QMP 7.1 D/F

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belagavi & Approved by AICTE, NewDelhi)
(**NAAC Accredited & ISO 9001:2015 Certified Institution)**
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216.Karnataka.

Department of Electronics & Communication Engineering

Digital Communication Lab

BECL504

B.E - V Semester

Lab Manual 2024-25

Name : ______________________________

USN : ______________________________

Batch : ________________ Section : __________

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belagavi & Approved by AICTE, NewDelhi)
(**NAACAccredited & ISO 9001:2015 Certified Institution**)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216.Karnataka.

Department of Electronics and Communication Engineering

Digital communication Lab

Version 1.0

September 2024

Prepared by:

Mrs.Noor Afza
Mr.Harsha G
Mr.Harshavardhana GS
Mr. Prajwal H S
Assistant Professors, Dept. of ECE,
CIT, Gubbi.

Reviewed by:

Dr. Sanjeevakumar Harihar
Associate Professor,
Dept. of ECE,
CIT, Gubbi

Approved by:

Dr. Suresh D S
Professor & Head,
Dept. of ECE,
CIT, Gubbi

INSTITUTE VISION

- To create centres of excellence in education and to serve the society by enhancing the quality of life through value based professional leadership.

INSTITUTE MISSION

- To provide high quality technical and professionally relevant education in a diverse learning environment.
- To provide the values that prepare students to lead their lives with personal integrity, professional ethics and civic responsibility in a global society.
- To prepare next generation of skilled professionals to successfully compete in the diverse global market.
- To promote campus environment that welcomes and honors women and men of all races, creeds and cultures, values and intellectual curiosity, pursuit of knowledge and academic integrity and freedom.
- To offer wide variety of off-campus education and training programmes to individuals and groups.
- To stimulate collaborative efforts with Industry, Universities, Government and Professional Societies.
- To facilitate public understanding of technical issues and achieve excellence in the operations of the institute.

Quality Policy

- Our Organization delights customers (Student, Parents and Society) by providing value added quality education to meet the National and International requirements.
- We also provide necessary steps to train the students for placement and continue to improve our methods of education to the students through effective. Quality Management System, Quality Policy and Quality Objectives.

Department of Electronics & Communication Engineering

Department Vision

To create globally competent Electronics and Communication Engineering professionals with ethical and moral values for the betterment of the society.

Department Mission

- To nurture the technical/professional/engineering and entrepreneurial skills for overall self and societal upliftment through co-curricular and extra-curricular events.
- To orient the Faculty/Student community towards the higher education, research and development activities.
- To create the Centres of Excellence in the field of electronics and communication in collaboration with industries/Universities by training the faculty through latest technologies.
- To impart quality technical education in the field of electronics and communication engineering to meet over the current/future global industry requirements.

PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

After four Years of Graduation, our graduates are able to:

- Provide technical solutions to real world problems in the areas of electronics and communication by developing suitable systems.
- Pursue engineering career in Industry and/or pursue higher education and research.
- Acquire and follow best professional and ethical practices in Industry and Society.
- Communicate effectively and have the ability to work in team and to lead the team.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

At the end of the B.E Electronics & Communication Engineering program, students are expected to have developed the following program specific outcomes.

PSO1: Build Analog and Digital Electronic systems for Multimedia Applications, VLSI and Embedded Systems in Interdisciplinary Research / Development.

PSO2: Design and Develop Communication Systems as per Real Time Applications and Current Trends.

COURSE OBJECTIVES

1. Design of basic digital modulation techniques using electronic hardware.
2. Simulation of vector computations and derive the orthonormal basis set using Gram Schmidt procedure.
3. Simulate the digital transmission and reception in AWGN channel
4. Simulate the digital modulations using software and display the signals and its vector representations.
5. Implement the source coding algorithms using a suitable software platform.
6. Simulate the channel coding techniques and perform decoding for error detection and correction.

COURSE OUTCOMES

1. Design the basic digital modulation and demodulation circuits for different engineering applications.
2. Design of optimum communication receivers for AWGN channels.
3. Illustration of different digital modulations using the signals and its equivalent vector representations.
4. Implement the source coding and channel coding procedures using suitable software.

PROGRAM OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs.
4. **Conduct investigations of complex problems:** An ability to design and conduct scientific and engineering experiments, as well as to analyze and interpret data to provide valid conclusions
5. **Modern tool usage:** Ability to apply appropriate techniques, modern engineering and IT tools, to engineering problems.
6. **The engineer and society:** An ability to apply reasoning to assess societal, safety, health and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** An ability to understand the impact of professional engineering solutions in societal and environmental contexts
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Ability to function effectively as an individual, and as a member or leader in a team, and in multidisciplinary tasks.
10. **Communication:** Ability to communicate effectively on engineering activities with the engineering community such as, being able to comprehend and write effective reports and design documentation, make effective presentations.
11. **Project management and finance:** An ability to apply knowledge, skills, tools, and techniques to project activities to meet the project requirements with the aim of managing project resources properly and achieving the project's objectives.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belagavi & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SYLLABUS

Digital Communication LAB

Subject Code : BECL504	IA Marks : 50
No. of Practical Hrs/Week : 03	Exam Hours : 03
Total no. of Practical Hrs.: 42	Exam Marks : 50

List of Hardware Experiments

1. Generation and demodulation of the Amplitude Shift Keying signal.
2. Generation and demodulation of the Phase Shift Keying signal.
3. Generation and demodulation of the Frequency Shift Keying signal.
4. Generation of DPSK signal and detection of data using DPSK transmitter and receiver.

List of Simulation Experiments

5. Gram-Schmidt Orthogonalization: To find orthogonal basis vectors for the given set of vectors and plot the orthonormal vectors.
6. Simulation of binary baseband signals using a rectangular pulse and estimate the BER for AWGN channel using matched filter receiver.
7. Perform the QPSK Modulation and demodulation. Display the signal and its constellation.
8. Generate 16-QAM Modulation and obtain the QAM constellation.
9. Encoding and Decoding of Huffman code.
10. Encoding and Decoding of binary data using a Hamming code.
11. For a given data, use CRC-CCITT polynomial to obtain the CRC code. Verify for the cases, a) Without error b) With error
12. Encoding and Decoding of Convolution code

Suggested Learning Resources:

1. B. P Lathi, Zhi Ding, “Modern Digital and Analog Communication Systems” 4th Edition, Oxford University Press, 2017, ISBJ:978-0-19-947628-2
2. Herbert Taub, Donald L Schilling, Goutam Saha, “Principles of Communication Systems”, Mc Graw Hill Education, 2013, ISBN: 978-1-25-902985-1.

COURSE ASSESSMENT AND EVALUATION						
Direct Assessment Methods						
What		**To whom**	**When/Where (Frequency inthe course)**	**Max. Marks**	**Evidence Collected**	**Contributing to Course outcomes**
CIE	Record & Observation	Students	Every lab session (Avg. of all experiment marks)	30	Observation book written at each lab + Record submitted at each lab + Viva	CO1 – CO4
	IA Test		Two	20	Blue Books	CO1 – CO4

MAPPING OF COURSE OUTCOMES WITH PROGRAMOUTCOMES															
		PO												**PSO**	
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**
CO	**1**	2			3										
	2	1			3										
	3	3													
	4					3									
	Sum	6			6	3									
	Avg	2			3	3									

3: High correlation, 2: Medium correlation, 1: Low correlation

INSTRUCTIONS TO THE CANDIDATES

- Student should come with thorough preparation for the experiment to be conducted.

- Student should take prior permission from the concerned faculty before availing the leave.

- Student should come with proper dress code and to be present on time in the laboratory.

- Student will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiment conducted in the previous class.

- Student will not be permitted to attend the laboratory unless they bring the observation book fully completed in all respects pertaining to the experiment to be conducted in present class.

- Experiment should be started conducting only after the staff-in-charge has checked the circuit diagram.

- All the calculations should be made in the observation book. Specimen calculations for one set of readings have to be shown in the practical record.

- Wherever graphs to be drawn, A-4 size graphs only should be used and the same should be firmly attached in the practical record.

- Practical record and observation book should be neatly maintained.

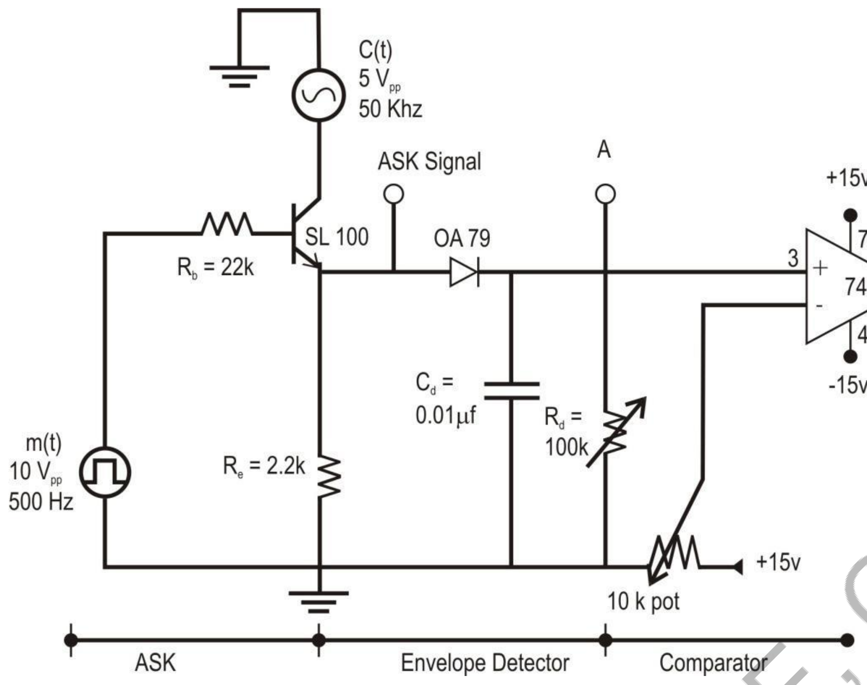
- Student should obtain the signature of the staff-in-charge in the observation book after completing each experiment.

- Theory related to each experiment should be written in the practical record before procedure in your own words with appropriate references.

CONTENTS

Sl No.	Name of the Experiment	Date			Observation Marks (max. 20)	Record Marks (max. 10)	Signature(Student)	Signature(Faculty)
		Conduction	Repetition	Submission of Record				
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
Average								

Circuit Diagram-1: ASK modulator and demodulator



Design:

Let V_c = 5 volts peak-to-peak, V_m =10 volts peak-to-peak, f_m = 500 Hz, f_c =50 kHz. Assume h_{fe}= 30, V_{BEsat}= 0.7 volts, V_{CEsat}= 0.3 volts, I_c=1 mA, $I_c = I_e$.

$V_{c\ peak} = V_{CEsat} + I_eR_e$

2.5 = 0.3 + (1 m)R_e, => R_e = 2.2 kΩ

$V_{mpeak} = R_bI_b+V_{BEsat}+I_eR_e$

5 = R_bI_b + 0.7 + 2.2, where $I_b=I_c/h_{fe}$

then R_{bmax} = 63 kΩ , Choose R_b = 22 k Ω

Envelope Detector:

$1/f_m>R_dC_d>1/f_c$, hence $2ms>R_dC_d>20\mu s$

Let R_dC_d =50/fc=1 ms

Assume C_d=0.01 µF, then R_d=100 kΩ

Tabular column:

V_c in volts	f_c in Hz	V_m in volts	f_m in Hz	Error in detection in ms

Experiment No. 1 **Date: __ /__ / _____**

AMPLITUDE SHIFT KEYING

Aim: To generate ASK signal and to demodulate it.

Apparatus Required:

Sl. No.	Apparatus	Range	Quantity
1	Op-Amp	IC 741	1
2	Transistor	SL 100	1
3	Diode	OA79	1
4	Resistors	As Per the design	
5	Potentiometer	10KΩ	1
6	Capacitor	0.01µF	1

Procedure:

1. Connections are made as shown in the circuit diagram-1.
2. Apply a square wave modulating signal of 500 Hz (1000bits/sec) of $10V_{P-P}$
3. Apply a sine wave carrier signal of 50 kHz of 5V peak-to-peak amplitude.
4. Observe ASK waveform at point A.
5. Demodulate the ASK signal using the envelope detector.
6. To find minimum frequency of carrier signal for proper detection:

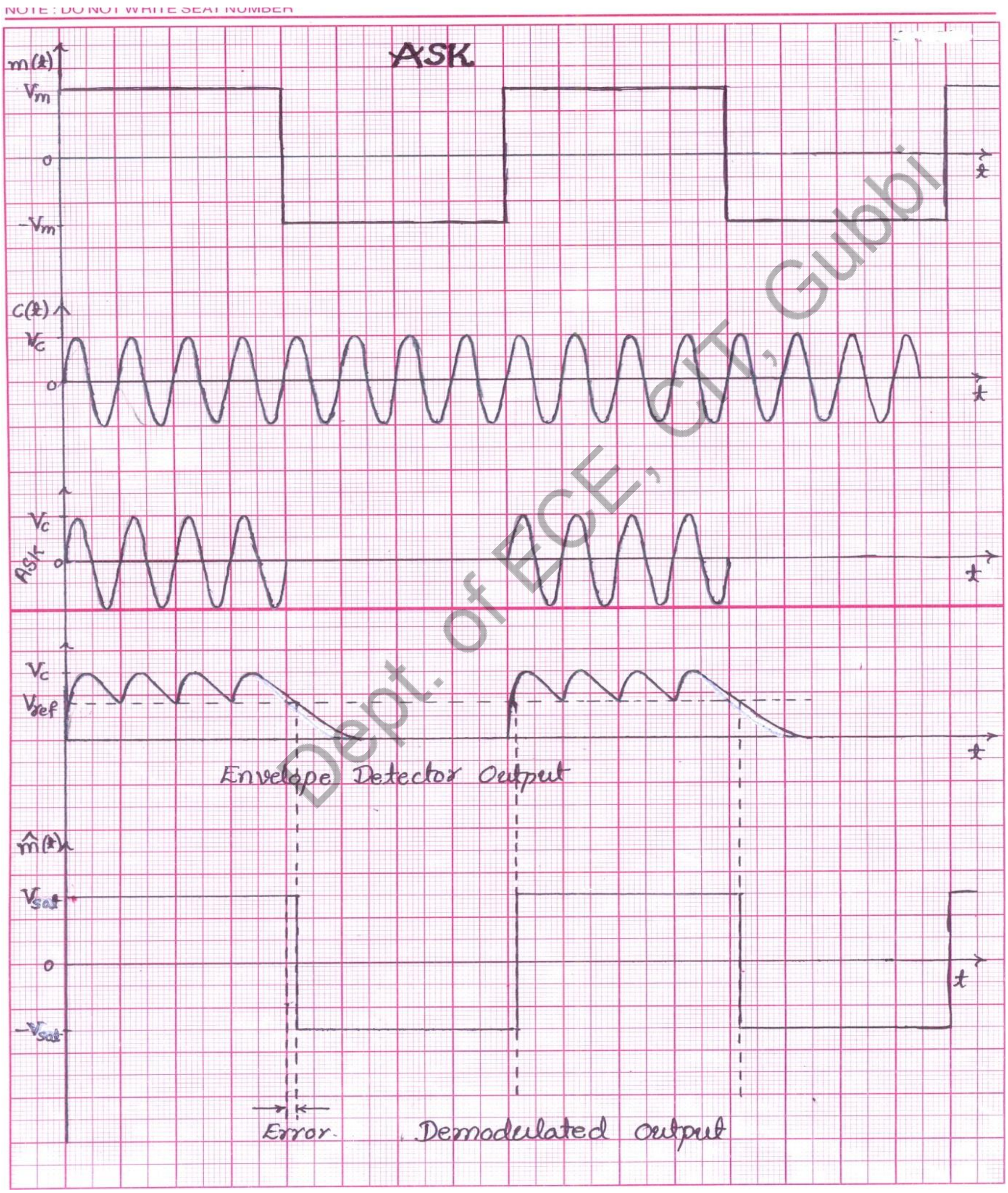
After Step 5 start reducing the frequency of the sine wave carrier signal from 50 kHz gradually. At a particular frequency of carrier signal, the demodulated signal does not tally with the modulating square wave. Note the carrier frequency just before the mismatch.

Result:

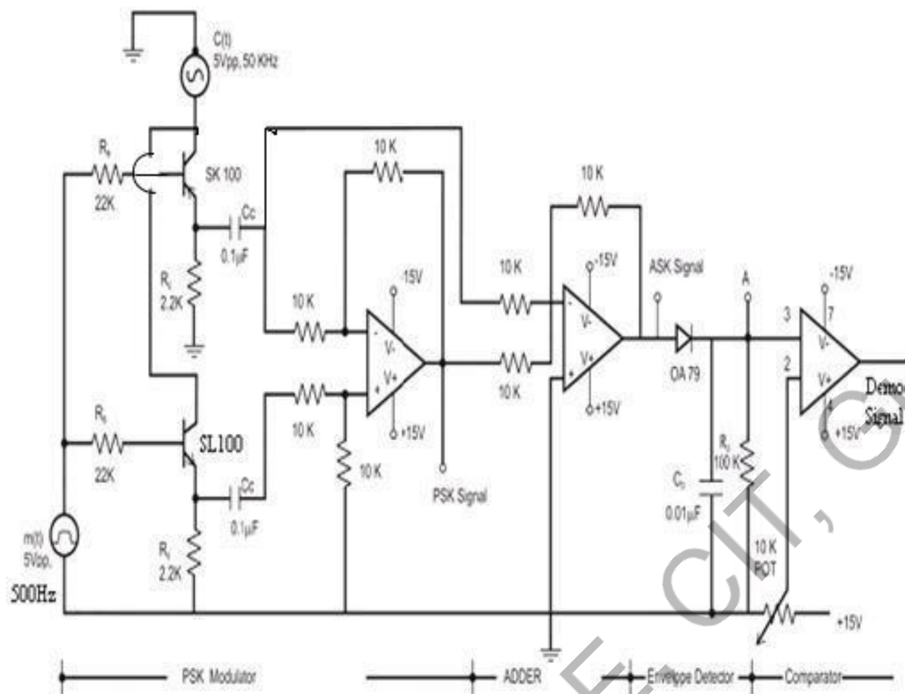
Error = ……….. ms

Minimum frequency for proper detection: ………….Hz

ASK modulation and demodulation waveforms:



Circuit Diagram-2: BPSK modulator and demodulator



Design:

Let V_c = 5 volts peak-to-peak, V_m =10 volts peak-to-peak, f_m = 500 Hz, f_c =50 kHz. Assume h_{fe}= 30, V_{BEsat}= 0.7 volts, V_{CEsat}= 0.3 volts, I_c=1 mA, $I_c = I_e$.

$V_{c\ peak} = V_{CEsat} + I_eR_e$

2.5 = 0.3 + (1 m)R_e, => R_e = 2.2 kΩ

$V_{m\ peak} = R_bI_b+V_{BEsat}+I_eR_e$

5 = R_bI_b + 0.7 + 2.2, where $I_b=I_c/h_{fe}$

Then R_{bmax} = 63 kΩ , Choose R_b = 22 k Ω

Envelope Detector:

$1/f_m>R_dC_d>1/f_c$, hence $2ms>R_dC_d>20μs$

Let R_dC_d =50/fc=1 ms

Assume C_d=0.01 μF, then R_d=100 k Ω

Tabular Column:

Vc in volts	fc in Hz	Vm in volts	fm in Hz	Error in detection in ms

Experiment No. 2 **Date: __ /__ / _____**

BINARY PHASE SHIFT KEYING

Aim: To generate PSK signal and to demodulate the PSK signal.

Apparatus Required:

Sl. No.	Apparatus	Range	Quantity
1	IC 1458		2
2	Transistor	SL 100 SK 100	1 1
3	Diode	OA 79	1
4	Resistors	As Per Design	16
5	Potentiometer	10KΩ	1
6	Capacitor	As Per Design	3

Procedure:

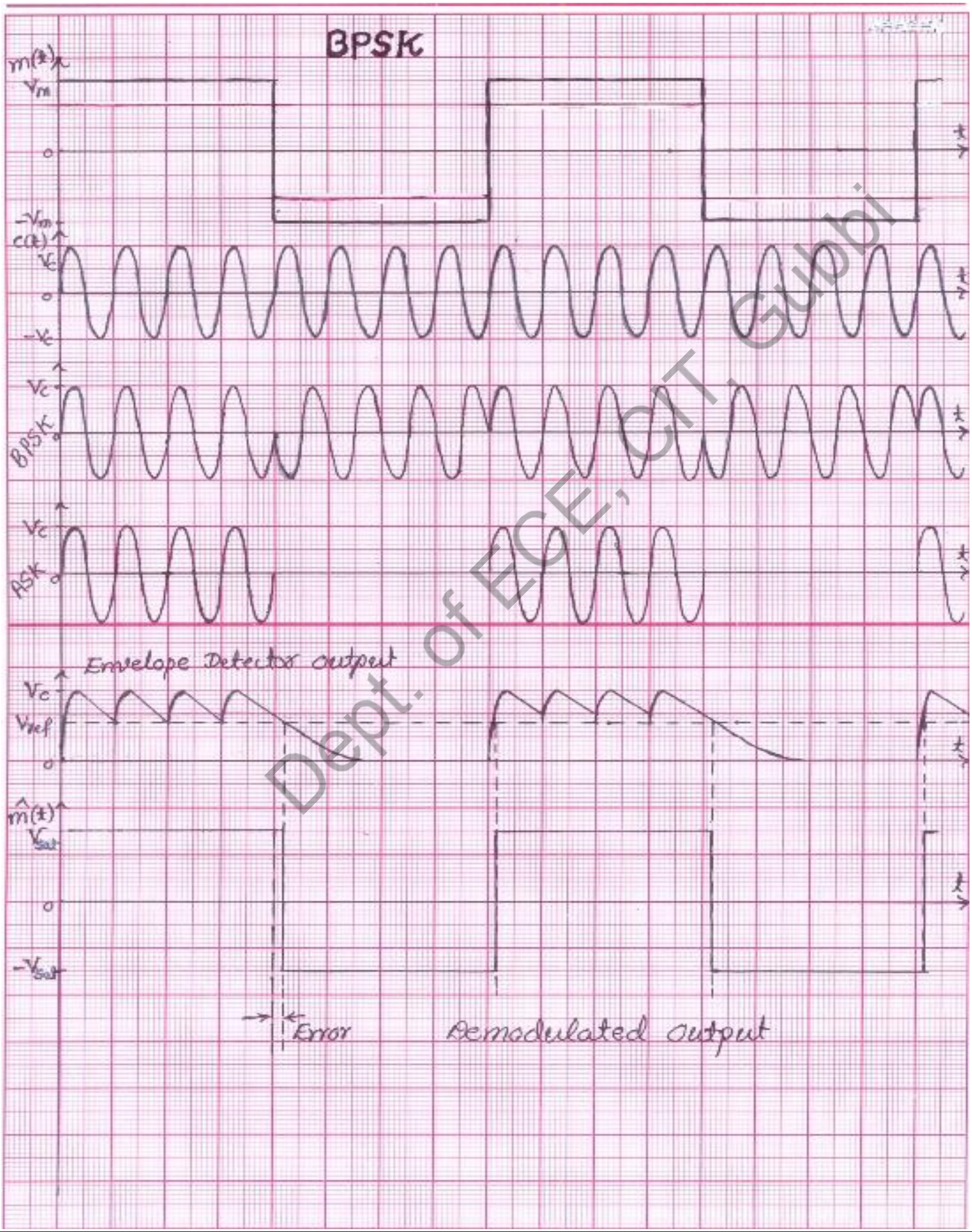
1. Connections are made as shown in the circuit diagram-4.
2. Apply square wave modulating signal of 500 Hz (1000bits/sec) of 10 V_{P-P}.
3. Apply a sine wave carrier signal of 50 kHz of 5V peak amplitude.
4. Observe BPSK waveform at point A.
5. Demodulate the BPSK signal using the coherent detector (Adder + Envelope Detector). The error in the demodulated wave can be minimized by adjusting the V_{ref} using 10k pot.

Result:

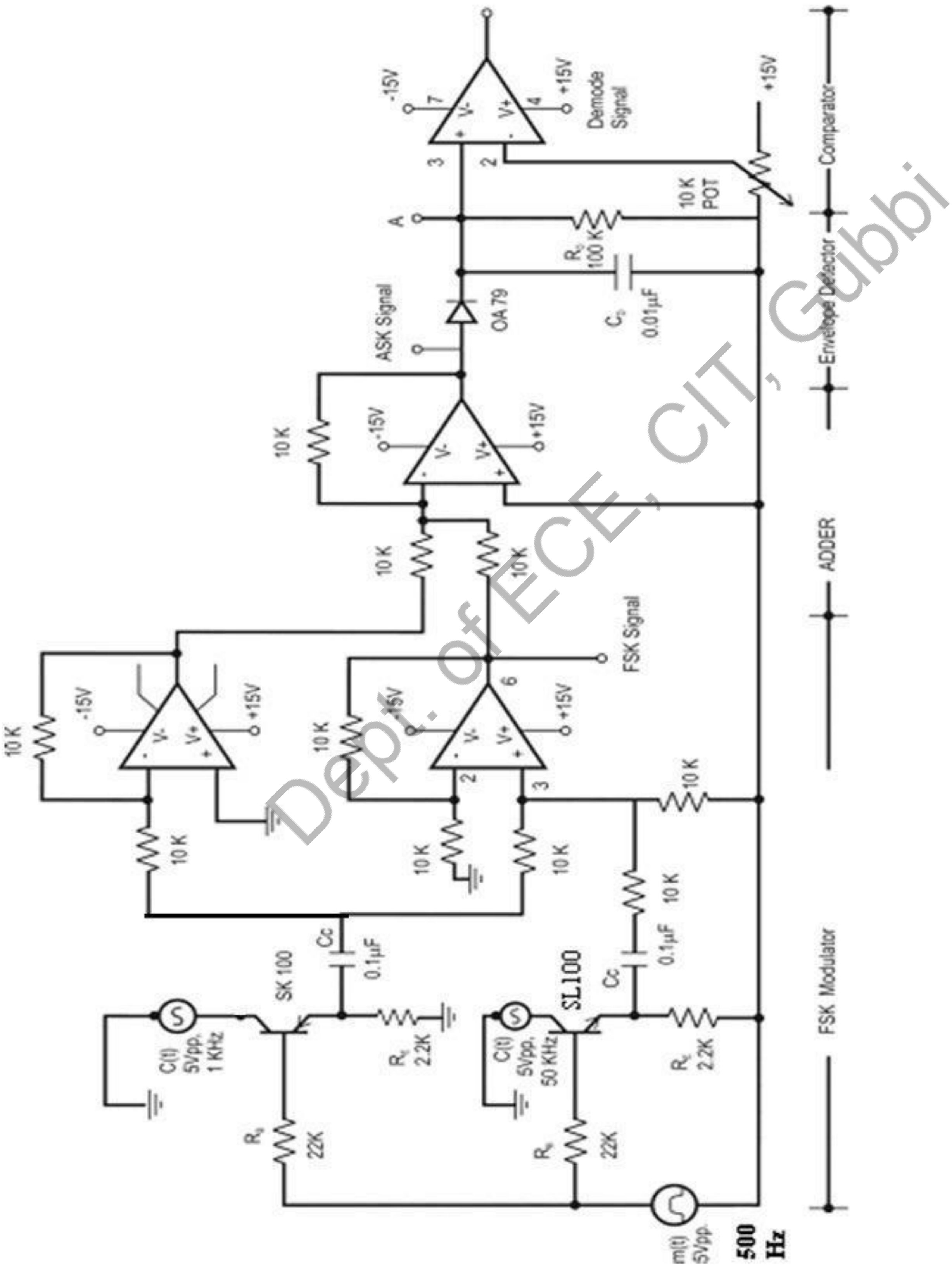
Error = ……..… ms

Minimum frequency for proper detection: ………….Hz.

BPSK modulation and demodulation waveforms:



Circuit Diagram-3: FSK modulator and demodulator



Experiment No. 3 **Date: __ /__ / _____**

FREQUENCY SHIFT KEYING

Aim: To generate FSK signal and to demodulate the FSK signal.

Apparatus Required:

Sl.No	Apparatus	Range	Quantity
1	IC 1458		2
2	Transistor	SL 100 SK 100	1 1
3	Diode	OA 79	1
4	Resistors	As Per Design	16
5	Potentiometer	10KΩ	1
6	Capacitor	As Per Design	3

Procedure:

1. Connections are made as shown in circuit diagram-11.
2. Apply a square wave modulating signal of 100 Hz (200bits/sec) and 10 V_{P-P} amplitude.
3. Apply a sine wave carrier signal-1 of 1 kHz, 5V peak to peak amplitude and signal-2 of 50 kHz, 5V peak to peak amplitude.
4. Observe FSK waveform at point A.
5. Demodulate the FSK signal using the coherent detector (Adder + Envelope Detector). The error in the demodulated waveform can be minimized by adjusting the V_{ref} using 10k POT.

Design:

Let V_c = 5 volts peak-to-peak, V_m =10 volts peak-to-peak, f_m = 500 Hz, f_c =50 kHz.

Assume h_{fe}= 30, V_{BEsat}= 0.7 volts, V_{CEsat}= 0.3 volts, I_c=1 mA, $I_c = I_e$.

$V_{cpeak} = V_{CEsat} + I_eR_e$

$2.5 = 0.3 + (1\ m)R_e$, => $R_e = 2.2\ k\Omega$

$V_{mpeak} = R_bI_b+V_{BEsat}+I_eR_e$

$5 = R_bI_b + 0.7 + 2.2$, where $I_b=I_c/h_{fe}$

then $R_{bmax} = 63\ k\Omega$, Choose $R_b = 22\ k\ \Omega$

Envelope Detector:

$1/f_m>R_dC_d>1/f_c$, hence $2ms>R_dC_d>20\mu s$

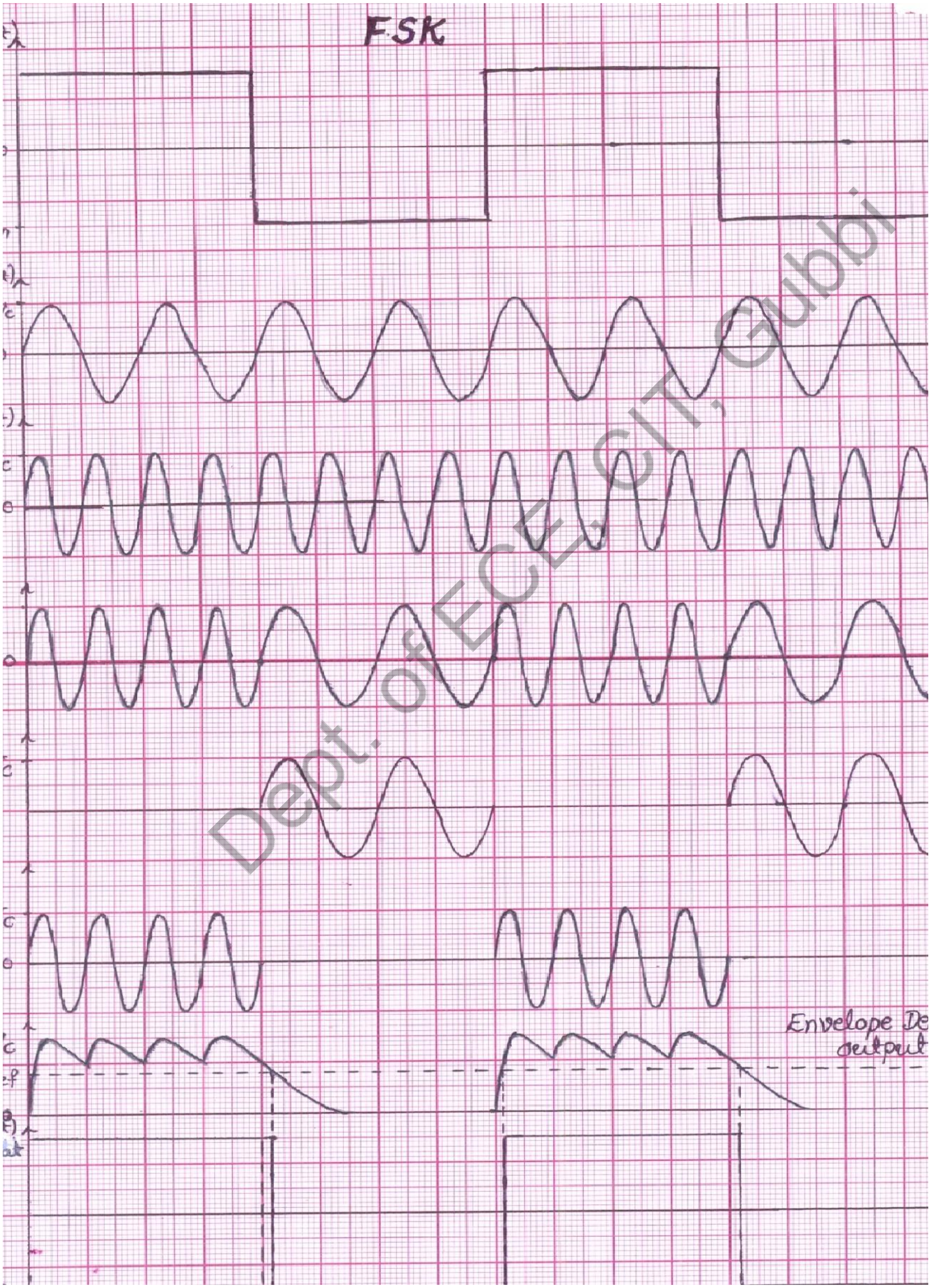
Let R_dC_d =50/fc=1 ms

Assume C_d=0.01 µF, then R_d =100 k Ω

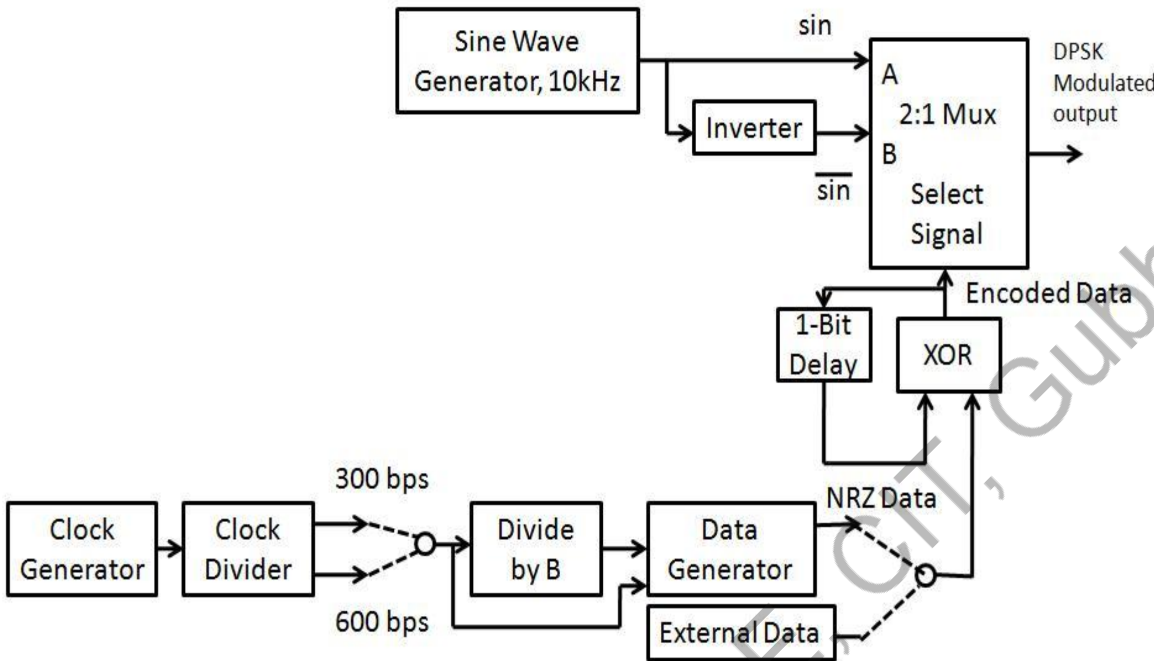
Tabular Column:

Vc in volts	f_{c1} in Hz	f_{c2} in Hz	Vm in volts	f_m in Hz	Error in detection in ms

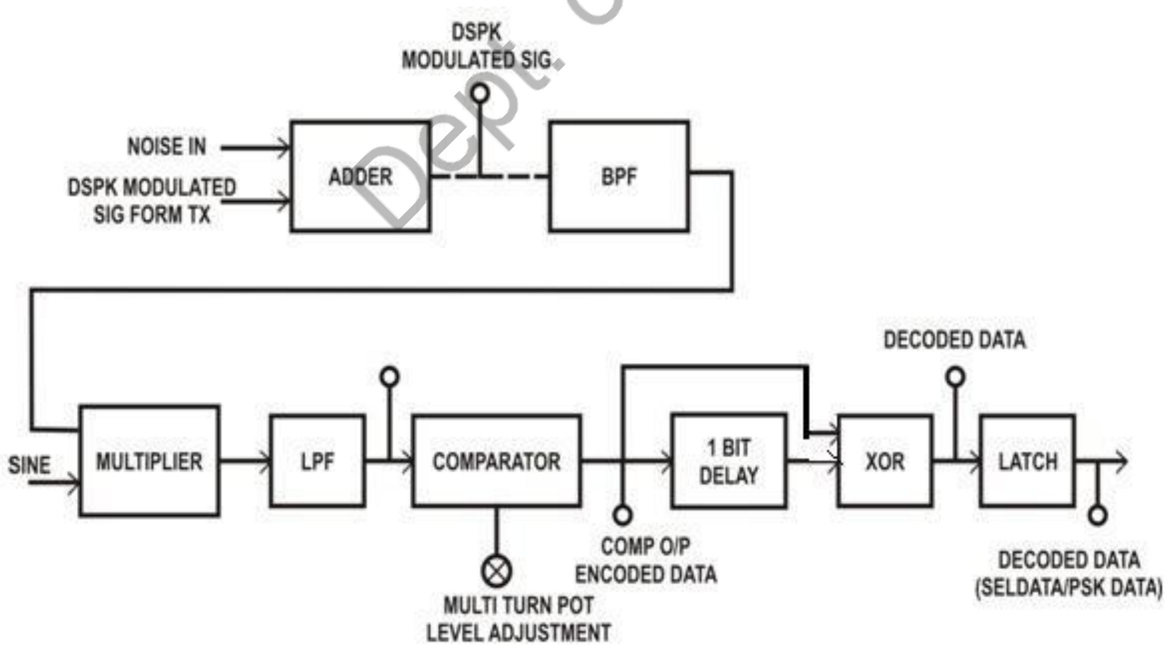
FSK modulation and demodulation waveforms:



Circuit Diagram-4: DPSK Transmitter



Circuit Diagram-5: DPSK Receiver



Experiment No. 4 **Date: __ /__ / _____**

DIFFERENTIAL PHASE SHIFT KEYING

Aim: Study of Carrier Modulation Techniques by DPSK.

Apparatus Required:

Sl. No.	Apparatus	Range	Quantity
1	DPSK Kit	Kit No.	1
2	Power Supply		1
3	Patch Cards		

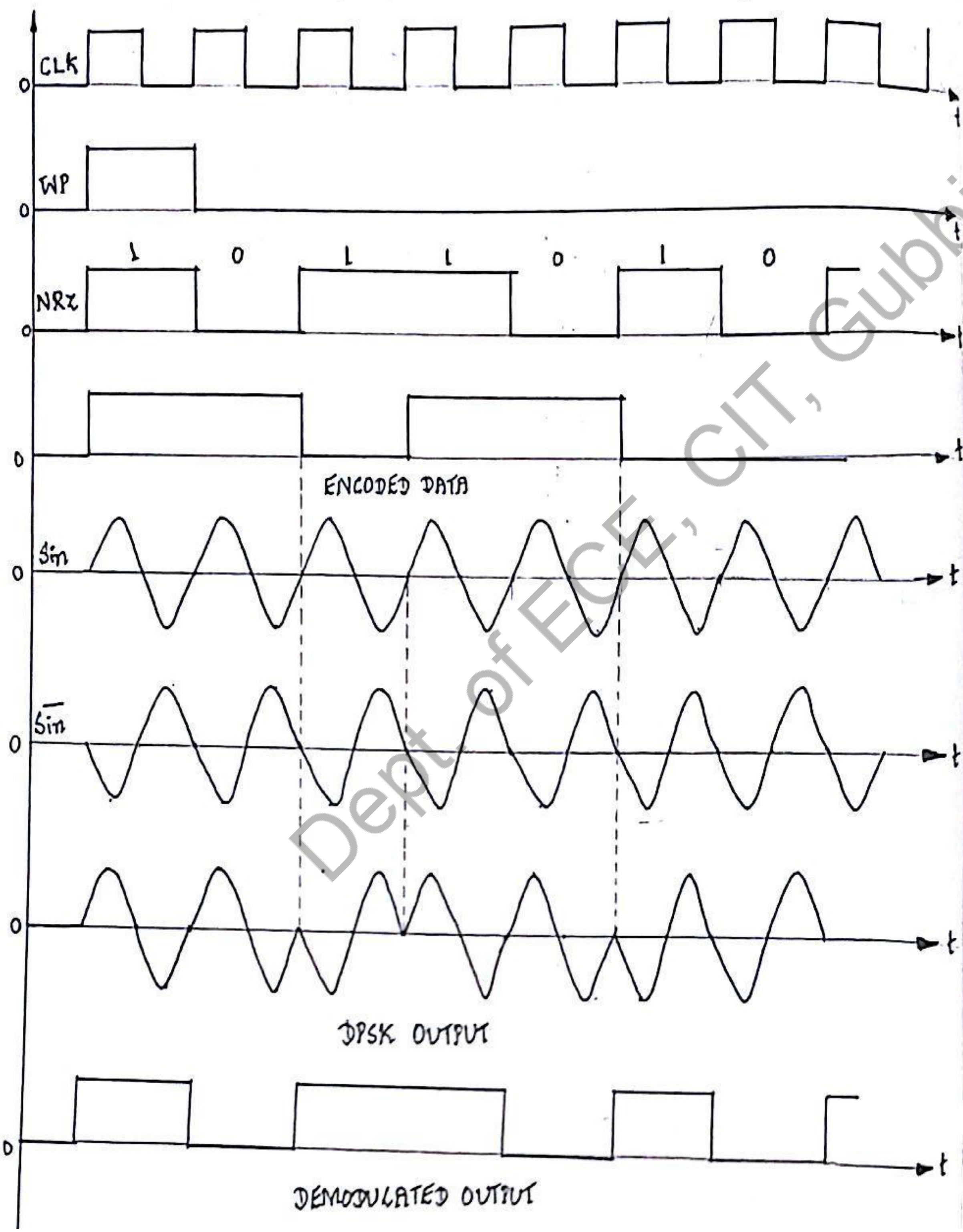
Procedure for Transmitter:

1. Switch on the power supply.
2. Connect either 300 or 600 bps clock to SEL CLK socket, connect measuring probe of CRO to SEL CLK and WORDPULSE to observe the selected clock (300bps or 600bps).
3. Connect MARK (Sine) and MARK (SINE) Mux input.
4. Connect NRZ DATA to SEL DATA. Adjust the dip switch to any digital patterns of 8 bits by keeping them for ‘1’ or ‘0’ positions also observe selected data in CRO.
5. Connect DPSK data to Mux input, also observe DPSK data.

Procedure for Receiver:

1. Interconnect the Transmitter and Receiver modules through the interconnecting cable required.
2. Connect Adder output to BPF input.
3. If external clock is required give from SEL CLK from Tx socket.
4. If External carrier is required, feed it from CARRIER FROM Tx socket.
5. Observe the modulated signal at DPSK IN, DPSK O/P in the CRO. The output should match the data transmitted from the transmitter.

DPSK WAVEFORMS:



Experiment No. 5 **Date: __ /__ / _____**

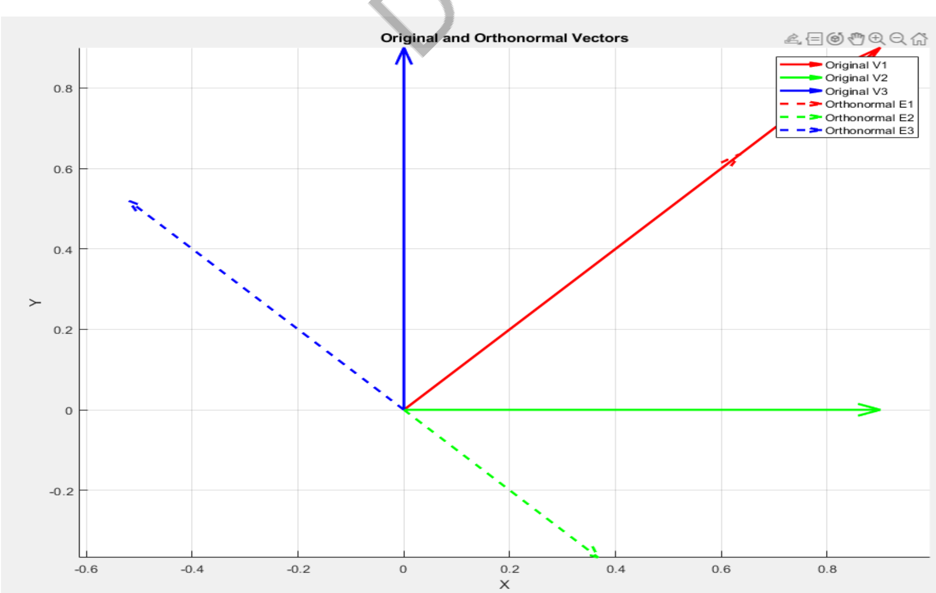
Gram-Schmidt Orthogonalization: To find orthogonal basis vectors for the given set of vectors and plot the orthonormal vectors

```
% Given set of vectors as columns
V = [1 1 0; 1 0 1; 0 1 1]';
% Number of vectors
num_vectors = size(V, 2);
% Initialize orthogonal and orthonormal matrices
U = zeros(size(V));
E = zeros(size(V));
% Gram-Schmidt Process
U(:,1) = V(:,1);  % First orthogonal vector is the first input vector
E(:,1) = U(:,1) / norm(U(:,1));  % First orthonormal vector
for i = 2:num_vectors
    U(:,i) = V(:,i);
    for j = 1:i-1
        U(:,i) = U(:,i) - (dot(V(:,i), E(:,j)) * E(:,j)); % Subtract projection onto previous orthonormal vectors
    end
    E(:,i) = U(:,i) / norm(U(:,i));  % Normalize to get orthonormal vector
end
% Display the orthonormal vectors
disp('Orthonormal basis vectors:');
disp(E);
% Plot the original and orthonormal vectors
figure;
hold on;
grid on;
axis equal;
% Plot original vectors
```

```
quiver3(0, 0, 0, V(1,1), V(2,1), V(3,1), 'r', 'LineWidth', 2);
quiver3(0, 0, 0, V(1,2), V(2,2), V(3,2), 'g', 'LineWidth', 2);
quiver3(0, 0, 0, V(1,3), V(2,3), V(3,3), 'b', 'LineWidth', 2);
 % Plot orthonormal vectors
quiver3(0, 0, 0, E(1,1), E(2,1), E(3,1), 'r--', 'LineWidth', 2);
quiver3(0, 0, 0, E(1,2), E(2,2), E(3,2), 'g--', 'LineWidth', 2);
quiver3(0, 0, 0, E(1,3), E(2,3), E(3,3), 'b--', 'LineWidth', 2);
 xlabel('X');
ylabel('Y');
zlabel('Z');
legend('Original V1', 'Original V2', 'Original V3', 'Orthonormal E1', 'Orthonormal E2', 'Orthonormal E3');
title('Original and Orthonormal Vectors');
hold off;
```

Output:

```
Orthonormal basis vectors:
    0.7071    0.4082   -0.5774
    0.7071   -0.4082    0.5774
         0    0.8165    0.5774
```



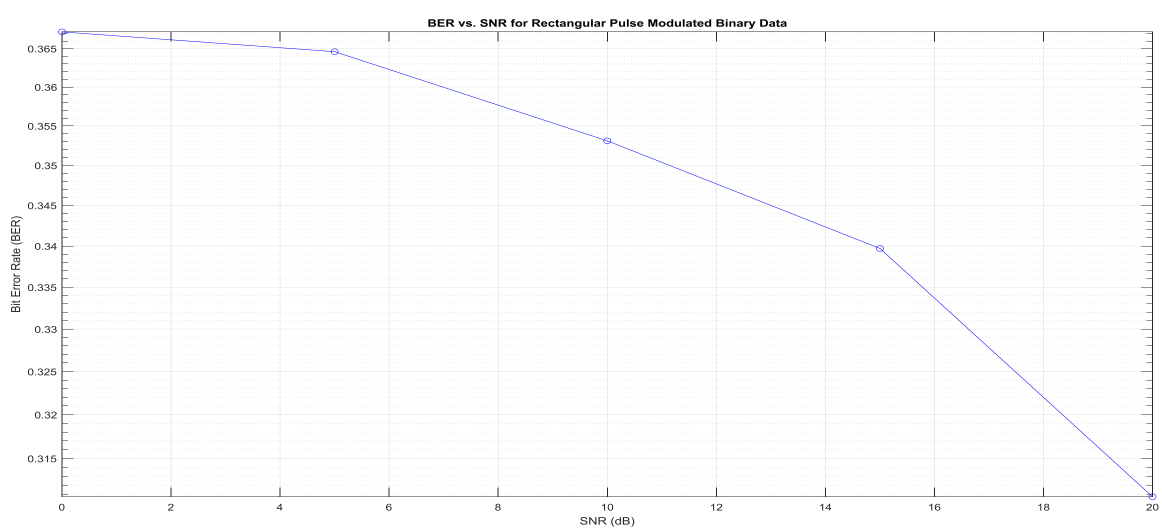
Experiment No. 6 **Date: __ /__ / _____**

Simulation of binary baseband signals using a rectangular pulse and estimate the BER for AWGN channel using matched filter receiver

```
% Simplified Parameters
N = 1e4;            % Number of bits
SNR_dB = 0:5:20;      % SNR values in dB
pulse_width = 1;      % Pulse width for rectangular pulse
 % Generate random binary data
data = randi([0 1], N, 1);
 % Define the rectangular pulse
t = 0:0.01:pulse_width;
rect_pulse = ones(size(t));
 % Initialize BER vector
BER = zeros(length(SNR_dB), 1);
 for snr_idx = 1:length(SNR_dB)
   % Modulate binary data
   tx_signal = [];
   for i = 1:N
       if data(i) == 1
           tx_signal = [tx_signal; rect_pulse'];
       else
           tx_signal = [tx_signal; zeros(size(rect_pulse'))];
       end
   end
       % Add AWGN
   SNR = 10^(SNR_dB(snr_idx) / 10);
   noise_power = 1 / (2 * SNR);
   noise = sqrt(noise_power) * randn(length(tx_signal), 1);
   rx_signal = tx_signal + noise;
```

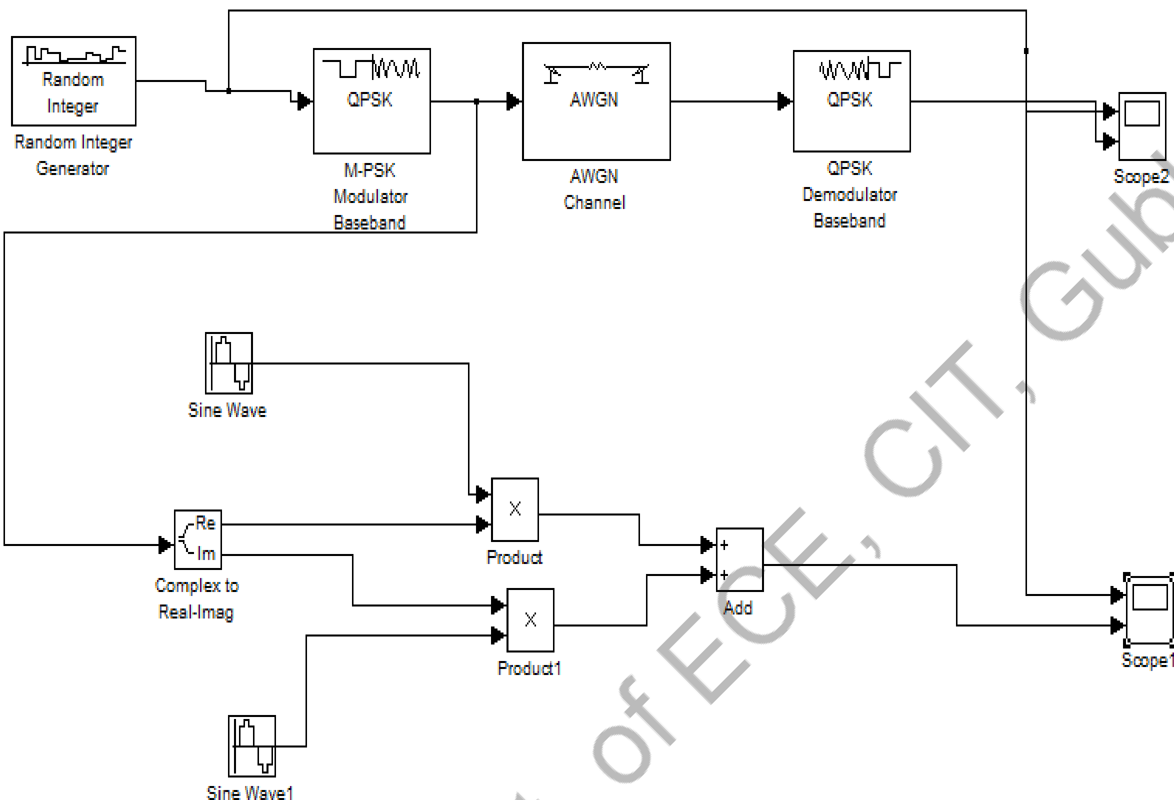
```
    % Matched Filter
  matched_filter = rect_pulse;
  filtered_signal = conv(rx_signal, matched_filter, 'same');
    % Sample the output of the matched filter
  sample_interval = round(length(filtered_signal) / N);
  sampled_signal = filtered_signal(1:sample_interval:end);
    % Decision (Threshold = 0.5)
  estimated_bits = sampled_signal > 0.5;
    % Compute BER
  num_errors = sum(estimated_bits ~= data);
  BER(snr_idx) = num_errors / N;
end
 % Plot BER vs. SNR
figure;
semilogy(SNR_dB, BER, 'b-o');
grid on;
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
title('BER vs. SNR for Rectangular Pulse Modulated Binary Data');
```

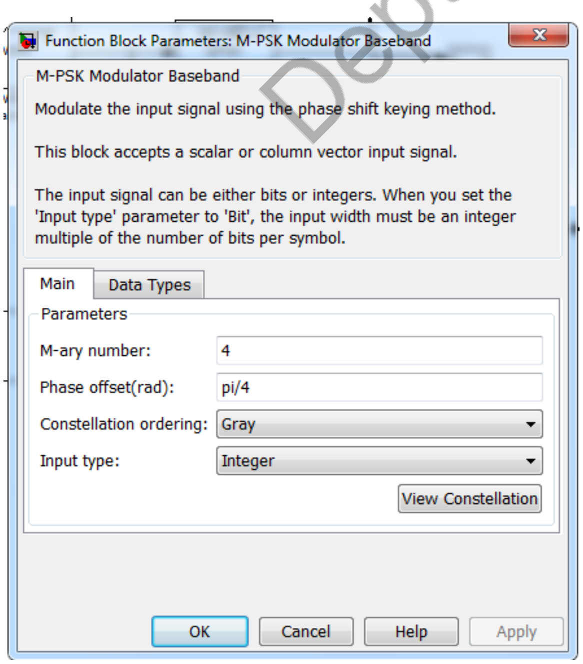

Output:

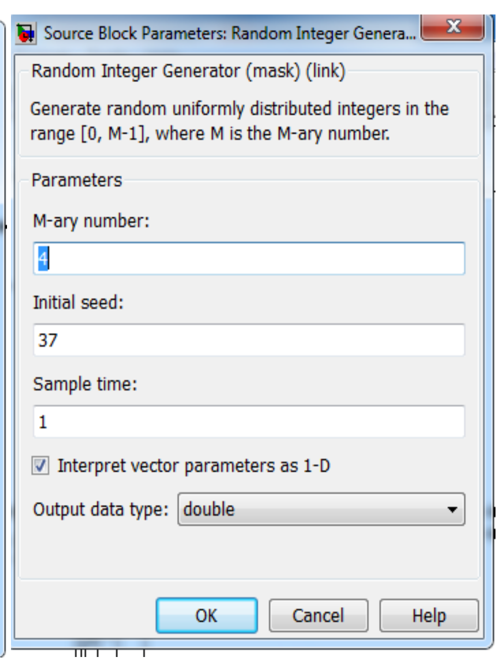


Experiment No. 7 **Date: __ /__ / _____**

QPSK Modulation and demodulation







EXPERIMENT-7

QPSK Transmitter and Receiver in Simulink.

To create a QPSK (Quadrature Phase Shift Keying) transmitter and receiver in Simulink, you can follow these steps. This example assumes a basic QPSK modulation and demodulation system without considering channel effects or noise.

QPSK Transmitter in Simulink:

Create a New Simulink Model: Open MATLAB and then type simulink in the Command Window to open the Simulink environment.

Click on "New" and select "Simulink Model" to create a new model.

QPSK Modulator: In the Simulink Library Browser, navigate to "Communications" > "Digital Modulation" and drag the "QPSK Modulator Baseband" block into your model.

Source for Binary Data: Add a source block to provide binary data. You can use the "Random Integer" block from the "Sources" library or any other source depending on your application.

Scope (Optional): To visualize the QPSK-modulated signal, add a "Scope" block from the "Sinks" library.

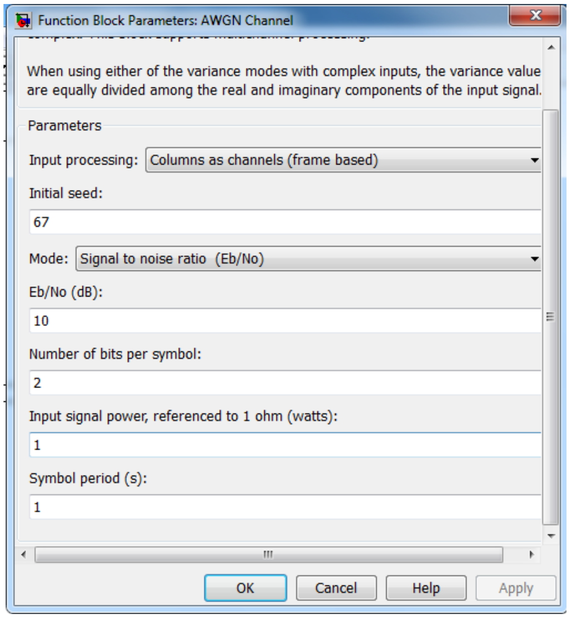
Connect Blocks: Connect the output of the binary data source to the input of the QPSK Modulator. Also, connect the output of the QPSK Modulator to the input of the Scope block if you want to visualize the modulated signal.

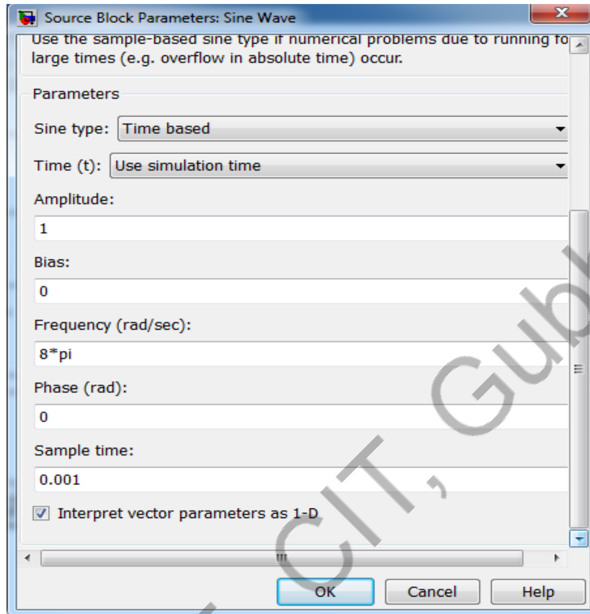
QPSK Receiver in Simulink:

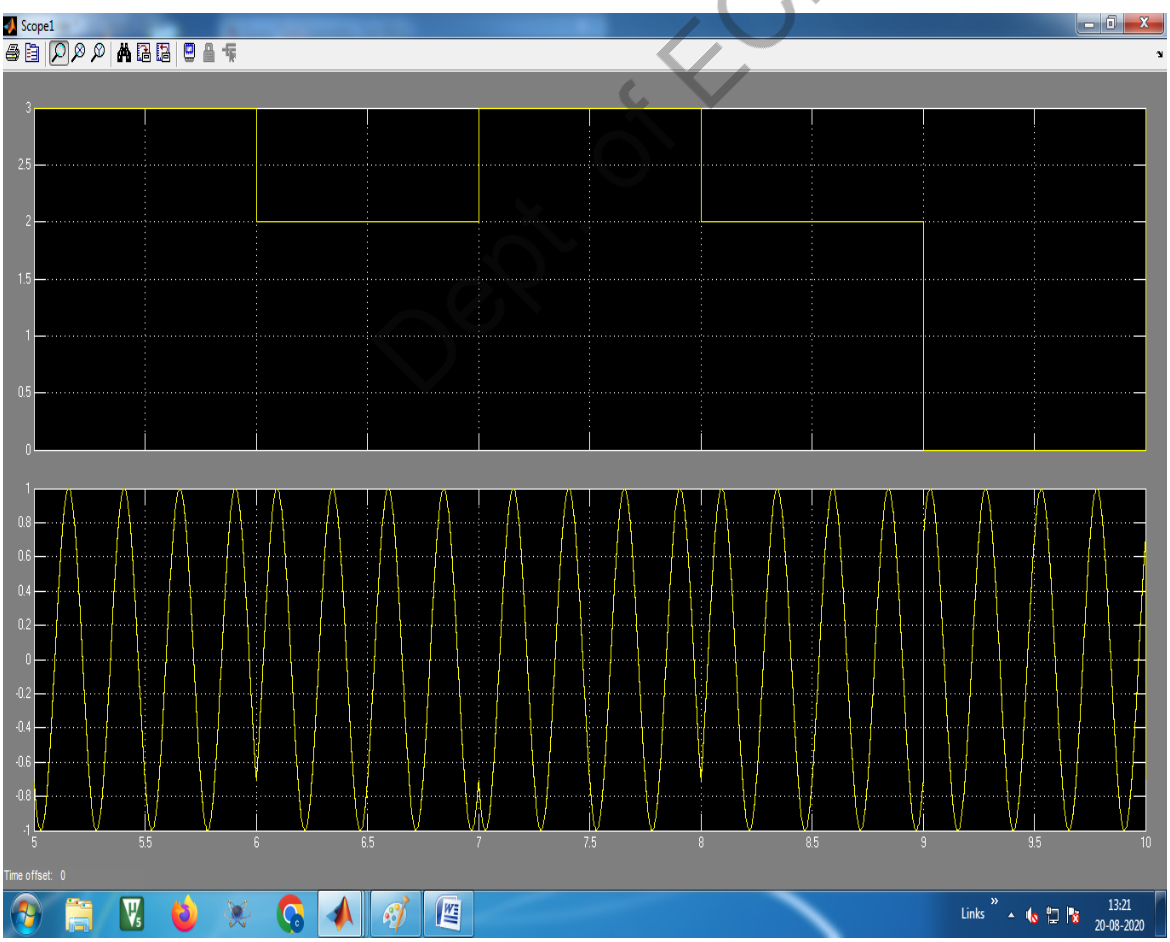
QPSK Demodulator: In the Simulink Library Browser, navigate to "Communications" > "Digital Modulation" and drag the "QPSK Demodulator Baseband" block into your model.

Scope (Optional): To visualize the QPSK-demodulated signal, add another "Scope" block from the "Sinks" library.

Connect Blocks: Connect the output of the QPSK Modulator (or any other source of QPSK-modulated signal) to the input of the QPSK Demodulator. Also, connect the output of the QPSK Demodulator to the input of the Scope block if you want to visualize the demodulated signal.


Function Block Parameters: AWGN Channel
When using either of the variance modes with complex inputs, the variance value are equally divided among the real and imaginary components of the input signal.
Parameters
Input processing: Columns as channels (frame based)
Initial seed:
67
Mode: Signal to noise ratio (Eb/No)
Eb/No (dB):
10
Number of bits per symbol:
2
Input signal power, referenced to 1 ohm (watts):
1
Symbol period (s):
1
OK
Cancel
Help
Apply


Source Block Parameters: Sine Wave
Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.
Parameters
Sine type: Time based
Time (t): Use simulation time
Amplitude:
1
Bias:
0
Frequency (rad/sec):
8*pi
Phase (rad):
0
Sample time:
0.001
Interpret vector parameters as 1-D
OK
Cancel
Help


Scope1
Time offset: 0
Links
13:21
20-08-2020

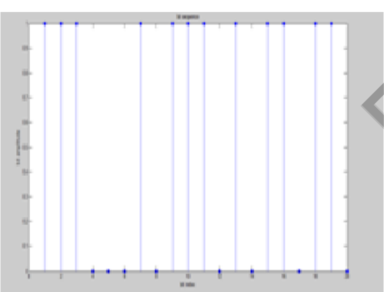
Experiment No. 8 **Date: __ /__ / _____**

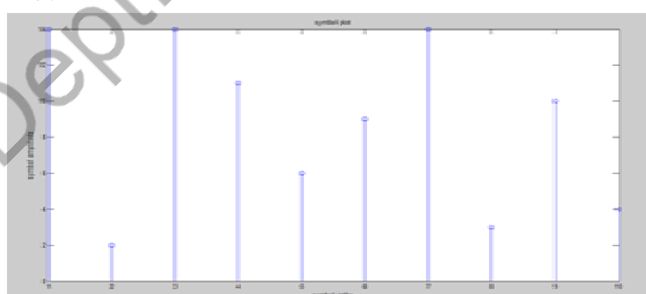
Generate 16-QAM Modulation and obtain the QAM constellation

```
close all
m=16
k=log2(m);
n=9e3;
nsamp=1;
x=randint(n,1);
stem(x(1:20),'filled');
title('bit sequence');
xlabel('bit index');
ylabel('bit amplitude');
xsym=bi2de(reshape(x,k,length(x)/k).','left-msb');
figure;
stem(xsym(1:10));
title('symbol plot');
xlabel('symbol index');
ylabel('symbol amplitude');
y=modulate(modem.qammod(m),xsym);
ytx=y;
ebno=10
snr=ebno+10*log(k)-10*log10(nsamp);
yn=awgn(ytx,snr);
yrx=yn;
scatterplot(y);
scatterplot(yrx,30);
```

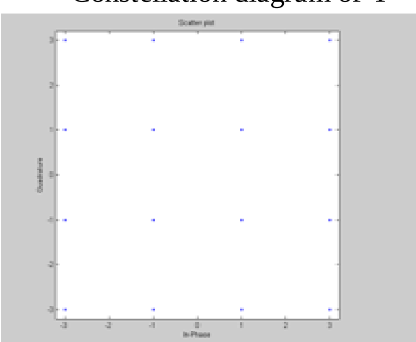

Output:

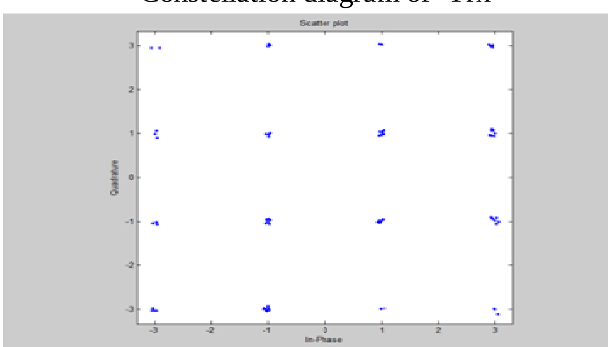
bit sequence Symbol Plot





Constellation diagram of Y Constellation diagram of Yrx





EXPERIMENT NO-8

Modulation & demodulation of a random binary data stream using 16 – QAM

Aim: Modulation & demodulation of a random binary data stream using 16 – QAM.

Theory: 16-QAM, or 16 Quadrature Amplitude Modulation, is a modulation scheme used in digital communication to transmit data. It is a type of QAM where each symbol represents four bits, and there are 16 possible combinations of amplitude and phase.

In QAM, two carriers that are 90 degrees out of phase (quadrature) are used to transmit data. The amplitude and phase of each carrier are varied to represent different combinations of bits. In the case of 16-QAM, there are 16 possible combinations, and each combination represents a unique pattern of four bits.The advantage of using higher-order QAM, such as 16-QAM, is that it allows for higher data rates compared to lower-order modulation schemes like BPSK (Binary Phase Shift Keying) or QPSK (Quadrature Phase Shift Keying).

However, higher-order QAM is more susceptible to noise and interference, making it more challenging to maintain a reliable connection in the presence of adverse conditions. Other common QAM schemes include 4-QAM, 64-QAM, and 256-QAM, each offering different trade-offs between data rate and robustness.

The choice of modulation scheme depends on factors such as the available bandwidth, signal-to-noise ratio, and the desired data rate in a specific communication system.

Experiment No. 9 **Date: __ /__ / _____**

Encoding and Decoding of Huffman code.

```
x = input ('Enter the number of symbols:');
N = 1:x;
disp('The number of symbols are N:');
disp(N);
P = input('Enter the probabilities = ');
disp(P);
S  = sort(P, 'descend');
disp('The sorted probabilities are:');
disp(S);
[dict,avglen] = huffmandict(N,S);
disp('The average length of the code is:');
disp(avglen);
H = 0;
for     i = 1:x
        H = H + (P(i) * log2(1/P(i)));
end
disp('Entropy is:');
disp(H);
disp('bits/msg');
E = (H/avglen) * 100;
disp('Efficiency is:');
disp(E);
codeword = huffmanenco(N,dict);
disp('The codewords are:');
disp(codeword);
decode = huffmandeco(codeword,dict);
disp('Decoded output is:');
disp(decode);
```

Output:

Enter the number of symbols: 5

The number of symbols is N:

1 2 3 4 5

Enter the probabilities =

0.4 0.25 0.2 0.1 0.05

The sorted probabilities are:

0.4000 0.2500 0.2000 0.1000 0.0500

The average length of the code is:

2.1000

Entropy is:

2.0414 bits/msg

Efficiency is:

97.217

The codewords are:

10 100 000 100 011

Decoded output is:

1 2 3 4 5

Experiment No. 10 **Date: __ /__ / _____**

Encoding and Decoding of binary data using a Hamming code.

```
% Hamming Code Encoding and Decoding Example
% Parameters of the Hamming Code
n = 15; % Codeword length
k = 11; % Message length
% Generate random binary data of length k (message length)
data = randi([0 1], k, 1); % Random message of k bits
disp('Original Message:');
disp(data');
% Encode the data using Hamming code (binary)
encData = encode(data, n, k, 'hamming/binary');
disp('Encoded Message:');
disp(encData');
% Introduce an error in the encoded message
errLoc = randerr(1, n); % Generate random error positions (1-bit error)
encData_corrupted = mod(encData + errLoc', 2); % Corrupt the encoded message

disp('Corrupted Encoded Message:');
disp(encData_corrupted');

% Decode the corrupted message using Hamming decoding
decData = decode(encData_corrupted, n, k, 'hamming/binary');

disp('Decoded Message (after error correction):');
disp(decData');

% Compare original data and decoded data
numerr = biterr(data, decData); % Count number of bit errors between original and decoded data
disp(['Number of bit errors between original and decoded message: ', num2str(numerr)]);
```

Output:

Original Message:

1 0 1 1 0 1 1 0 0 1 0

Encoded Message:

1 0 1 1 1 0 0 1 1 1 1 0 0 1 0

Corrupted Encoded Message:

1 0 1 1 1 0 0 1 1 0 1 0 0 1 0

Decoded Message (after error correction):

1 0 1 1 0 1 1 0 0 1 0

Number of bit errors between original and decoded message: 0

Experiment No. 11 **Date: __ /__ / _____**

ERROR DETECTING CODE Using CRC-CCITT (16-bit)

AIM: *C Program for ERROR detecting code using CRC-CCITT (16bit).*

THEORY :

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The **CRC** calculation or *cyclic redundancy check* was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it. Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.

The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation—by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

```
                    1 0 1 = 5
            -------------
1 0 0 1 1 / 1 1 0 1 1 0 1
            1 0 0 1 1 | |
            --------- | |
              1 0 0 0 0 |
              0 0 0 0 0 |
              --------- |
              1 0 0 0 0 1
                1 0 0 1 1
                ---------
                  1 1 1 0 = 14 = remainder
```

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra to check only costs one binary comparison and in 50% of the cases one binary subtraction. You can easily increase the number of bits of the test data string—for example to 56 bits if we use our example value "*Lammert*"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.

All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an *exclusive or* operation. Though some differences exist in the specifics across different CRC formulas, the basic mathematical process is always the same:

- The message bits are appended with *c* zero bits; this *augmented message* is the dividend
- A predetermined *c+1*-bit binary sequence, called the *generator polynomial*, is the divisor
- The checksum is the *c*-bit remainder that results from the division operation

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC-CCITT	**CRC-16**	**CRC-32**
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101	100000100110000010001110110110111

Table 1. International Standard CRC Polynomials

Error detection with CRC

Consider a message represented by the polynomial M(x)

Consider a *generating polynomial* G(x)

This is used to generate a CRC = C(x) to be appended to M(x).

Note this G(x) is prime.

Steps: 1. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).

1. Divide the result by G(x). The remainder = C(x).

   Special case: This won't work if bitstring =all zeros. We don't allow such an M(x).But M(x) bitstring = 1 will work, for example. Can divide 1101 into 1000.

2. If: x div y gives remainder c

   that means: x = n y + c , Hence (x-c) = n y

   (x-c) div y gives remainder 0

Here (x-c) = (x+c)

Hence (x+c) div y gives remainder 0

3. Transmit: T(x) = M(x) + C(x)
4. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

Note if G(x) has order n - highest power is x^n,

then G(x) will cover (n+1) bits

and the *remainder* will cover n bits. i.e. Add n bits (Zeros) to message.

Some CRC polynomials that are actually used

Some CRC polynomials

- CRC-8:
  x^8+x^2+x+1
  - Used in: 802.16 (along with error *correction*).
- CRC-CCITT:
  $x^{16}+x^{12}+x^5+1$
  - Used in: HDLC, SDLC, PPP default
- IBM-CRC-16 (ANSI):
  $x^{16}+x^{15}+x^2+1$
- 802.3:
  $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
  - Used in: Ethernet, PPP rootion

<u>C- LANGUAGE PROGRAM CODE</u>

```
#include<stdio.h>
int a[100],b[100],i,j,len,k,count=0;

//Generator Polynomial:g(x)=x^16+x^12+x^5+1
int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,};

int main()
{
	void div();
	system("clear");
	printf("\nEnter the length of Data Frame :");
	scanf("%d",&len);
```

```
    printf("\nEnter the Message :");
    for(i=0;i<len;i++)
    scanf("%d",&a[i]);

    //Append r(16) degree Zeros to Msg bits
    for(i=0;i<16;i++)
        a[len++]=0;
    //Xr.M(x)  (ie. Msg+16 Zeros)
    for(i=0;i<len;i++)
        b[i]=a[i];

    //No of times to be divided ie.Msg Length
    k=len-16;
    div();
    for(i=0;i<len;i++)
        b[i]=b[i]^a[i];  //MOD 2 Substraction
    printf("\nData to be transmitted : ");
    for(i=0;i<len;i++)
        printf("%2d",b[i]);

    printf("\n\nEnter the Reveived Data : ");
    for(i=0;i<len;i++)
        scanf("%d",&a[i]);

    div();

    for(i=0;i<len;i++)
        if(a[i]!=0)
        {
            printf("\nERROR in Recived Data");
            return 0;
        }
    printf("\nData Recived is ERROR FREE");
}

void div()
```

```
{
        for(i=0;i<k;i++)
        {
                if(a[i]==gp[0])
                {
                        for(j=i;j<17+i;j++)
                        a[j]=a[j]^gp[count++];
                }
                count=0;
        }
}
```

Output:

Enter the length of Data Frame :4

Enter the Message :1 0 1 1

Data to be transmitted : 1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1

Enter the Reveived Data : 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1

ERROR in Recived Data

Remender is : 0000000100000000

Experiment No. 12 **Date: __ /__ / _____**

Encoding and Decoding of Convolution code

```
% Complete Example of Convolutional Code Encoding and Decoding

% Input message to be encoded

msg = [1 0 1 1 0 1 0 0];

% Define constraint length and generator polynomial

constraint_length = 3;

generator_polynomials = [7 5];

% Create trellis

trellis = poly2trellis(constraint_length, generator_polynomials);

% Encode the message

encoded_msg = convenc(msg, trellis);

% Introduce some noise (optional)

% Example: Flip a bit in the encoded message to simulate noise

encoded_msg_noisy = encoded_msg;

encoded_msg_noisy(4) = ~encoded_msg_noisy(4);  % Flip the 4th bit


% Decode the noisy message using Viterbi decoder

traceback_length = 5;

decoded_msg = vitdec(encoded_msg_noisy, trellis, traceback_length, 'trunc', 'hard');

% Display results

disp('Original Message:');

disp(msg);

disp('Encoded Message:');

disp(encoded_msg);

disp('Noisy Encoded Message:');
```

```
disp(encoded_msg_noisy);
disp('Decoded Message:');
disp(decoded_msg);
```

Output:

```
Original Message:
     1     0     1     1     0     1     0     0

Encoded Message:
     1     1     0     0     1     0     0     0     1     1     1     0     1     1     0

Noisy Encoded Message:
     1     1     0     1     1     0     0     0     1     1     1     0     1     1     0

Decoded Message:
     1     0     1     1     0     1     0     0
```

Beyond the Syllabus Experiment

Experiment No. 13 **Date: __ /__ / _____**

```
        % Parameters
numCarriers = 64;          % Number of subcarriers
numSymbols = 100;           % Number of OFDM symbols
symbolPeriod = 1e-6;        % Symbol period in seconds
samplingFrequency = 1e6;    % Sampling frequency in Hz
channelTapDelay = [0, 2, 5];  % Multipath channel tap delays in samples
channelTapGain = [1, 0.8, 0.5]; % Multipath channel tap gains

% Generate random input data
dataIn = randi([0, 1], numCarriers, numSymbols);
 % OFDM modulation
ofdmSymbols = ifft(dataIn, numCarriers);
 % Add cyclic prefix
cpLength = 16;
ofdmSymbolsWithCP = [ofdmSymbols(end - cpLength + 1:end, :); ofdmSymbols];

% Transmit through a frequency-selective multipath channel
channelOutput = zeros(size(ofdmSymbolsWithCP));
for i = 1:length(channelTapDelay)
   channelOutput = channelOutput + channelTapGain(i) * [zeros(channelTapDelay(i), size(ofdmSymbolsWithCP, 2));
ofdmSymbolsWithCP(1:end - channelTapDelay(i), :)];
end
 % Plot time-domain signals
figure;
 % Plot transmitted signal in time domain
subplot(2, 1, 1);
plot(real(ofdmSymbolsWithCP(:)));
title('Transmitted Signal in Time Domain');
xlabel('Sample Index');
ylabel('Amplitude');
 % Plot received signal in time domain
subplot(2, 1, 2);
plot(real(channelOutput(:)));
title('Received Signal in Time Domain');
xlabel('Sample Index');
ylabel('Amplitude');

% Plot frequency-domain signals
figure;

% Plot transmitted signal in frequency domain
subplot(2, 1, 1);
plot(linspace(0, samplingFrequency, length(ofdmSymbolsWithCP)), 20 * log10(abs(fft(ofdmSymbolsWithCP))));
title('Transmitted Signal in Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
 % Plot received signal in frequency domain
subplot(2, 1, 2);
plot(linspace(0, samplingFrequency, length(channelOutput)), 20 * log10(abs(fft(channelOutput))));title('Received Signal in Frequency
Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
```

EXPERIMENT NO-13

Perform OFDM modulation and obtain time domain and frequency domain plots to show a lowrate signal, a high-rate signal, and a frequency selective multipath channel response.

Orthogonal Frequency Division Multiplexing (OFDM) is a modulation technique widely used in modern communication systems, particularly in wireless communication and broadband access technologies. OFDM provides several advantages that make it suitable for high-data-rate communication in various environments. Here's a brief overview of OFDM modulation

- **Frequency Division Multiplexing (FDM):** Traditional FDM involves dividing the available frequency bandwidth into multiple non-overlapping subchannels, each assigned to a different user or signal.
- **Orthogonality:** OFDM takes advantage of the orthogonality between subchannels, meaning that the frequency spectra of the subchannels do not overlap, minimizing interference.

2. Subcarrier Modulation:

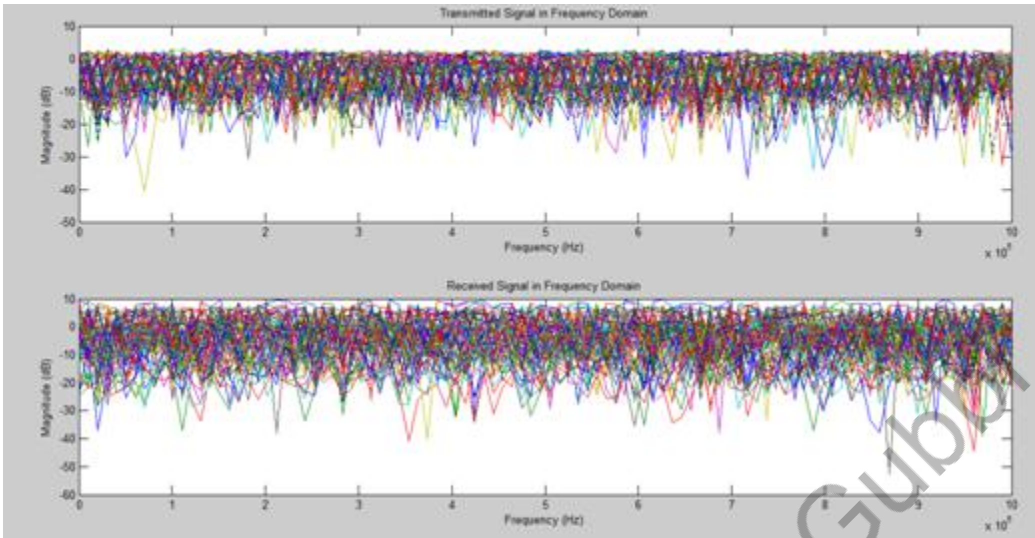
- OFDM divides the available bandwidth into multiple narrowband subcarriers.
- Each subcarrier is modulated independently, allowing simultaneous transmission of data on multiple subcarriers.

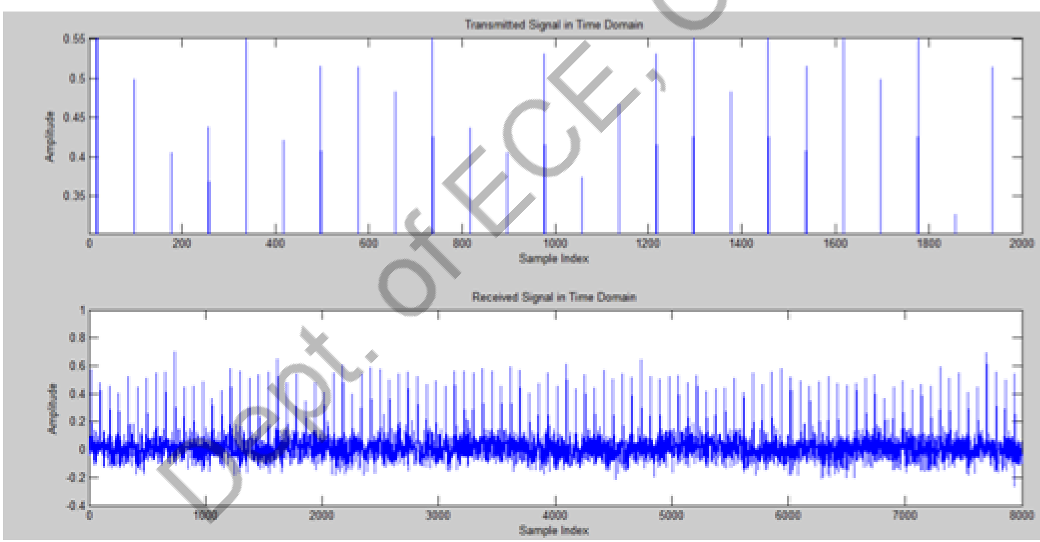
3. Orthogonal Subcarriers:

- The key feature of OFDM is the orthogonality between the subcarriers. This orthogonality is maintained by carefully selecting the frequency spacing between subcarriers.
- Orthogonality ensures that the subcarriers do not interfere with each other, even when transmitted simultaneously.

4. Advantages:

- **Resistance to Multipath Fading:** OFDM is robust against multipath fading, a phenomenon where signals take multiple paths to reach the receiver due to reflections and diffractions.
- **High Spectral Efficiency:** OFDM allows efficient utilization of available bandwidth, providing high data rates.
- **Tolerance to Frequency Selective Fading:** OFDM is suitable for channels with frequency-selective fading, as it can mitigate the impact by distributing data across multiple subcarriers.


Transmitted Signal in Frequency Domain
Magnitude (dB)
Frequency (Hz)
Received Signal in Frequency Domain
Magnitude (dB)
Frequency (Hz)


Transmitted Signal in Time Domain
Amplitude
Sample Index
Received Signal in Time Domain
Amplitude
Sample Index

Sample Viva questions

1. State the difference between Analog systems and digital systems.
2. Explain why digital systems are considered superior to Analog systems.
3. Mention the disadvantages of Analog communication.
4. Explain the basic steps involved in digitizing a signal.
5. Explain ASK operation.
6. Explain BPSK operation.
7. Explain DPSK operation.
8. State the difference between discrete and digital signals.
9. State the difference between pulse modulation and digital modulation.
10. Explain FSK circuit operation.
11. Explain different types of channels.
12. Mention the basic blocks of formatting and transmission of base band signals.
13. Mention the difference between broad band transmission and base band transmission.
14. Explain Gram-Schmidt Orthogonalization Procedure.
15. Explain M-ary transmission techniques
16. What is QAM?
17. What is 16-QAM?
18. What is Huffman Coding? Explain with an example.
19. What is Hamming Code? Explain with an example.
20. Explain different types of error detection and correction techniques.
21. What is convolution code? Explain with an example.