COURSE CONTENT

COMMUNICATION SYSTEMS LABORATORY								
IV Semester: ECE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
AECD15	**Core**	-	-	2	1.5	40	60	100
Contact Classes: Nil	**Tutorial Classes: Nil**	**Practical Classes: 45**				**Total Classes: 45**		
Prerequisite: There are no prerequisites to take this course								

I. COURSE OVERVIEW:

Communication engineering is the field of study concerned with the transmission of information either in analog or digital form. The objective of this lab course provides a platform to the students to understand the basics of analog and digital communication systems, modulation techniques, data transmission, multiplexing, etc

II. COURSES OBJECTIVES:

The students will try to learn

I. The basic theory of communication system in practice.
II. The concept of analog to digital conversion for pulse modulation techniques
III. The analog and digital modulation techniques using MATLAB tool.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

CO 1: Discriminate the generation and detection of amplitude modulated and frequency modulated signals to calculate the modulation index and frequency deviation.
CO 2: Analyze the analog pulse modulation and demodulation methods for transmitting the information by pulses
CO 3: Apply the concept of pulse code modulation and demodulation for encoded data in analog to digital conversion
CO 4: Select the time division or frequency division multiplexing techniques for transmitting multiple signals at a time in the communication system.
CO 5: Examine the digital modulation techniques for convey more information, high quality and security
CO 6: Choose appropriate techniques for signal processing and filtering in communication systems.

EXERCISES FOR COMMUNICATION SYSTEMS LABORATORY

1. Getting Started Exercises

To be proficient in Matlab programming, you need to be able to Communication Systems:

1. Overview of communication systems and their significance
2. Basic components and functions of a communication system

2. Exercises on Analog Modulation Techniques

To be skillful in Matlab programming, you need to be able to realize Modulation Techniques

1 Design an amplitude modulation (AM) system

2 Implement: Investigating depth of modulation

3 Design of AM-Double Side Band Suppressed Carrier (DSB-SC) signal using Balanced Modulator

4 Design of Single Side band Suppressed Carrier (SSBSC).

3. Exercises on Angle Modulation Techniques

To be proficient in programming, you need to be able to realize Angle Modulation Techniques:

1. Calculation of Frequency Deviation in Frequency Modulation (FM).
2. Determination of Modulation Index in Frequency Modulation (FM).
3. Create a Phase Modulation (PM) waveform using a programming language (like MATLAB) with a carrier frequency of and modulating signal frequency.
4. Phase Deviation Calculation in phase modulation.
5. Generate an FM waveform using MATLAB programming language with the given carrier frequency (e.g., 100 MHz), modulating signal frequency (e.g., 10 kHz), and modulation index (e.g., 2). Plot and visualize the FM waveform.

4. Exercises on Analog Pulse Modulation Techniques

To be proficient in programming, you need implement the following Analog Pulse Modulation Techniques

1. Demonstrate reconstruction of the original signal from the sampled PAM signal.
2. Calculation of Duty Cycle in Pulse Width Modulation (PWM)
3. Design a reconstruction circuit to convert the PWM signal back to its original analog form.
4. Calculate the pulse width and separation between pulses for a PPM signal.
5. Encode a simple analog signal into a PPM waveform

5. Exercises on Digital Pulse Modulation Techniques

To be proficient in programming, you need implement the following Digital Pulse Modulation Techniques

1. Design a PCM system with a specific number of quantization levels
2. Determine the minimum sampling rate required for PCM encoding of an analog signal
3. Implement a delta modulation system with a given step size
4. Analyze the performance of step size affects the in delta modulation system
5. Design an adaptive DPCM system

6. Exercises on Digital Modulation Techniques

To be proficient in programming, you need implement the following Digital Modulation

1. Implement: Generating an ASK signal
2. Implement: Generating an FSK signal
3. Implement: Generating an BPSK signal
4. Implement: Generating a QPSK signal

7. Exercises on Signal Processing in Communication Systems

To be proficient in programming, you need implement the signal processing

1. Implement: Spectrum of the impulse train
2. Implement: Spectrum of the filtered impulse train
3. Implement: Spectrum of pseudorandom sequence
4. Implement: Analog noise generation (AWGN)

8. Exercises on Signal filtering

To be proficient in programming, you need implement the filtering of signals

1. Basics of Filtering
2. Implement: Frequency domain analysis in signal filtering
3. Design of filter optimization in signal filtering

9. Exercises on Wireless Communication Systems

To be proficient in programming, you need implement the Wireless Communication Systems

1. Implement: Okumura model
2. Multipath fading

10 Exercises on Data transmission

To be proficient in programming, you need implement the Data transmission

1 Scrambler
2 Implement: Spread spectrum modulation

11. Exercises on Multiplexing

To be proficient in programming, you need implement the Data transmission

1 Implement: Time division multiplexing (TDM)
2 Implement: Frequency division multiplexing (FDM)

12. Exercises on Generation of Noises

This exercise aims to provide hands-on experience on generation of noises. The term is used, with this or similar meanings, in many scientific and technical disciplines, including physics, acoustic engineering, telecommunications, statistical forecasting, and many more.

1. Generation of White noise
2. Generation of Brown noise
3. Generation of Pink noise

13. Exercises on case study: Software-Defined Radio (SDR)

Implement: A software-defined radio (SDR) is a wireless device that typically consists of a configurable RF front end with an FPGA or programmable system-on-chip (SoC) to perform digital functions.

14. Exercises on case study: communication system project

Create a simple voice communication system using microcontrollers and wireless modules.

EXERCISES FOR COMMUNICATION SYSTEMS LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

To be proficient in programming, you need to be able to:

1.0 Overview of communication systems.

1.1 Introduction to EMONA Communications board

1.2 Introduction to MATLAB

1.3 Generation of various Signals and Sequence

1.4 Operation on signals and sequences

1.0 Overview of communication system

Communication is the process of establishing connection or link between two points for information exchange. The electronics equipments which are used for communication purpose are called communication equipments. Different communication equipments when assembled together form a communication system.

Typical example of communication system are line telephony and line telegraphy, radio telephony and radio telegraphy, radio broadcasting, point-to-point communication and mobile communication, computer communication, radar communication, television broadcasting, radio telemetry, radio aids to navigation, radio aids to aircraft landing etc.

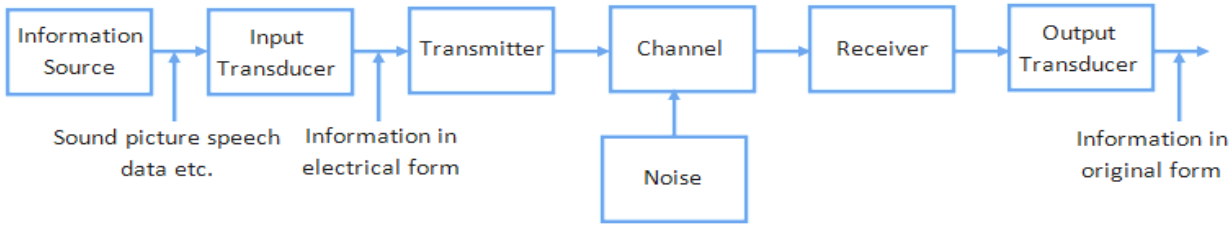


Figure 1.0 shows the block diagram of a general communication system, in which the different functional elements are represented by blocks.

1.1 Introduction to EMONA Communications board

Getting to know the NI ELVIS III platform

Required Tools and Technology

Platform: NI ELVIS III

Hardware: Emona Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Getting to know the EMONA Communications Board

The EMONA Communications Board(ECB) for the NI ELVIS III is used to help learn about communications and telecommunications principles. It lets you bring to life the block diagrams that fill communications textbooks. A "block diagram" is a simplified representation of a more complex circuit. An example is shown in Figure 1.1

Each block represents a part of the circuit that performs a separate task and is named according to what it does. Examples of common blocks in communications equipment include the adder, filter, phase shifter and so on. The ECB has a collection of blocks (called modules) that you can put together to implement dozens of communications and telecommunications block diagrams

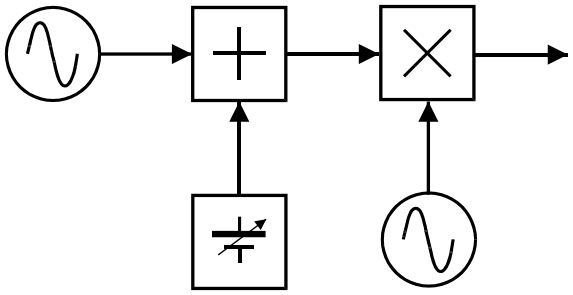
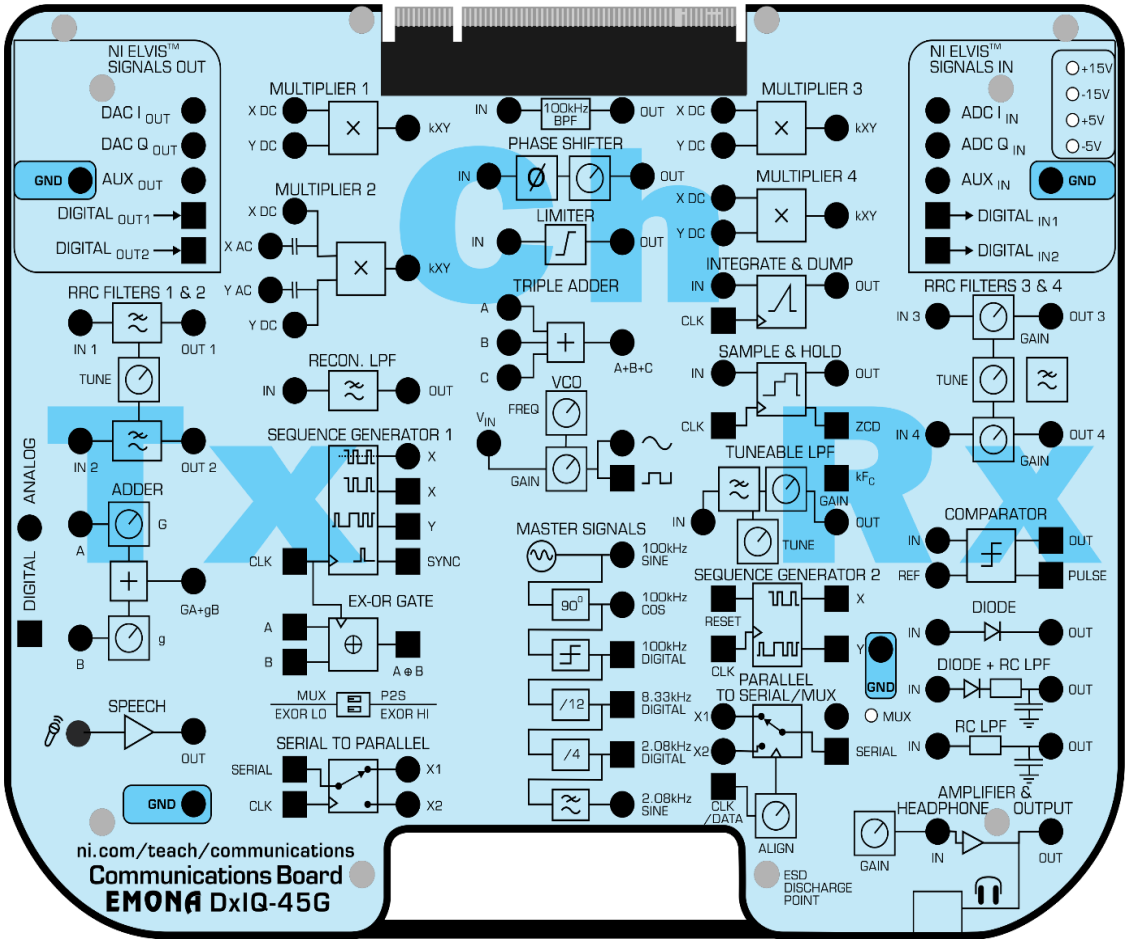


Figure 1.1: A sample communications block diagram



EMONA Communications Board overlay

1.2 Introduction to MATLAB

MATLAB is a high-level programming language and environment designed primarily for numerical computing, data analysis, and visualization. It is widely used in academia, research, and industry for tasks ranging from mathematical modeling to algorithm development and simulation.

Basic Syntax:

```
% MATLAB script example
A = [1 2; 3 4];      % Define a matrix
b = [5; 6];          % Define a vector
x = A \ b;           % Solve the system of linear equations
disp(x);             % Display the result
```

In summary, MATLAB is a versatile tool for scientific computing and engineering applications. Its strengths lie in its ease of use, rich set of functions, and powerful matrix-based operations, making it a preferred choice for professionals and researchers in various fields.

1.3 Generation of various Signals and Sequence

Generation of sinusoidal signal

1.3a : Implement: Generating Sinusoidal Signal
Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or ear buds

Powering up the ELVIS III + EMONA Communications Board

1. Connect the set-up shown in Figure 1.3.
2. Launch and run the NI ELVIS III Oscilloscope and set it up ensuring that the Trigger Source control is set to Channel 1

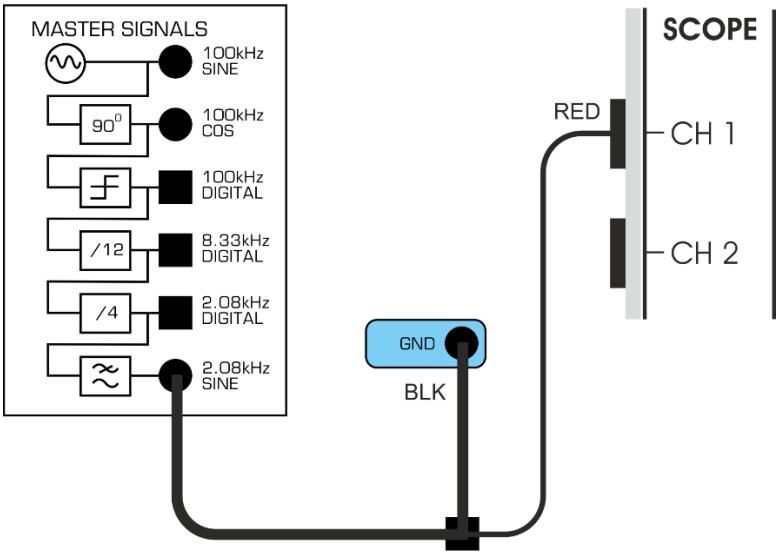


Figure 1.3: Viewing the 2.08kHz SINE signal

This set-up can be represented by the block diagram in Figure 1.3a

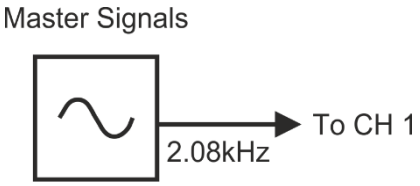


Figure1.3a : 2.08kHz sine wave block diagram

Generating a sinusoidal signal in MATLAB involves defining the time vector and specifying the amplitude, frequency, and phase of the sine wave.

Hint

```
% Parameters
A = 1;          % Amplitude
f = 2;          % Frequency (in Hz)
phi = pi/4;     % Phase (in radians)
% Time vector
t = 0:..:1;   % Time vector from 0 to 1 second with a step of 0.01 seconds
% Sinusoidal signal
x = A * sin(…….);
% Plotting the sinusoidal signal
plot(t, x);
title('Sinusoidal Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

Try

1 Generate a sinusoidal signal with the parameters amplitude 5 volts and frequency 1KHz . Plot the Sinusoidal signal

2 Generate a sinusoidal signal with the parameters amplitude 12 volts and time frequency 10KHz, Plot the Sinusoidal signal

1.3b -Generating Square wave

Generating a square wave in MATLAB involves defining the time vector and specifying the amplitude, frequency, and phase of the square wave.

Hint

```
% Parameters
amplitude = 1;    % Amplitude of the square wave signal
frequency = 2;    % Frequency of the square wave signal in Hz
duty Cycle = 0.5;  % Duty cycle of the square wave (percentage)
% Time vector
t = 0:…:2;      % Time vector from 0 to 2 seconds with a step of 0.01 seconds
% Square wave signal
Square Wave Signal = amplitude * square(……., duty Cycle);

% Plotting the square wave signal figure;
plot(t, square Wave Signal, 'b-', 'Line Width', 2);
title('. .. . .');
xlabel('Time (seconds)');
ylabel('Amplitude');
ylim([-…, …]);  % Set y-axis limits for better visualization
grid on;
```

Try

1. Generate a square signal with the parameters amplitude 2 volts , frequency 1.5KHz and duty cycle is 0.8 . Plot the Sinusoidal signal
2. Generate a square signal with the parameters amplitude 5 volts , frequency 500Hz and duty cycle is 0.5 . Plot the Sinusoidal signal

1.3c -Generating Unit step signal

Generating a square wave in MATLAB involves defining the time vector and specifying the amplitude, frequency, and phase of the square wave.

Hint

```
% Parameters
% Time vector
t = ……..;   % Time vector from -5 to 5 seconds with a step of 0.01 seconds
```

```
% Unit step function
Unit Step Function = (t >= 0);
% Plotting the unit step function
figure;

xlabel('Time');
ylabel('Amplitude');
ylim([-……]);  % Set y-axis limits for better visualization
grid on;
```

Try

1. Generate a unit step function with the parameter t=-10 to 10 seconds with a step of 0.05 seconds. Plot the unit step function.
2. Generate a unit step function with the parameter t=-8 to 8 seconds with a step of 0.05 seconds . Plot the unit step function.

1.3d-Generating Exponential signal

Generating a square wave in MATLAB involves defining the time vector and specifying the amplitude, frequency, and phase of the square wave.

Hint

```
% Parameters
% Generate y values using the exponential function
y = exp(x);
% Plot the exponential function
plot(x, y);

xlabel('x');
ylabel('y');
grid on;
```

Try

1. Generate a exponential function with the x values range from -5 to +5. Plot the exponential function.
2. Generate a exponential function with the x values range from -3 to +3. Plot the exponential function.

2. Exercises on Analog Modulation Techniques

To be proficient in programming, you need to be able to:

1	Design an amplitude modulation (AM) system
2	Implement: Investigating depth of modulation

3 Design of AM-Double Side Band Suppressed Carrier (DSB-SC) signal using Balanced Modulator

4 Design of Single Side band Suppressed Carrier (SSBSC).

2.1 Design an amplitude modulation (AM) system

Realize the amplitude modulation (AM) system

In the early days of wireless, communication was carried out by telegraphy, the radiated signal being an interrupted radio wave. Later, the amplitude of this wave was varied in sympathy with (modulated by) a speech message (rather than on/off by a telegraph key), and the message was recovered from the envelope of the received signal. The radio wave was called a 'carrier', since it was seen to carry the speech information with it. The process and the signal were called amplitude modulation, or 'AM' for short.

Required Tools and Technology

Platform: NI ELVIS III

Hardware: Emona Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Generating Amplitude modulation (AM)

Powering up the ELVIS III + EMONA Communications Board

1. Ensure that the NI ELVIS III Application Board power button at the top left corner of the unit is OFF (not illuminated).
2. Connect the set-up shown in Figure 2.3.

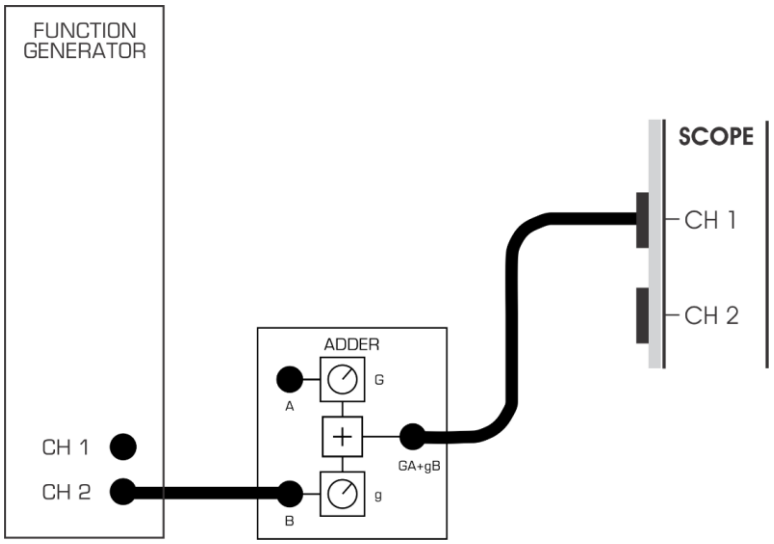


Figure 2.3: Create DC voltage

8. Locate the Adder module on the board and turn its *G* control fully anti-clockwise.

9. Connect the set-up shown in Figure 4.
Note: Insert the black plugs of the oscilloscope lead into a ground (*GND*) socket.

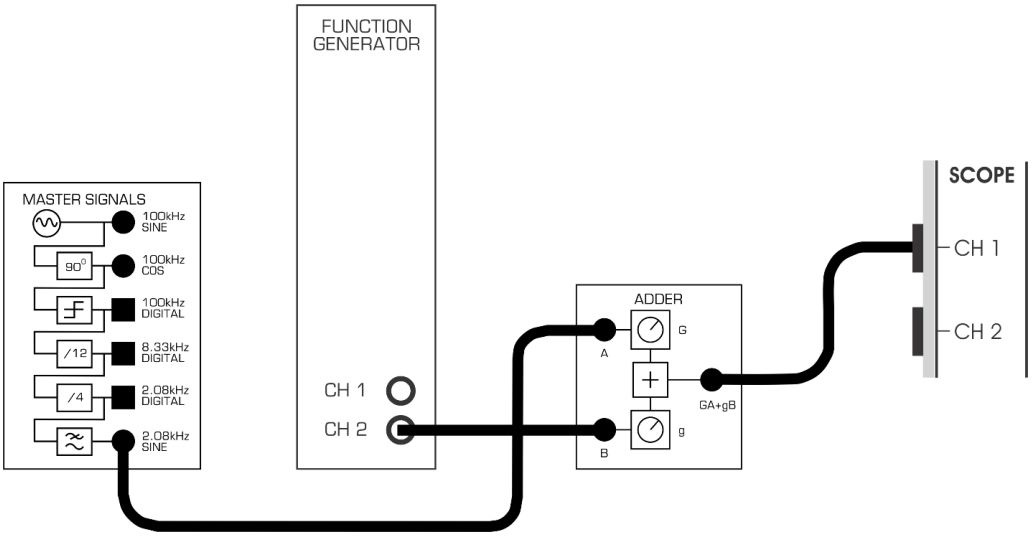


Figure 2.4: Add DC to the message

This set-up can be represented by the block diagram in Figure 5. It implements the highlighted part of the equation: AM = (DC + message) × the carrier.

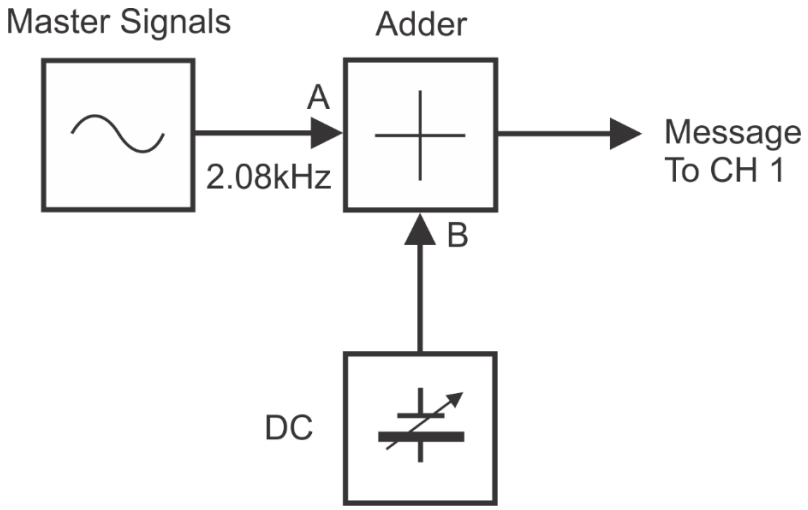


Figure 5: Block diagram for addition

Modify the set-up as shown in Figure 2.6.

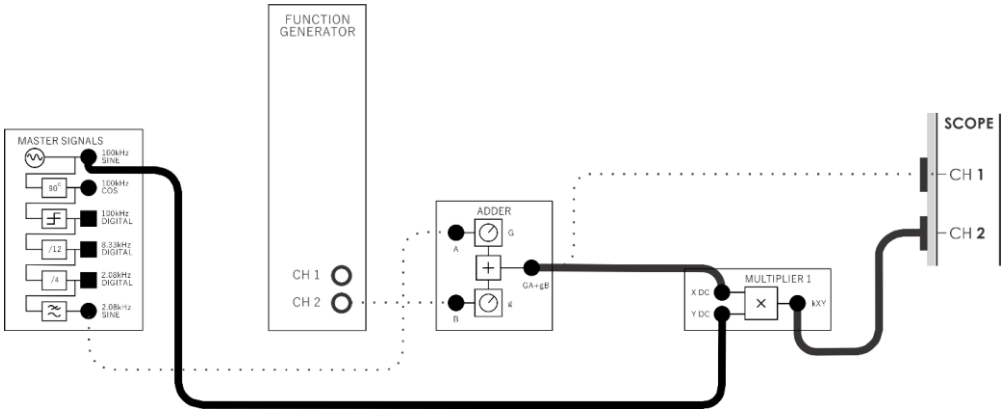


Figure 2.6: Multiply the baseband message by the carrier

This set-up can be represented by the block diagram in Figure 7. The additions that you've made to the original set-up implement the highlighted part of the equation:

AM = (DC + message) × the carrier.

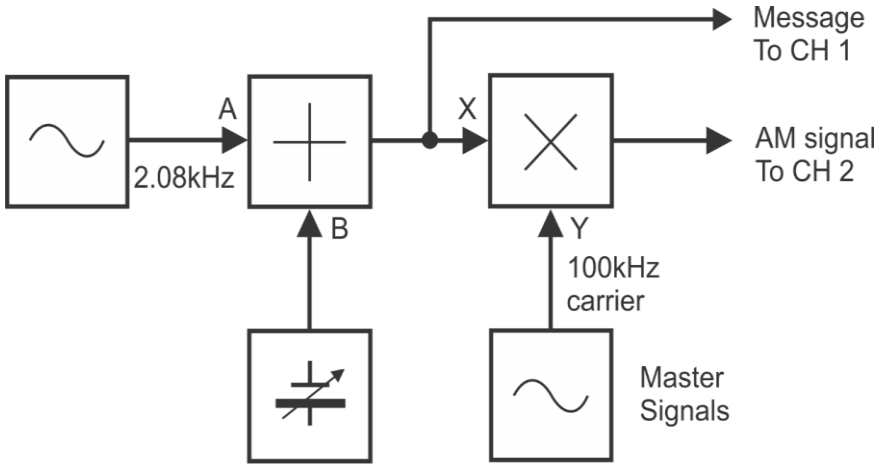


Figure 2.7: Block diagram for AM

With values, the equation on the previous page becomes:

AM = (1VDC + 1Vp-p 2kHz sine) × 4Vp-p 100kHz sine.

13. Adjust the scope's Timebase control to view only two or so cycles of the message signal i.e.: 100us/div or even 50us/div for one cycle. Change the Volts per division control for Channel 2 to 2 V.

Hint

```
% Parameters
carrierFreq = . . . . . .; % Carrier frequency (1 MHz)
modulatingFreq = . . . . . .; % Modulating signal frequency (20 kHz)
time = 0:1/1000000:0.1; % Time vector (0.1 seconds)

% Generating Carrier Signal
carrierSignal = cos( ... ... );

% Generating Modulating Signal (Simple sine wave)
```

```
modulatingSignal = 0.5*sin(. . . . .);

% Plotting Carrier and Modulating Signals
subplot(.,.,.);
plot(time, carrierSignal);
title('Carrier Signal');
xlabel('. . . .');
ylabel('. . . . .');

subplot(.,.,.);
plot(time, modulatingSignal);
title('Modulating Signal');
xlabel('. . . . . .');
ylabel('. . . . . .');

% Amplitude Modulation
modulatedSignal = (1 + . . . . . .) .* carrierSignal;

% Plotting the Modulated Signal
figure;
plot(time, modulatedSignal);
title('Modulated Signal (AM)');
xlabel('. . . . . ');
ylabel('Amplitude');

% Demodulation (Envelope Detection)
envelope = abs(hilbert(modulatedSignal));

% Low-pass Filter (To retrieve the original modulating signal)
fc = 50000; % Cut-off frequency for the low-pass filter
[b, a] = butter(6, fc/(carrierFreq/2), 'low');
recoveredSignal = filter(b, a, envelope);

% Plotting the Recovered Signal
figure;
plot(time, recoveredSignal);
title('Recovered Modulating Signal');
xlabel('. . . . . .');
ylabel('. . . . . .');
```

Try

1. An AM signal, depth of modulation 100%.from a single tone message, has peak-to-peak amplitude of 4 volts. What would an RMS voltmeter read if connected to this signal?
2. Given an AM signal with a carrier amplitude of 10 volts and a modulated signal with a peak amplitude of 14 volts and a trough amplitude of 6 volts: Calculate the modulation depth and Determine the percentage modulation.

2.2 Implement: Investigating depth of modulation

Realize setting and measuring the depth of modulation

The depth of modulation in an amplitude modulation (AM) signal refers to the extent to which the carrier wave's amplitude is varied by the modulating signal. It's often expressed as a percentage and indicates the strength or intensity of the modulating signal in affecting the carrier wave.

Hint

```
% Parameters
carrierFreq = 1000; % Carrier frequency (Hz)
modulatingFreq = 100; % Modulating signal frequency (Hz)
time = 0:0.001:1; % Time vector (1 second)

% Carrier Signal
carrierSignal = cos(. . . . . .);

% Modulating Signal with adjustable amplitude
modulatingAmplitude = 0.5; % Adjust this value to set modulation depth
modulatingSignal = modulatingAmplitude * sin(. . . . . .);

% Modulated Signal (AM)
modulatedSignal = (1 + . . . . . .) .* carrierSignal;

% Plotting Modulating and Modulated Signals
figure;
subplot(.,.,.);
plot(time, modulatingSignal);
title('Modulating Signal');
xlabel('Time');
ylabel('Amplitude');

subplot(.,.,.);
plot(time, modulatedSignal);
title('Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
% Calculate Peak-to-Peak Amplitudes
Vc = max(. . . . . .) - min(. . . . . .); % Carrier amplitude
```

```
Vm = max(. . . . . ) - min(. . . . . ); % Modulated signal amplitude
Vt = max(. . . . . ) - mean(. . . . . .); % Trough amplitude

% Calculate Depth of Modulation (%)
depthOfModulation = ((Vc - Vt) / (Vc + Vt)) * 100;
disp(['Depth of Modulation: ', num2str(. . . . .), '%']);
% Calculate RMS Voltages
RMS_Vc = rms(. . . . . ); % RMS of carrier signal
RMS_Vm = rms(. . . . . ); % RMS of modulated signal

% Calculate Depth of Modulation (%)
depthOfModulation_RMS = ((RMS_Vc - RMS_Vm) / RMS_Vc) * 100;
disp(['Depth of Modulation (RMS): ', num2str(depthOfModulation_RMS), '%']);
```

Try

1 For an AM signal with a carrier signal of 50 volts and a modulating signal with a peak amplitude of 20 volts: Calculate the modulation index and determine if this modulation index indicates over-modulation, under-modulation, or 100% modulation.
2 Given a carrier signal with an RMS voltage of 20 volts and a modulation index of 0.5: Calculate the total power in the modulated signal. Compare it with the total power of the carrier signal alone.

2.3 Design of AM-Double Side Band Suppressed Carrier (DSB-SC) signal using Balanced Modulator

Realize AM-Double Side Band Suppressed Carrier (DSB-SC) signal using Balanced Modulator

A balanced modulator is essentially multiplying the message signal with the carrier signal. The resulting signal represents the Double Side Band Suppressed Carrier (DSB-SC) modulation where both sidebands are present, but the carrier component is suppressed.

Hint

```
% Define parameters
fs = 1000; % Sampling frequency
t = 0:1/fs:1; % Time vector (1 second duration)

% Generate message signal (e.g., a sine wave)
messageSignal = 2 * sin(. . . . . ); % 5 Hz sine wave (adjust frequency as needed)

% Generate carrier signal (e.g., a cosine wave)
carrierFrequency = 100; % Carrier frequency (adjust as needed)
carrierSignal = cos(. . . . . .);
```

```
% Plot message and carrier signals
figure;
subplot(.,.,.);
plot(t, messageSignal);
title('Message Signal');
xlabel('Time');
ylabel('Amplitude');

subplot(.,.,.);
plot(t, carrierSignal);
title('Carrier Signal');
xlabel('Time');
ylabel('Amplitude');

% Balanced Modulator (Multiplication)
DSB_SC_signal = messageSignal .* . . . . . .;

% Plot DSB-SC Modulated Signal
figure;
plot(t, . . . .);
title('DSB-SC Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
```

Try

1. Generate a DSB-SC signal using MATLAB. Assume a message signal of a 3 kHz sine wave and a carrier frequency of 50 kHz. Plot the DSB-SC modulated signal.
2. Design a circuit using a balanced modulator to generate a DSB-SC signal with a message signal of 2 kHz and a carrier frequency of 10 kHz. Assume an ideal balanced modulator and an input message signal amplitude of 1V.

2.4 Design of Single Side band Suppressed Carrier (SSBSC)

To design of Single Side band Suppressed Carrier (SSBSC).

Creating a Single Side Band Suppressed Carrier (SSB-SC) signal involves generating a modulating signal, generating a carrier signal, performing modulation, and then filtering to suppress one sideband and the carrier while preserving the desired sideband.

Hints:

```
% Define parameters
fs = 10000; % Sampling frequency
t = 0:1/fs:0.1; % Time vector (100 ms duration)

% Generate message signal (e.g., a sine wave)
messageSignal = 2 * sin(. . . . . .); % 50 Hz sine wave (adjust frequency as
needed)

% Generate carrier signal (e.g., a cosine wave)
carrierFrequency = 1000; % Carrier frequency (adjust as needed)
carrierSignal = cos(. . . . . .);

% Plot message and carrier signals
figure;
subplot(.,.,.);
plot(t, messageSignal);
title('Message Signal');
xlabel('Time');
ylabel('Amplitude');

subplot(.,.,.);
plot(t, carrierSignal);
title('Carrier Signal');
xlabel('Time');
ylabel('Amplitude');

% Hilbert Transform to obtain the analytic signal
analyticSignal = hilbert(. . . . . .);


% Generate the SSB-SC signal by suppressing the unwanted sideband
SSB_SC_signal = real(analyticSignal .* exp(1i*2*pi*. . . . .*t));


% Plot SSB-SC Modulated Signal
figure;
plot(t, SSB_SC_signal);
title('SSB-SC Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
```

Try:

1. Generate an SSB-SC signal using MATLAB. Assume a message signal of a 4 kHz sine wave and a carrier frequency of 50 kHz. Plot the SSB-SC modulated signal.
2. Design a circuit to generate an SSB-SC signal with a message signal of 4 kHz and a carrier frequency of 20 kHz using the phasing method. Assume an ideal multiplier and phase shifter and an input message signal amplitude of 2V.

3. Exercises on Angle Modulation Techniques

To be proficient in programming, you need implement the following Angle Modulation Techniques:

1. Implement: Generating an FM signal using speech
2. Calculation of Frequency Deviation in Frequency Modulation (FM).
3. Determination of Modulation Index in Frequency Modulation (FM).
4. Create a Phase Modulation (PM) waveform using a programming language (like MATLAB) with a carrier frequency of and modulating signal frequency.
5. Phase Deviation Calculation in phase modulation.
6. Generate an FM waveform using MATLAB programming language with the given carrier frequency (e.g., 100 MHz), modulating signal frequency (e.g., 10 kHz), and modulation index (e.g., 2). Plot and visualize the FM waveform.

3.1 Implement: Generating an FM signal using speech

Required Tools and Technology

Platform: NI ELVIS III
Instruments used in this lab:

- Oscilloscope-Time
- Oscilloscope-FFT
- Function Generator

Hardware: Emona Communications Board
Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or ear-buds.

.

1. Connect them to the Speech module's output as shown in Figure 3.1.5.

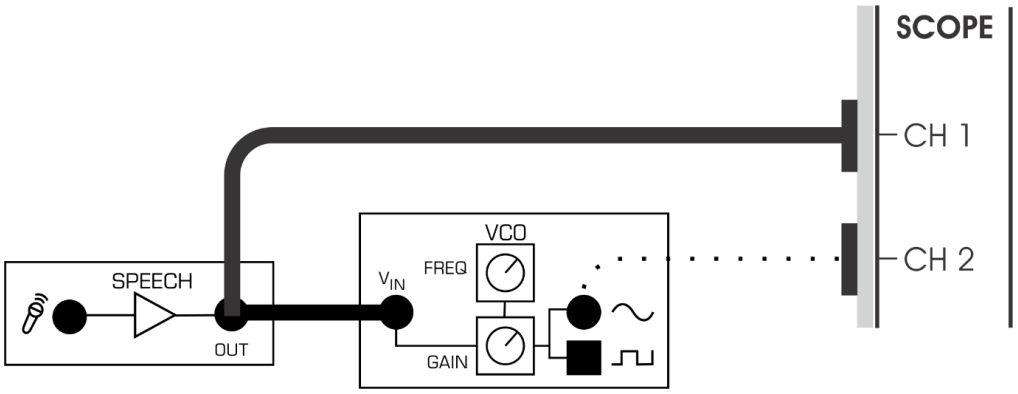


Figure 3.1. 5: Modulating the VCO with speech

2. Set the scope's Time base control to the 100µs/div position.
3. Hum, clap and talk into the microphone while watching the scope's display.

3.2 Calculation of Frequency Deviation in Frequency Modulation (FM)

In Frequency Modulation (FM), frequency deviation refers to the maximum change in frequency from the carrier frequency due to modulation by the information-bearing signal. This deviation is related to the amplitude of the modulating signal and the modulation index.

Hint

```
// Calculation of Frequency Deviation in Frequency Modulation (FM)

% Define carrier and modulating signal parameters
carrier_freq = 1000; % Carrier frequency in Hz
modulating_freq = 200; % Modulating signal frequency in Hz
A_mod = 2; % Peak amplitude of the modulating signal
A_carr = 1; % Peak amplitude of the carrier signal

% Calculate the modulation index


% Given maximum frequency deviation per unit amplitude (in Hz/V)
kf = 5e3; % 5 kHz/V

% Calculate frequency deviation


% Display results

fprintf('Modulation Index (m): %.2f\n', m);
fprintf('Frequency Deviation: %.2f Hz\n', frequency_deviation);
```

Try

1. Consider a carrier signal with a frequency of 1 MHz and a modulating signal with a frequency of 100 kHz. The peak amplitude of the modulating signal is 0.5 Volts. Assume the maximum frequency deviation per unit amplitude (*kf*) is 10 kHz/V. Calculate the modulation index and the frequency deviation.
2. Design an FM system with a modulation index of 2.5 and a maximum modulating frequency of 10 kHz. Determine the frequency deviation and the bandwidth occupied by the FM signal.

3.3 Determination of Modulation Index in Frequency Modulation (FM)

The modulation index in Frequency Modulation (FM) measures the extent of frequency deviation caused by the modulating signal relative to the carrier frequency. It's a crucial parameter that helps define the bandwidth and characteristics of an FM signal.

Hints

```
//Write the program for Determination of Modulation Index in Frequency Modulation (FM)

% Given parameters
kf = 15e3; % Maximum frequency deviation per unit amplitude in Hz/V
delta_f = 30e3; % Frequency deviation caused by the modulating signal in Hz

% Calculate modulation index

% Display result
fprintf('Modulation Index (m): %.2f\n', m);
```

Try

1. Given a carrier signal with a frequency of 100 MHz and a modulating signal with a frequency of 10 kHz, calculate the modulation index for the frequency modulation.
2. In an FM radio transmission, the maximum frequency deviation is 75 kHz when modulated by an audio signal with a frequency range of 20 Hz to 15 kHz. Determine the modulation index for this FM system.

3.4 Create a Phase Modulation (PM) waveform using a programming language (like MATLAB) with a carrier frequency of and modulating signal frequency

This MATLAB code generates a sine wave as the modulating signal and then generates a phase modulated signal by modulating the phase of a cosine carrier wave with the modulating signal.

Adjust the carrier_freq, modulating_freq, and modulation_index variables to see how they affect the resulting phase modulated signal. The modulation_index determines the extent of phase deviation caused by the modulating signal. This code will plot both the modulating signal and the resulting phase modulated signal in separate plots to visualize the modulation effect.

Hints

```
/**     Implementation of Phase Modulation **/

% Define parameters
carrier_freq = 1000; % Carrier frequency in Hz
modulating_freq = 50; % Modulating signal frequency in Hz
modulation_index = 1; % Modulation index (change as needed)

% Time specifications
duration = 1; % Duration of the signal in seconds
sampling_freq = 10 * carrier_freq; % Sampling frequency

% Time vector
t = linspace(. . . . . .);

% Generate the modulating signal (sine wave)
modulating_signal = sin(. . . .. );
```

```
% Generate the phase modulated signal
phase_modulated_signal = cos(2 * pi * carrier_freq * t + modulation_index *
modulating_signal);

% Plot the modulating signal

% Plot the phase modulated signal

% Display the plots
sgtitle('Phase Modulation');

% Demodulate the phase-modulated signal (using simple differentiation)
demodulated_signal = diff(phase_modulated_signal) / (2 * pi * carrier_freq /
sampling_freq);

% Adjust the length of the demodulated signal
demodulated_signal = [demodulated_signal, demodulated_signal(end)];

% Plot the demodulated signal
figure;

xlabel('Time');
ylabel('Amplitude');
```

Try

1. Generate a Phase Modulation (PM) signal let's consider a carrier frequency of 1000 Hz and a modulating signal frequency of 100 Hz.
2. For a phase modulation system, the phase deviation is 45° when modulated by a sinusoidal signal with a frequency of 5 kHz. Calculate the modulation index for this PM system.

3.5 Phase Deviation Calculation in phase modulation

The modulation index (m) is calculated assuming peak amplitudes of 1 for both the carrier and modulating signals (you might need to adjust this based on your actual signal amplitudes). The maximum phase deviation (max_phase_deviation) is calculated using the modulation index and the maximum phase shift possible, which is π radians for full-scale modulation.

You can then use the max_phase_deviation value in further calculations or simulations involving phase modulation. Adjust the carrier and modulating signal parameters as needed for your specific scenario.

Hints

```
/**
  Calculation of Phase Deviation.
 **/

% Define carrier and modulating signal parameters
carrier_freq = 1000; % Carrier frequency in Hz

modulation_index = 1; % Modulation index
```

```
% Calculate the modulation index (assuming peak amplitudes of 1 for simplicity)
A_mod = 1; % Peak amplitude of the modulating signal

m = A_mod / A_carr; % Modulation index

% Calculate maximum phase deviation
max_phase_deviation = m * pi; % Maximum phase deviation in radians
```

Try

1 In a phase modulation system, a sinusoidal modulating signal with a frequency of 10 kHz is used to modulate a carrier. If the modulation index is 0.4, calculate the phase deviation produced by this modulation.
2 For a phase modulation system, a modulating signal with a frequency of 2 kHz causes a phase deviation of 30°. Calculate the modulation index for this PM system.

3.6 Generate an FM waveform using MATLAB programming language with the given carrier frequency (e.g., 100 MHz), modulating signal frequency (e.g., 10 kHz), and modulation index (e.g., 2). Plot and visualize the FM waveform.

```
% Define parameters
carrier_freq = . . . . . ; % Carrier frequency in Hz (e.g., 100 MHz)
modulating_freq = . . . . . ; % Modulating signal frequency in Hz (e.g., 10 kHz)
modulation_index = 2; % Modulation index

% Time parameters
sampling_freq = . . * carrier_freq; % Sampling frequency (higher than Nyquist)
time = 0:1/sampling_freq:0.01; % Time vector (0 to 0.01 seconds)

% Generate modulating signal
modulating_signal = sin(. . . . . .);

% Generate FM waveform
fm_waveform = cos(. . . . . .);

% Plot FM waveform
```

Try

1 Assume you have a carrier signal with a frequency of 1 kHz and a modulating signal with a frequency of 200 Hz. The peak amplitude of both signals is 1 Volt. Calculate the modulation index and the maximum phase deviation.
2 For an FM system, the modulation index is 22 and the modulating signal has a frequency of 5 kHz5 kHz. Determine the maximum frequency deviation and the bandwidth of the FM signal.

4. Exercises on Analog Pulse Modulation Techniques

To be proficient in programming, you need implement the following Analog Pulse Modulation Techniques

1. Calculation of Duty Cycle in Pulse Width Modulation (PWM)
2. Design a reconstruction circuit to convert the PWM signal back to its original analog form.
3. Calculate the pulse width and separation between pulses for a PPM signal.
4. Encode a simple analog signal into a PPM waveform

4.1 Calculation of Duty cycle in Pulse width modulation (PWM)

In Pulse Width Modulation (PWM), the duty cycle refers to the ratio of the time a signal is in the high state (or "on" state) to the total time of its period. It's commonly expressed as a percentage.

The formula to calculate the duty cycle in PWM is:

Duty Cycle = High Time / Total Time ×100%

Where:

High Time: The duration the signal is in the high state (on time).

Total Time: The total period of the signal (sum of the high and low times).

Hints

```
/** Calculation of Duty cycle **/

% High time and total time values (in microseconds)
high_time = 25; % Example high time (in microseconds)
total_time = 100; % Example total time (in microseconds)

% Calculate duty cycle
duty_cycle = (. . . . . / . . . . .) * 100;
fprintf('The duty cycle is: %.2f%%\n', duty_cycle);
```

Try

1. Given a PWM signal with a period of 50 milliseconds and a high time (on time) of 10 milliseconds, calculate the duty cycle.
2. A PWM signal has a period of 20 microseconds and a duty cycle of 40%. Calculate the high time and low time of the signal.

4.2 Design a reconstruction circuit to convert the PWM signal back to its original analog form

Reconstructing a PWM signal back to its original analog form typically involves using a low-pass filter. The low-pass filter smooths out the rapid switching in the PWM signal, resulting in an analog voltage that represents the average value of the PWM waveform.

The RC low-pass filter averages out the PWM signal by charging and discharging the capacitor through the resistor. The time constant of the RC circuit (determined by R and C) affects the cutoff frequency of the filter, determining how much of the high-frequency PWM signal is filtered out. As a result, the output across the capacitor will approximate the average voltage of the PWM waveform.

Hints

```
/**
 Realize a reconstruction circuit to convert the PWM signal back to its original
analog form   **/

% Define parameters
sampling_freq = 100e3; % Sampling frequency (Hz)
t = 0:1/sampling_freq:1; % Time vector (1 second)

% Generate a PWM signal (example)
pwm_signal = square(. . . . . .); % 50 Hz PWM signal with 25% duty cycle

% Reconstruction using an RC low-pass filter
R = 1000; % Resistor value (ohms)
C = 1e-6; % Capacitor value (farads)

% Simulate the low-pass filter response
fc = 1 / (. . . . .); % Cutoff frequency
[num, den] = . . . . .(1, fc / (. . . . .  / 2)); % Butterworth filter design
analog_signal = filter(num, den, pwm_signal);

% Plot the signals

subplot(2,1,2);
```

Try

1. Given a PWM signal with a frequency of 1 kHz and a duty cycle of 70%, determine the pulse width and the period of the PWM signal.
2. A PWM signal has a frequency of 1.5 kHz and a duty cycle of 40%. Calculate the pulse width and the period of this PWM signal.

4.3 Calculate the pulse width and separation between pulses for a PPM signal

In Pulse Position Modulation (PPM), pulse width and separation between pulses are fundamental parameters that define the characteristics of the signal. The pulse width is the duration of the individual pulses, and the separation is the time between the leading edges of consecutive pulses.

For PPM: Pulse Width (PW): This is the duration of each pulse in the signal.

Separation Between Pulses (SP): This refers to the time interval between the leading edges of consecutive pulses.

The formulas to calculate Pulse Width and Separation Between Pulses in PPM are generally defined as:

Pulse Width (PW) = (Modulation Index) * (Time for One Symbol)

Separation Between Pulses (SP) = (Time for One Symbol) - Pulse Width

Hints

```
% Given parameters
modulation_index = 0.4; % Modulation index
time_one_symbol = 10e-3; % Time for one symbol (10 milliseconds)

% Calculate pulse width and separation between pulses
pulse_width = . . . . . ;
separation_between_pulses = . . . . . .;

% Display the results
fprintf(. . . . . );
fprintf(. . . . . );
```

Try

1. Given a PPM signal with a modulation index of 0.6 and a symbol time of 5 milliseconds, calculate the pulse width and separation between pulses.
2. For a PPM signal with a modulation index of 0.4 and a symbol time of 8 microseconds, determine the pulse width and separation between pulses.

4.4 Encode a simple analog signal into a PPM waveform

To encode an analog signal into a Pulse Position Modulation (PPM) waveform, you'll need to sample the analog signal, quantize it, and then represent each sample as a pulse position within a symbol period.

This MATLAB code generates a simple analog signal (a sine wave in this case), quantizes it to four levels (using 4-bit quantization), and encodes it into a PPM signal. The PPM signal represents the quantized samples by the positions of pulses within the symbol duration.

Adjust the parameters, such as the analog signal characteristics, quantization levels, and symbol duration, to explore how changes impact the encoding process and the resulting PPM signal.

Hints

```
// Encode a simple analog signal into a PPM waveform

% Define parameters
sampling_freq = 1000; % Sampling frequency (Hz)
t = 0:1/sampling_freq:1; % Time vector (1 second)

% Generate a simple analog signal (e.g., a sine wave)
analog_signal = sin(. . . . . .); % 5 Hz sine wave (as an example)

% Quantization (assuming 4-bit quantization for simplicity)
quantized_signal = round((. . . . . .) * 7 / 2); % Quantize to 4 levels (0, 1, 2,
3)

% PPM encoding
symbol_duration = 0.1; % Duration for each symbol (0.1 second)
ppm_signal = zeros(1, length(t)); % Initialize PPM signal

for i = 1:length(t)
    sample_index = round(t(i) / symbol_duration) + 1; % Calculate the sample index
within symbols
    if sample_index > length(quantized_signal)
        sample_index = length(quantized_signal);
    end
    ppm_signal(i) = quantized_signal(sample_index);
end

% Plot the original analog signal and the PPM signal
```

5. Exercises on Digital Pulse Modulation Techniques

To be proficient in programming, you need implement the Digital Pulse Modulation Techniques

1. Design a PCM system with a specific number of quantization levels
2. Determine the minimum sampling rate required for PCM encoding of an analog signal
3. Implement a delta modulation system with a given step size
4. Analyze the performance of step size affects the in delta modulation system
5. Design an adaptive DPCM system

5.1 Design a PCM system with a specific number of quantization levels

PCM (Pulse Code Modulation) is a digital modulation technique used to represent analog signals by sampling the amplitude of the signal at regular intervals and quantizing each sample into a digital code. The number of quantization levels determines the resolution and accuracy of the representation of the analog signal.

Hints

```
// Design a PCM system
% Parameters
Fs = 1000;                  % Sampling frequency (Hz)
                            % Time vector for 1 second
f = 5;                      % Frequency of the sine wave (Hz)
A = 5;                      % Amplitude of the sine wave
                            % Number of quantization levels

% Create a sine wave signal
x = A * sin(. . . . .);

% Calculate the quantization step size
max_amplitude = max(abs(x));
delta = max_amplitude * 2 / n_levels;

% Quantize the signal

% Plot the original and quantized signals
subplot(2,1,1);
plot(t, x);
title('Original Signal');
xlabel('Time');
ylabel('Amplitude');

subplot(2,1,2);
plot(t, quantized_signal);
title('Quantized Signal');
xlabel('Time');
```

```
ylabel('Amplitude');
```

Try

1. Create a PCM system for an input signal that needs to be quantized with a variable number of quantization levels based on user input.
2. Calculate and display the bit rate required for different numbers of quantization levels.

5.2 Determine the minimum sampling rate required for PCM encoding of an analog signal

The minimum sampling rate required for PCM encoding of an analog signal depends on the Nyquist theorem. The theorem states that to accurately reconstruct an analog signal from its samples, the sampling frequency should be at least twice the highest frequency present in the analog signal.

Formula for Nyquist Theorem:

Minimum Sampling Rate = 2 × Maximum Frequency in the Signal

Hints

```
//Write Matlab code for minimum sampling rate required for PCM encoding of an analog signal

% Maximum frequency in the analog signal
max_frequency = 4000; % in Hz

% Calculate the minimum sampling rate
min_sampling_rate = 2 * max_frequency;
fprintf('The minimum sampling rate required is %d Hz.\n', min_sampling_rate);
```

Replace max_frequency with the maximum frequency component present in your analog signal. Running this MATLAB code will calculate the minimum sampling rate required for PCM encoding based on the Nyquist theorem.

Try

1. Determining Minimum Sampling Rate from Maximum Frequency
2. Demonstrates the Nyquist criterion by plotting the spectrum of a sine wave and showcasing the Nyquist frequency.

5.3 Implement a delta modulation system with a given step size

Delta modulation is a form of analog-to-digital signal conversion where the difference (delta) between successive samples of the analog signal is encoded and transmitted as a single bit.

```
//Write Matlab code for a delta modulation system

% Parameters
Fs = 1000;                      % Sampling frequency (Hz)
t = 0:1/Fs:1-1/Fs;              % Time vector for 1 second
f = 5;                          % Frequency of the analog signal (Hz)
A = 5;                          % Amplitude of the analog signal
step_size = 0.5;                % Delta modulation step size

% Generate an analog signal (sine wave)
x = A * sin(. . . . .);

% Initialize variables for delta modulation

% Delta modulation encoding
for i = 1:length(x)
    if x(i) > prev_code
```

```
        delta_modulated_signal(i) = 1;
    else
        delta_modulated_signal(i) = 0;
    end
    % Update previous code using the delta and step size
    prev_code = prev_code + step_size * (2 * delta_modulated_signal(i) - 1);
end

% Plot the original and delta modulated signals

subplot(2,1,2);
stairs(t, delta_modulated_signal);
title('Delta Modulated Signal');
xlabel('Time');
ylabel('Amplitude');
```

Try

1. Implementing a delta modulation system with a specified step size and analyzing its performance.
2. Develop a method to optimize the selection of the step size in delta modulation for a given signal. Consider factors such as signal characteristics, quantization noise, and distortion to determine an optimal step size dynamically.

5.4 Analyze the performance of step size effects in the Delta modulation system

The step size in a delta modulation system plays a crucial role in determining the performance of the system. The step size affects the accuracy of the encoded signal and influences the signal-to-noise ratio (SNR) of the reconstructed signal.

```
//Write Matlab code for analyze the performance of different step sizes in
a delta modulation system

% Parameters
Fs = 1000;                      % Sampling frequency (Hz)
t = 0:1/Fs:1-1/Fs;              % Time vector for 1 second
f = 5;                          % Frequency of the analog signal (Hz)
A = 5;                          % Amplitude of the analog signal
                                % Different delta modulation step sizes
```

```
% Generate an analog signal (sine wave)
% Initialize variables for delta modulation
% Perform delta modulation for different step sizes
for s = 1:length(step_sizes)
    step_size = step_sizes(s);
    prev_code = 0;

    for i = 1:length(x)
        if x(i) > prev_code
            delta_modulated_signals(s, i) = 1;
        else
            delta_modulated_signals(s, i) = 0;
        end
        % Update previous code using the delta and step size
        prev_code = prev_code + step_size * (2 * delta_modulated_signals(s, i) -
1);
    end
end

% Plot the original and delta modulated signals for different step sizes
figure;

subplot(length(step_sizes) + 1, 1, 1);
plot(t, x);
title('Analog Signal (Sine Wave)');
xlabel('Time');
ylabel('Amplitude');

for s = 1:length(step_sizes)
   .
   .
   .
    ylabel('Amplitude');
end
```

Try

1. To analyze the impact of different step sizes on the performance of a delta modulation system.
2. Encode a sinusoidal signal using delta modulation with different step sizes. Plot the output signal for each step size and observe the impact on signal fidelity, especially in terms of distortion and quantization noise.

3. Calculate the Signal-to-Noise Ratio (SNR) for delta-modulated signals encoded with different step sizes. Analyze how the SNR changes as the step size vary and its effect on the quality of the reconstructed signal.

5.5 Design an adaptive DPCM system

An Adaptive Differential Pulse Code Modulation (ADPCM) system adjusts its step size or quantization levels based on the characteristics of the input signal. Let's create a simple adaptive DPCM system in MATLAB using a basic adaptive algorithm called the LMS (Least Mean Squares) algorithm to adjust the step size based on the input signal.

```
//Write Matlab code for an adaptive DPCM system
% Parameters
Fs = 1000;                  % Sampling frequency (Hz)
t = 0:1/Fs:1-1/Fs;          % Time vector for 1 second
f = 5;                      % Frequency of the analog signal (Hz)
A = 5;                      % Amplitude of the analog signal
initial_step_size = 1;      % Initial step size for ADPCM

% Generate an analog signal (sine wave)
x = A * sin(. . . . . );

% Initialize variables for adaptive DPCM
adaptive_step_size = initial_step_size;
prev_code = 0;
% Adaptive DPCM encoding using LMS algorithm
for i = 1:length(x)
    if i == 1
        delta_modulated_signal(i) = round(x(i) / adaptive_step_size);
    else
        delta_modulated_signal(i)     =     round((x(i)     -     prev_code)     /
adaptive_step_size);
    end

    % Update previous code using the adaptive step size
    prev_code = prev_code + adaptive_step_size * delta_modulated_signal(i);

    % Adapt the step size using the LMS algorithm (adjustment rule)
    . . . . .
end
```

```
% Reconstruct the signal using inverse ADPCM
. . . . . .
    end
end

% Plot the original and reconstructed signals
figure;
subplot(2,1,1);
plot(t, x);
title('Original Analog Signal');
xlabel('Time');
ylabel('Amplitude');

subplot(2,1,2);
plot(t, reconstructed_signal);
title('Reconstructed Signal (ADPCM)');
xlabel('Time');
ylabel('Amplitude');
```

Try

1. Implement an Adaptive Differential Pulse Code Modulation (ADPCM) system in MATLAB and analyzing its performance.
2. In an ADPCM system, the quantizer step size is initially set to 4. If the received signal deviates from the predicted signal by 6 units and the step size adjustment factor α is 0.5, calculate the updated quantizer step size.

6. Exercises on Digital Modulation Techniques

To be proficient in programming, you need implement the following Digital Modulation

1 Implement: Generating an ASK signal
2 Implement: Generating an FSK signal
3 Implement: Generating an BPSK signal
4 Implement: Generating a QPSK signal

6.1 Implement: Generating an Amplitude shift keying (ASK) signal

Generating an ASK signal

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

1. Connect the set-up shown in Figure 6.1

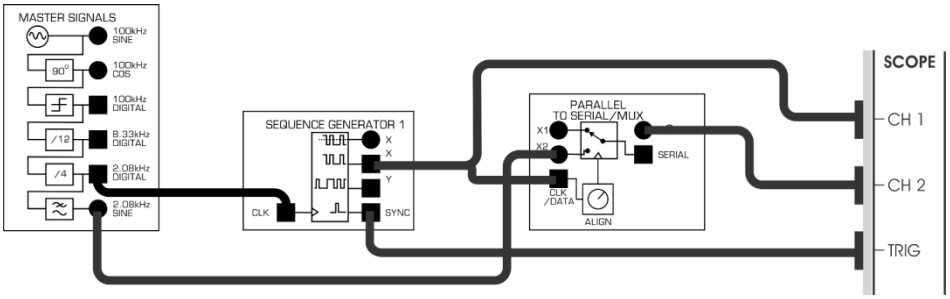


Figure 6.1: Patching for baseband ASK signal generation

2. Modify the set-up as shown in Figure 6.1a
3. Compare the signals

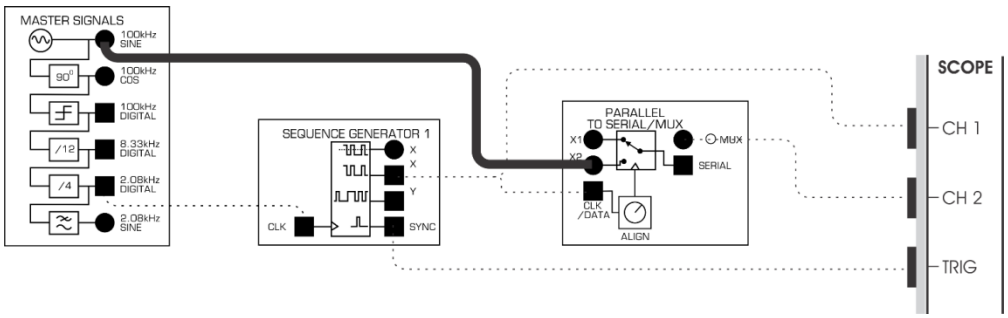


Figure 6.1a : Patching for ASK with 100kHz carrier

Hint

```
% Parameters
fs = 1000;          % Sampling frequency
fc = 50;            % Carrier frequency
T = 1;              % Duration of the signal in seconds
bitsPerSymbol = 2;  % Number of bits per symbol
% Generate random digital data
. . . . . .
% Modulation
. . . . . .
% Create ASK signal
askSignal = data .* carrier;
% Demodulation
```

```
receivedSignal = askSignal .* carrier;
% Low-pass filter to remove high-frequency components
cutoffFreq = 2 * fc / fs;
. . . . ..
demodulatedSignal = filter(b, a, receivedSignal);
% Plot original data, ASK signal, and demodulated signal
figure;
subplot(3, 1, 1);
plot(data);
title('Original Data');
subplot(3, 1, 2);
plot(askSignal);
title('ASK Modulated Signal');
subplot(3, 1, 3);
plot(demodulatedSignal);
title('Demodulated Signal');
xlabel('Time (s)');
```

Try

1. Generate ASK signal with the sample signal frequency is 1200 and carrier frequency is with 100. Plot the ASK Signal.
2. Generate ASK signal with the sample signal frequency is 1500 and carrier frequency is with 125. Plot the ASK Signal.

6.2 Implement: Generating a Frequency shift keying (FSK) signal

Generating an FSK signal

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

Frequency division multiplexing (FDM) allows a channel to be shared among a set of users. Recall that this is achieved by superimposing the message onto a carrier signal inside the user's allocated portion of the radio-frequency spectrum. Recall also that any of the analog modulation schemes can be used to transmit digital data in this way. When frequency modulation (FM) is used it is known as binary frequency shift keying (BFSK or more commonly just FSK).

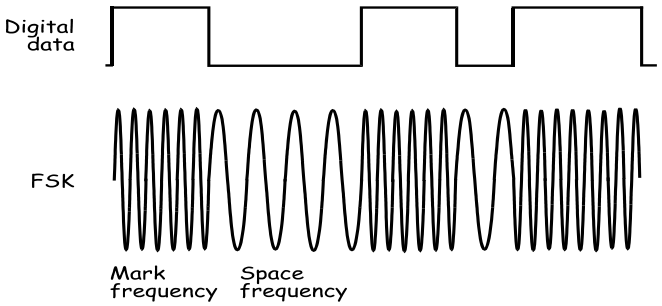


Figure 6.2: Typical FSK signal

Try

1. Ensure that the NI ELVIS III Application Board power button at the top left corner of the unit is OFF (not illuminated).

2. Connect the set-up shown on Figure . Make sure that the micro-switch found under the EX-OR GATE module is set to MUX. Doing so sets the PARALLEL SERIAL/MUX module to MUX mode.
3. Set up the Scope to the values on the following table:

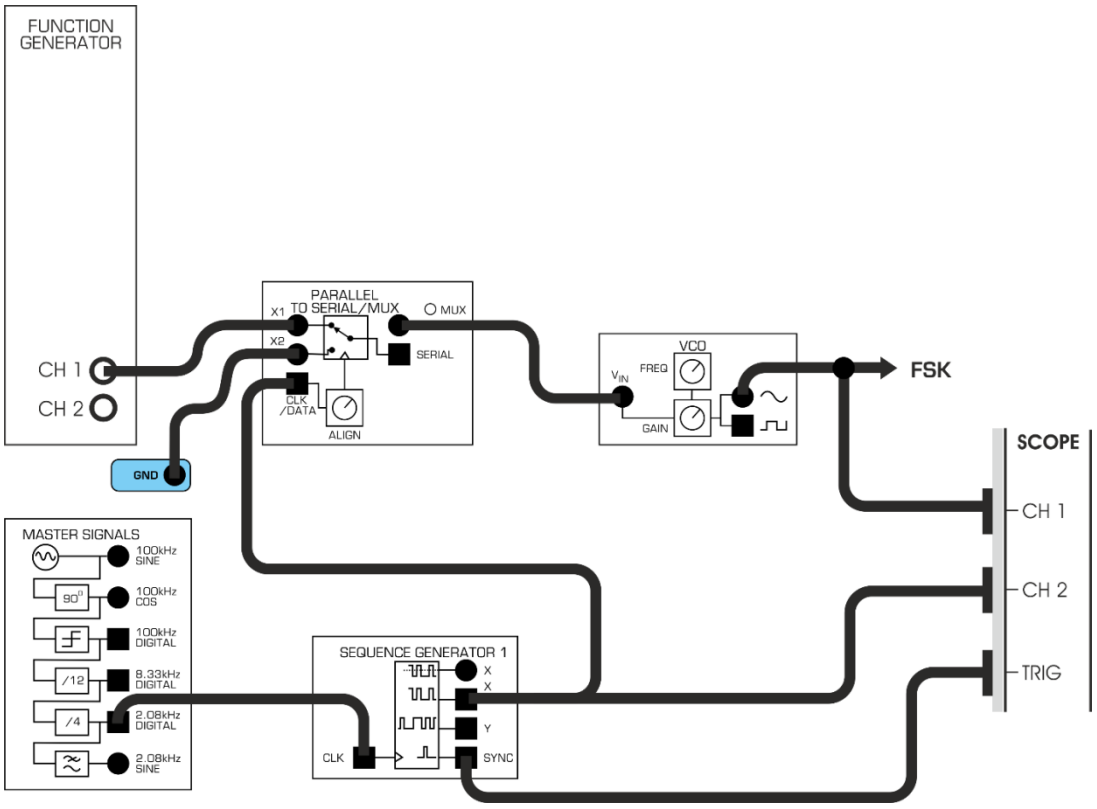


Figure 6.2a : Patching diagram for dual-tone FSK generation

Hint

```
% Frequency Shift Keying (FSK) modulation and demodulation
% Parameters
fs = 1000;          % Sampling frequency
fc1 = 50;           % Carrier frequency for bit 0
fc2 = 100;          % Carrier frequency for bit 1
```

```
T = 1;              % Duration of the signal in seconds
bitsPerSymbol = 2;  % Number of bits per symbol
% Generate random digital data
. . . . . .
% Modulation
. . . . . .
% Create FSK signal
fskSignal = zeros(1, length(t));
for i = 1:length(data)
. . . . . .
    end
end
% Demodulation
. . . . . .
% Low-pass filter to remove high-frequency components
cutoffFreq = (fc1 + fc2) / (2 * fs);
[b, a] = butter(6, cutoffFreq);
. . . . . .
% Perform thresholding to demodulate the signal
demodulatedData = zeros(1, length(data));
for i = 1:length(data)
    . . . . . .
    end
end
% Plot original data, FSK signal, and demodulated data figure;
. . . . ..
stem(demodulatedData);
title('Demodulated Data');
xlabel('Symbol Index'
```

Try

1. Generate FSK signal with the sample signal frequency is 1200 and carrier frequency is with 100. Plot the FSK Signal.
2. Generate FSK signal with the sample signal frequency is 1500 and carrier frequency is with 125. Plot the FSK Signal.

6.3 Implement: Generating an BPSK signal

Generating an BPSK signal

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

AM and FM modulation schemes can be used to transmit digital signals and this allows for the channel to be shared. As digital data forms the message instead of speech and music, it is preferred that these two systems are called ASK and FSK instead.

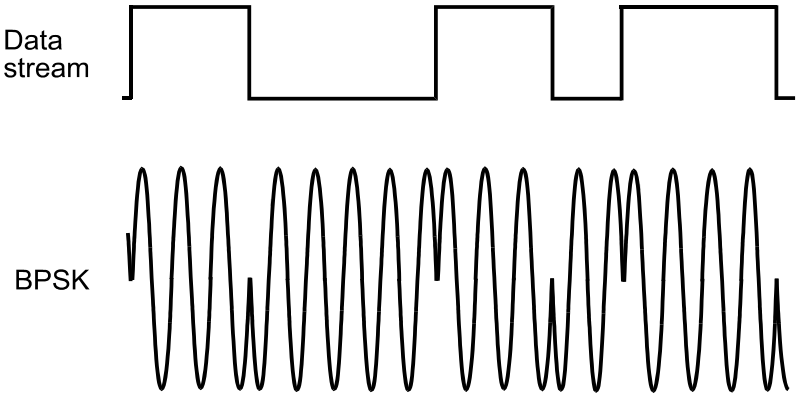


Figure 6.3: BPSK waveform

1. Ensure that the NI ELVIS III Application Board power button at the top left corner of the unit is OFF (not illuminated).
2. Open the Instrument Launcher software in your browser and select the required instruments
3. Connect the set-up shown in Figure 6.3a

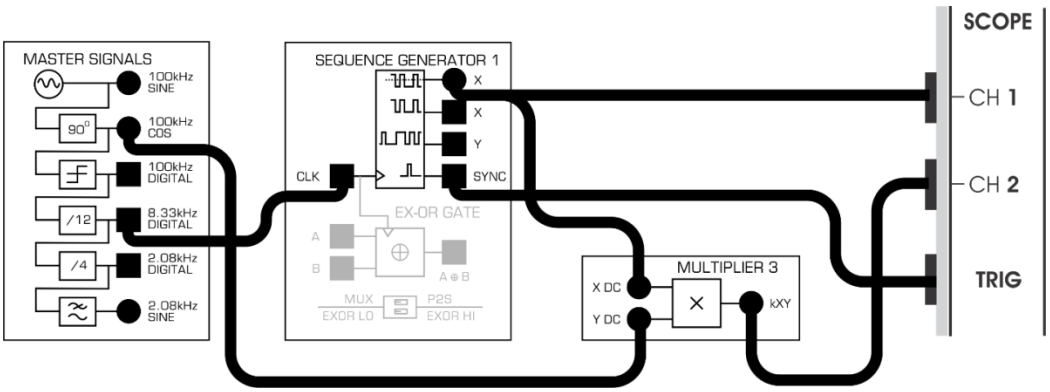


Figure 6.3a: Patching for BPSK generation

4. Launch and run the NI ELVIS III Oscilloscope instrument.

5. Modify the set-up as shown in Figure 6.3b

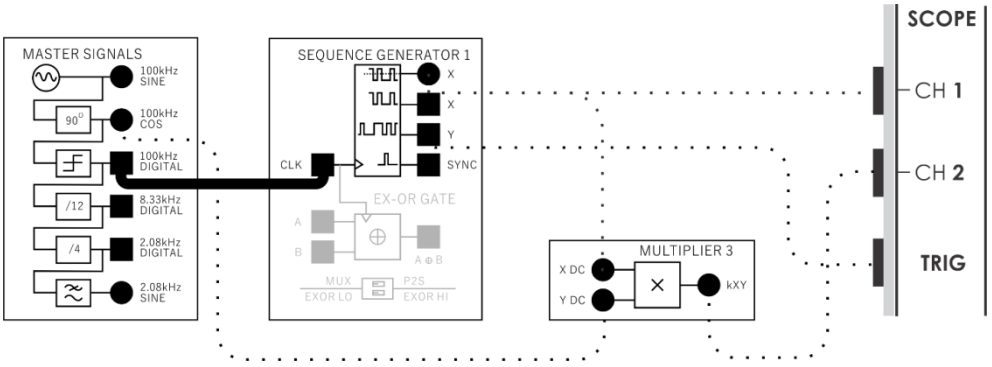


Figure 6.3b: Patching for modified BPSK

6. Set the scope's Timebase control to the 10µs/div position
7. Switch the carrier source from 100k Cos to be 100kHz Sine for a moment and notice the difference at the transition points. Ensure that you understand why this difference occurs. Change it back to 100kHz Cos once you do

Hint

```
% Binary Phase-Shift Keying (BPSK) modulation and demodulation
% Parameters
fs = 1000;          % Sampling frequency
fc = 50;            % Carrier frequency
T = 1;              % Duration of the signal in seconds
bitsPerSymbol = 2;  % Number of bits per symbol
% Generate random digital data
data = randi([0 1], 1, bitsPerSymbol * fs * T);
% Modulation
t = linspace(0, T, fs * T);
carrier = cos(2 * pi * fc * t);
% Create BPSK signal
bpskSignal = zeros(1, length(t));
for i = 1:length(data)
    . . . . .
    end
end
% Demodulation
receivedSignal = bpskSignal .* carrier;
% Low-pass filter to remove high-frequency components
. . . . .
% Perform thresholding to demodulate the signal
demodulatedData = zeros(1, length(data));
```

```
for i = 1:length(data)
    . . . . .
    end
end
% Plot original data, BPSK signal, and demodulated data
figure;
subplot(3, 1, 1);
plot(data);
title('Original Data');
subplot(3, 1, 2);
plot(bpskSignal);
title('BPSK Modulated Signal');
subplot(3, 1, 3);
stem(demodulatedData);
title('Demodulated Data');
xlabel('Symbol Index');
```

Try

1. For a BPSK modulation scheme, if the bit rate is 10 kbps and the carrier frequency is 100 MHz, calculate the duration of each bit (bit period) and the phase shift used in this BPSK modulation.
2. In a BPSK system, if the carrier frequency is 1 MHz and the bit rate is 100 kbps, determine the phase shift used for modulation.

6.4 Implement: Generating an QPSK signal

Generating an QPSK signal

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

As its name implies, quadrature phase shift keying (QPSK) is a variation of binary phase shift keying (BPSK). Recall that BPSK is basically a DSBSC modulation scheme with digital information for the message. Importantly though, the digital information is sent at a symbol rate of one bit at a time. QPSK is a DSBSC modulation scheme also but it sends has a symbol rate of two bits at a time (without

the use of another carrier frequency).

1. Connect the set-up shown in Figure. Make sure that the micro-switch found under the EX-OR GATE module is set to P2S. Doing so sets the PARALLEL SERIAL/MUX module to Parallel-to-Serial mode
2. Activate the scope's CH 1 and CH 2 inputs to observe the Serial-to-Parallel Converter module X1 and X2outputs
3. Pause the Scope and compare the signals. You should see two digital signals output from the Serial-to-Parallel Converter module that are different from each other.
4. Modify the set-up as shown in **Error! Reference source not found.**.4a

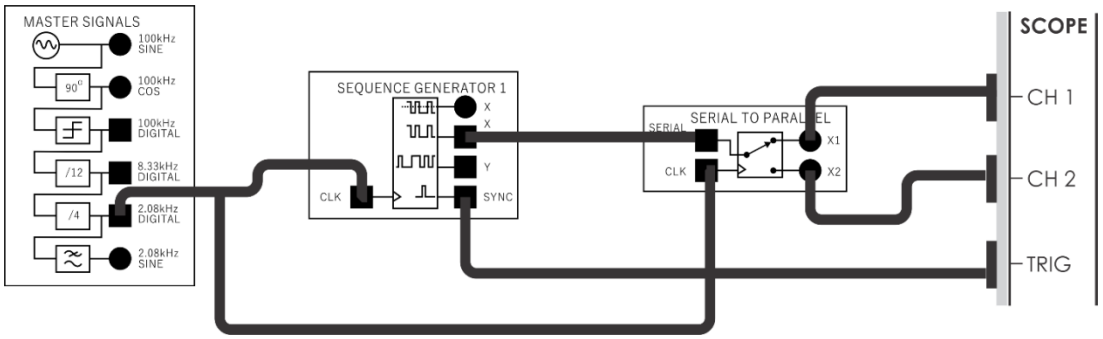


Figure 6.4: Patching for data bit splitting

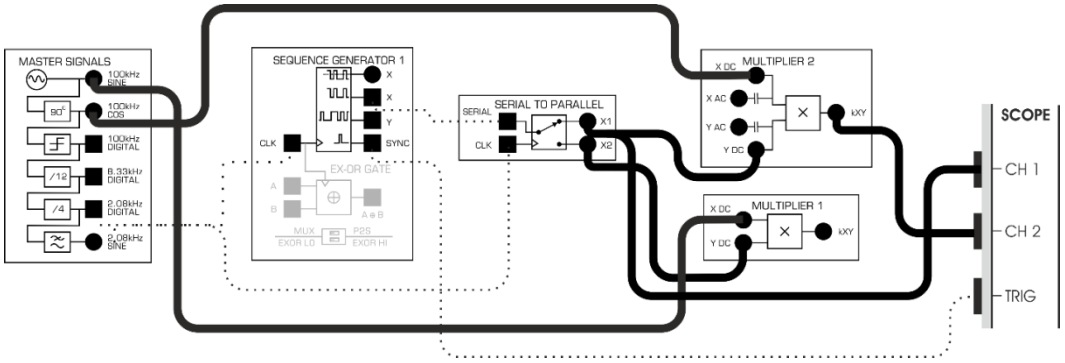


Figure 6.4a: Patching for dual BPSK generation while examining the PSK_1 signal

5. Move the scope's connections as shown in Figure 6.4b

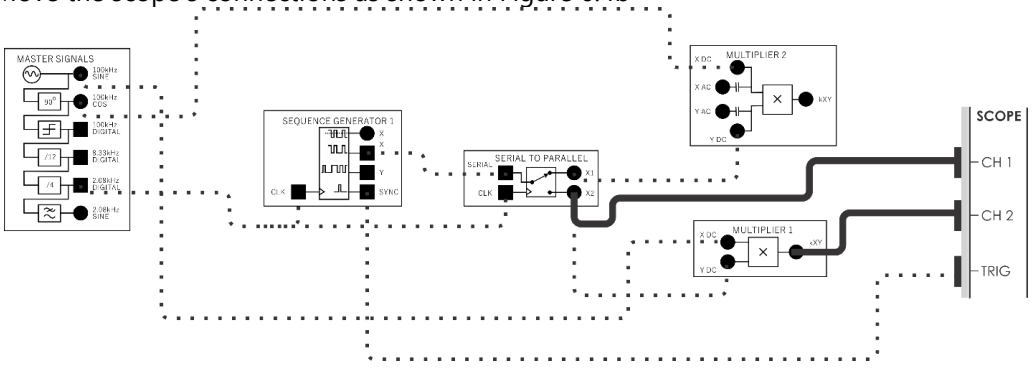


Figure 6.4b: Patching for dual BPSK generation while examining the PSKQ signal

.

6. Modify the set-up as shown in Figure 6.4c

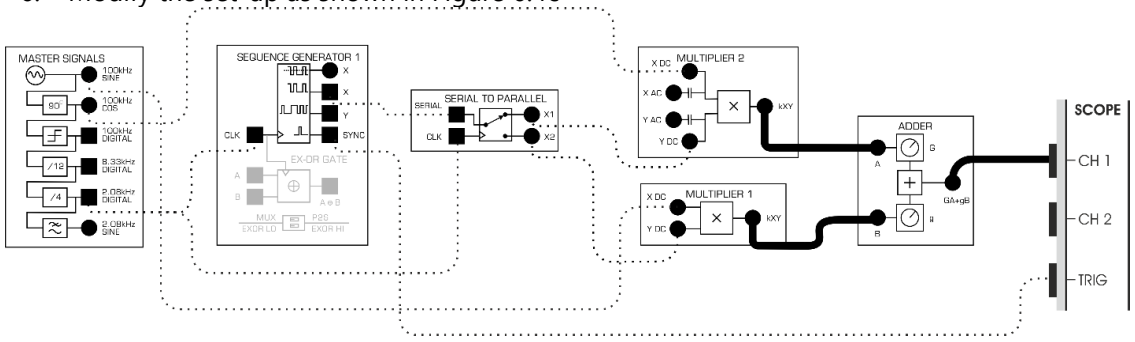


Figure 6.4c 1: Patching for addition of dual BPSK channels

Hint

```
% Quadrature Phase-Shift Keying (QPSK) modulation and demodulation
% Parameters
fs = 1000;          % Sampling frequency
fc = 50;            % Carrier frequency
T = 1;              % Duration of the signal in seconds
Bits Per Symbol = 2;  % Number of bits per symbol
% Generate random digital data
data = randi([0 3], 1, bits Per Symbol * fs * T);
% Modulation
t = linspace(0, T, fs * T);
carrier I = cos(2 * pi * fc * t);
carrier Q = sin(2 * pi * fc * t);
% Create QPSK signal
Qpsk Signal = zeros(1, length(t));
for i = 1:length(data)
    . . . . .
end
% Demodulation
Received I = qpsk Signal .* carrier I;
Received Q = qpsk Signal .* carrier Q;
% Low-pass filter to remove high-frequency components
. . . . .
% Perform phase detection to demodulate the signal
demodulatedData = zeros(1, length(data));
for i = 1:length(data)
    . . . . . .
    end
end
% Plot original data, QPSK signal, and demodulated data figure;
```

```
subplot(3, 1, 1);
stem(data);
title('Original Data');
subplot(3, 1, 2);
plot(qpskSignal);
title('QPSK Modulated Signal');
subplot(3, 1, 3);
stem(demodulatedData);
title('Demodulated Data');
xlabel('Symbol Index'
```

Try

1. In a QPSK modulation system, if the carrier frequency is 2 MHz and the bit rate is 500 kbps, calculate the duration of each symbol (symbol period) and the phase shifts used in QPSK modulation.
2. In a QPSK modulation system, if the carrier frequency is 10 MHz and the symbol rate is 2 Mbps, calculate the duration of each symbol (symbol period) and the phase shifts used in QPSK modulation.

7. Exercises on Signal Processing in Communication Systems

7.1 Implement: Spectrum of the impulse train

Generating an Spectrum of the impulse train

EMONA Communications board to generate a BPSK signal with the Multiplier module to implement its mathematical model. Digital data for the message is modeled by the Sequence Generator module

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

1. Ensure that the NI ELVIS III Application Board power button at the top left corner of the unit is OFF (not illuminated).
2. Patch together the experiment as shown in Figure 7.1

3. Observe the pulse train on the scope. Use the Scope Cursors to measure the pulse width and repetition interval and confirm that these are as you would expect for the settings on the Function Generator
4. Observe the FFT spectrum of this pulse train on Scope Channel 1. Vary the Scope time base, which in return varies the resolution of the frequency scale, to provide a convenient balance between range and resolution in the frequency display

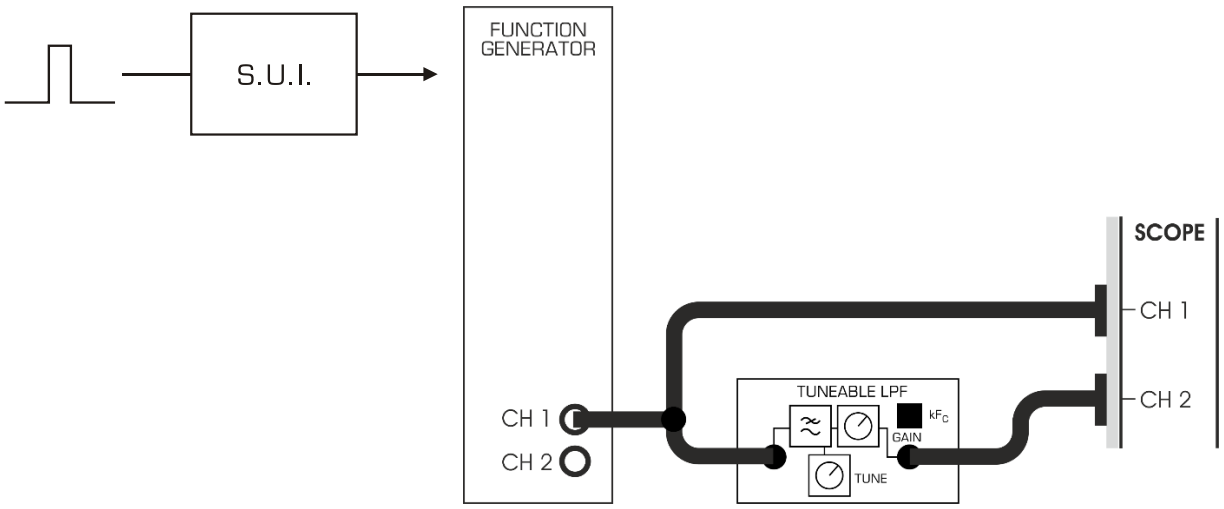


Figure 7.1 : Pulse source to System Under Investigation.

5. Try a final reading with a duty cycle of 1%. Set the Function Generator duty cycle to 0.01 (1%)

Hint

```
% Parameters
fs = 1000;          % Sampling frequency (Hz)
T = 1/fs;           % Sampling period
f0 = 10;            % Fundamental frequency of impulse train (Hz)
duration = 1;       % Duration of the signal (seconds)
% Time vector
t = 0:T:duration;
% Impulse train
x = zeros(size(t));
. . .  . . .
% Compute the spectrum
. . . . .
% Plot the spectrum
figure;
stem(frequencies, abs(X));
title('Spectrum of Impulse Train');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

Try

1. Generate an impulse train with an interval of T=0.1 seconds, starting at time t=0 and ending at t=1 second.
2. Generate an impulse train with an interval of T=0.2 seconds, starting at time t=0 and ending at t=2 second.

7.2 Implement: Spectrum of the filtered impulse train

Generating a Spectrum of the impulse train

EMONA Communications board to generate a BPSK signal with the Multiplier module to implement its mathematical model. Digital data for the message is modeled by the Sequence Generator module

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

1. Ensure that the NI ELVIS III Application Board power button at the top left corner of the unit is OFF (not illuminated).
2. Patch together the experiment as shown in Figure 7.2
3. Observe the pulse train on the scope. Use the Scope Cursors to measure the pulse width and repetition interval and confirm that these are as you would expect for the settings on the Function Generator

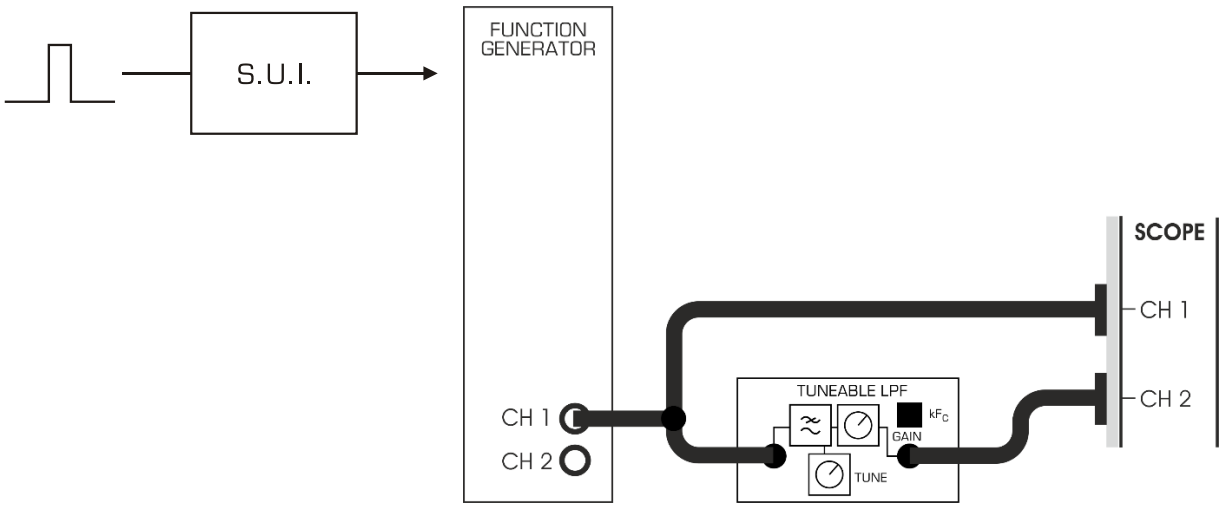


Figure 7.2 : Pulse source to System Under Investigation.

4. Observe the FFT spectrum of this pulse train on Scope Channel 1. Vary the Scope time base, which in return varies the resolution of the frequency scale, to provide a convenient balance between range and resolution in the frequency display.
5. Try a final reading with a duty cycle of 1%. Set the Function Generator duty cycle to 0.01 (1%)

Hint

```
% Parameters
fs = 1000;          % Sampling frequency (Hz)
T = 1/fs;           % Sampling period
f0 = 10;            % Fundamental frequency of impulse train (Hz)
duration = 1;       % Duration of the signal (seconds)
filter_freq = 50;   % Cut-off frequency of the filter (Hz)
% Time vector
t = 0:T:duration;
% Impulse train
x = zeros(size(t));
x(1:round(fs/f0):end) = 1;
% Design a simple low-pass filter
filter_order = 50;
filter_coeff = fir1(filter_order, filter_freq/(fs/2));
% Apply the filter to the impulse train
filtered_signal = filter(filter_coeff, 1, x);
. . . . .  .
% Plot the spectrum
. . . . . .
title('Spectrum of Impulse Train (Original)');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
subplot(2, 1, 2);
stem(frequencies, abs(X_filtered));
title('Spectrum of Filtered Impulse Train');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
linkaxes;
```

7.3 Implement: Spectrum of pseudorandom sequence

Generating a Spectrum of pseudorandom sequence

Usually known as PRBS (Pseudo random bit sequences). We will begin with short clocked Maximal Length sequences, i.e. the kind of sequence available at the SEQUENCE GENERATOR 1 module. Next, we will examine longer length clocked Maximal Length sequences.

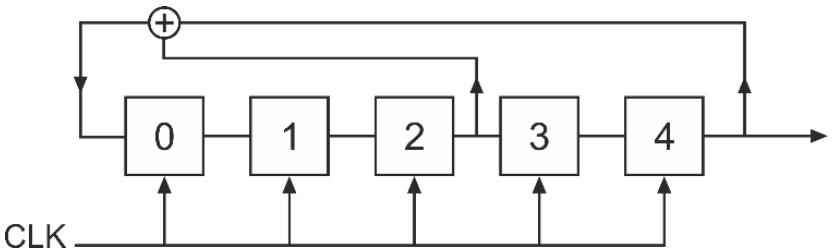


Figure 7.3a :Block diagram for the of DxIQ-45GSEQUENCE GENERATOR 1

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

1. Patch together the experiment in Figure 7.3b and view the output X from SEQUENCE GENERATOR 1

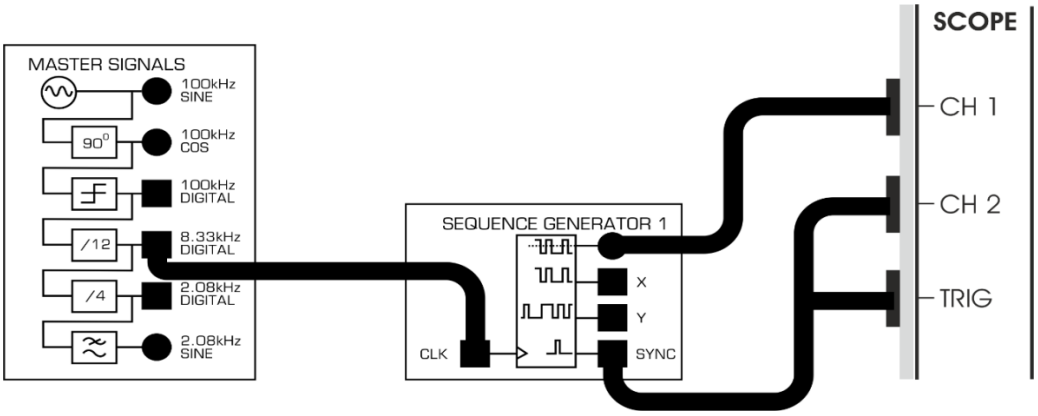


Figure 2.3b Patching diagram for PN sequence from SEQUENCE GENERATOR 1

2. View the spectrum of this signal. Can you see where the nulls of the envelope of the spectrum occur. Vary the scope timebase so that you can see the separation between the individual harmonics of the PN sequence. Notice the sin(x)/x form of the envelope of the spectrum.

Hint

```
% Generate a pseudorandom sequence
% Parameters
N = 1024;   % Length of the sequence
seq = randn(1, N);  % You can use rand() for a uniform distribution
```

```
% Compute the FFT
fft_result = fft(seq);
% Compute the single-sided spectrum
. . . . . .
% Create the frequency vector
.  .. . .
% Plot the spectrum
. . . . .
title('Single-Sided Amplitude Spectrum of Pseudorandom Sequence');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
```

7.4 Implement: Analog noise generation (AWGN)

Implementation of Analog noise generation (AWGN)

The spectrum of composed of many evenly spaced harmonics with a roll off to repetitive nulls. If we wish to create a signal composed of many evenly spaced harmonics of equal amplitude, we can isolate a small region of the PN sequence spectrum and use that as our signal. Using a low pass filter, we can attenuate all harmonics above say 10% PN CLOCK frequency and then we are left with a signal with White Gaussian Noise characteristics which is useful in various experiments as a noise source. AWGN stands for "additive white Gaussian noise" meaning the type of noise which is imposed on a signal travelling through a particular noisy channel.

Required Tools and Technology

Platform: NI ELVIS III

Hardware: EMONA Communications Board

Components used in this lab:

- Four BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads
- Set of headphones or earbuds

Implement: Setting up the data signal

1. Patch together Figure 7.4b and view the output of the TUNEABLE LPF. Set the TUNEABLE LPF controls both to fully clockwise at this stage. View the input and output signals to the filter in the time domain and the frequency domain

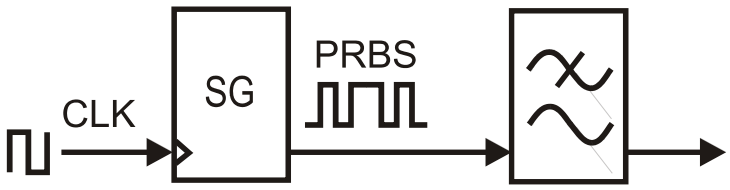


Figure 7.4: Block diagram for PRBS generated analog noise

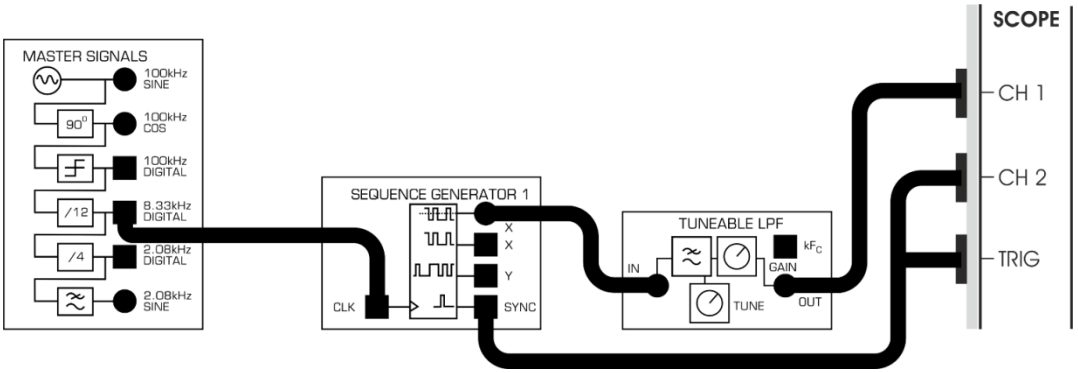


Figure 7.4b: noise Patching diagram for PRBS analog noise

2. With the cutoff frequency of the TUNEABLE LPF above 12 kHz, you can view the unfiltered PN sequence as a bipolar sequence. We are using the bipolar output from SEQUENCE GENERATOR 1 as we want a bipolar analog output with no DC component

8. Exercises on signal filtering

Filtering signals in MATLAB for communication systems can be a crucial task. The process involves several steps, such as signal generation, adding noise, filtering, and analyzing the filtered signal. Here's a simple example of filtering a noisy signal in MATLAB:

To be proficient in programming, you need to be able to:

BASICS OF FILTERING

1. Generate a noisy signal (you can simulate this using a sine wave and adding random noise).

2. Design a basic low-pass filter (such as a moving average filter or a simple RC filter).

3. Apply the filter to the noisy signal to obtain a filtered output.

4. Compare the filtered signal with the original noisy signal to observe noise reduction.

8.1 BASICS OF FILTERING

In the early days of wireless, communication was carried out by telegraphy, the radiated signal being an interrupted radio wave. Later, the amplitude of this wave was varied in sympathy with (modulated by) a speech message (rather than on/off by a telegraph key), and the message was recovered from

the envelope of the received signal. The radio wave was called a 'carrier', since it was seen to carry the speech information with it. The process and the signal were called amplitude modulation, or 'AM' for short.

Implement: FILTERING

Filtering in signal processing is a technique used to modify or extract information from a signal by allowing certain frequencies to pass while attenuating others. Here are the basics of filtering in signal processing:

Common Filter Designs:

1. **Butterworth filter:** Maximally flat frequency response in the passband with a gradual roll-off.
2. **Chebyshev filter:** Can have steeper roll-off than Butterworth but introduces passband ripple.
3. **Elliptic filter:** Steep roll-off with a flat response in both passband and stopband but has ripples in both regions.

Filtering Process:

1. Signal Representation: Signals can be represented in time domain or frequency domain.
2. Filter Design: Determine the type, order, and characteristics of the filter based on the application requirements.
3. Filter Implementation: Apply the designed filter to the signal using techniques like convolution or filter functions available in programming languages like MATLAB.
4. Filter Analysis: Evaluate the filtered signal to ensure it meets the desired specifications, such as removing noise or isolating specific frequencies.

MATLAB Functions for Filtering:

1. filter() function: Implements digital filter on given input signal using filter coefficients.
2. designfilt() function: Creates filter designs based on filter specifications.
3. butter(), cheby1(), ellip() functions: Generate filter coefficients for Butterworth, Chebyshev, and elliptic filters, respectively.

Considerations:

- Trade-offs: Filters involve trade-offs between characteristics like steepness of roll-off, passband ripple, and transition width.
- Computational Complexity: Higher-order filters might be computationally more intensive.
- Real-world Effects: Noise, precision limitations, and implementation constraints can affect filter performance.

These basics should provide a starting point for understanding signal filtering, its types, characteristics, and considerations when applying filters in signal processing applications.

Hint

```
% Parameters
Fs = 1000;                  % Sampling frequency
t = 0:1/Fs:1;               % Time vector
f = 10;                     % Frequency of the signal

% Generate a simple sine wave signal
```

```
. . . . . .

% Add random noise
. . . . . .

% Plot the noisy signal
. . . . . .
grid on;

% Design a simple moving average filter
window_size = 20;
filtered_signal = movmean(noisy_signal, window_size);

% Plot the filtered signal
. . . . . .
grid on;
```

Try

1. Design low pass filter for signal filtering in communication systems.
2. Design high pass filter for signal filtering in communication systems.

8.2 Implement: Frequency domain analysis in signal filtering.

Here's an example MATLAB code that demonstrates frequency domain analysis and filtering of a signal using a band-stop filter:

Hint

```
% Parameters
% Parameters
Fs = 1000;                  % Sampling frequency
t = 0:1/Fs:1;               % Time vector
f1 = 50;                    % Frequency of the signal

% Generate a simple sine wave signal
. . . . . .

% Add noise at specific frequency
. . . . . .

% Perform Fourier Transform to analyze frequency spectrum
. . . . . .
% Plot the frequency spectrum of the noisy signal
figure;
```

```
subplot(2,1,1);
plot(f, P1);
title('Single-Sided Amplitude Spectrum of Noisy Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
grid on;

% Design a band-stop filter to remove noise at 'noise_freq'
. . . . . .

% Design the filter using FIR band-stop method
. . . . . .

% Apply the filter to the noisy signal
filtered_signal = filtfilt(d, noisy_signal);
```

```
% Perform Fourier Transform of the filtered signal
. . . . . .

% Plot the frequency spectrum of the filtered signal
. . . . . .
```

Try

1. Perform frequency domain analysis for signal filter filtering using band pass filter.
2. Perform frequency domain analysis for signal filtering using band stop filter.

8.3 Design of filter optimization in signal filtering.

Filter design is the process of designing a signal processing filter that satisfies a set of requirements, some of which may be conflicting. The purpose is to find a realization of the filter that meets each of the requirements to a sufficient degree to make it useful.

The filter design process can be described as an optimization problem where each requirement contributes to an error function that should be minimized.

Hint

```
% Define parameters
% Parameters
Fs = 1000;                  % Sampling frequency
t = 0:1/Fs:1;               % Time vector
f1 = 50;                    % Frequency of the signal
```

```
% Generate a simple sine wave signal
signal = sin(2*pi*f1*t);

% Add noise at specific frequency bands
. . . . . .
% Design the filter using FIR band-pass method
d = designfilt('bandpassfir','PassbandFrequency1',Fpass1, ...
    'PassbandFrequency2',Fpass2,'StopbandAttenuation',Astop, ...
    'PassbandRipple',Apass,'SampleRate',Fs);

% Apply the filter to the noisy signal
filtered_signal = filtfilt(d, noisy_signal);

% Perform Fourier Transform of the filtered signal
. . . . . .

% Plot the frequency spectrum of the filtered signal
. . . . . .

% Plot the filtered signal in time domain
subplot(3,1,3);
plot(t, filtered_signal);
title('Filtered Signal in Time Domain');
xlabel('Time');
ylabel('Amplitude');
grid on;
```

Try

1. Design a Butterworth low-pass filter with the following specifications: Cutoff frequency: 1 kHz, Maximum pass band ripple: 0.5 dB, Stop band attenuation: 40 dB. Design the filter and plot its frequency response.
2. Design a Chebyshev Type I high-pass filter with the following specifications: Cutoff frequency: 500 Hz, Maximum pass band ripple: 1 dB, Stop band attenuation: 30 dB Design the filter and visualize its frequency response.

9. Exercises on Wireless Communication Systems

9.1 Implement: To design a model of wireless communication systems using Matlab (Two ray channel and Okumura –Hata model)

The Okumura model is a radio propagation model that was built using the data collected in the city of Tokyo, Japan. The model is ideal for using in cities with many urban structures but not many tall blocking structures. The model served as a base for the Hata model.

Okumura model was built into three modes. The ones for urban, suburban and open areas. The model for urban areas was built first and used as the base for others.

Hints

```
 % System Parameters
frequency = 900e6;       % Frequency in Hz
transmitterHeight = 50; % Transmitter height in meters
receiverHeight = 10;    % Receiver height in meters
distance = 100:100:1000; % Distance between transmitter and receiver in meters %
% Two-ray Channel Model
Pt = 1;                 % Transmitted power in Watts
Gt = 1;                 % Transmitter antenna gain
Gr = 1;                 % Receiver antenna gain
L = 1;                  % System loss
% Calculate received power using Two-ray channel model
Pr_two_ray  =  Pt  *  (Gt  *  Gr  *  (transmitterHeight  *  receiverHeight)^2)  ./
(distance.^4 * L); % Okumura-Hata Model
A = 69.55;              % Model parameter
B = 26.16;              % Model parameter
C = 13.82;              % Model parameter
D = 44.9;               % Model parameter
X = 6.55;               % Model parameter
hb = 30;                % Base station height in meters
% Calculate path loss using Okumura-Hata model
PL_okumura_hata = A + B * log10(distance) + C * log10(frequency/1e6) + D - X *
log10(hb);
% Plotting figure;
plot(distance, Pr_two_ray, 'b-', 'LineWidth', 2);
hold on;
plot(distance, PL_okumura_hata, 'r--', 'LineWidth', 2);
xlabel('Distance (m)');
ylabel('Received Power/Path Loss (dB)');
legend('Two-ray Channel Model', 'Okumura-Hata Model');
title('Wireless Communication System Modeling');
grid on;
```

Try

1. Is the Hata model used for signal strength prediction

9.2 To design a Model and simulation of Multipath fading channel

Write a program to perform Model and simulation of Multipath fading channel using MATLAB.

Hints

```
% Simulation parameters
numSamples = 1000;  % Number of samples
numPaths = 3;      % Number of multipath paths
fadePower = 0.5;   % Fading power
% Generate Rayleigh fading channel coefficients
h   =   sqrt(fadePower/2)*(randn(numPaths,   numSamples)   +   1i*randn(numPaths,
```

```
numSamples));
% Generate transmitted signal
txSignal = randn(1, numSamples) + 1i*randn(1, numSamples);
% Simulate multipath fading channel
rxSignal = zeros(1, numSamples);
for    path = 1:numPaths ;
    rxSignal = rxSignal + h(path, :) .* txSignal;
end
% Plot the transmitted and received signals
t = 1:numSamples;
figure;
subplot(2,1,1);
plot(t, real(txSignal), 'b', t, imag(txSignal), 'r');
title('Transmitted Signal');
legend('In-phase', 'Quadrature');
xlabel('Time');
ylabel('Amplitude');
subplot(2,1,2);
plot(t, real(rxSignal), 'b', t, imag(rxSignal), 'r');
title('Received Signal');
legend('In-phase', 'Quadrature');
xlabel('Time');
ylabel('Amplitude');
```

Try

1. Write a program to perform Channel Modeling using MATLAB.

10.Exercises on Data transmission

To be proficient in programming, you need to be able to:

Data transmission is the transfer of data from one digital device to another. This transfer occurs via point-to-point data streams or channels.

10.1 Scrambler

Hints

```
% Parameters
% System Parameters
B = 4 ;
% bits in shift register
. . . . . .
```

```
% shift register seed . 11 dec = 1011 % bin
. . . . . .
% connection register . 9 dec = 1001 bin
. .  . . .
% binary 1000
. . . . .
% binary 0001
. . . . . .
% select rightmost bit of shift register
. . .. . . .
%main shift register shifted right and OR in MSB
. . . . . .
```

Try

1. Perform the experiment on Signal Bandwidth Analysis in MATLAB

10.2 Implement: To design modulation: Spread Spectrum – DSSS Modulation & Demodulation

Hints

```
% Parameters
% Simulation parameters
data = [1 0 1 0 1 1 0 0];
% Original data signal
spreadingCode = [1 1 0 1];
% Spreading code
spreadingFactor = length(spreadingCode);
% DSSS Modulation
. . . . . .  .
% DSSS Demodulation
. . . . . .
% Display Results
disp('Original Data:');
disp(data);
disp('Demodulated Data:');
disp(demodulatedSignal);
```

Try

1. Input data 111000111

11. Exercises on Multiplexing and Demultiplexing techniques.

In telecommunications and computer networking, multiplexing (sometimes contracted to muxing) is a method by which multiple analog or digital signals are combined into one signal over a shared medium. The aim is to share a scarce resource – a physical transmission medium

To study and analyze the process of Multiplexing and Demultiplexing Techniques.

1. Time division multiplexing (TDM).
2. Frequency division multiplexing (FDM)
3. Code division multiple access

11.1 Implement: Time division multiplexing.

Time Division is a technique of transmitting more than one information on the same channel. The samples consist of short pulses followed by another pulse after a long-time interval. This no-activity time intervals can be used to include samples from the other channels as well. This means that several information can be transmitted over a single channel by sending samples from different information sources at different moments in time. This technique is known as Time Division Multiplexing or TDM.

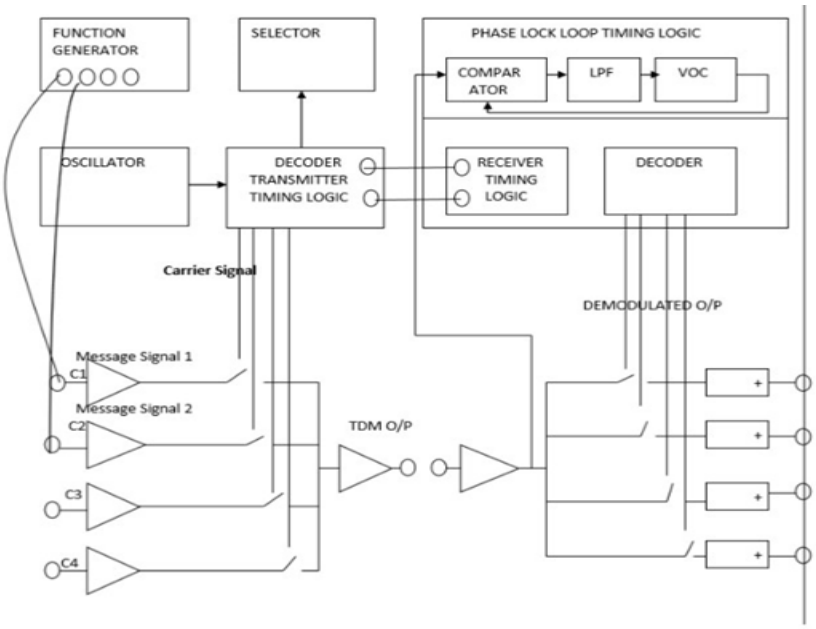


Figure 11.1: Time division multiplexing

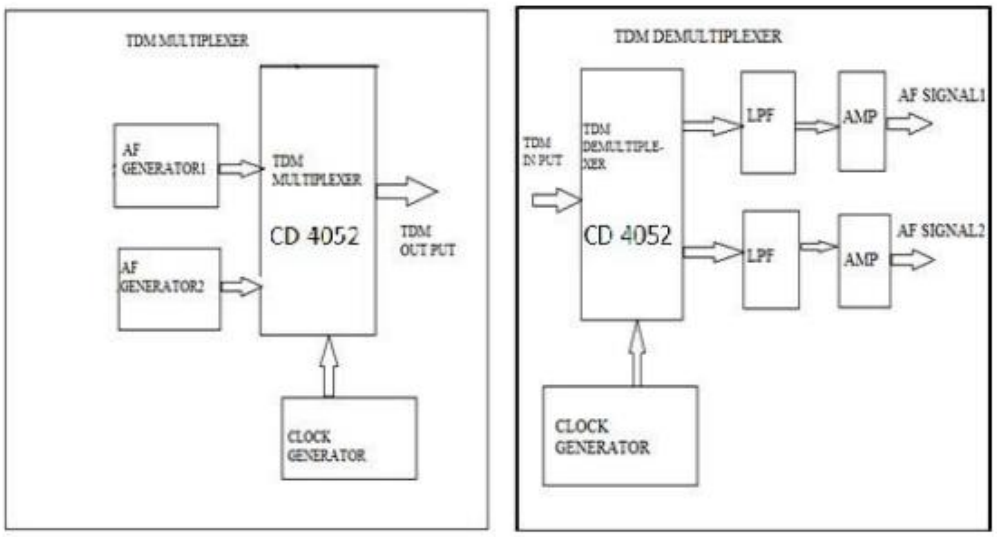


Figure 11.2: Block diagram of Time division multiplexer and demultiplexer

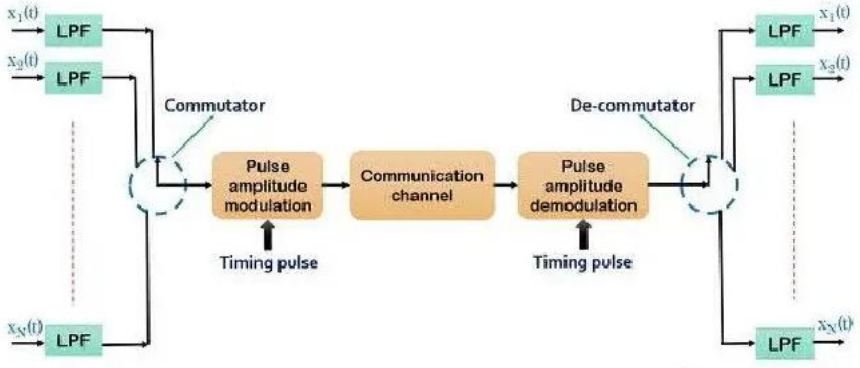


Figure 11.3: Block diagram of Time division multiplexing system.

The technique efficiently utilizes the complete channel for data transmission hence sometimes known as PAM/TDM. This is so because a TDM system uses a pulse amplitude modulation. In this modulation technique, each pulse holds some short time duration allowing maximal channel usage. Here, at the beginning, the system consists of multiple LPF depending on the number of data inputs. These low pass filters are basically anti-aliasing filters that eliminate the aliasing of the data input signal.

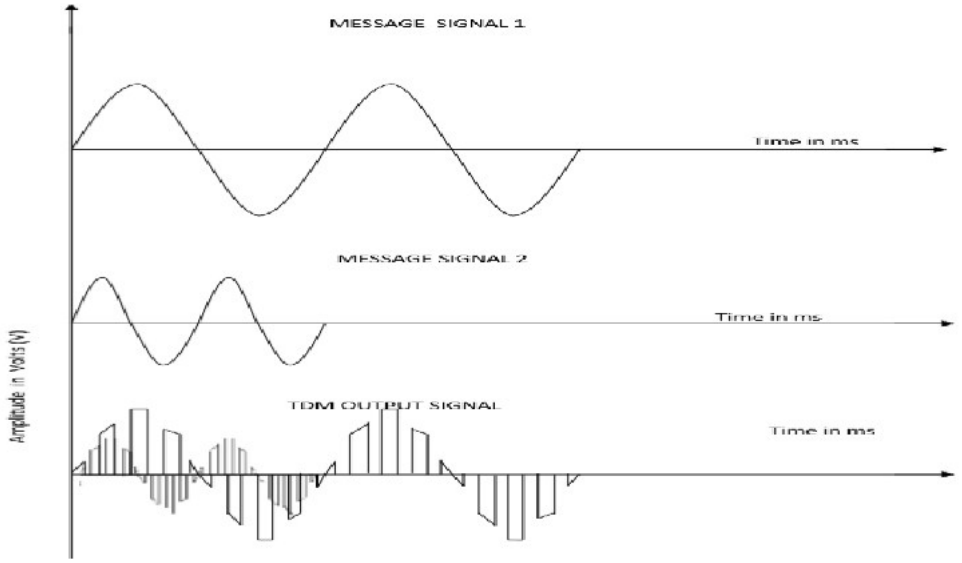


Figure 11.4: Input and output waveforms of Time division multiplexing.

Hints

```
% Time Division Multiplexing (TDM) MATLAB Code

% Number of channels
numChannels = 4;

% Number of time slots
Num
TimeSlots = 8;

% Sampling frequency
fs = 1000; % 1000 Hz

% Time duration of each time slot
slotDuration = 1/fs;

% Total duration of the signal
totalDuration = numTimeSlots * slotDuration;

% Generate time vector
timeVector = 0:slotDuration:totalDuration-slotDuration;

% Create random data for each channel
data = randi([0, 1], numChannels, numTimeSlots);

% Initialize multiplexed signal
multiplexedSignal = zeros(1, length(timeVector));

% Perform Time Division Multiplexing
for channel = 1:numChannels
    multiplexedSignal((channel-1)*numTimeSlots+1:channel*numTimeSlots) = data(channel, :);
end
```

```
% Plot original signals
figure;
subplot(2, 1, 1);
hold on;
title('Original Signals');
for channel = 1:numChannels
. . . . . .
legend('show');
hold off;

% Plot multiplexed signal
subplot(2, 1, 2);
. . . . . .

% Display the plot
legend('Multiplexed Signal');
```

Try

1. What is the time allocated for each channel if the number of samples per frame is 4 and the frame rate is 100 frames/sec?
2. Design a TDMA system for three users with variable data rates: User 1 requires 20 milliseconds, User 2
   requires 15 milliseconds, and User 3 requires 25 milliseconds for transmission. Allocate time slots efficiently to accommodate these varying data rates within a frame of 60 milliseconds.
3. Given a TDMA system with six users sharing a channel, each allocated a time slot of 10 milliseconds in a frame of 60 milliseconds, calculate the maximum achievable throughput in bits per second assuming ideal conditions (no overhead, no gaps between time slots).

11.2 Implement: Frequency division multiplexing (FDM).

Frequency division multiplexing is the process of combining several information channels by shifting their signals to different frequency groups within the frequency spectrum sothat they can all be transmitted over a common transmission channel. The information signals are shifted in different frequency groups by making them modulate carrier signals at different frequencies. The deriving of two or more simultaneous, continuous channels from a transmission medium by assigning a separate portion of the available frequency spectrum to each of the individual channels is known as FDM.

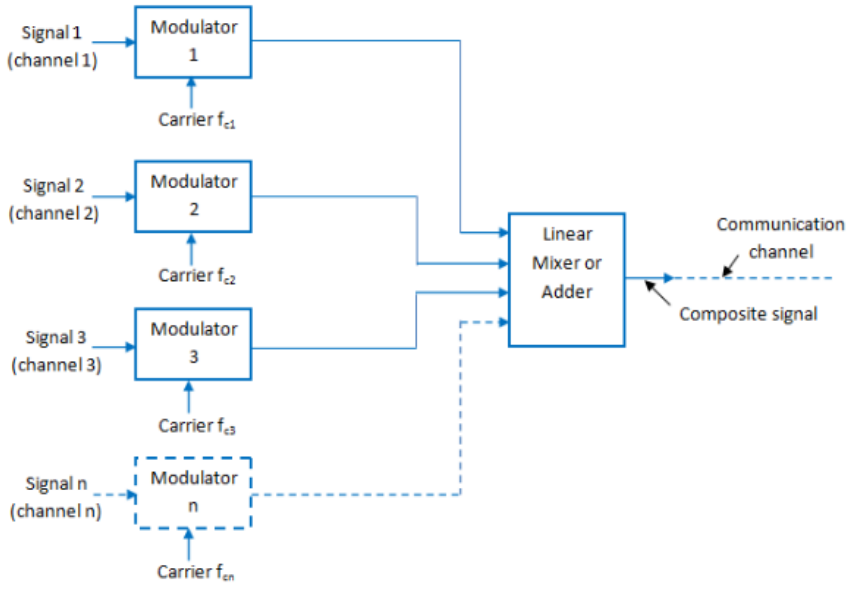


Figure 11.2.3: FDM Transmitter

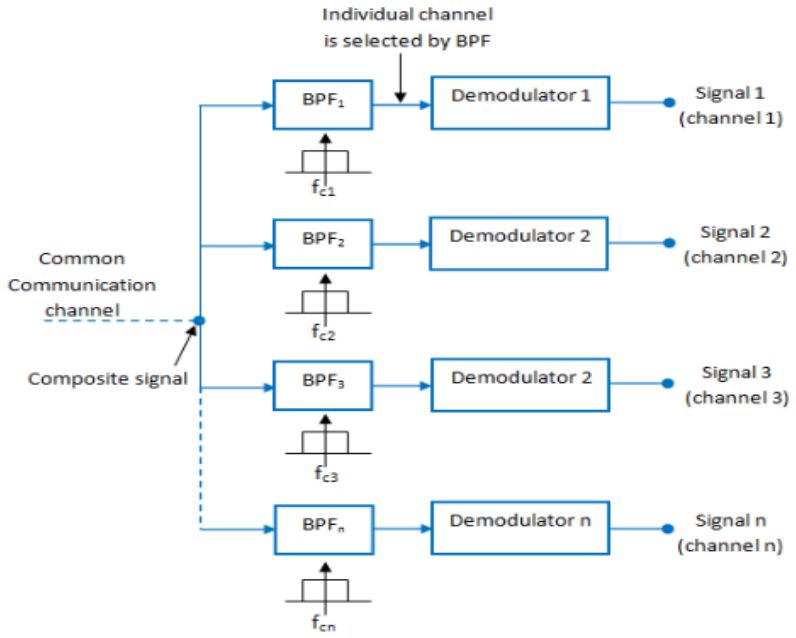


Figure 11.2.4: FDM Receiver

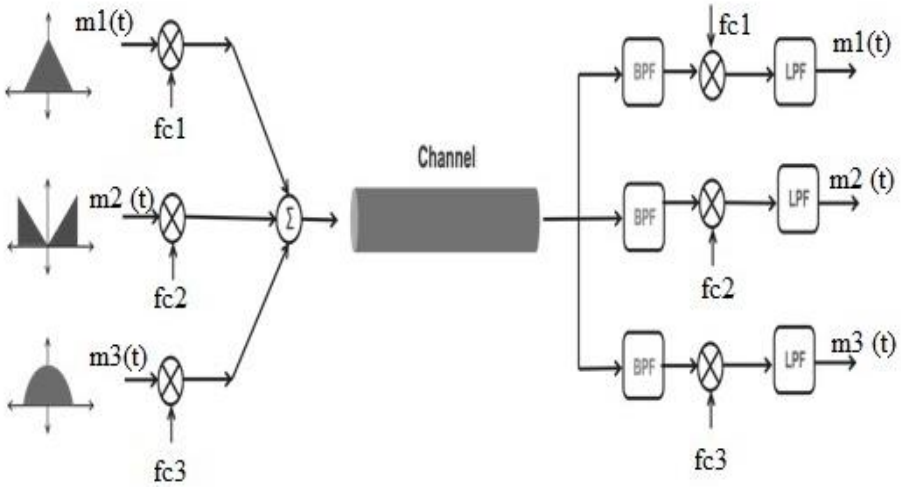


Figure 11.2.5: Block diagram of FDM

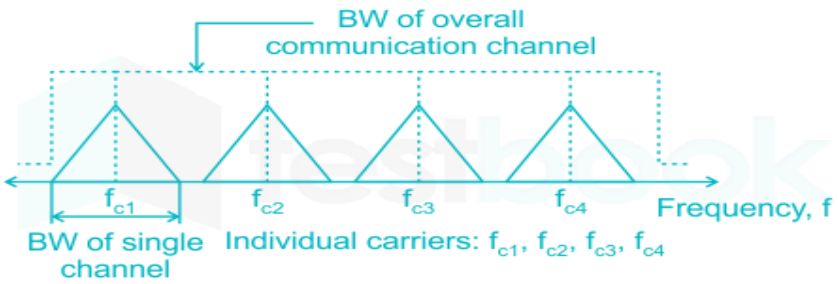


Figure 11.2.6: spectrum of FDM signal

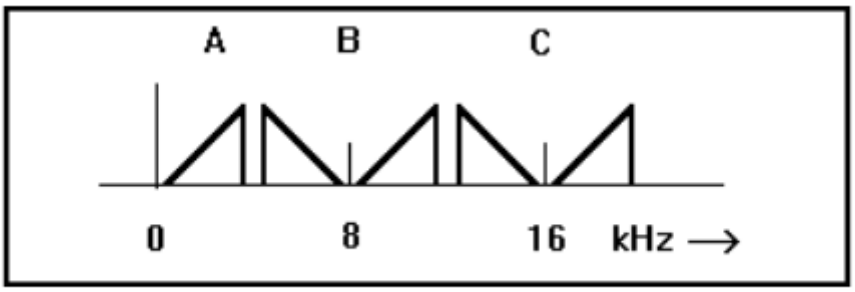
Hints

```
/** Frequency division multiplexing **/
% Frequency Division Multiplexing (FDM) Example in MATLAB
% Number of signals to multiplex
numSignals = 3;
% Signal parameters
Fs = 1000; % Sampling frequency
t = 0:1/Fs:1; % Time vector for one second
% Generate example signals
. . . . . .
% Frequency bands for each signal
. . . . . .
% Combine signals using FDM
. . . . . .
end
% Plot original signals
. . . . .
```

```
. . . . .
. . . . .
% Plot FDM signal
. . . . . .
% Display frequency bands
disp('Frequency Bands for Each Signal:');
disp(freqBands);
```

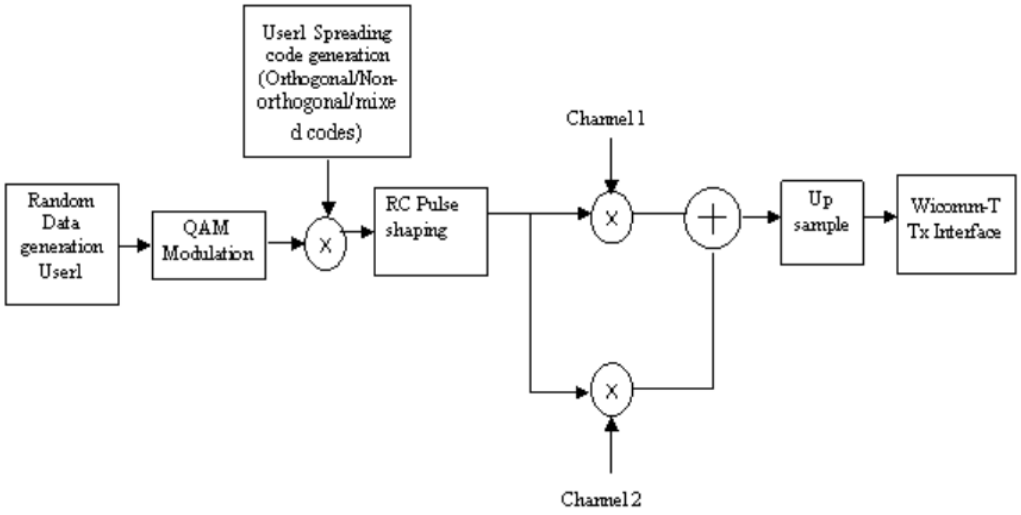
Try

1. To avoid the need for SSB generators to make the FDM signals, an alternative scheme could generate the spectrum shown below. Could a DSBSC demodulator be used in principle to recover the three channels independently? Can you suggest a possible practical problem(s) with this scheme?

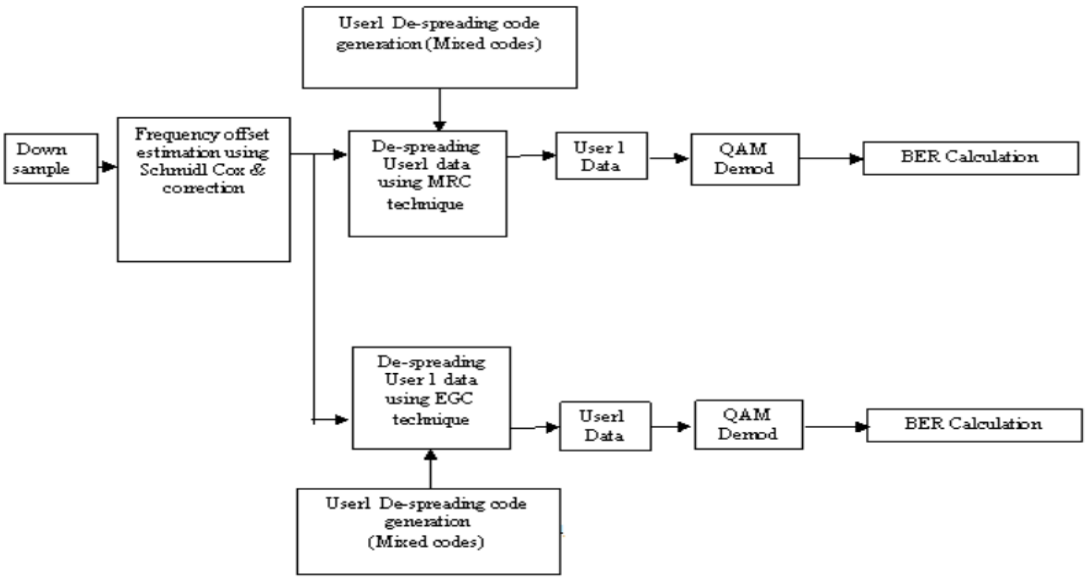


11.3 Implement code division multiple access (CDMA)

1. To observe the BER performance of DS-CDMA using mixed codes in multipath channel using RAKE receiver for single user case.
2. Observe the BER performance of MRC combining and equal combining varying with SNR.



Transmitter Block diagram of CDMA- Multipath Transmitter



Hints

```
/** % CDMA with Multipath Simulation ** %/
clear all;
close all;
clc;
% Parameters
numUsers = 4;           % Number of users
bitsPerUser = 100;      % Number of bits per user
SNR_dB = 10;            % Signal-to-noise ratio in dB
delaySpread = 5;        % Multipath delay spread (in samples)
% Generate random data for each user
data = randi([0 1], numUsers, bitsPerUser);
```

```
% CDMA spreading code (random for each user)
spreadingCode = randi([0 1], numUsers, bitsPerUser);

% Modulation
modulatedData = 2 * data - 1;

% CDMA encoding
encodedData = modulatedData .* spreadingCode;

% Multipath channel simulation
h = (1/sqrt(2)) * (randn(delaySpread, numUsers) + 1i * randn(delaySpread,
numUsers));

% Transmit through the multipath channel
transmittedSignal = filter(h, 1, encodedData);

% Add AWGN (Additive White Gaussian Noise)
noisePower = 10^(-SNR_dB/10);
receivedSignal = transmittedSignal + sqrt(noisePower/2) *
(randn(size(transmittedSignal)) + 1i * randn(size(transmittedSignal)));

% CDMA decoding
decodedData = sum(receivedSignal .* spreadingCode, 2);

% Decision threshold
threshold = 0;

% Demodulation
demodulatedData = (real(decodedData) > threshold);

% BER calculation
ber = sum(xor(data, demodulatedData)) / (numUsers * bitsPerUser);

% Display results
disp(['Bit Error Rate (BER): ', num2str(ber)]);

% Plotting
figure;
subplot(2, 1, 1);
stem(data(1, :), 'Marker', 'none');
```

```
title('Transmitted Data for User 1');

subplot(2, 1, 2);
stem(demodulatedData(1, :), 'Marker', 'none');
title('Received and Demodulated Data for User 1');
```

Try:

1. Given two data sequences: $D1=\{1,-1,1,1\}$ and $D2=\{-1,1,-1,-1\}$. Multiply these sequences with spreading codes }$C1=\{+1,-1,+1,+1\}$ and $C2=\{-1,-1,+1,-1\}$ for CDMA transmission.
2. Suppose you have a received signal $R=\{2,-2,4,2\}$ after CDMA transmission with code $C=\{+1,-1,+1,+1\}$. Decode the received signal to retrieve the original data sequence.
3. Consider two spreading codes: $C1=\{+1,-1,+1,+1\}$ and $C2=\{-1,-1,+1,-1\}$. Determine if these codes are orthogonal to each other.

12. Exercises on Generation of Noises

12.1. Generation of White noise:

This exercise aims to provide hands-on experience on Generation of White Noise. The term is used, with this or similar meanings, in many scientific and technical disciplines, including physics, acoustic engineering, telecommunications, statistical forecasting, and many more.

White noise is a random signal with a constant power spectral density, and it has equal intensity at different frequencies.

Hints

```
% White noise

% Parameters
duration = 5;   % Duration of the white noise signal in seconds
fs = 44100;     % Sampling frequency in Hz (adjust as needed)

% Generate white noise
white_noise = randn(1, duration * fs);

% Normalize to have zero mean and unit variance
white_noise = white_noise / std(white_noise);

% Plot the white noise signal
t = linspace(0, duration, duration * fs);
figure;
plot(t, white_noise);
title('White Noise Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

```
grid on;

% Play the white noise (optional)
% sound(white_noise, fs
```

Try

1. Consider a random signal with the mean of 1 and the variance of 2. What is the power of this signal? How do you calculate signal power in MATLAB
2. How do you use fftshift() with pwelch() in MATLAB®? Give an example.

12.2. Generation of Brown noise:

This exercise aims to provide hands-on experience on Generation of Brown Noise. Brown noise, also known as Brownian noise or red noise, is a type of random signal or noise that exhibits a power spectral density inversely proportional to the frequency. This means that as the frequency increases, the power in the signal decreases, resulting in a roll-off that gives the noise a deeper or "bassier" sound.

Brown noise is characterized by a more pronounced low-frequency component compared to white noise, which has equal power across all frequencies. The term "brown" is derived from the concept of Brownian motion, a type of random motion observed in particles suspended in a fluid. Brown noise is often described as having a rumbling or thunder-like quality.

Hints

```
% Brown noise
% Parameters
duration = 5;   % Duration of the white noise signal in seconds
fs = 44100;     % Sampling frequency in Hz (adjust as needed)

% Generate Brown noise
brown_noise = cumsum(randn(1, duration * fs));

% Normalize to have zero mean and unit variance
brown_noise = brown_noise / std(brown_noise);

% Plot and play the brown noise
t = linspace(0, duration, duration * fs);
figure;
plot(t, brown_noise);
title('Brown Noise Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

```
% Play the brown noise (optional)
% sound(brown_noise, fs);
```

Try

1. Consider a random signal with the mean of 3 and the variance of 5. What is the power of this signal? How do you calculate signal power in MATLAB
2. How do you use repmat () with reshape () in MATLAB? Give an example.

12.3. Generation of Pink noise:

This exercise aims to provide hands-on experience on Generation of Pink Noise. Pink noise uses a consistent frequency, or pitch, to create a more even, flat sound, like a steady rain, wind rustling through trees, or waves on a beach. Its added depth and lower waves filter out higher sounds.

Pink noise, 1/*f* noise or fractional noise or fractal noise is a signal or process with a frequency spectrum such that the power spectral density (power per frequency interval) is inversely proportional to the frequency of the signal. In pink noise, each octave interval (halving or doubling in frequency) carries an equal amount of noise energy.

Hints

```
% Pink noise

% Parameters
duration = 5;   % Duration of the white noise signal in seconds
fs = 44100;     % Sampling frequency in Hz (adjust as needed)

% Generate Pink noise
pink_noise = pinknoise(duration * fs);

% Normalize to have zero mean and unit variance
pink_noise = pink_noise / std(pink_noise);

% Plot and play the pink noise
t = linspace(0, duration, duration * fs);
figure;
plot(t, pink_noise);
title('Pink Noise Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Play the pink noise (optional)
% sound(pink_noise, fs);
```

Try

1. Consider a random signal with the mean of 2 and the variance of 3. What is the power of this signal? How do you calculate signal power in MATLAB
2. How do you use hist () with hist3 () in MATLAB? Give an example.

13. Exercises on case study: Software-Defined Radio (SDR)

Wireless system engineers often want to use a software-defined radio (SDR) prototype to verify and optimize system architecture, baseband signal processing algorithms, and system control protocols before first silicon implementation. In some cases, the SDR prototype can be the final product.

Equipment needed

1. The hardware components required for SDR implementation
2. Highlight advancements in hardware technology that enable SDR

Introduction:

The top-level structure of the Simulink model consists of a fading radio network block and multiple wireless transceiver nodes (Figure). The nodes communicate in an ad hoc way: Any node in the network can transmit and receive signals to or from any other node. The wireless transceiver works in half-duplex mode. Within this network structure, the output of each transceiver node can reach all the inputs of the other transceivers. Following the principle of radio channel reciprocity, the two channels between any two nodes must have the same settings, while the channel settings of different wireless transceiver pairs can be different. The signal detector (SD), is used to detect the signal arrivals to the wireless transceiver. The SD waveforms are shown in Figure . The absolute value of the signal is filtered by a two-stage exponential moving average filter. Once the output of the moving average filter exceeds the threshold, the detector generates the enable signal SigDe to start the DFE and DCRC blocks.

The buffer block at the output SigOut generates the two-dimension (sample) data to the fractional DFE.

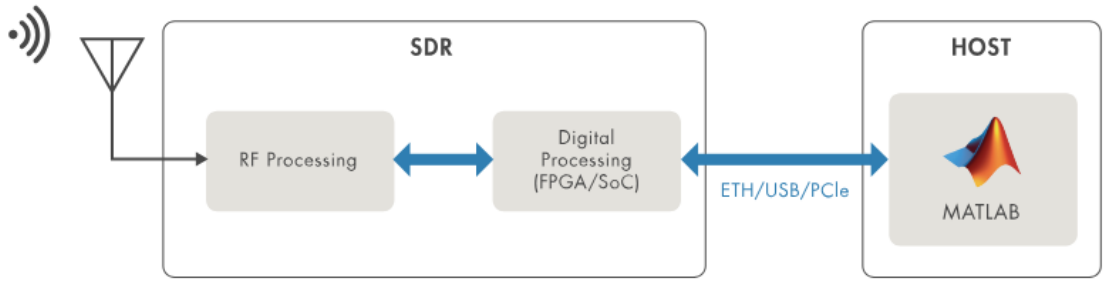


Figure 13.1: Circuit Diagrams

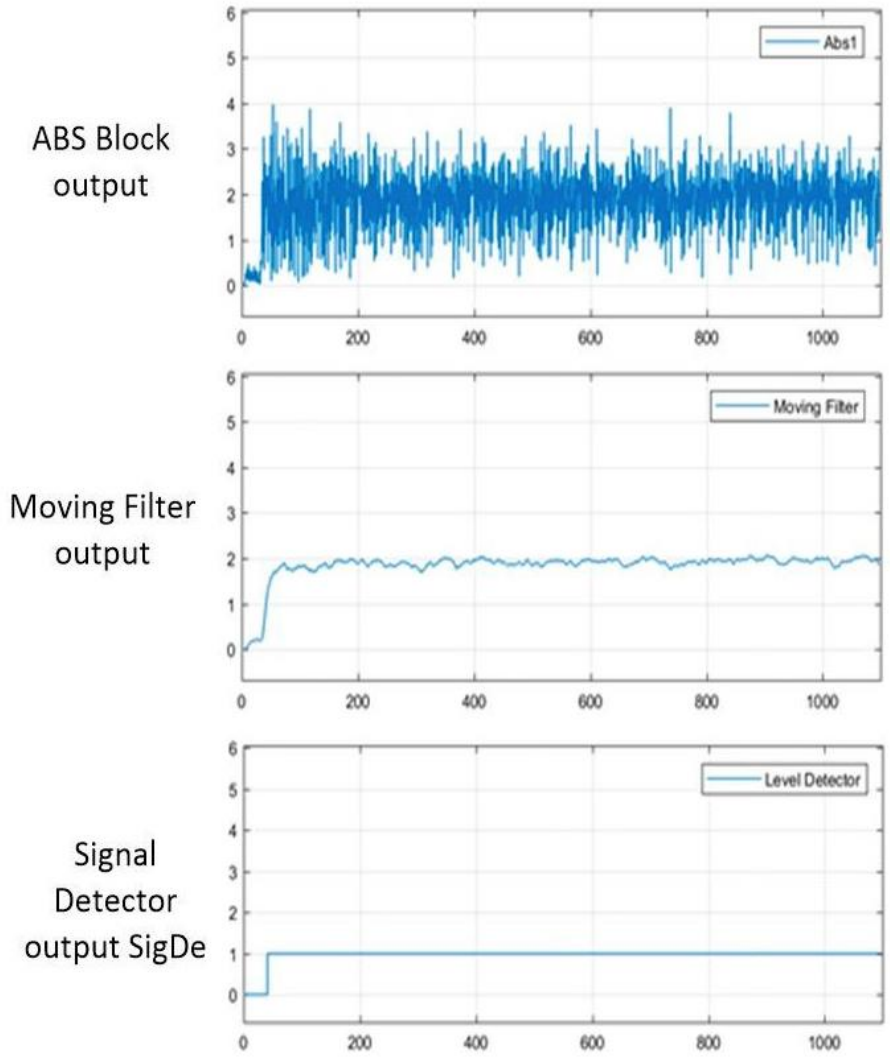


Figure 13.2: Signal waveforms from the signal detector

14. Exercises on case study: communication system project

Create a simple voice communication system using microcontrollers and wireless modules.

Creating a voice communication system using microcontrollers and wireless modules involves several steps, including hardware setup and programming. Here's a simplified example using Arduino microcontrollers and RF modules like nRF24L01 for wireless communication:

Hardware Setup:

1. Arduino Uno and Genuino Uno
2. HC-05 Bluetooth Module
3. Jumper Wires
4. Two AC Bulbs (Red, Yellow)
5. Relay.

Arduino Code

```
String voice ;
int RED = 2 ;
int YELLOW = 3 ;
void RedOn( )
{ digital Write (RED, LOW) ;
}
void RedOff ( ) {
digital Write (RED, HIGH) ;
}
void YellowOn ( ) {
digital Write (YELLOW, LOW) ;
}
. . . . .
. . . . .
void setup ( )
{
Serial . begin ( 9 6 0 0 ) ;
pinMode (RED, OUTPUT) ;
pinMode (YELLOW, OUTPUT) ;
digital Write (RED, HIGH) ;
digital Write (YELLOW, HIGH) ;
}
void loop ( )
{
while ( Serial . available ( ) )
{
delay ( 1 0 ) ;
char c=Serial . read ( ) ;
i f ( c==’#’)
{
break ;
}
. . . . .
. . . . .
{
RedOn ( ) ;
}
else if ( voice == “red off”){ RedOff ( ) ;
}
else if ( voice == “yellow” | | voice == “yellow on”)
{
YellowOn ( ) ;
}
else if ( voice == “yellow off ”)
{
YellowOff ( ) ;
}
voice = “ ”;
```

```
  }
}
```

Try

1. Extend the functionality of RAM memory block by allowing write and read operations to construct RAM block of size 16 x 8 RAM.

15. Final Notes

The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)). Check out these sites:

- MathWorks Certification Program
- National Institute of Electronics and Information Technology (https://reg.nielitchennai.edu.in)

Student can have any one of the following certifications:

- NPTEL – Analog Communications
- NPTEL – Digital Communications

IV. TEXT BOOKS:

1. J. G. Proakis, Digital Communications, McGraw-Hill, 5th edition, 2006.
2. B.P.Lathi, "Modern Analog and Digital Communication", Oxford reprint, 3rd edition, 2004.

V. REFERENCE BOOKS:

1. Wayne Tomasi, "Electronics Communication Systems-Fundamentals" 5th edition, 2009.

VI. ELECTRONICS RESOURCES:

1. https://archive.org/details/analog communications
2. https://archive.org/details/digital communications

VII. MATERIALS ONLINE

1. Course template
2. Lab Manual