ADVANCED LTE MODEM DESIGN WITH INTERFERENCE CANCELLATION

By

NAIMATULLAH
MUHAMMAD HARIS
ASAD AMIR KHAN

Project Advisors
LT COL Dr. ADNAN AHMED KHAN
MAJ Dr. RIZWAN GHAFFAR

Submitted to the Faculty of Electrical Engineering Department National University of
Sciences and Technology, Islamabad in partial fulfillment for the requirements of a
B.E. Degree in Telecom Engineering

JUNE 2013

ABSTRACT

3rd &4th Generation wireless communication systems are targeting to achieve these objectives by more aggressive use of spectrum which leads to interference-limited scenarios. Effective mitigation of these interferences is necessary to achieve the objectives of high spectral efficiency. Interference Mitigation is the currently the focus of attention of both academia and industry. This project will look at interference mitigation through advanced receiver structure.

CERTIFICATE OF CORRECTNESS AND APPROVAL

It is certified that the work contained in the thesis titled “Advanced LTE Modem Design With Interference Cancellation”, carried out by Naimatullah, Muhammad Haris, Asad Amir Khanunder the supervision of Lt Col Dr. Adnan Ahmed Khan and Maj Dr. RizwanGhaffarin partial fulfillment of the degree of Bachelor of Telecommunication Engineering, is correct and approved.

Approved By

_____

Lt Col Dr. Adnan Ahmed Khan

Project Supervisor

Military College of Signals, NUST

DEDICATION

To Almighty Allah, for Whose greatness we do not have enough words,

To our parents and friends, without whose unflinching support and unstinting cooperation, a work of this magnitude would not have been possible

ACKNOWLEDGEMENTS

First of all, we praise Allah Almighty, who gave us strength to undertake this project and to complete it in a timely and efficient manner.

We wish to express our gratitude to our supervisor Lt Col Dr. Adnan Ahmed Khan, from theFaculty of Electrical Engineering, Military College of Signals, National University of Sciences and Technology, for his continuous support and supervision during the course of the project. His persistent guidance is certainly commendable.

We would specially like to thank Maj Dr. RizwanGhaffar. It would not have been possible to complete the project on time without his technical and moral support.

We would also like to acknowledge the struggle of our parents for our well being and education. They have done endless efforts to make us standout. We would designate our success to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

4G	4th Generation
LTE	Long Term Evaluation
3GPP	3rd Generation Partnership Project
Tx	Transmitter
Rx	Receiver
BICM	Bit Interleaver Coded Modulation
OFDM	Orthogonal Frequency Division Multiplexing
SINR	Signal to Interference Noise Ratio
USRP	Universal Software Radio Peripheral
DSP	Digital Signal Processing
CP	Cyclic Prefix
FEC	Forward Error Correction
DFT	Discrete Fourier Transform

IDFT	Inverse Discrete Fourier Transform
BER	Bit Error Rate
RF	Radio Frequency
ZF	Zero Forcing
MMSE	Minimum Mean Square Error
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
SNR	Signal to Noise Ratio
A/D	Analog to Digital Converter
D/A	Digital to Analog Converter
GMSK	Gaussian Minimum Shift Keying
BPSK	Binary Phase Shift Keying
QPSK	Quadrature Phase Shift Keying

Chapter 1

Introduction

1.1 Overview

LTE-Advanced is latest cellular communication standard, targeting high data rates and is considered to be true 4G technology. The objective of high data rate introduces the interference in the system, which degrades system performance. Techniques to mitigate this interference are the focus of attention both in industry and academia. Our aim is to implement advanced receiver design and state of the art transmitters as per LTE standard(3 GPP LTE Release 8) for interference mitigation.

1.2 Project Scope

This project shall develop an interference scenario, which shall comprise of two transmitters (Txs) and a receiver (Rx), where the transmitters will be single antennas, while receiver will be equipped with multiple antennas (possibly two or more). The

system will be quasi compliant with 4G wireless system i.e., it will based on the key features of Physical layer of LTE, i.e. bit interleaved coded modulation (BICM) orthogonal frequency division multiplexing (OFDM) based transmission system.

1.3 Problem statement

Interference in wireless communication system results in reducing the signal to interference noise ratio (SINR). It results in increased noise level for single-user receivers, thereby leading to significant performance degradation. There is a need to design intelligent receivers, which can mitigate the interferences and can lead to enhanced data rates.

1.4Project Objectives

1. To get the knowledge about establishing a communication between two ends using USRPs.
2. Develop an interference scenario with multiple transmitters and a single receiver; where receiver employs advanced interference exploitation techniques.

3. The goal of LTE was to increase the capacity and speed of wireless data networks. It will have significant applications in both military and commercial sectors.

Chapter 2

LTE Based Transmitter

2.1 Design

The design of the project is based upon two LTE transmitters and one Advanced LTE receiver, which is interference aware receiver. Transmitter contains different LTE blocks, which is shown in Figure1.

First of all, the SISO link is established, then applying interference mitigation technique.to implement the advanced interference aware receiver.for the SISO link, the LTE transmitter will be discussed in this chapter, similarly, in the next chapter the LTE receiver will be discussed.

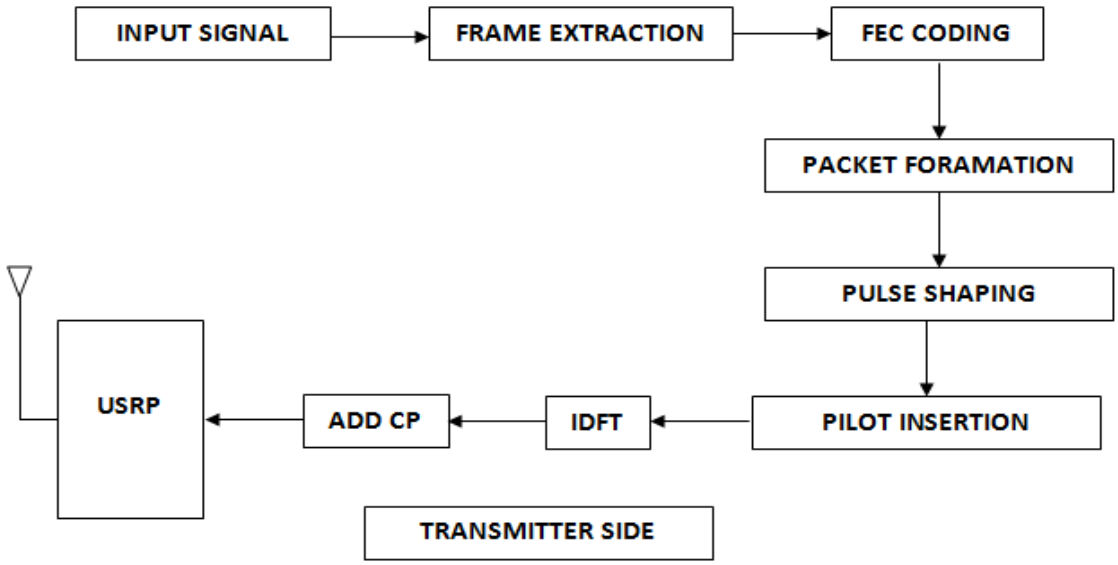

Figure 1: LTE Based Transmitter

2.2Input signal

Input signal may be video signal, image, voice signal, sinusoidal signal or any random data. In this project, input signal is video signal in the format of .avi (file extension). Actually, video is a combination of pictures (frame) or moving images, which are displayed consecutively in a specific frame rate. For simulation, input signal is random bits and for hardware implementation, video or image is the input signal.video signal is the real time video, which is generated with the integration of webcam from transmitter. The resolution of the real time video signal is 240x160 pixels.

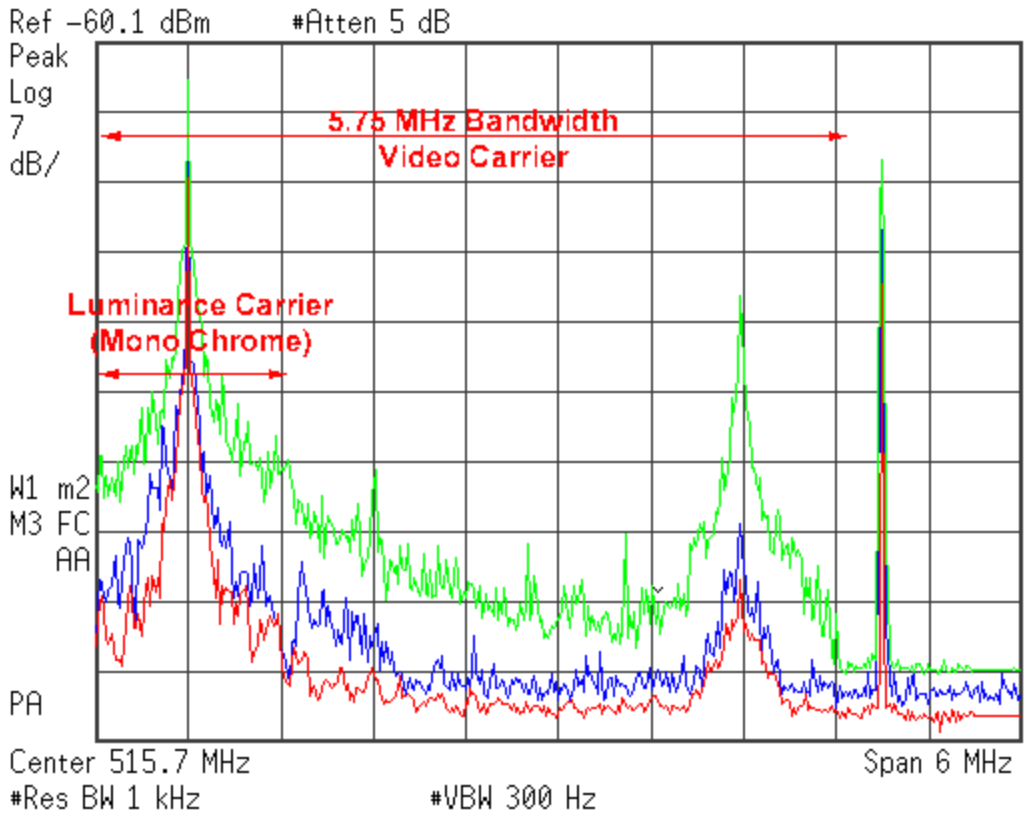

Figure 2: Input signal (Video signal)

2.3 Frame extraction

Frame extraction is the process of extracting the still image frames from the video file for processing. Here, the real time video is the input signal and extraction of frames from that video, will be processed. Extracted frames will be saved in the virtual memory. When the new frames will come, and occupy the same memory, then the previous frames will go to the next block.

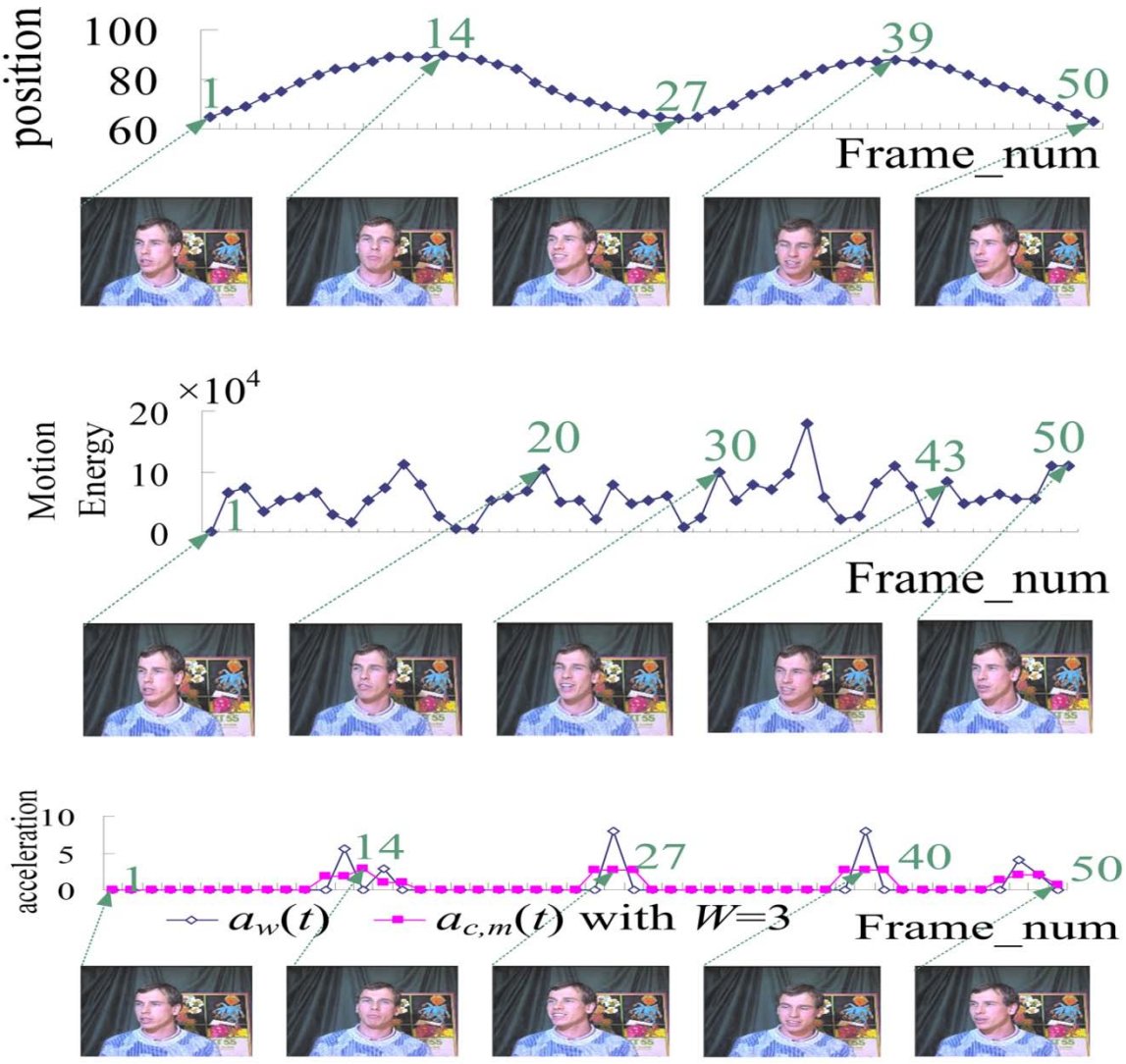

Figure 3: Frame extraction from video

2.4 FEC coding

The basic purpose of using the forward error correction(FEC) coding, is to minimize the bit error rate. The function is to correct and detect the limited number of errors. In FEC coding, retransmission of data stream(bits) is not possible.

Convolutional codes are used in this project.

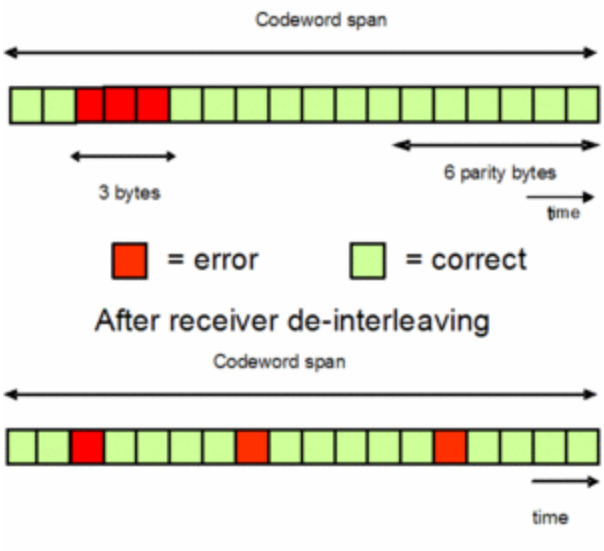

Figure 4: Forward error correction (FEC)

2.4.1 Convolutional codes

Convolutional codes have two basic parameters; the code rate and the constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. K, denotes the constraint length of convolutional encoder, i.e. how many, k-bit stages are required to feed the combinatorial logic, that produces the output symbols. The m parameter is closely related to K , which indicates, how many encoder cycles an input bit is retained and used for encoding, after it first appears at the input to the convolutional encoder. The m parameter represents,the memory length of the convolutional encoder.

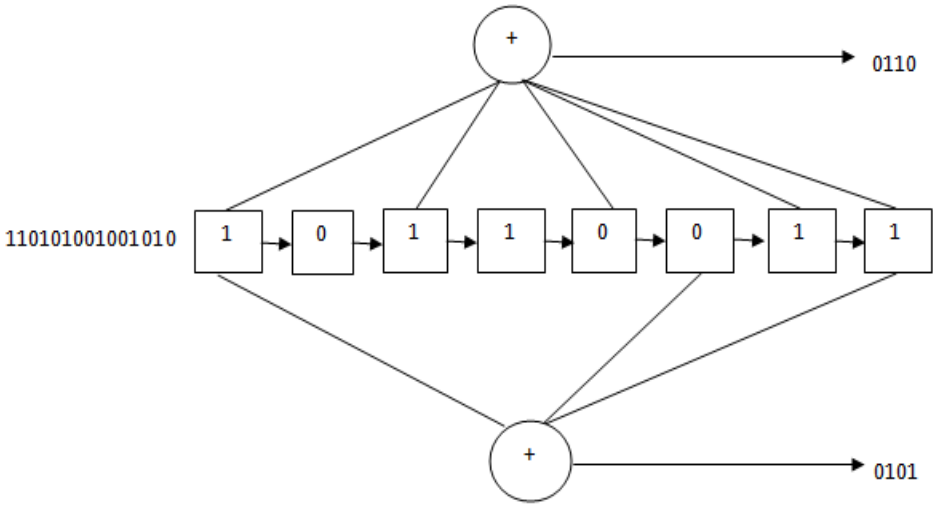

Figure 5:Convolutional encoder with constraint length 7, Rate 1/2

2.5 Packet formation

Packet of data is started with thepreamble, some address information, some other transmission-related information, followed by the raw data, and finishes up with a few more bytes of transmission-related-error-detection information. Convolutional codes are used in this project.

10001001	Payload

2.6 Modulation

The aim of the modulation is to convey a message signal. QPSK, is one of the most popular digital modulation techniques, used for wireless communication and sending data

over cable networks. Its popularity comes from both;it's easy implementation and resilience to noise. A QPSK modulated carrier undergoes, four distinct changes in phase, that are represented, as symbols and can take on the values of π/4, 3π/4, 5π/4, and 7π/4. Each symbol, which represents two binary bits of data.Gray Coding, which is used in this project.

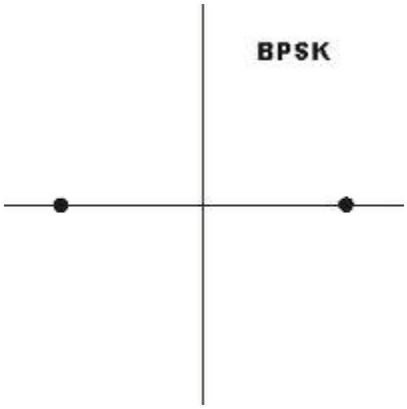

Figure 6: Constellation diagram of BPSK

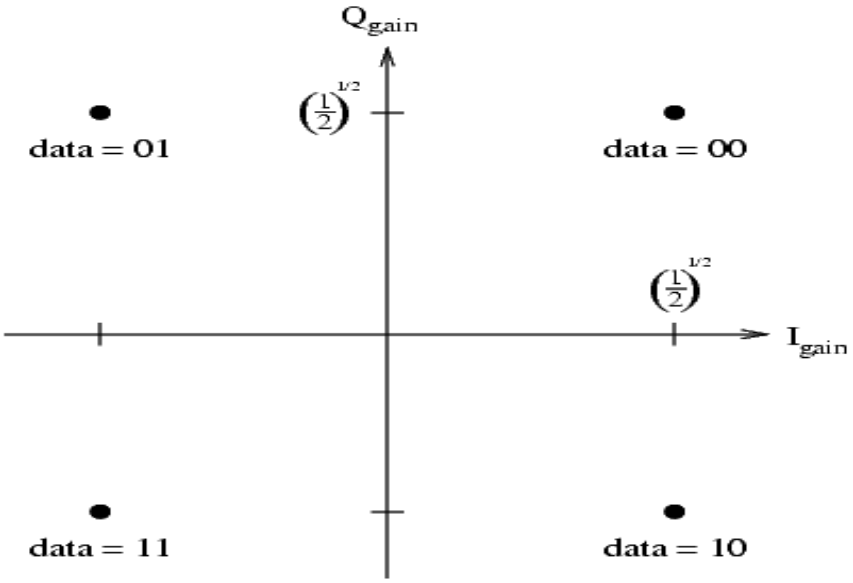

Figure 7: Constellation diagram of QPSK

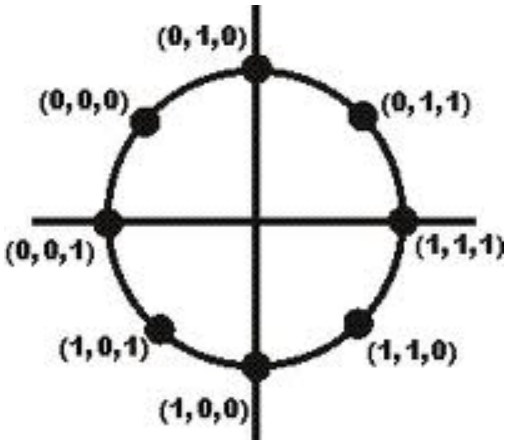

Figure 8: Constellation diagram of 8 QAM

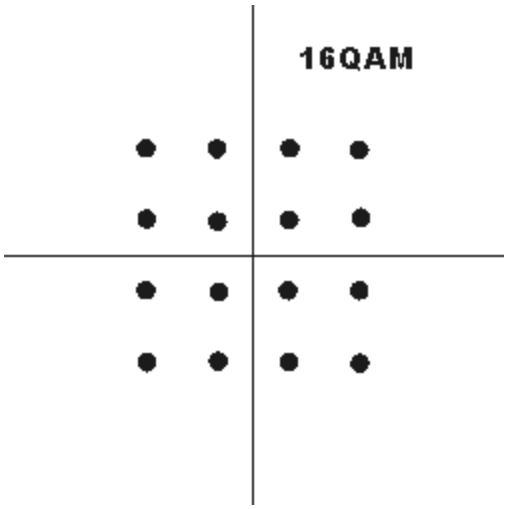

Figure 9: Constellation diagram of 16 QAM

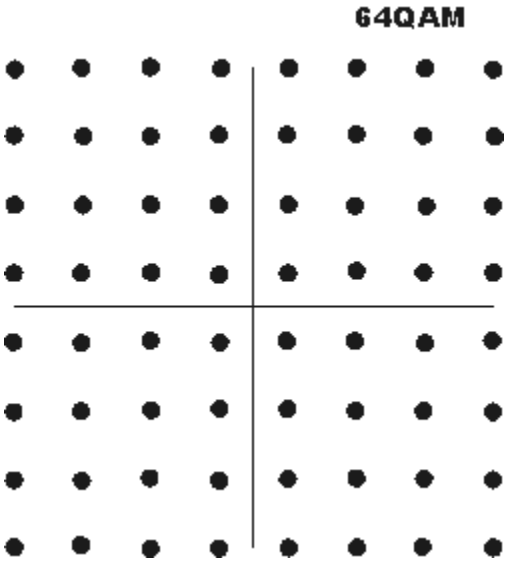

Figure 10: Constellation diagram of 64 QAM

2.7 Pulse shaping

Pulse shaping, which is the process of changing the waveform of transmitted pulses. Its purpose, is to make the transmitted signal better suited to its purpose or the communication channel, typically by limiting, the effective bandwidth of the

transmission. By filtering, the transmitted pulses this way, the inter-symbol interference caused by the channel, can be kept in control.

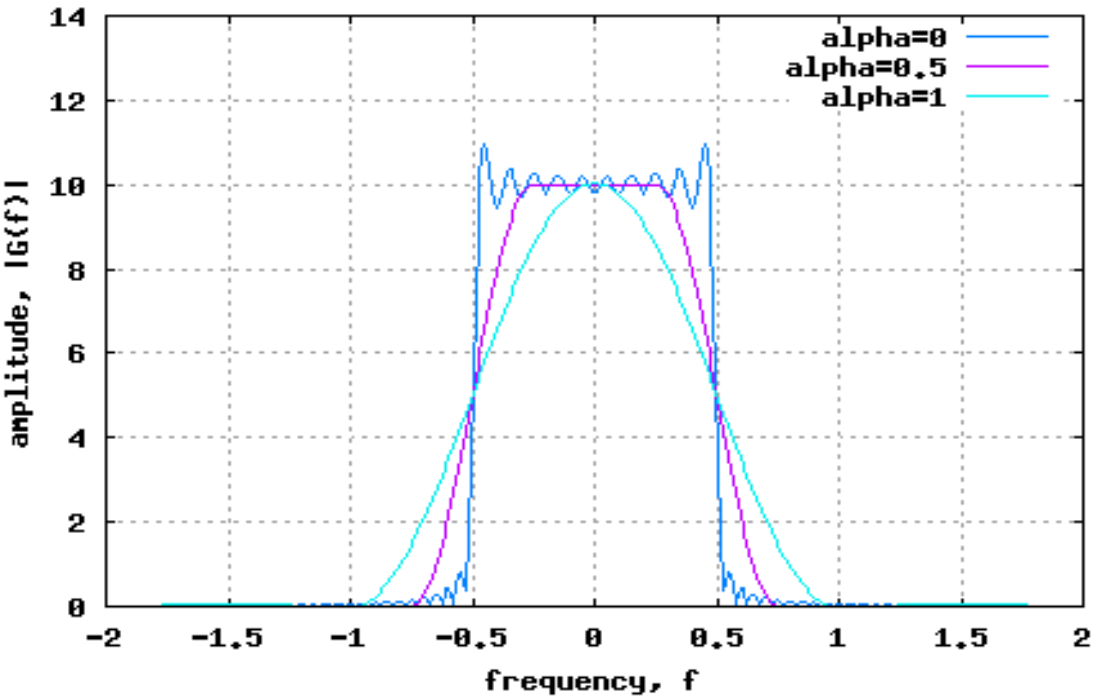

Figure 11: Pulse Shaping filters

2.9 Pilot insertion

OFDM, which requires synchronization and estimation.Pilots, that are sent along with signal, which helps in determining the condition of channel, and its effect on the signal.

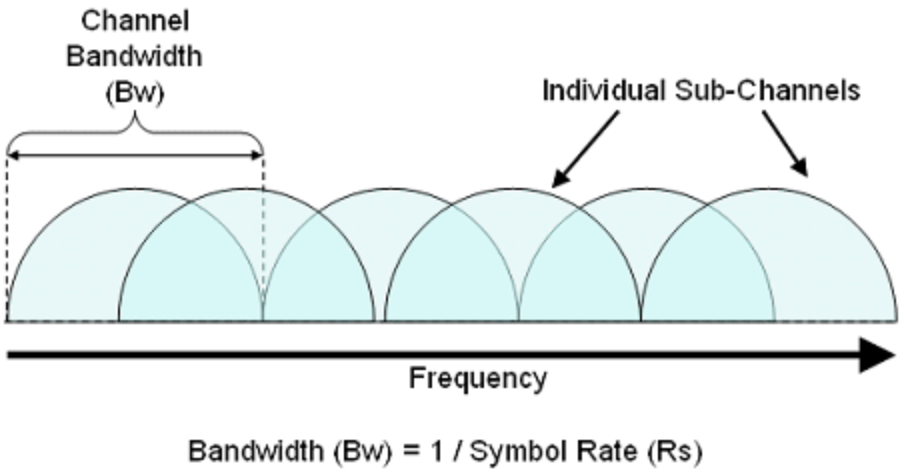

Figure 12: Pilot insertion

2.10 IDFT (Inverse Discrete Fourier Transform)

The inverse Fourier transform, maps the signal back from the frequency domain into the time domain. The formula of calculating the IDFT for N subcarriers of any signal is given below:

$$s \quad = \frac{1}{N} \sum_{k=0}^{N-1} S \quad \exp\left\{ j \frac{2\pi n k}{N} \right\} \quad 0 \leq n \leq N-1$$

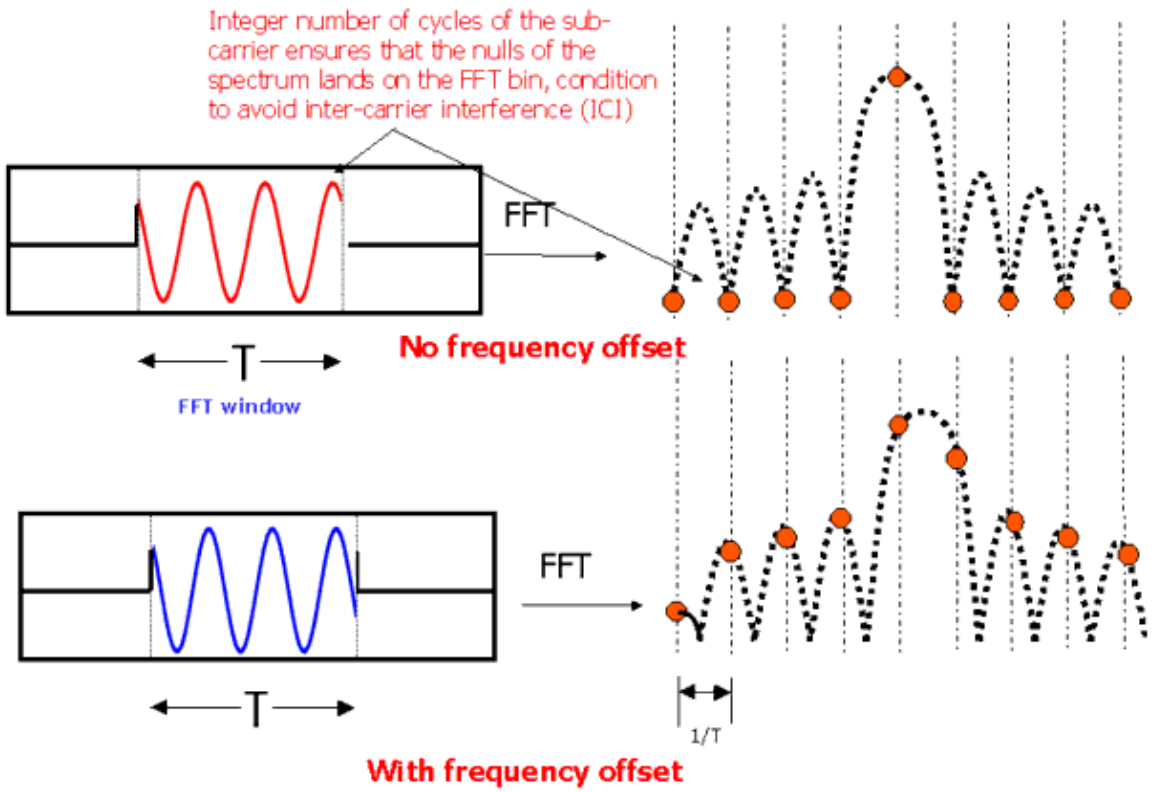

Figure 13: Inverse discrete Fourier transform

2.11 ADD CP (Cyclic Prefix)

Cyclic Prefixes, which are used in OFDM, in order to combat multipath, by making channel estimation easy. Length of CP is 25% of the symbol, which is placed in front of symbol. The main purpose of using CP is to remove the ISI.

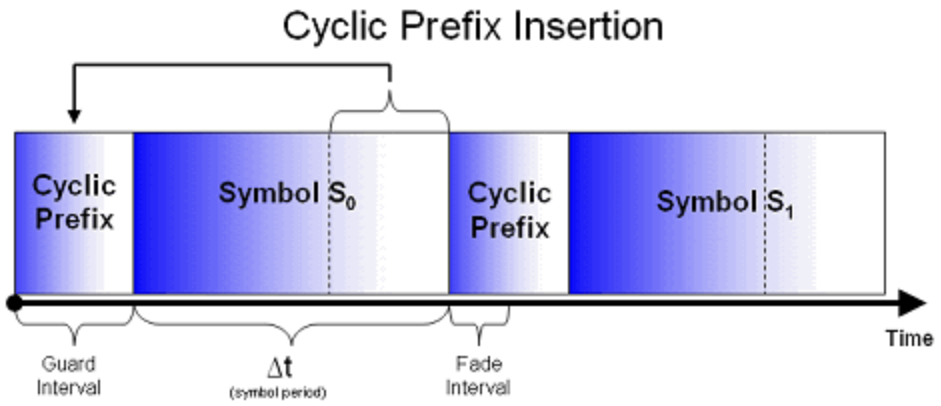

Figure 14: Insertion of cyclic prefix

2.12 RF front end

RF front end, has two basic operations; transmission and reception. The RF front end of Transmitter, that transmits the signal at the frequency of 2.4GHz.

Figure 15: USRP with front end

Chapter 3

LTE Based Receiver

3.1 Design

A simple LTE based receiver has been implemented, in order to receive the signal, transmitted, from LTE based transmitter. The basic purpose of implementing, this receiver is to minimize, the Bit Error Rate (BER) and maximize, the efficiency. For that purpose, the MMSE detection technique is used. Different LTE blocks are shown:

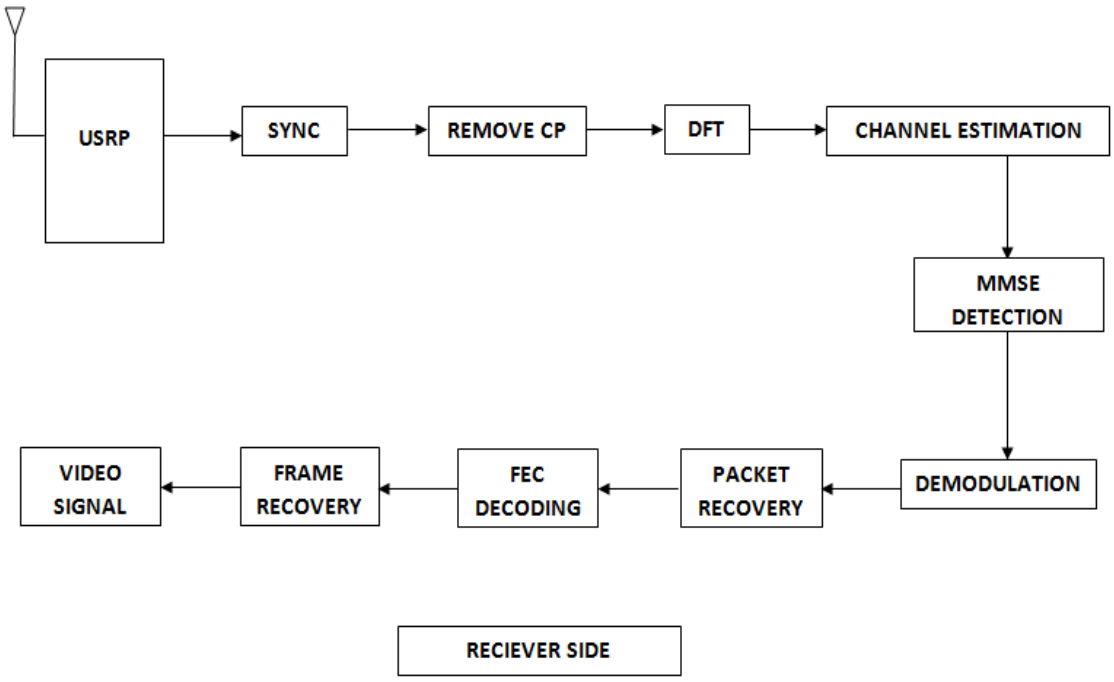

Figure 16: LTE Based Receiver

3.2 RF front end

At the receiver, RF front end receives the signal at the same frequency, i.e. 2.4G Hz. The purpose of receiver daughter board, is to convert the RF signal to base-band signal.

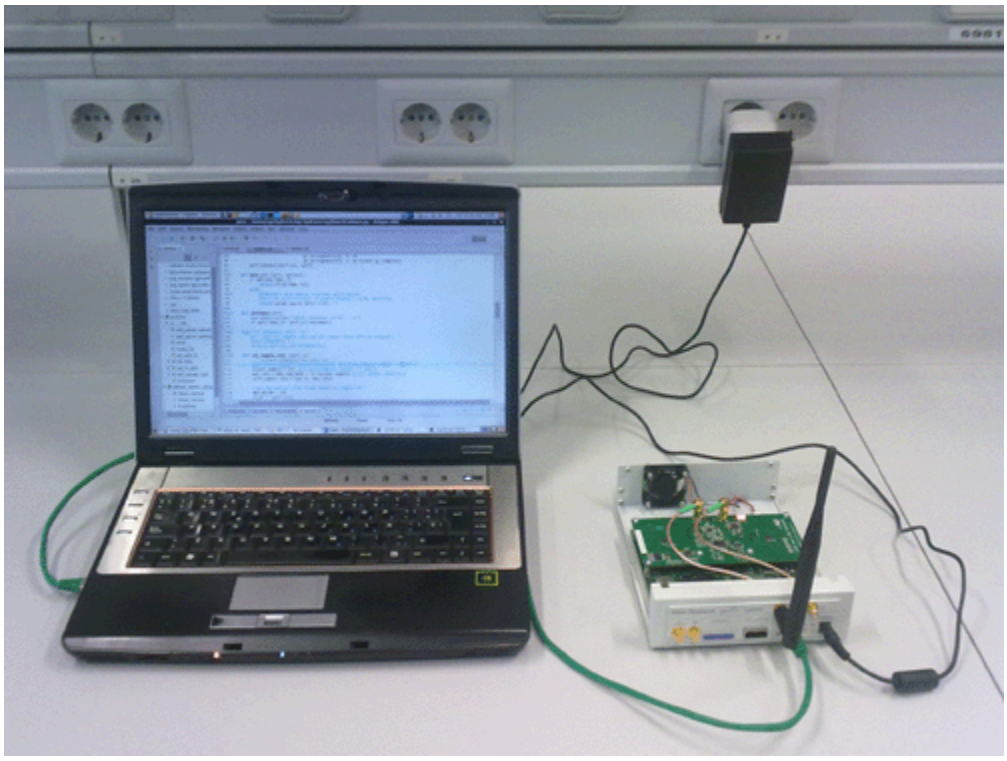

Figure 17: USRP Rx front end

3.3USRP

USRP, it is the main component of our project. Two daughter boards of RFX2400 and two antennas, that are placed on it. There are two basic sub-blocks of USRP; A/D or D/A and FPGA.

A/D is used to convert the analog signal to digital. The processing of wireless communication, is similar to digital communication,because, it is very easy to work with

the digital signal as compare to analog signal. That's why; first of all, analog signal is to convert into digital signal at the transmitter.

Similarly, D/A is used to convert the digital signal to analog . The input and output of USRP is analog signal.

The second sub-block of USRP is FPGA, all processing should be done in FPGA.

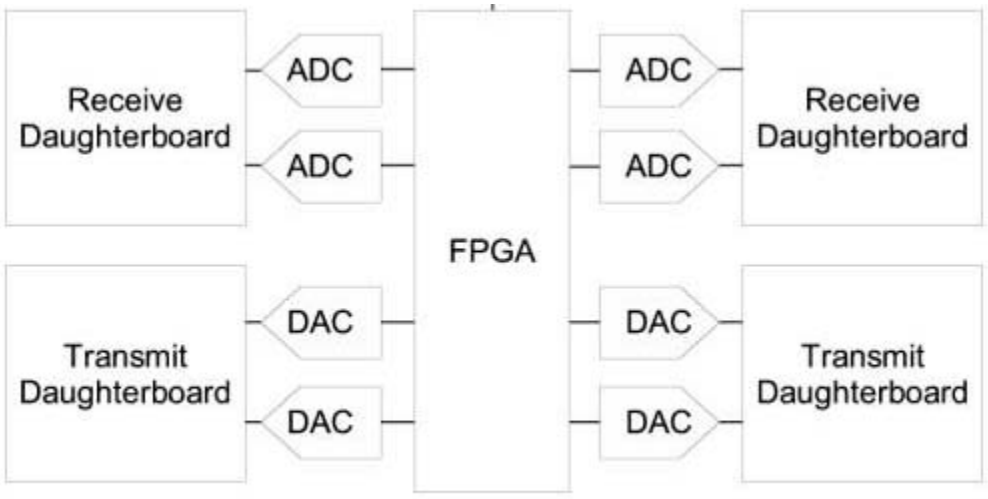

Figure 18: USRP

3.4 Synchronization

OFDM, requires synchronization in both the time and frequency. Time synchronization, involves finding the best possible time instant , for the start of received and down converted OFDM frame. Frequency synchronization, deals with finding an estimate of the difference in the frequencies, between the transmitter and receiver local oscillators.

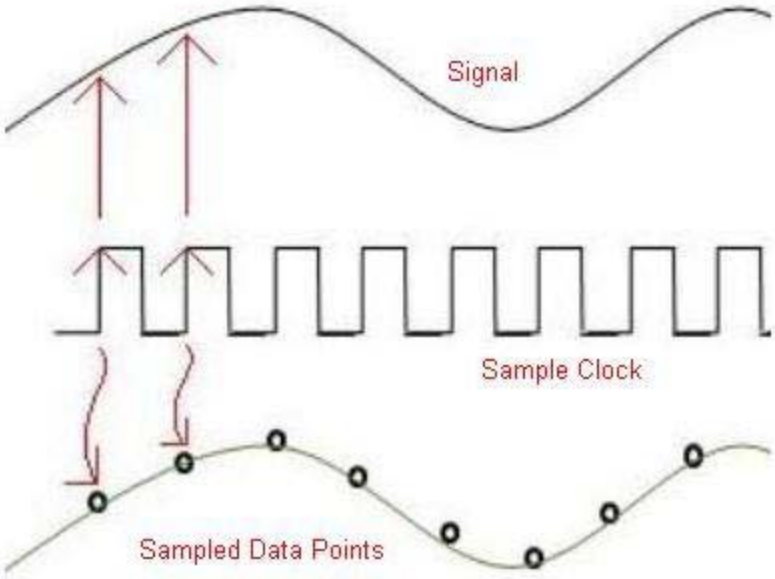

Figure 19: Time synchronization

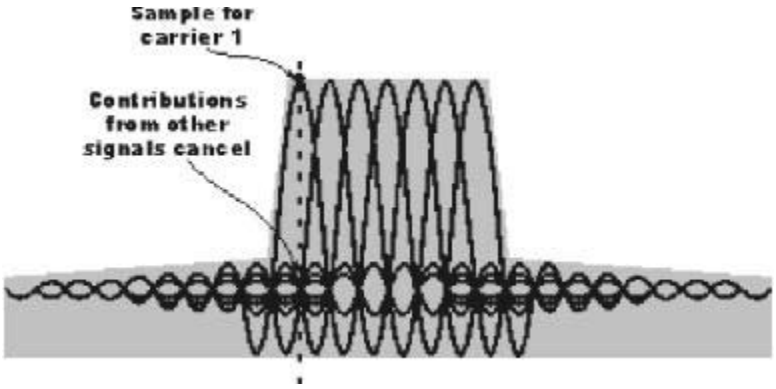

Figure 20: Frequency synchronization

3.5 Remove CP

The next step is to discard the cyclic prefix(CP), which is added at the transmitter side.

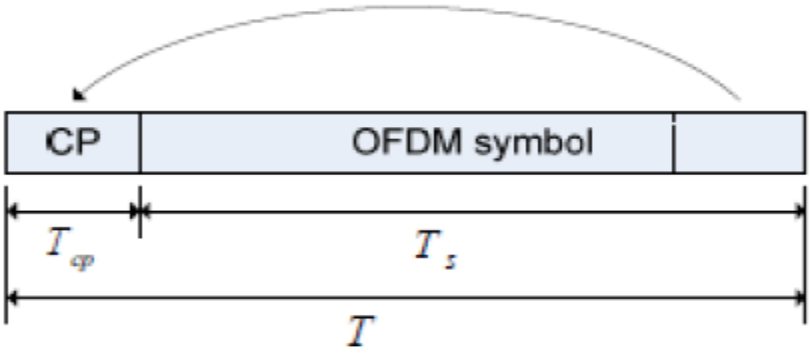

Figure 21: Remove cyclic prefix

3.6 DFT

The basic operation of using DFT is to represents the signal into frequency domain representation.

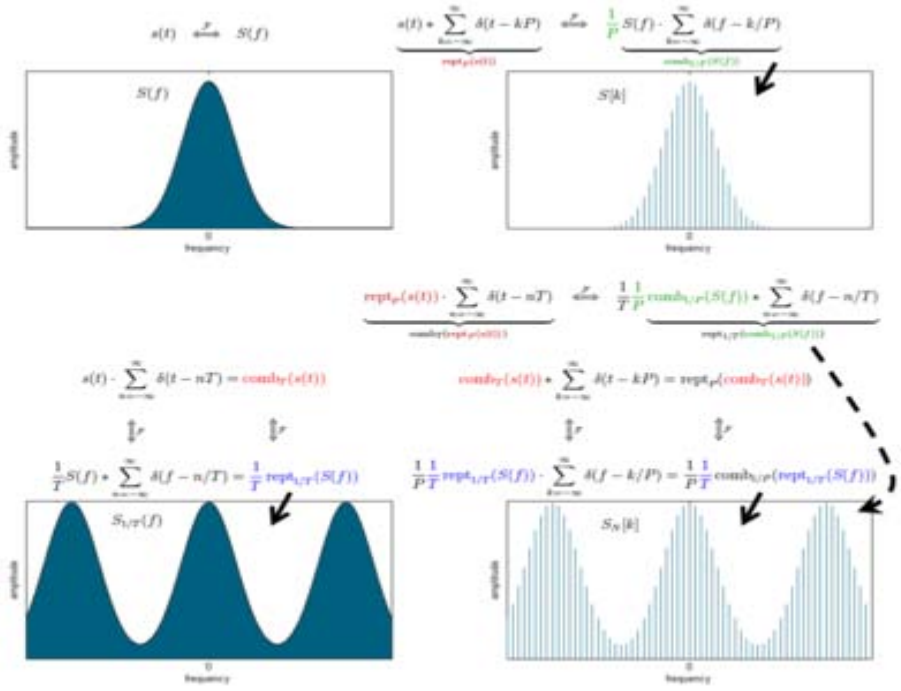

Figure 22: DFT

3.7 Channel estimation

In this step, channel estimation process have to be done by using the pilots and sequence bits comparision.

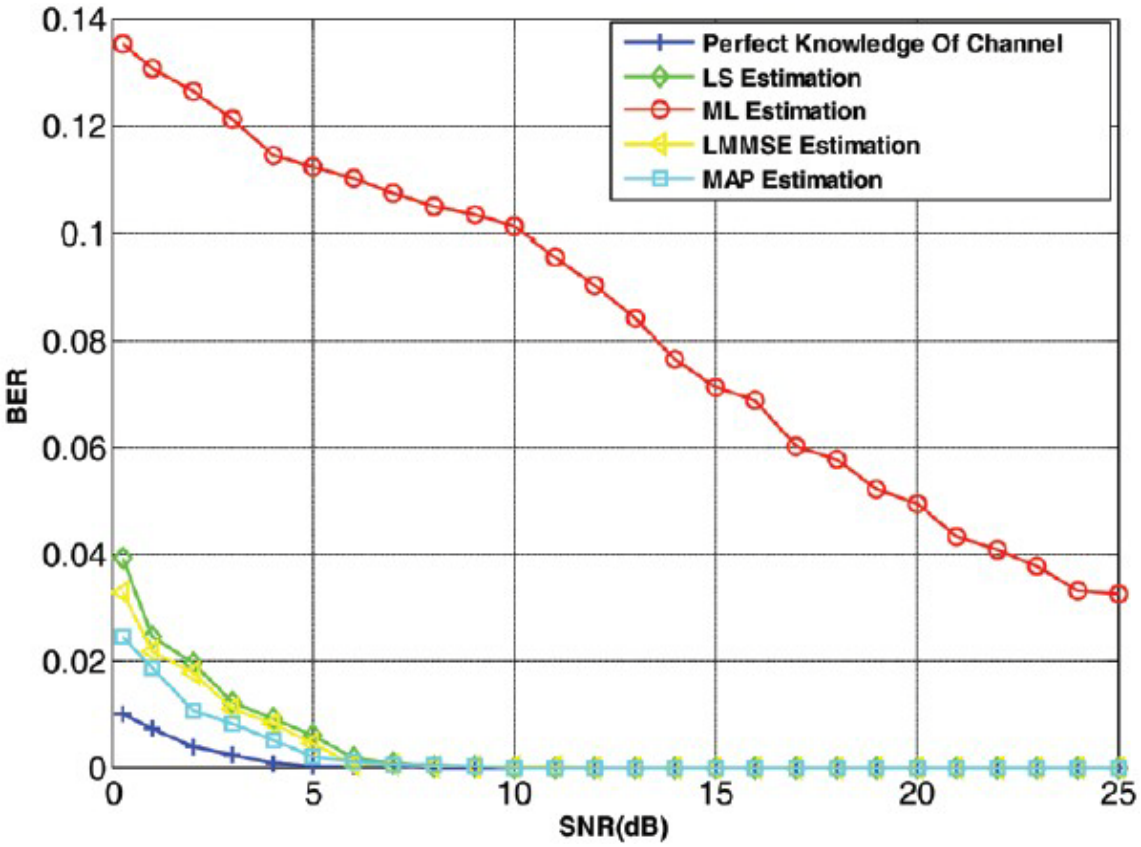

Figure 23: Channel estimation (BER)

3.8 Detection Techniques

There are different detection technique some of them are discussed.

3.8.1 Zero Forcing detection technique

Zero Forcing receiver, that cancels interference but in the process of cancelling the interference, the noise level is enhanced. This, then decreases SNR, so the BER performance is degraded.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & .. \\ h_{21} & h_{22} & .. \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \end{bmatrix} + \mathbf{n}$$

$$\hat{\mathbf{s}} = \mathbf{H}^{\#}\mathbf{y}$$

where

$$\mathbf{H}^{\#} = (\mathbf{H}^{H}\mathbf{H})^{-1}\mathbf{H}^{H}$$

This detection technique, is not efficient because, by applying this technique, the noise will be enhanced, and the performance of the system is degraded.

3.8.2 MMSE detection techniques

MMSE receiver, that is a better solution than ZF receiver. It does not completely cancel the interference, but it mitigates (attenuates ,i.e., decreases the power of interference).It

has better performance than ZF, and is a candidate receiver for 4G wireless systems. In this project, MMSE detection technique is used.

The MMSE receiver optimizes the following criteria:

$$\mathbf{W} := \operatorname{argmin}\{E|\mathbf{W}^H\mathbf{y} - \mathbf{s}|^2\}$$

We find:

$$\hat{\mathbf{s}} = \mathbf{H}^H(\mathbf{H}\mathbf{H}^H + \mathbf{R}_n)^{-1}\mathbf{y}$$

Where,Rn is Noise/Interference Covariance.

This technique is efficient, because, it completely mitigate the interference, and noise level will be remain same, so that's why, this has the better performance than the ZF receiver. In this project, MMSE technique is used, because the basic aim of this project, is to cancel the interference, since the interference mitigation technique, is used to mitigate the interference, which will be discussed in next chapter. The important role of using the interference mitigation technique is, to improve the performance of the system; including the minimum bit error rate(BER) and maximum data rate.

3.9 Demodulation

The basic aim of demodulation, is to extract the original signal information, by using different demodulation techniques including; GMSK,BPSK,QPSK etc. In this project, QPSK demodulation is used to recover the transmitted signal.

3.10 Packet recovery

In this step, packet will be recovered, after removing the flag; which is inserted at the transmitter.

3.11 FEC decoding

convolutional decoding is selected at the transmitter, the Viterbi decoder, is selected at the receiver to decode the data stream.

3.11.1 Viterbi decoder

At the transmitting side the FEC decoding had done, to decode the symbols, Viterbi decoder is used.The characteristics of Viterbi decoder is:

Table length: 35

States: 64

Bits: 6

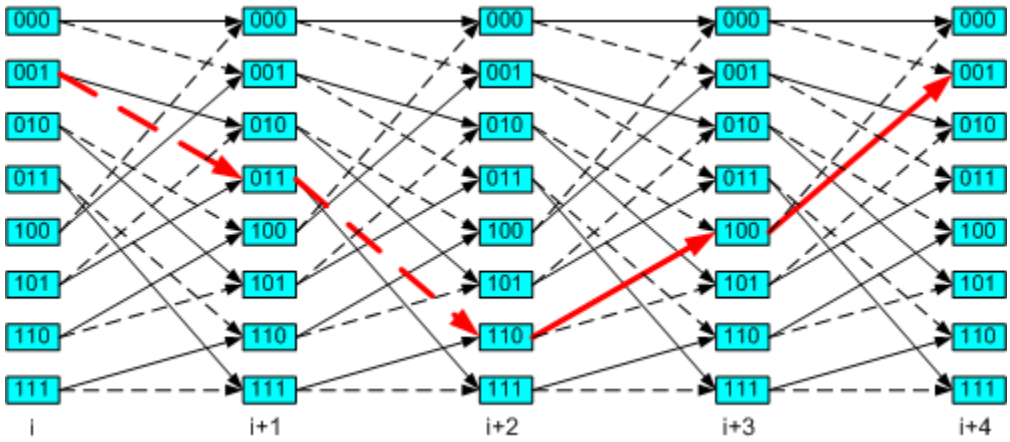

Figure 24: Trellis diagram

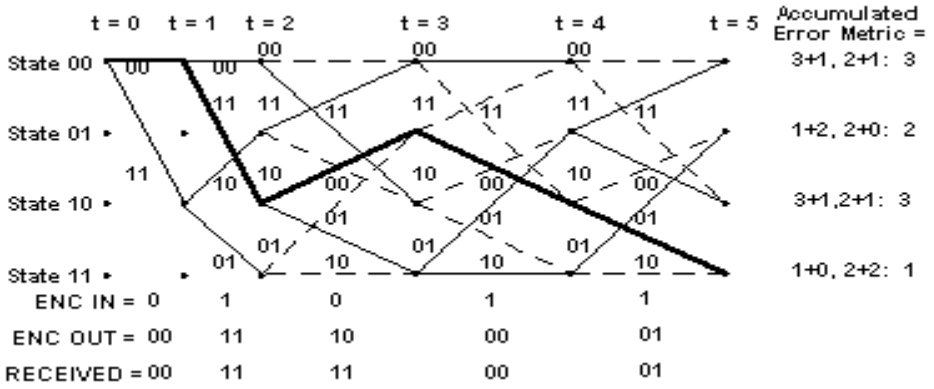

Figure 25: Viterbi decoder using trellis diagram

3.12 Frame recovery

At the second last stage, frames are recovered. Each Frame, that represents the image from the video, as it is mentioned in the transmitter. The frames are transmitted for the processing, here all these frames are recovered.

3.13 Video signal

At the final stage of the receiver, the real time video signal, which is transmitted from the webcam, is received.

Chapter 4

Interference Aware Receiver

4.1 Interference Scenario

This project focuses the interference scenario, which is shown in figure 4.1.

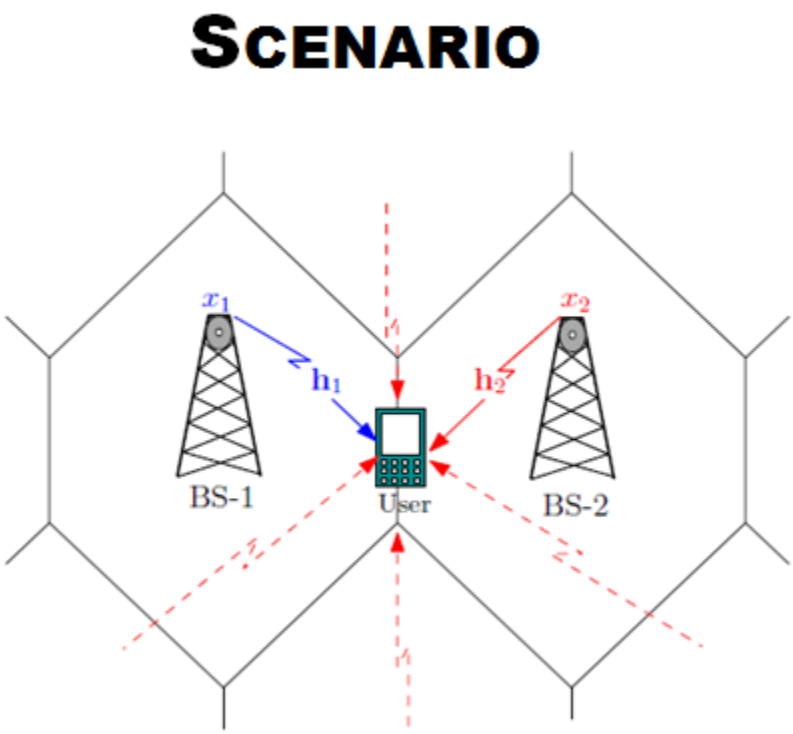

Figure 26: Interference Scenario

The considered scenario is similar to inter-cell interference, where transmitters are two adjacent base stations, while receiver is the cell-edge user. BS-1 and BS-2 are transmitting the video signals at the frequency of 2.4G Hz. Since,They are transmitting at

the same frequency, so they produce the interference at the cell edge user. The basic aim of this project is to use the interference mitigation technique, to cancel out the interference at the cell edge user. Here, Two USRP are transmitting the video signal, and the third USRP is able to receive the Video signal, but due to interference, the video signal will be distorted and the communication will be effected. The second important factor is to BER, due to interference, the BER of the system is increased, and data rate is decreased.

This complete information is necessary for development of the effective communication system. Since our basic aim is to achieve the quality communication therefore, interference mitigation techniques are employed, to decrease the bit error rate(BER) and enhance the data rate of the communication system.

4.2 Interference Mitigation Techniques

There are different interference mitigation techniques, to mitigate the interference and attain the quality communication. Important techniques are given below:

- Adaptive System Identification(ASI)
- Adaptive Noise Cancellation(ANC)
- Adaptive Linear Prediction(ALP)
- Adaptive Inverse System(AIS)

All these techniques are efficient, in this project, third technique Adaptive Linear Prediction is used to mitigate the interference, because this technique is most commonly used in the communication to reduce the noise level and interference level of the system. The mathematical calculation of, signal to interference noise ratio (SINR) is mainly

depending upon, the signal level and interference level of the system. There are three similar terms, which are used to refer the interference including noise, interference anddistortion of the signal. The noise is "Additive White Gaussian Noise", which is mainly known as the surrounding noise.

The interference is the signal, which is generated from the source. The main function of this interference is, to disrupts or alters the signal. Here is the similar case, because the interference signal is generated from the source, called the USRP. Since SISO link is established between the transmitter-1 and receiver. When transmitter-2 transmits the signal at the same time, then there will be interference at the receiver, to mitigate this interference, Adaptive filter algorithm is used.

4.3 Adaptive filter Algorithm

As, it is mentioned above, this is very efficient technique, because the need is the quality communication. The algorithm is depending upon two basic signals including transmitted signal from transmitter-1 and transmitted signal from transmitter-2. Since the detection process will be done on the receiver, so the problem is, how to differentiate both signals. For that purpose, the different pilot bits are to be added with the both transmitting signal. Both the pilot signals are helping to synchronize the signal. This whole process will be done by using this algorithm, as shown in figure4.2.

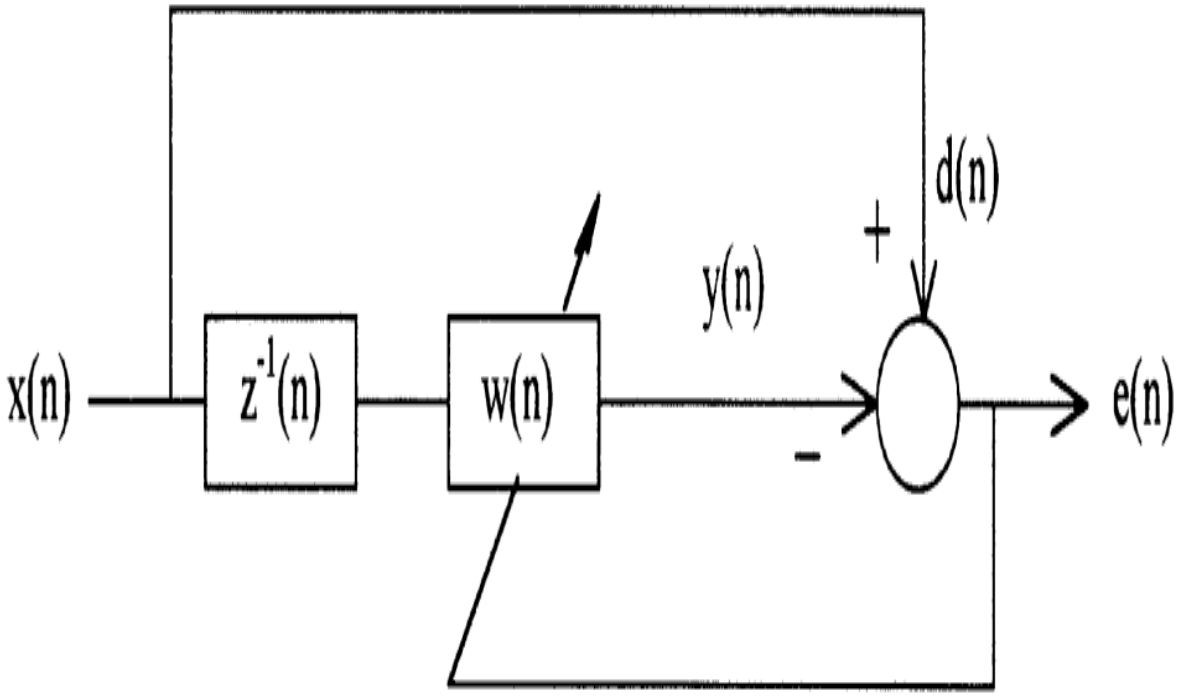

Figure 27: Adaptive filter prediction

The above figure shows:

x(n), which represents the input signal,

$Z^{-1}(n)$, represents the inverse of input signal,

d(n) is the desired signal,

and e(n) is the error.

This configuration essentially performs two basic operations:

1- First operation: If the output is taken from the error signal e(n), is linear prediction. The adaptive filter coefficients are being trained to predict, from the statistics of the input signal x(n), what the next input signal will be.

2- Second operation: If the output is taken from y(n), is a noise filter.

4.4 Advantages of Adaptive filter prediction

1- Decrease the noise and interference level

2- Signal prediction

3- Adaptive feedback cancellation

4- Echo cancellation

Chapter 5

Summary and Results

5.1 Summary

This project develops an interference scenario, in which two LTE transmitters and one Advanced LTE interference aware receiver is implemented. Two transmitters are transmitting the real time video signal at the frequency of 2.4G Hz and an advanced LTE interference aware receiver are receiving the real time video signal at the frequency of 2.4G Hz. This scenario is similar to inter-cell interference. The main focus of this project, is to cancel out the interference at the cell edge user, which is able to receive the signal at the same time, resulting the interference is generated. To apply the interference mitigation technique, to achieve the goal of this project or in other words, it would be an implementation of interference aware receiver. The factors, which is necessary for developing the complete wireless communication system, is to minimize the Bit error rate (BER) and enhanced the date rate, which will be discussed in the next section.

5.2 Results

5.2.1 Matlab Results

5.2.1.1 Zero forcing Receiver

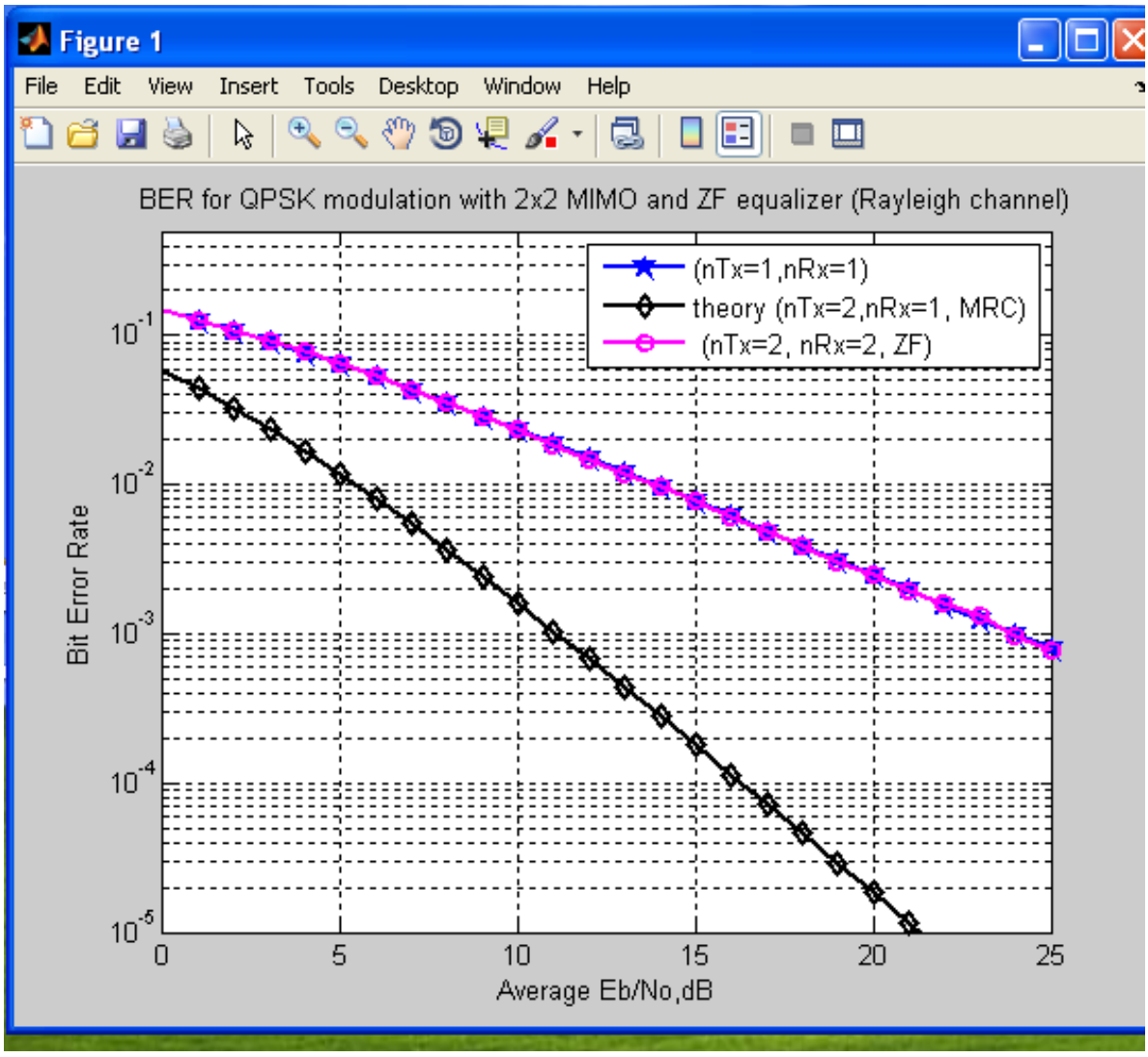

Figure 28: Bit Error Rate graph of Zero Forcing

5.2.1.1MMSE Receiver

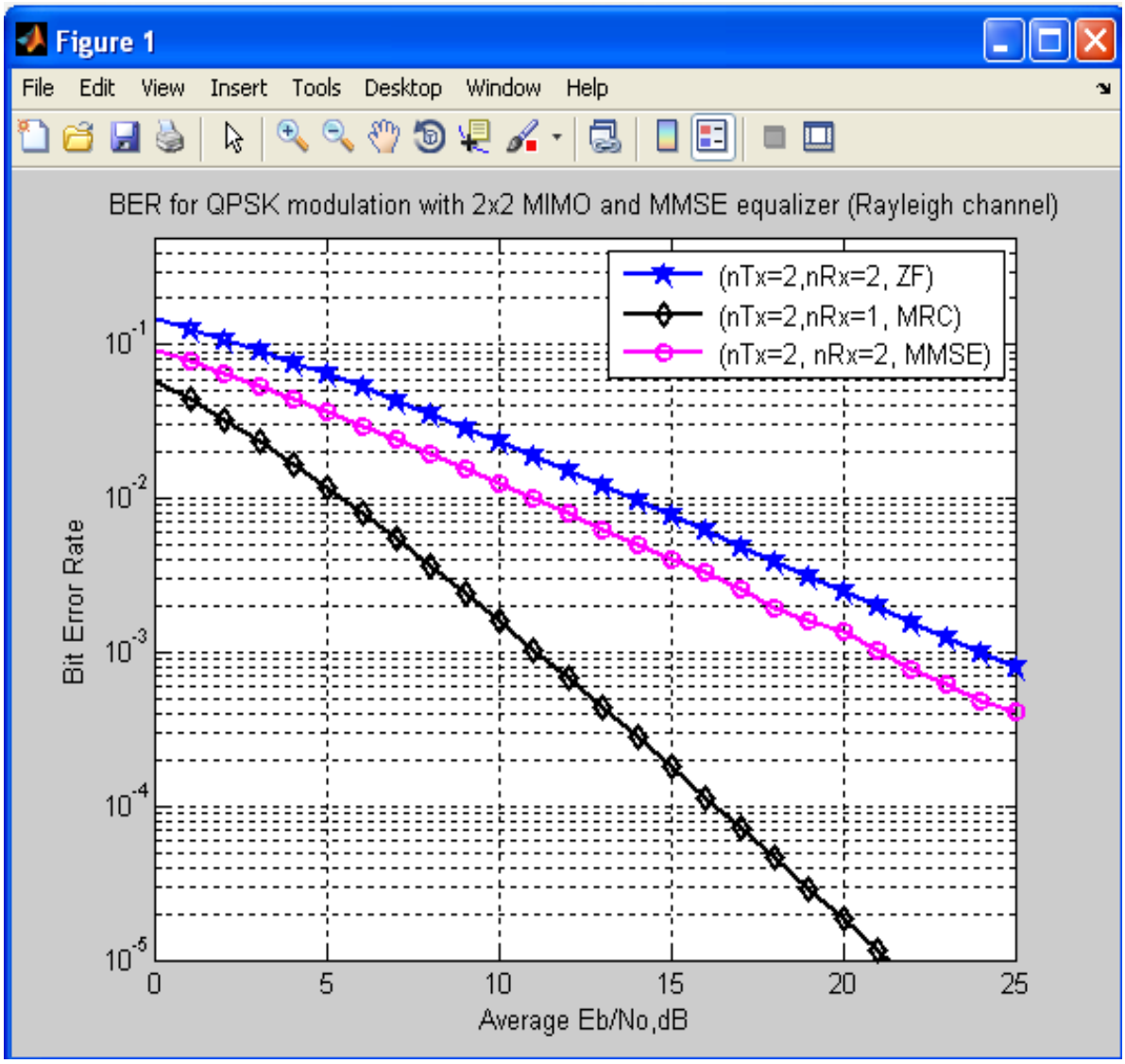

Figure 29: Bit Error Rate graph of MMSE Receiver

5.2.2 GNU Radio Results

5.2.2.1 Establishment of SISO link

Transmitter side

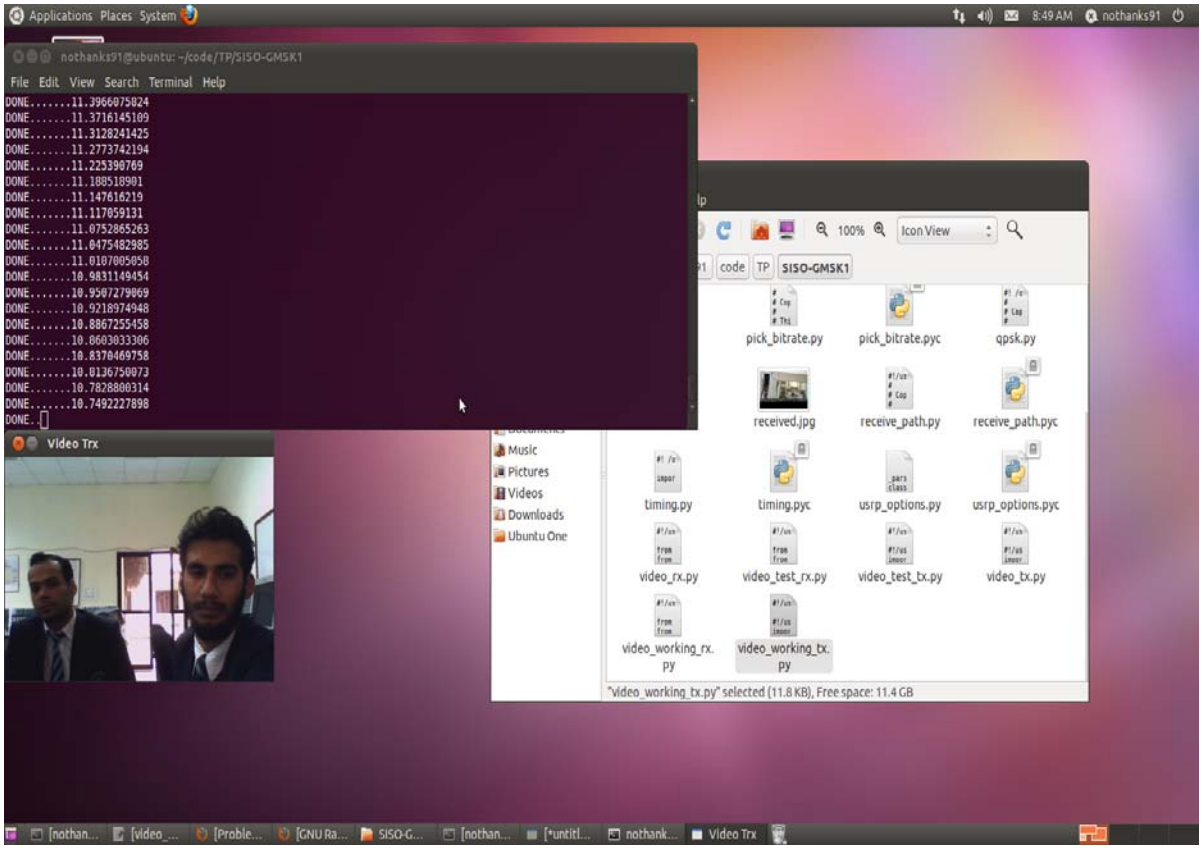

Figure 30: SISO Link Transmitter

Video is transmitting at the rate of 10 frame per seconds.

Receiver Side

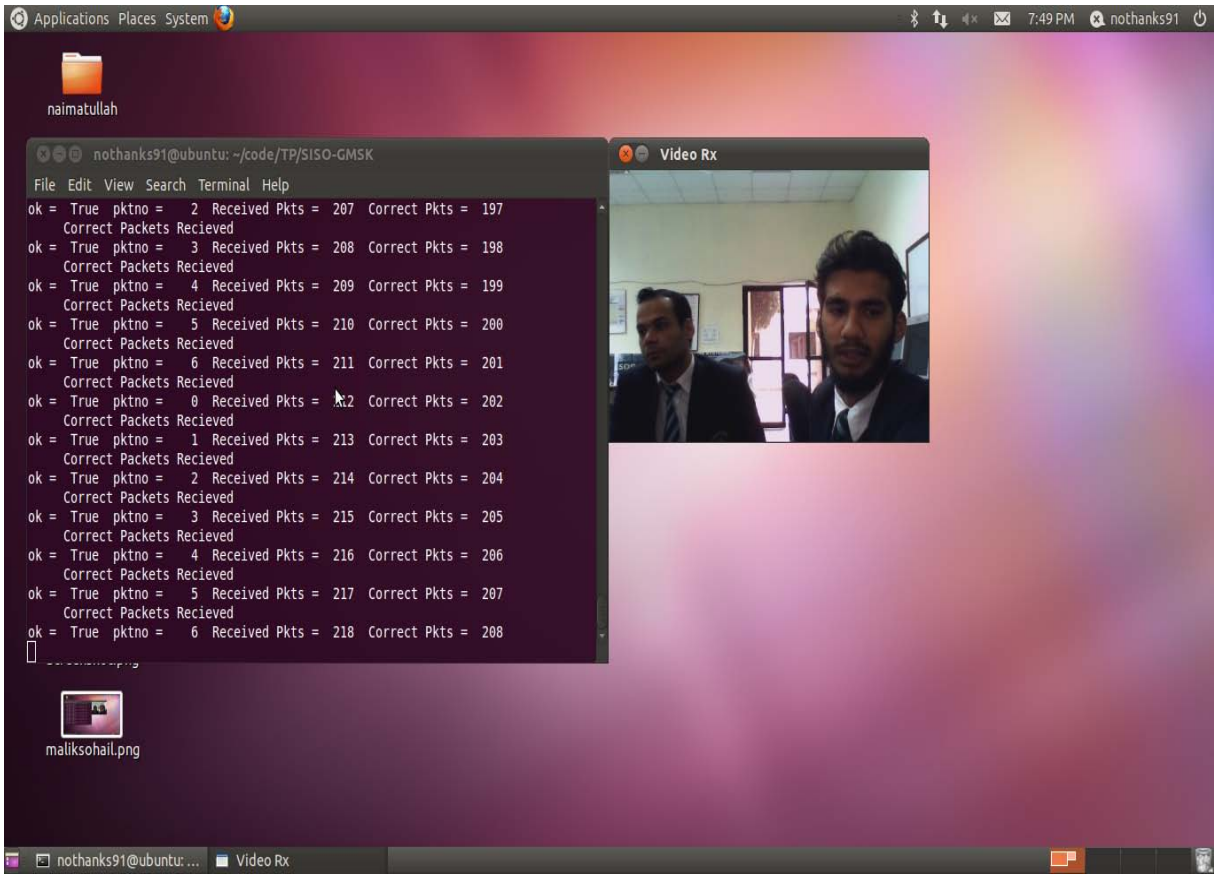

Figure 31: SISO link Receiver

Video is received at the rate of 10 FPS, since the received packets are 218 and correct packets are 208.

APPENDIX A

```
fromgnuradio import gr

fromgnuradio import usrp

fromgnuradio import eng_notation

defxor(a,b):

if a==b:

return 0

else:

return 1


f = open('Input.txt','r') # open in read mode

payload = f.read()       # reading the data

#print (len(data))

#print (ord(data[0]))

#count_payload=2;

count_payload=0;

print ('_____')

print ('length of actual data : ',len(payload))

print ('_____')

payload_data=[];

payload_data_string=[];

whilecount_payload<len(payload):

print ('character : ',payload[count_payload])
```

```
binary=ord(payload[count_payload]);      #ord function returns the ascii value of any
charactor
print ('ascii code : ',binary)
count=1;
b_number=[];
number=0;
binary=int(binary)
while binary >= 1:
number=binary%2;
      #print (number)
      #number_str=str(number);
number_str=int(number)
b_number.append(number_str);
binary=binary/2;
binary=int(binary)
         #if binary==1:
            #b_number.append(1)
count_add=0;
whilecount_add<6:
iflen(b_number)!=7:
b_number.append(0);
count_add=count_add+1;
count=count+1;
```

```
    #print count
    #print b_number[count-2]
    #print ('reverse binary code : ',b_number)
    #print ('length of reverse binary code : ',len(b_number))
cnt=0;
    #print b_number[cnt]
b_number_new=[];
count_b_number=len(b_number)-1;
    #print (count_b_number)
whilecount_b_number>=0:
            #print b_number[count_b_number]
payload_data.append(b_number[count_b_number])
payload_data_string.append(str(b_number[count_b_number]))
b_number_new.append(b_number[count_b_number])
count_b_number=count_b_number-1;
print ('length of binary code : ',len(b_number_new))
print ('binary code : ',b_number_new)
count_payload=count_payload+1;
print ('_____')
    #payload_data.append(b_number_new)
    #print ('binary code : ',b_number_new)
    #while i
    #print binary_number
```

```
    #print (ord(payload[2]),ord(payload[3]),ord(payload[4]),ord(payload[5]))
    #print bits(ord('p'))
    #binary = []
    #for char in "abcdefgh":
        #binary.append(ascii_to_bin(char))
    #print binary
    #print " ".join(binary)
    #print bytes([payload])
    #print payload[1]
    #print payload[2]
print('length of actual data : ',len(payload_data))
print ('data in bits : ',payload_data)
payload_data_string_count=0;
whilepayload_data_string_count<len(payload_data_string):
with open("Input_bits.txt", "a") as myfile:
myfile.write(payload_data_string[payload_data_string_count])
payload_data_string_count=payload_data_string_count+1;
print ("Done!")
msg_encode_output_count=0;
msg_encode_output1=[0,0,0,0,0,0,0,0];
msg_encode_output2=[0,0,0,0,0,0,0,0];
msg_encode_output=[];
'''
```

```
whilemsg_encode_output_count<=6:
    msg_encode_output1[msg_encode_output_count]=0;
    msg_encode_output2[msg_encode_output_count]=0;
msg_encode_output_count=msg_encode_output_count+1;
'''
#print ('bits',payload_data[0])
msg_transfer=[];
msg_encode_count=0;
whilemsg_encode_count<len(payload_data):
    msg_encode_output1[6]=msg_encode_output1[5];
    msg_encode_output1[5]=msg_encode_output1[4];
    msg_encode_output1[4]=msg_encode_output1[3];
    msg_encode_output1[3]=msg_encode_output1[2];
    msg_encode_output1[2]=msg_encode_output1[1];
    msg_encode_output1[1]=msg_encode_output1[0];
    msg_encode_output1[0]=payload_data[msg_encode_count];
    xor_a1=xor(msg_encode_output1[6],msg_encode_output1[5]);
    xor_b1=xor(msg_encode_output1[3],msg_encode_output1[2]);
    xor_c1=xor(xor_a1,xor_b1);
    xor_d1=xor(xor_c1,msg_encode_output1[0]);
    xor_a2=xor(msg_encode_output1[0],msg_encode_output1[1]);
    xor_b2=xor(msg_encode_output1[3],msg_encode_output1[2]);
    xor_c2=xor(xor_a2,xor_b2);
```

```
    xor_d2=xor(xor_c2,msg_encode_output1[6]);
msg_transfer.append(str(xor_d1))
msg_transfer.append(str(xor_d2))
msg_encode_output.append(xor_d1)
msg_encode_output.append(xor_d2)
msg_encode_count=msg_encode_count+1;
print ('encoder output length : ',len(msg_encode_output))
print ('encoder output : ',msg_encode_output)
msg_transfer_count=0;
f1=open("encoded.txt", "a")
f1.truncate();
i=0;
whilemsg_transfer_count<len(msg_transfer):
f1.write(msg_transfer[msg_transfer_count])
msg_transfer_count=msg_transfer_count+1;
f1.close();
```

APPENDIX B

```
fromgnuradio import gr

fromgnuradio import usrp

fromgnuradio import eng_notation

defdecimal_to_binary(decimal_number):

decimal_number=int(decimal_number);

binary_reverse_value=[];

whiledecimal_number>= 1:

binary_value=decimal_number%2;

binary_reverse_value.append(int(binary_value));

decimal_number=int(decimal_number/2);

count_padding_zeros=0;

whilecount_padding_zeros<=6:

iflen(binary_reverse_value)!=7:

binary_reverse_value.append(0);

count_padding_zeros=count_padding_zeros+1;

binary_forward_value=[];

count_binary_forward_value=len(binary_reverse_value)-1;

whilecount_binary_forward_value>=0:

    binary_forward_value.append(binary_reverse_value[count_binary_forward_value]);

count_binary_forward_value=count_binary_forward_value-1;

return(binary_forward_value)
```

```
defxor(a,b):
if a==b:
return 0
else:
return 1

defmsg_encoder(data):

count_msg_encoder=0;
encoder_register=[0,0,0,0,0,0,0];
msg_encoder_output=[];

whilecount_msg_encoder<len(data):

encoder_register[6]=encoder_register[5];
encoder_register[5]=encoder_register[4];
encoder_register[4]=encoder_register[3];
encoder_register[3]=encoder_register[2];
encoder_register[2]=encoder_register[1];
encoder_register[1]=encoder_register[0];
encoder_register[0]=data[count_msg_encoder];

      xor_a1=xor(encoder_register[6],encoder_register[5]);
```

```
        xor_b1=xor(encoder_register[3],encoder_register[2]);
        xor_c1=xor(xor_a1,xor_b1);
        U1=xor(xor_c1,encoder_register[0]);

        xor_a2=xor(encoder_register[0],encoder_register[1]);
        xor_b2=xor(encoder_register[3],encoder_register[2]);
        xor_c2=xor(xor_a2,xor_b2);
        U2=xor(xor_c2,encoder_register[6]);

msg_encoder_output.append(U1);
msg_encoder_output.append(U2);

count_msg_encoder=count_msg_encoder+1;

return (msg_encoder_output)

defactual_bits(register_state):

register_state[0]=0;

    xor_a1=xor(register_state[6],register_state[5]);
    xor_b1=xor(register_state[3],register_state[2]);
    xor_c1=xor(xor_a1,xor_b1);
```

```
    U1_output_0=xor(xor_c1,register_state[0]);

    xor_a2=xor(register_state[0],register_state[1]);
    xor_b2=xor(register_state[3],register_state[2]);
    xor_c2=xor(xor_a2,xor_b2);
    U2_output_0=xor(xor_c2,register_state[6]);

register_state[0]=1;

    xor_a1=xor(register_state[6],register_state[5]);
    xor_b1=xor(register_state[3],register_state[2]);
    xor_c1=xor(xor_a1,xor_b1);
    U1_output_1=xor(xor_c1,register_state[0]);

    xor_a2=xor(register_state[0],register_state[1]);
    xor_b2=xor(register_state[3],register_state[2]);
    xor_c2=xor(xor_a2,xor_b2);
    U2_output_1=xor(xor_c2,register_state[6]);


return (U1_output_0,U2_output_0,U1_output_1,U2_output_1)

defhamming_distance(actual_data_bits,recieved_data_bits):
```

```
ifactual_data_bits[0]==recieved_data_bits[0] and
actual_data_bits[1]==recieved_data_bits[1]:
    a=0;
elifactual_data_bits[0]==recieved_data_bits[0] and
actual_data_bits[1]!=recieved_data_bits[1]:
    a=1;
elifactual_data_bits[0]!=recieved_data_bits[0] and
actual_data_bits[1]==recieved_data_bits[1]:
    a=1;
elifactual_data_bits[0]!=recieved_data_bits[0] and
actual_data_bits[1]!=recieved_data_bits[1]:
    a=2;
ifactual_data_bits[2]==recieved_data_bits[0] and
actual_data_bits[3]==recieved_data_bits[1]:
    b=0;
elifactual_data_bits[2]==recieved_data_bits[0] and
actual_data_bits[3]!=recieved_data_bits[1]:
    b=1;
elifactual_data_bits[2]!=recieved_data_bits[0] and
actual_data_bits[3]==recieved_data_bits[1]:
    b=1;
```

```
elifactual_data_bits[2]!=recieved_data_bits[0] and actual_data_bits[3]!=recieved_data_bits[1]:
    b=2;

return (a,b)



defmsg_decoder(data):
count_msg_decoder_table_length=0;
table_length_number=0;

actual_bits_array=[[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[
],[],[],[],[]]];
data_index_count=0;

hamming_distance_output=[[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[
],[],[],[],[],[],[],[],[]]];
hamming_distance_output_index=0;
whilecount_msg_decoder_table_length<70:
count_msg_decoder_each_table_length=0;
count_msg_decoder_each_table_length_state=0;
msg_encoder_each_table_length_state=0;
    recieved_data_bits=[data[data_index_count],data[data_index_count+1]];
```

```
whilecount_msg_decoder_each_table_length<2**table_length_number:

msg_decoder_each_table_length_state=int((64/(2**table_length_number))*count_msg_decoder_each_table_length);

whole_register_value=decimal_to_binary(msg_decoder_each_table_length_state);
        #print ('msg_encoder_each_table_length_state',msg_decoder_each_table_length_state)
        #print ('whole_register_value',whole_register_value)
actual_data_bits=actual_bits(whole_register_value);

actual_bits_array[0][hamming_distance_output_index].append(actual_data_bits[0]);

actual_bits_array[0][hamming_distance_output_index].append(actual_data_bits[1]);

actual_bits_array[0][hamming_distance_output_index].append(actual_data_bits[2]);

actual_bits_array[0][hamming_distance_output_index].append(actual_data_bits[3]);

hamming_distance_output[0][hamming_distance_output_index].append(hamming_distance(actual_data_bits,recieved_data_bits)[0])
```

```
hamming_distance_output[0][hamming_distance_output_index].append(hamming_distan
ce(actual_data_bits,recieved_data_bits)[1])

count_msg_decoder_each_table_length=count_msg_decoder_each_table_length+1;

count_msg_decoder_table_length=count_msg_decoder_table_length+2;
iftable_length_number< 7:
table_length_number=table_length_number+1;
eliftable_length_number ==7:
table_length_number=7;
data_index_count=data_index_count+2;
hamming_distance_output_index=hamming_distance_output_index+1;
print ('data',data)
   #print ('actual_bits',actual_bits_array)
count_display_actual_bits_array_output=0;
whilecount_display_actual_bits_array_output<35:
print ('Actual Bits',actual_bits_array[0][count_display_actual_bits_array_output]);

count_display_actual_bits_array_output=count_display_actual_bits_array_output+1;
count_display_hamming_distance_output=0;
whilecount_display_hamming_distance_output<35:
```

```
print ('Hamming
Distance',hamming_distance_output[0][count_display_hamming_distance_output]);

count_display_hamming_distance_output=count_display_hamming_distance_output+1;
count_path_metric_outer=0;

path_metric=[[[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]
,[],[]]];
table_length_path_metric=1;
whilecount_path_metric_outer<35:
count_path_metric_inner=1;
whilecount_path_metric_inner<=2**table_length_path_metric:
count_path_metric_inner=count_path_metric_inner+1;
count_path_metric=count_path_metric=1;
iftable_length_path_metric< 7:
table_length_path_metric=table_length_path_metric+1;
eliftable_length_path_metric ==7:
table_length_path_metric=7;

data=[0,1,1,1,0,1,0,1,1,0,0,0,1,1,0,1,1,0,0,1,1,0,1,1,1,1,1,0,1,0,0,0,0,0,0]
#print(len(msg_encoder(data)))
encoder_output=msg_encoder(data);
msg_decoder(encoder_output)
```