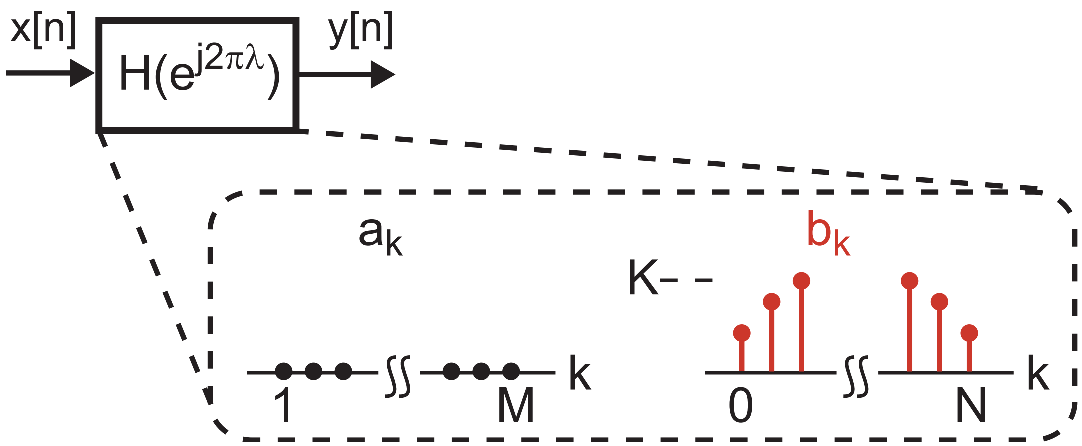


N=7



- Higher *N* leads to steeper filter response
  - We refer to *N* as the *order* of the filter

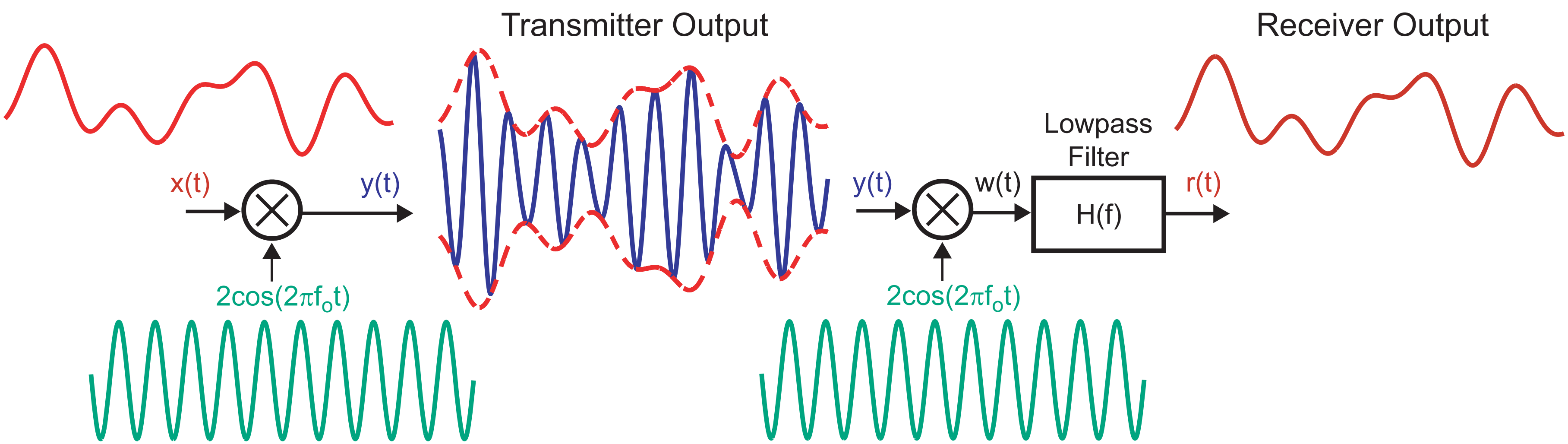
FIR Filter Design in Matlab



- Lowpass, highpass, and bandpass filters can be realized by appropriately scaling the relative value of the *b* coefficients
  - Higher order (i.e., higher *N*) leads to steeper responses
- Perform FIR filter design using *fir1* command
- Frequency response observed with *freqz* command

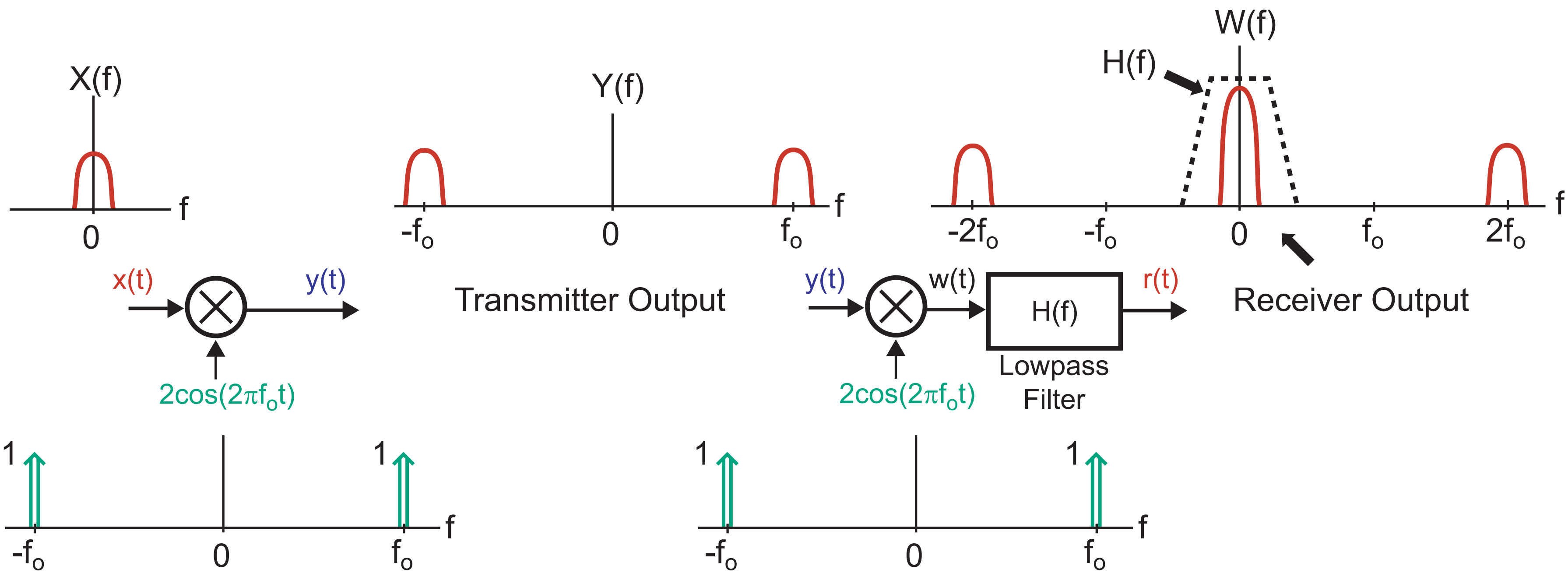
See Prelab portion of Lab 3 for details …

AM Modulation and Demodulation



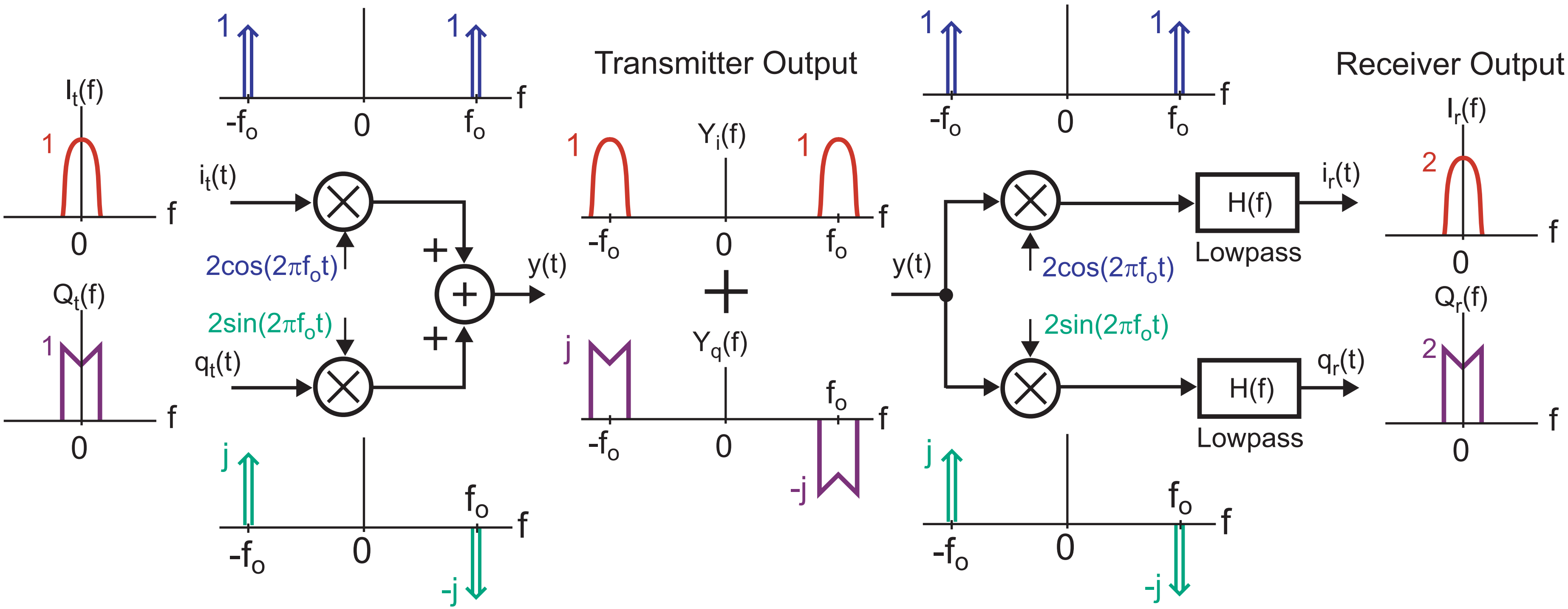
- **Multiplication (i.e., *mixing*) operation shifts in frequency**
  - **Also creates undesired high frequency components at receiver**
- **Lowpass filtering passes only the desired *baseband* signal at receiver**

Frequency Domain Analysis



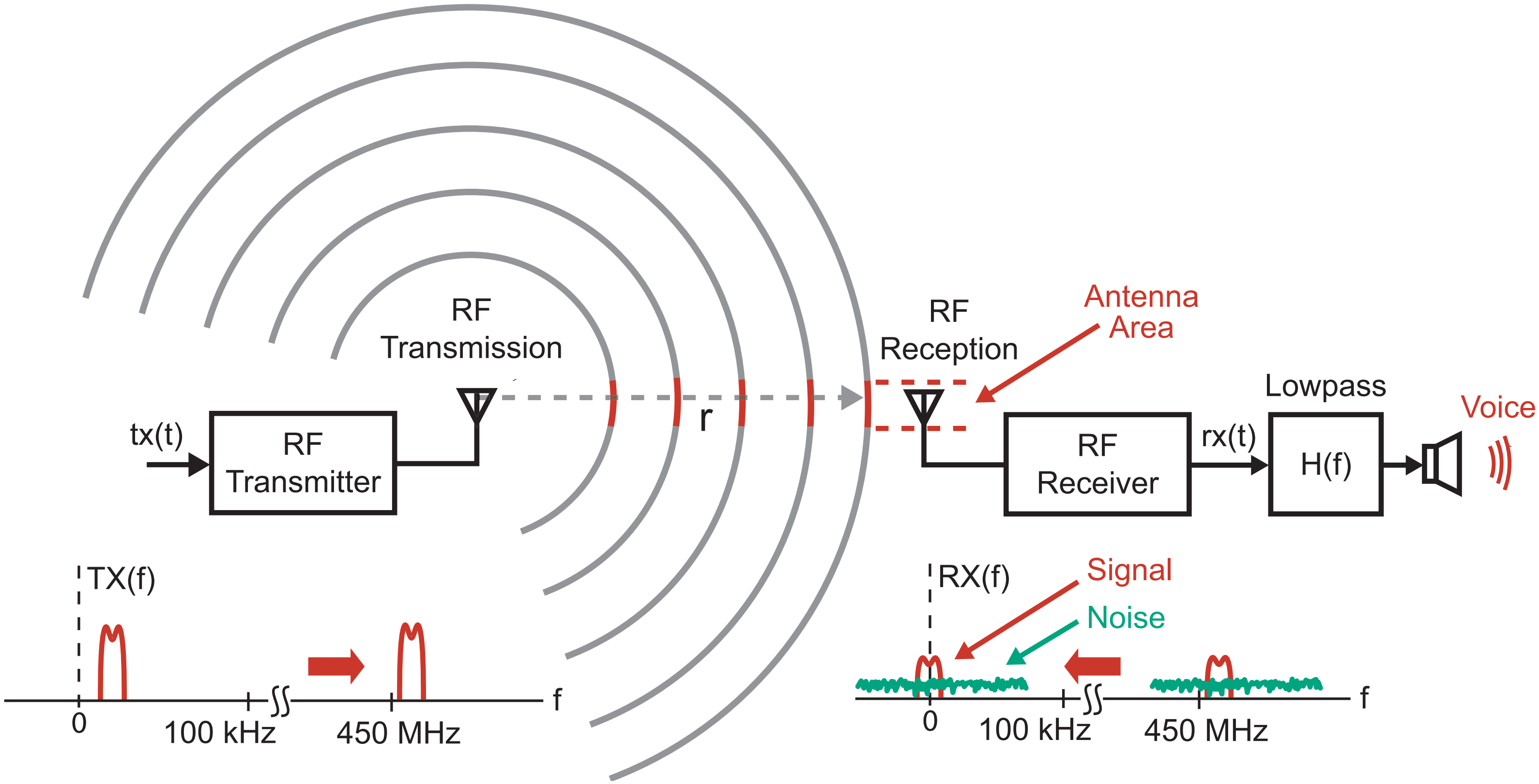
- **When transmitter and receiver local oscillators are matched in phase:**
  - **Demodulated signal *constructively* adds at baseband**

I/Q Modulation



- **Modulate with *both* a cosine and sine wave**
  - I and Q channels can be broadcast over the *same* frequency band
- **I/Q modulation allows twice the amount of *information* to be sent compared to basic AM modulation with same *bandwidth***

Energy Transfer in Wireless Communication

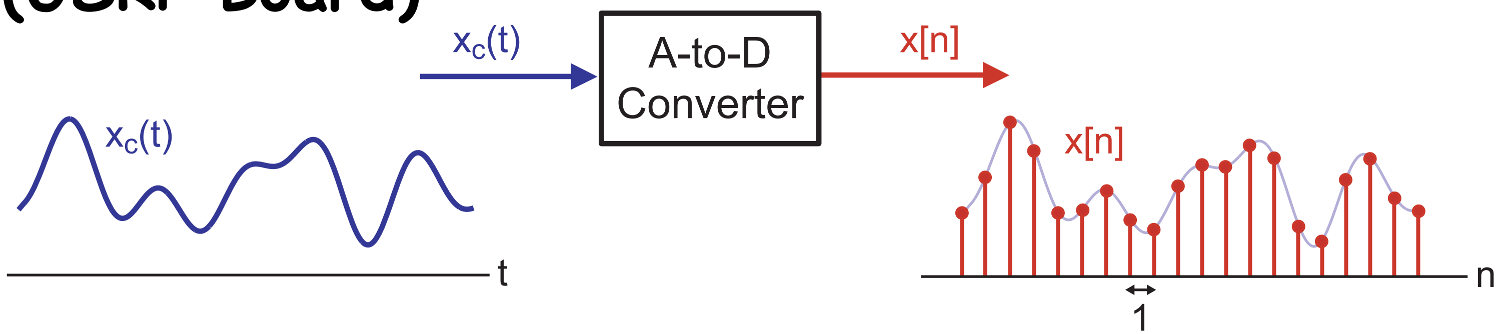


- Receiver antenna is limited in its ability to capture transmitter energy according to its *area* and *distance, r,* from transmitter
- Noise in the receiver causes corruption
  - Amount of corruption depends on signal-to-noise ratio

The Need for Sampling

Real World (USRP Board)

Matlab

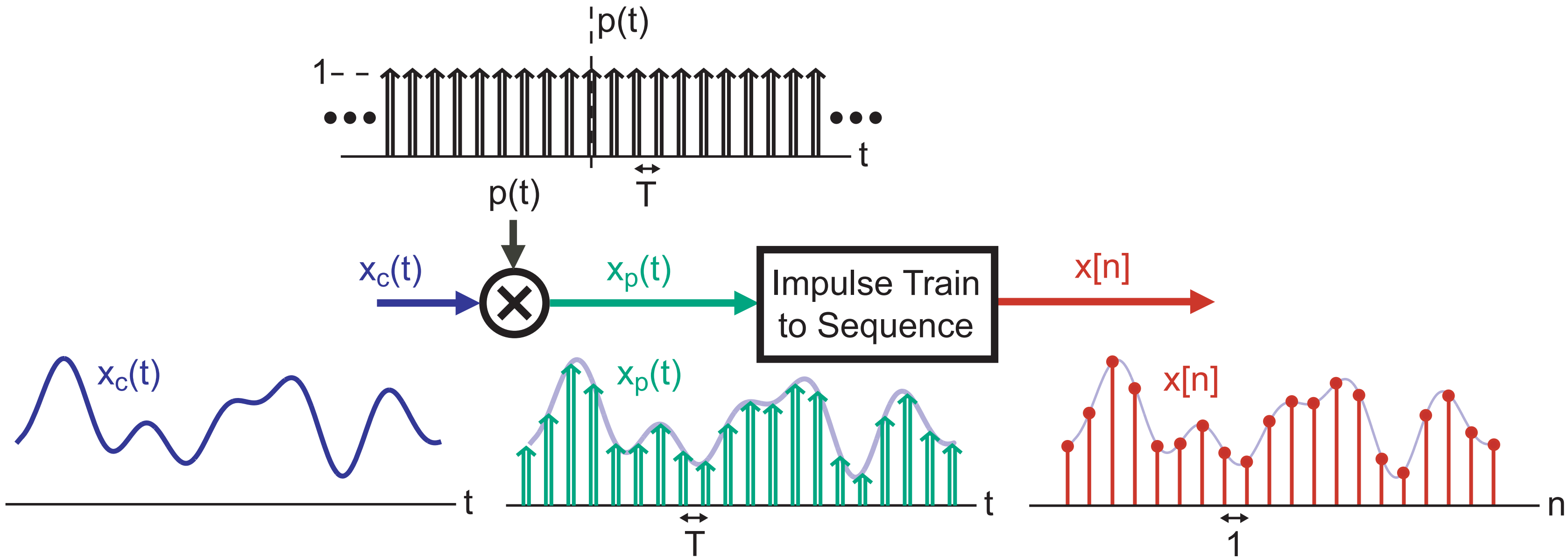


- **The boundary between *analog* and *digital***
  - **Real world is filled with *continuous-time signals***
  - **Computers (i.e. Matlab) operate on *sequences***
- **Crossing the analog-to-digital boundary requires *sampling* of the continuous-time signals**

Sampling Continuous-Time Signals

- Impulse train and its Fourier Transform
- Impulse samples versus discrete-time sequences
- Aliasing and the Sampling Theorem
- Anti-alias filtering
- Comparison of FT, DTFT, Fourier Series

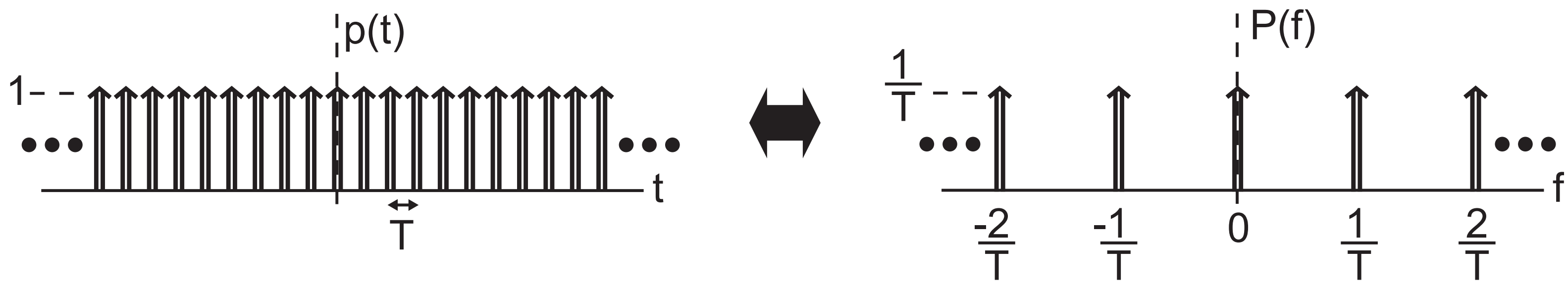
An Analytical Model for Sampling



- **Two step process**
  - Sample continuous-time signal every *T* seconds
    - Model as *multiplication* of signal with *impulse train*
  - Create sequence from amplitude of scaled impulses
    - Model as *rescaling* of time axis (*T* goes to 1)
    - Notation: replace impulses with stem symbols

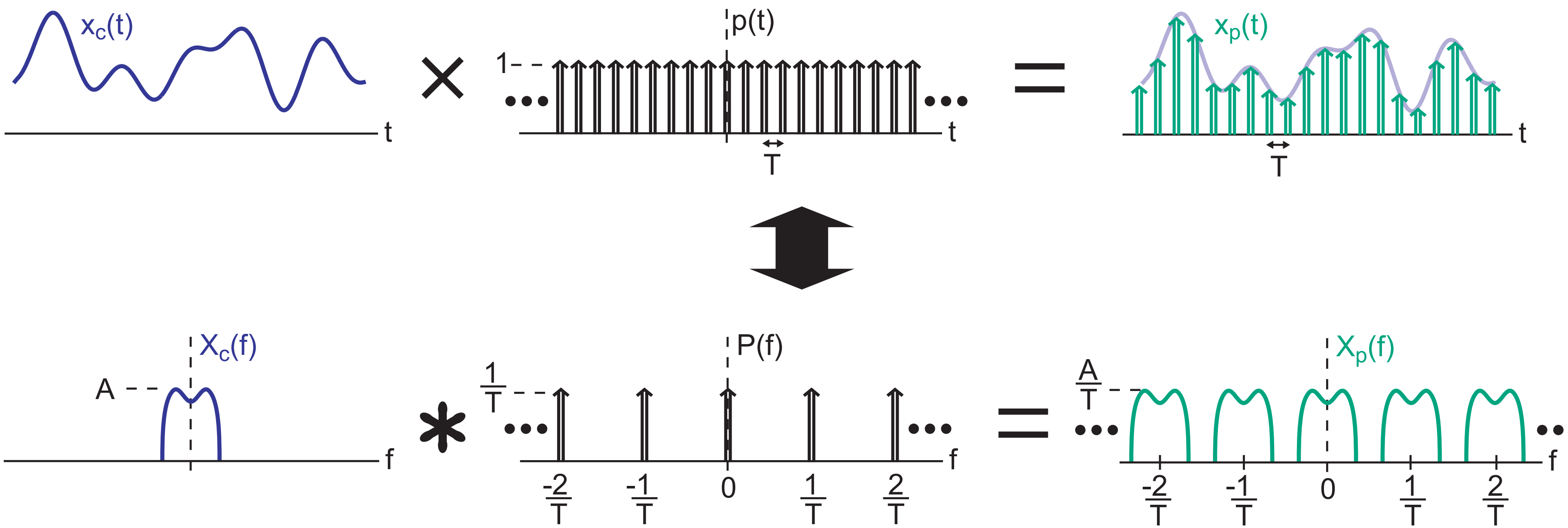
Can we model this in the frequency domain?

Fourier Transform of Impulse Train



- Impulse train in time corresponds to impulse train in frequency
  - Spacing in time of *T* seconds corresponds to spacing in frequency of *1/T* Hz
  - Scale factor of *1/T* for impulses in frequency domain
  - Note: this is painful to derive, so we won't …
- The above transform pair allows us to see the following with *pictures*
  - Sampling operation in frequency domain
  - Intuitive comparison of FT, DTFT, and Fourier Series

Frequency Domain View of Sampling

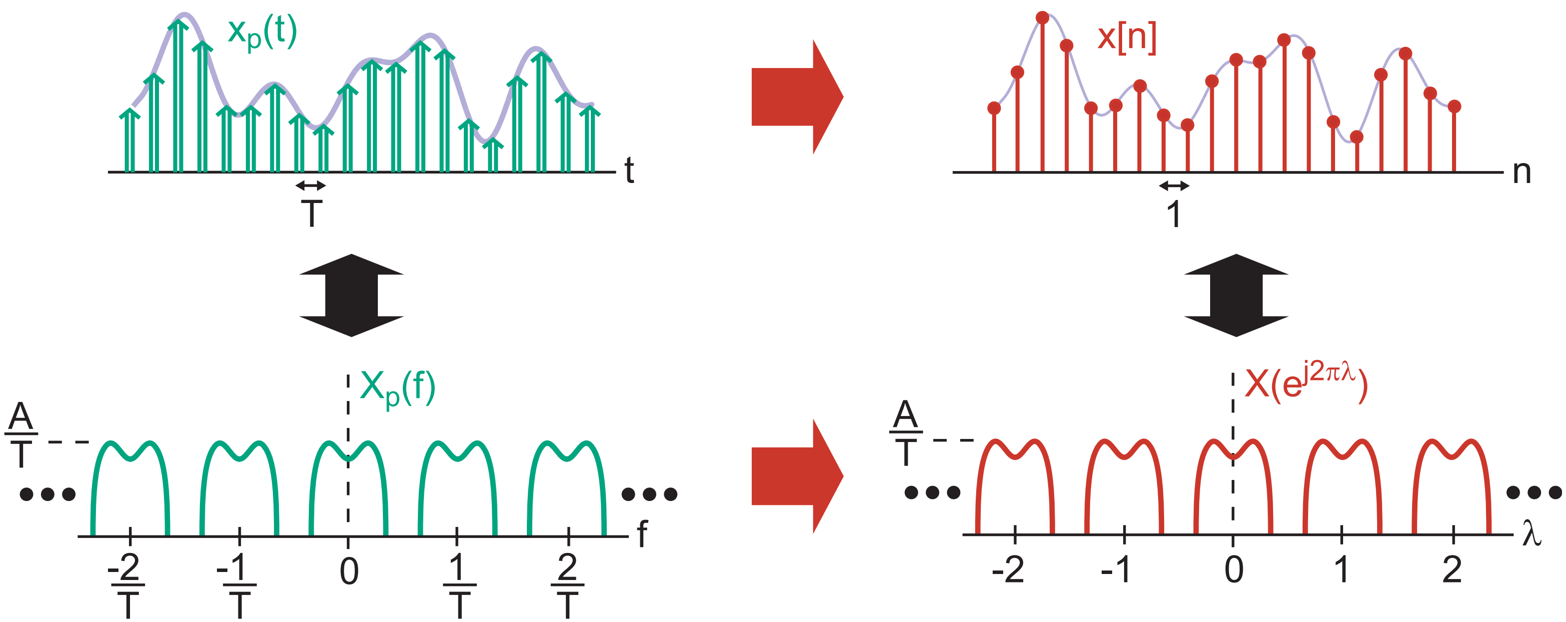


- Recall that *multiplication* in *time* corresponds to *convolution* in *frequency*

$$x(t)y(t) \Leftrightarrow X(f) * Y(f)$$

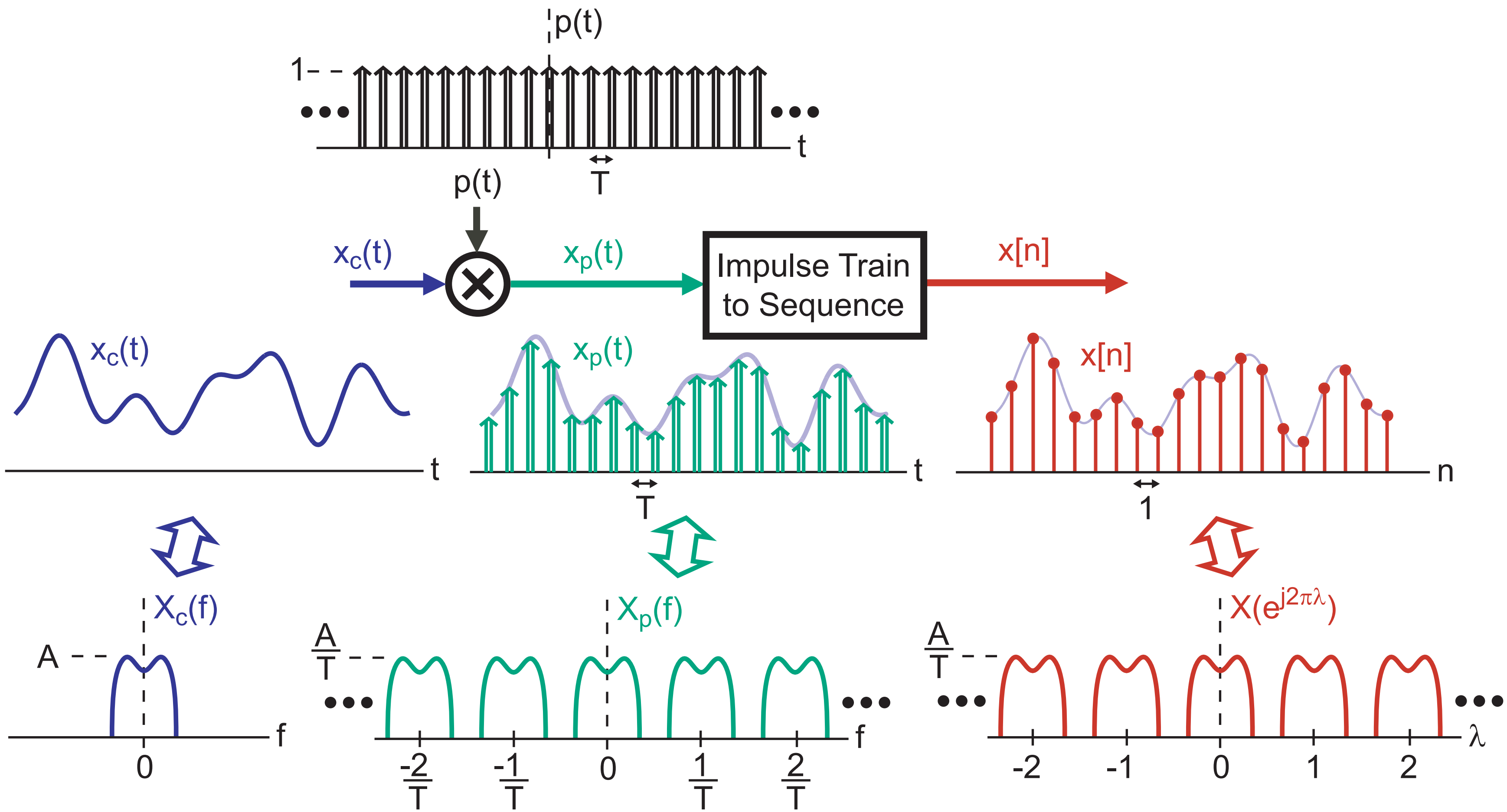
- We see that sampling in time leads to a *periodic* Fourier Transform with period *1/T*

Frequency Domain View of Output Sequence



- Scaling in time leads to scaling in frequency
  - Compression/expansion in time leads to expansion/compression in frequency
- Conversion to sequence amounts to *T* going to 1
  - Resulting Fourier Transform is now periodic with period 1
  - Note that we are now essentially dealing with the DTFT

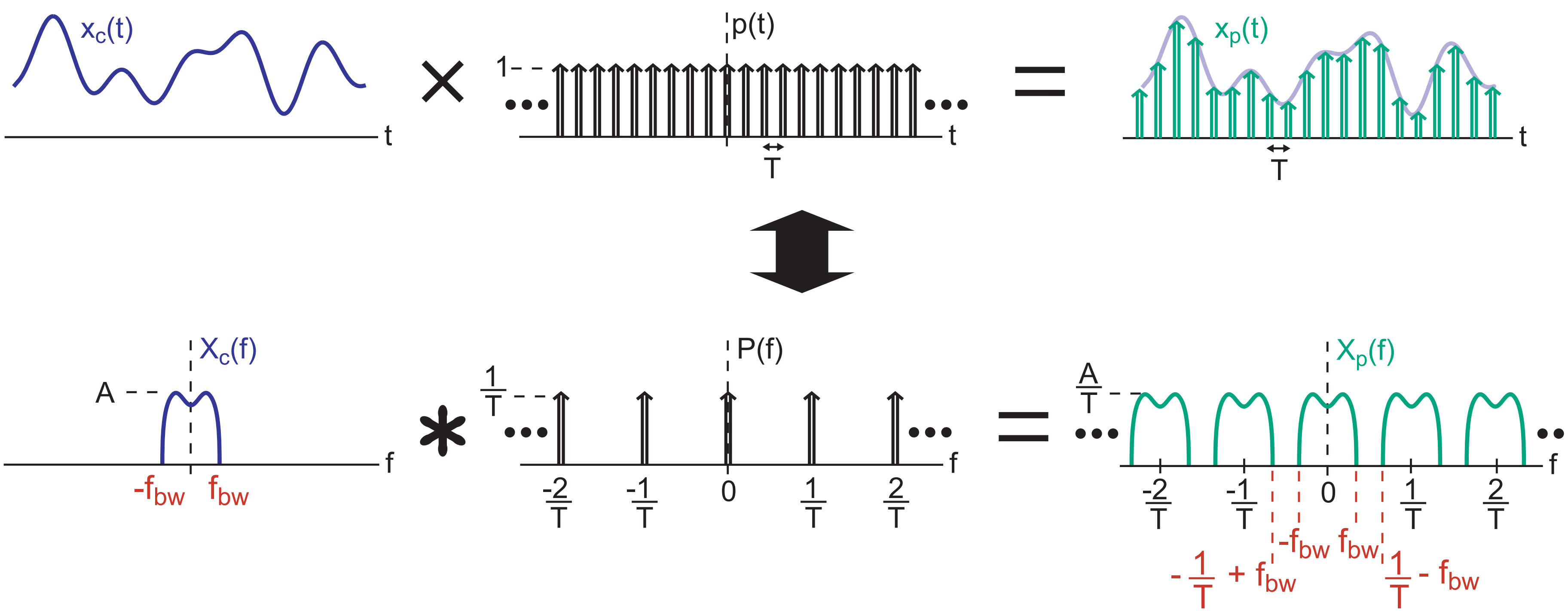
Summary of Sampling Process



- Sampling leads to periodicity in frequency domain

We need to avoid overlap of replicated signals in frequency domain (i.e., aliasing)

The Sampling Theorem

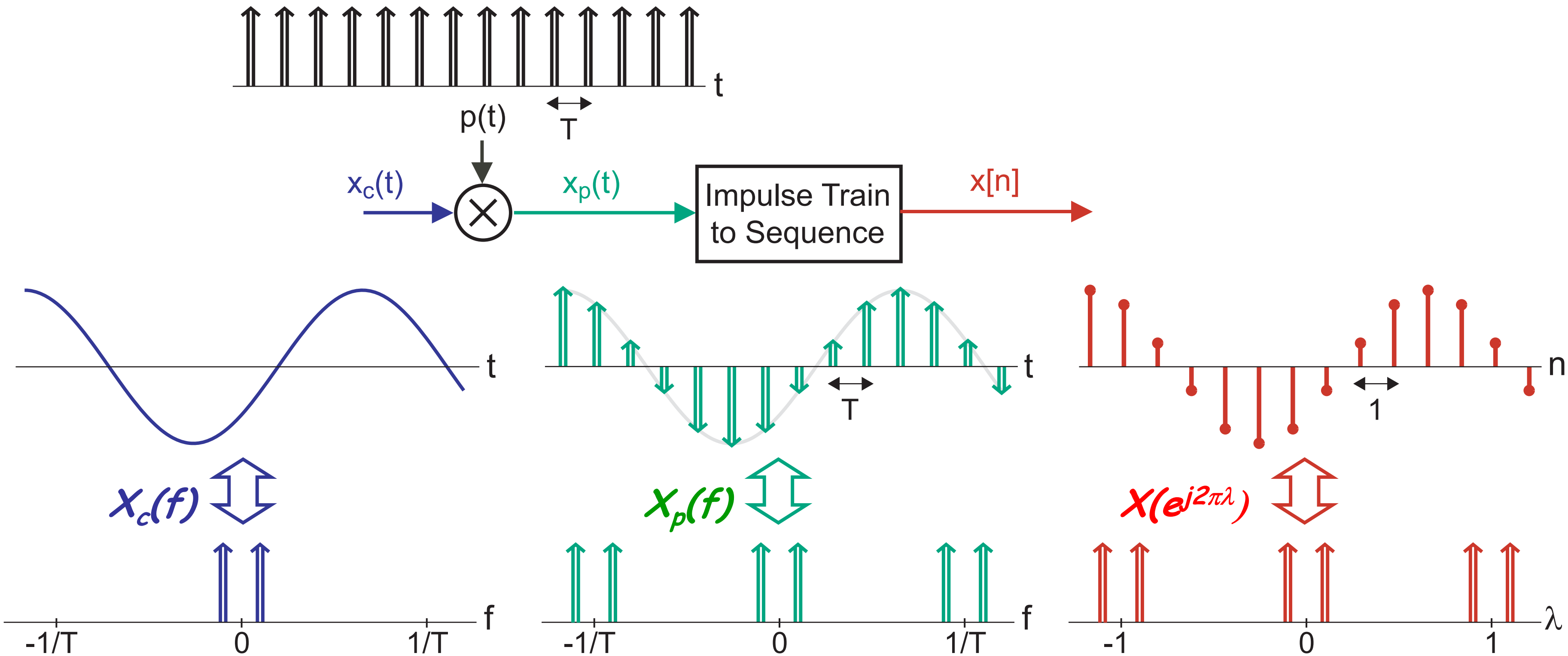


- **Overlap in frequency domain (i.e., aliasing) is avoided if:**

$$\frac{1}{T} - f_{bw} \geq f_{bw} \quad \Rightarrow \quad \boxed{\frac{1}{T} \geq 2f_{bw}}$$

- **We refer to the minimum *1/T* that avoids aliasing as the *Nyquist* sampling frequency**

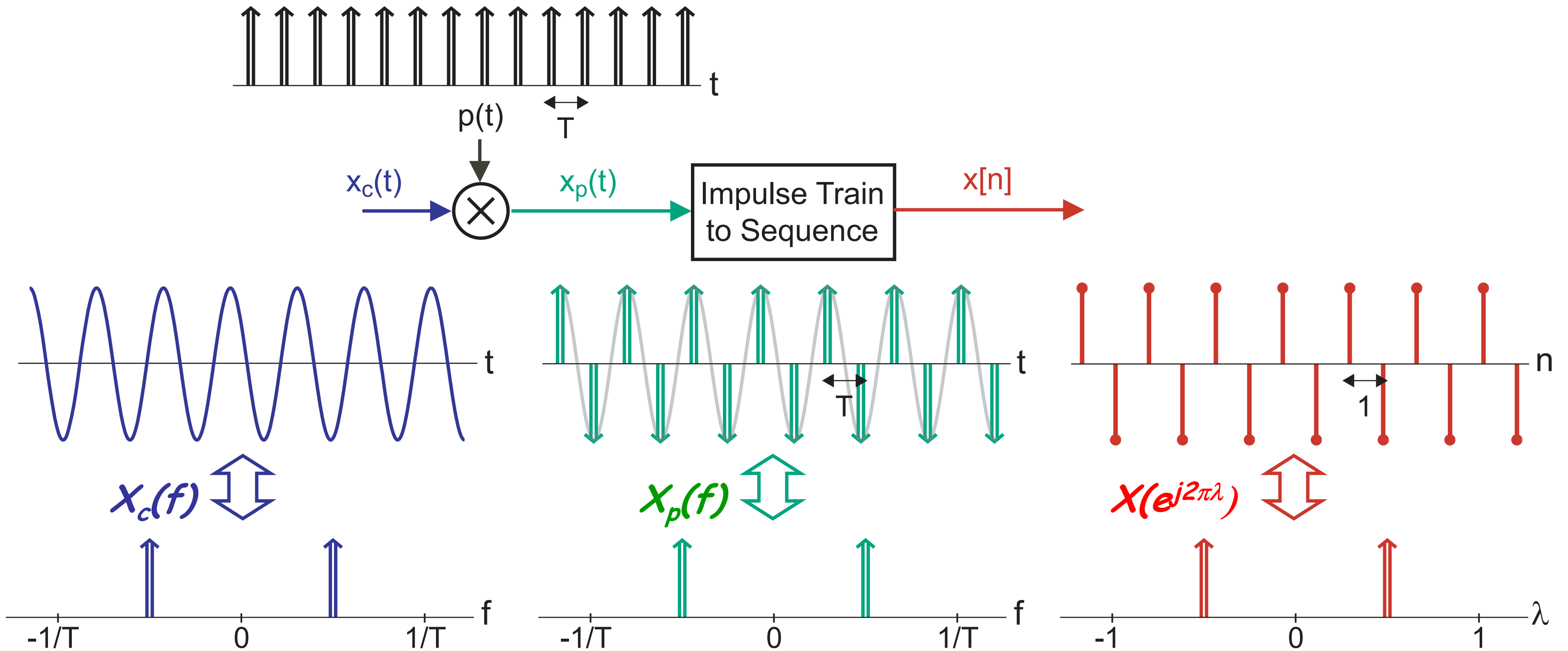
Example: Sample a Sine Wave



Sample rate is well *above* Nyquist rate

- Time domain: resulting sequence maintains the same period as the input continuous-time signal
- Frequency domain: no aliasing

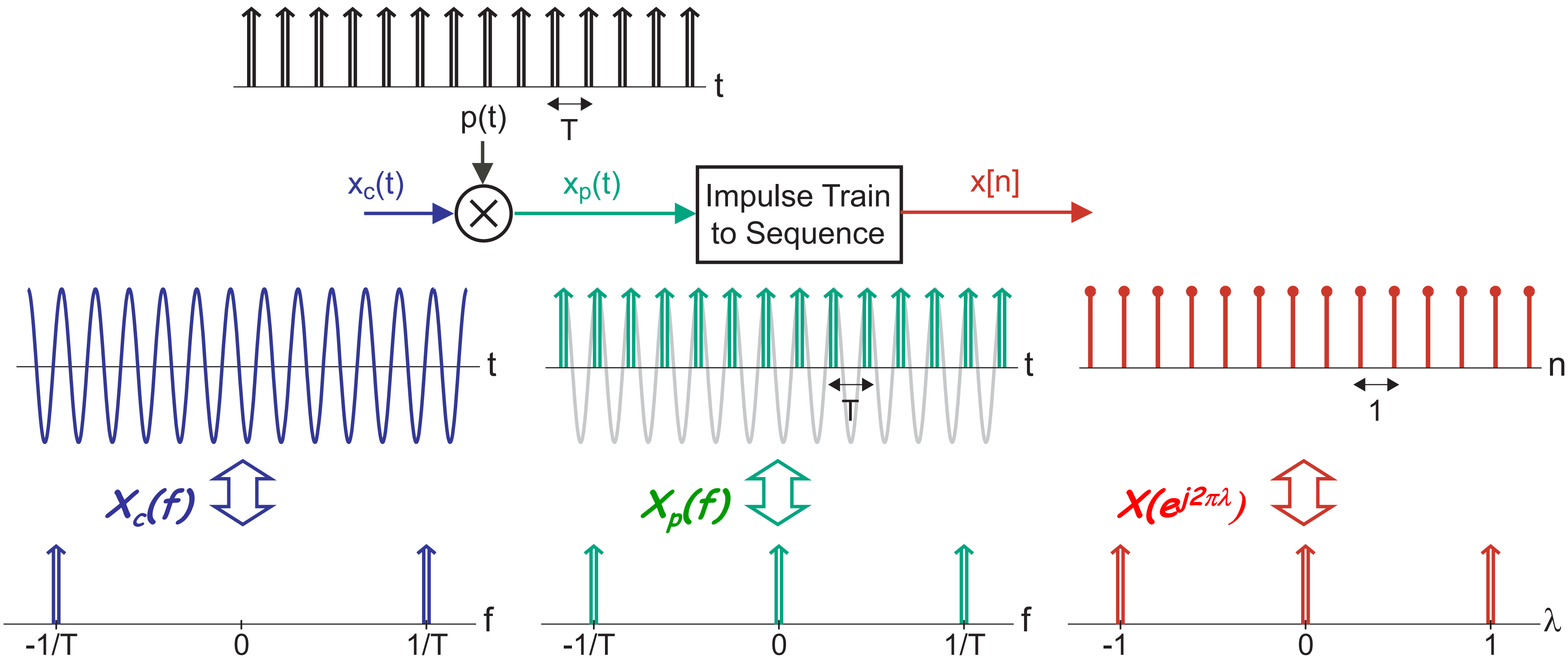
Increase Input Frequency Further …



Sample rate is *at* Nyquist rate

- Time domain: resulting sequence still maintains the same period as the input continuous-time signal
- Frequency domain: no aliasing

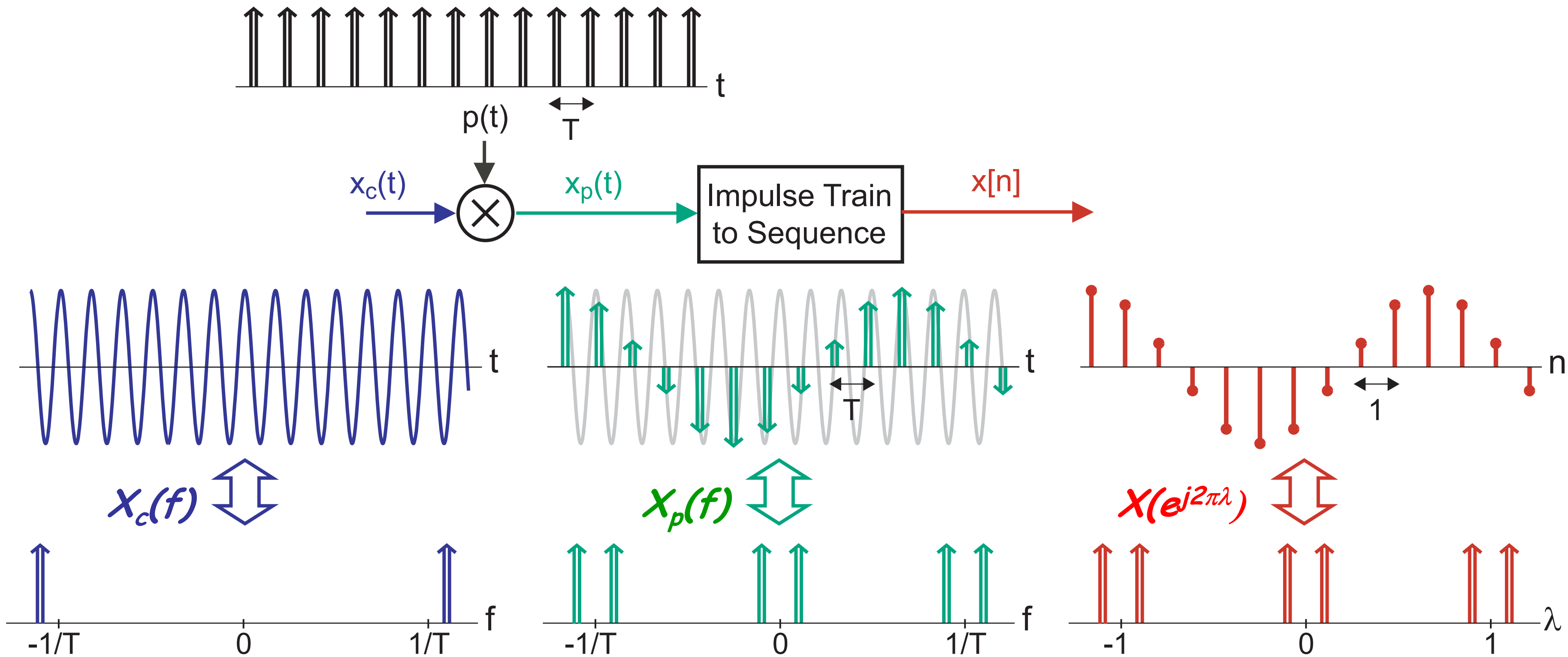
Increase Input Frequency Further …



Sample rate is at *half* the Nyquist rate

- Time domain: resulting sequence now appears as a DC signal!
- Frequency domain: aliasing to DC

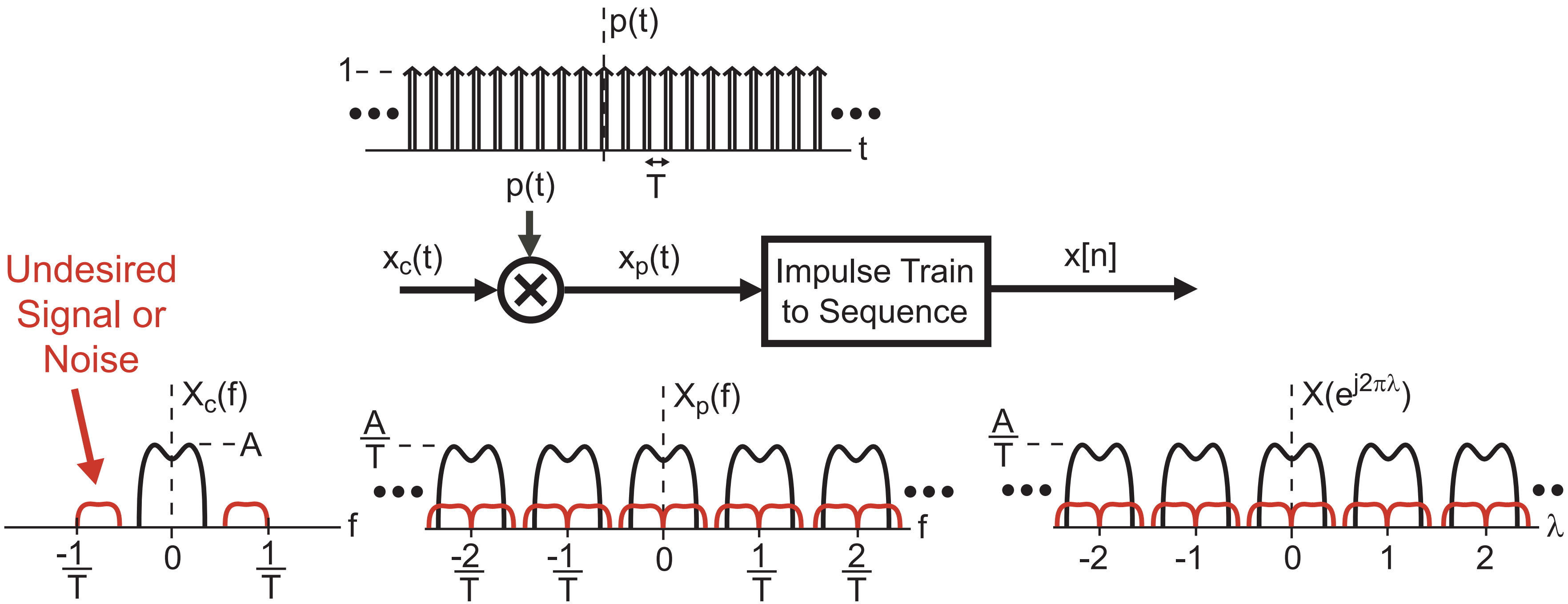
Increase Input Frequency Further …



Sample rate is well *below* the Nyquist rate

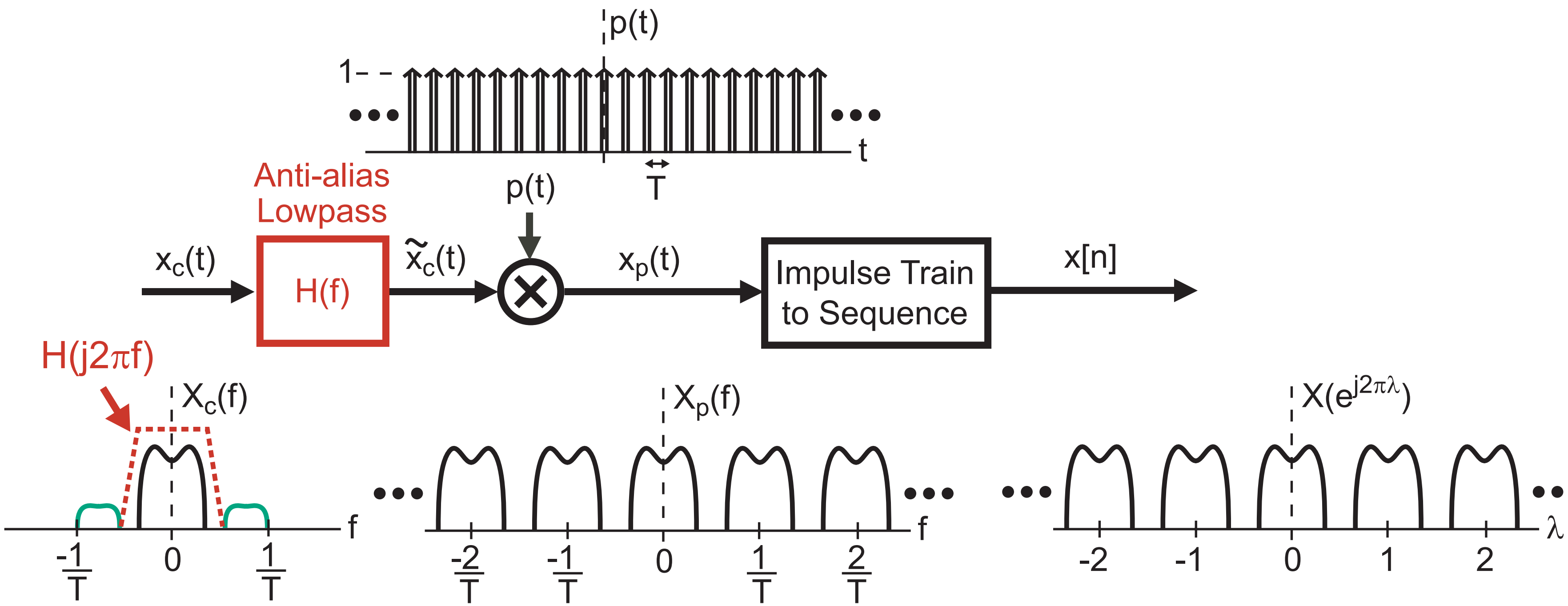
- Time domain: resulting sequence is now a sine wave with a *different* period than the input
- Frequency domain: aliasing to lower frequency

The Issue of High Frequency Noise



- **We typically set the sample rate to be large enough to accommodate full bandwidth of *signal***
- **Real systems often introduce *noise* or other interfering signals at *higher* frequencies**
  - **Sampling causes this noise to *alias* into the desired signal band**

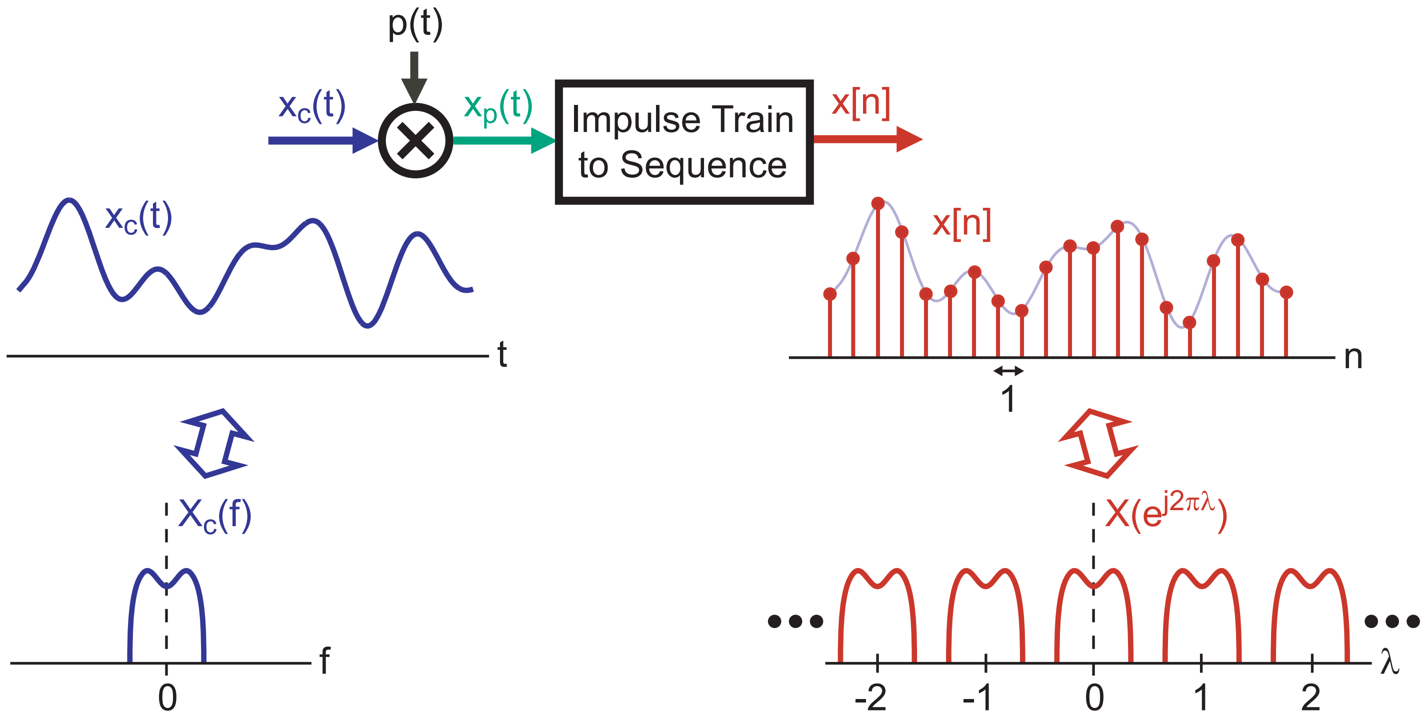
Anti-Alias Filtering



- Practical A-to-D converters include a continuous-time filter *before* the sampling operation
  - Designed to filter out all noise and interfering signals above *1/(2T)* in frequency
  - Prevents aliasing

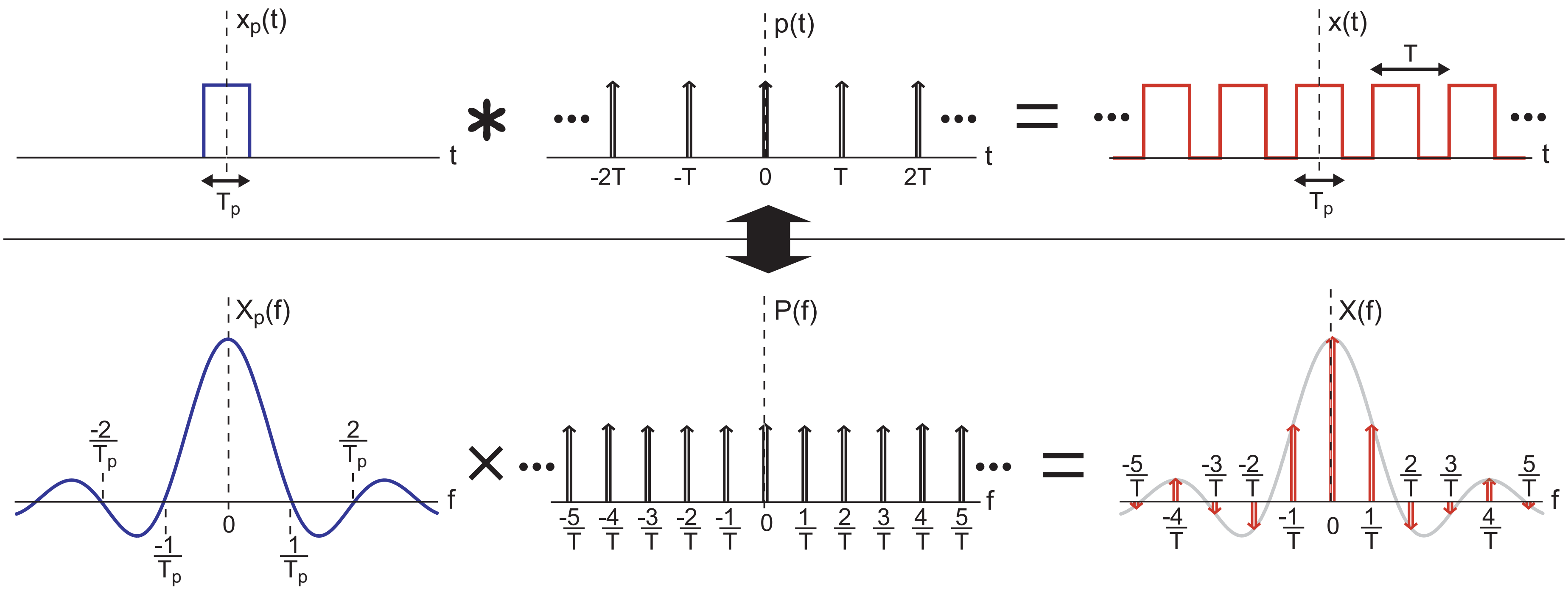
Using the Impulse Train to Compare the FT, DTFT, and Fourier Series

Relationship Between FT and DTFT



	FT	DTFT
Time:	Continuous, Non-Periodic	Discrete, Non-Periodic
Freq:	Non-Periodic, Continuous	Periodic, Continuous

Relationship Between FT and Fourier Series

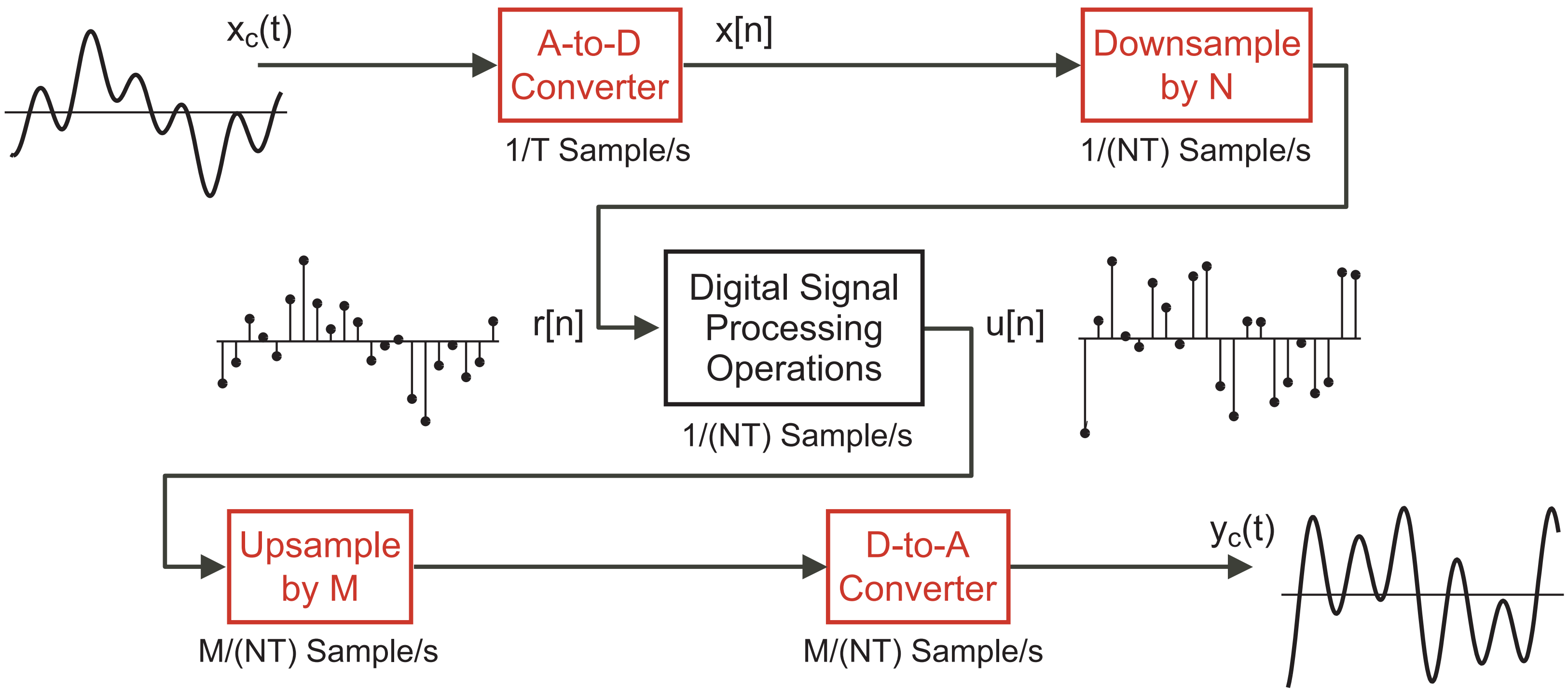


	FT	Fourier Series
Time:	Continuous, Non-Periodic	Continuous, Periodic
Freq:	Non-Periodic, Continuous	Non-Periodic, Discrete

Downsampling, Upsampling, and Reconstruction

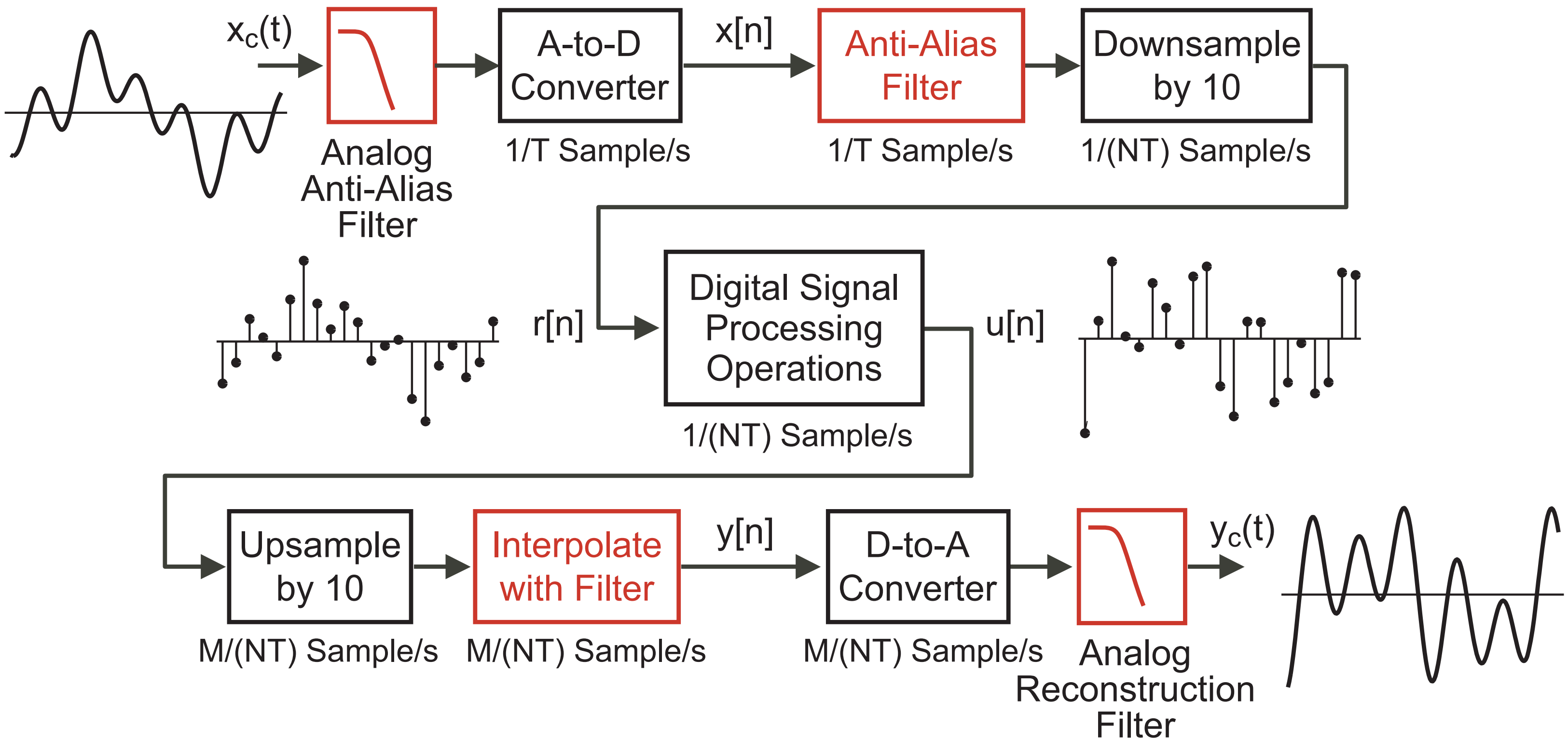
- A-to-D and its relation to sampling
- Downsampling and its relation to sampling
- Upsampling and interpolation
- D-to-A and reconstruction filtering
- Filters and their relation to convolution

Digital Processing of Analog Signals



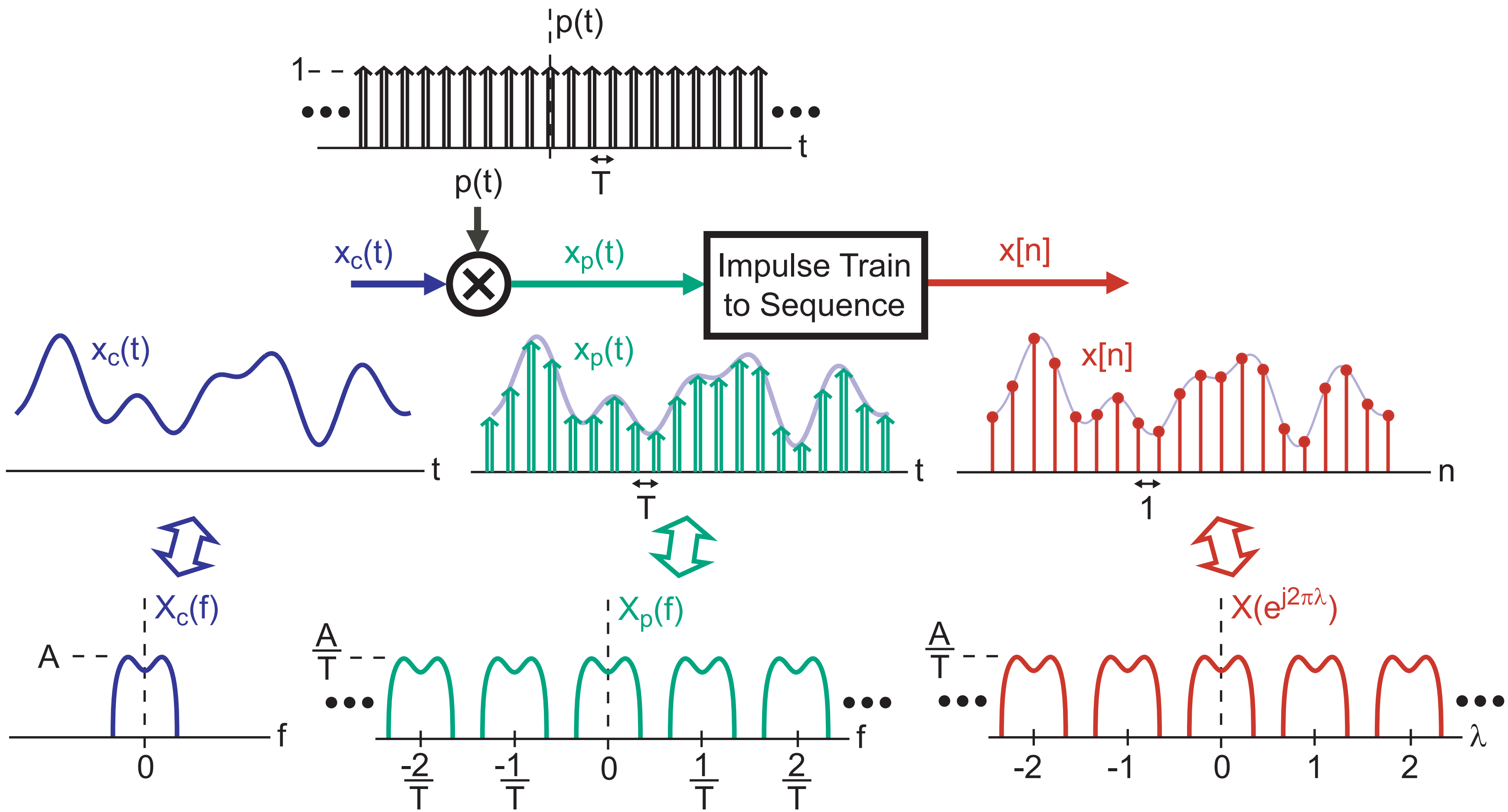
- ***Digital*** **circuits can perform very complex processing of *analog* signals, but require**
  - **Conversion of analog signals to the digital domain**
  - **Conversion of digital signals to the analog domain**
  - **Downsampling and upsampling to match sample rates of A-to-D, digital processor, and D-to-A**

Inclusion of Filtering Operations



- **A-to-D and downsampler require *anti-alias* filtering**
  - **Prevents aliasing**
- **D-to-A and upsampler require *interpolation* (i.e., *reconstruction*) filtering**
  - **Provides `smoothly' changing waveforms**

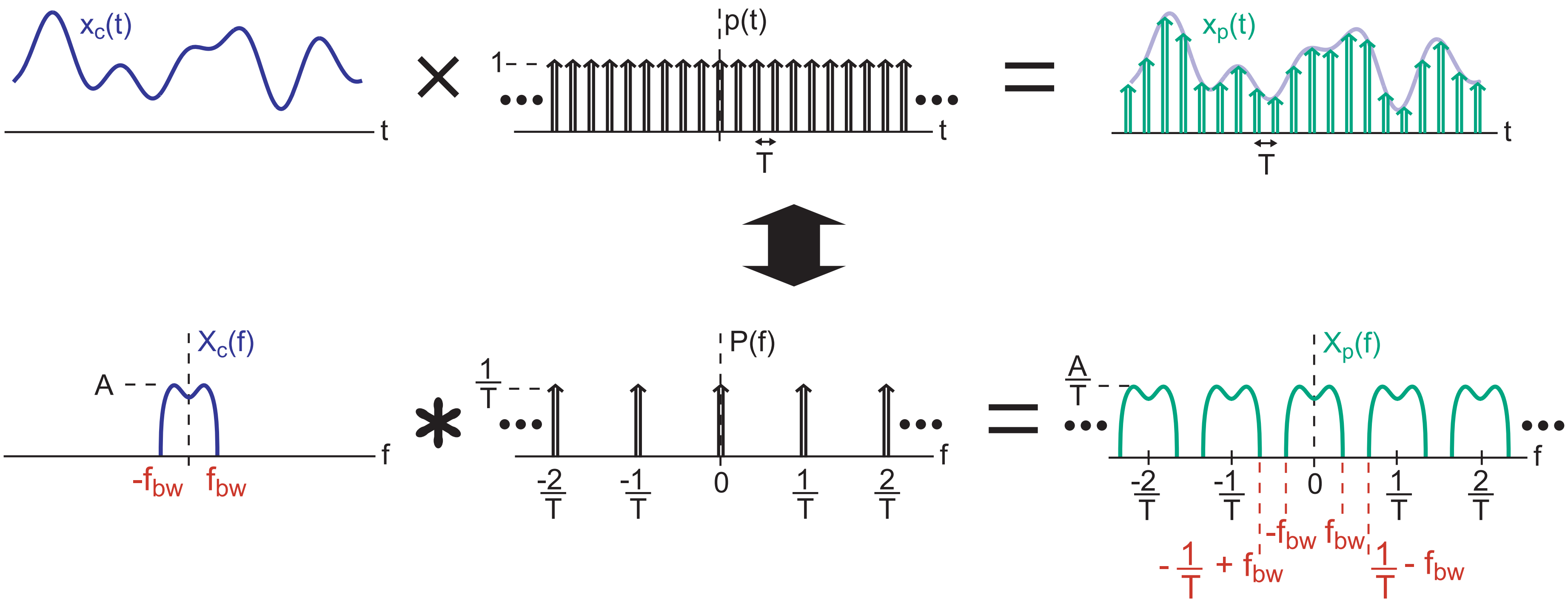
Summary of Sampling Process (Review)



- Sampling leads to periodicity in frequency domain

We need to avoid overlap of replicated signals in frequency domain (i.e., aliasing)

The Sampling Theorem (Review)

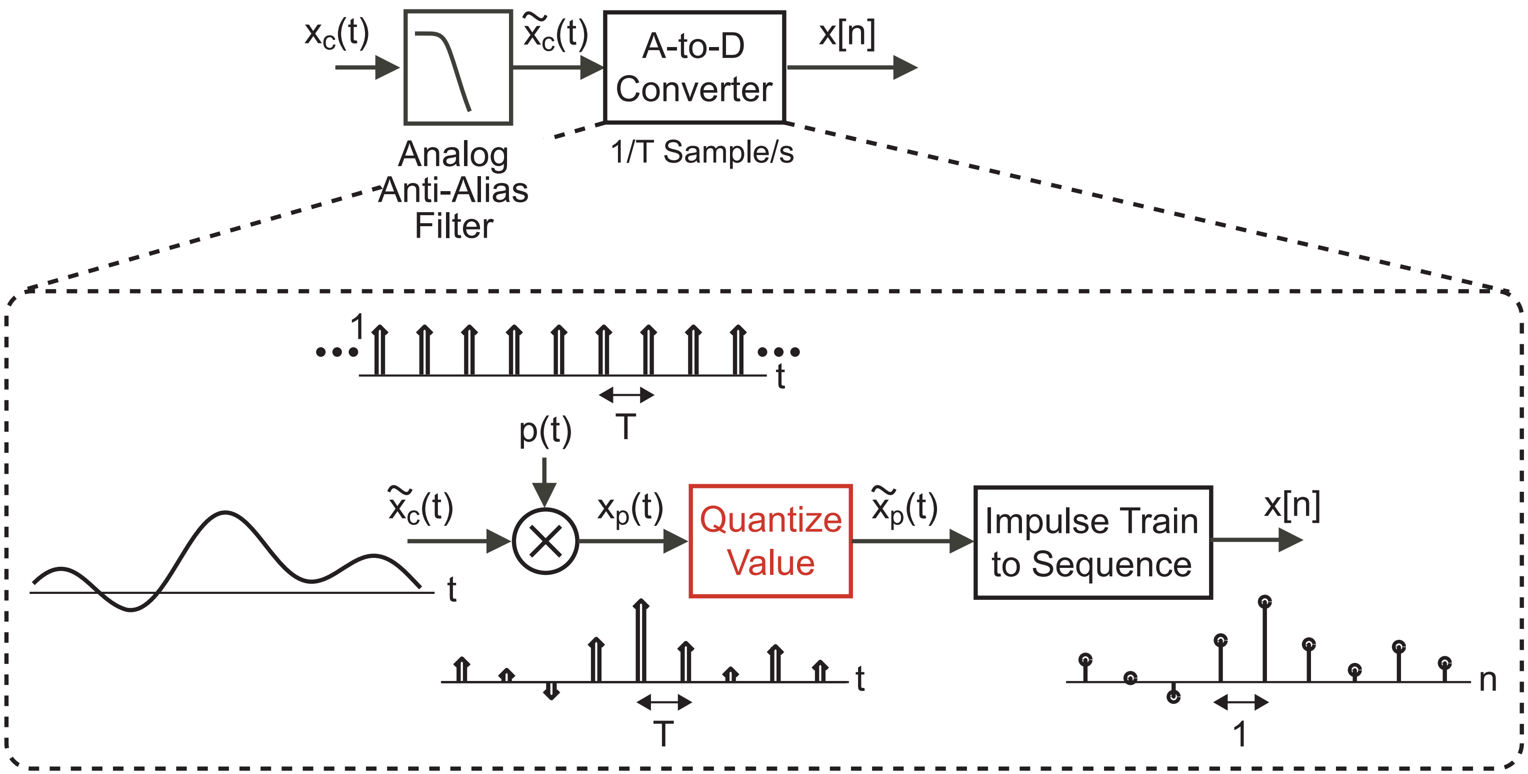


- **Overlap in frequency domain (i.e., aliasing) is avoided if:**

$$\frac{1}{T} - f_{bw} \geq f_{bw} \quad \Rightarrow \quad \boxed{\frac{1}{T} \geq 2 f_{bw}}$$

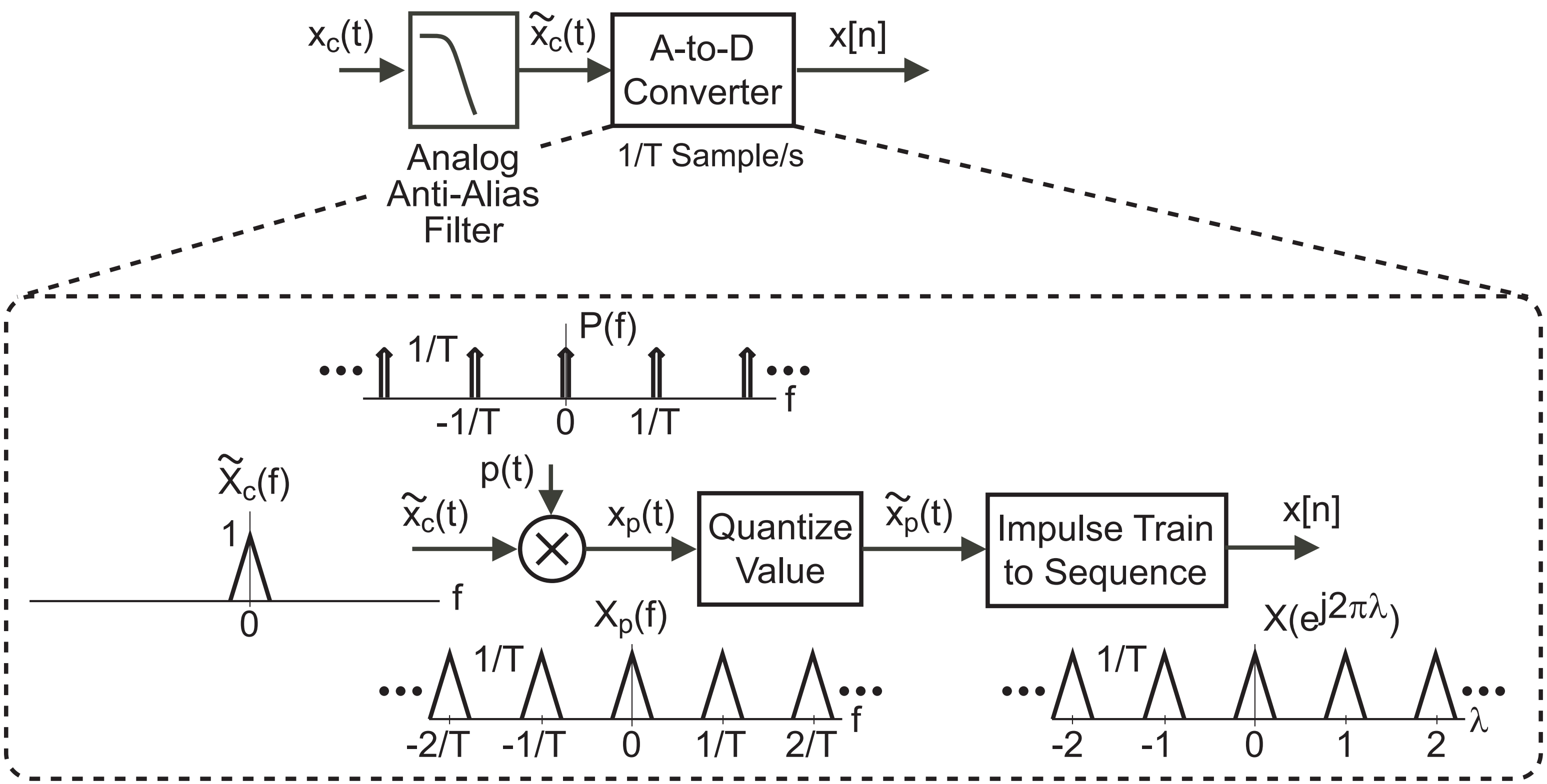
- **We refer to the minimum *1/T* that avoids aliasing as the *Nyquist* sampling frequency**

A-to-D Converter



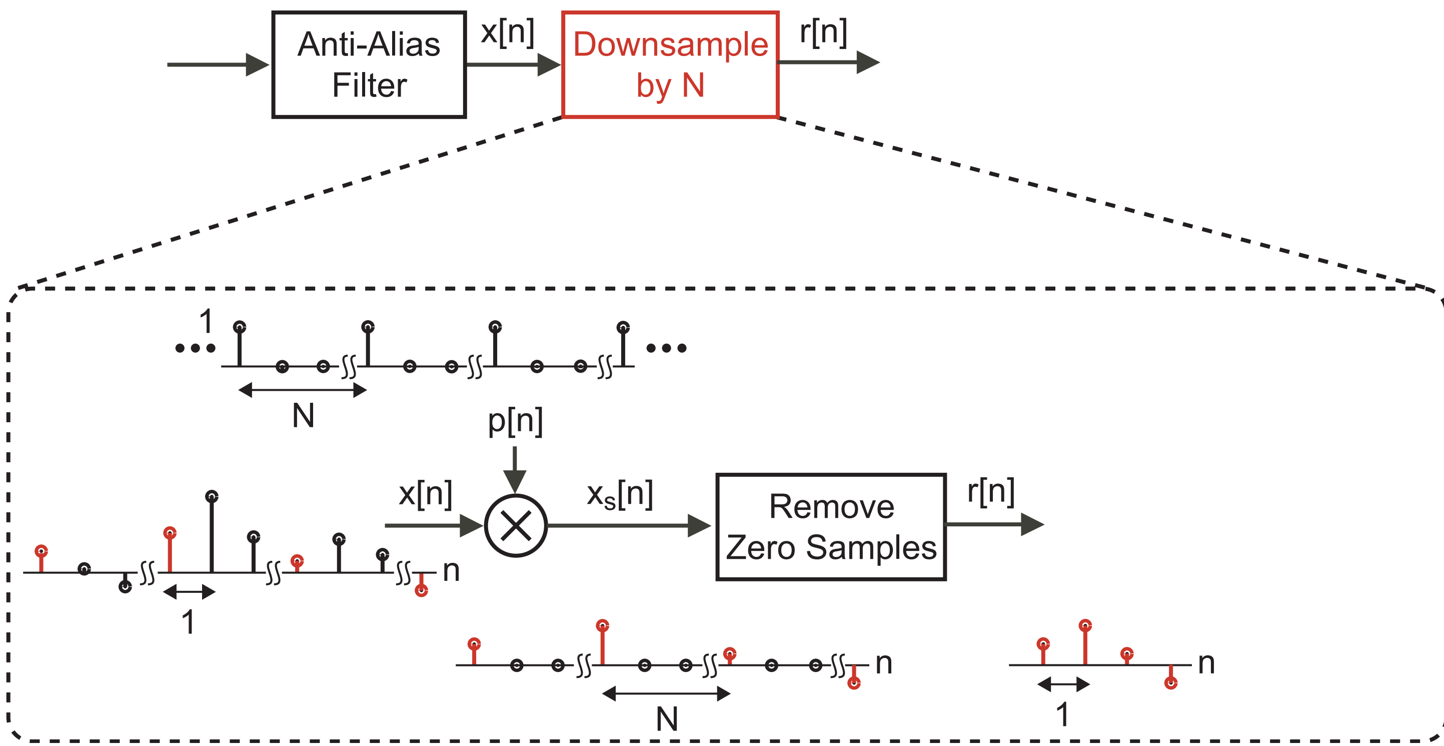
- Operates using both a *sampler* and *quantizer*
  - Sampler converts *continuous-time* input signal into a *discrete-time* sequence
  - Quantizer converts *continuous-valued* signal/sequence into a *discrete-valued* signal/sequence
    - Introduces *quantization noise* as discussed in Lab 4

Frequency Domain View of A-to-D



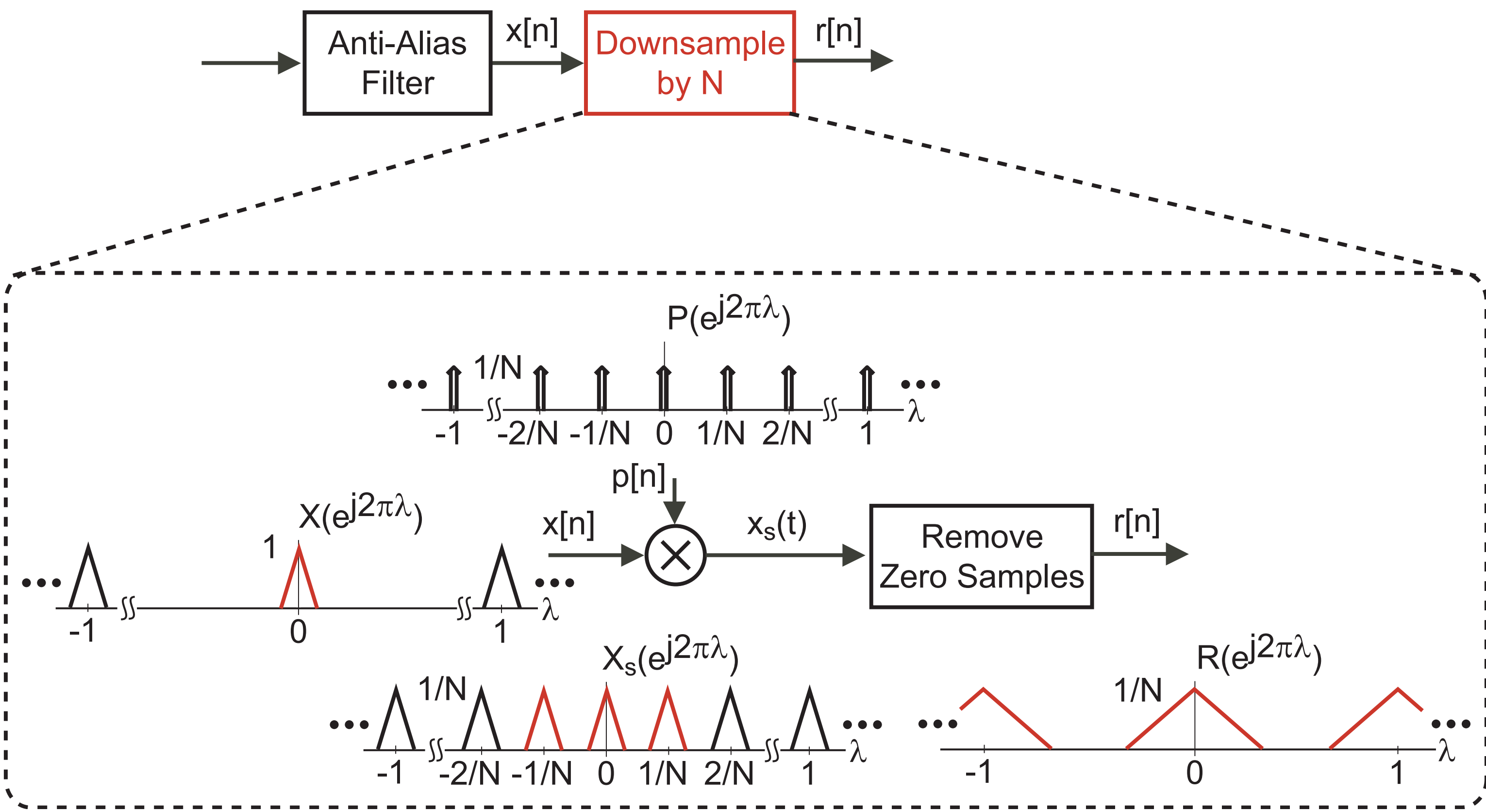
- **Analysis of A-to-D same as for sampler**
  - **For simplicity, we will ignore the influence of quantization noise in our picture analysis**
    - **In lab 4, we will explore the influence of quantization noise using Matlab**

Downsampling



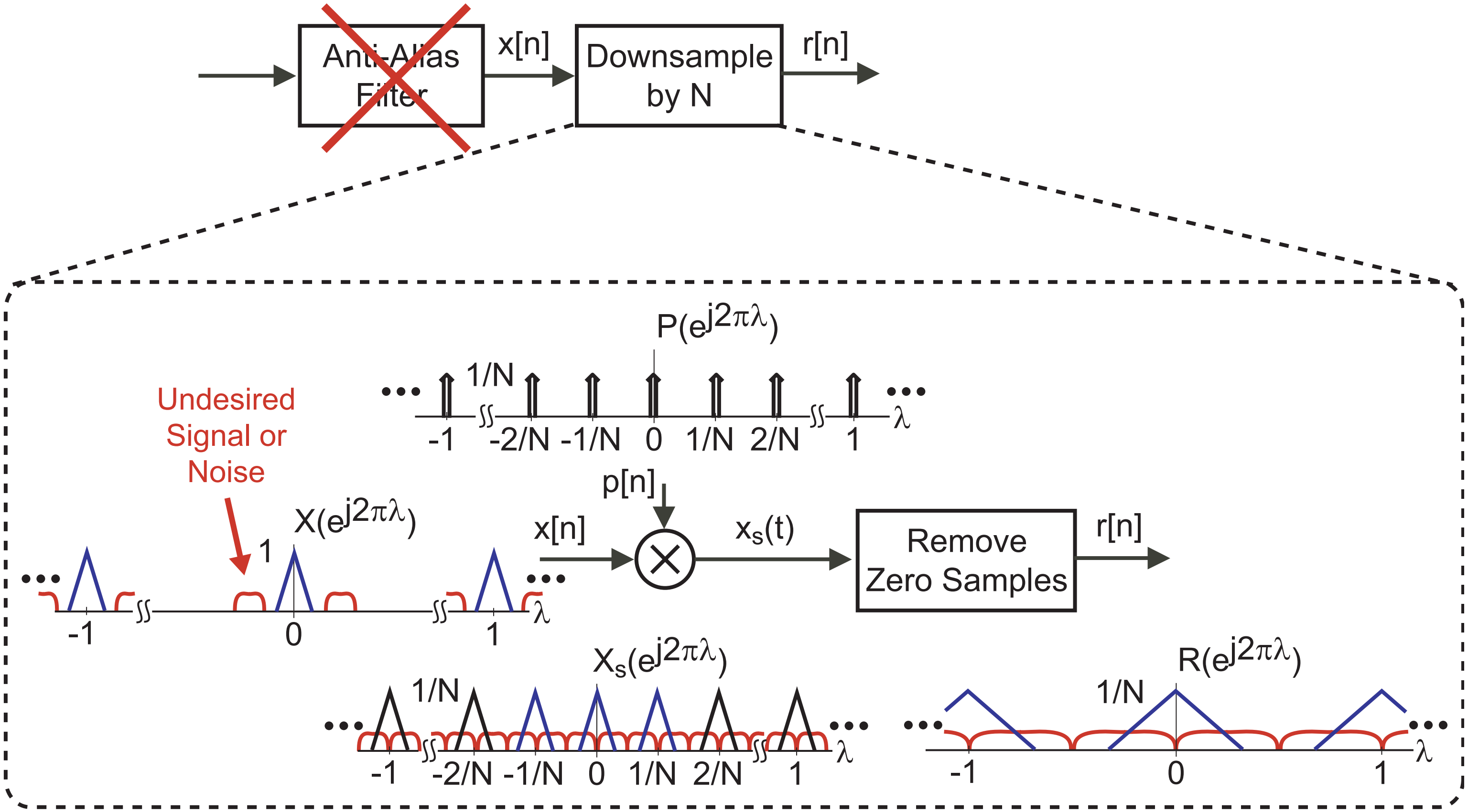
- Similar to sampling, but operates on *sequences*
- Analysis is simplified by breaking into two steps
  - *Multiply* input by impulse sequence of period *N* samples
  - Remove all samples of $x_s[n]$ associated with the zero-valued samples of the impulse sequence, *p[n]*
    - Amounts to *scaling* of time axis by factor *1/N*

Frequency Domain View of Downsampling



- Multiplication by impulse sequence leads to replicas of input transform every *1/N Hz* in frequency
- Removal of zero samples (i.e., scaling of time axis) leads to scaling of frequency axis by factor *N*

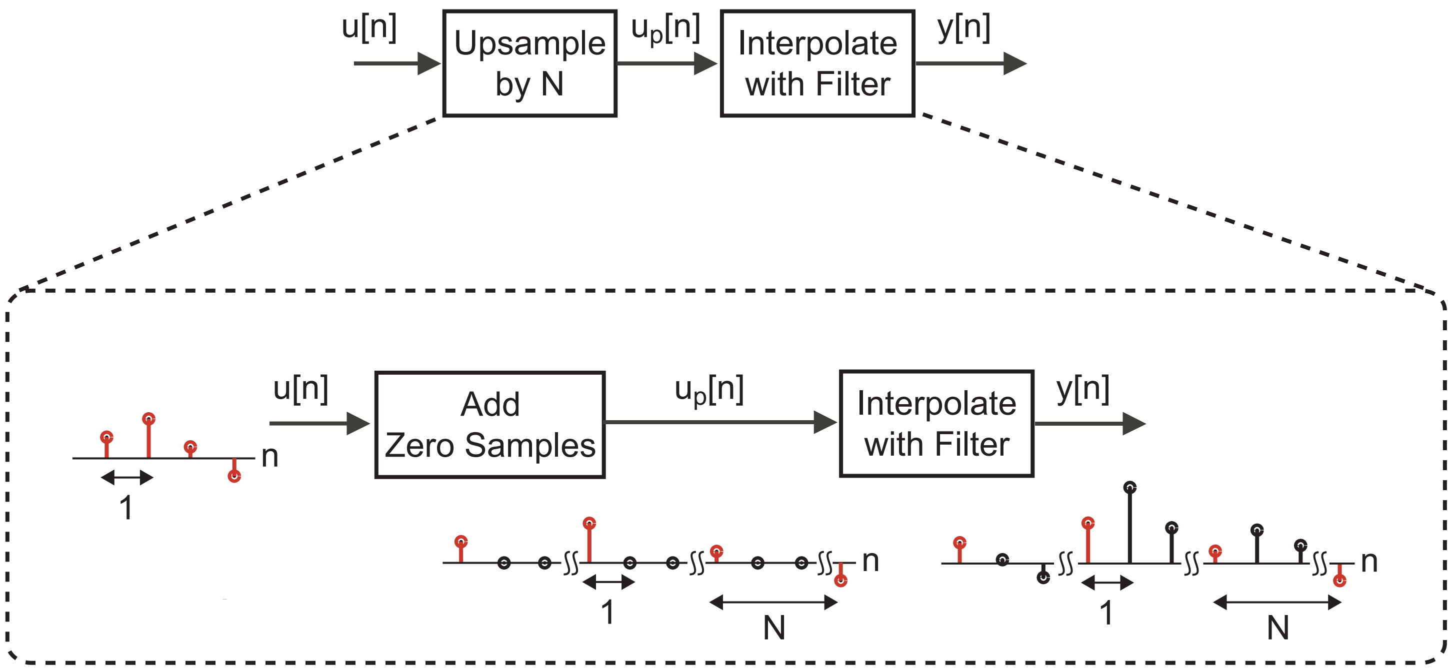
The Need for Anti-Alias Filtering



- Removal of anti-alias filter would allow undesired signals or noise to alias into desired signal band

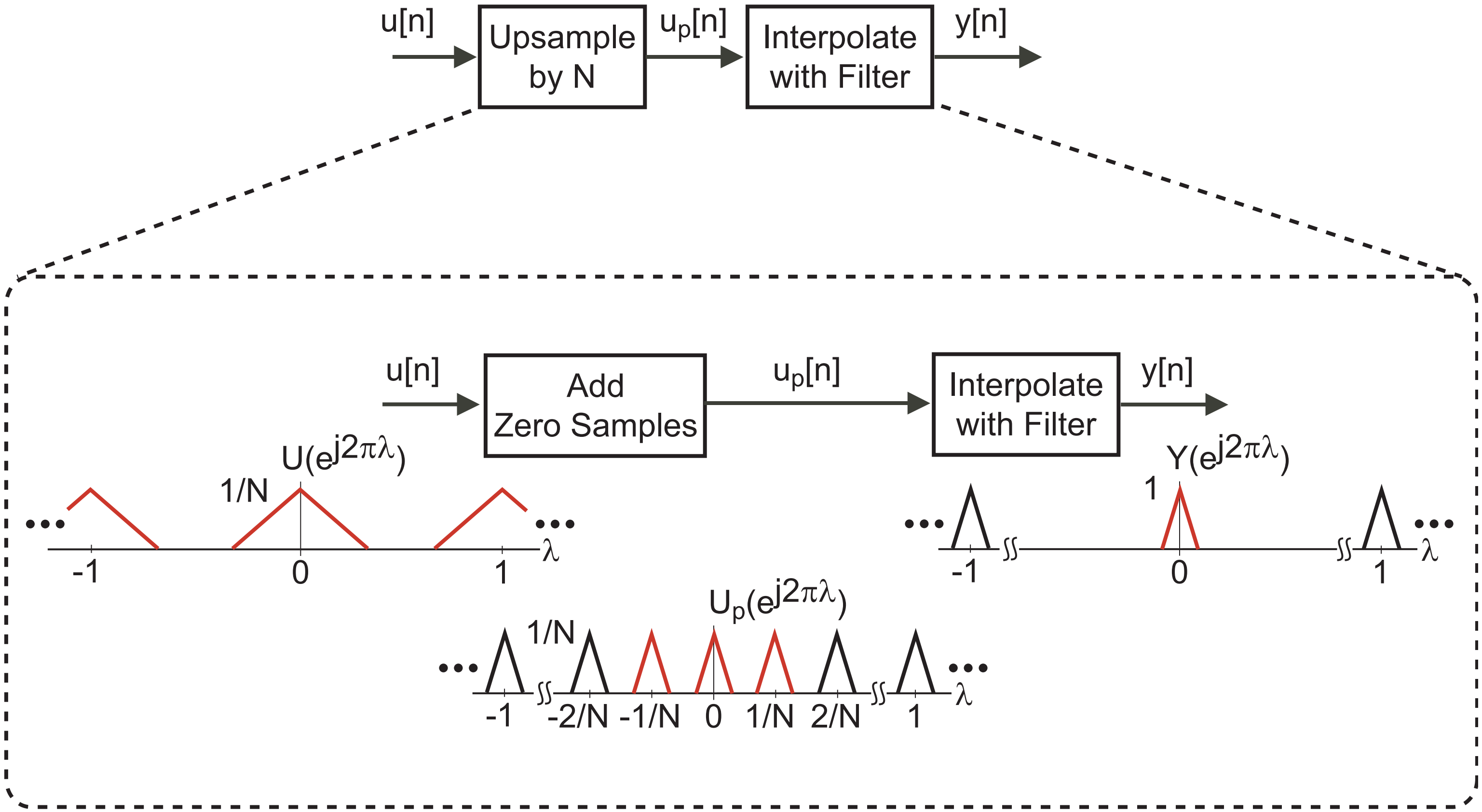
What is the appropriate bandwidth of the anti-alias lowpass filter?

Upsampler



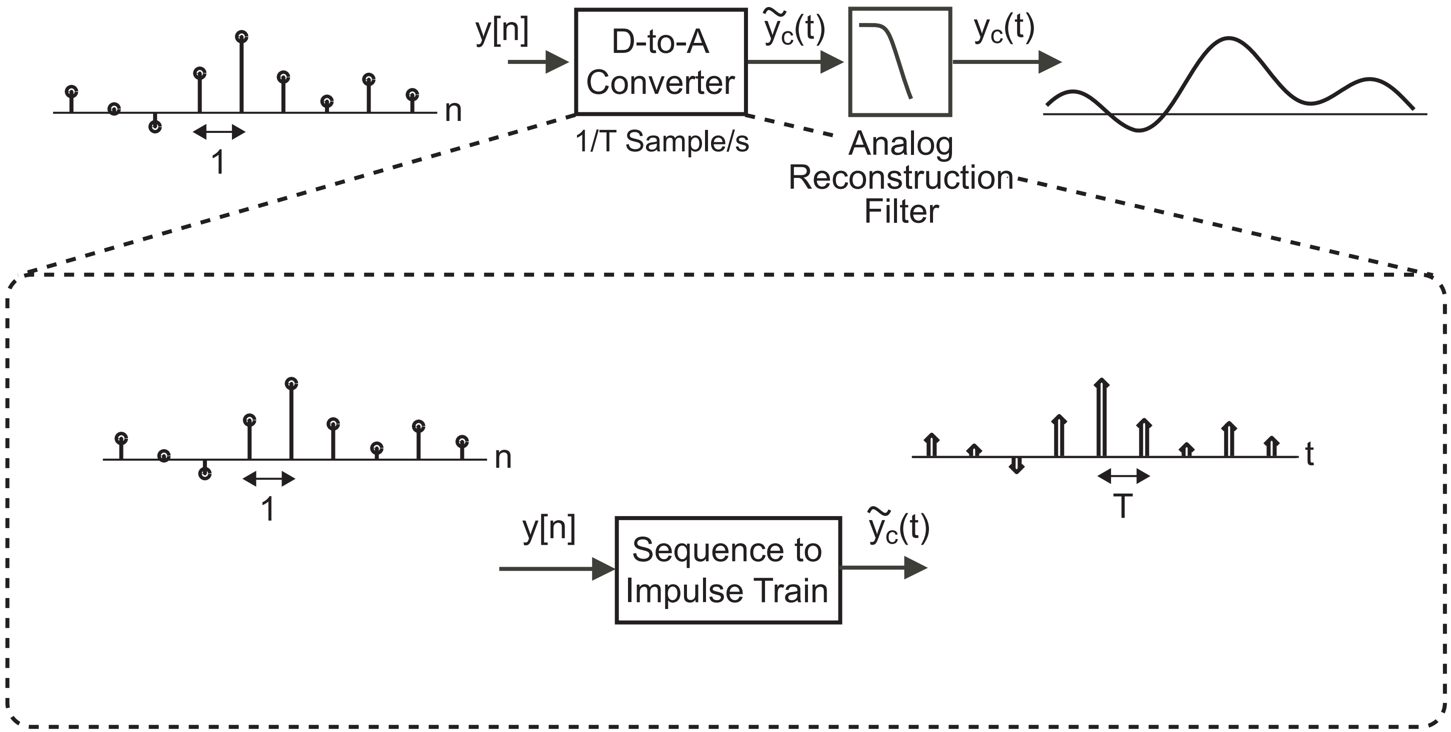
- **Consists of two operations**
  - **Add *N-1* zero samples between every sample of the input**
    - **Effectively scales time axis by factor *N***
  - **Filter the resulting sequence, $u_p[n]$, in order to create a *smoothly* varying set of sequence samples**
    - **Proper choice of the filter leads to *interpolation* between the non-zero samples of sequence $u_p[n]$ (discussed in Lab 5)**

Frequency Domain View of Upsampling



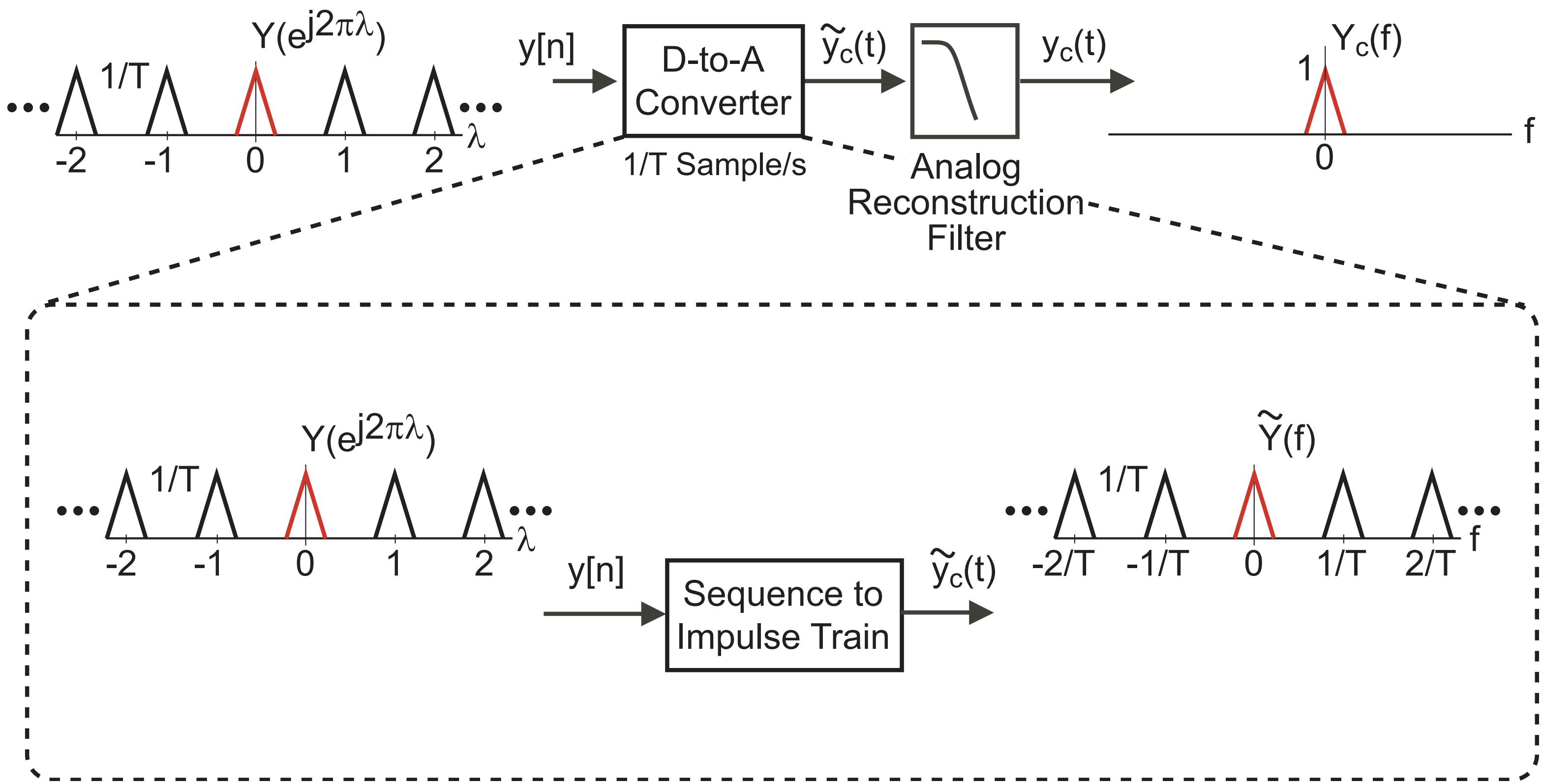
- **Addition of zero samples (scaling of time axis) leads to scaling of frequency axis by factor *1/N***
- **Interpolation filter removes all replicas of the signal transform *except* for the baseband copy**

D-to-A Converter



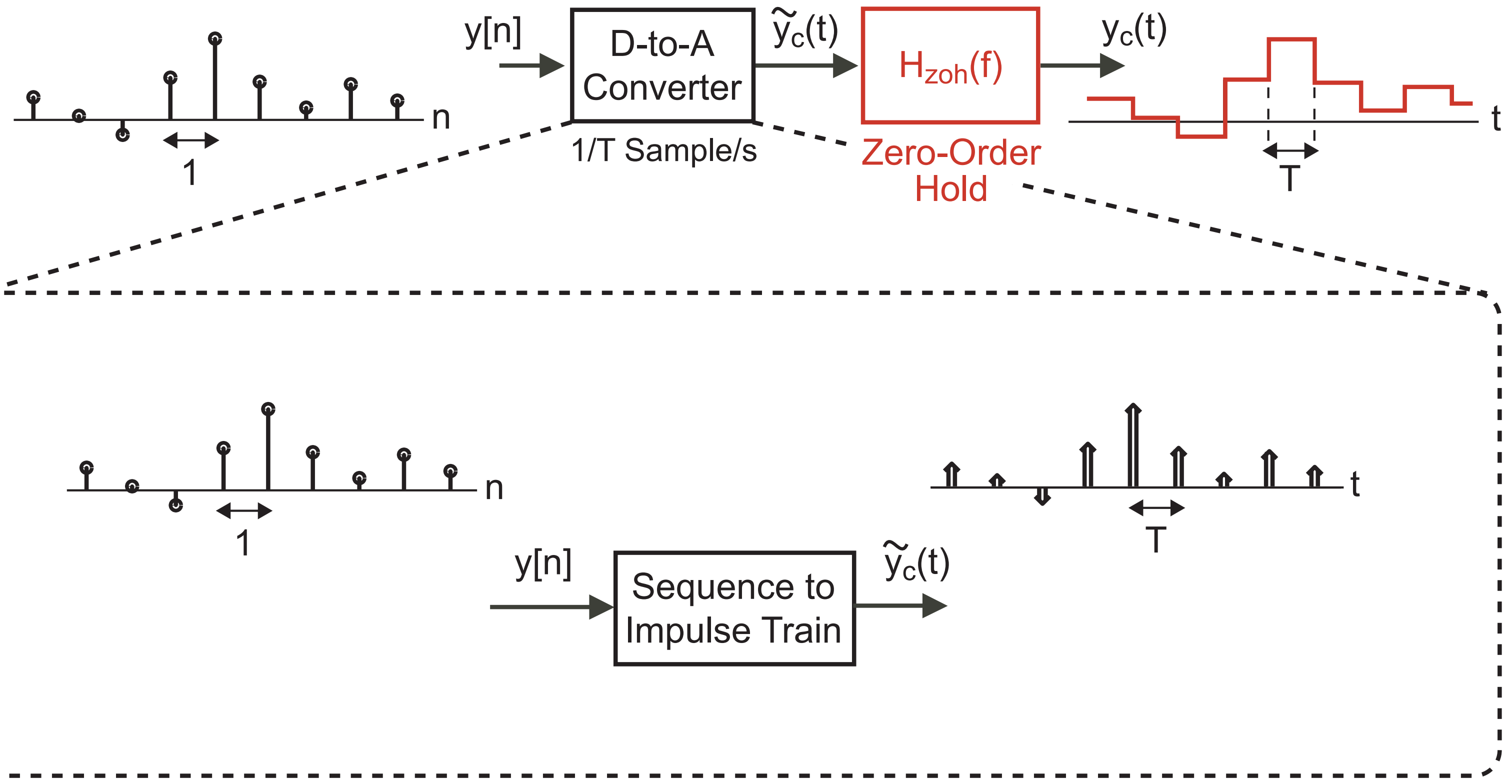
- **Simple analytical model includes two operations**
  - **Convert input sequence samples into corresponding impulse train**
  - **Filter impulse train to create a smoothly varying signal**
    - **Proper choice of the *reconstruction filter* leads to *interpolation* between impulse train values**

Frequency Domain View of D-to-A



- **Conversion from sequence to impulse train amounts to scaling the frequency axis by sample rate of D-to-A (*1/T*)**
- **Reconstruction filter removes all replicas of the signal transform *except* for the baseband copy**

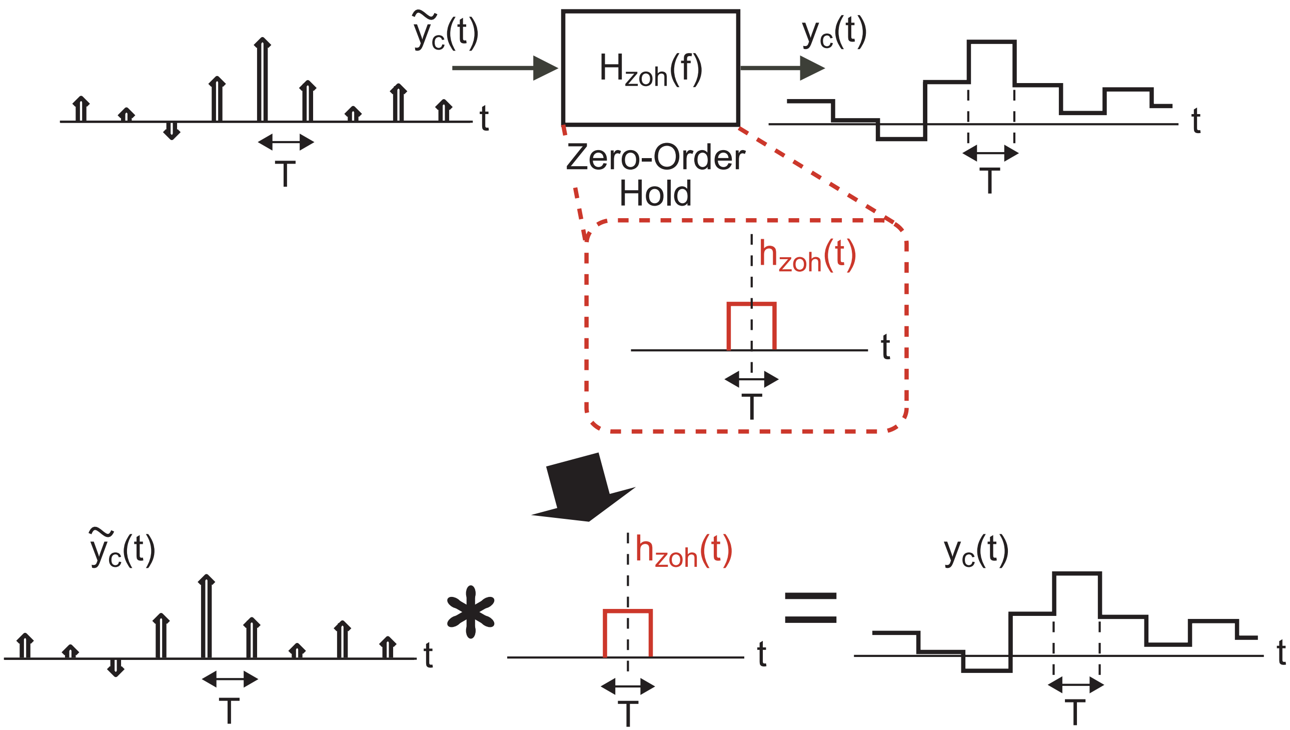
A Common Reconstruction Filter



- Zero-order hold circuit operates by maintaining the impulse value across the D-to-A sample period
  - Easy to implement in hardware

How do we analyze this?

Filtering is Convolution in Time

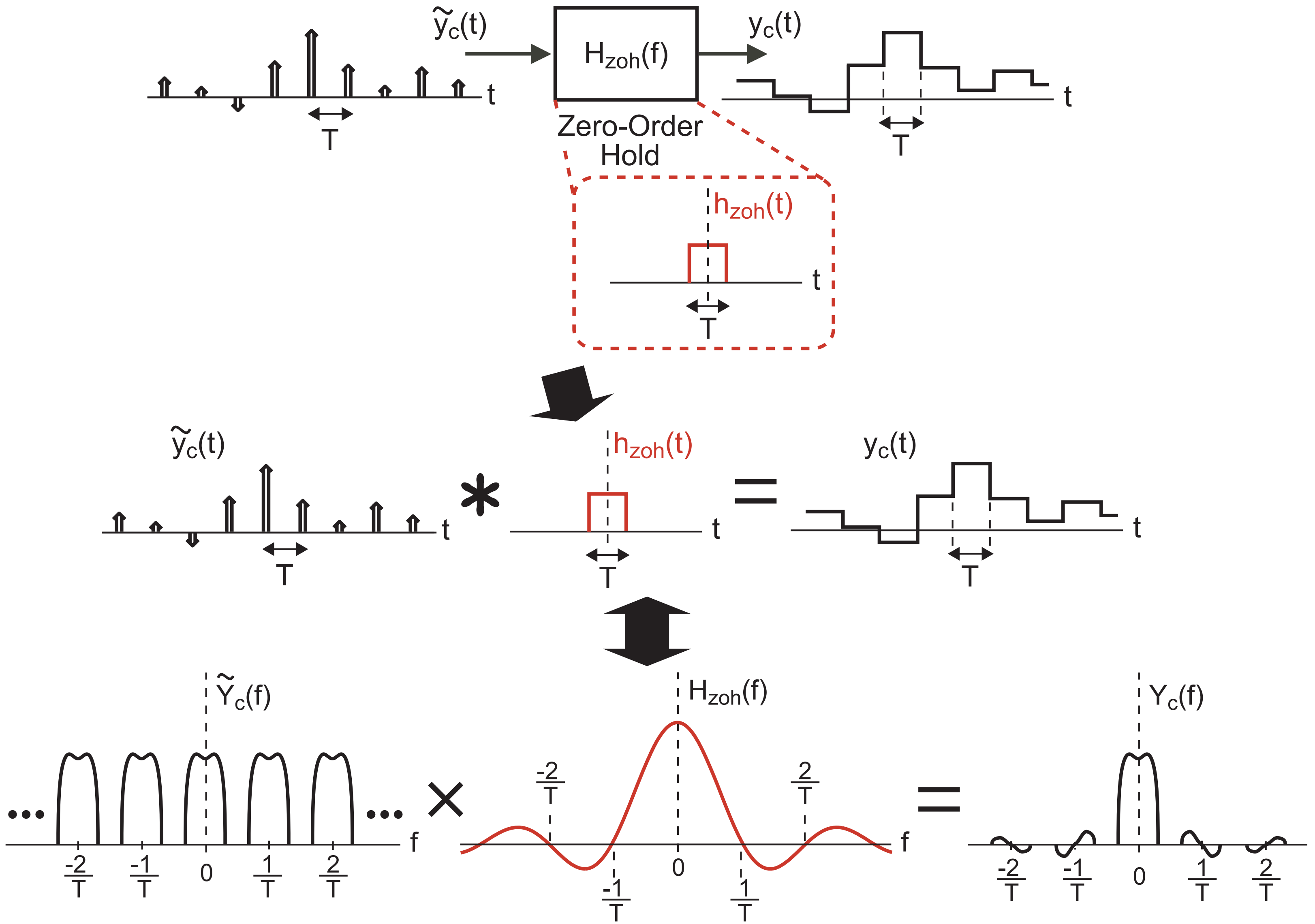


- **Recall that *multiplication* in *frequency* corresponds to *convolution* in *time***

$$x(t) * y(t) \Leftrightarrow X(f)Y(f)$$

- **Filtering corresponds to convolution in time between the input and the filter *impulse response***

Frequency Domain View of Filtering

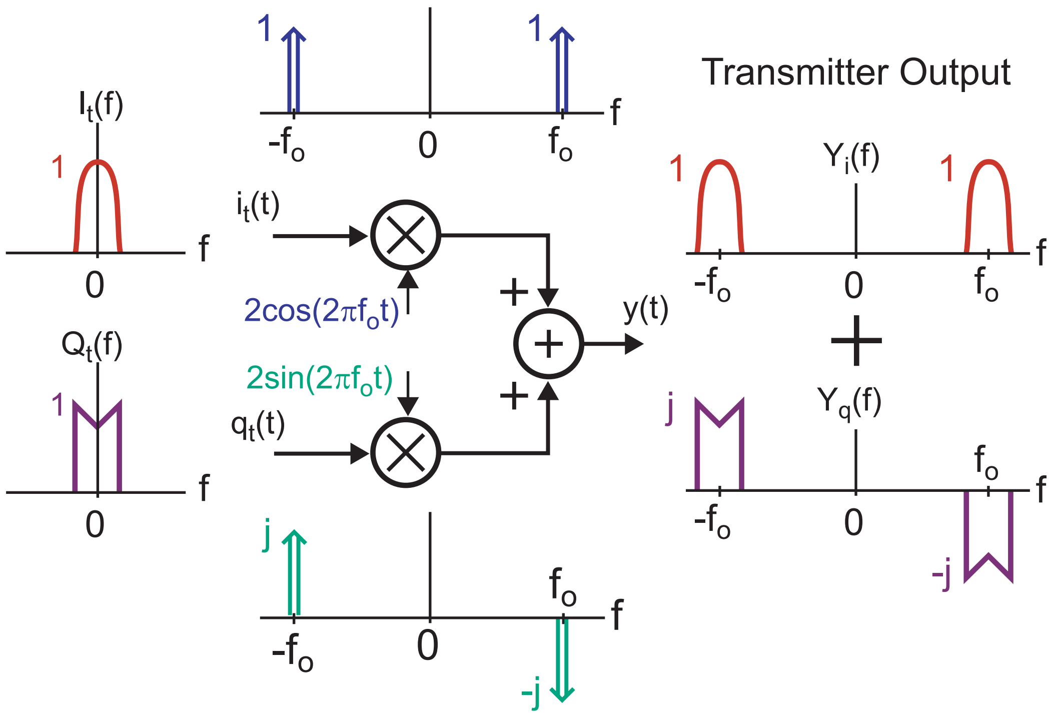


- Zero-order hold is not a great filter, but it's simple…

Advantages of Digital Processing

- Digital components correct small analog errors at each processing step
  - We can build large, reliable systems despite non-ideal components and the presence of bounded noise
- We can accommodate more precision by representing information with longer sequences of symbols
  - Except for the conversion steps, we can use simple digital components do achieve arbitrary precision in processing
- We abstract out the notion of "real time" when converting to sequences of discrete values
  - The speed of intervening digital processing steps is independent of the speed of conversion steps (e.g., we can combine many analog streams into a single high-speed digital stream).

Review of Analog I/Q Modulation

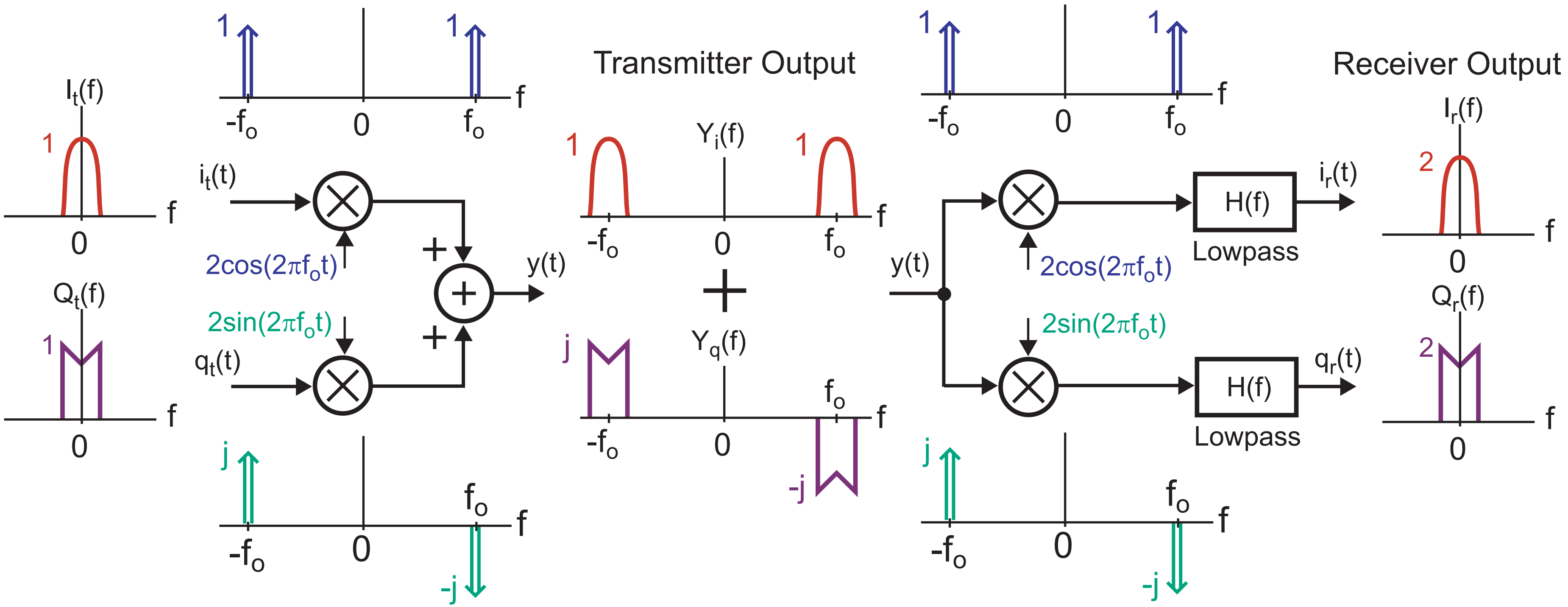


I stands for *in-phase* component

Q stands for *quadrature* component

- Consider modulating with both a cosine and sine wave and then adding the results
  - This is known as I/Q modulation
- The I/Q signals occupy the same frequency band, but one is *real* and one is *imaginary*
  - We can recover *both* of these signals

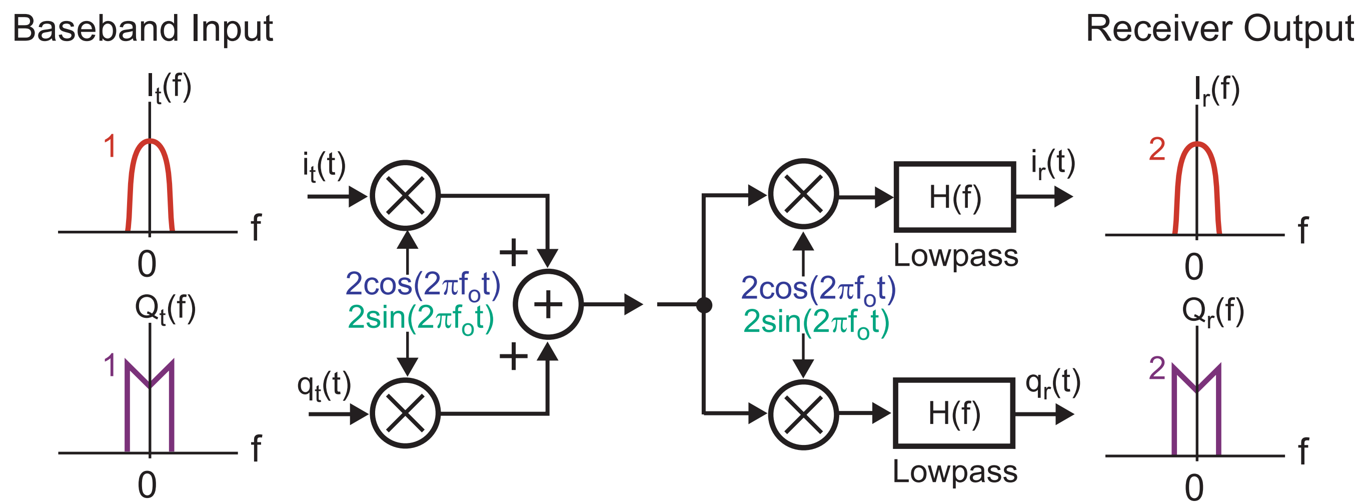
Review of Analog I/Q Demodulation



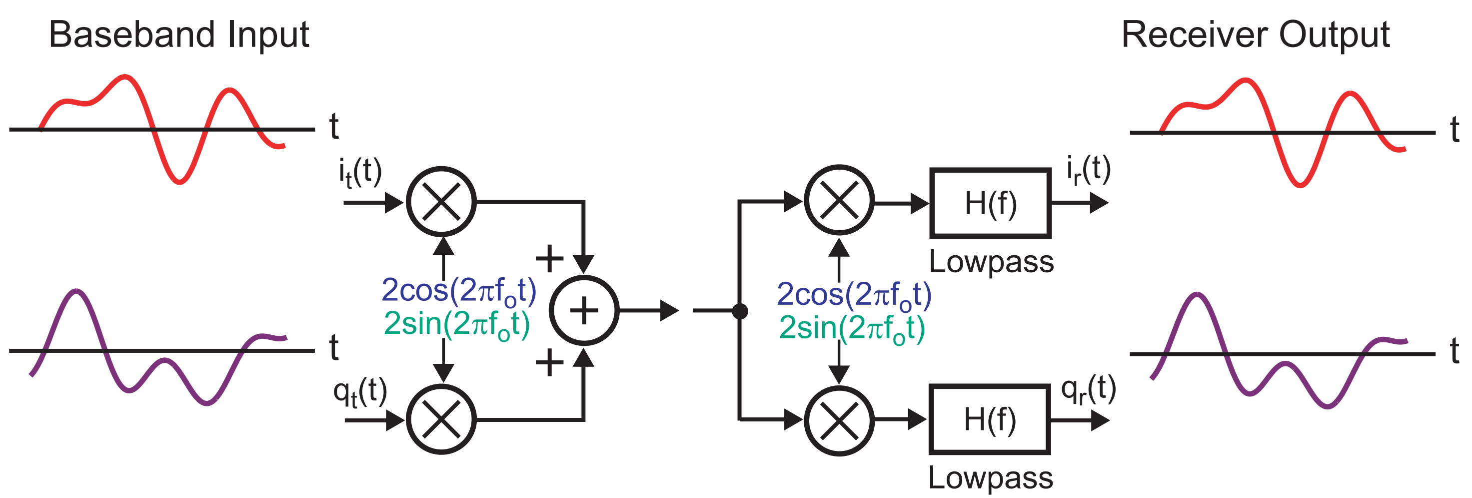
- Demodulate with *both* a cosine and sine wave
  - Both I and Q channels are recovered!
- I/Q modulation allows twice the amount of *information* to be sent compared to basic AM modulation with same *bandwidth*

Summary of Analog I/Q Demodulation

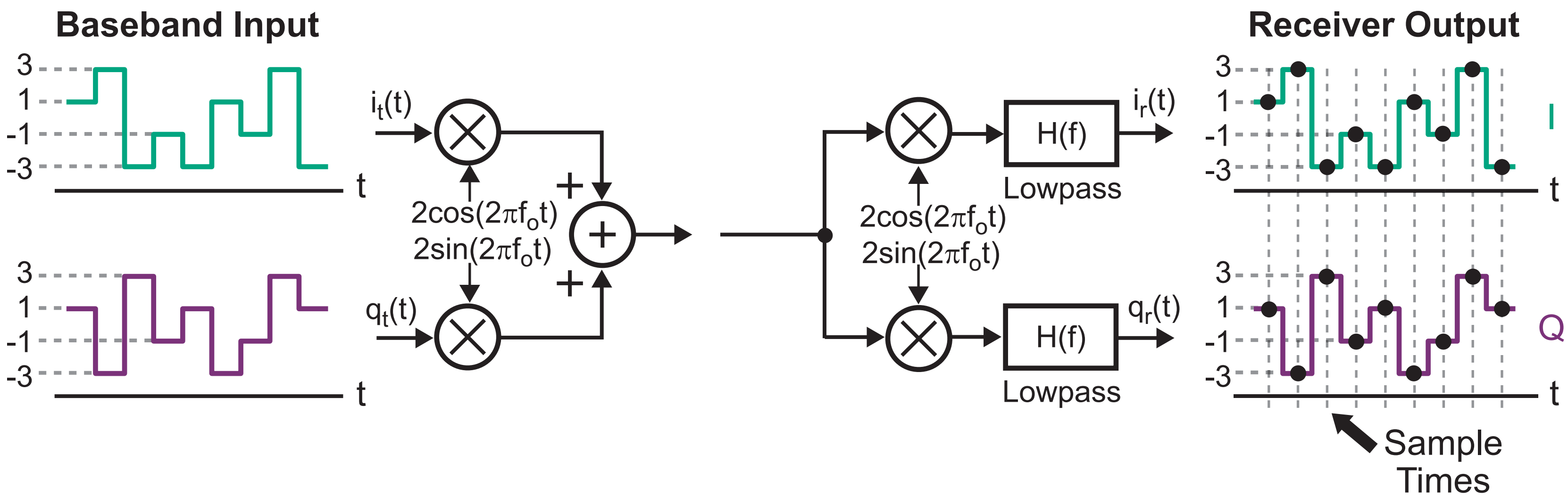
- **Frequency domain view**



- **Time domain view**



Digital I/Q Modulation

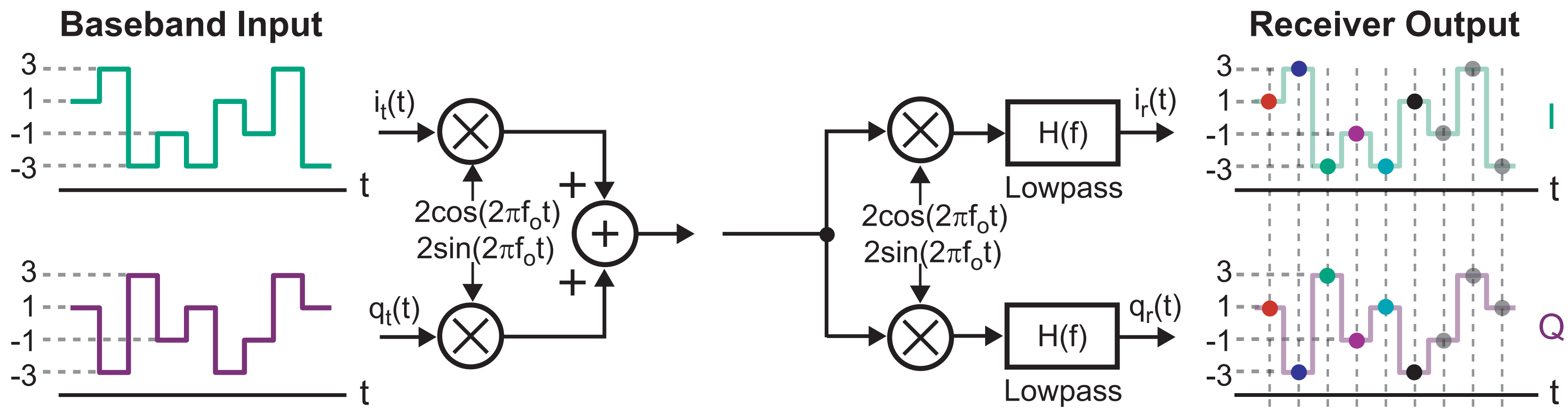


- **Leverage *analog communication channel* to send *discrete-valued symbols***
  - Example: send symbol from set {-3,-1,1,3} on both I and Q channels each *symbol period*
- **At receiver, sample I/Q waveforms every symbol period**
  - Associate each sampled I/Q value with symbols from set {-3,-1,1,3} on both I and Q channels

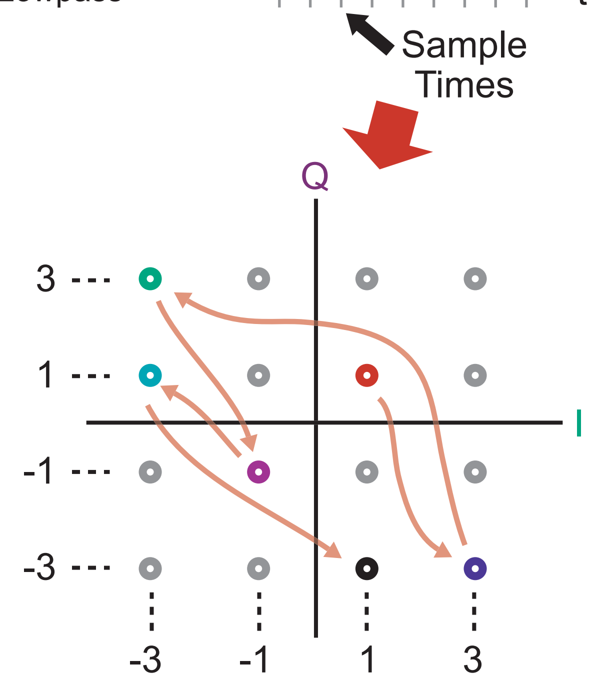
Advantages of going Digital

- Allows information to be "packetized"
  - Can compress information in time and efficiently send as packets through network
  - In contrast, analog modulation requires "circuit-switched" connections that are continuously available
    - Inefficient use of radio channel if there is "dead time" in information flow
- Allows error correction to be achieved
  - Less sensitivity to radio channel imperfections
- Enables compression of information
  - More efficient use of channel
- Supports a wide variety of information content
  - Voice, text and email messages, video can all be represented as digital bit streams

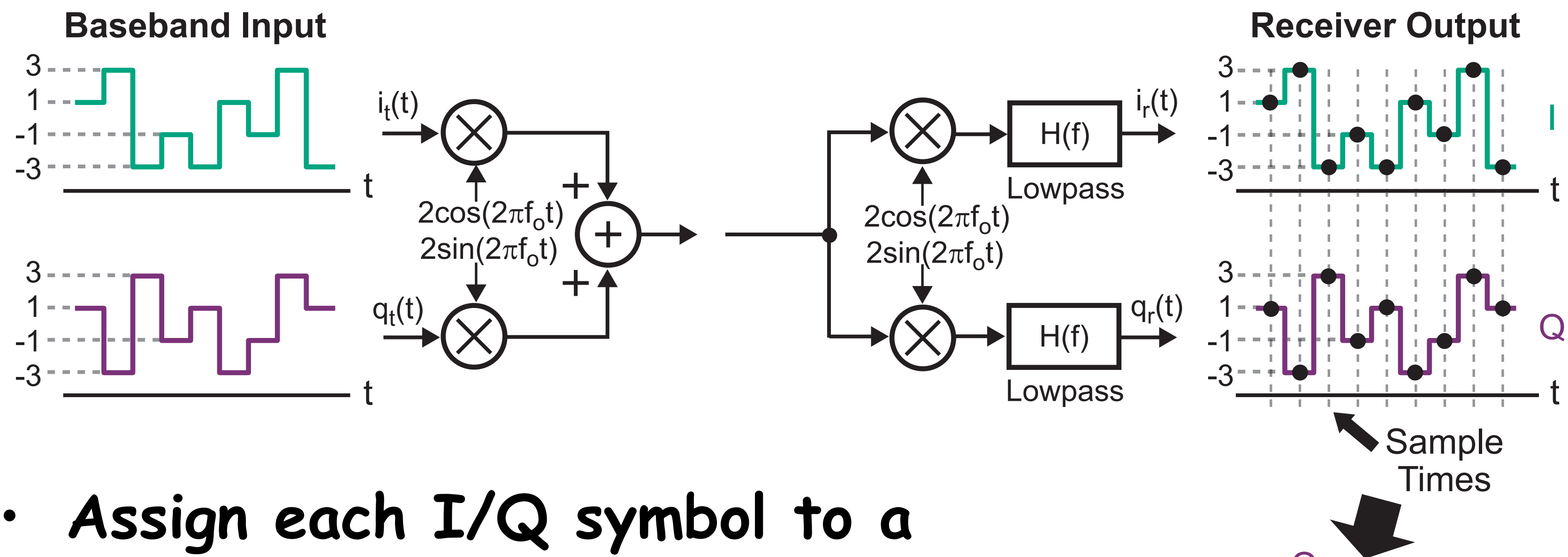
Constellation Diagrams



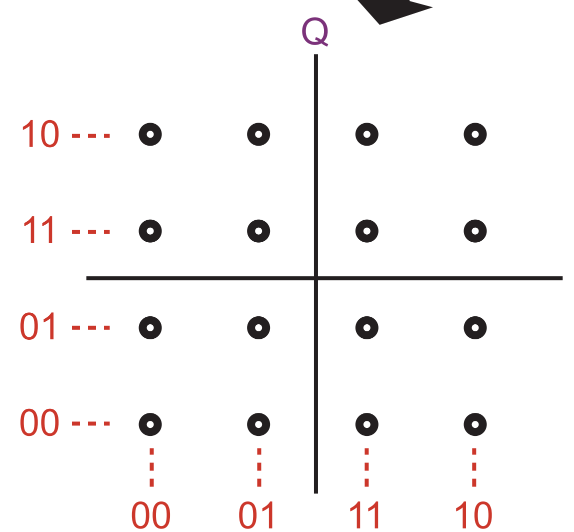
- Plot I/Q samples on x-y axis
  - Example: sampled I/Q value of {1,-3} forms a dot at x=1, y=-3
  - As more samples are plotted, constellation diagram eventually displays all possible symbol values
- Constellation diagram provides a sense of how easy it is to distinguish between different symbols



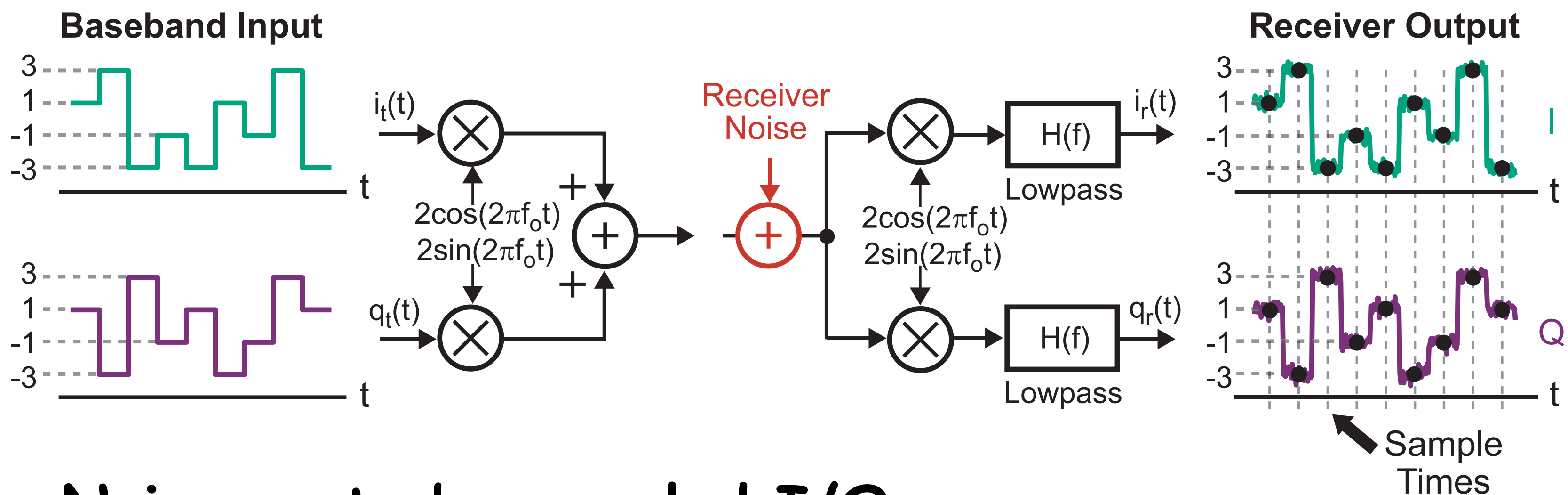
Sending Digital Bits



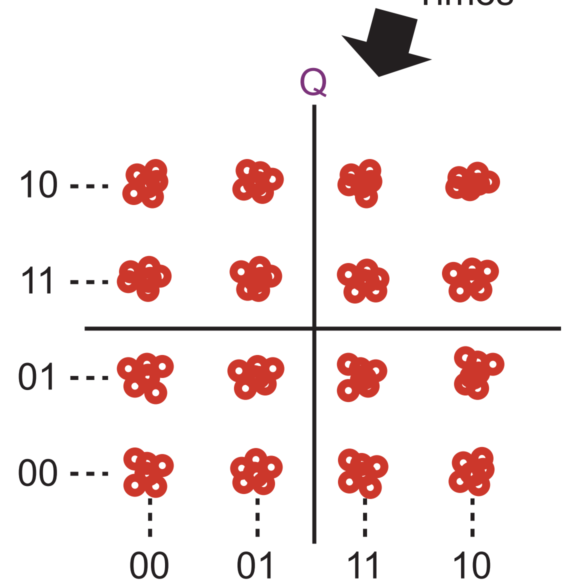
- Assign each I/Q symbol to a set of digital bits
  - Example: I/Q = {1,3} translates to bits of 1110
  - Gray coding minimizes *bit errors* when symbol errors are made
    - Example: I/Q = {1,1} translates to bits of 1010
      - Only one bit change from I/Q = {1,3}



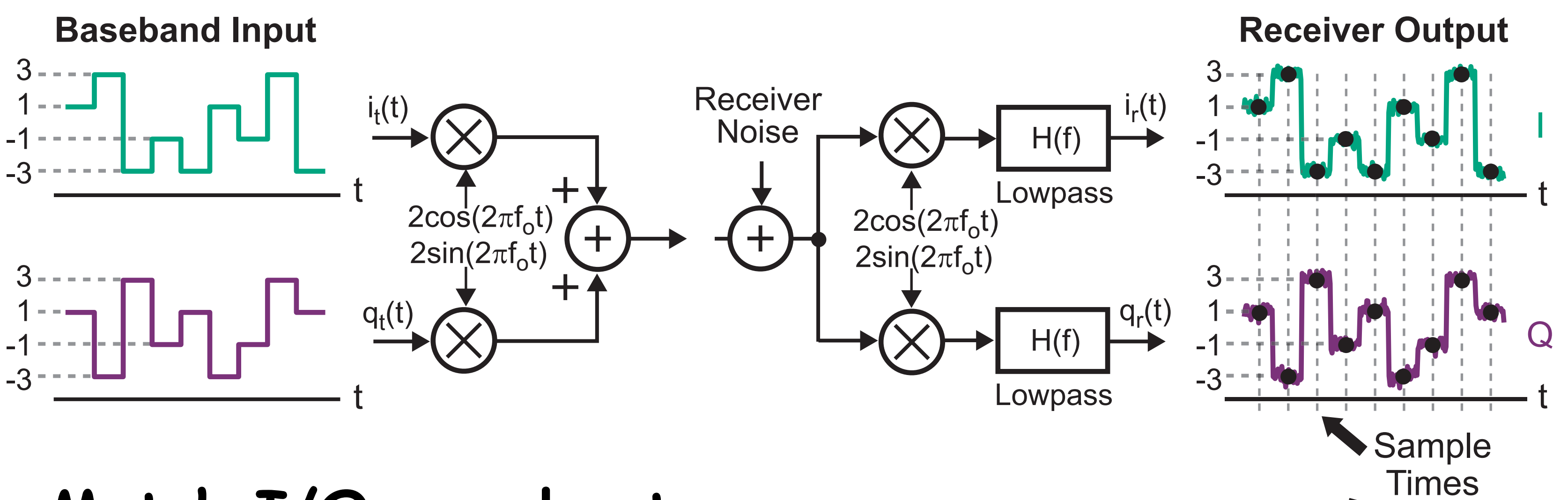
The Impact of Noise



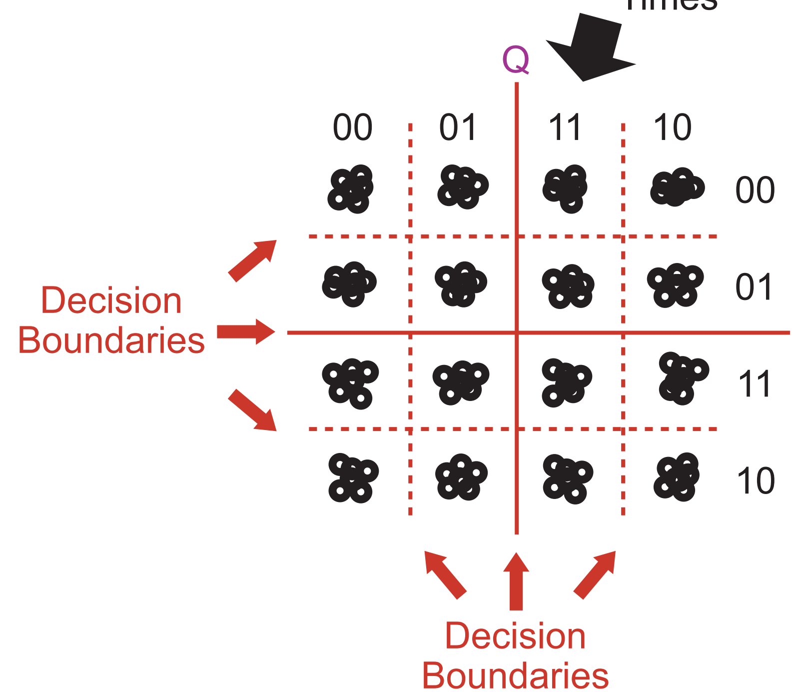
- Noise perturbs sampled I/Q values
  - Constellation points no longer consist of single dots for each symbol
- Issue: what is the best way to match received I/Q samples with their corresponding symbols?



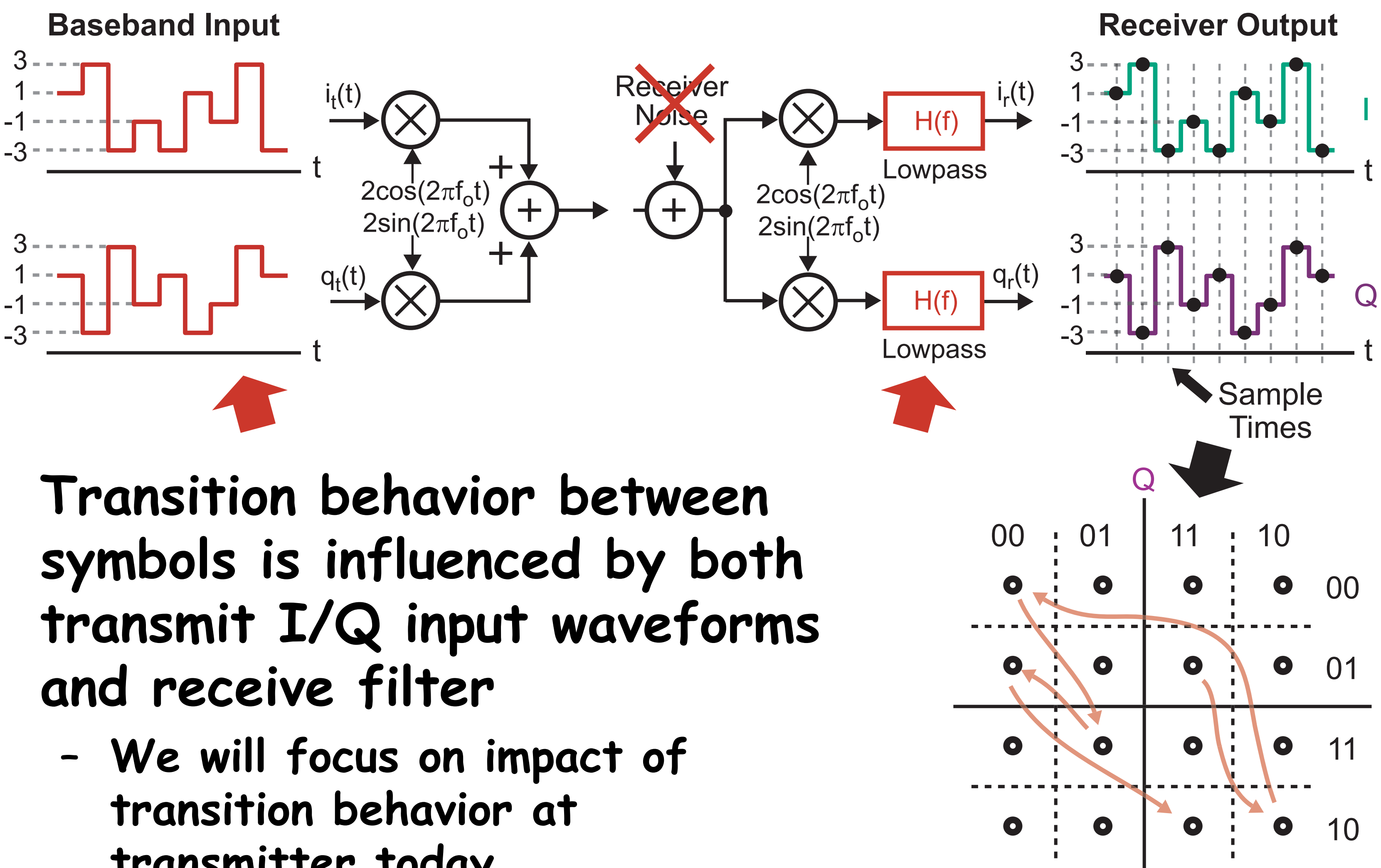
Symbol Selection Based on Slicing



- **Match I/Q samples to their corresponding symbols based on decision regions**
  - Choose decision regions to minimize symbol errors
  - Decision boundaries are also called slicing levels

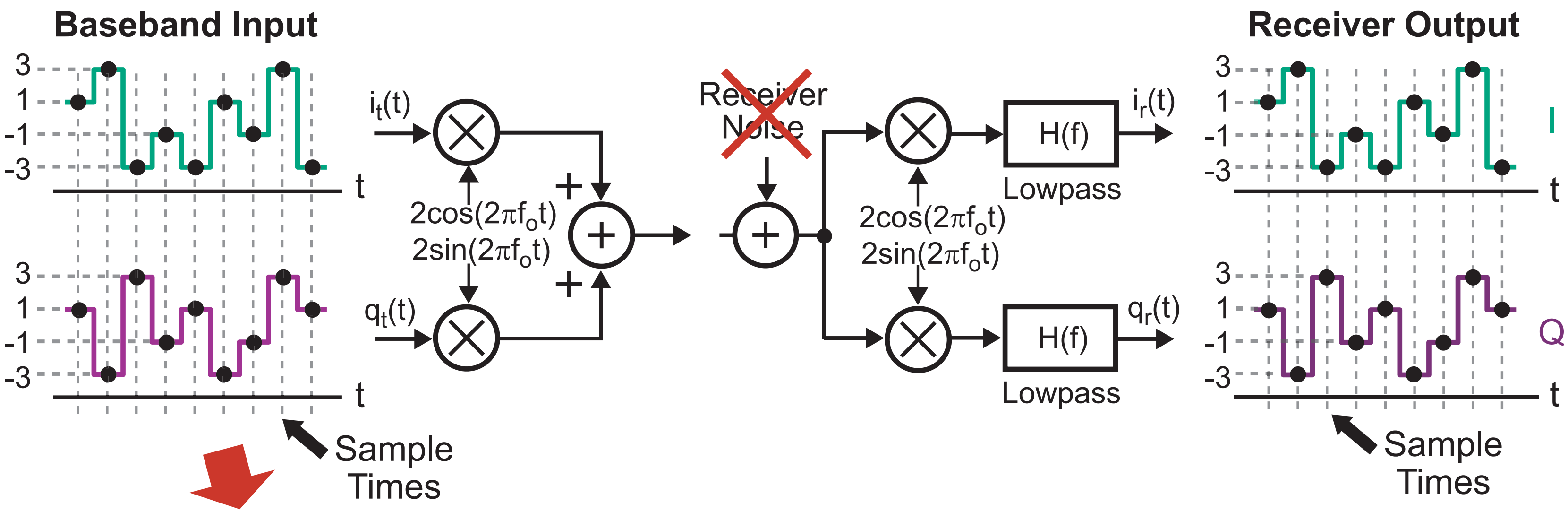


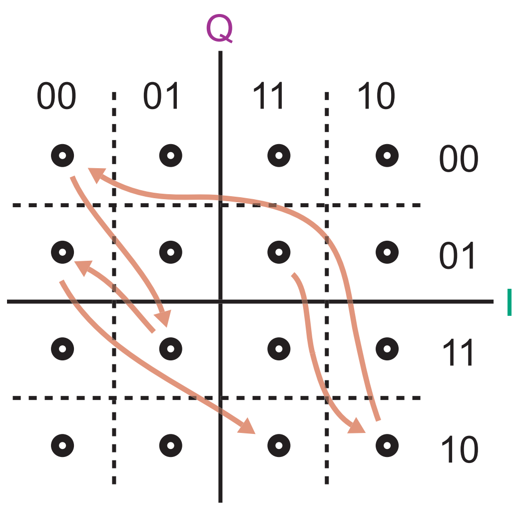
Transitioning Between Symbols



- **Transition behavior between symbols is influenced by both transmit I/Q input waveforms and receive filter**
  - We will focus on impact of transition behavior at transmitter today
  - Ignore the impact of noise for this analysis

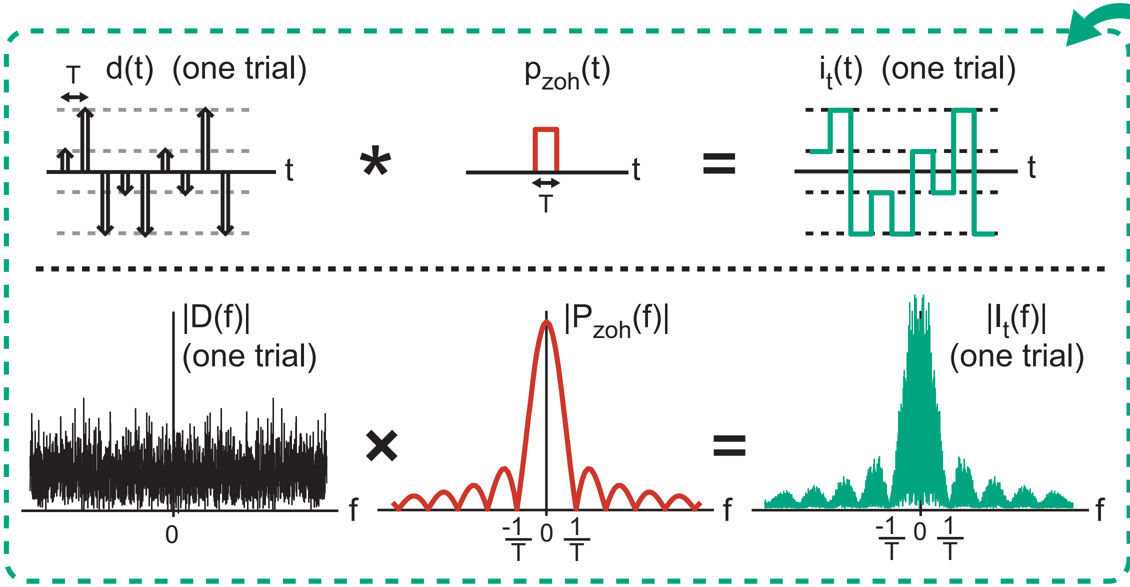
Influence of Transitions at Transmitter

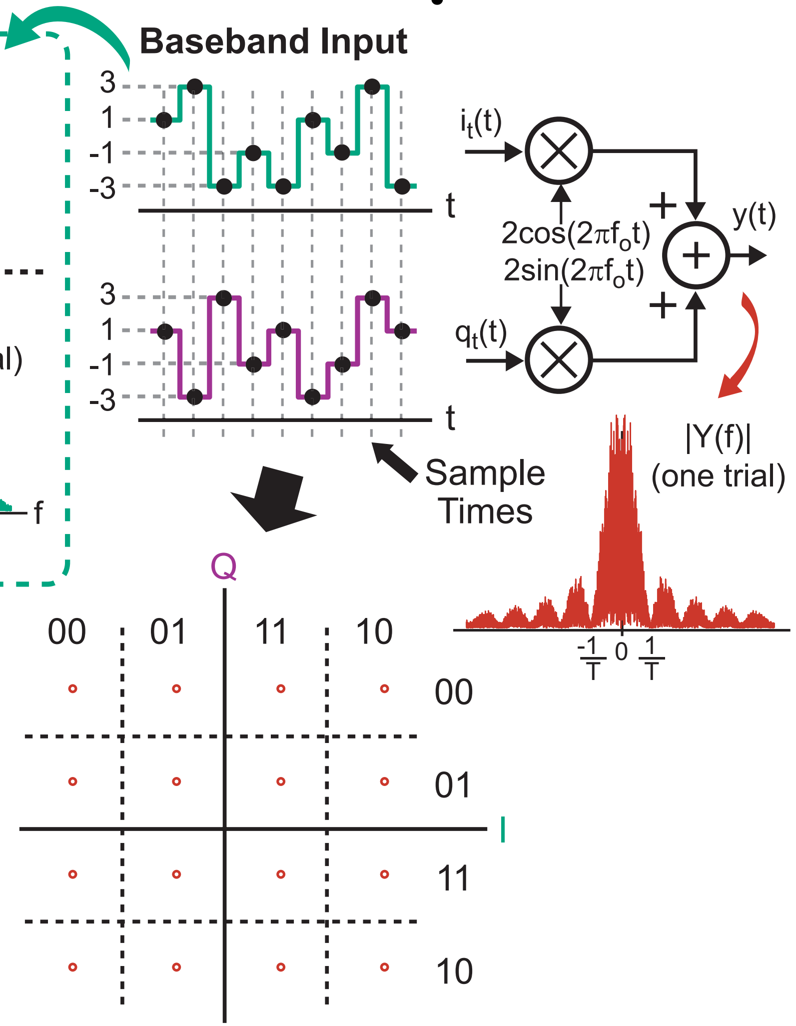




- *Ideal* analog communication channel simply transports the transmitter I/Q signals to the receiver
- Constellation diagram can be constructed at *transmitter* to evaluate its performance
  - Bad constellation at transmitter implies bad one at receiver

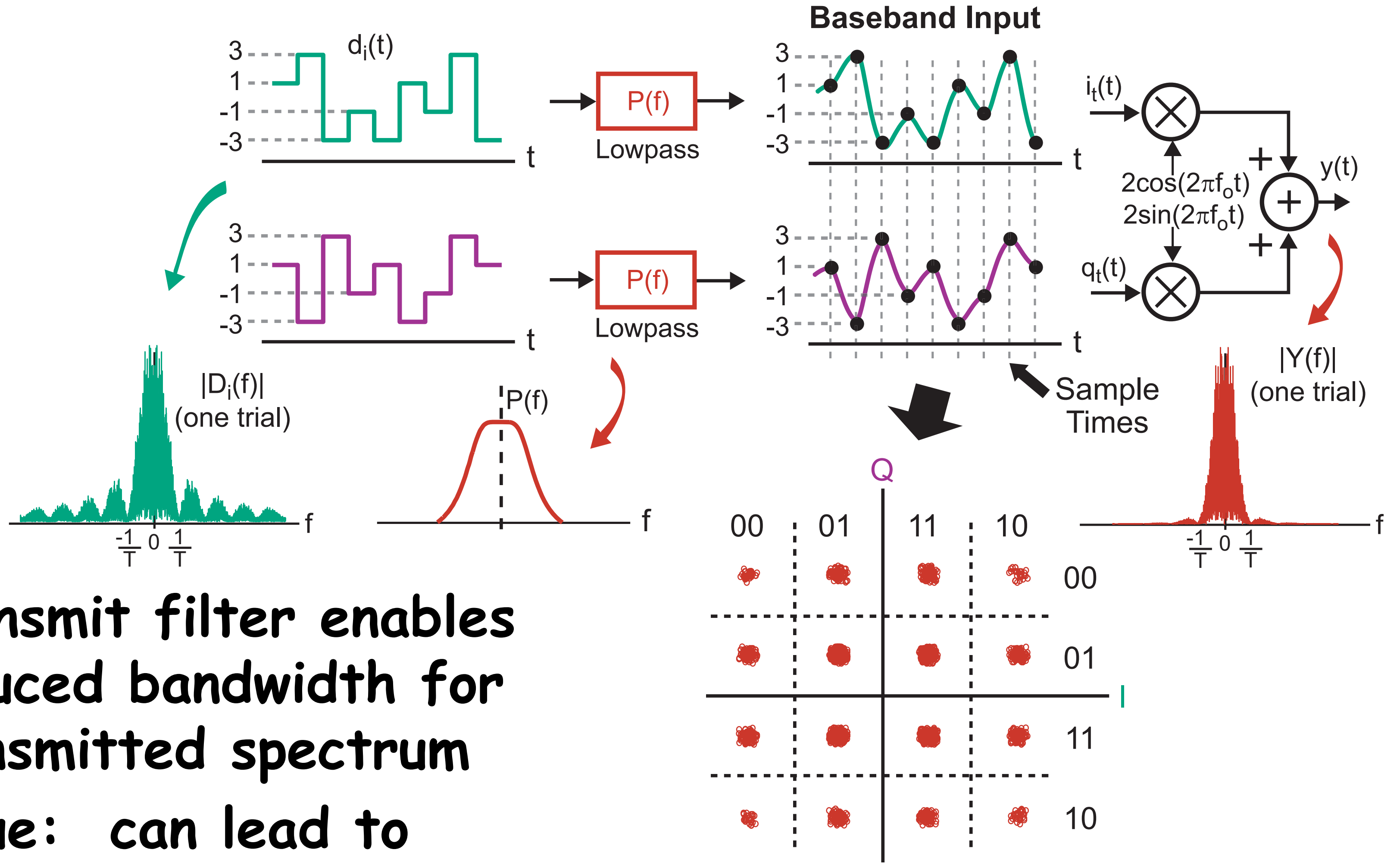
Transitions and the Transmitted Spectrum





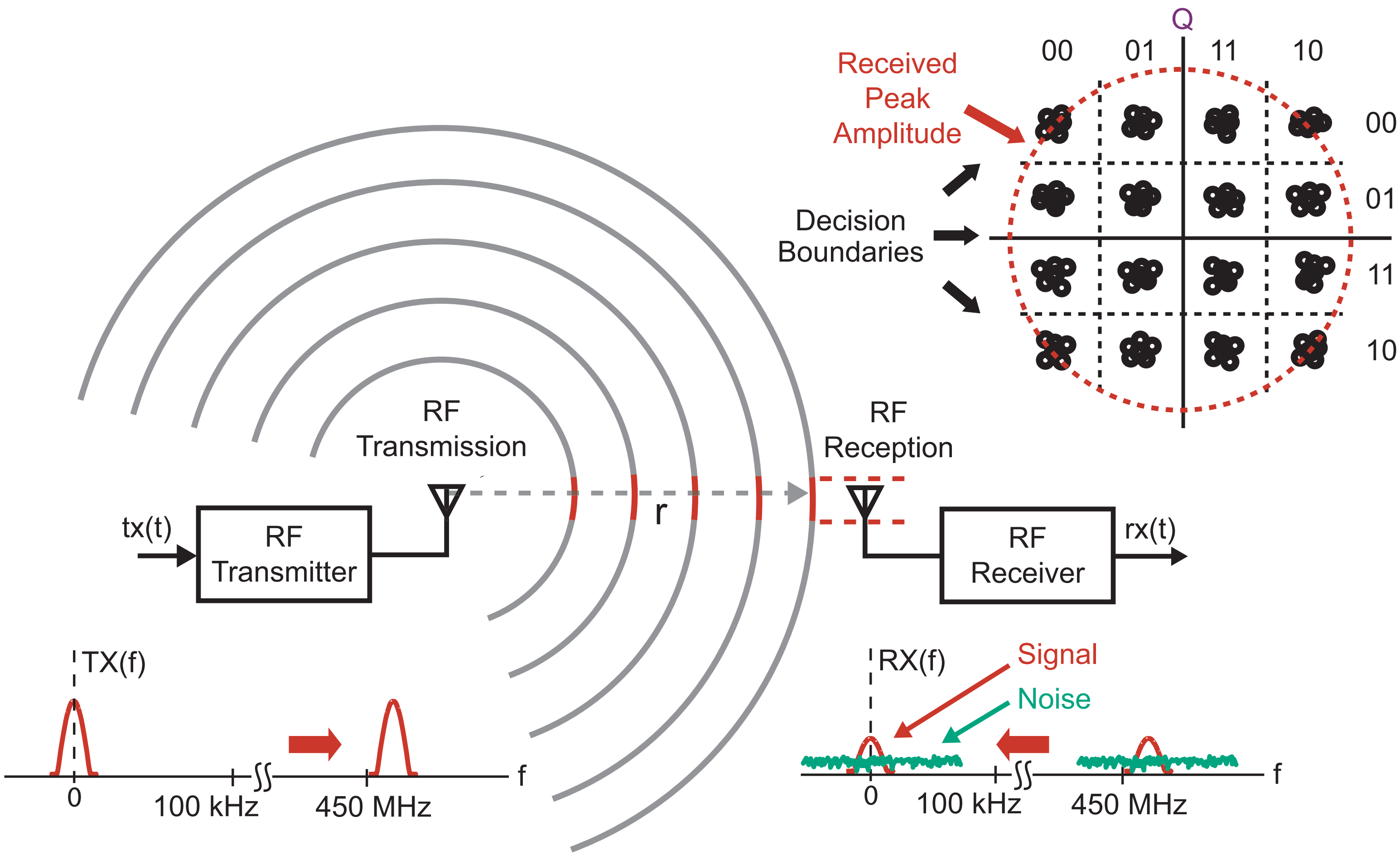
- **Want transmitted spectrum with minimal bandwidth**
  - **Wireless communication channels are a *shared* resource**
- **Issue: sharply changing I/Q waveforms lead to a *wide bandwidth* spectrum**

Impact of Transmit Filter



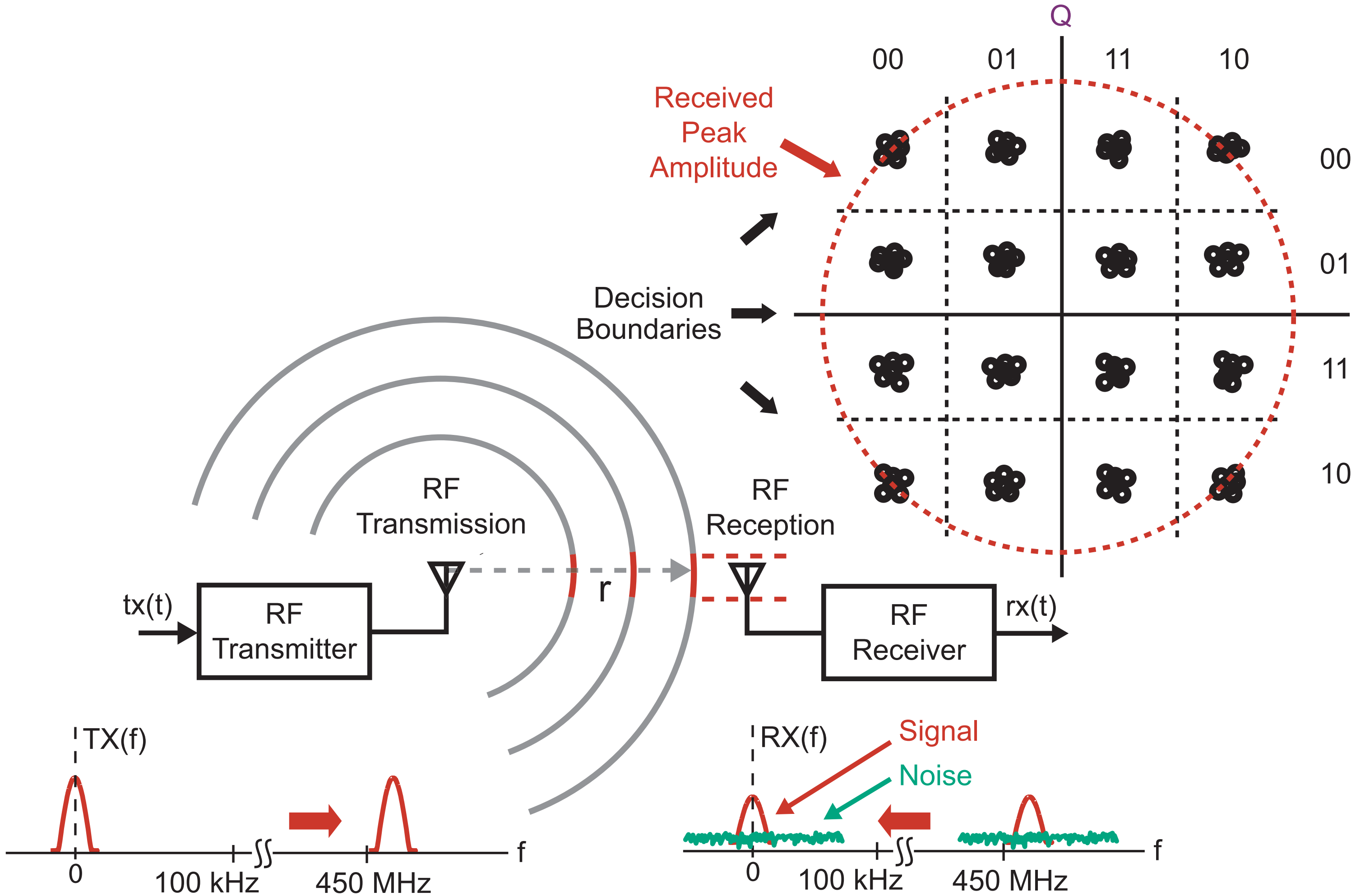
- **Transmit filter enables reduced bandwidth for transmitted spectrum**
- **Issue: can lead to *intersymbol interference (ISI)***
  - Constellation diagram displays vulnerability to making bit errors

Impact of SNR on Receiver Constellation



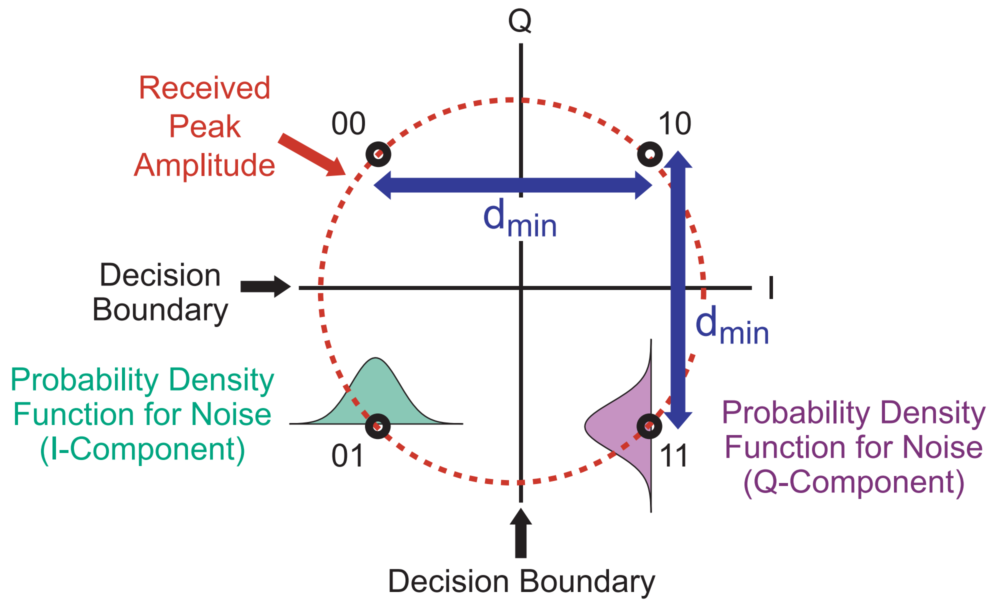
- SNR influenced by transmitted power, distance between transmitter and receiver, and noise

Impact of Increased Signal on Constellation



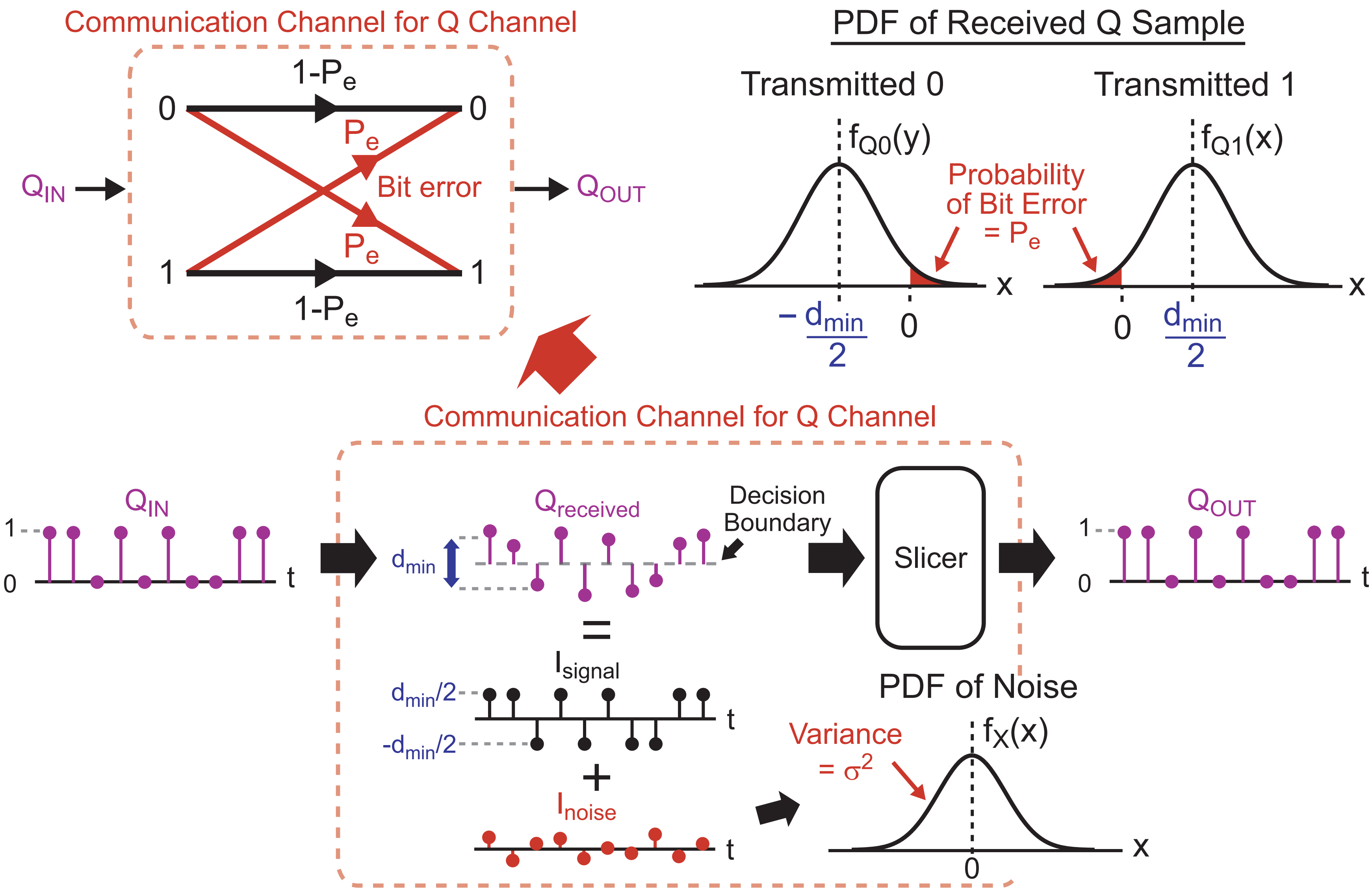
- **Increase in received signal power leads to increased separation between symbols**
  - **SNR is improved if noise level unchanged**

Quantifying the Impact of Noise



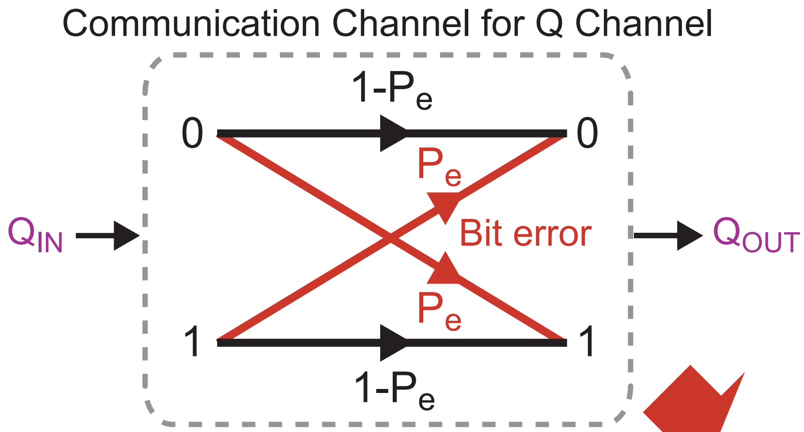
- Minimum separation between symbols: d_{min}
- PDF of noise: zero mean Gaussian PDF
  - Variance of noise sets the spread of the PDF
- Bit errors: occur when noise moves a symbol by a distance more than dmin/2

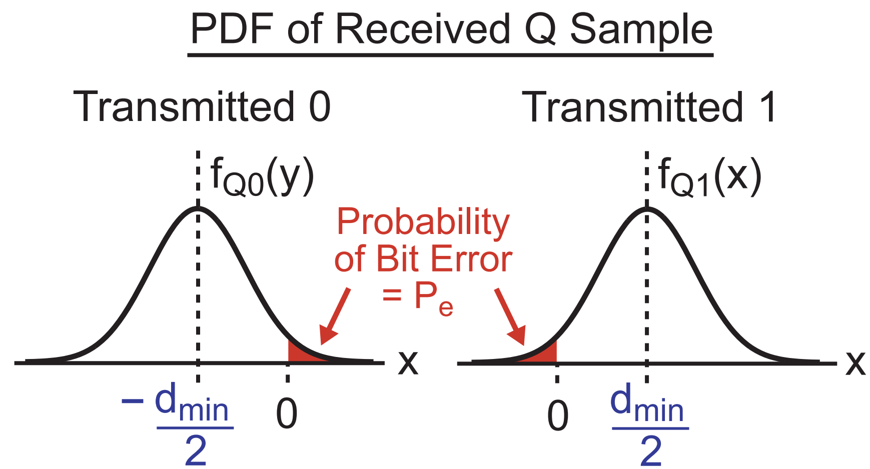
The Binary Symmetric Channel Model

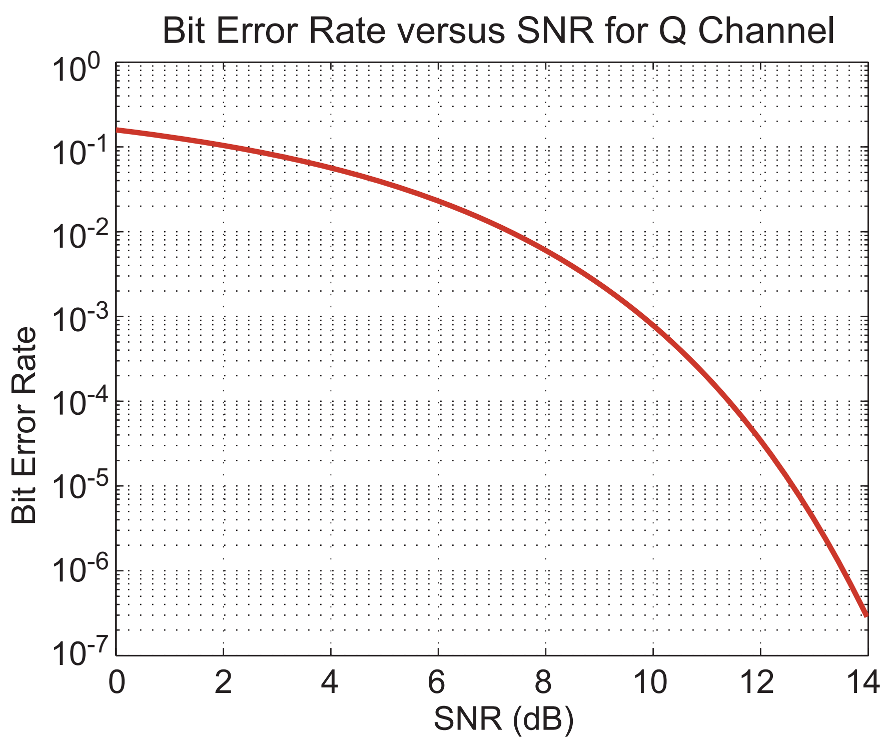


- **Provides a binary signaling model of channel**

Resulting Bit Error Rate Versus SNR







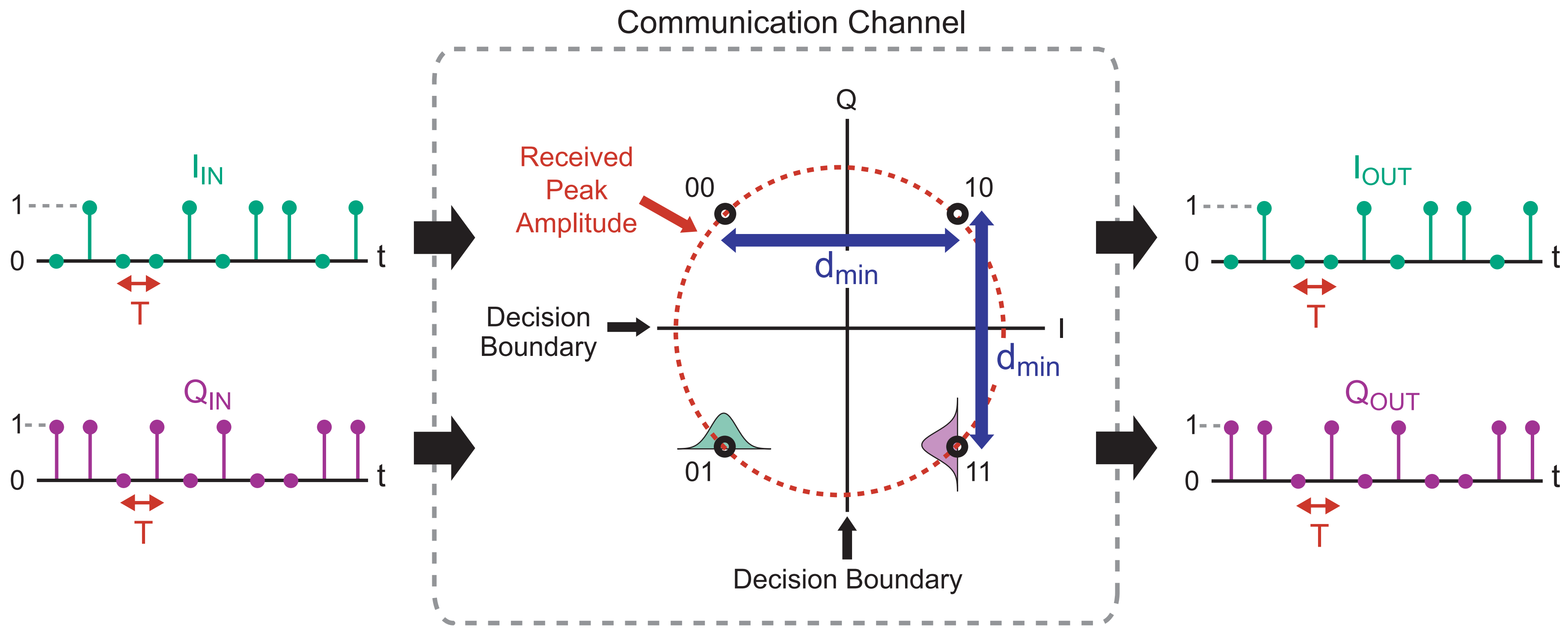
Note:

- **Bit Error Rate = P_e**
- **SNR (dB) =**

$$10\log\left(\frac{(d_{min}/2)^2}{\sigma^2}\right)$$

- **Gaussian PDF for noise**

Shannon Capacity



- **In 1948, Claude Shannon proved that**
  - **Digital communication can achieve arbitrary low bit-error-rates if appropriate *coding* methods are employed**
  - **The capacity of a *Gaussian channel* with bandwidth *BW* to support arbitrary low bit-error-rate communication is:**

$$\boxed{C = BW\log_2(1+SNR)\ \text{bits/second}}$$

Summary

- The Fourier Transform provides a powerful tool for analysis of sampling, modulation, and filtering
- The digital abstraction provides a practical implementation framework for complicated systems
  - Analog signaling is highly susceptible to noise
  - Digital signaling provides noise margin
- We can represent a digital communication channel with a binary signaling model
  - Bit errors are quantified in terms of the signal-to-noise ratio of the overall channel
- Claude Shannon introduced the concept of using coding methods to achieve arbitrarily low bit error rates across practical communication channels