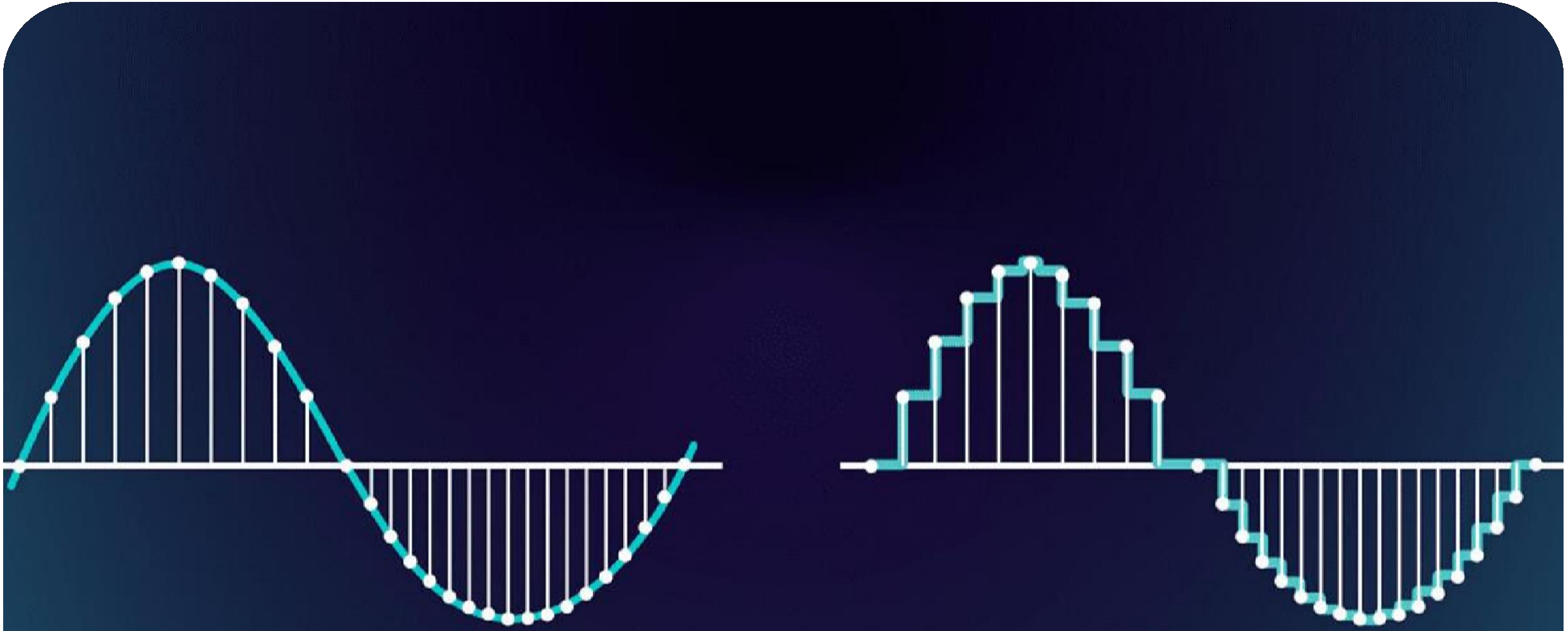
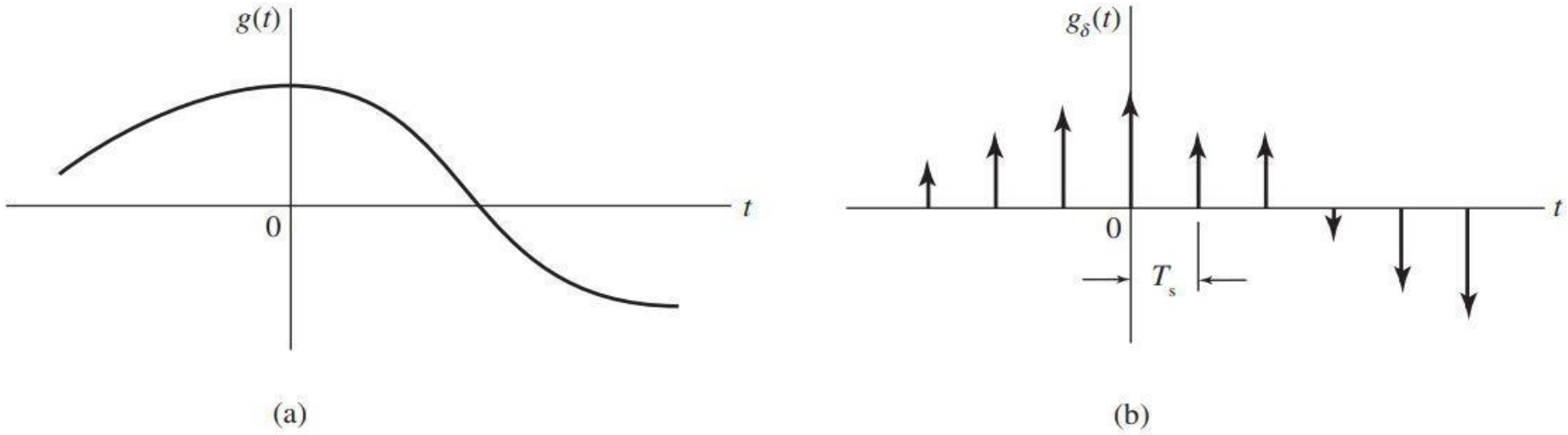


Sampling and Reconstruction

Exp-1

Sampling Process

- **Sampling** is defined as, "The process of measuring the instantaneous values of continuous-time signal in a discrete form."
- **Sample** is a piece of data taken from the whole data which is continuous in the time domain.



The sampling process: (a) Analog signal, (b) Instantaneously sampled version of the analog signal

Sampling Process

Let $g(t)$ be an arbitrary signal of finite energy as shown in figure above, sampled at uniform rate 'T_s' seconds denoted by $g(nT_s)$ where 'n' is an integer

Sampling Rate

- To discretize the signals, the gap between the samples should be fixed.
- That gap can be termed as a **sampling period T_s**.

$$Sampling\ Frequency = \frac{1}{T_s} = f_s$$

Sampling frequency

- **Sampling frequency f_s** is the reciprocal of the sampling period.
- This sampling frequency, can be simply called as **Sampling rate**.
- The sampling rate denotes the number of samples taken per second, or for a finite set of values.

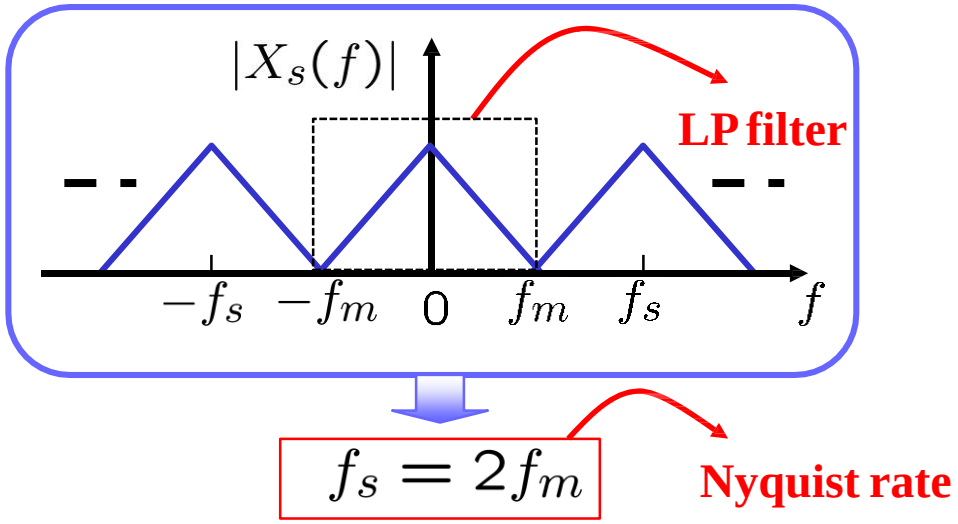
Nyquist Rate

- Suppose that a signal is band-limited with no frequency components higher than **W** Hertz.
- That means, **W** is the highest frequency.
- For such a signal, for effective reproduction of the original signal, the sampling rate should be twice the highest frequency.

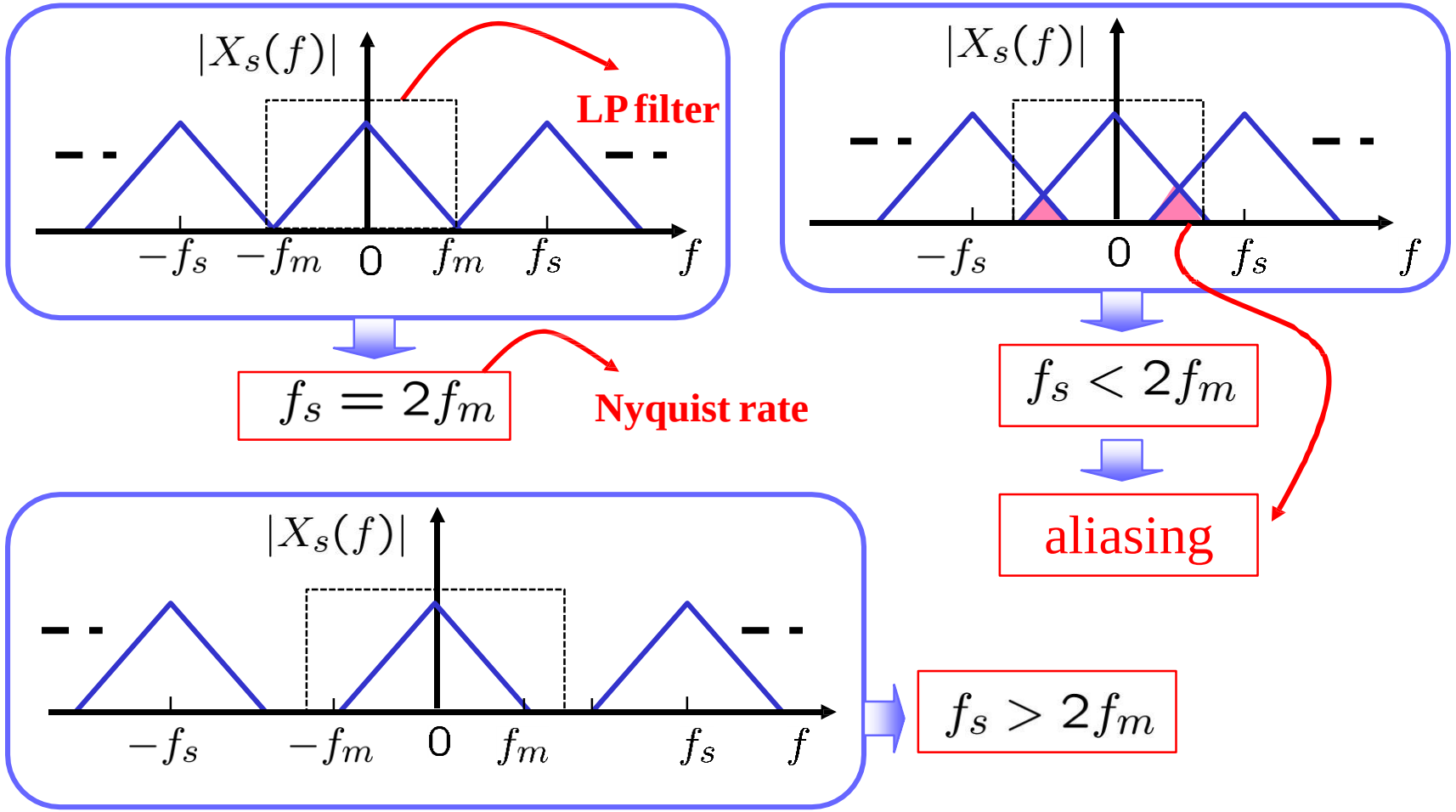
$$f_s = 2W$$

- This rate of sampling is called as **Nyquist rate**
- **Nyquist interval**

$$T_s = \frac{1}{2W}$$



Sampling theorem and Aliasing



Matlab code

```
clc;
clear all
close all;
t=0:0.01:0.2;
fm=input('Enter the frequency of the input signal fm=');
fu=input('Enter the sample rate of the signal (under sampling) fu=');
fc=input('Enter the sample rate of the signal (critical sampling) fc=');
fo=input('Enter the sample rate of the signal (over sampling) fo=');


%Generation of input signal
x=sin(2*pi*fm*t);
subplot(3,3,2);
plot(t,x);
grid on;
xlabel('Time period (s)');
ylabel('Amplitude (v)');
title('Input analog signal');
```

Matlab code

```
%Under sampling
tu=0:1/fu:0.2;
xu=sin(2*pi*fm*tu);
subplot(3,3,4);
stem(tu,xu);
grid on;
xlabel('Time period (s)');
ylabel('Amplitude (v)');
title('Under sampling');


%Critical sampling
tc=0:1/fc:0.2;
xc=sin(2*pi*fm*tc);
subplot(3,3,5);
stem(tc,xc);
grid on;
xlabel('Time period (s)');
ylabel('Amplitude (v)');
title('Critical sampling');
```

Matlab code

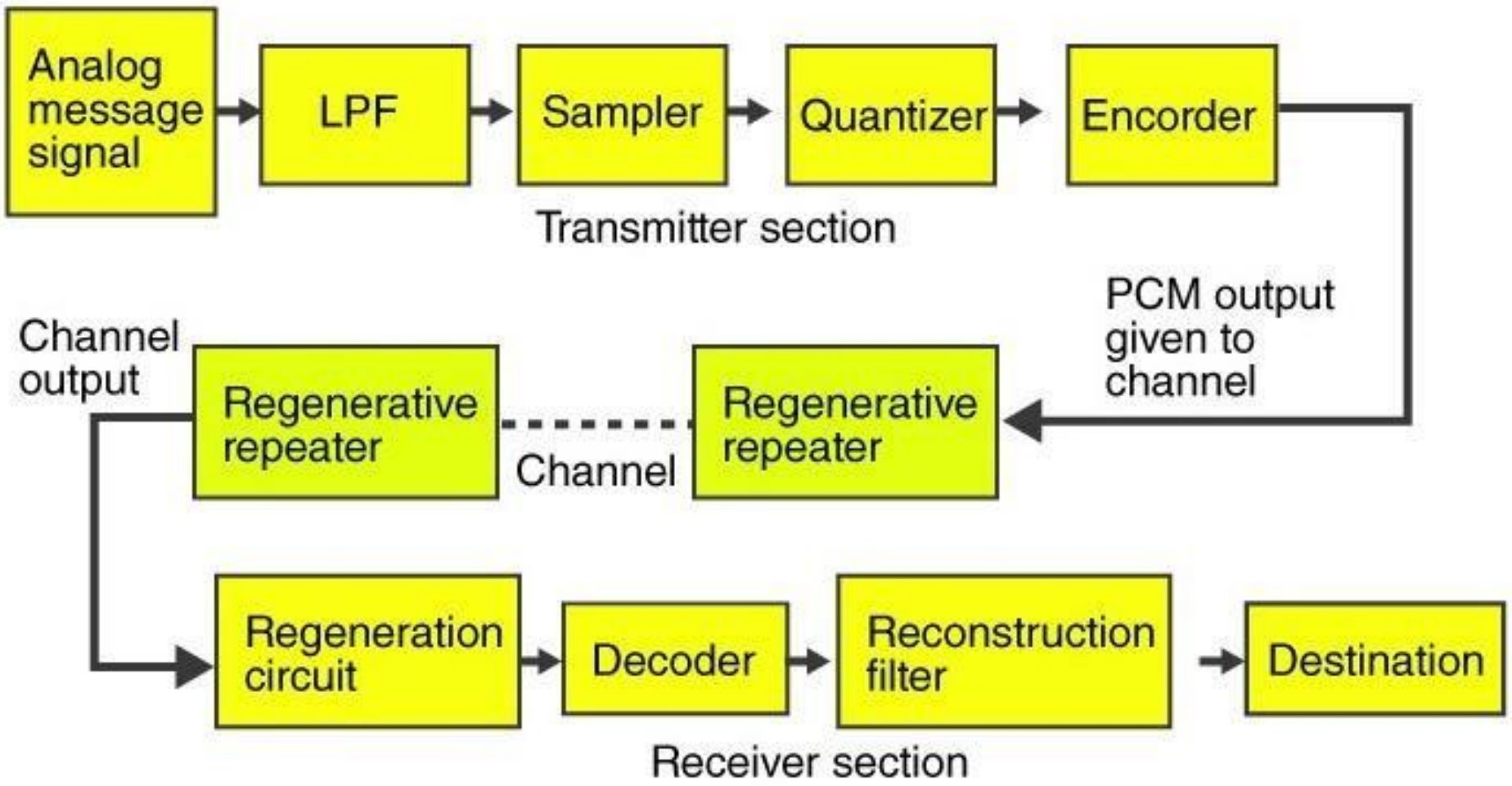
```
%Over sampling
to=0:1/fo:0.2;
xo=sin(2*pi*fm*to);
subplot(3,3,6);
stem(to,xo);
grid on;
xlabel('Time period (s)');
ylabel('Amplitude (v)');
title('Over sampling');

%Reconstruction
tre=0:1/fc:0.2;
xre=interp1(tc,xc,tre);
subplot(3,3,8);
plot(tre,xre);
grid on;
xlabel('Time period (s)');
ylabel('Amplitude (v)');
title('Reconstructed input signal');
```

Experiment:3

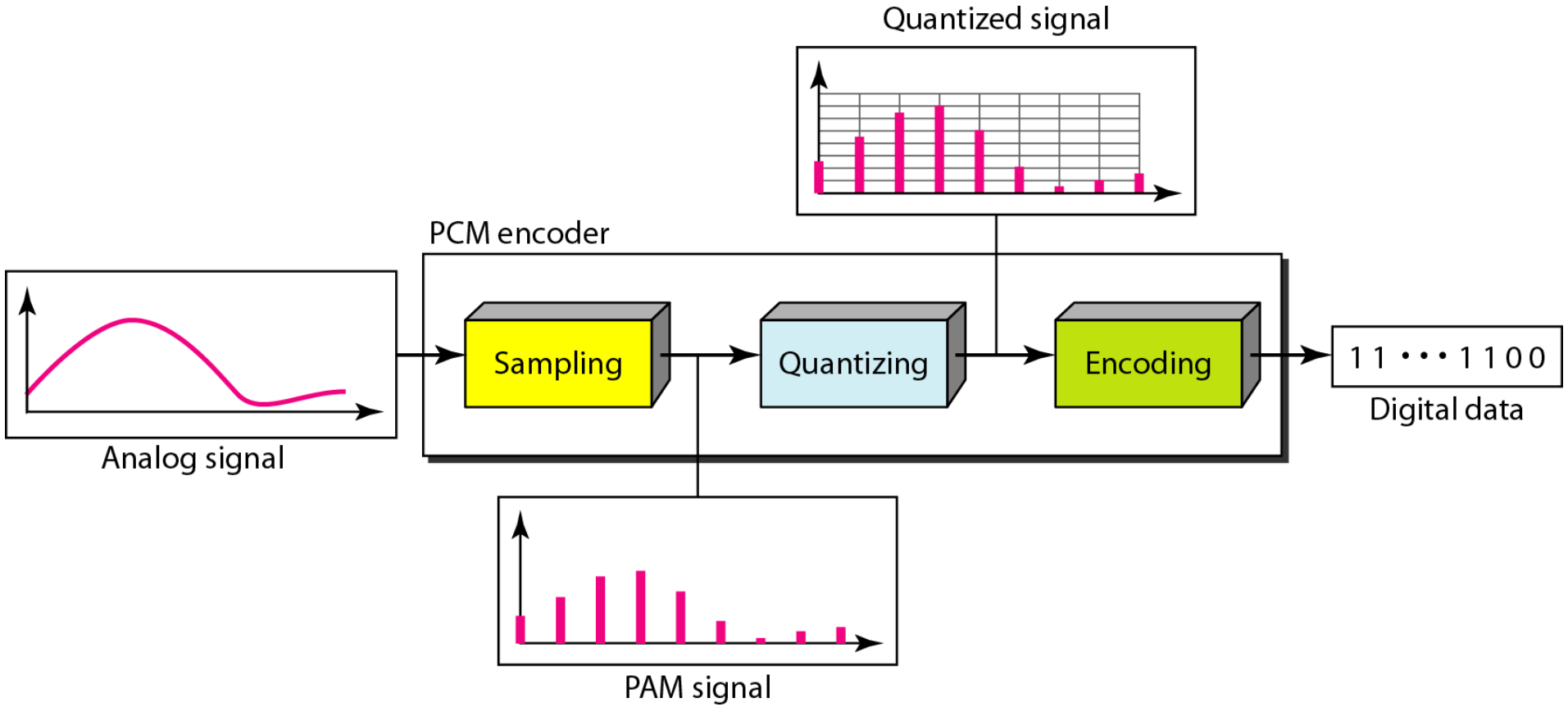
PCM Encoder and Decoder

Dr. T.Ilavarasan, Associate Professor,SENSE, VIT

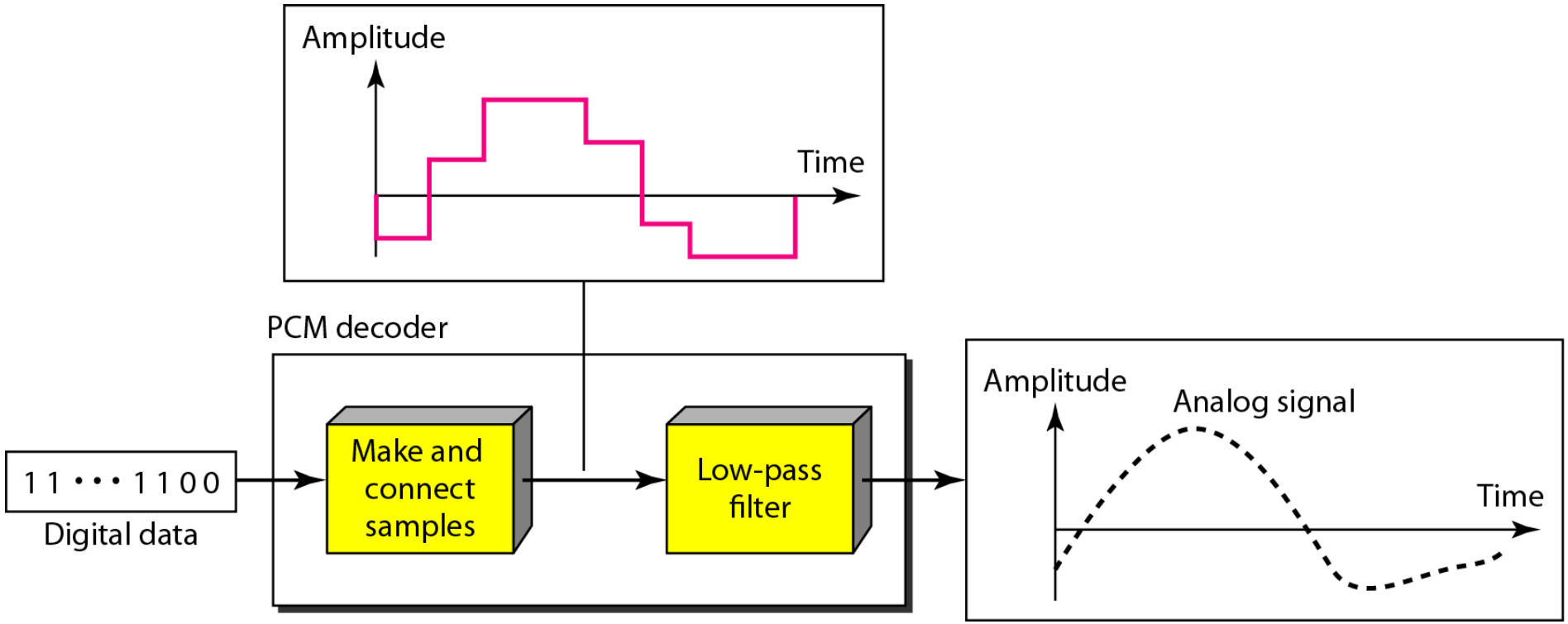
PCM building blocks



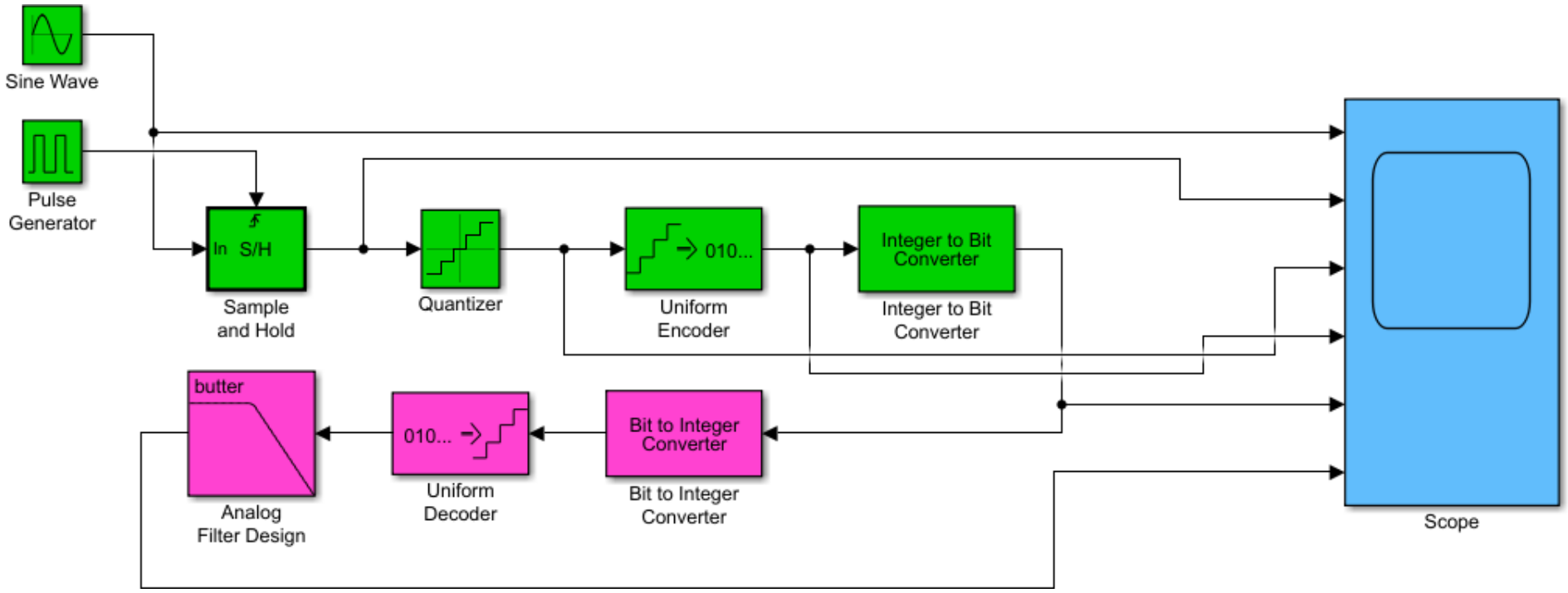
PCM Encoder


Quantized signal
PCM encoder
Sampling
Quantizing
Encoding
1 1 • • • 1 1 0 0
Digital data
Analog signal
PAM signal

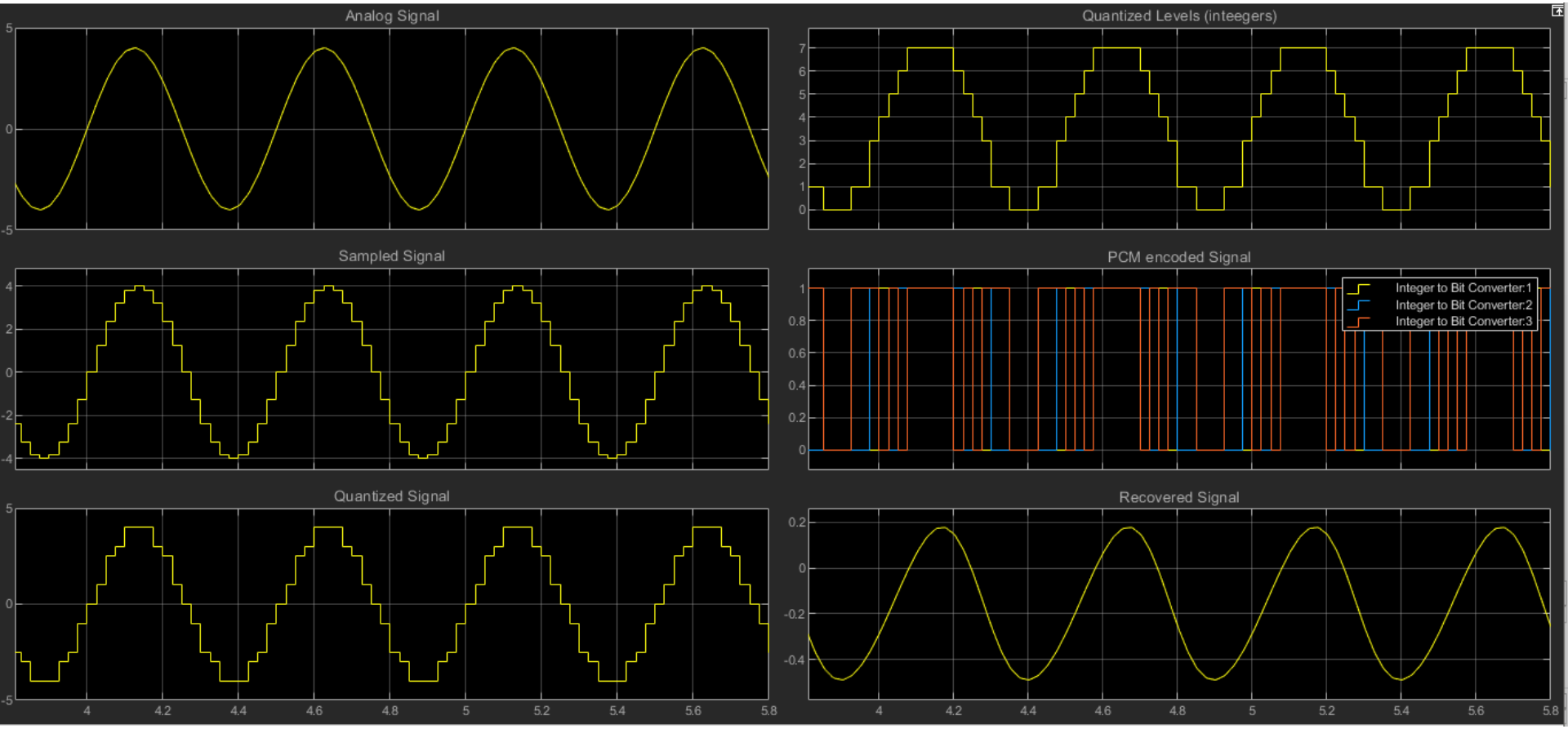
PCM Decoder


Amplitude
Time
PCM decoder
11 • • • 1100
Digital data
Make and connect samples
Low-pass filter
Amplitude
Analog signal
Time

Simulink Block Diagram



Results from Simulink



Experimentation

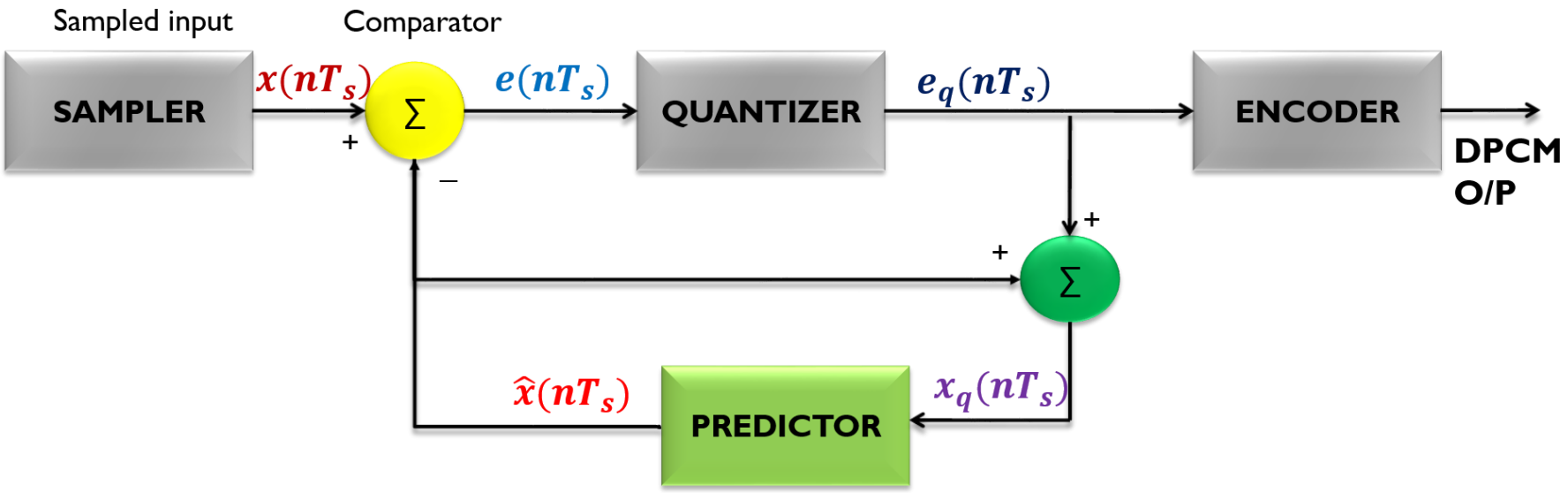
- With initial frequency setup for Analog Signal and Pulse Stream generator as discussed perform following experiments

1. Generate the outputs (all 6 waveforms) for Quantizer Levels of (4,8,16) and amplitude of signals V_p=10V.
2. Generate the outputs (all 6 waveforms) for Various Quantization Intervals of (0.1, 5, 20) with L=8.
3. Generate the outputs (all 6 waveforms) for Various Amplitudes of (5V, 15V, 20V) with L=16.

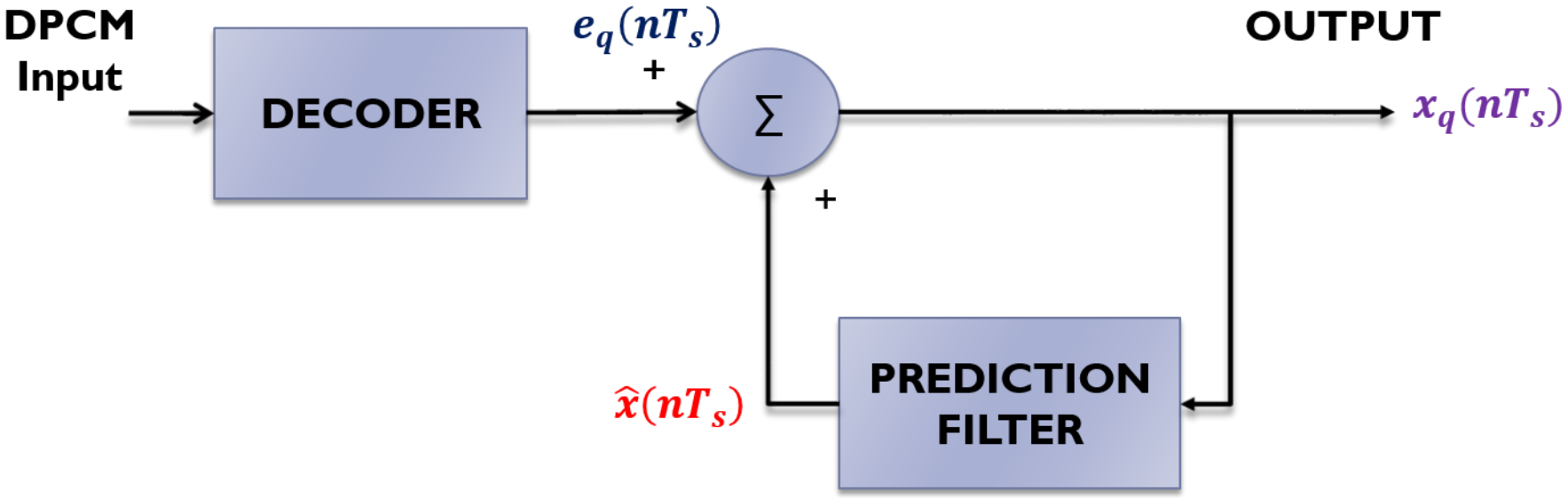
Experiment:5

DPCM Encoder and Decoder

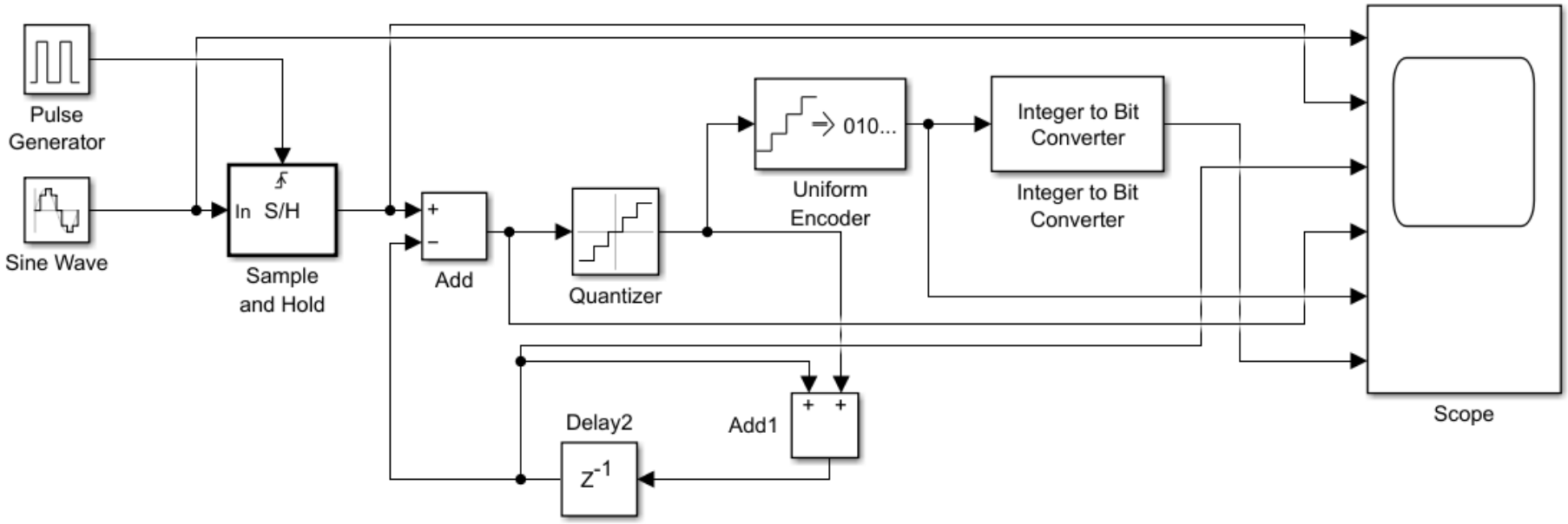
DPCM Transmitter (Encoding)



DPCM Receiver (Decoding)



Simulink Block (DPCM Encoding)



Experimental Setups

- Sine Wave:
  - Amplitude: 5 V
  - Freq: 2*pi*2
- Pulse Generator:
  - Amplitude: 5 V
  - Period: 1/40
- Qunatizer Interval: 0.5
- Number of Levels: 8

Experiment

1. Implement DPCM encoding and compare it with PCM (using same Setup in previous slide) and show how many bits are reduced per sample in DPCM.

2. Implement DPCM encoding (using same Setup in previous slide) but change the sample period in Sine wave block from zero(0) to (1/1000; 1/10000; 1/100000) and write the inference from them.

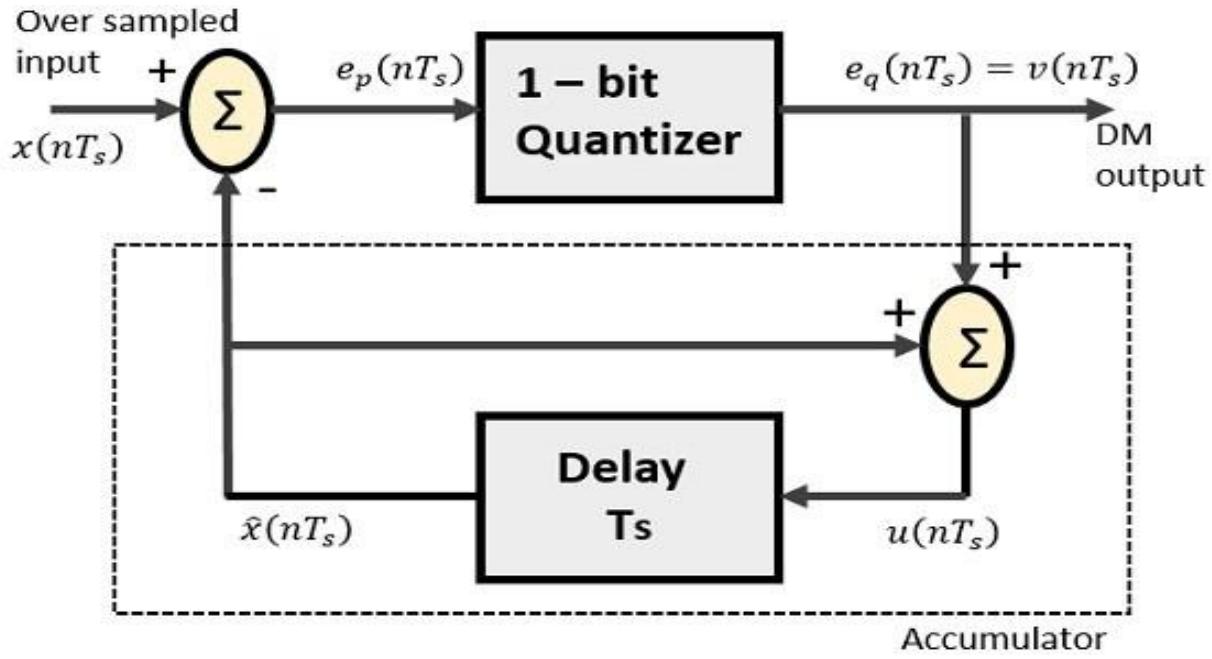
3. Implement DPCM decoder

Experiment:7

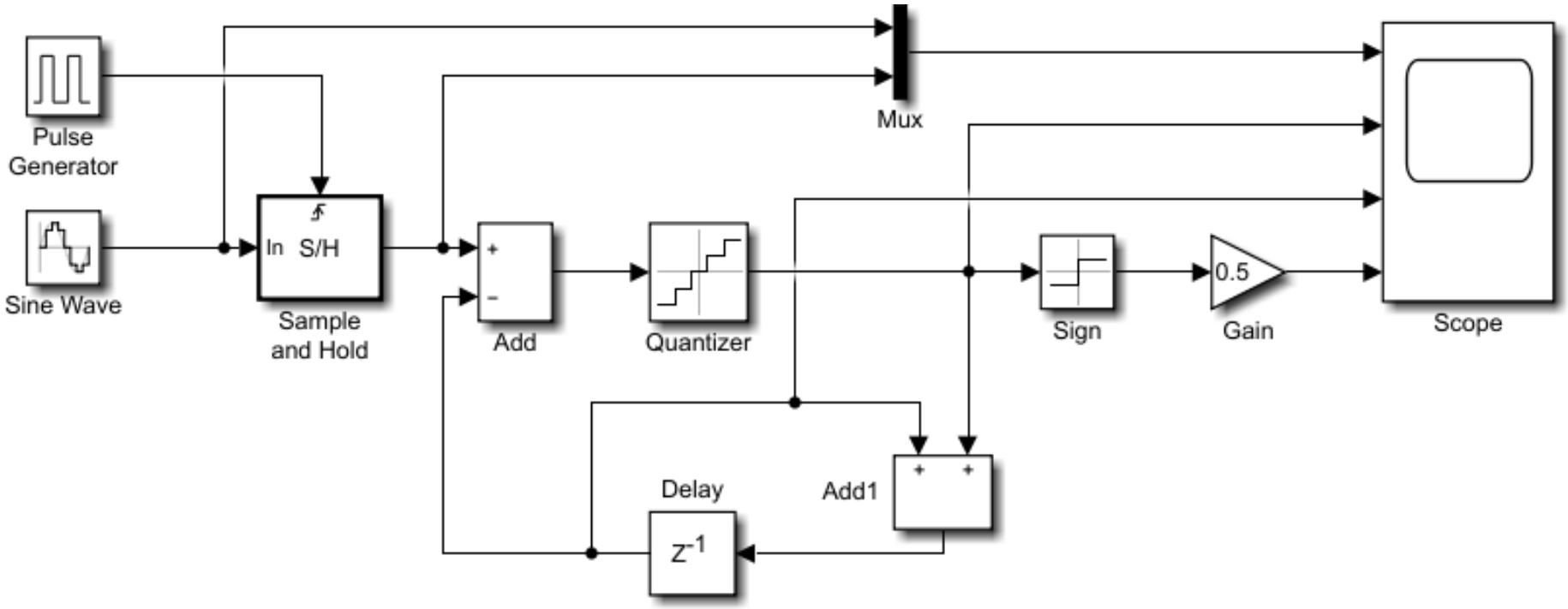
Delta Modulation Encoder and Decoder

Delta Modulation (DM)

Transmitter (encoder/modulator)

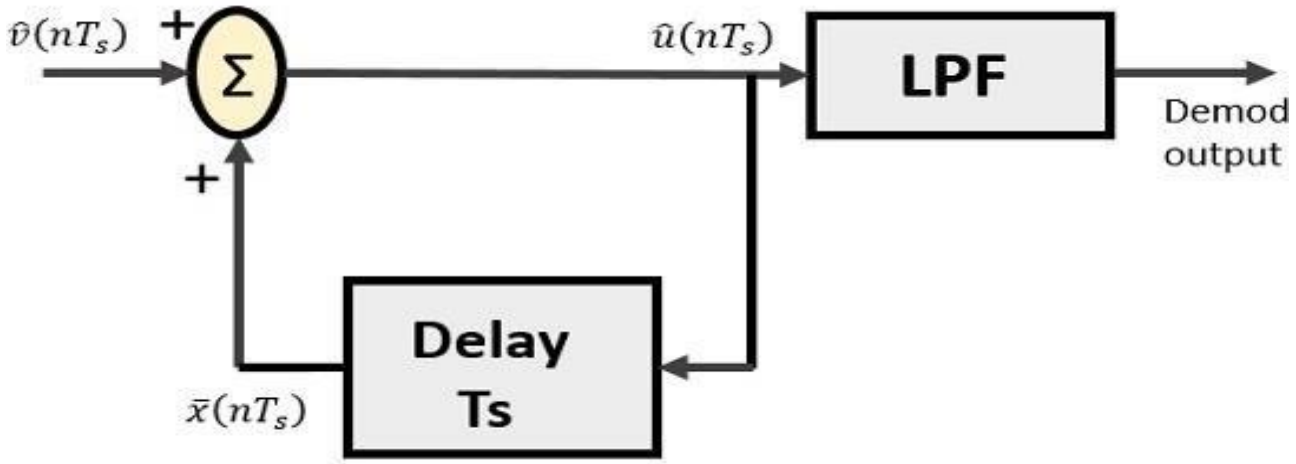


DM Simulink Block



Delta Modulation (DM)

Receiver (decoder/demodulator)



Experimental Setups

- Sine Wave:
  - Amplitude: 1 V
  - Freq: 2*pi*2
  - Sample time: 1/10000
- Pulse Generator:
  - Amplitude: 1 V
  - Period: 1/40
- Qunatization Interval: 0.2

Experiment

1. Implement DM encoding and compare it with PCM, DPCM (using same Setup in previous slide) and portray the significance of every method.

2. Implement DM encoding (using same Setup in previous slide) but change the sample period in Sine wave block to zero and write the inference from them.

3. Implement DM decoder

Exp 9
Line Coding Techniques
(Software)

```
%%%%% Unipolar NRZ%%%%

clc
clear all


bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration
n=200;                              %bit duration
dt=1/n;


TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                        % power spectra frequency


t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

%line coding :Uni-polar
for(i=0:length(bits)-1)
    if bits(i+1)==1
        x((i*n)+1:(i+1)*n)=1;
    else
        x((i*n)+1:(i+1)*n)=0;
    end
end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('NRZ UNI POLAR-LINE CODING')
axis([0 2500 0 2])

sx=((a^2)/4)*TB*(sinc(x1).^2)+((a^2)/4)*dirac(f);      %power spectra UNIPOLAR
formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title('NRZ UNI POLAR-POWER SPECTRUM-LINE CODING')
```

```
%%%%% Unipolar RZ%%%%

clc
clear all


bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration
n=200;                              %bit duration
dt=1/n;


TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                         % power spectra frequency


t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

%line coding :Uni-polar
for(i=0:length(bits)-1)
    if bits(i+1)==1
        x((i*n)+1:(i+0.5)*n)=1;
        x(((i+0.5)*n)+1:(i+1)*n)=0;

    else
         x((i*n)+1:(i+0.5)*n)=0;
        x(((i+0.5)*n)+1:(i+1)*n)=0;
    end
end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('RZ UNI POLAR-LINE CODING')
axis([0 2500 0 2])

sx=((a^2)/4)*TB*(sinc(x1).^2)+((a^2)/4)*dirac(f);      %power spectra UNIPOLAR
formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title('RZ UNI POLAR-POWER SPECTRUM-LINE CODING')
```

```
%%****************** POLAR NRZ******************%%
clc
clear all


bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration
n=200;                              %bit duration
dt=1/n;


TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                        % power spectra frequency


t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

%line coding -polar
for(i=0:length(bits)-1)
    if bits(i+1)==1
        x((i*n)+1:(i+1)*n)=1;
    else
        x((i*n)+1:(i+1)*n)=-1;
    end
end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('NRZ POLAR-LINE CODING')
axis([0 2500 -1 1])

sx=(a^2)*TB*(sinc(x1).^2);      %power spectra POLAR formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title('NRZ POLAR-POWER SPECTRUM-LINE CODING')
```

```
%%****************** POLAR RZ*******************%%
clc
clear all


bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration
n=200;                              %bit duration
dt=1/n;


TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                        % power spectra frequency


t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

%line coding -polar
for(i=0:length(bits)-1)
    if bits(i+1)==1
        x((i*n)+1:(i+0.5)*n)=1;
        x(((i+0.5)*n)+1:(i+1)*n)=0;

    else
         x((i*n)+1:(i+0.5)*n)=-1;
        x(((i+0.5)*n)+1:(i+1)*n)=0;
    end
end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('RZ POLAR-LINE CODING')
axis([0 2500 -1 1])

sx=(a^2)*TB*(sinc(x1).^2);      %power spectra POLAR formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title('RZ POLAR-POWER SPECTRUM-LINE CODING')
```

```
%%***************BI-POLAR - AMI**************
clc
clear all

bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration – Full time of
bit sequence
n=200;                              %bit duration
dt=1/n;

TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                         % power spectra frequency

t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

count=0;

%line coding :BI-polar
for(i=0:length(bits)-1)
    if bits(i+1)==1
        if mod(count,2)==0
            x((i*n)+1:(i+1)*n)=1;
        else
            x((i*n)+1:(i+1)*n)=-1;
        end
        count=count+1;
    else
        x((i*n)+1:(i+1)*n)=0;


    end



end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('BI-POLAR-LINE CODING - AMI')
axis([0 2500 -2 2])


 sx=(a^2)*TB*(sinc(x1)).*(sinc(x1)).*(sin(pi*x1)).*(sin(pi*x1));     %power
spectra BIPOLAR formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title(' BI-POLAR-POWER SPECTRUM-LINE CODING-AMI')
```

```
%%***************BI-POLAR - PSEUDOTERNARY**************
clc
clear all

bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration – Full time of
bit sequence
n=200;                              %bit duration
dt=1/n;

TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                         % power spectra frequency


t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

count=0;

%line coding :BI-polar
for(i=0:length(bits)-1)
    if bits(i+1)==0
        if mod(count,2)==0
            x((i*n)+1:(i+1)*n)=1;
        else
            x((i*n)+1:(i+1)*n)=-1;
        end
        count=count+1;
    else
        x((i*n)+1:(i+1)*n)=0;


    end


end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('BI-POLAR-LINE CODING-PSUEDOTERNARY')
axis([0 2500 -2 2])


 sx=(a^2)*TB*(sinc(x1)).*(sinc(x1)).*(sin(pi*x1)).*(sin(pi*x1));     %power
spectra BIPOLAR formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title(' BI-POLAR-POWER SPECTRUM-LINE CODING-PSUEDOTERNARY')
```

```
%%************MANCHESTER*****************
clc
clear all


bitrate=1;              %Rb-bitrate
bits=[1 0 0 1 0 1 1 1 0 1 0];
T= length(bits)/bitrate;                        %Symbol duration
n=200;                              %bit duration
dt=1/n;


TB=1/bitrate;                        %1/bitrate value
a=1;
f=0:bitrate/100:2*bitrate;

x1=f*TB;                        % power spectra frequency


t=0:dt:T;                          %t takes the valu 0 to T with innerspace equal
to dt
x=zeros(1,length(t));           %initialisation of x

%line coding :MANCHESTER
for(i=0:length(bits)-1)
    if bits(i+1)==1
        x((i*n)+1:(i+0.5)*n)=1;
        x(((i+0.5)*n)+1:(i+1)*n)=-1;

    else
         x((i*n)+1:(i+0.5)*n)=-1;
        x(((i+0.5)*n)+1:(i+1)*n)=1;
    end
end

subplot(2,1,1)
plot(x);
xlabel('Descrete time');
ylabel('Amplitude');
title('MANCHESTER-LINE CODING')
axis([0 2500 -2 2])
 sx=(a^2)*TB*(sinc(x1/2)).*(sinc(x1/2)).*(sin(pi*x1/2)).*(sin(pi*x1/2));
%power spectra MANCHESTER formila


subplot(2,1,2)
plot(sx);
xlabel('Frequency');
ylabel('Power');
title('MANCHESTER-POWER SPECTRUM-LINE CODING')
```