

Department of Electronics Engineering



Lab Manual
Of
Digital Communication Lab (ECC 308)
Academic Complex, 2nd Floor, Room No.215

INDIAN INSTITUTE OF TECHNOLOGY
(Indian School of Mines), DHANBAD
JHARKHAND-826004

List of The Experiments

<u>Sl.No.</u>	<u>Name of Experiments</u>	Page No.
1.	To generate uniform and normal random variables, also find the mean and variance of the distribution.	03-06
2.	To generate continuous sine waveform, square waveform, and saw-tooth waveform.	07-10
3.	(a) Generation of Pseudo Noise (PN) random sequence using hardware. (b) Recovery of the clock from PN sequence using Hardware.	11-16
4.	Study of PCM modulation, and demodulation using Hardware (Multisim).	17-22
5.	To study Pulse Code Modulation and Demodulation using MATLAB Simulink	23-33
6.	To study BPSK Modulation and plot BER using MATLAB	34-39
7.	To study QPSK/4-QAM Modulation and plot BER using MATLAB	40-45
8.	To study the Digital Signal transmission using Quadrature Amplitude Modulation (QAM) using MATLAB Tool	46-48
9.	To generate the line codes for a 10-bit dataset in a MATLAB simulation environment.	49-57
10.	To analyze the Delta Modulation and Demodulation using Multisim	58-63

EXPERIMENT NO. -1

AIM: -

- a. To generate uniform random variables, also find mean and variance of the distribution.
- b. To generate normal random variables, also find mean and variance of the distribution.

SOFTWARE USED: -

MATLAB R2016a

FUNCTIONS USED: -

- `rand (1,n)`: - To generate a defined 'n' number of random variables having uniform distribution in the range of [0,1].
- `randn (1,n)`: - To generate a defined 'n' number of random variables having normal distribution in the range $(-\infty, \infty)$.
- `mean ()`: - To find the mean of the generated data for the defined function.
- `variance ()`: - To find the variance of the generated data for the defined function.
- `plot (a,b)`: - For a graphical representation of data with 'a' belonging to the x-axis and 'b' belonging to the y-axis.
- `subplot (l,m,n)`: - Used for compiling more than one plot where l,m,n represents a total number of rows, columns, and figure position for the divided grid.
- `hist(a,b)`: - For bar chart representation of data with 'a' belonging to x-axis and 'b' belonging to y-axis.

THEORY: -

- a. Uniform random number generation is being carried out with the help of "rand" function.
- b. The probability density function of variables should be uniform over the range of [0,1].

Mean is the average value of numbers generated, symbolized as ‘ μ ’ can be calculated from data as $\frac{\sum_{i=1}^n a_i f_i}{\sum_{i=1}^n f_i}$ and can be narrowed to $\frac{a+b}{2}$; where a,b are the distribution ranges for uniform probability density function.

Theoretically mean of the random numbers should be 0.5, accuracy of which depends on the number of samples generated.

Variance is an aspect which measures the deviation of random variables from its mean value. For a uniform random function ranges from [a,b] variance can be calculated by $\frac{(a-b)^2}{12}$. Theoretically variance of uniform

random numbers distributed in [0,1] as calculated from the given formula equals to 0.83

a) Generation of random numbers having normal(gaussian) probability density function is being carried out with the help of “randn” function. The probability density function of variables ranges in $(-\infty, \infty)$ defined by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} .$$

Mean ‘ μ ’ for standard normal distribution function should be zero while the variance ‘ σ^2 ’ should be one.

MATLAB CODE: -

- a) Code for generating uniform distribution random variables

```
clc
clear allclose
all

a = rand (1,5500);b =
mean (a);
c = var (a);
disp (b);
disp (c); figure (1);
subplot(211);plot
(a);
title ('Uniform Distribution of Random Variables');xlabel ('Sample Number')
ylabel ('Amplitude')
subplot(212); hist(a);
title ('Histogram of Uniform Distribution');xlabel ('Numerical
Value');
ylabel ('Total no. of Occurence');
```

- b) Code for generating normal distribution random variables

```
clc
clear allclose
all

a = randn (1,5500);b = mean
(a);
c = var (a);
disp (b);
disp (c); figure (1);
subplot(211);plot
(a);
title ('Normal Distribution of Random Variables');xlabel ('Sample
Number')
ylabel ('Amplitude')
subplot(212); hist(a);
title ('Histogram of Normal Distribution');xlabel ('Numerical
Value');
ylabel ('Total no. of Occurence');
```

OUTPUT: -

- a) For the generated random variables having uniform density function

Mean = 0.5037; Variance = 0.0825

- b) For the generated random variables having normal density function

Mean = 0.0065; Variance = 0.9911

Figure 1(a)

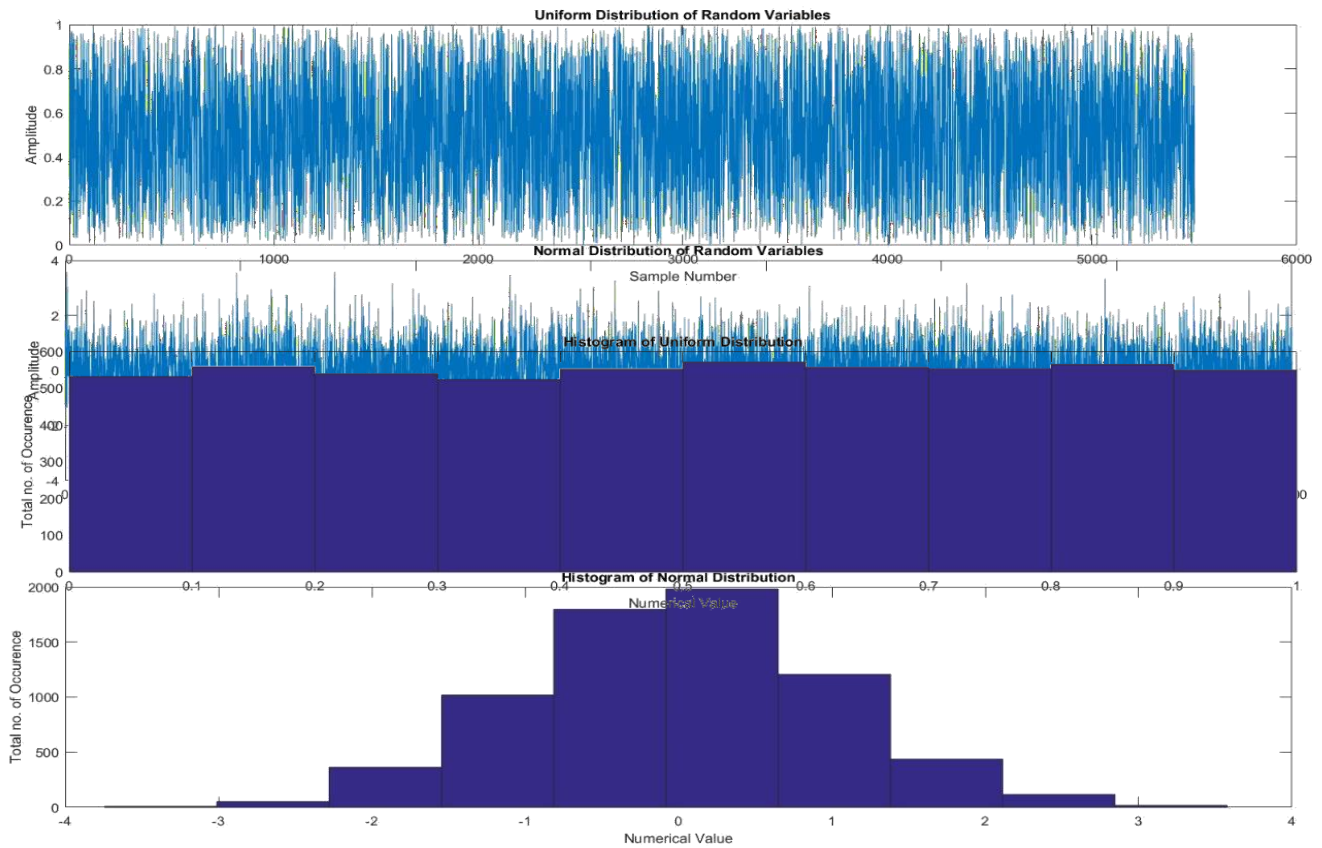


Figure 1(b)

CONCLUSIONS: -

- With the generated number of random variables the histogram is close to theoretical uniform and normaldistributions.
- For uniform random variables error in mean and variance is observed as 0.74% and 0.96% respectively.
- For normal random variables error in mean and variance is observed as 0.65% and 0.89% respectively.
- As the number of generated random variable increases the distribution function, mean and varianceattains standard values.

EXPERIMENT 2.

AIM: -

- a. To generate continuous sine waveform, square waveform and sawtooth waveform.
- b. To generate discrete sine waveform, square waveform and sawtooth waveform.

SOFTWARE USED: -

Matlab R2016a

FUNCTIONS USED: -

- $\sin(2*\pi*f*t)$: - It generates sine waveform with time period $1/f$.
- $\text{square}(t)$: - It generates square waveform with time period 2π for the elements of time vector t .
- $\text{sawtooth}(t)$: - It generates sawtooth waveform with time period 2π for the elements of time vector t .
- $\text{stem}(_,_)$: - It provides graphical representation of data in discrete manner.
- $\text{axis}([a\ b\ c\ d])$: - It sets the limit for current axes. The four element vector a,b,c,d specifies minimum and maximum for x and y axis respectively.
- $\text{plot}(a,b)$: - For graphical representation of data with 'a' belonging to x axis and 'b' belonging to y axis.
- $\text{subplot}(l,m,n)$: - Used for compiling more than one plot where l,m,n represents total number of rows, columns and figure position for the divided grid.

THEORY:-

The sine function generates sine wave with a period of $1/f$ where 'f' is specified frequency.

The sawtooth function generates a sawtooth wave with peaks at +/- 'A' and a period of 2π . An optional width parameter specifies a fractional multiple of 2π at which the signal's maximum occurs.

The square function generates a square wave with a period of 2π . An optional parameter specifies duty cycle, the percent of the period for which the signal is positive.

Waveforms generated can be amplified or attenuated by multiplying the function by a scalar element. Timescaling is a technique that alters the time period of the signal thus expanding or contracting the same.

MATLAB CODE: -

a) To generate continuous waveforms

```
clc clear all
close all
t = 0:0.01:10
f = 3;
a = 3*sin(2*pi*f*t);
b = 5*square(3*t);
c = 4*sawtooth(3*t);
figure(1);
subplot(311);
plot(t,a);
title ('Sine waveform');
xlabel ('Time(continuous)');
ylabel ('Amplitude')
axis ([0 3 -6 6]);
subplot(312);
plot(t,b);
axis ([0 8 -7 7]);
title ('Square waveform');
xlabel ('Time(continuous)')
ylabel ('Amplitude')
subplot(313);
plot(t,c);
axis ([0 8 -5 5]);
title ('Sawtooth waveform');
xlabel ('Time(continuous)')
ylabel ('Amplitude')
```

b) To generate discrete waveforms

```
n = 0:0.05:10
f = 2
a = 3*sin(2*pi*f*n)
b = 5*square(5*n);
c = 4*sawtooth(3*n);
figure(1);
subplot(311);
stem(n,a);
title ('Sine waveform');
xlabel ('Time(discrete)')
ylabel ('Amplitude')
axis ([0 3 -6 6]);
subplot(312);
stem(n,b);
axis ([0 8 -7 7]);
title ('Square waveform');
xlabel ('Time(discrete)')
ylabel ('Amplitude')
subplot(313);
stem(n,c);
```



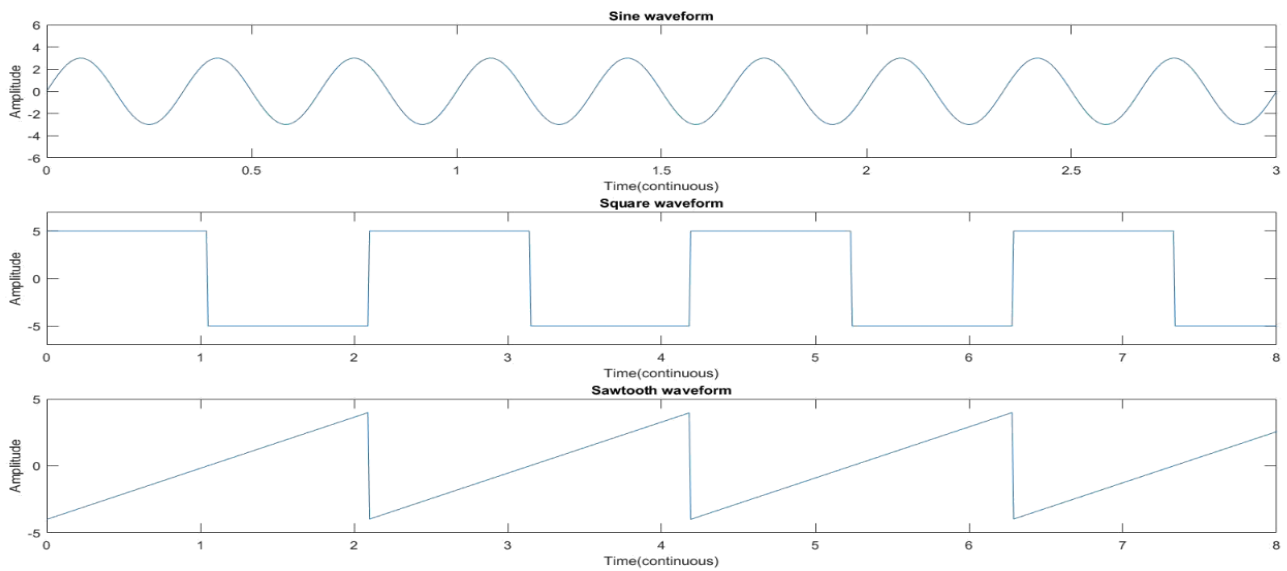
```
axis ([0 8 -5 5]);  
title ('Sawtooth waveform');  
xlabel ('Time(discrete)')  
ylabel ('Amplitude')
```

OUTPUT: - a) Generated continuous sine, square and sawtooth waveforms. Figure 2(a) b) Generated discrete sine, square and sawtooth waveforms. Figure 2(b)

OUTPUT: -

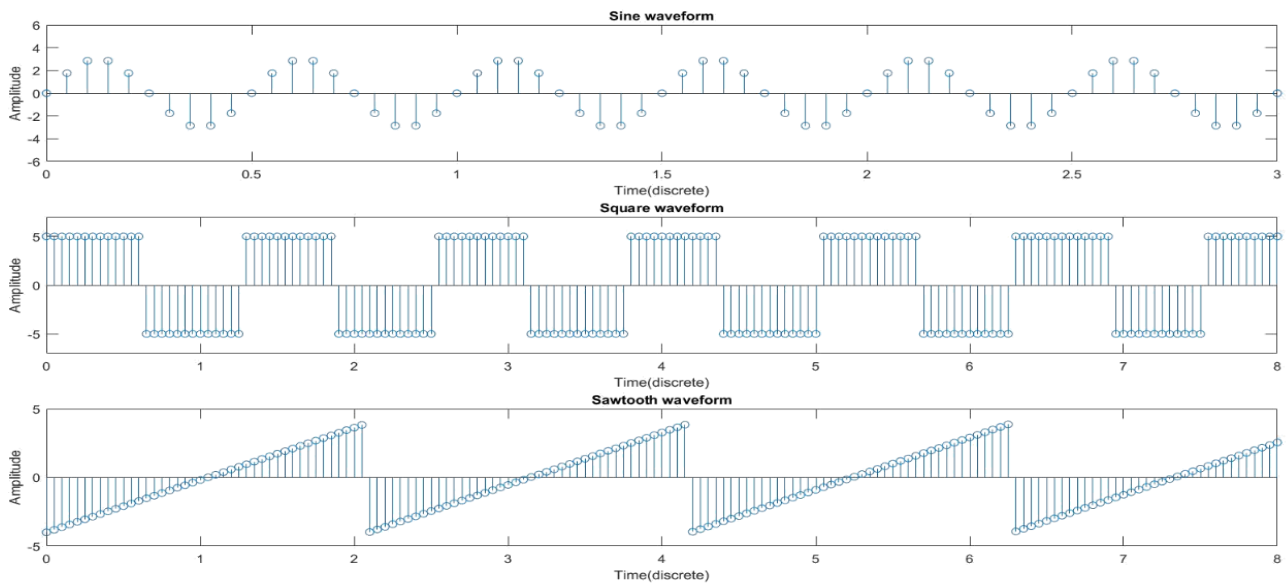
a) Generated continuous sine, square and sawtooth waveforms.

Figure 2(a)



b) Generated discrete sine, square and sawtooth waveforms

Figure 2(b)



CONCLUSIONS: -

- a. The process of time scaling and amplification is applied and observed along with the frequency change in the generated signals.
- b. The interval distribution for the continuous signal generation should be low to obtain optimum waveforms.
- c. The time spacing for discrete signal generation should be observed for the functional values to be differentiable.

EXPERIMENT NO.-3

Aim: Generation of Pseudo Noise sequence and recovery of the clock.

Objective:

- (i) Verification of maximal length sequence by changing the Tapping and SEED.
- (ii) Verification of amplitude and frequency of recovered clock with input clock.

Apparatus/ Software Required:

- (i) MultisimSoftware
- (ii) PC/ Laptop

Theory:

Pseudo-Noise (PN) sequences are commonly used to generate noise that is approximately "white". It has applications in scrambling, cryptography, and spread-spectrum communications. It is also commonly referred to as the Pseudo-Random Binary Sequence (PRBS). These are very widely used in communication standards these days. The qualifier "pseudo" implies that the sequence is not truly random. Actually, it is periodic with a (possibly large) period, and exhibits some characteristics of a random white sequence within that period.

Pseudo random noise generator built from Linear Feedback Shift Register (LFSR) with judicious selection of the XOR taps feedback path. Pseudo random number generators generate a stream of numbers in a known pattern. The pattern is typically very long and it is hard to recognize the sequence of numbers is ordered. LFSR is a linear feedback shift register whose input bit is a linear function of previous function that contains the signal through the register from one bit to the next most significant when it is clocked. Figure 1 shows the block diagram of LFSR, a 4-bit LFSR generate 24-1 different non zero bit pattern by performing exclusive-OR gate on the outputs of two or more of the flip-flops and feeding those outputs back in to the input of one of the flip-flops.

Clock recovery from the data stream is expedited by modifying the transmitted data. Wherever a serial communication channel does not transmit the clock signal along with the data stream, the clock must be regenerated at the receiver, using the timing information from the data stream. Clock recovery is a common component of systems communicating over wires, optical fibers, or by radio.

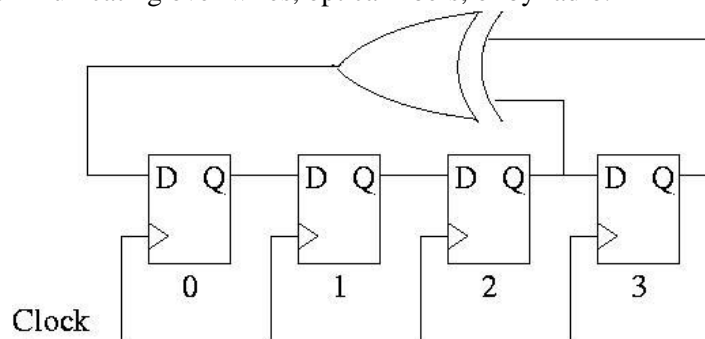
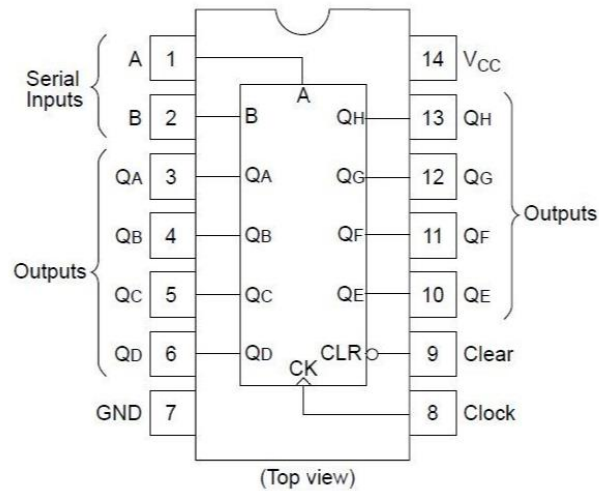


Figure1: Block diagram of LFSR

Pin Configuration of IC74164N:



Procedures:

1. Take the length of PN sequence (N).
2. Find no. of shift registers using $2^n - 1 = N$, where n is the no. of shift registers.
3. Select IC 74164N as a shift register from the component library.
4. Make the connection according to the given circuit diagram.
5. Verify that PN sequence must repeat after every $2^n - 1$ period.
6. Apply generated PN sequence to IC 74123 to get rising and falling edge pulses.
7. After xoring, pass this to BFP followed by a comparator to get original clock.

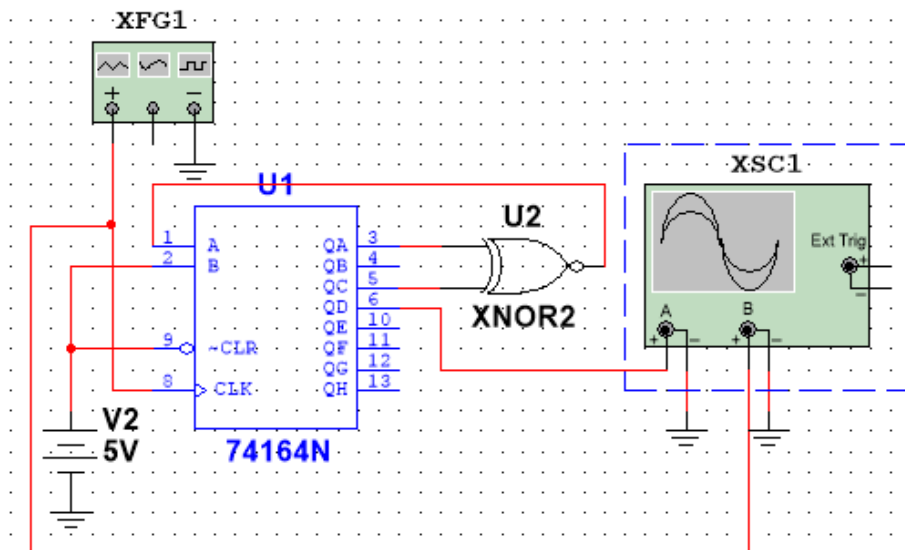


Figure 2: Circuit diagram to generate required PN sequence.

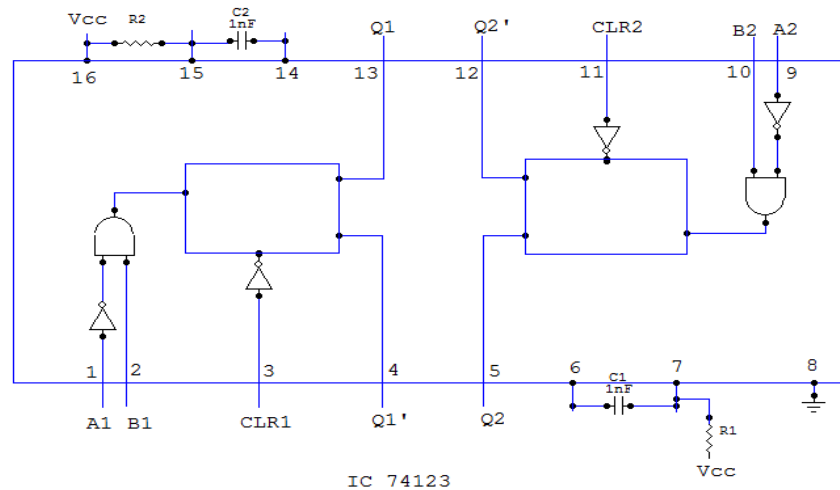
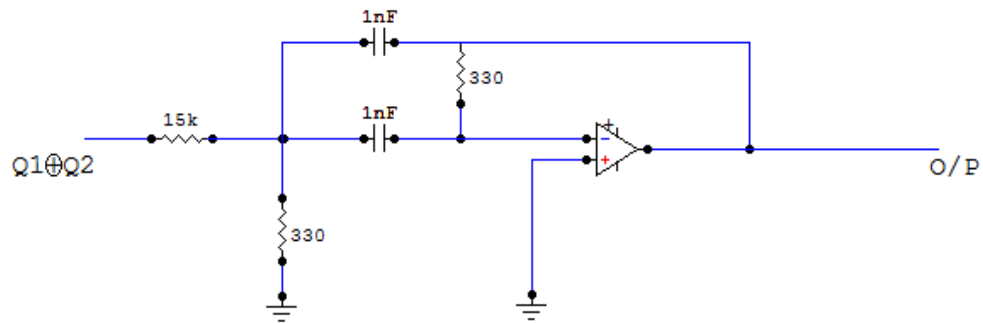
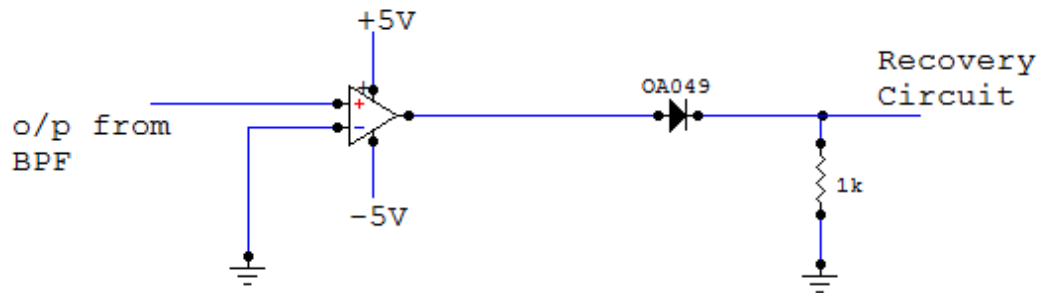


Figure 3: Interfacing IC 74123 with pn sequence generator



A BAND-PASS FILTER

Figure 4: BPF with cut-off frequency equal to the clock frequency



COMPARATOR

Figure 5: Comparator Circuit

Observation Table:

1. Take initial sequence as 0100, 0101 and 0110.
2. Find the output after every clock.
3. Make sure the output is repeated after 2^n-1 clock.

Table I

clock	Q _A	Q _B	Q _C	Q _D	Output
1.	0	1	0	0	
2.	
3.	
4.	
5.	
...	
...	
...	
...	
...	
...	
...	
...	
...	
...	

Table II

n	Primitive Polynomial
1	$1+x$
2	$1+x+x^2$
3	$1+x+x^3, 1+x^2+x^3$
4	$1+x+x^4, 1+x^3+x^4$

Results:

1. The above sequences are generated and verified.
2. If n registers are used then a PN sequence with $2^n - 1$ bits are generated if it is a maximal length sequence.
3. Clock is recovered which has slight modification in amplitude and frequency because of poorly tuned BPF.

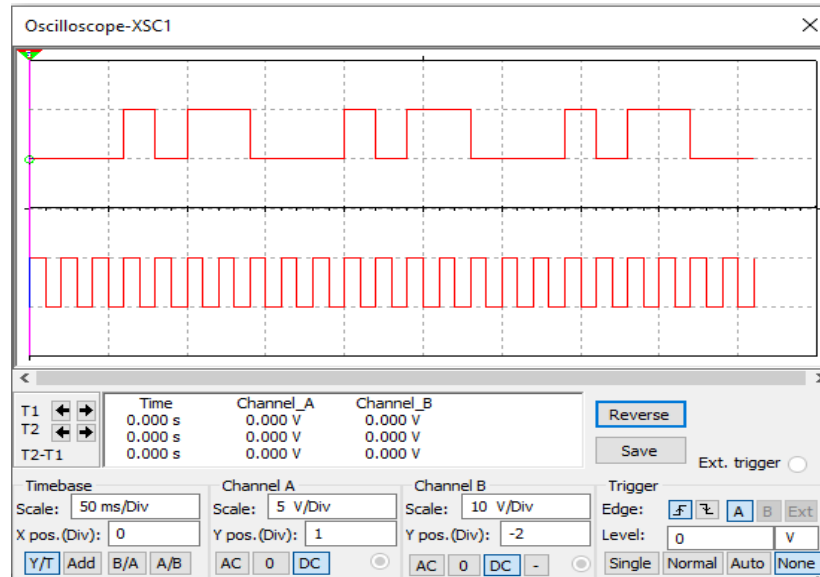


Figure 6: Generated PN sequence in Multisim software.

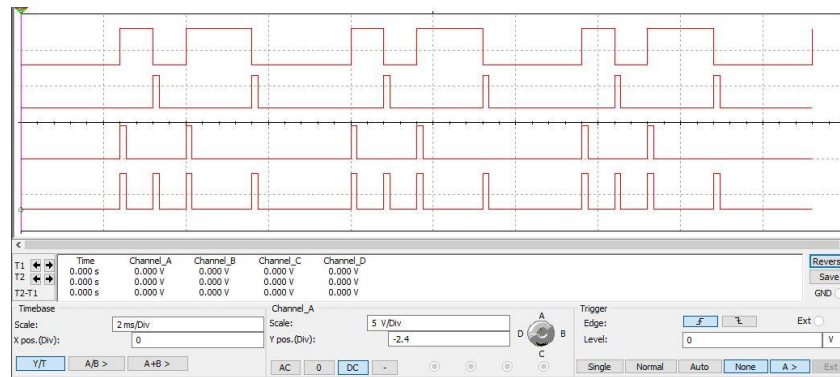


Figure 7: Rising and falling edge pulses with respect to input clock.



Figure 8: Output of BPF and recovered clock.

EXPERIMENT 4

Aim: To study Pulse Code Modulation and Demodulation technique.

Apparatus and Component Required: IC 7474, IC 7486, IC 741, Resistors and capacitor, function generator, and connecting wires

Theory:

The pulse code modulator technique samples the input signal $x(t)$ at a sampling frequency. This sampled variable amplitude pulse is then digitalized by the analog to digital converter. Figure (1) shows the PCM generator.

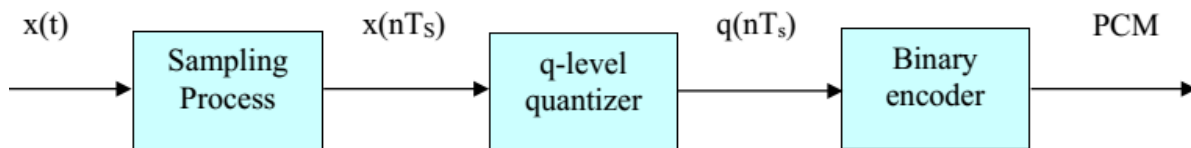


Figure.(1): PCM modulator

In the PCM generator, the signal is first passed through sampler which is sampled at a rate of (f_s) where:

$$f_s \geq 2f_m$$

The output of the sampler $x(nT_s)$ which is discrete in time is fed to a q level quantizer. The quantizer compares the input $x(nT_s)$ with its fixed levels. It assigns any one of the digital level to $x(nT_s)$ that results in minimum distortion or error. The error is called quantization error, thus the output of the quantizer is a digital level called $q(nT_s)$. The quantized signal level $q(nT_s)$ is binary encode. The encoder converts the input signal to v digits binary word.

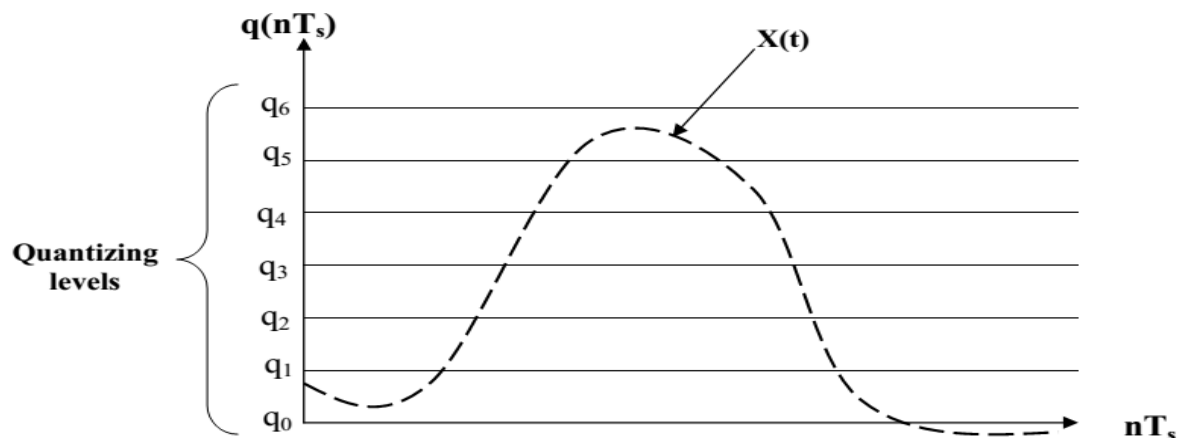


Figure.(3) shows the block diagram of the PCM receiver. The receiver starts by reshaping the received pulses, removes the noise and then converts the binary bits to analog. The received samples are then filtered by a low pass filter; the cut off frequency is at f_c .

$f_c = f_m$

where f_m : is the highest frequency component in the original signal.

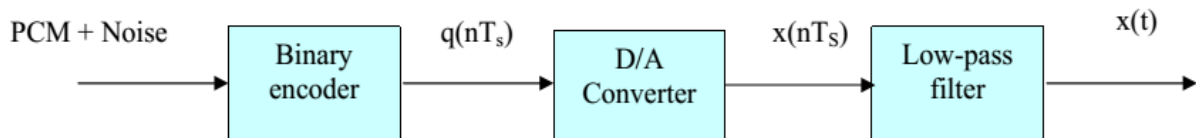


Figure. (3): PCM demodulator

It is impossible to reconstruct the original signal $x(t)$ because of the permanent quantization error introduced during quantization at the transmitter. The quantization error can be reduced by the increasing quantization levels. This corresponds to the increase of bits per sample (more information). But increasing bits (ν) increases the signaling rate and requires a large transmission bandwidth. The choice of the parameter for the number of quantization levels must be acceptable with the quantization noise (quantization error). Figure.(4) shows the reconstructed signal

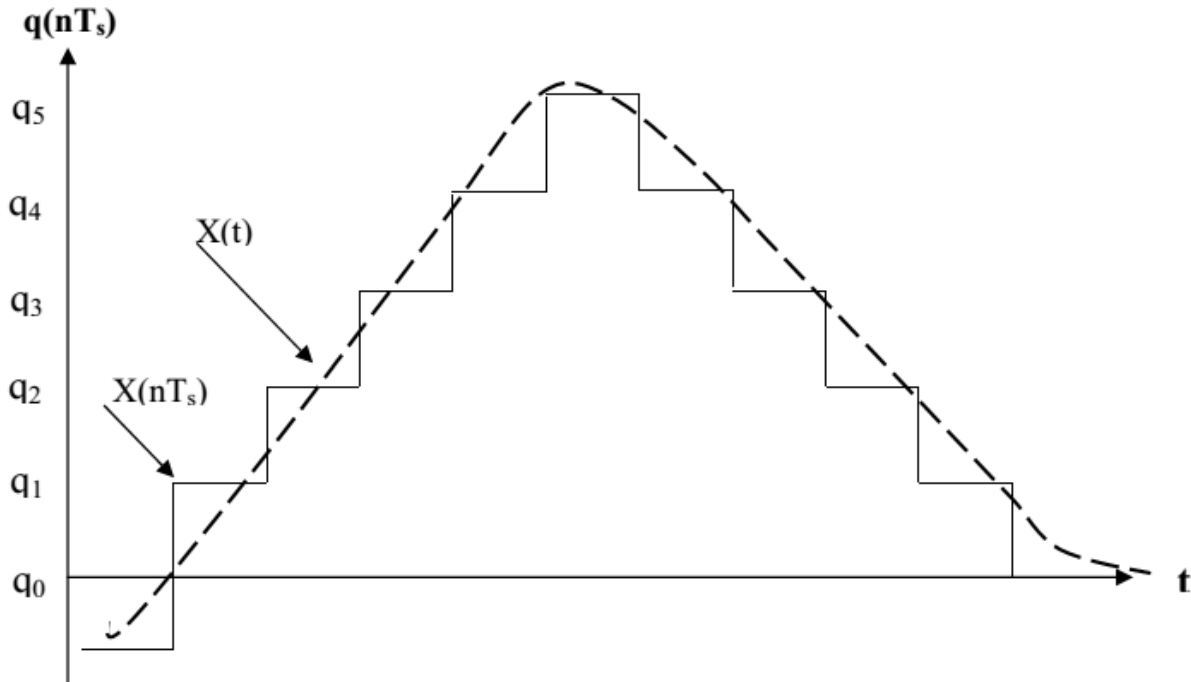
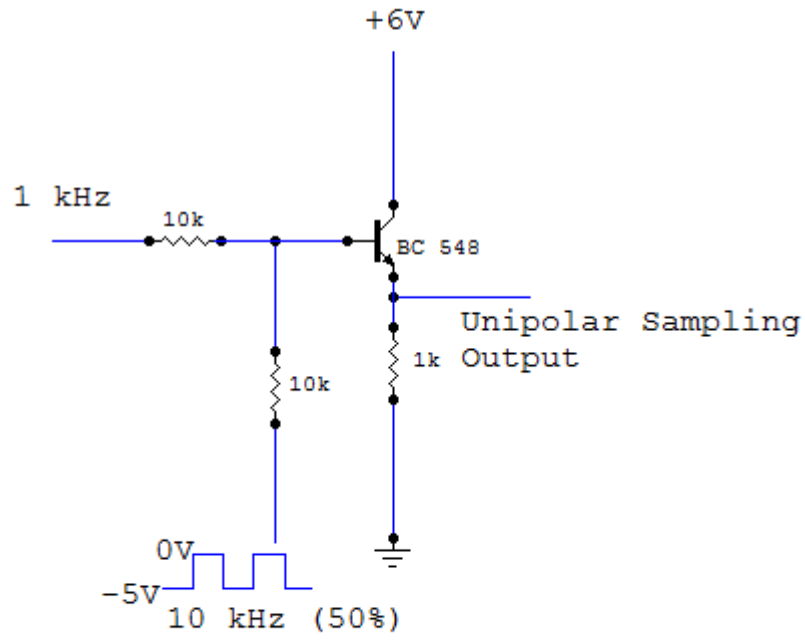


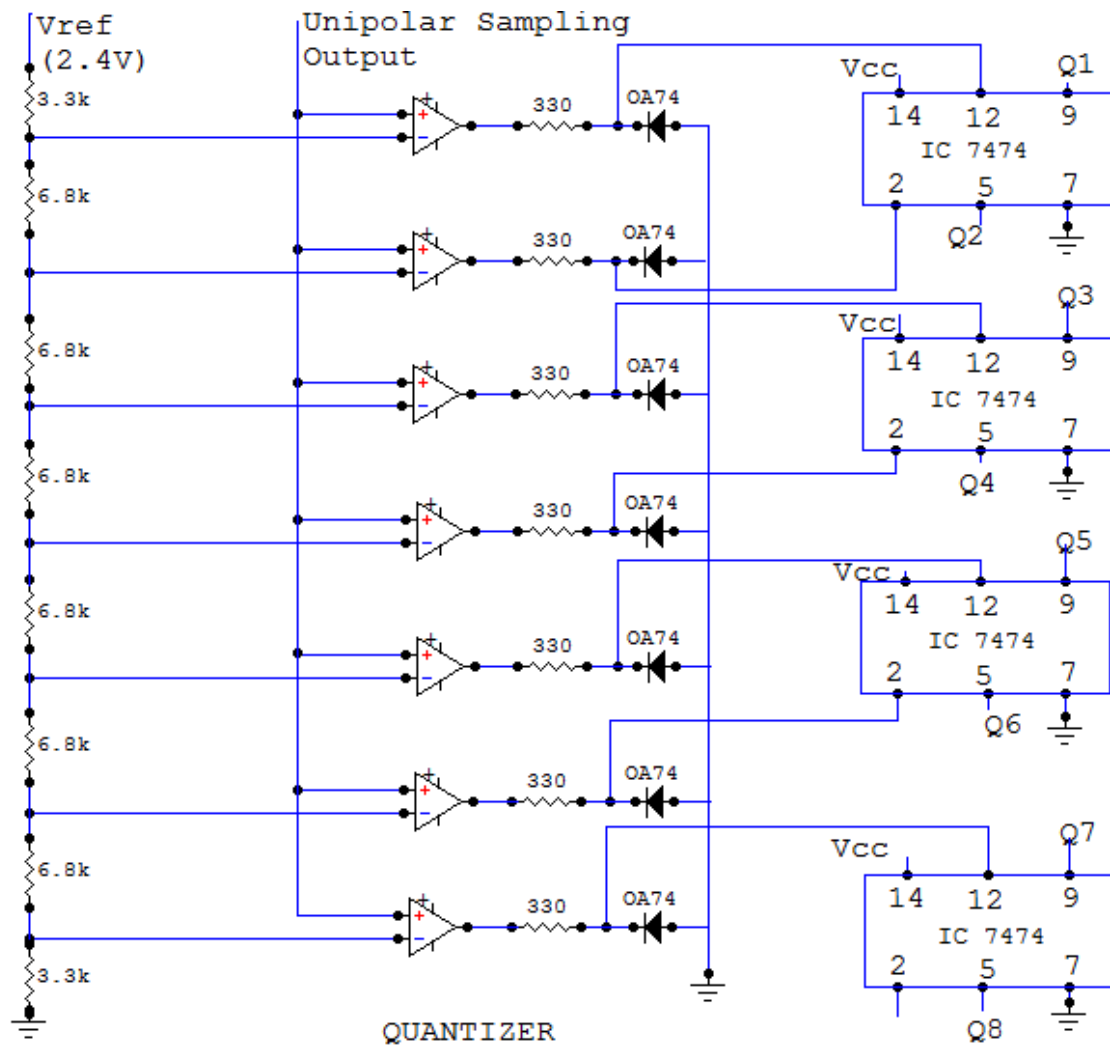
Figure (4): The reconstructed signal

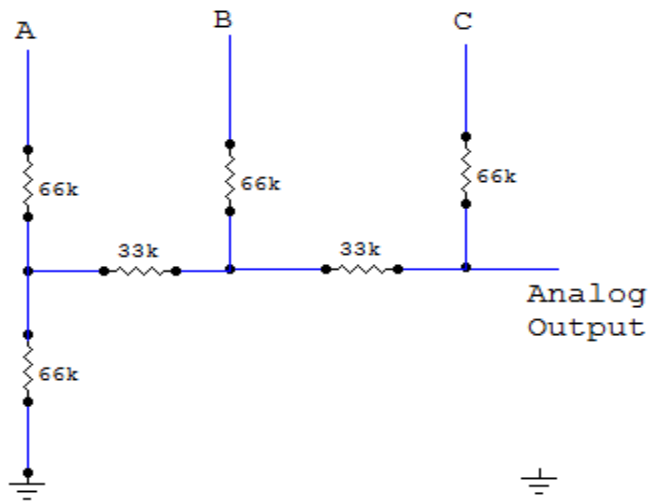
Procedure:

1. The given below circuit is used for 3 bit PCM.
2. In PCM we start with the sampling of input sinusoidal 1 kHz analog signal. In sampling we multiply input signal with 10 kHz (two times greater than input signal frequency) square wave signal. Sampling output is taken from common collector BJT.
3. Use sampling output as one input of comparator and the other input of comparator is different potential of V_{ref} . For 3 bit PCM 7 comparator is use. The output of comparator is in digital form which store in D flip flop using 7474 IC.
4. The digital output is encoded using XOR gates. The composition of XOR gates is given in encoder circuit. Here 3 outputs are because we encode each sampled value using 3 bit.
5. In demodulation we convert encoded digital data to analog data. The output of D/A converter is not smooth. To make it smooth (or accurate) pass from low pass filter.

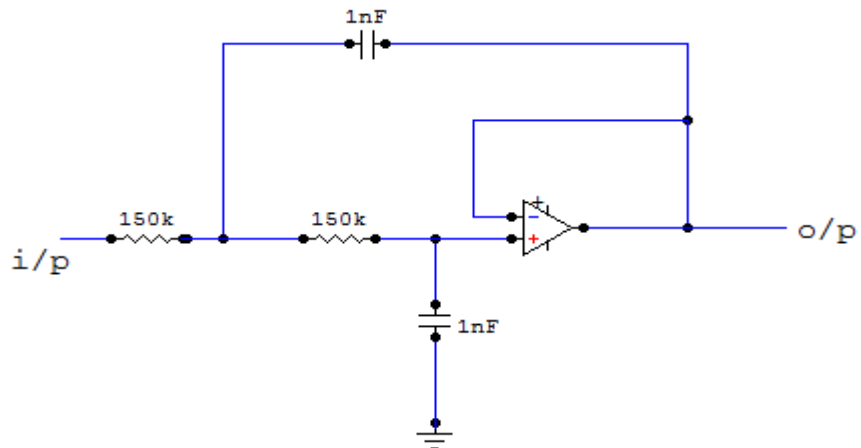


Circuit of Sampler





D/A Converter



2nd Order low pass unity gain active filter

Result:

Precaution:

1. The connections should be made properly and tightly.
2. Check all the connections before switching ON the kit.

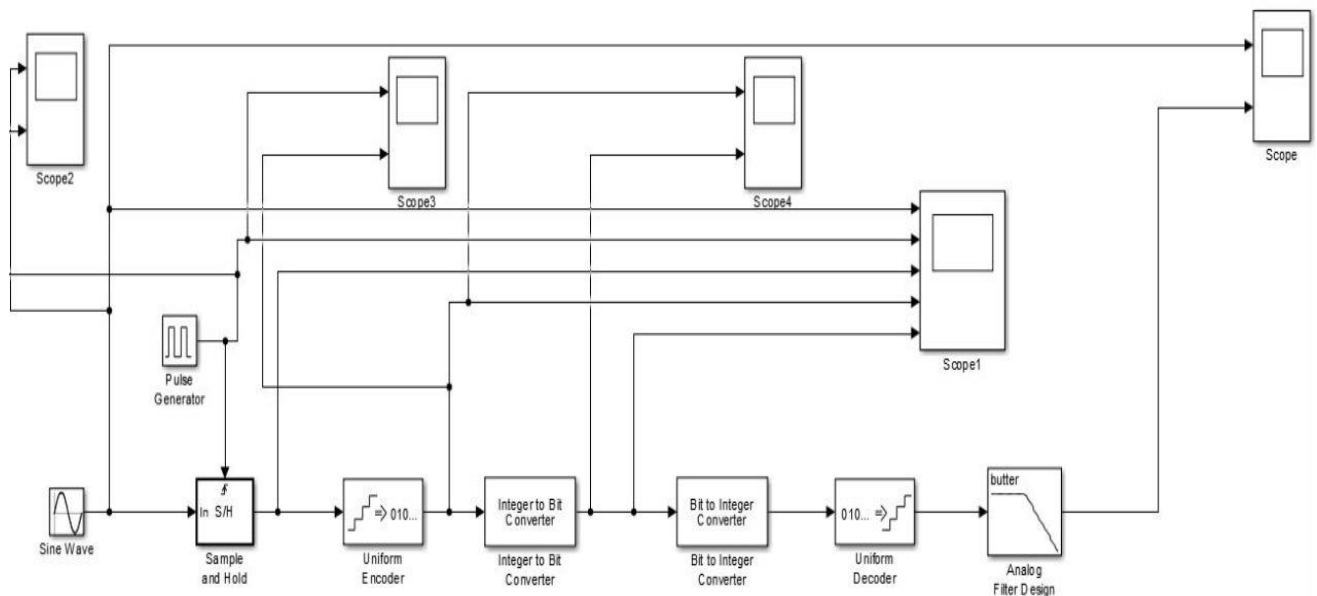
EXPERIMENT – 05

AIM: Study of PULSE CODE MODULATION (PCM) and its DEMODULATION Scheme through SIMULINK Models on MATLAB.

COMPONENTS REQUIRED:

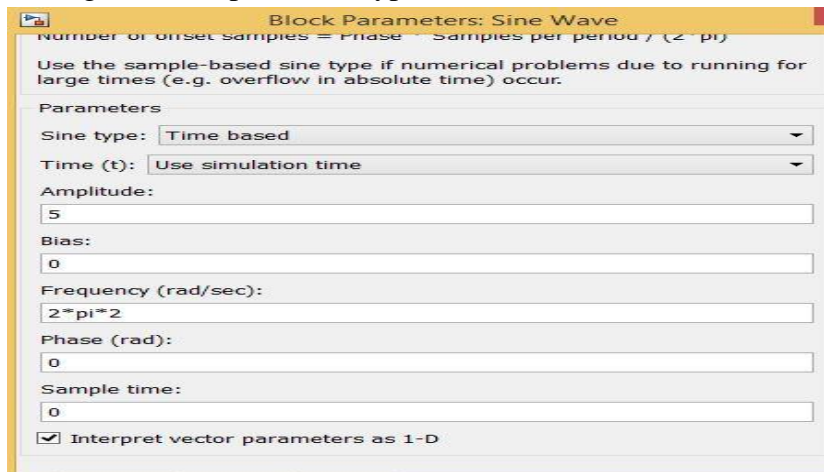
- ❖ Sinusoidal Wave Generator
- ❖ Pulse Generator
- ❖ Sample & Hold
- ❖ Uniform Encoder
- ❖ Integer To Bit Converter
- ❖ Bit To Integer Converter
- ❖ Uniform Decoder
- ❖ Analog Filter Design
- ❖ Scope(S)

Simulink Model of Pulse Code Modulation (PCM)

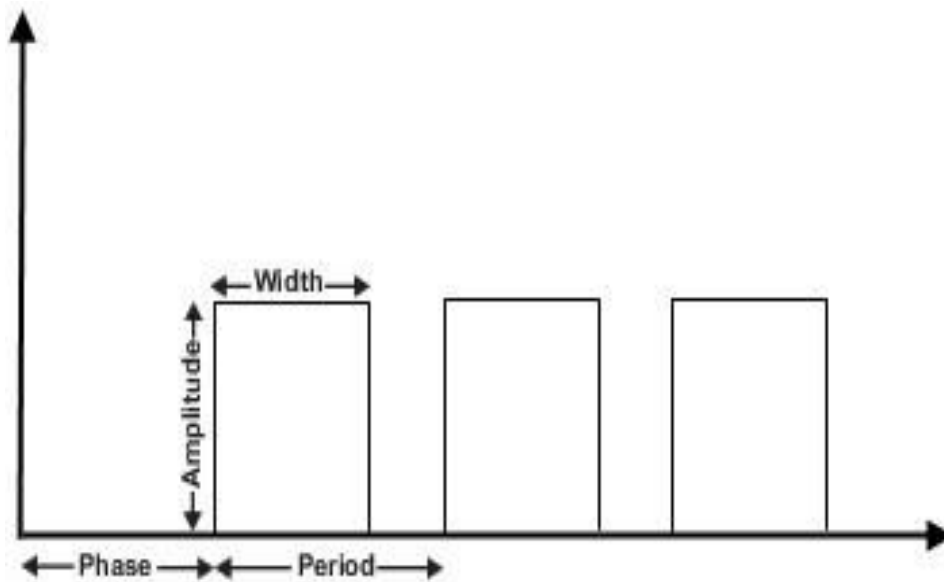
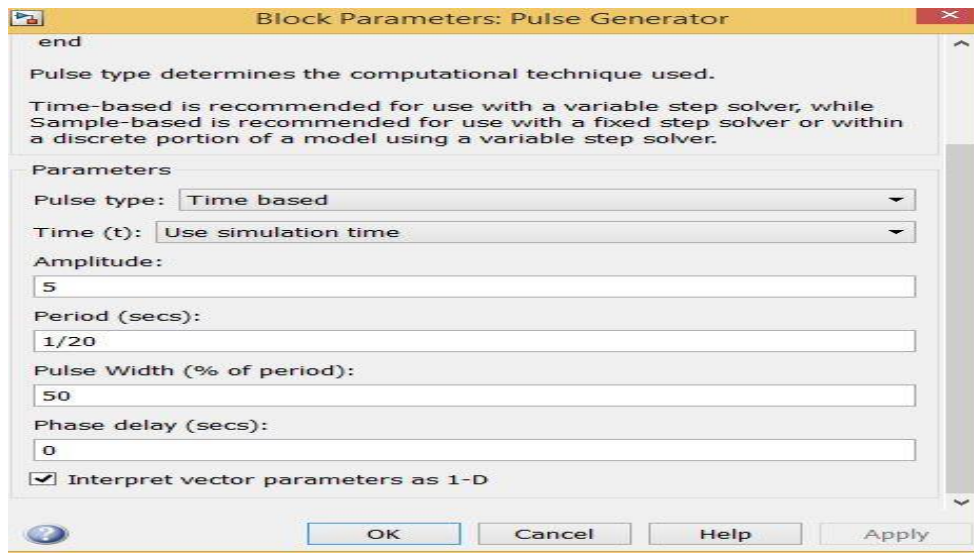



BLOCK(s) used in PCM scheme:

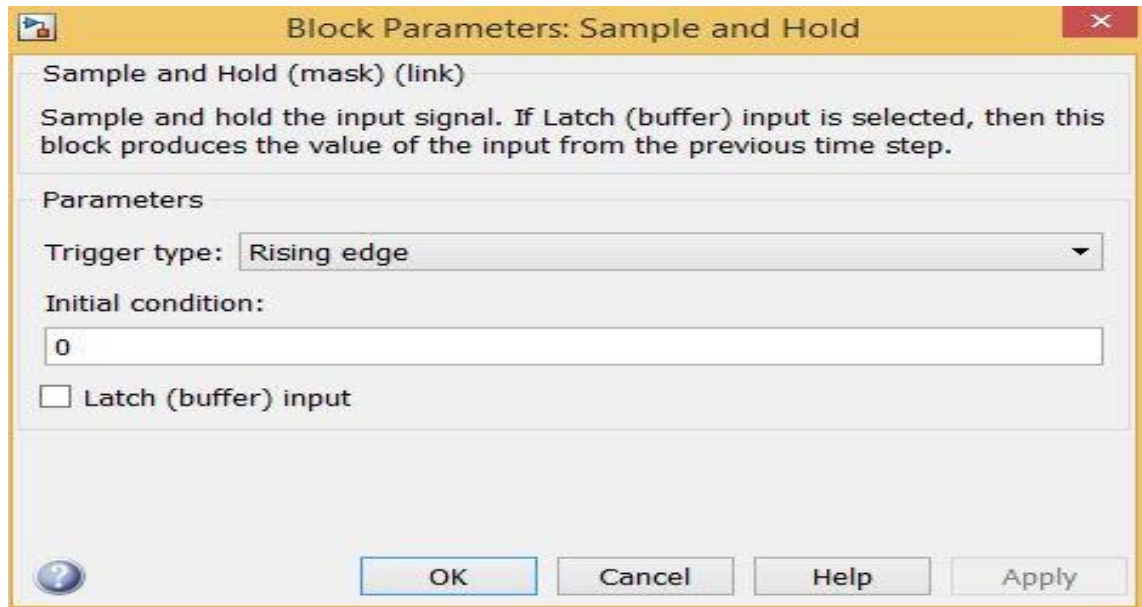
- **SINUSOIDAL WAVE GENERATOR:** The Sine Wave block generates a multichannel real or complex sinusoidal signal, with independent amplitude, frequency, and phase in each output channel. The block supports floating point and signed fixed-point data types.



- **PULSE GENERATOR:** The Pulse Generator block generates square wave pulses at regular intervals. The block waveform parameters, Amplitude, Pulse Width, Period, and Phase delay, determine the shape of the output waveform. The following diagram shows how each parameter affects the waveform. The Pulse Generator block can emit scalar, vector, or matrix signals of any real data type. To emit a scalar signal, use scalars to specify the waveform parameters. To emit a vector or matrix signal, use vectors or matrices, respectively, to specify the waveform parameters. Each element of the waveform parameters affects the corresponding element of the output signal. For example, the first element of a vector amplitude parameter determines the amplitude of the first element of a vector output pulse. All the waveform parameters must have the same dimensions after scalar expansion. The data type of the output is the same as the data type of the Amplitude parameter.

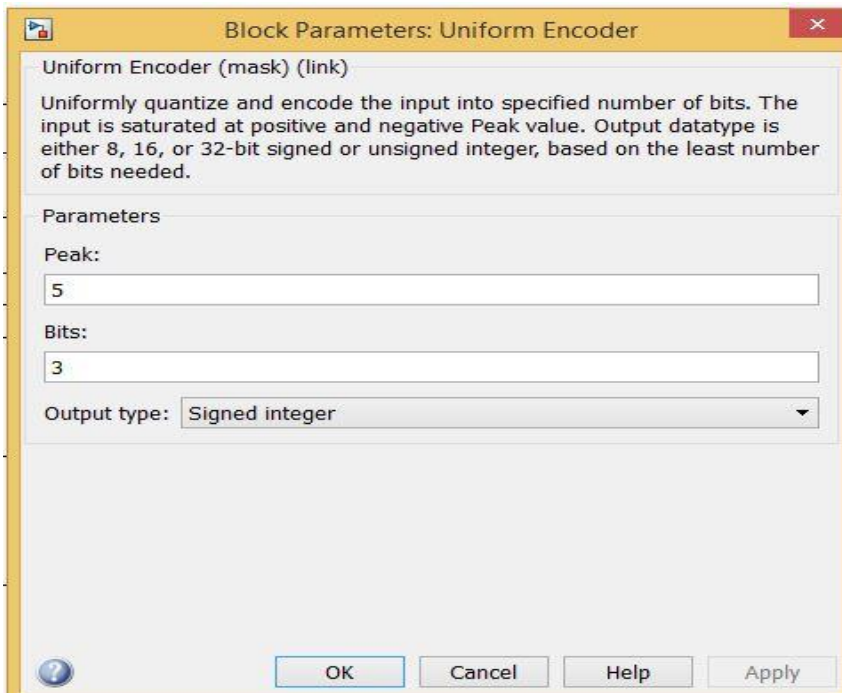


- **SAMPLE and HOLD:** The Sample and Hold block acquires the input at the signal port whenever it receives a trigger event at the trigger port (marked by ). The block then holds the output at the acquired input value until the next triggering event occurs.

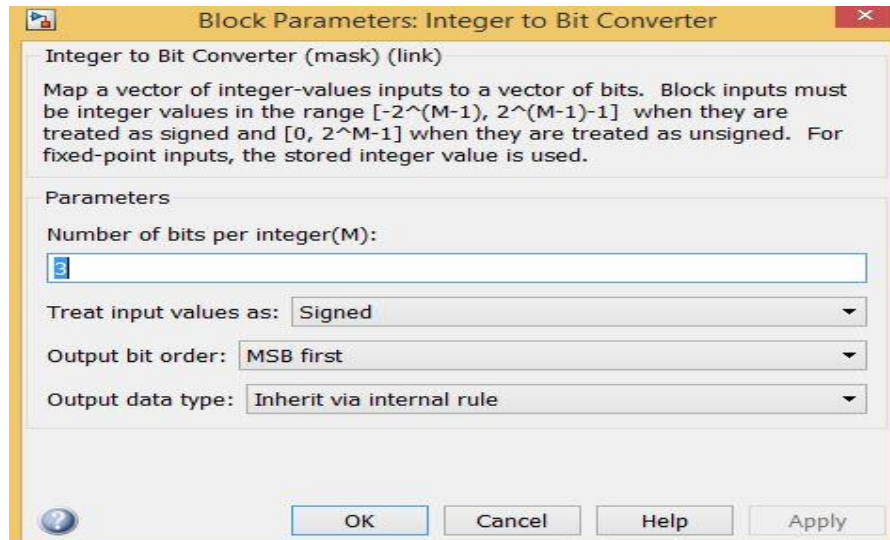


UNIFORM ENCODER: The Uniform Encoder block performs the following two operations on each floating-point sample in the input vector or matrix:

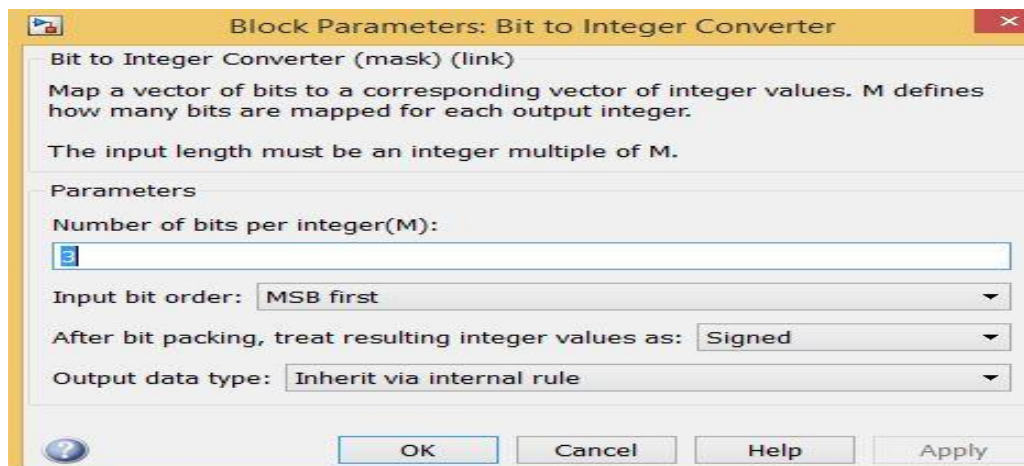
- Quantizes the value using the same precision.
- Encodes the quantized floating-point value to an integer value.



- **INTEGER TO BIT CONVERTER:** The Integer to Bit Converter block maps each integer (or fixed-point value) in the input vector to a group of bits in the output vector. This block is single-rate and single-channel.

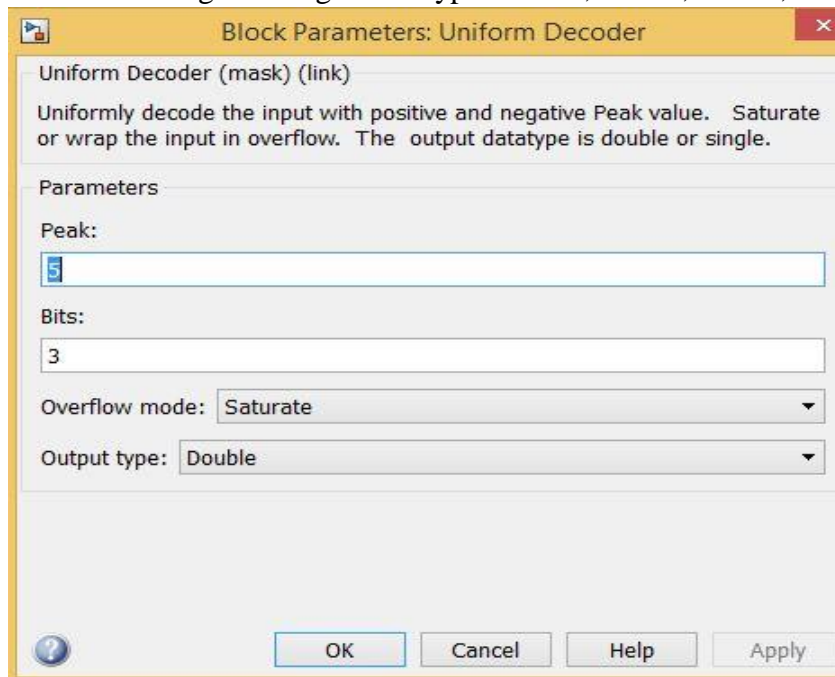


- **BIT TO INTEGER CONVERTER:** The Bit to Integer Converter block maps groups of bits in the input vector to integers in the output vector. If M is the Number of bits per integer parameter, then the block maps each group of M bits to an Integer between 0 and 2^M-1 . As a result, the output vector length is 1/M times the input vector length.



- **UNIFORM DECODER:** The Uniform Decoder block performs the inverse

operation of the Uniform Encoder block, and reconstructs quantized floating-point values from encoded integer input. Inputs can be real or complex Values of the following six integer data types: uint8, uint16, uint32, int8, int16.



- **ANALOG FILTER DESIGN:** The Analog Filter Design block designs and implements a Butterworth, Chebyshev type I, Chebyshev type II, elliptic, or Bessel filter in a high pass, low pass, band pass, or band stop configuration. The Analog Filter Design block uses a state-space filter representation, and applies the filter using the State-Space (Simulink) block in the Simulink Continuous library. All of the design methods use Signal Processing Toolbox functions to design the filter.
- **SCOPE:** The Scope block displays its input with respect to simulation time. The Scope block can have multiple axes (one per port); all axes have a common time range with independent y-axes. The Scope allows you to adjust the amount of time and the range of input values displayed.

OBSERVATION:

❖ Output of Scope 2 & Scope 3

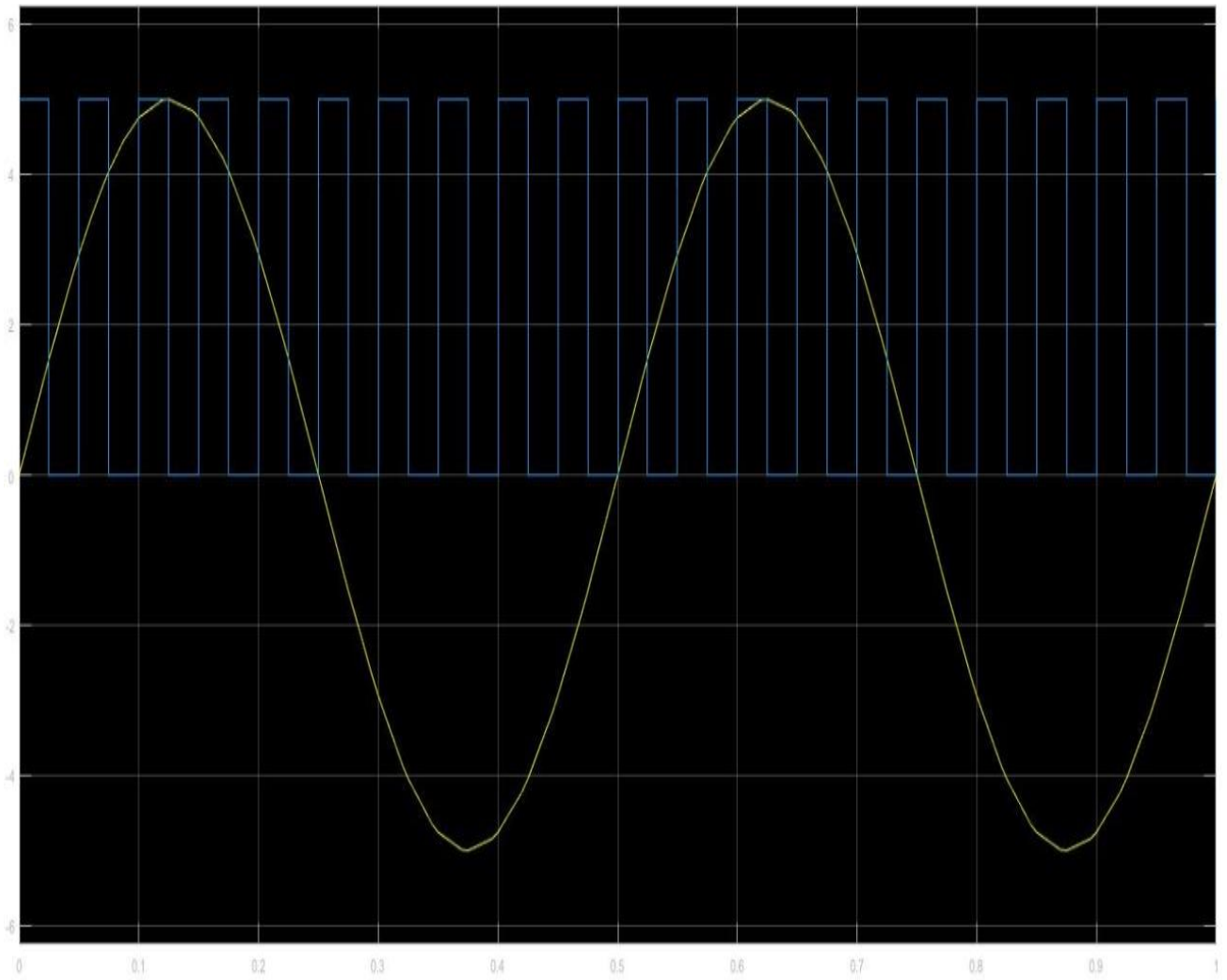


Fig.1 Sine wave & Pulse generator output

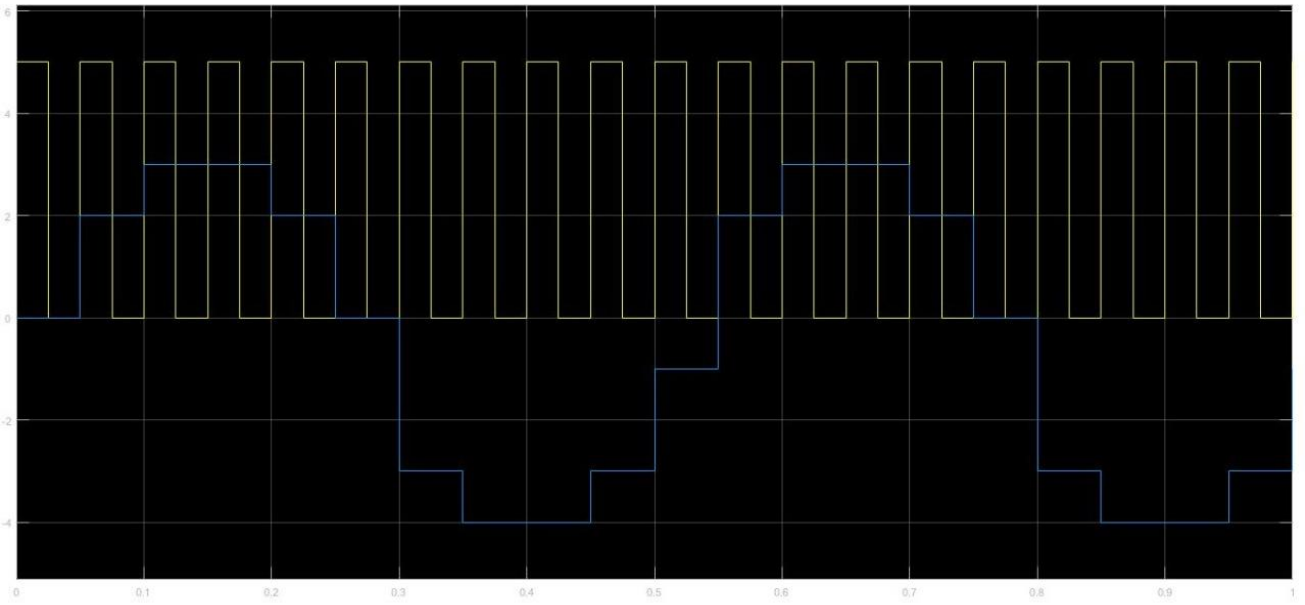


Fig.2 Pulse generator & Uniform Encoder Output

❖ Output of Scope 4

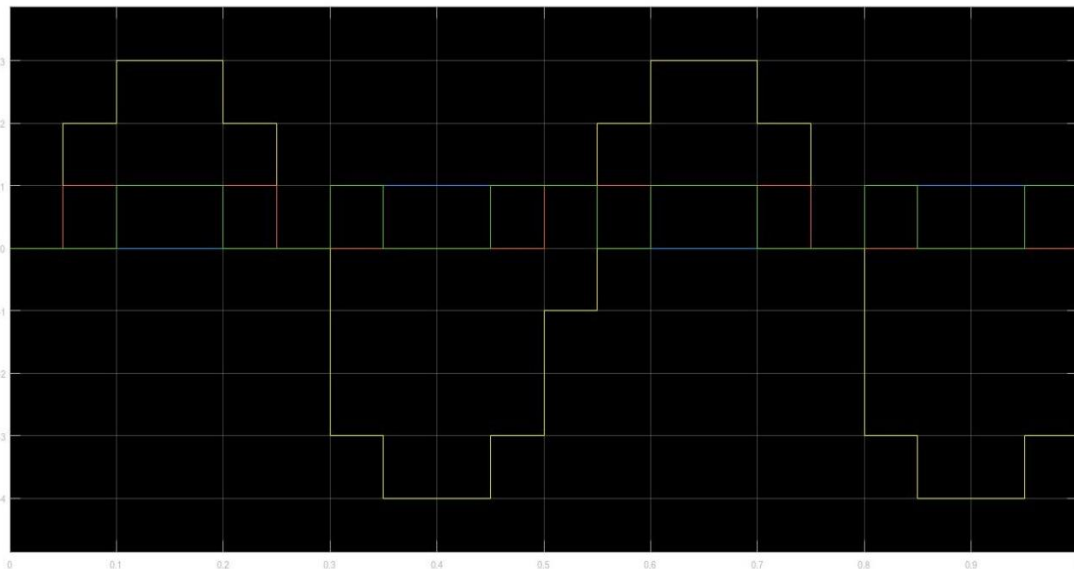


Fig. 3 Uniform Encoder & Integer to Bit Converter Output

❖ **Output of Scope 1**

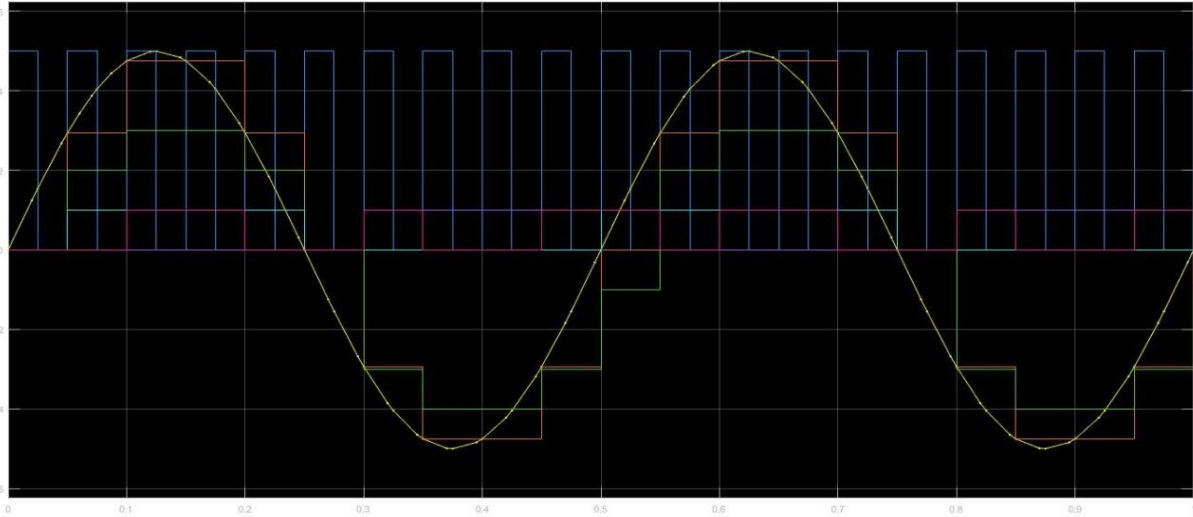


Fig.4 Sine wave, Pulse generator, Sample & hold, Uniform Encoder, Integer to Bit Converter output

❖ **Final Output**

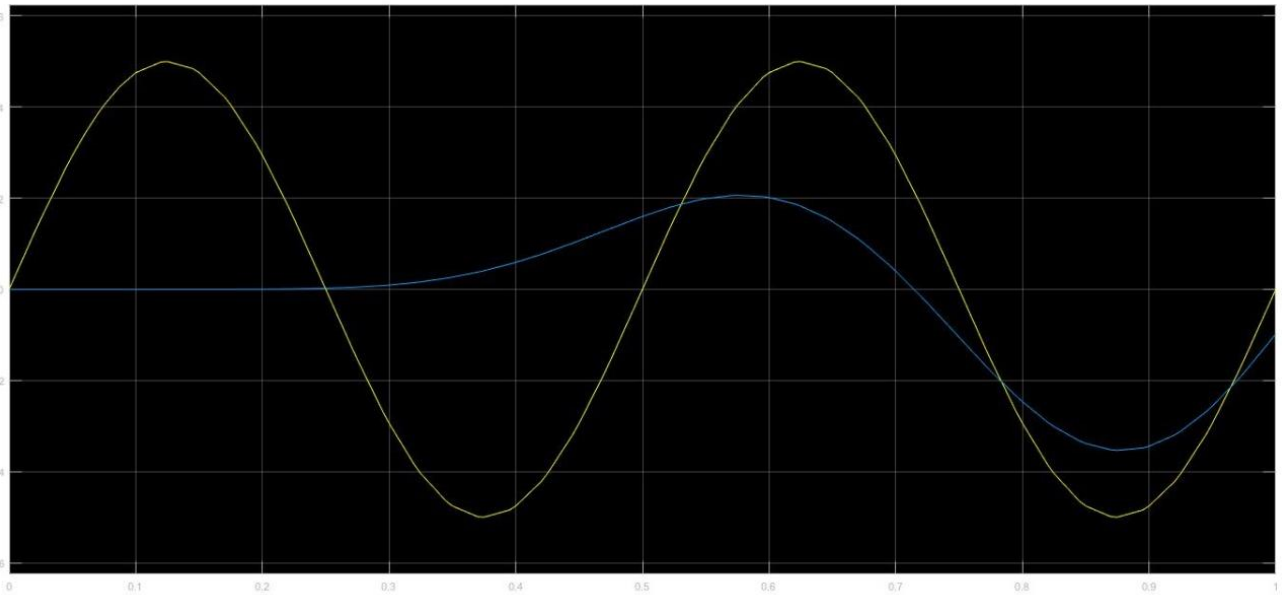


Fig.5 Sine wave & Analog Filter Design output

❖ SINE WAVE & ANALOG FILTER DESIGN OUTPUT:

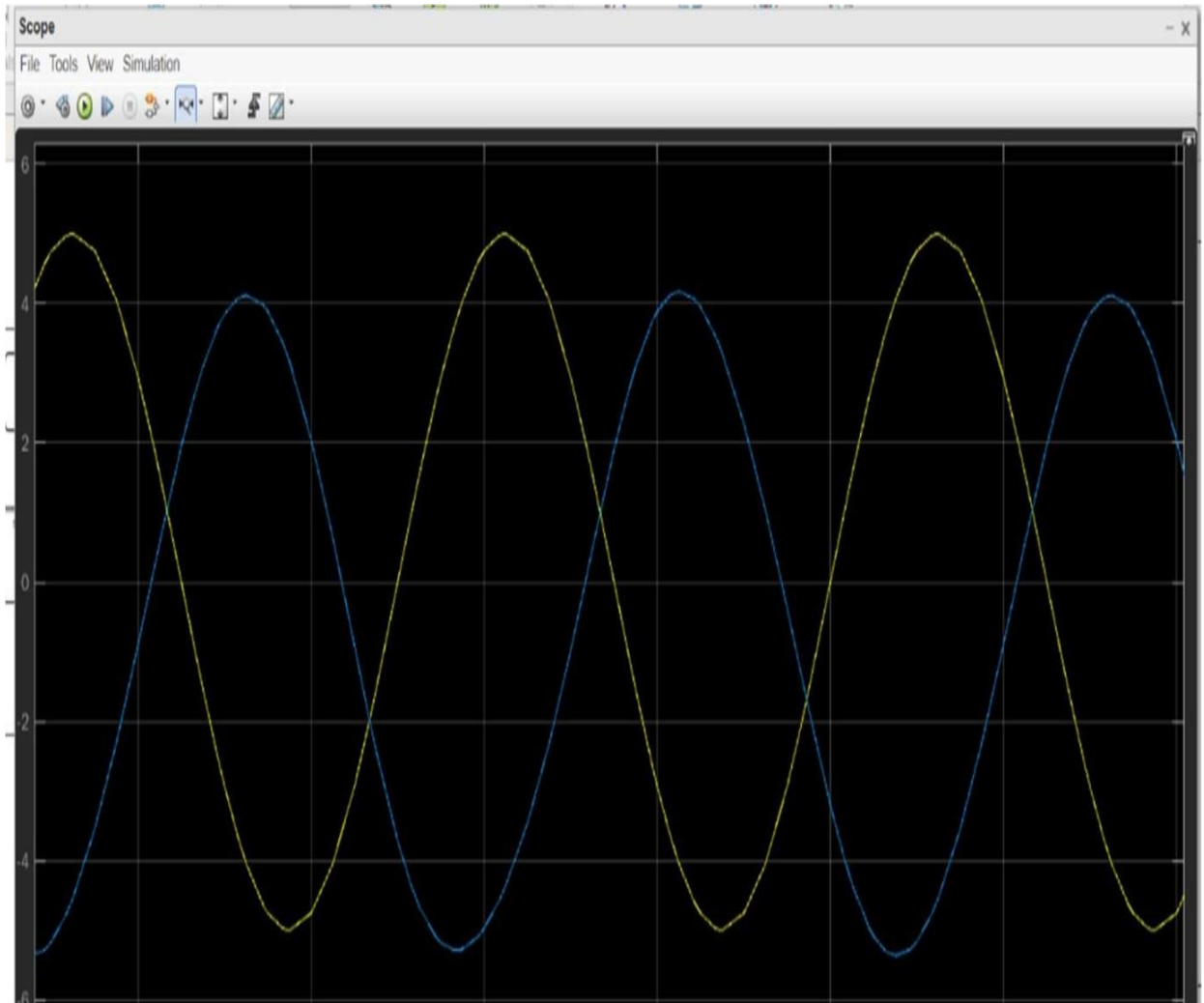


Fig.6 Demodulated Output Along With Input

❖ RESULT AND DISCUSSION:

- From a setup for PCM block diagram done in MATLAB Simulink Software, the first input, scope-2 is shown above. The result shows that the input analog signal is a sine wave pattern and pulse generator.
- The result display in Output scope-4 is as shown above. This signal is Pulse Amplitude Signal (PAM) which signals after the sampling and quantizing processes. Next, this input will go through the process of sampling that functionally converts the analog to digital signal by using the sample and hold which its initial condition is set to 0. The result was also formed with the quantizer at the end. The signal would go through a quantization process that functionally measures the numerical values of the samples and gives suitable scale.
- SCOPE-1 shows the Output 2 scope after multiplexing blocks. From the observation, this is due to usage of multiplexer as a device that has multiple inputs and shows in one output. The quantized PAM signal is converted to a serial binary code before transmission.
- FINAL OUTPUT IN THE SCOPE is the output of PCM Demodulation and Input Sine Wave. The Demodulated Output Is Obtained after passing through the PCM modulated output to BIT TO INTEGER CONVERTER & then through UNIFORM DECODER and finally passing it through the ANALOG FILTER DESIGN BLOCK.
- Based on the theoretical concept of PCM, it shows that the whole result of simulation MATLAB SIMULINK has achieved 100% accuracy.

❖ CONCLUSION:

The MATLAB SIMULINK of the PCM block diagram was studied and executed. We have successfully done a performance analysis of Pulse Code Modulation and the output has been depicted in the figures above. In this work, according to the basics of the PCM system, every block is implemented sequentially in MATLAB SIMULINK. Every function of PCM system is included in a single block of MATLAB SIMULINK, which is very helpful for the students to understand the whole PCM system.

EXPERIMENT NO: 6

Aim: To study BPSK Modulation and plot BER using MATLAB

Theory:

Digital modulation:

There are three basic types of modulation methods for the transmission of a digital signal. The methods are based on three attributes of a sinusoidal signal, amplitude, frequency and phase. The corresponding digital modulation methods are amplitude shift keying (ASK), frequency shift keying (FSK) and phase shift keying (PSK).

Amplitude shift keying (ASK): Amplitude shift keying (ASK) is the simple form of digital modulation. Digital input is unipolar NRZ signal. In ASK carrier amplitude is multiplied by high amplitude for binary “1” or by low amplitude for a binary “0”. However, when the low amplitude is 0 for binary “0” then the ASK is called On-Off keying or OOK which shown in Figure 1. In OOK the amplitude modulated carrier signal can be written as

$$v(t) = A \sin(2\pi f_c t) \dots \dots \dots (1)$$

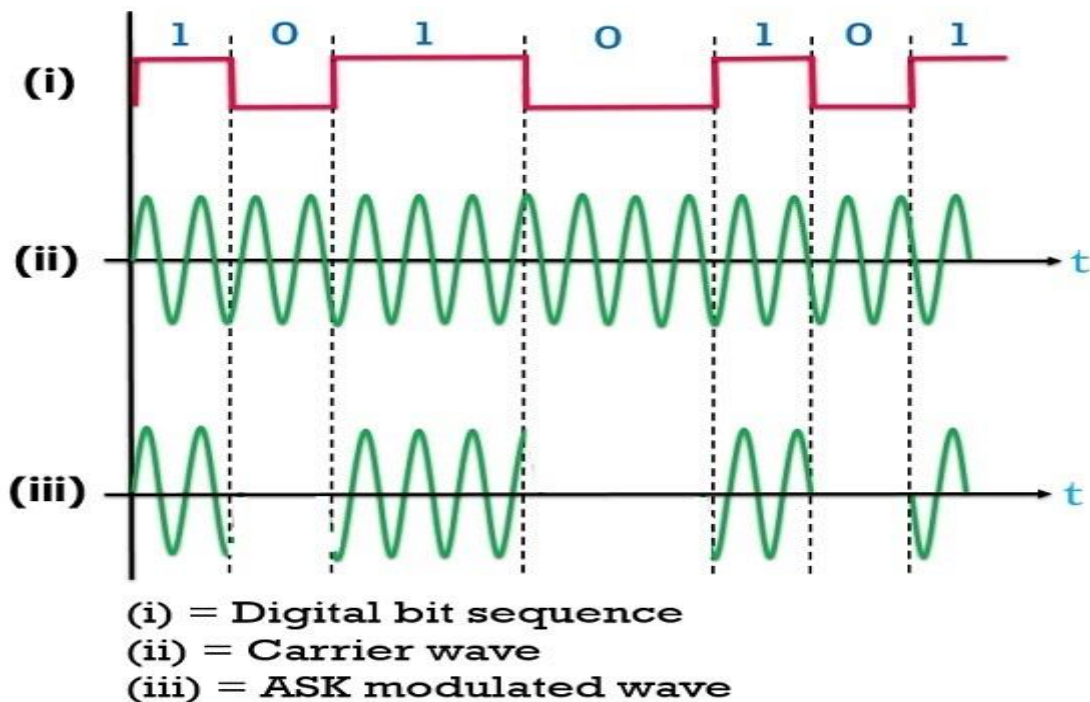


Fig.1

Frequency shift keying (FSK): In frequency shift keying (FSK), the frequency of the carrier is shifted between two discrete values, one representing binary “1” and representing binary “0” but the carrier amplitude does not changes. The simple form of FSK is BFSK. The instantaneous vale of the FSK signal is given by

$$v(t) = A \sin(2\pi f_1 t) + A \sin(2\pi f_2 t) \dots \dots \dots (2)$$

Where, f_1 and f_2 are the frequencies corresponding to binary “1” and “0” respectively and $f_1 > f_2$. From above equation, it is clear that the FSK signal can be considered to be comprising of two ASK signal with carrier frequencies f_1 and f_2 .

FSK Modulation and Demodulation

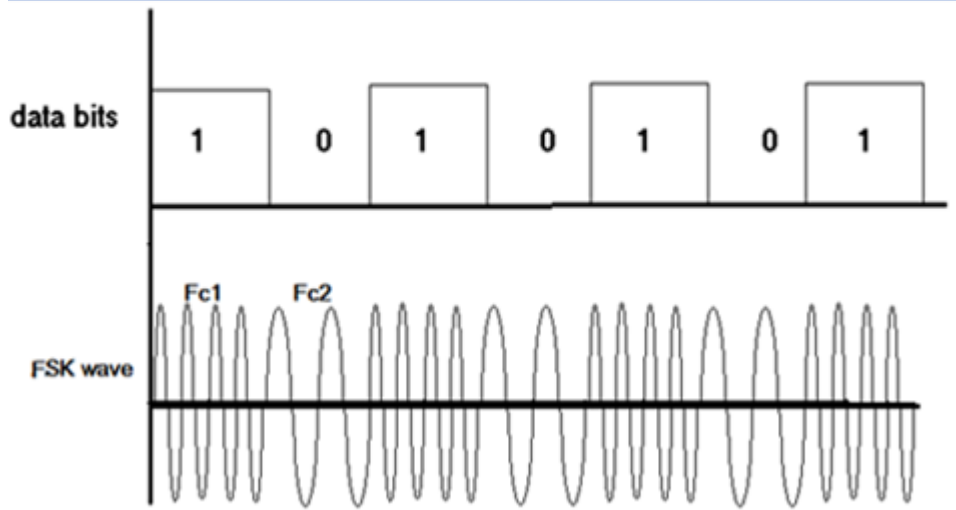


Fig.2

Phase shift keying (PSK):

In Phase shift keying (PSK), the phase of the carrier is modulated to represent the binary values. The carrier phase change between 0 and π by the bipolar digital signal. Binary states “1” and “0” are represented by the positive and negative polarity of the digital signal. The simplest form of PSK is BPSK is shown in Figure 3. The instantaneous value of the digital signal can be written as

$$v(t) = A \sin(2\pi f_c t)$$

Where, $A = \pm 1$; $A = 1$ for binary state “1” and $A = -1$ for binary state “0”.

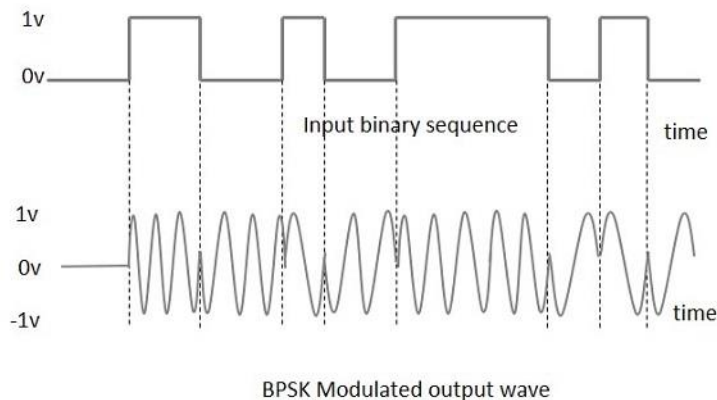


Fig.3

With Binary Phase Shift Keying (BPSK), the binary digits 1 and 0 maybe represented by the analog levels $+\sqrt{E_b}$ and $-\sqrt{E_b}$ respectively. The system model is as shown in the Figure below.

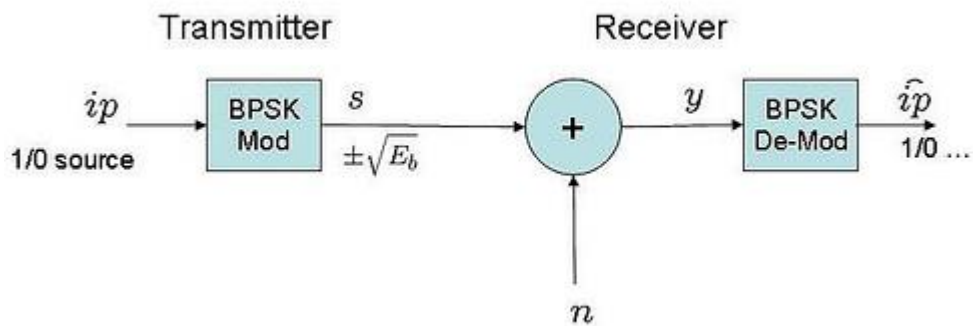


Fig.4: Simplified block diagram with BPSK transmitter-receiver

Channel Model:

The transmitted waveform gets corrupted by noise n , typically referred to as **Additive White Gaussian Noise (AWGN)**.

Additive : As the noise gets ‘added’ (and not multiplied) to the received signal

White : The spectrum of the noise is flat for all frequencies.

Gaussian : The values of the noise n follows the Gaussian probability distribution

function,
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$
 with $\mu = 0$ and $\sigma^2 = \frac{N_0}{2}$.

Computing the probability of error

The received signal,

$$y = s_1 + n \text{ when bit 1 is transmitted and}$$

$$y = s_0 + n \text{ when bit 0 is transmitted.}$$

The conditional probability distribution function (PDF) of y for the two cases are:

$$p(y|s_0) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(y+\sqrt{E_b})^2}{N_0}}$$

$$p(y|s_1) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(y-\sqrt{E_b})^2}{N_0}}$$

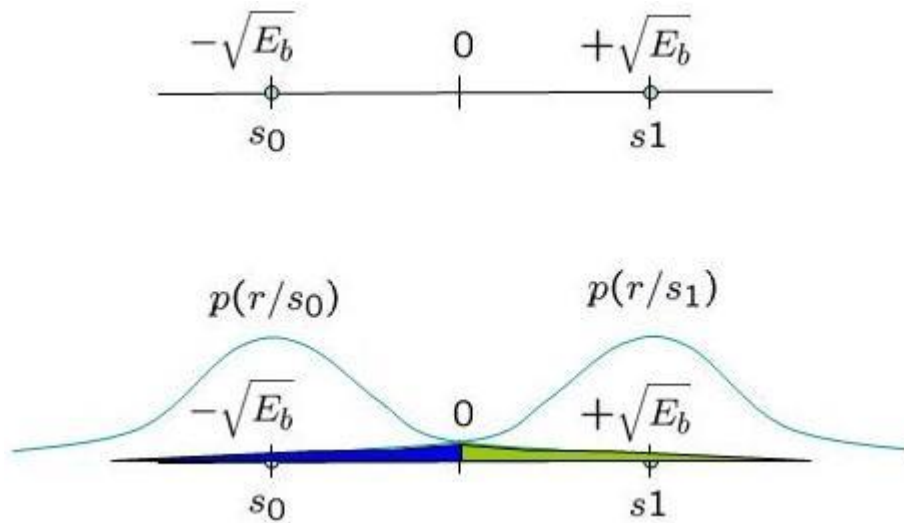


Figure: Conditional probability density function with BPSK modulation

Assuming that s_1 and s_0 are equally probable i.e. $p(s_1) = p(s_0) = 1/2$ the **threshold 0** forms the optimal decision boundary.

- if the received signal is y is greater than 0, then the receiver assumes s_1 was transmitted.
- if the received signal is y is less than or equal to 0, then the receiver assumes s_0 was transmitted.

i.e.

$$y > 0 \Rightarrow s_1 \text{ and}$$

$$y \leq 0 \Rightarrow s_0 .$$

Probability of error given s_1 was transmitted

With this threshold, the probability of error given s_1 is transmitted is (the area in blue region):

where,

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-x^2} dx$$

is the complementary error function.

Probability of error given s_0 was transmitted

Similarly the probability of error given s_0 is transmitted is (the area in green region):

Total probability of bit error

$$P_b = p(s_1)p(e|s_1) + p(s_0)p(e|s_0).$$

Given that we assumed that s_1 and s_0 are equally probable i.e. $p(s_1) = p(s_0) = 1/2$, the **bit error probability** is,

$$P_b = \frac{1}{2} \operatorname{erfc} \left(\sqrt{\frac{E_b}{N_0}} \right).$$

Simulation model:

Matlab code for computing the bit error rate with BPSK modulation from theory and simulation. The code performs the following:

- (a) Generation of random BPSK modulated symbols +1's and -1's.
- (b) Passing them through Additive White Gaussian Noise channel.
- (c) Demodulation of the received symbol based on the location in the constellation.
- (d) Counting the number of errors.
- (e) Repeating the same for multiple E_b/N_0 value.

Result:

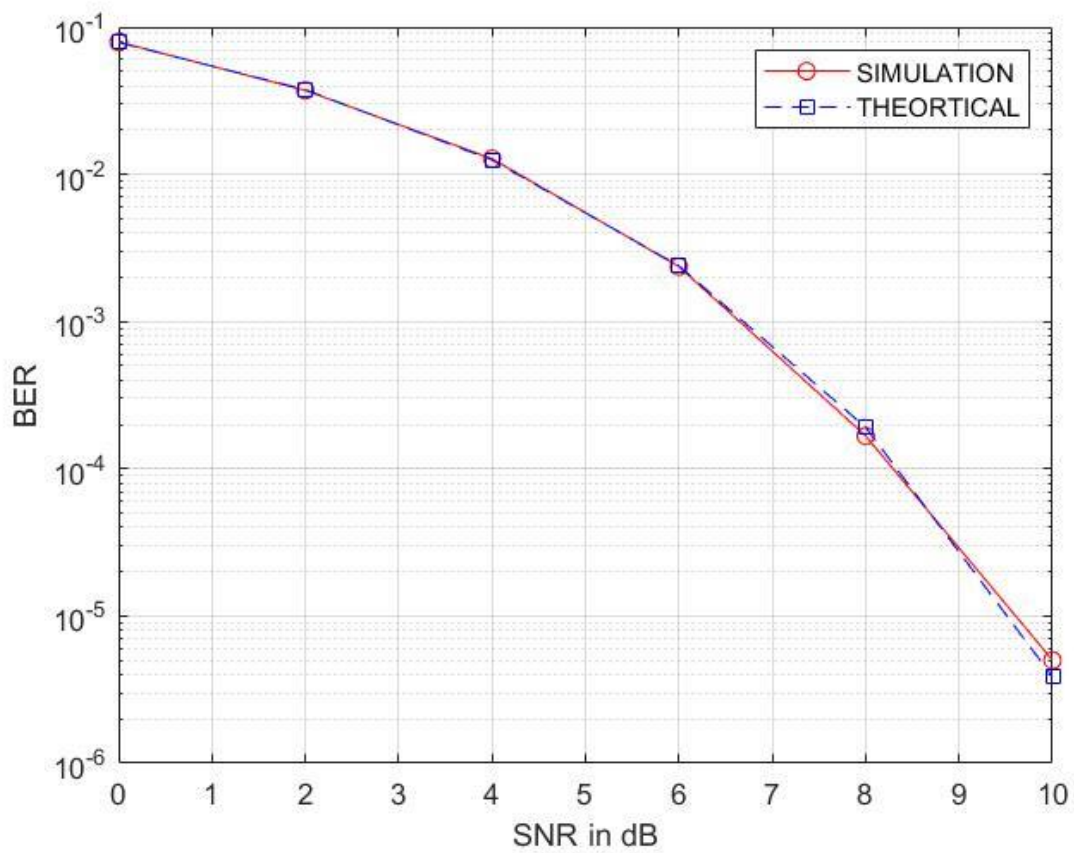


Fig.5

EXPERIMENT 7

AIM: Quadrature phase shift keying (QPSK) Modulation and Demodulation.

SOFTWARE: MATLAB R2021a

THEORY:

The Quadrature Phase Shift Keying (QPSK) is a variation of BPSK, and it is also a Double Side Band Suppressed Carrier (DSBSC) modulation scheme, which sends two bits of digital information at a time, called as digits.

Instead of the conversion of digital bits into a series of digital stream, it converts them into bit pairs. This decreases the data bit rate to half, which allows space for the other users.

QPSK Modulator:

The QPSK Modulator uses a bit-splitter, two multipliers with local oscillator, a 2-bit serial to parallel converter, and a summer circuit. Following is the block diagram for the same.

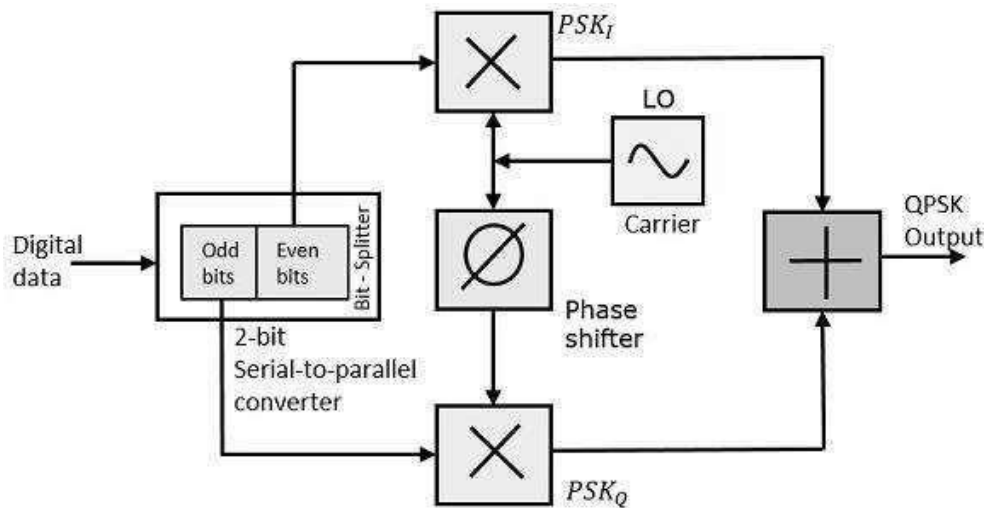


Figure : QPSK Modulator

At the modulator's input, the message signal's even bits (i.e., 2nd bit, 4th bit, 6th bit, etc.) and odd bits (i.e., 1st bit, 3rd bit, 5th bit, etc.) are separated by the bits splitter and are multiplied with the same carrier to generate odd BPSK (called as PSKI) and even BPSK (called as PSKQ). The PSKQ signal is anyhow phase shifted by 90° before being modulated.

QPSK Demodulator

The QPSK Demodulator uses two product demodulator circuits with local oscillator, two band pass filters, two integrator circuits, and a 2-bit parallel to serial converter. Following is the diagram for the same.

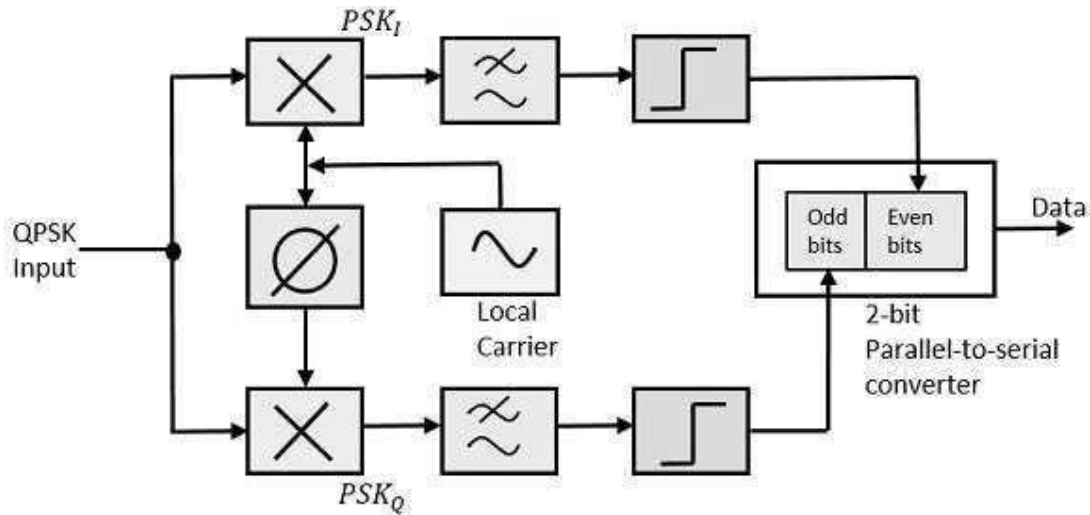


Figure : QPSK Demodulator

The two product detectors at the input of demodulator simultaneously demodulate the two BPSK signals. The pair of bits are recovered here from the original data. These signals after processing, are passed to the parallel to serial converter.

MATLAB Code:

```

clc;
clear all;
close all;

n=input('Enter size of message in bits ');
disp('Bits should be in 0 or 1');
for i=1:n
    ml(i)= input('enter bit');
end
disp(ml);

m2=[ ;

%NRZ coding
for i=1:n
    if ml(i)==1
        m2=[m2 ones(1,1)];
    else
        m2=[m2 -ones(1,1)];
    end
end
for i=1:n
    if ml(i)==1
        m3=[m3 ones(1,100)];
    else
        m3=[m3 -ones(1,100)];
    end
end
figure(1)
subplot(211)

```

```

stem(m3);
grid on
title('NRZ Coded input Signal','fontsize',14);
xlabel('Sample Numbers');
ylabel('Amplitude');

%Demultiplexing into odd and even bits
n_odd=0;
i=1;
for i=1:800
    xodd(i)=m2(n_odd+1);
    xeven(i)=m2(n_odd+2);
    if i==j*200
        n_odd=n_odd+2;
        j=j+1;
    end
end
end

%carrier
fs=500;
ts=1/fs;
fc=10;
t=0:ts:0.2*n-ts;
c1=xeven.*sin(2*pi*fc*t);
c2=xodd.*cos(2*pi*fc*t);
qpsk=c1+c2;
subplot(212);
plot(t,qpsk);
grid on;
title('QPSK Modulated signal', 'fontsize',14);
xlabel('Time [Sec]');
ylabel('Amplitude');

figure(2)
subplot(411);
stem(xodd);
title('NRZ Coded odd Signal','fontsize',14);
xlabel('Sample Numbers');
ylabel('Amplitude');
subplot(412);
plot(t,c2);

subplot(413);
stem(xeven);
title('NRZ Coded even Signal','fontsize',14);
xlabel('Sample Numbers');
ylabel('Amplitude');
grid on;
subplot(414);
plot(t,c1);

%Demodulation
reven=qpsk.*sin(2*pi*fc*t);
rodd=qpsk.*cos(2*pi*fc*t);

receven=[];
re_codd=[1:
for i=1:200:800;
    receven=[receven trapz(t(i:i+199),reven(i:i+199)) ]:
    recodd=[recodd trapz(t(i:i+199),rodd(i:i+199))]:

```

```

end
disp(receven)
disp(recodd)

rec=[];
for j=1:4;
    rec=[rec recodd(j) receven(j)];
end
disp(rec)

recbits=[]
for k=1:8;
    if rec(k)>0
        recbits=[recbits 1];
    else
        recbits=[recbits 0];
    end
end
disp(recbits)

figure(3)
stem(recbits);
title('Demodulated output Binary signal');
xlabel('Bits');
ylabel('Amplitude');

```

RESULT:

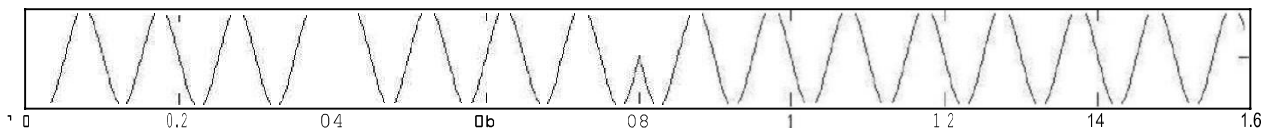
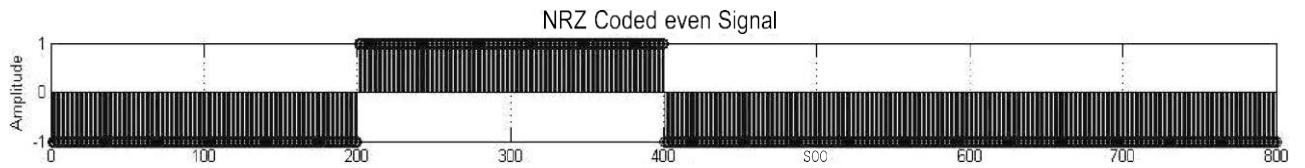
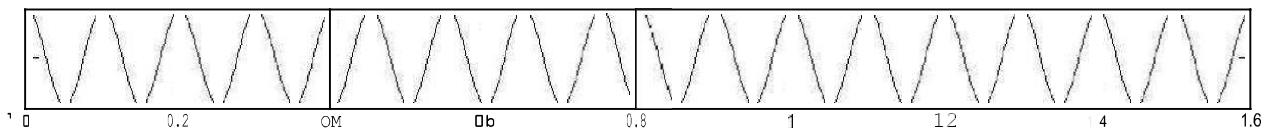
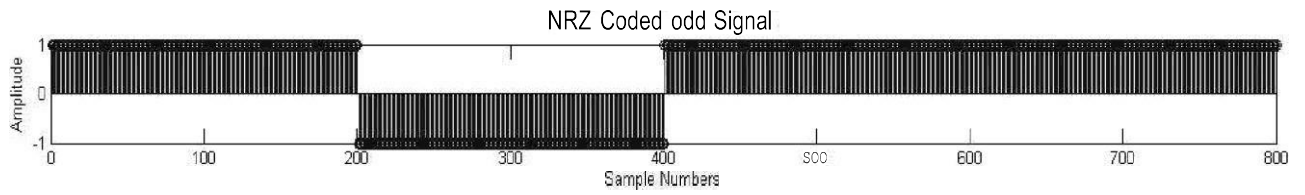
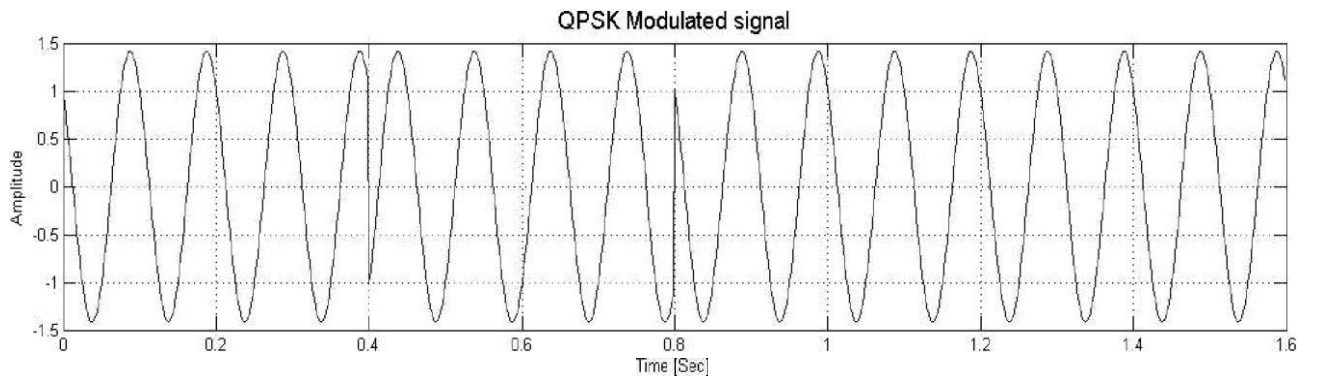
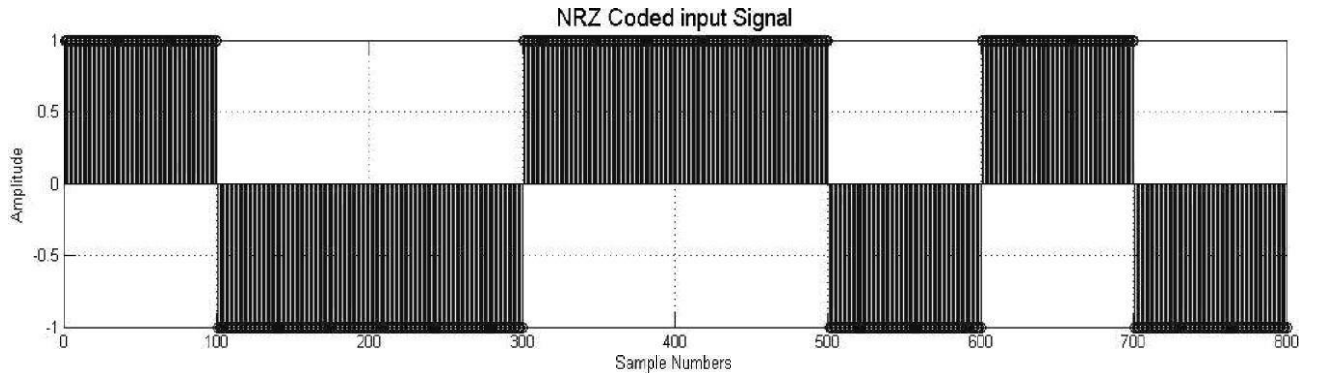
The screenshot shows a MATLAB Command Window with the following text:

```

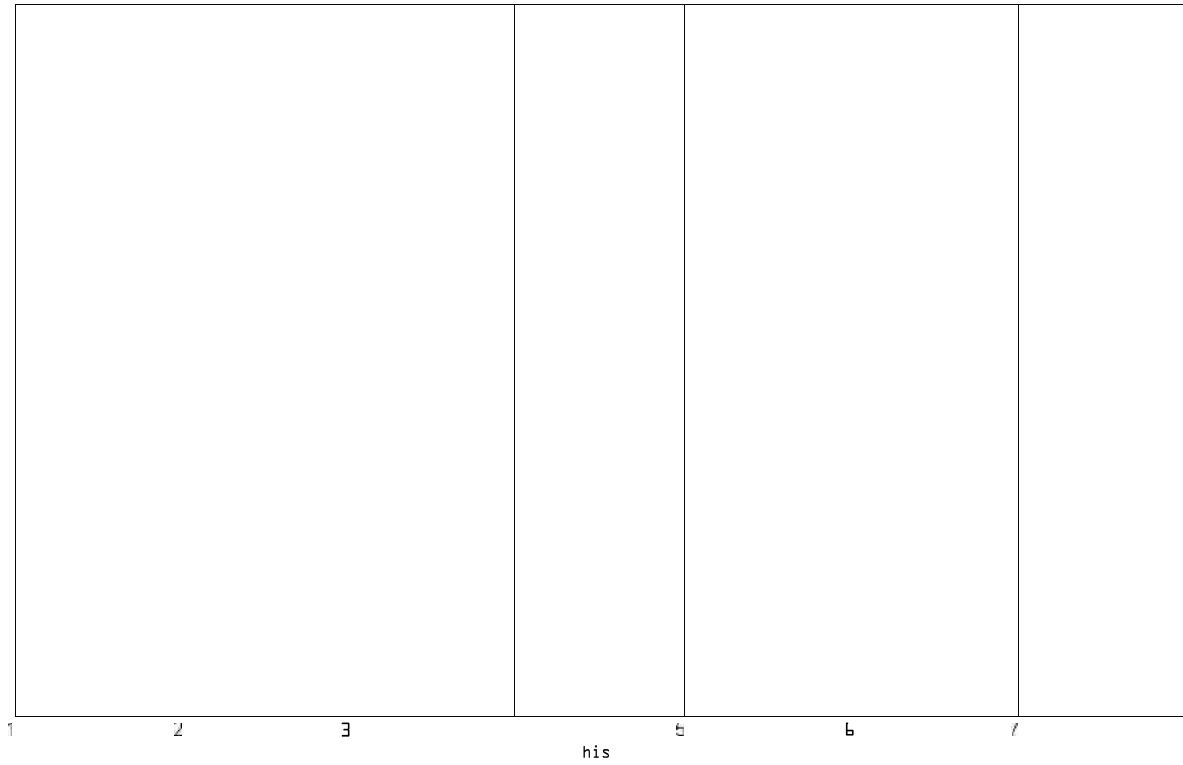
New to MATLAB? Watch this Video, see Demos, or read Getting started.
Enter size of message in bits 5
bits stored in 'J' or 'i'
enter also
enter rit')
e-ter kitG
enter Citi
enter also
enter hit0
e-ter kiti
enter bitf

recbits =

```



Demodulated output Binary signal



CONCLUSION:

Hence QPSK Modulation and Demodulation has been studied. Our message Binary data was successfully recovered at the receiver end.

EXPERIMENT- 8

1. **AIM :** To study the Digital Signal transmission using Quadrature Amplitude Modulation (QAM) using MATLAB Tool
2. **LEARNING OUTCOME:** After the competing this experiment we can learn that
 - ❖ How does change the Amplitude and phase of a higher frequency signal (carrier signal) with 16-symbols baseband binary polar pulse sequence (Digital Message signals) i.e. study of 16-QAM.
 - ❖ The effects of the channel Gaussian noise on the modulated signal i.e. study of the SNR of QAM.

3. THEORY:

The Quadrature Amplitude Modulation (QAM) is mainly used to increase the transmission rate and decrease the transmission bandwidth. This modulation scheme uses in the both analog and digital version of the communication. Digital signal transmission using QAM has same concept of the analog version of the QAM .But basic difference in the base band signal, the message signals are two digital bit streams, (binary polar pulse sequence) instead of the two analog message signals in this case. These two base band signals are modulated ($m_1(t)$ and $m_2(t)$) by a carrier of the same frequency ($\cos(\omega_c t)$) but in quadrature phase ($\sin(\omega_c t)$). A basic block diagram of QAM Modulator and demodulator are shown in Fig.1.(a).

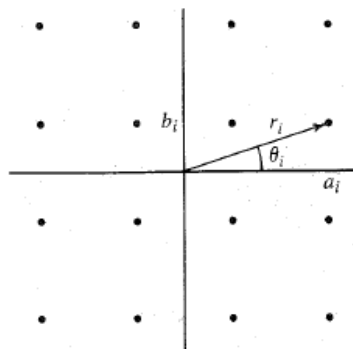
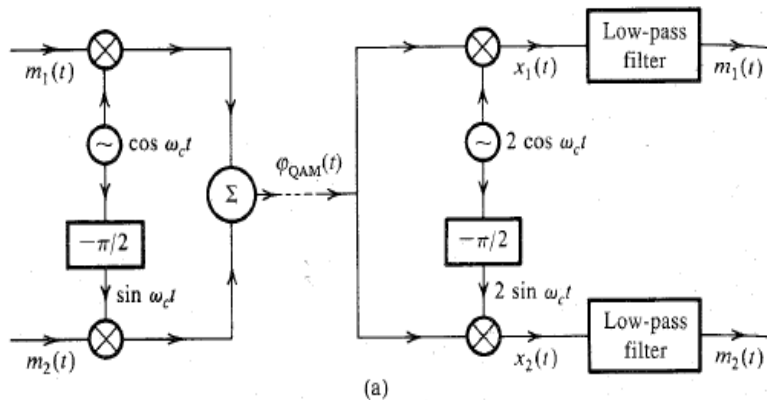


Fig.1. a) QAM Modulator and demodulator b) 16-QAM

A Mathematic representation of the digital QAM is given by Eq.(1)

$$p_i(t) = a_i p(t) \cos(w_c t) + b_i p(t) \sin(w_c t) \tag{1}$$

$$= r_i p(t) \cos(w_c t - \theta_i) \quad i = 1, 2, \dots, M$$

Where $r_i = \sqrt{a_i^2 + b_i^2}$ and $\theta_c = \tan^{-1} \frac{b_i}{a_i}$ $m_1(t) = a_i p(t)$ and $m_2(t) = b_i p(t)$

$p(t)$ is properly shaped baseband pulse . The signal $p_i(t)$ can be generated using QAM . One possible choice of r_i and θ_i for 16 pulses is shown in Fig.1.(b). The transmitting pulse $p_i(t)$ can take on 16 distinct forms and is, therefore , a 16-ary pulse. Since $M=16$, each pulse can transmit the information on 4 binary digits. This can be done as follows: there are 16 possible sequences of four binary digits and there are 16 combinations (a_i, b_i) in Fig.1.(b). Thus, every possible 4-bit sequence is transmitted by a particular (a_i, b_i) or (r_i, θ_i) . Therefore. One signal pulse $r_i p(t) \cos(w_c t - \theta_i)$ transmits 4 bits. The bit rate quadrated without increasing the bandwidth. The transmission rate can be increased further by increasing the value of M .

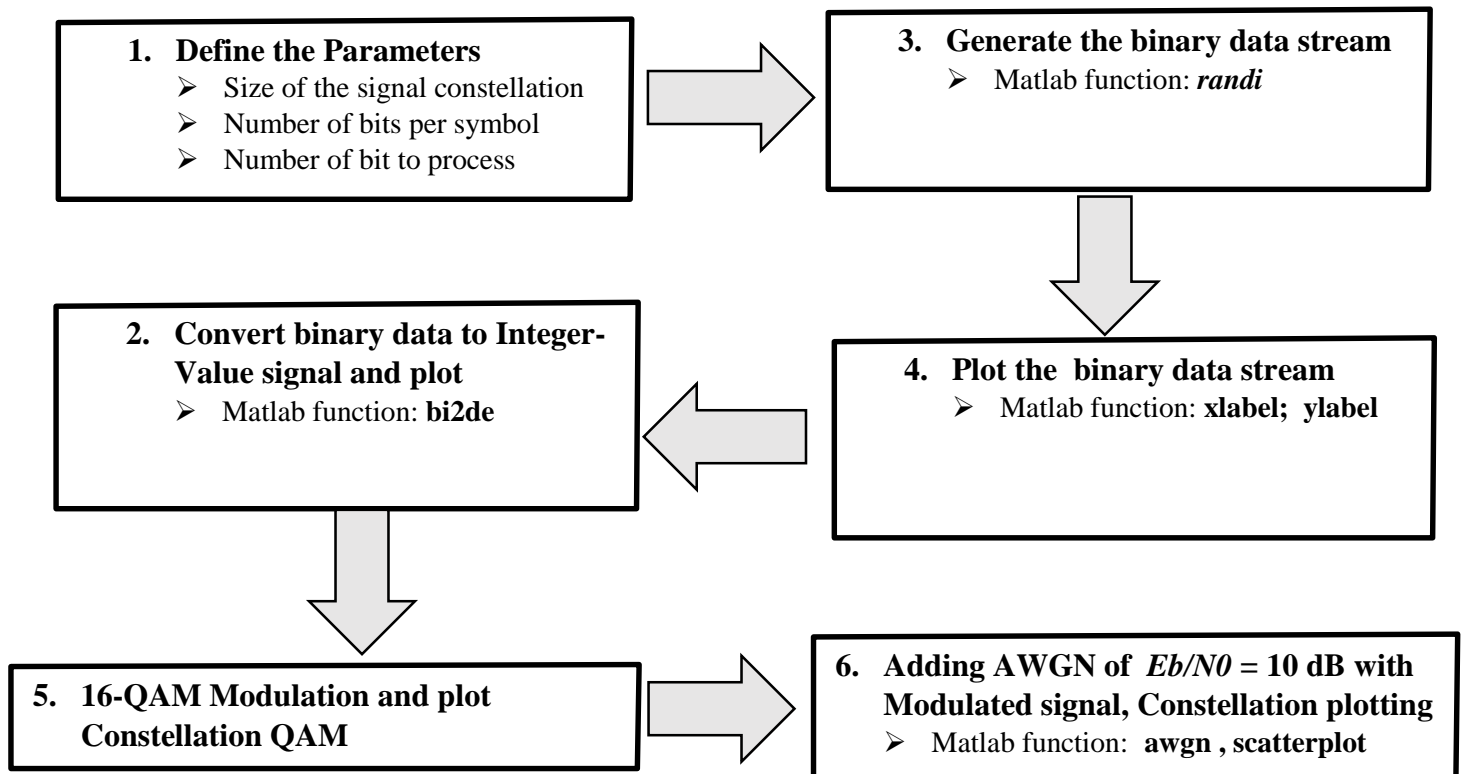
After modulation, this modulated signal will be transmitted through a channel which having the AWGN noise with energy to noise power spectrum density $E/N = 10$ dB.

4. Apparatus and Component required:

❖ **MATLAB TOOL**

5. Experiment Step/Simulation Flow Chart:

❖ **MATLAB FLOW CHART**



6. Results and observation:

- ❖ Observe the generated binary data stream and compare this with its equivalent integers signal.
- ❖ Observe Constellation plots of the modulated 16-QAM with and without AWGN channel noise, and compare results of the both cases.
- ❖ Comments on the SNR of the modulated signal at case of the AWGN channel noise.

7. Questions:

- ❖ What are basic differences between the digital QAM and Analog QAM?
- ❖ Why it is called the Quadrature PSK (QPSK)?
- ❖ Write the some advantages of QAM over the other modulation schemes.
- ❖ How does it increase the transmission rate and increase the bandwidth efficiency with increase of the bit in symbol of signal?

EXPERIMENT 9

Aim: Generate the line codes for a 10-bit dataset in MATLAB simulation environment.

1. **Learning Outcome:** After the completing this experiment we becomes familiar to generate line codes of a binary dataset in digital communication.
2. **Theory:** A computer network is used for communication of data from one station to another station in the network. We have seen that analog or digital data traverses through a communication media in the form of a signal from the source to the destination. The channel bridging the transmitter and the receiver may be a guided transmission medium such as a wire or a wave-guide or it can be an unguided atmospheric or space channel. But, irrespective of the medium, the signal traversing the channel becomes attenuated and distorted with increasing distance. Hence a process is adopted to match the properties of the transmitted signal to the channel characteristics so as to efficiently communicate over the transmission media. There are two alternatives; the data can be either converted to digital or analog signal. Both the approaches have pros and cons. What to be used depends on the situation and the available bandwidth.

Now, either form of data can be encoded into either form of signal. For digital signaling, the data source can be either analog or digital, which is encoded into digital signal, using different encoding techniques. The basis of analog signaling is a constant frequency signal known as a *carrier signal*, which is chosen to be compatible with the transmission media being used, so that it can traverse a long distance with minimum of attenuation and distortion. Data can be transmitted using these carrier signals by a process called *modulation*, where one or more fundamental parameters of the carrier wave, i.e. amplitude, frequency and phase are being modulated by the source data. The resulting signal, called *modulated signal* traverses the media, which is *demodulated* at the receiving end and the original signal is extracted. A line code is a specific code (with precisely defined parameters) used for transmitting a digital signal over a channel. Line coding is used in digital data transport – the pattern of voltage, current used to represent digital data on a transmission link is called line encoding.

3. Linear coding characteristics:

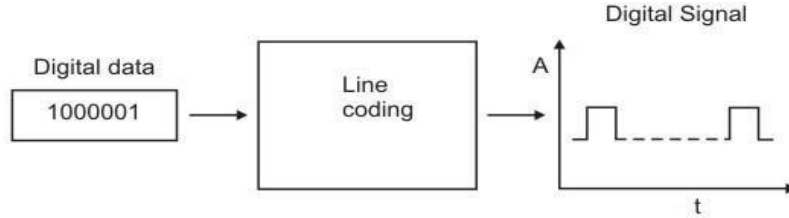


Fig.1 Line coding to convert digital data to digital signal

Important parameters those characteristics line coding techniques are mentioned below.

No of signal levels: This refers to the number values allowed in a signal, known as **signal levels**, to represent data. Figure 2 (a) shows two signal levels, whereas Fig. 2 (b) shows three signal levels to represent binary data.

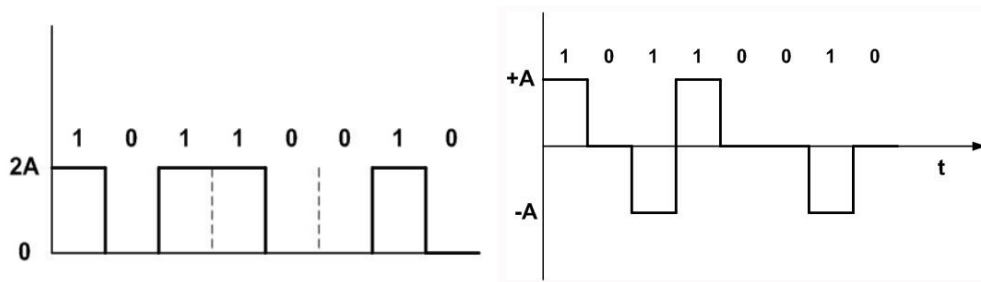


Fig. 2 (a) Signal with two voltage levels, (b) Signal with three voltage levels

Bit rate versus Baud rate: The **bit rate** represents the number of bits sent per second, whereas the **baud rate** defines the number of signal elements per second in the signal. Depending on the encoding technique used, baud rate may be more than or less than the data rate.

DC components: After line coding, the signal may have zero frequency component in the spectrum of the signal, which is known as the direct-current (**DC**) **component**. DC component in a signal is not desirable because the DC component does not pass through some components of a communication system such as a transformer. This leads to distortion of the signal and may create error at the output. The DC component also results in unwanted energy loss on the line.

Signal Spectrum: Different encoding of data leads to different spectrum of the signal. It is necessary to use suitable encoding technique to match with the medium so that the signal suffers minimum attenuation and distortion as it is transmitted through a medium.

Synchronization: To interpret the received signal correctly, the bit interval of the receiver should be

exactly same or within certain limit of that of the transmitter. Any mismatch between the two may lead wrong interpretation of the received signal. Usually, clock is generated and synchronized from the received signal with the help of a special hardware known as Phase Lock Loop (PLL). However, this can be achieved if the received signal is self-synchronizing having frequent transitions (preferably, a minimum of one transition per bit interval) in the signal.

Cost of Implementation: It is desirable to keep the encoding technique simple enough such that it does not incur high cost of implementation.

4. Line Coding Techniques: Line coding techniques can be broadly divided into three broad categories: Unipolar, Polar and Bipolar as shown in fig. 3.

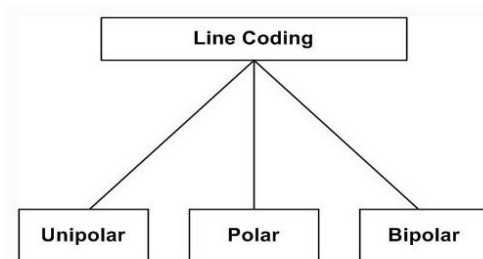


Fig. 3 Three basic categories of line coding techniques

4.1 Unipolar: In unipolar encoding technique, only two voltage levels are used. It uses only one polarity of voltage level as shown in Fig. 4. In this encoding approach, the bit rate same as data rate. Unfortunately, DC component present in the encoded signal and there is loss of synchronization for long sequences of 0's and 1's. It is simple but obsolete.

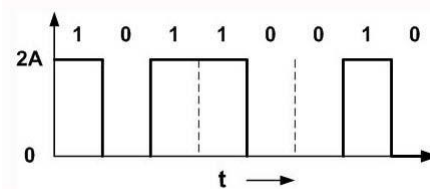


Fig. 4 Unipolar encoding with two voltage levels

There are two unipolar line coding schemes- i) Unipolar RZ, ii) Unipolar NRZ (ON-OFF Keying).

5.1. i) Unipolar RZ: In unipolar RZ, the waveform has zero value when '0' is transmitted and waveform has 'A' volts when '1' is transmitted. In RZ form, the 'A' volts is presented for $\frac{T_b}{2}$,

period if symbol '1' transmitted and remaining $\frac{T_b}{2}$, waveform returns to zero value, i.e., for unipolar RZ form.

If symbol '1' is transmitted, then we have

$$x(t) = \begin{cases} A & \text{for } 0 \leq t < T_b/2 \\ 0 & \text{for } T_b/2 \leq t < T_b \end{cases} \quad (1)$$

If symbol '0' is transmitted, then

$$x(t) = 0 \text{ for } 0 \leq t < T_b \quad (2)$$

If the input bit pattern is 1101000011, then the unipolar RZ waveform will be

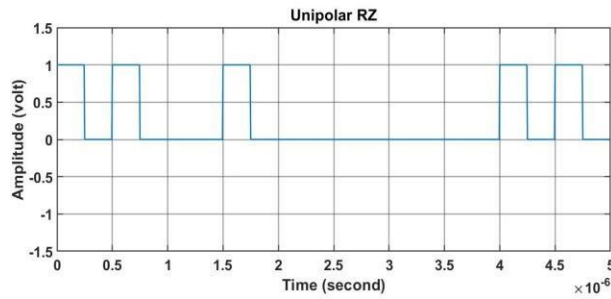


Fig. 5 Unipolar RZ

ii) **Unipolar NRZ (ON-OFF keying):** In unipolar NRZ format, when symbol '1' is to be transmitted, the symbol has 'A' volts for full duration. When symbol '0' is to be transmitted, the signal has zero volts for complete symbol duration.

If symbol '1' is transmitted

$$(t) = A \text{ for } 0 \leq t < T_b \quad (3)$$

If symbol '0' is transmitted

$$(t) = 0 \text{ for } 0 \leq t < T_b \quad (4)$$

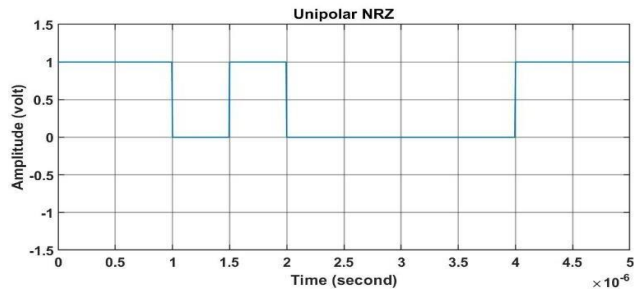


Fig. 6 Unipolar NRZ

5.2. Polar: Polar encoding technique uses two voltage levels – one positive and the other one negative. Four different encoding schemes shown in Fig. 2.4.6 under this category discussed below:

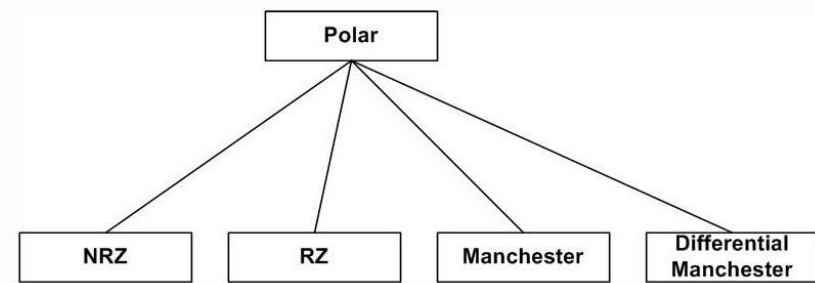


Fig. 5 Encoding Schemes under polar category

5.2. i) Polar Return to Zero RZ: To ensure synchronization, there must be a signal transition in each bit. Key characteristics of the RZ coding are: Three levels, Bit rate is double than that of data rate, No dc component, Good synchronization, and Main limitation is the increase in bandwidth.

In polar RZ, symbol '1' is represented by positive voltage polarity whereas symbol '0' is represented by negative voltage polarity. For RZ format pulse is transmitted only for half duration.

If '1' is transmitted, then

$$x(t) = \begin{cases} +A/2 & \text{for } 0 \leq t < T_b/2 \\ 0 & \text{for } \frac{T_b}{2} \leq t < T \end{cases} \quad (5)$$

If '0' is transmitted, then

$$x(t) = \begin{cases} -A/2 & \text{for } 0 \leq t < T_b/2 \\ 0 & \text{for } \frac{T_b}{2} \leq t < T \end{cases} \quad (6)$$

2

b

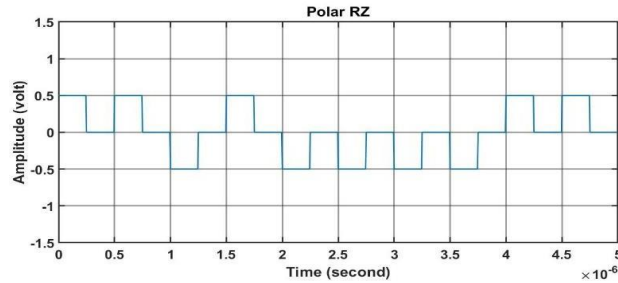


Fig. 7 Polar RZ

ii) Polar Non Return to zero (NRZ): The most common and easiest way to transmit digital signals is to use two different voltage levels for the two binary digits. Usually a negative voltage is used to represent one binary value and a positive voltage to represent the other. The data is encoded as the presence or absence of a signal transition at the beginning of the bit time. As shown in the figure below, in NRZ encoding, the signal level remains same throughout the bit-period. There are two encoding schemes in NRZ: NRZ-L and NRZ-I.

ii) a) NRZ-L: For polar NRZ format, symbol '1' is represented by negative polarity and symbol '0' is represented by positive polarity.

If symbol '1' is transmitted, then

$$(t) = A/2 \text{ for } 0 \leq t < T_b \quad (7)$$

If symbol '0' is transmitted, then

$$(t) = -A/2 \text{ for } 0 \leq t < T_b \quad (8)$$

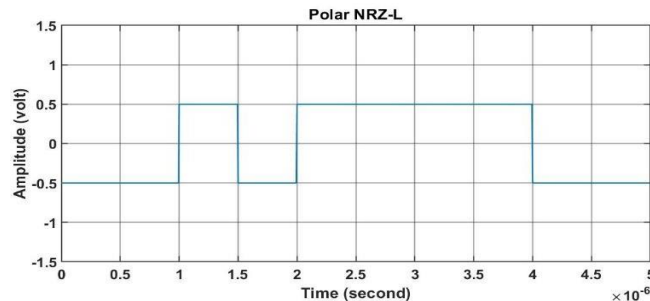


Fig. 8 Polar NRZ-L

b) NRZ-I: For polar NRZ format, symbol '0' is represented by no transition state and symbol '1' is represented by transition state.

If symbol '1' is transmitted, then

$$(t) = \text{no transition for } 0 \leq t < T_b \quad (7)$$

If symbol '0' is transmitted, then

$$(t) = \text{transition for } 0 \leq t < T_b \quad (8)$$

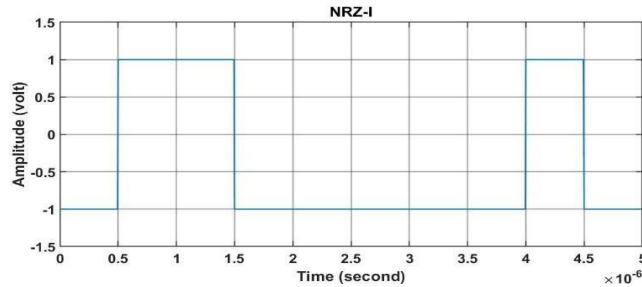


Fig. 9 Polar NRZ-L

The **advantages** of NRZ coding are:

Detecting a transition in presence of noise is more reliable than to compare a value to a threshold. NRZ codes are easy to engineer and it makes efficient use of bandwidth.

Biphase: To overcome the limitations of NRZ encoding, biphase encoding techniques can be adopted. *Manchester* and *differential Manchester Coding* are the two common Biphase techniques in use, as shown in Fig. 8. In Manchester coding the mid-bit transition serves as a clocking mechanism and also as data.

ii) Manchester Code: In the standard Manchester coding there is a transition at the middle of each bit period. A binary '0' corresponds to a *low-to-high transition* and a binary '1' to a *high-to-low transition* in the middle.

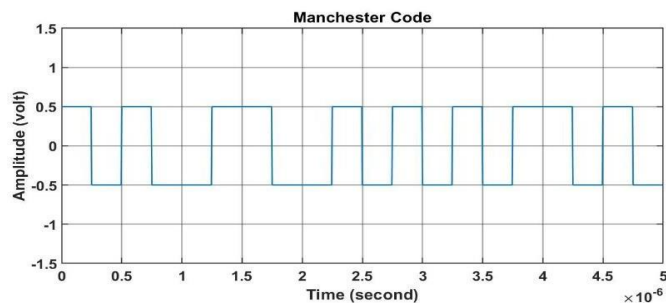


Fig. 10 Manchester Code

iii) Differential Manchester code: In Differential Manchester, inversion in the middle of each bit is used for synchronization. The encoding of a '0' is represented by the presence of a transition both at the beginning and at the middle and '1' is represented by a transition only in the middle of the bit period.

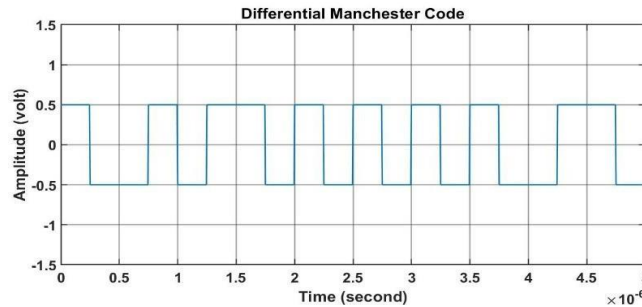


Fig. 11 Differential Manchester Code

Key characteristics are: Two levels, No DC component, Good synchronization, higher bandwidth due to doubling of bit rate with respect to data rate. The bandwidth required for biphasic techniques are greater than that of NRZ techniques, but due to the predictable transition during each bit time, the receiver can synchronize properly on that transition.

5.3 Bipolar NRZ: Bipolar encoding technique uses Alternate Mark Inversion Technique (AMI). Unlike RZ 0-level is used to represent a '0' and a binary 1's are represented by alternating positive and negative voltages, as shown in Fig.12.

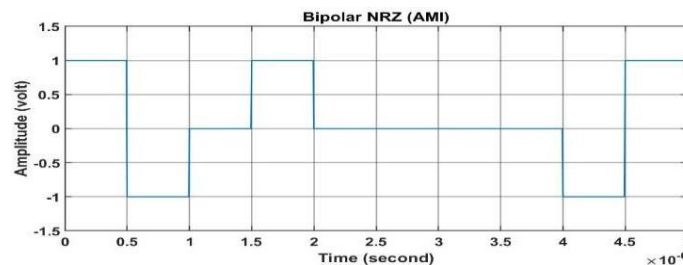


Fig. 12 Bipolar (AMI) NRZ

5. Apparatus and Software: Computer with MATLAB software environment.

6. Results and observation: Generate waveforms for line codes.

Take amplitude $A=1$; Samples/bit (T_b) = 1000; bit width = clock period and

Input Bit Pattern: [1 1 0 1 0 0 0 0 1 1]

7. Questions:

- I. Why do you need encoding of data before sending over a medium?
- II. What are the four possible encoding techniques? Give examples.
- III. Between RZ and NRZ encoding techniques, which requires higher bandwidth and why?
- IV. How does Manchester encoding differ from differential Manchester encoding?
- V. How Manchester encoding helps in achieving better synchronization?
- VI. Distinguish between PAM and PCM signals?
- VII. What is quantization error? How can it be reduced?

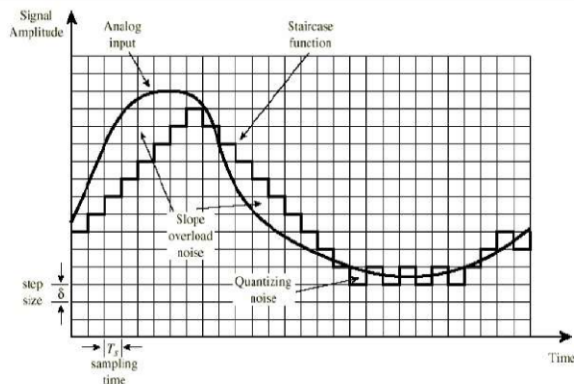
EXPERIMENT 10

Aim:- To analyse the Delta modulation and Demodulation using Multisim

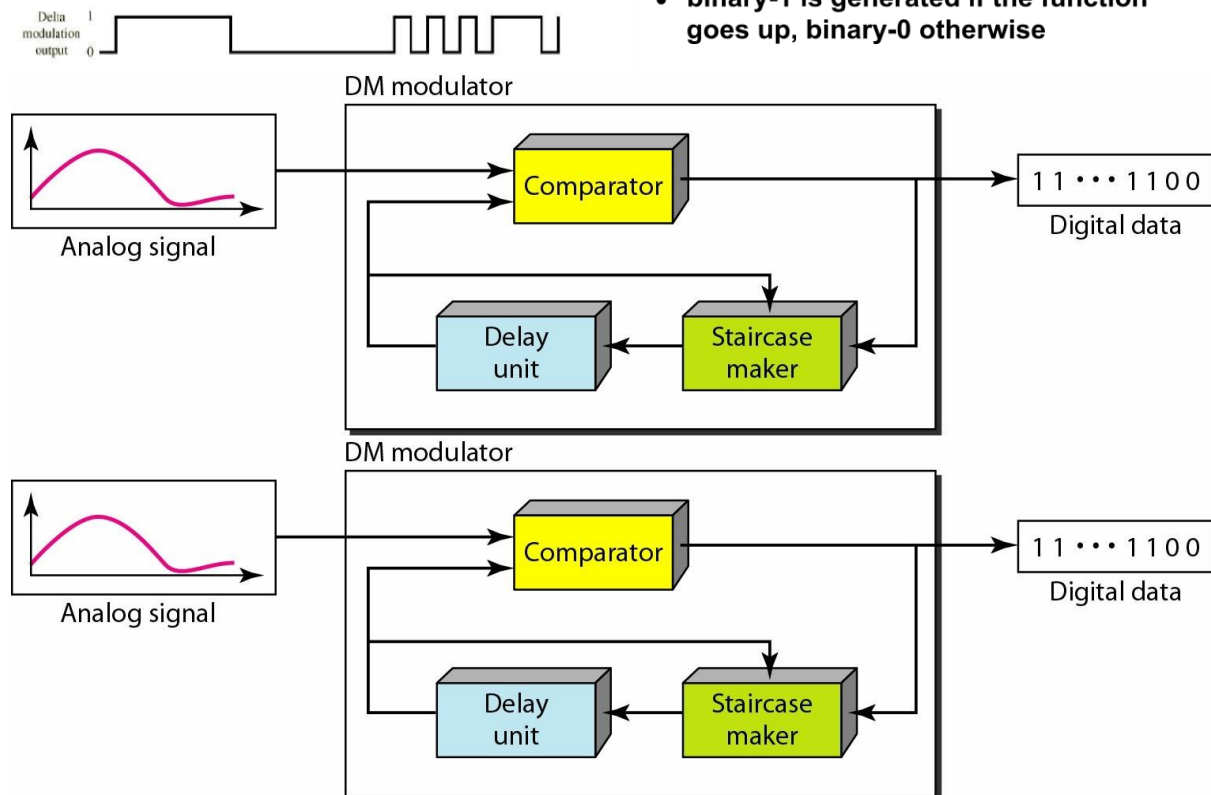
Theory

Delta-Modulation – most popular alternative to PCM

- analog signal is approximated by staircase function
- **only a single binary digit is required for each sample !!!**

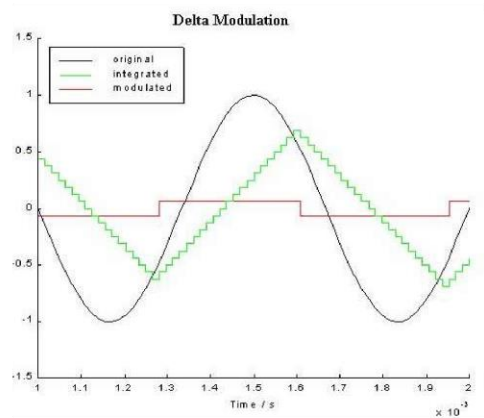


- at each sampling time (kT), the function moves up or down a constant amount δ (step size) – the staircase function attempts to track the original waveform as closely as possible
- at each sampling time, the analog input is compared to the most recent value of the approximating staircase function
- binary-1 is generated if the function goes up, binary-0 otherwise

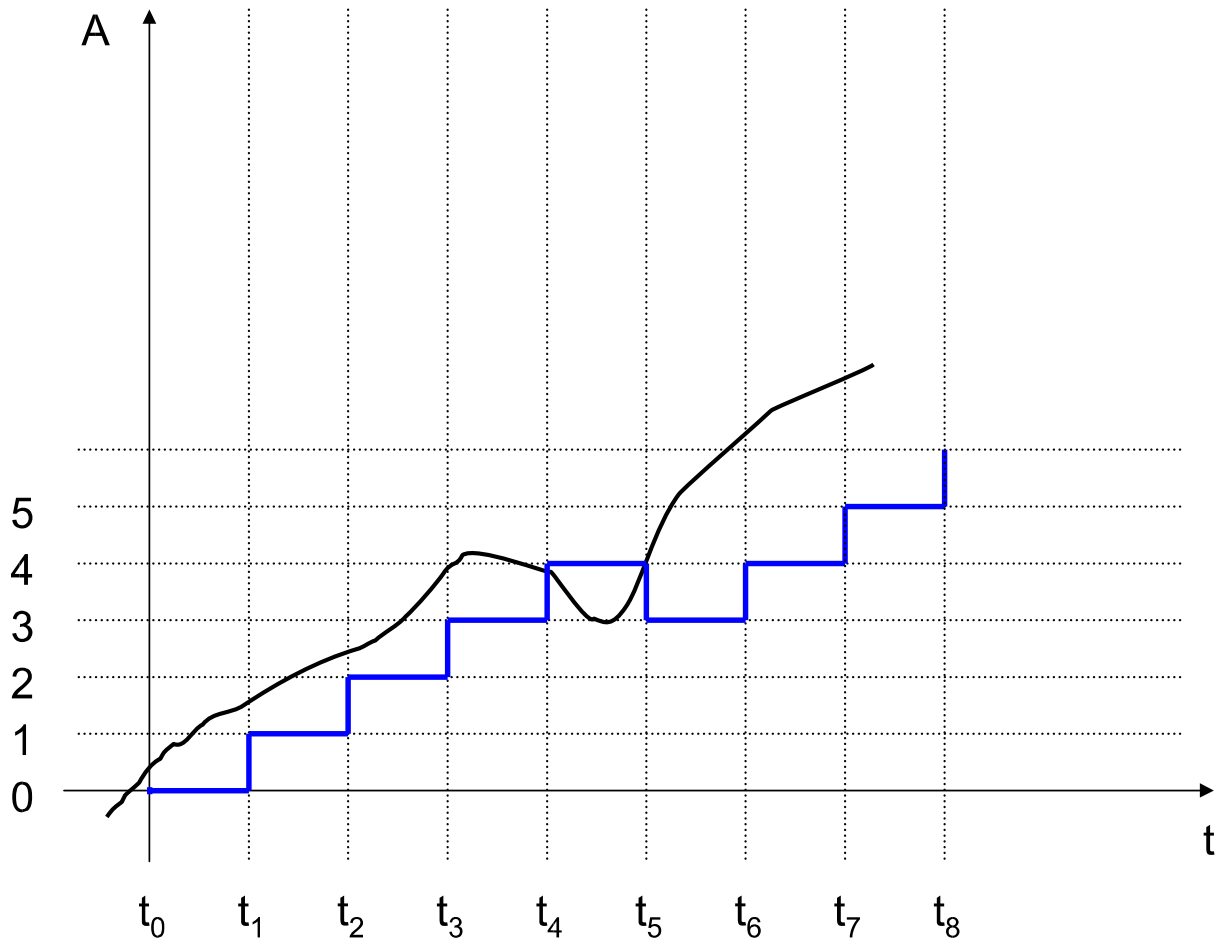


Delta Modulation Parameters

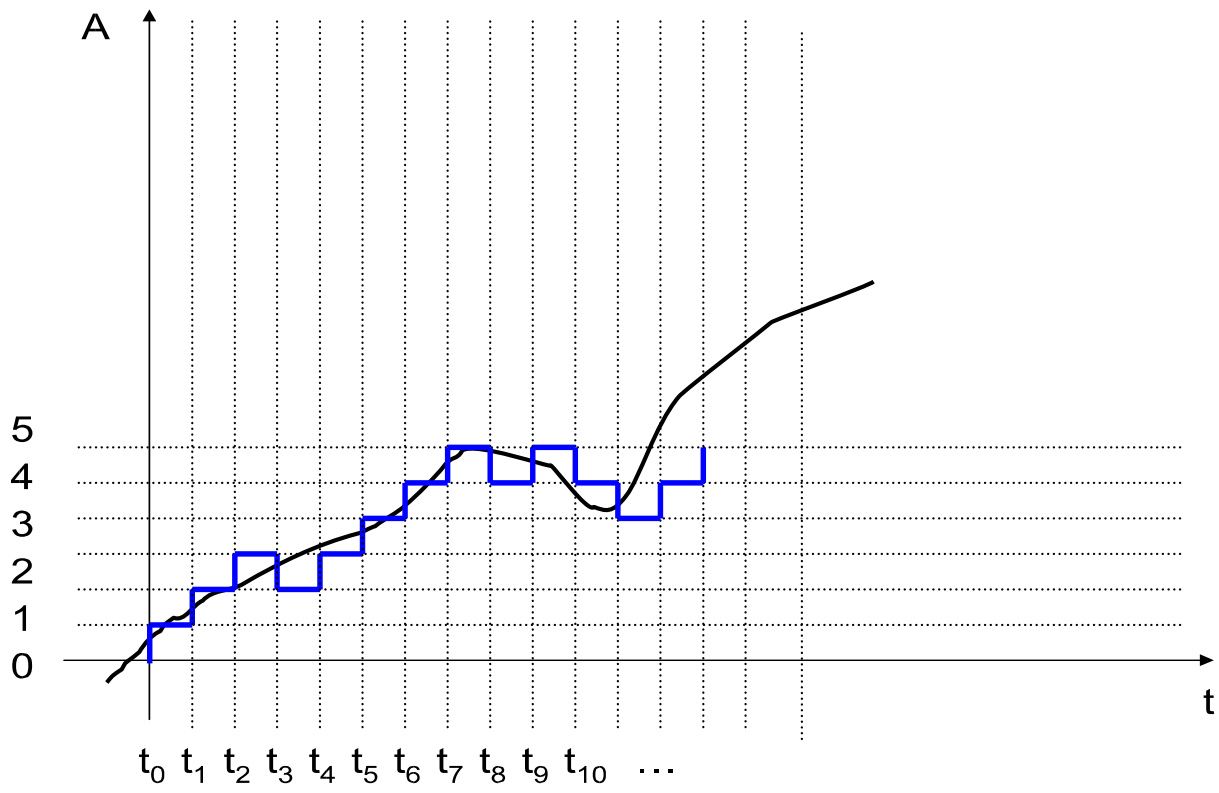
- (1) **step size (δ)** – should not be too small, nor too large
 - small δ + signal changes rapidly \Rightarrow underestimation
 - large δ + signal changes slowly \Rightarrow overestimation
- (2) **sampling time (T)**
 - smaller T increase overall accuracy
 - but, small T increases output data rate, i.e. # of bps

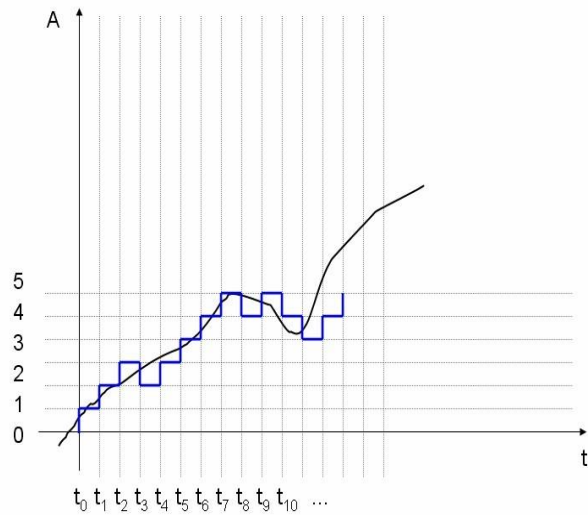
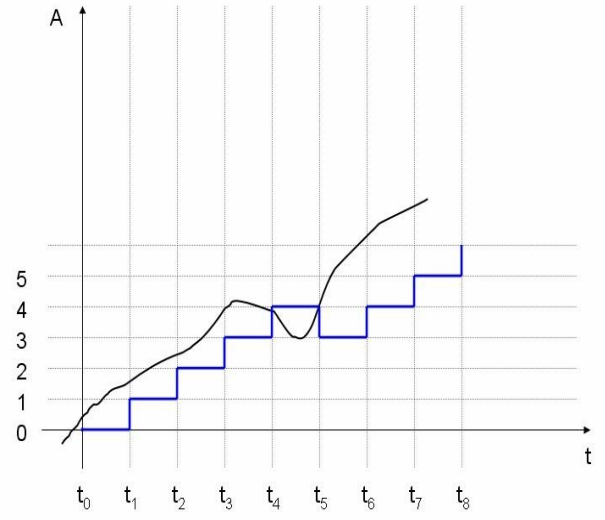
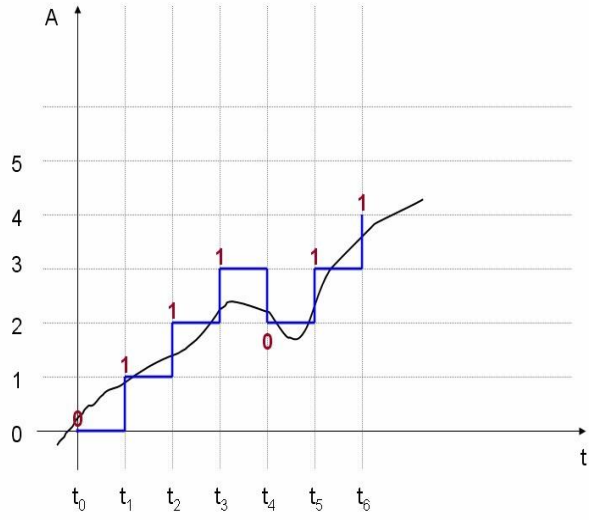


Delta-modulation rule: **smaller $\delta \Rightarrow$ smaller T, larger $\delta \Rightarrow$ larger T.**



Example [Delta Modulation: both δ -step and T reduced 50%]





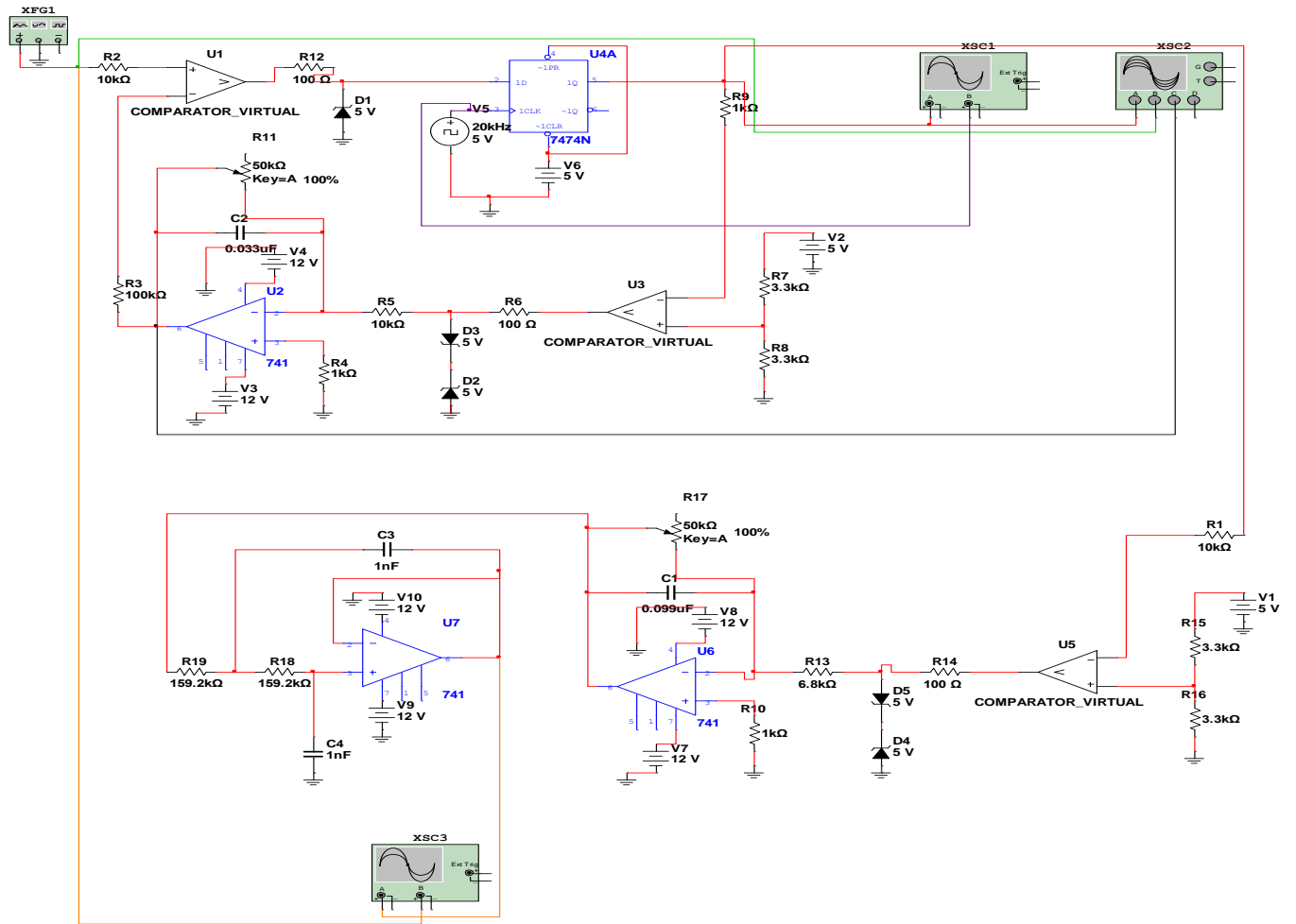


Fig. 1. Delta Modulation and Demodulation Circuit