Al-Azhar University
Faculty Of Engineering
Communication and Electronics Dep.

SOFTWARE IMPLEMENTATION OF LTE PHYSICAL LAYER 4G.

Prepared by:

Muhammad Ahmed Mahmoud Ahmed Khedawy

Muhammad Magdy Soliman Hamda

Ahmed Muhammad Abdulrahman Elwan

Muhammed Soliman Alsaid Soliman

Ahmed Hamdy Abdelfattah Saber Alesh

Muhamed Saied Muhammed Osman

SUPERVUSOR BY

Dr.Ahmed Abd Al Rhman

Study & Software Implementation of LTE. Physical Layer 4G

دراسة وتنفيذ برنامج الطبقة الفيزيائية لشبكات الجيل الرابع

July 2021

Team Contacts

Mohamed Ahmed Mahmoud Ahmed Khedawy

muhammadkhedawy@outlook.com

Mohamed Magdi Soliman Hamda

Mohamedmagdi940@gmail.com

Ahmed Mohamed Abdulrahman Elwan

ahmadmohabdelrhman@gmail.com

Ahmed Hamdy Abdelfattah Saber Alesh

eleshahmed562@gmail.com

Mohamed Soliman Alsaid Soliman Hassab

mohamedsolimanhassab@gmail.com

Mohamed Saied Mohamed Osman

MohamedSaaied.2020@azhar.edu.eg

Acknowledgments

This book was written during our B.Sc.-year time at the Department of Electronics and Communications Engineering at the Al-Azhar University and basically describes our work and study in graduation project. Certainly, it could not have been written without the support and patience of many people. Therefore, we're obliged to everyone who assisted us during that time. In particular, we want to express our gratitude to our supervisor **Dr.Ahmed Emran** for all the valuable advice, encouragement, and discussions. The opportunity to work with him was a precious experience, he exerts all the effort and time to help us to learn, search, and do our best in this project.

Also, we want to thank our Professors in the communications department, who made their best to teach us the soul of Communication and electronic Engineering. Most of all, we thank our beloved family for their immeasurable support, encouragement, and patience while working on this project. Without their love and understanding, this book and our project would not have come to fruition.

At the end and the beginning, we would be remiss if we fail to express our profound gratitude to Allah who always we're asking for his assistance and we're owing to him with any success and progress we made in our life.

Preface

Market needs for higher data rates are driving the evolution of wireless cellular systems from narrowband 2G GSM systems to 4G LTE systems supporting peak data rates up to 100 Mbps. For LTE specifications, complex signal processing techniques such as multiple-input multiple output (MIMO), along with radio technologies like OFDMA, are considered key to achieving target throughputs in excess of 100 Mbps In-building coverage is also regarded as a key requirement for future wireless growth, with technologies such as Pico and Femto base stations trying to address this issue. The emerging wireless technologies described above pose significant challenges for operating equipment manufacturers needing to design products that are not only scalable and cost-effective but also flexible and reusable. These diverse requirements ultimately make FPGA the hardware platform of choice. Cyclic redundancy check (CRC) code provides a simple, yet powerful, method for the detection of burst errors during digital data transmission and storage. CRC implementation can use either hardware or software methods. The aim of my project is to implement CRC according to LTE.3GPP standards.

Abbreviations

3GPP	Third Generation Project Partnership
4G	Fourth generation
64-QAM	64Quadrature Amplitude Modulation
AMC	Adaptive Modulation and Coding
AWGN	Additive White Gaussian Noise
BCH - Code	Bose–Chaudhuri–Hocquenghem codes
BER	Bit Error Rate
BPSK	Binary phase shift keying
CBRM	Circular Buffer Rate Matching
CCE	Control Channel Element
CDD	Cyclic Delay Diversity
CP	Cyclic Prefix
CRC	Cyclic Redundancy Check
CSI	Channel-State Information
DFT	Discrete Fourier Transform
DPSK	Differential Phase Shift King
EPDCCH	Enhanced Physical Downlink Control Channel
EREG	Enhanced Resource-Element Group
FDD	Frequency Division Duplex
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
ICI	Inter-Carrier Interference
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
ISI	Inter-Symbol Interference
ITU-R	International Telecommunication Union - Radiocommunication Sector
LLR	Log Likelihood Ratio
LTE	Long Term Evolution
LTE-A	Long Term evolution Advanced

MIMO	Multiple Input Multiple Output
N_{data}	Number of Coded Bits
NPDCCH	Narrowband Physical Downlink Control Channel
OFDMA	Orthogonal Frequency Division Multiple Access
PAPR	Peak-to-Average Power Ratio
PCFICH	Physical Control Format Indicator Channel
PDCCH	Physical Downlink Control Channel
PDF	Probability Density Function
PDSCH	Physical Downlink Shared Channel
PHY	Physical Layer
PMI	Precoding Matrix Index
PRB	Physical Resource Block
PRG	Precoding Resource Block Group
PSF	Pulse Shaping Filter
QPP	Quadrature Permutation Polynomial
RAN	Radio Access Network
RE	Resource Element
REG	Resource-Element Group
RLC	Radio Link Control
RM	Rate Matching
SINR	Signal to Interference and Noise Ratio
SNR	Signal to Noise ratio
SPDCCH	Short Physical Downlink Control Channel
TB	Transport Bocks
TDD	Time Division Duplex
ZF	Zero Forcing

Contents

List of Figures

List of Tables

1 Introduction

LTE (Long Term Evolution), marketed as 4G LTE, is a standard for wireless communication of high-speed data for mobile phones and data terminals. It is based on the GSM/EDGE and UMTS/HSPA network technologies, increasing the capacity and speed using a different radio interface together with core network improvements. The standard is developed by the 3GPP (3rd Generation Partnership Project) and submitted to ITU-T in 2009.

LTE, is based on OFDMA (Orthogonal Frequency Division Multiple Access) to be able to reach even higher data rates and data volumes. High order modulation (up to 64QAM), large bandwidth (up to 20 MHz) and MIMO transmission in the downlink (up to 4x4) is also a part of the solution. The highest theoretical data rate is 170 Mbps in uplink and with MIMO the rate can be as high as 300 Mbps in the downlink.
To achieve high radio spectral efficiency a multicarrier approach for multiple access was chosen by 3GPP. For the downlink, OFDMA (Orthogonal Frequency Division Multiple Access) was selected and for the uplink SC-FDMA (Single Carrier - Frequency Division Multiple Access) also known as DFT (Discrete Fourier Transform) spread OFDMA.

These high-level goals led to further expectations for LTE, including reduced latency for packets, and spectral efficiency improvements above Release 6 high speed packet access (HSPA) of three to four times in the downlink and two to three times in the uplink.

Flexible channel bandwidths—a key feature of LTE—are specified at 1.4, 3, 5, 10, 15, and 20 MHz in both the uplink and the downlink. This allows LTE to be flexibly deployed where other systems exist today, including narrowband systems such as GSM Speed is probably the feature most associated with LTE. Examples of downlink and uplink peak data rates for a 20 MHz channel bandwidth. Downlink figures are shown for single input single output (SISO) and multiple input multiple output (MIMO) antenna configurations at a fixed 64QAM modulation depth, whereas the uplink figures are for SISO but at different modulation depths. These figures represent the physical limitation of the LTE frequency division duplex

(FDD) radio access mode in ideal radio conditions with allowance for signaling overheads.

Lower rates are specified for specific UE categories, and performance requirements under non-ideal radio conditions have also been developed. Figures for LTE's time division duplex (TDD) radio access mode are comparable, scaled by the variable uplink and downlink ratio.

1.1 Background

In the recent years, the world was introduced to mobile broadband. Multimedia applications through the Internet have gathered more attention. Applications such as live streaming, online gaming, mobile TV require higher data rate. The Third-generation Partnership Project (3GPP) started to work on solutions to these challenges and came up with the HSPA.

The HSPA is currently used in 3G phones for such applications. Later, the 3GPP has worked on the Long-Term Evolution (LTE) and intends to surpass the performance of HSPA . Thus, LTE will enhance applications such as online gaming and interactive TV . It is expected that in 2014, 80% of broadband users will be mobile broadband subscribers and they will be served by HSPA and LTE networks the3GPP is the standards-developing body that specifies the 3G UTRA and GSM systems.

LTE as defined by the 3GPP is the evolution of the Third generation of mobile communications, UMTS. LTE intends to create a new radio-access technology which will provide high data rates, a low latency and a greater spectral efficiency.

The 3GPP has started with the RAN Evolution workshop in November 2004. A lot of research has been carried out and proposals have been presented on the evolution of the Universal Terrestrial Radio Access Network (UTRAN). The specifications related to LTE are formally known as the evolved UMTS terrestrial radio access (E-UTRA) and evolved UMTS terrestrial radio access network (E-UTRAN) but are in general referred as project LTE. In December 2008, the LTE specification was published as part of Release 8. The initial deployment of LTE was expected in 2009. The first release of LTE namely release-8 supports peak rates of 300Mb/s, a radio-network delay of less than 5ms.

Multiple Input Multiple Output (MIMO) have gathered a lot of attention recently. It allows the achievement of high peak data rates. Furthermore, LTE operates both Frequency Division Duplexing (FDD) and Time Division Duplexing (TDD) and can be deployed in different bandwidths. With TDD the uplink and downlink operate in same frequency band whereas with TDD the uplink and downlink operate in different frequency bands.

Error correction codes provide a means to detect and correct errors introduced by a transmission channel. Two main categories of code exist: block codes and convolutional codes. They both introduce redundancy by adding parity symbols to the message data.

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission. If an error occurred, the receiver sends a "negative acknowledgement" (NAK) back to the sender, requesting that the message be retransmitted.

Cyclic redundancy check (CRC) codes are a subset of cyclic codes that are also a subset of linear block codes. The theory behind block coding and more specifically CRC coding is briefly discussed in this application report as well as most common CRC codes.

CRC implementation can use either hardware or software methods. In the traditional hardware implementation, a simple shift register circuit performs the computations by handling the data one bit at a time. In software implementations, handling data as bytes or words becomes more convenient and faster.

Choosing a particular algorithm depending on which memory and speed constraints are required. In this research we going to show the capable circuit design for CRC on Simulation.

A downlink physical channel corresponds to a set of resource elements carrying information originating from higher layers and is the interface defined between 3GPP TS 36.212 and the present document 3GPP TS 36.211. The following downlink physical channels are defined:

- Physical Downlink Shared Channel, PDSCH.
- Physical Broadcast Channel, PBCH.
- Physical Multicast Channel, PMCH.
- Physical Control Format Indicator Channel, PCFICH.
- Physical Downlink Control Channel, PDCCH.
- Physical Hybrid ARQ Indicator Channel, PHICH.
- Enhanced Physical Downlink Control Channel, EPDCCH.
- MTC Physical Downlink Control Channel, MPDCCH.
- Short Physical Downlink Control Channel, SPDCCH.

Downlink physical signal corresponds to a set of resource elements used by the physical layer but does not carry information originating from higher layers. The following downlink physical signals are defined:

- Reference signal.
- Synchronization signal.
- Discovery signal.
- MTC wake-up signal, MWUS.
- A discovery signal occasion for a cell consists of a period with a duration of
- one to five consecutive subframes for frame structure type 1
- two to five consecutive subframes for frame structure type 2
- 12 OFDM symbols within one non-empty subframe for frame structure type 3

where the UE in the downlink subframes may assume presence of a discovery signal consisting of:

cell-specific reference signals on antenna port 0 in all downlink subframes and in DwPTS of all special subframes in the period for frame structure type 1 and 2.

cell specific reference signals on antenna port 0 when higher layer parameters indicate only one configured antenna port for cell specific reference signals for a serving cell using frame structure type 3.

cell specific reference signals on antenna port 0 and antenna port 1 when higher layer parameters indicate at least two configured antenna ports for cell specific reference signals for a serving cell using frame structure type 3.

cell specific reference signals on antenna port 0 and antenna port 1 when higher layer configured parameter *presenceAntennaPort1* is signaled to be 1, for a neighbor cell when using frame structure type 3.

primary synchronization signal in the first subframe of the period for frame structure types 1 and 3 or the second subframe of the period for frame structure type 2.

secondary synchronization signal in the first subframe of the period, and non-zero-power CSI reference signals in zero or more subframes in the period.

1.2 LTE. PHY. Layer

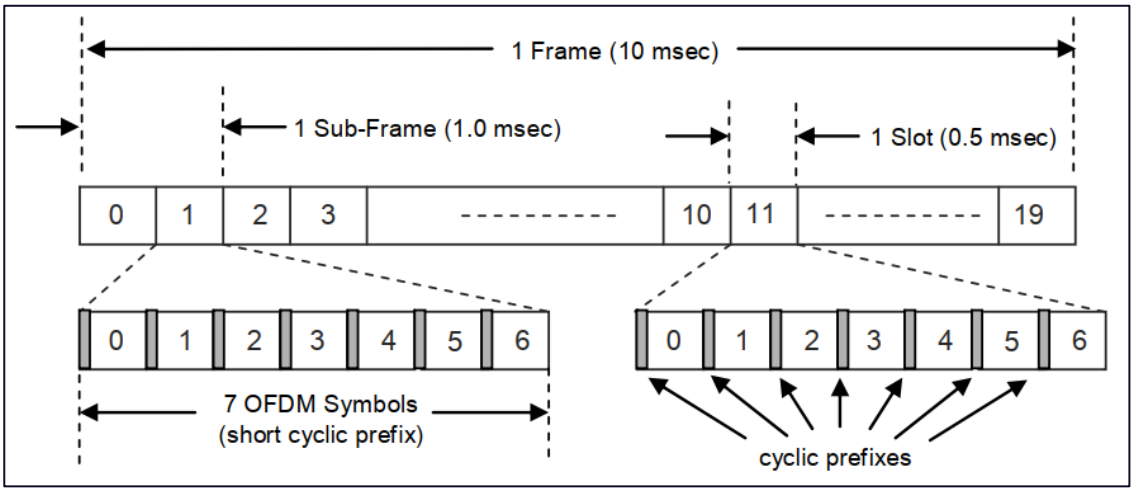


Figure 1-2-1:1.2 "LTE. Generic Frame Structure"

OFDMA is an excellent choice of multiplexing scheme for the 3GPP LTE downlink. Although it involves added complexity in terms of resource scheduling, it is vastly superior to packet-oriented approaches in terms of efficiency and latency. In OFDMA, users are allocated a specific number of

subcarriers for a predetermined amount of time. These are referred to as physical resource blocks (PRBs) in the LTE specifications. PRBs thus have both a time and frequency dimension. Allocation of PRBs is handled by a scheduling function at the 3GPP base station (eNodeB).

In order to adequately explain OFDMA within the context of the LTE, we must study the PHY layer generic frame structure. The generic frame structure is used with FDD. Alternative frame structures are defined for use with TDD.

However, TDD is beyond the scope of this paper. Alternative frame structures are therefore not considered.

As shown in figure 1 – 1.2, LTE frames are 10 msec in duration. They are divided into 10 subframes, each subframe being 1.0 msec long. Each subframe is further divided into two slots, each of 0.5 msec duration. Slots consist of either 6 or 7 ODFM symbols, depending on whether the normal or extended cyclic prefix is employed.

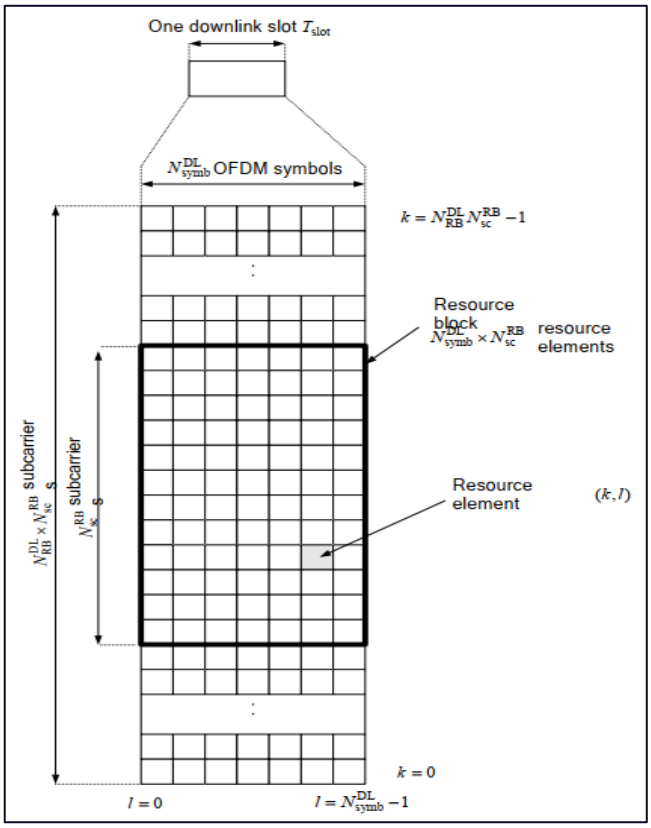


Figure 1-2-2:. "Downlink Resource Gride"

The total number of available subcarriers depends on the overall transmission bandwidth of the system.

The LTE specifications define parameters for system bandwidths from 1.25 MHz to 20 MHz PRB is defined as consisting of 12consecutives subcarriers for one slot (0.5 msec) in duration A PRB is the smallest element of resource allocation assigned by the base station scheduler.

The common functions of PHY. layer is as follows :

1. It converts MAC layer format in the format suitable as per medium used.
2. It adds FEC (forward error correction) functionality to enable error correction at the receiver.
3. It performs modulation and demodulation functionalities at transmitter and receiver end respectively. One of the key changes in the Layer 1 (PHY layer) of the 3GPP LTE is the change from CDMA (code-division multiple access) to OFDM (orthogonal frequency-division multiplexing). One of the main benefits of OFDM is that it reduces the problems associated with multiple paths in the radio channel. In CDMA, a significant amount of processing must be devoted to characterizing and tracking the radio channel to compensate for the effects of fading in the channel.

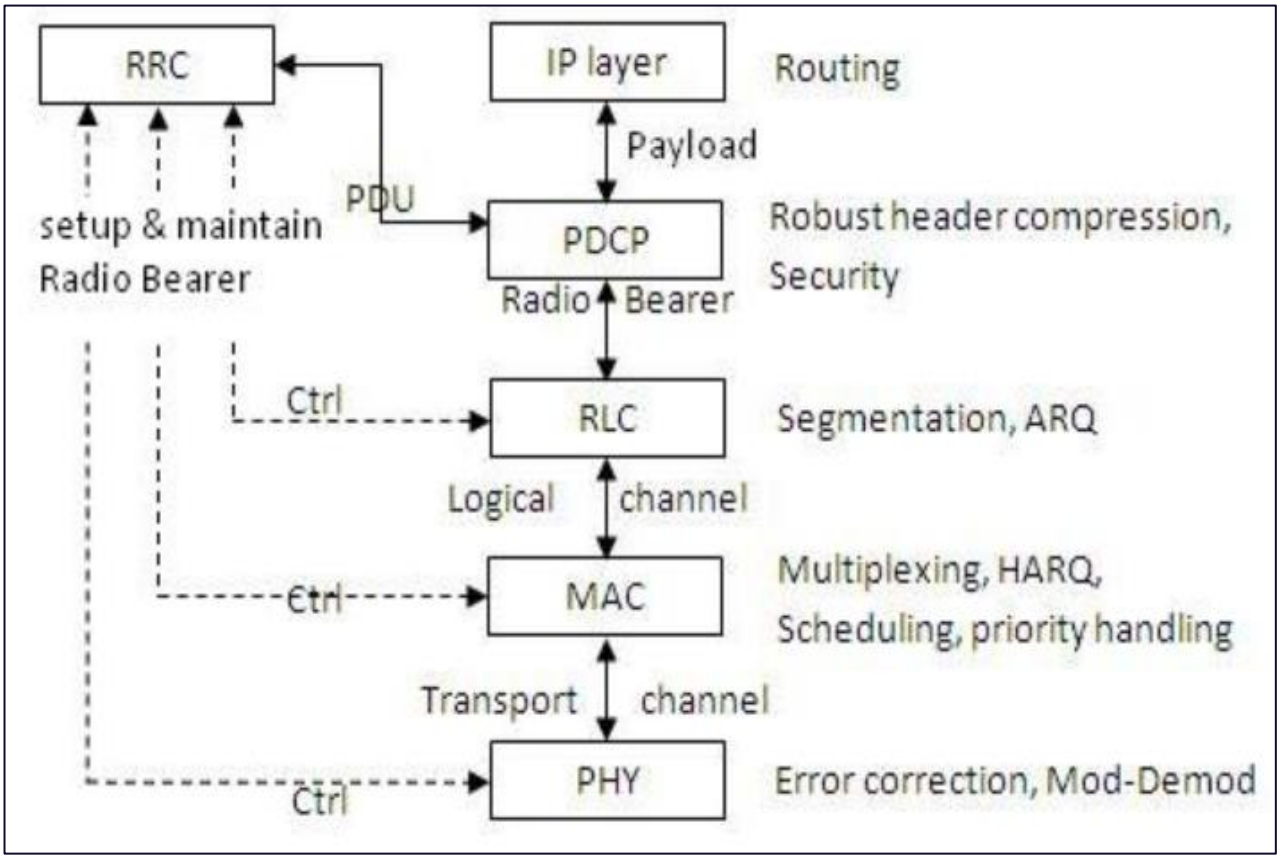


Figure 1-2-3:1.2 "LTE. Protocol Stack"

Following figure depicts LTE physical layer transmitter part of eNodeB i.e., LTE base station. LTE eNodeB is similar to Base station of other technologies such as WiMAX, GSM etc. eNodeB Physical layer consists of Channel coding, rate matching, scrambler, mapper, layer mapping, pre-coding, resource element mapper and OFDM module.
CRC is appended to the data from MAC layer before being passed through the PHY layer. To achieve its goals, LTE must satisfy the following requirements:

Data rates

LTE should support a data rate up to 100 Mb/s within a 20 MHz downlink spectrum allocation and 50 Mb/s within a 20 MHz uplink or, equivalently, spectral efficiency values of 5bps/Hz and 2.5 bps/Hz, respectively.

Throughput

The downlink average throughput per MHz is about 3 to 4 times higher than in their lease 6. The uplink average user throughput per MHz is about 2 to 3 times higher than in the release 6.

Bandwidth

LTE allows bandwidth ranging from 1.4 MHz up to 20 MHz, where the latter is used to achieve the highest LTE data rate. Furthermore, LTE operates in both paired and unpaired spectrum by supporting both Frequency-Division Duplex (FDD) and Time Division Duplex (TDD).

Mobility

The mobility is optimized for low terminals speeds ranging from 0 to 15 km/h. The connection should be maintained for very high UEs speeds up to 350 km/h or even up to 500 km/h.

Coverage

The above targets should be met for 5 km cells and some slight degradation in throughput and spectrum efficiency for 30 km cells. 100 km cells and up can’t meet the targets requirements.

Orthogonal Frequency Division Multiplex (OFDM) is the core of LTE downlink transmission. LTE downlink physical resource can be represented as a time-frequency resource grid as depicted in the Figure 2.2. A *Resource Block* (RB) has a duration of 0.5 msec (one slot) and a bandwidth of 180 kHz (12 subcarriers). It is a straightforward to see that each RB has 12x7 = 84 resource elements in the case of normal cyclic prefix and 12x6 = 72 resource elements in the case of extended cyclic prefix.

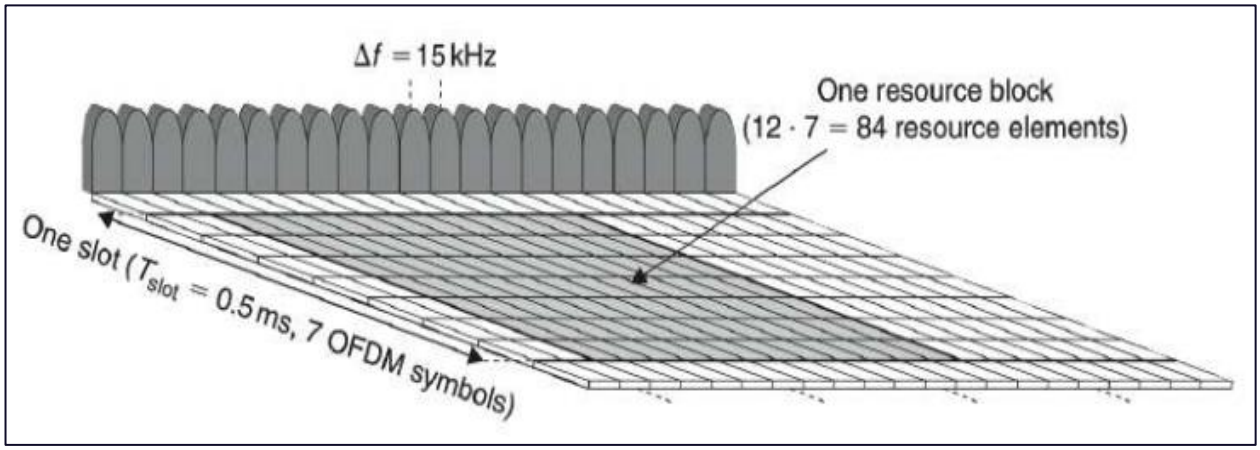


Figure 1-2-4:1.2 "LTE downlink physical resource based on OFDM"

The resource grid refers to a number of resource blocks in the available bandwidth. Each entry of the resource block is called a Resource Element (RE) which represents one ODFM subcarrier during one OFDM symbol interval. The number of RB in a resource grid varies according to the size of the bandwidth. The OFDM subcarrier spacing is 15 kHz. The table 2.1 shows the LTE bandwidth and resource configuration.

Bandwidth (MHz)	1.4	3	5	10	15	20
Number of Resources Blocks	6	15	25	50	75	100
Number of occupied subcarriers	72	180	300	600	900	1200
IFFT/FFT Size	128	256	512	1024	1536	2048
Subcarrier Spacing (KHz)	15	15	15	15	15	15

Figure 1-2-5 :1.2 "Bandwidth and Resource blocks specifications"

To perform channel estimation, reference symbols (reference signals) are embedded in the Physical resource block (PRB) as shown in Figure 5-1.2. Reference signals are inserted in the first and fifth OFDM symbols of each slot in the case of the short CP and during the first and fourth OFDM symbols in the case of the long CP. Thus, there are four reference symbols within one PRB.

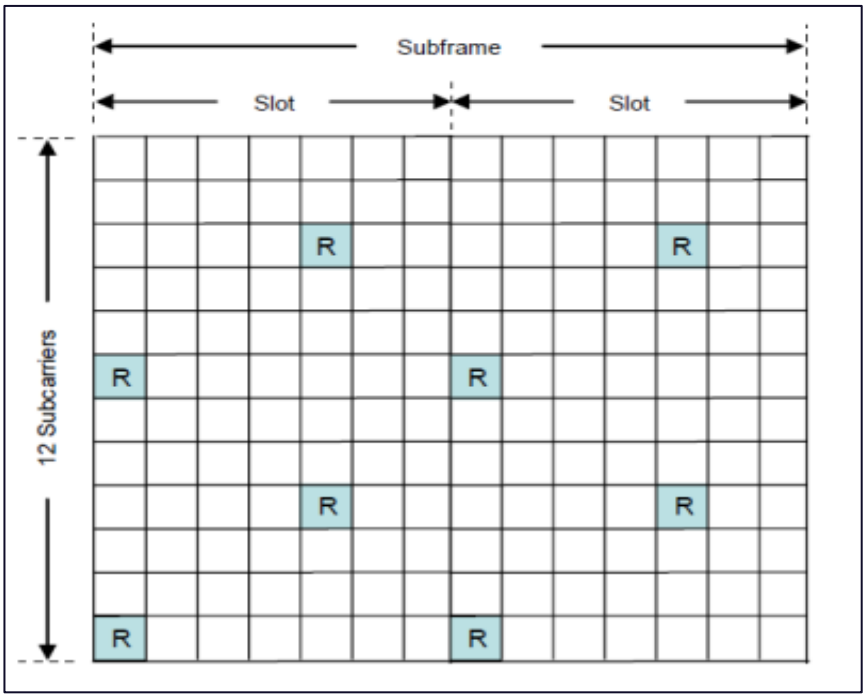


Figure 1-2-6:1.2 "LTE downlink reference signal assuming normal CP"

At the beginning of the transport channel processing, a *Cyclic Redundancy Check* (CRC) is computed and attached to each transport block (TB) for the detection of errors in the TB by the receiver. After the CRC insertion, the data (TB + CRC) to be sent are turbo coded with a coding rate of 1/3. The task of the hybrid-ARQ is to take care of the retransmission if erroneously received packets are received. Retransmission must represent the same information bits as the initial message, but the coded bits used for each retransmission can be different than the original message. Later the information to be transmitted is modulated using one of the following modulation schemes:

QPSK,16QAM, and 64QAM representing two, four, and six bits per modulation symbol, respectively (see Figure 7 – 1.2).

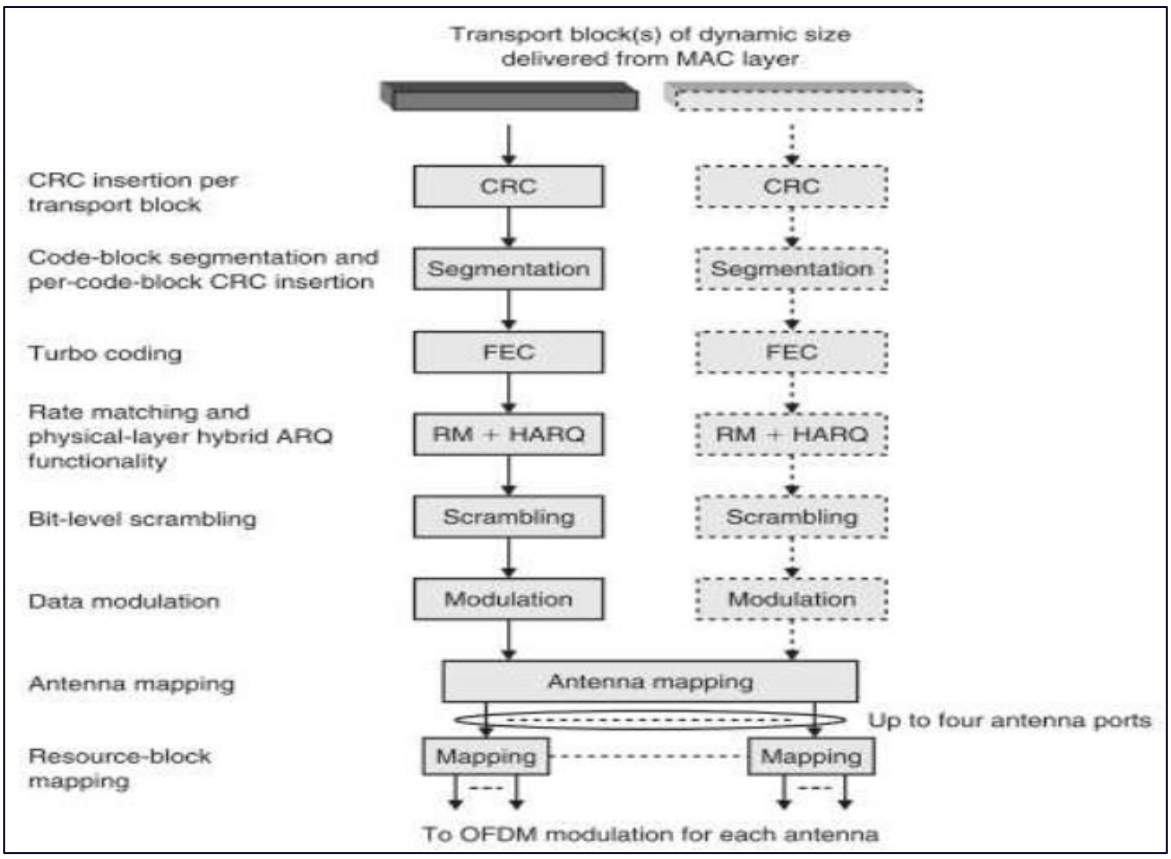


Figure 1-2-7:1.2 "LTE downlink transport-channel processing"

The Antenna mapping block maps the transport block to different antennas. LTE uses up to four transmit antennas. LTE supports different multiple transmit antennas schemes:
transmit diversity, beamforming and spatial multiplexing. The goal of the resource block mapping is to map the data to be sent on each antenna to a set of resource blocks assigned by the scheduler.

The Figure 8 – 1.2 displays the downlink resource block mapping.

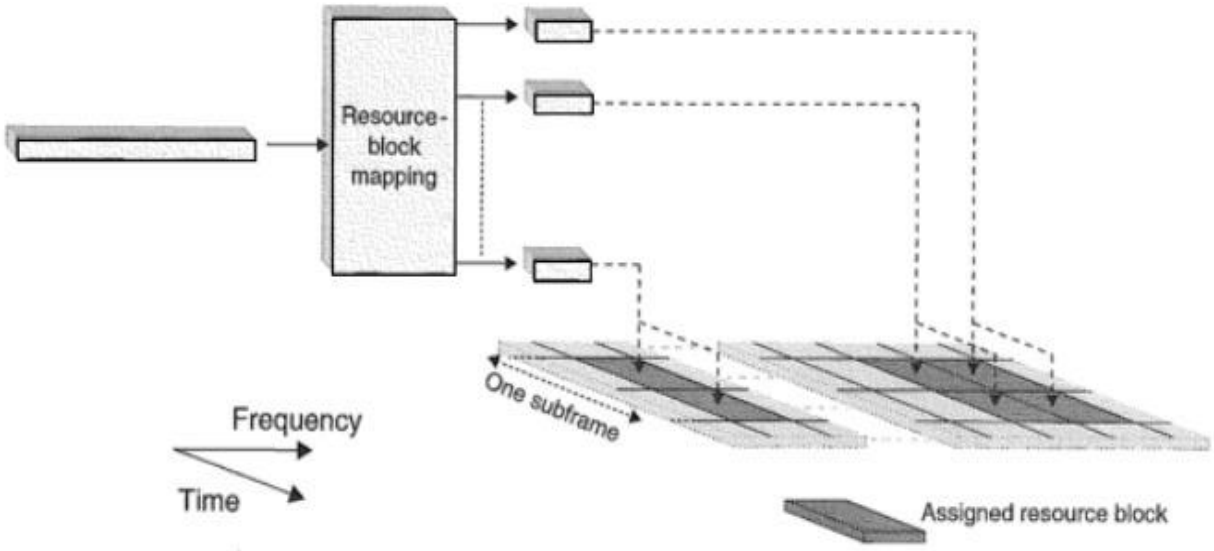


Figure 1-2-8: 1.2 "Downlink resource block mapping"

Link adaptation (LA) compensates the variations in the instantaneous channel conditions. In situations with advantageous channel conditions, the data rate is increased and vice versa. To adjust the data rate, LA uses Adaptive modulation and coding (AMC).

AMC matches the modulation and the channel coding scheme on resources assigned by the scheduler.

In situations with advantageous channel conditions, AMC selects a higher modulation order and coding rate and vice versa. This principle is illustrated in Figure (9 – 1.2).

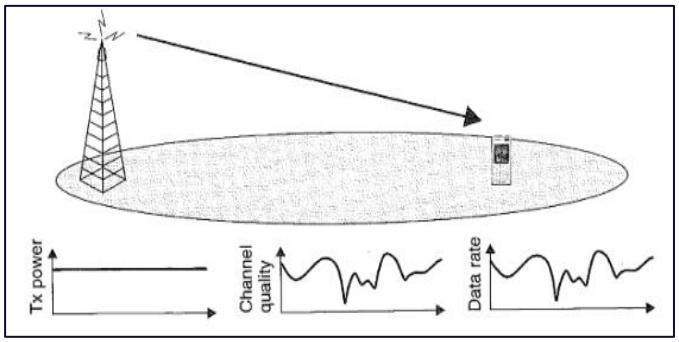


Figure 1-2-9:Link adaptation

1.2 LTE. Wave – Generation

The **LTE Waveform Generator** app enables you to create, impair, visualize, and export LTE waveforms.

The app provides these capabilities by using the Wireless Waveform Generator app configured for LTE waveform generation. Using the app, you can:

- Generate LTE test model (E-TM) waveforms, as defined in section 6 of TS 36.141 [1].
- Generate LTE downlink reference measurement channel (RMC) waveforms for user equipment (UE) performance testing, as defined in Annex A.3 of TS 36.101 [2].
- Generate LTE uplink RMC waveforms for base station (BS) performance testing, as defined in Annex A of TS 36.104 [3].
- Export the LTE waveform to your workspace or to a. mat or a .bb file.
- Export LTE waveform generation parameters to a runnable MATLAB® script. Visualize the LTE waveform in constellation diagram, spectrum analyzer, OFDM grid, 3D spectrogram, and time scope plots.
- Distort the LTE waveform by adding RF impairments, such as AWGN, phase offset, frequency offset, DC offset, IQ imbalance, and memoryless cubic nonlinearity.
- Generate an LTE waveform that you can transmit using a connected lab test instrument. The app can transmit a waveform by using instruments supported by the rfsiggen (Instrument Control Toolbox) function. Use of the transmit feature in the app requires Instrument Control Toolbox™ software. For more information, see the documentation for Instrument Control Toolbox.

1. Using 64-QAM, FDD.
2. Using 16-QAM, FDD.
3. Using 16-QAM, TDD.

1.

Downlink RMC

Reference channel:	R.8 (Port0, 75 RB, 64...
Duplex mode:	FDD
Transmission scheme:	Port0
Cell identity:	0
RNTI:	1
RV sequence:	[0 1 2 3]
Rho (dB):	0
OCNG:	PDSCH and PDCCH
Subframes:	10
Codewords:	1
PMI set:	[1]
HARQ processes:	8
Windowing (samples):	0
Antenna (for visuals):	1

RMC Parameter Summary

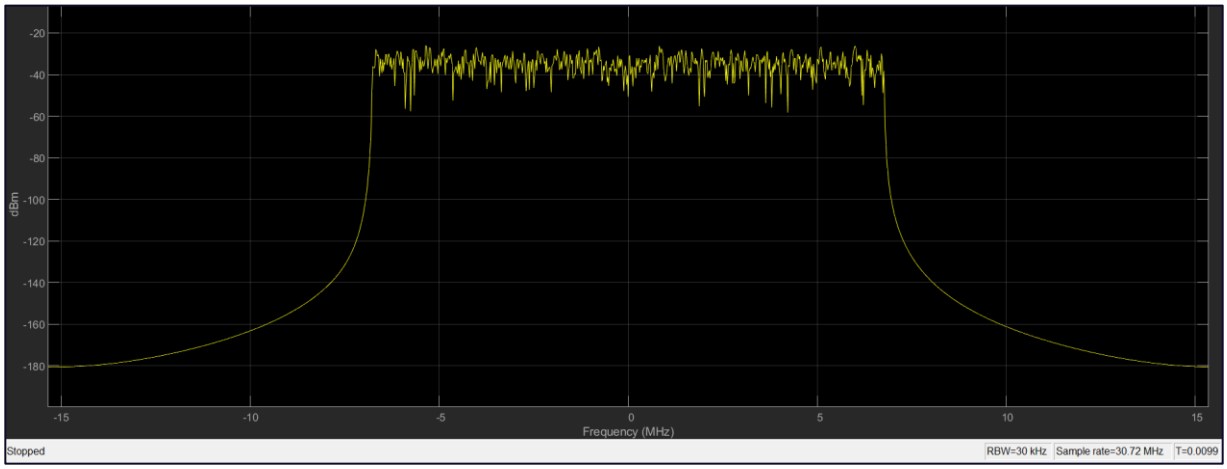
Transmission scheme:	Port0
Downlink resource blocks:	75
Allocated resource blocks:	75
Number of CellRS ports:	1
Modulation:	64QAM
Number of layers:	1
Total bits/frame:	420456

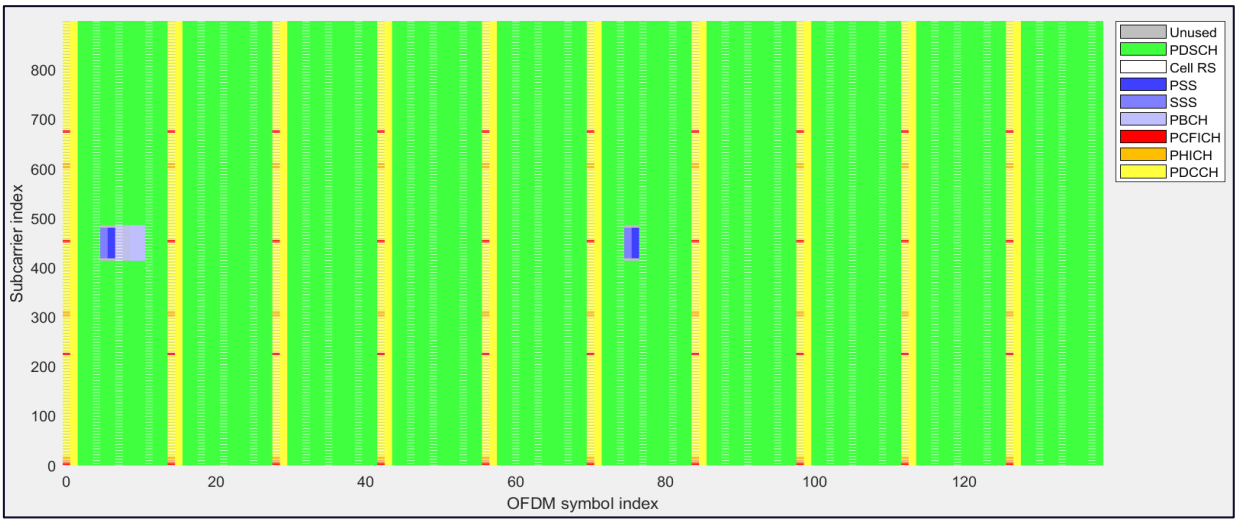
Bit Source

Bit source #1:	User-defined
Transport info bits (CW #1):	[1; 0; 0; 1]

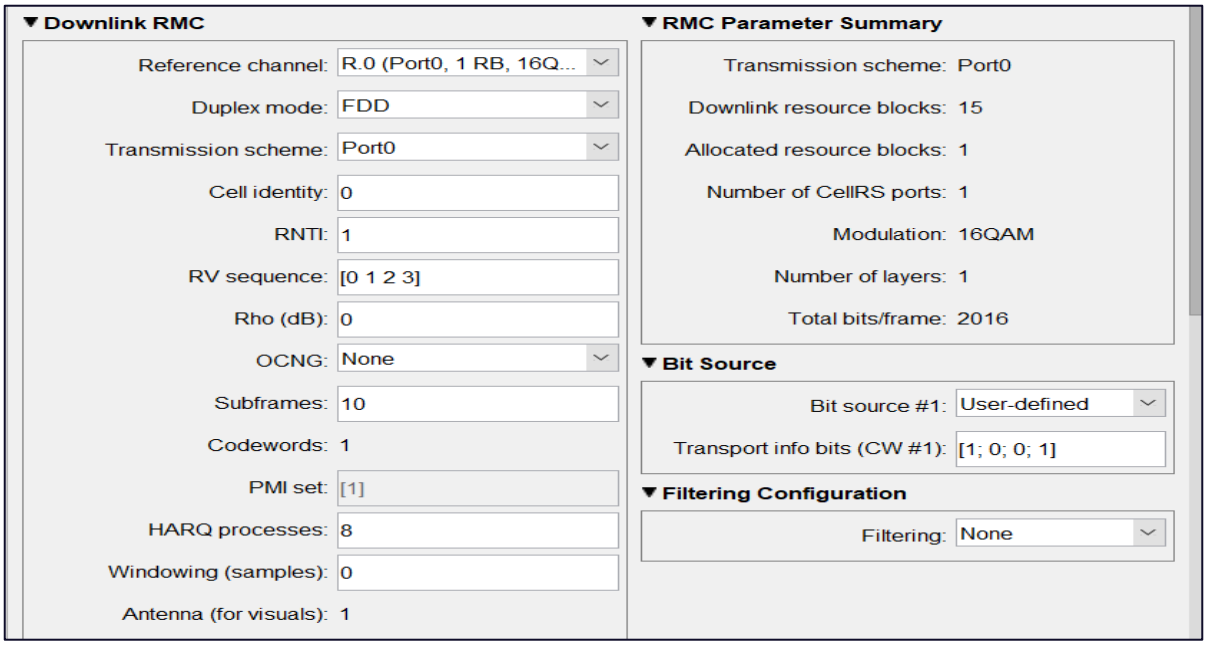
Filtering Configuration

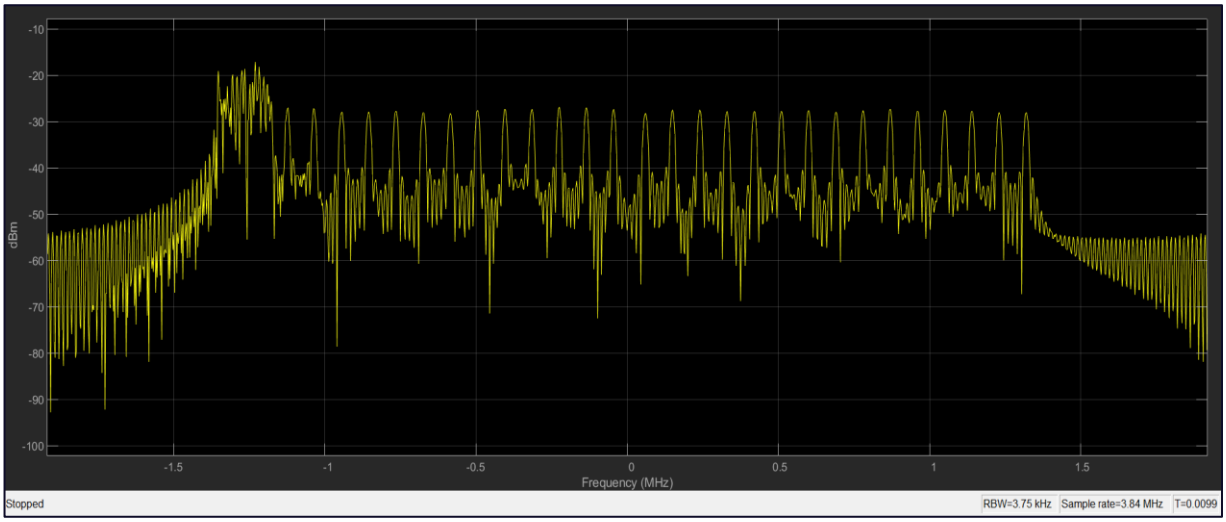
Filtering:	None

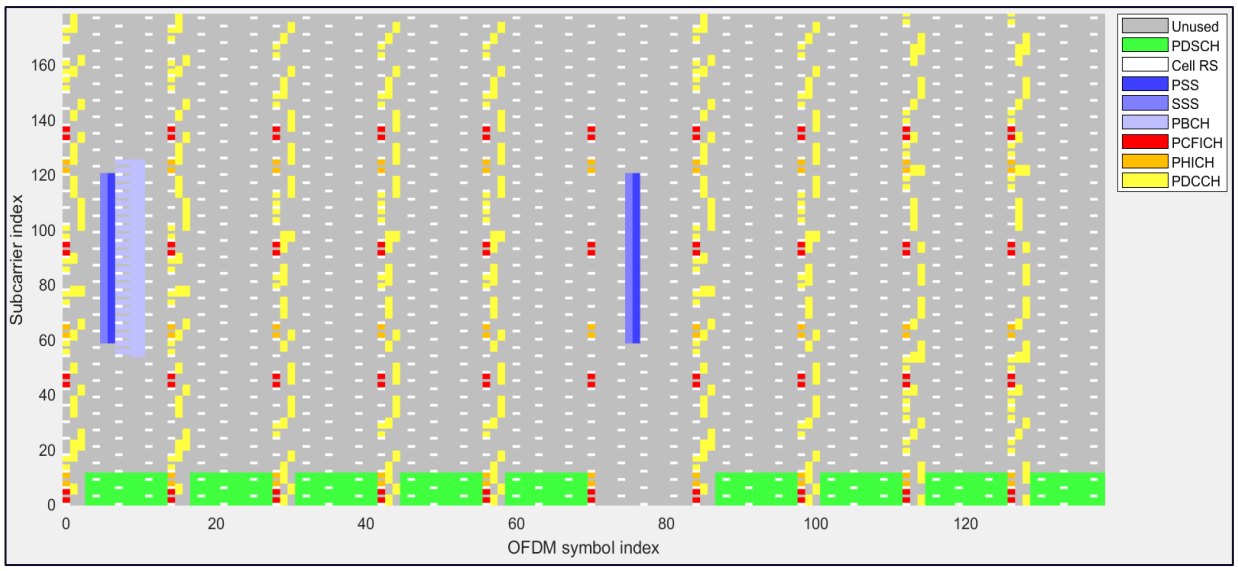




2.


Downlink RMC
Reference channel: R.0 (Port0, 1 RB, 16Q...
Duplex mode: FDD
Transmission scheme: Port0
Cell identity: 0
RNTI: 1
RV sequence: [0 1 2 3]
Rho (dB): 0
OCNG: None
Subframes: 10
Codewords: 1
PMI set: [1]
HARQ processes: 8
Windowing (samples): 0
Antenna (for visuals): 1
RMC Parameter Summary
Transmission scheme: Port0
Downlink resource blocks: 15
Allocated resource blocks: 1
Number of CellRS ports: 1
Modulation: 16QAM
Number of layers: 1
Total bits/frame: 2016
Bit Source
Bit source #1: User-defined
Transport info bits (CW #1): [1; 0; 0; 1]
Filtering Configuration
Filtering: None


dBm
Frequency (MHz)
Stopped
RBW=3.75 kHz
Sample rate=3.84 MHz
T=0.0099


Subcarrier index
OFDM symbol index
Unused
PDSCH
Cell RS
PSS
SSS
PBCH
PCFICH
PHICH
PDCCH

3.

Downlink RMC

Reference channel:	R.0 (Port0, 1 RB, 16Q...
Duplex mode:	TDD
Transmission scheme:	Port0
Cell identity:	0
RNTI:	1
RV sequence:	[0 1 2 3]
Rho (dB):	0
OCNG:	None
Subframes:	10
Codewords:	1
PMI set:	[1]
HARQ processes:	7
Windowing (samples):	0
Antenna (for visuals):	1

RMC Parameter Summary

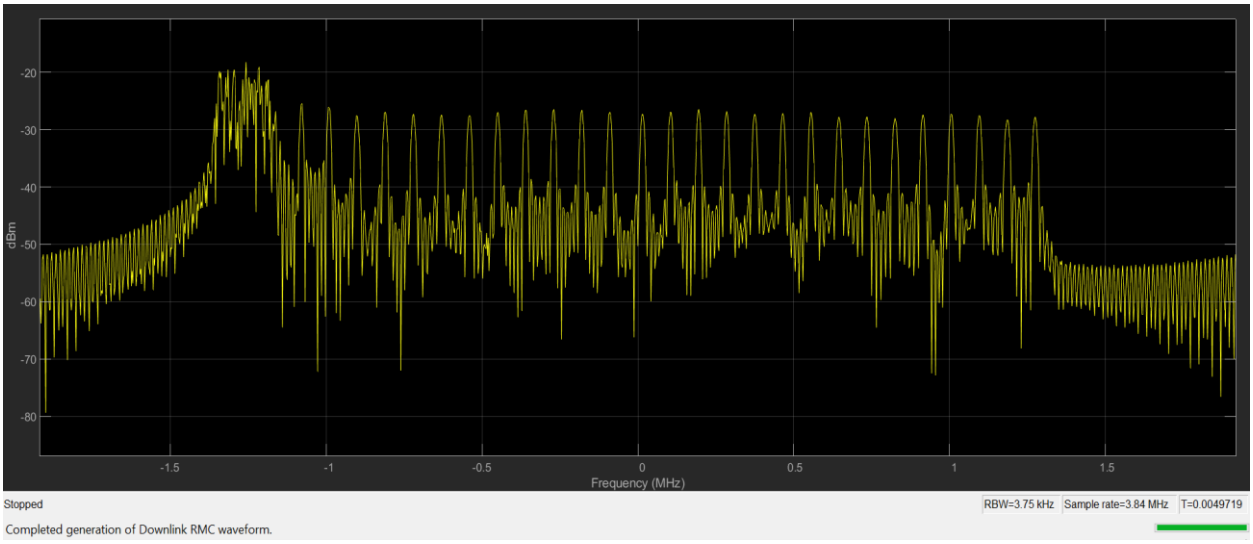
Transmission scheme:	Port0
Downlink resource blocks:	15
Allocated resource blocks:	1
Number of CellRS ports:	1
Modulation:	16QAM
Number of layers:	1
Total bits/frame:	1088

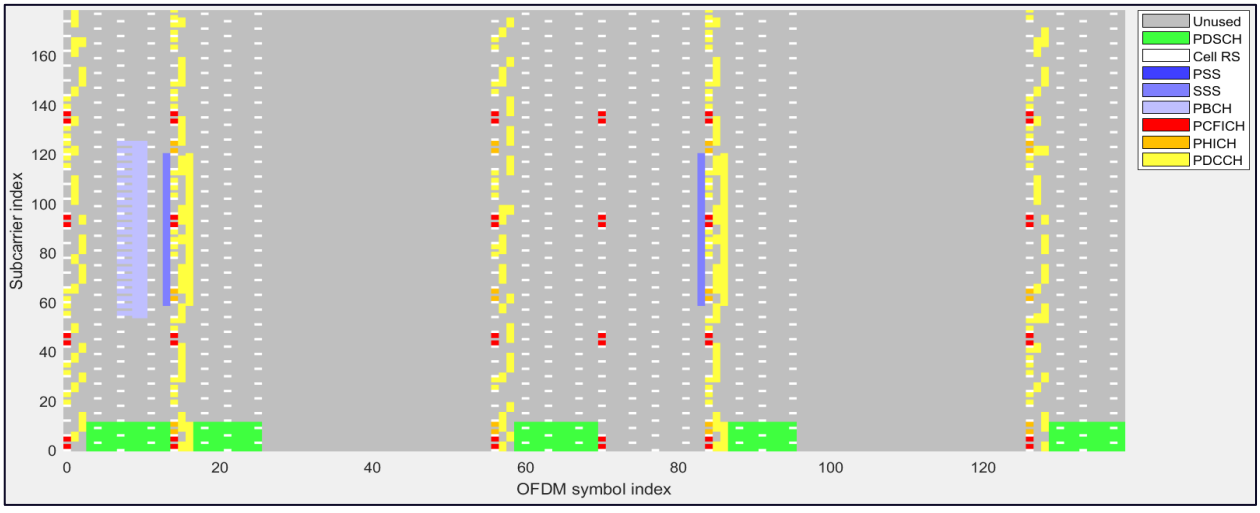
Bit Source

Bit source #1:	User-defined
Transport info bits (CW #1):	[1; 0; 0; 1]

Filtering Configuration

Filtering:	None





1.4 Coding Theory Behind CRC

A block code consists of a set of fixed-length vectors called code words. The length of a code word is the number of elements in the vector and is denoted by n. The elements of a code word are selected from an alphabet of q elements. When the alphabet consists of two elements, 0 and 1, the code is binary; otherwise, it is nonbinary code. In the binary case, the elements are bits. The following discussion focuses on binary codes.

There are 2n possible code words in a binary block code of length n. From these 2n code words you may select M = 2k code words (k < n) to form a code. Thus, a block of k information bits is mapped into a code word of length n selected from the set of M = 2k code words. This is called an (n,k) code and the ratio k/n is defined to be the rate of the code.

The encoding and decoding functions involve the arithmetic operations of addition and multiplication performed on code words. These arithmetic operations are performed according to the conventions of the algebraic field that has, as its elements, the symbols contained in the alphabet. For binary codes, the field is finite and has 2 elements, 0 and 1, and is called GF (2) (Galois Field).

A code is linear if the addition of any two-code vectors forms another code word. Code linearity simplifies implementation of coding operations.

The minimum distance dmin is the smallest hamming distance between two code words. The hamming distance is the number of symbols (bits in the binary case) in which they differ. The minimum distance is closely linked to the capacity of the code to detect and correct errors and is a function of the code characteristics. An () block code is capable of detecting:

$$d_{min} - 1 \text{ errors and correcting } \frac{1}{2}(d_{min} - 1) \text{ errors.}$$

2 CYCLIC REDUNDANCY CHECK

The cyclic redundancy check, or CRC, is a technique for detecting errors in digital data, but not for making corrections when errors are detected. It is used primarily in data transmission. In the CRC method, a certain number of check bits, often called a checksum, are appended to the message being transmitted. The receiver can determine whether or not the check bits agree with the data, to ascertain with a certain degree of probability whether or not an error occurred in transmission. If an error occurred, the receiver sends a "negative acknowledgement" (NAK) back to the sender, requesting that the message be retransmitted.

CRCs are so called because the check (data verification) value is a redundancy (it adds zero information to the message) and the algorithm is based on cyclic codes.
CRCs are popular because they are simple to implementing binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors
caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function. The CRC was invented by W. Wesley Peterson in 1961; the 32-bit polynomial used in the CRC function of Ethernet and many other standards is the work of several researchers and was published in 1975.

Its role in our communication system is one and the most important part i.e., to detect whether there is any error in the receive data or not. This is important because burst errors are common transmission errors in many communication channels, including magnetic and optical storage devices. Typically, an n-bit CRC, applied to a data block of arbitrary length, will detect any single error burst not longer than n bits, and will detect a fraction 1-2-nof all longer error bursts.
The CRC method treats the message as a polynomial in GF (2). For example, the message 11001001, where the order of transmission is from left to right (110…) is treated as a representation of the polynomial. The sender and receiver agree on a certain fixed polynomial called the generator polynomial.

For example, for a 16-bit CRC the CCITT has chosen the polynomial which is now widely used for a 16-bit CRC checksum. To compute an r-bit CRC checksum, the generator polynomial must be of degree r.

The sender appends r 0-bits to the m-bit message and divides the resulting polynomial of degree by the generator polynomial. This produces a remainder polynomial of degree (or less).

The remainder polynomial has r coefficients, which are the checksum. The quotient polynomial is discarded. The data transmitted (the code vector) is the original m-bit message followed by the r-bit checksum. There are two ways for the receiver to assess the correctness of the transmission. It can compute the checksum from the first m bits of the received data and verify that it agrees with the last r received bits. Alternatively, and following usual practice, the receiver can divide all the m + r received bits by the generator polynomial and check that the r-bit remainder is 0. To see that the remainder must be 0, let M be the polynomial representation of the message, and let R be the polynomial representation of the remainder that was computed by the sender. Then the transmitted data corresponds to the polynomial (or, equivalently,).

By the way R was computed, we know that where G is the generator polynomial and Q is the quotient (that was discarded). Therefore, the transmitted data, is equal to QG, which is clearly a multiple of G. If the receiver is built as nearly as possible just like the sender, the receiver will append r 0-bits to the received data as it computes the remainder R. But the received data with 0-bits appended is still a multiple of G, so the computed remainder is still 0. That's the basic idea, but in reality, the process is altered slightly to correct for such deficiencies as the fact that the method as described is insensitive to the number of leading and trailing 0-bits in the data transmitted. In particular, if a failure occurred that caused the received data, including the checksum, to be all-0, it would be accepted. This is Different Shapes of Polynomials are commonly used in CRC, the MSB must equal 1, to achieve the equation is permeative. The standard used in LTE. Physical Layer is CRC16,24A without segmentation,

1. $g_{CRC24A}(D) = [D_{24} + D_{23} + D_{18} + D_{17} + D_{14} + D_{11} + D_{10} + D_7 + D_6 + D_5 + D_4 + D_3 + D + 1]$
2. $g_{CRC24B}(D) = [D_{24} + D_{23} + D_6 + D_5 + D + 1]$
3. $g_{CRC16}(D) = [D_{16} + D_{12} + D_5 + 1]$
4. $g_{CRC8}(D) = [D_8 + D_7 + D_4 + D_3 + D + 1]$

2.1 Algorithms for CRC Computation

It generates parity bits by cyclic generator polynomials, which are then added at the end of the transport block. Detecting errors by adding some redundant bits along with Data bits (Transmitted). The polynomial must be chosen to maximize the error-detecting capabilities While minimizing overall collision probabilities. If we use the generator polynomial g(x)=p(1+x), where (p) is a primitive polynomial of degree (r-1), Then the maximal total block length is $(2^{r-1} - 1)$, and the code is able to detect single, double, triple and any odd number of errors. To compute an n-bit binary CRC, line the bits representing the input in a row, and position the (n + 1)-bit pattern representing the CRC's divisor (called a "polynomial") underneath the left-hand end of the row.

In this example, we shall encode 14 bits of message with a 3-bit CRC, with a polynomial $x^3 + x + 1$. The polynomial is written in binary as the coefficients; a 3rd-degree polynomial has 4 coefficients ($1x^3 + 0x^2 + 1x + 1$). In this case, the coefficients are 1, 0, 1 and 1. The result of the calculation is 3 bits long.
Start with the message to be encoded:

```
11010011101100
```

This is first padded with zeros corresponding to the bit length n of the CRC. This is done so that the resulting code word is in systematic form. Here is the first calculation for computing a 3-bit CRC:

```
11010011101100 000 <--- input right padded by 3 bits
1011               <--- divisor (4 bits) = x³ + x + 1
------------------
01100011101100 000 <--- result
```

The algorithm acts on the bits directly above the divisor in each step. The result for that iteration is the bitwise XOR of the polynomial divisor with the bits above it. The bits not above the divisor are simply copied directly below for that step. The divisor is then shifted right to align with the highest remaining 1 bit in the input, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the entire calculation:

```
11010011101100 000 <--- input right padded by 3 bits
1011               <--- divisor
01100011101100 000 <--- result (note the first four bits are the XOR with the divisor beneath, the rest of the bits are unchanged)
 1011              <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000 <--- note that the divisor moves over to align with the next 1 in the dividend (since quotient for that step
was zero)
      1011            (in other words, it doesn't necessarily move one bit per iteration)
00000000110100 000
       1011
00000000011000 000
        1011
00000000001110 000
         1011
00000000000101 000
          101 1
-----------------
00000000000000 100 <--- remainder (3 bits).  Division algorithm stops here as dividend is equal to zero.
```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end of the row. These n bits are the remainder of the division step and will also be the value of the CRC function (unless the chosen CRC specification calls for some postprocessing).

The validity of a received message can easily be verified by performing the above calculation again, this time with the check value added instead of zeroes. The remainder should equal zero if there are no detectable errors.

```
11010011101100 100 <--- input with check value
1011               <--- divisor
01100011101100 100 <--- result
 1011              <--- divisor ...
00111011101100 100


......

00000000001110 100
          1011
00000000000101 100
           101 1
------------------
00000000000000 000 <--- remainder
```

Linear feedback shift register is used to implement the necessary circuit of Cyclic Redundancy Check. A figure is showing the circuit diagram of LFSR. It takes serial inputs, and the output also serially comes. So, for very high-speed data transfer it should have very high clock frequency in our case it is not of concern.

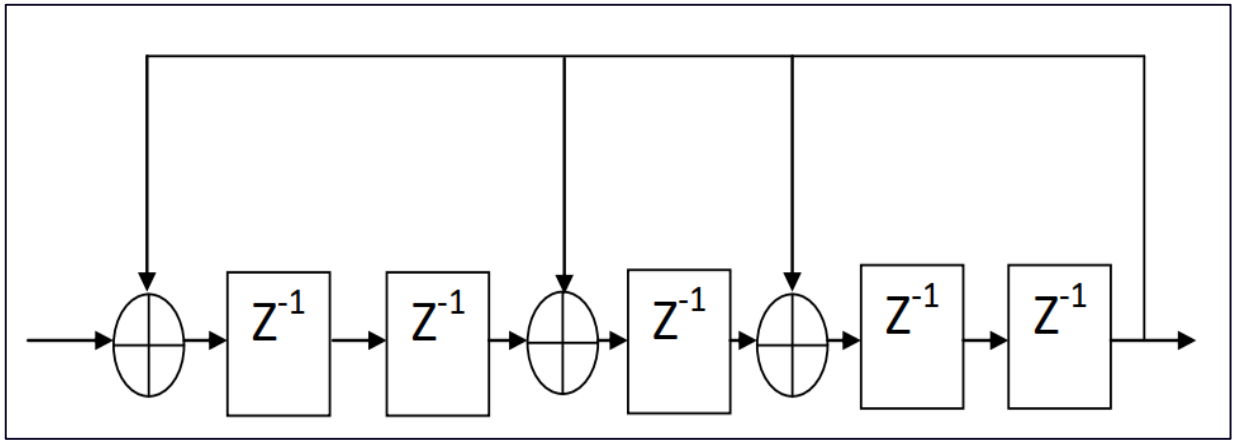


Figure 2-2-1:2.1 "Design Structure of CRC"

2.2 CRC for Noisy BPSK

The CRC generator and detector pair use a standard CRC-4 polynomial, $z^4 + z^3 + z^2 + z + 1$. The length of the CRC is 4 bits as determined the degree of the polynomial. The number of checksums per frame is 1, so the full transmission frame has one CRC appended at the end.

A binary signal frame gets a CRC code appended to the end of the frame. BSPK modulation is applied to the signal and the signal passes through an AWGN channel. The signal is demodulated, and then a CRC syndrome detector removes the CRC and calculates the CRC errors.

Generate 12-bit frames of binary data and append CRC bits. Based on the degree of the polynomial, 4 bits are appended to each frame. Apply BPSK modulation and pass the signal through an AWGN channel. Demodulate and use the CRC detector to determine if the frame is in error.

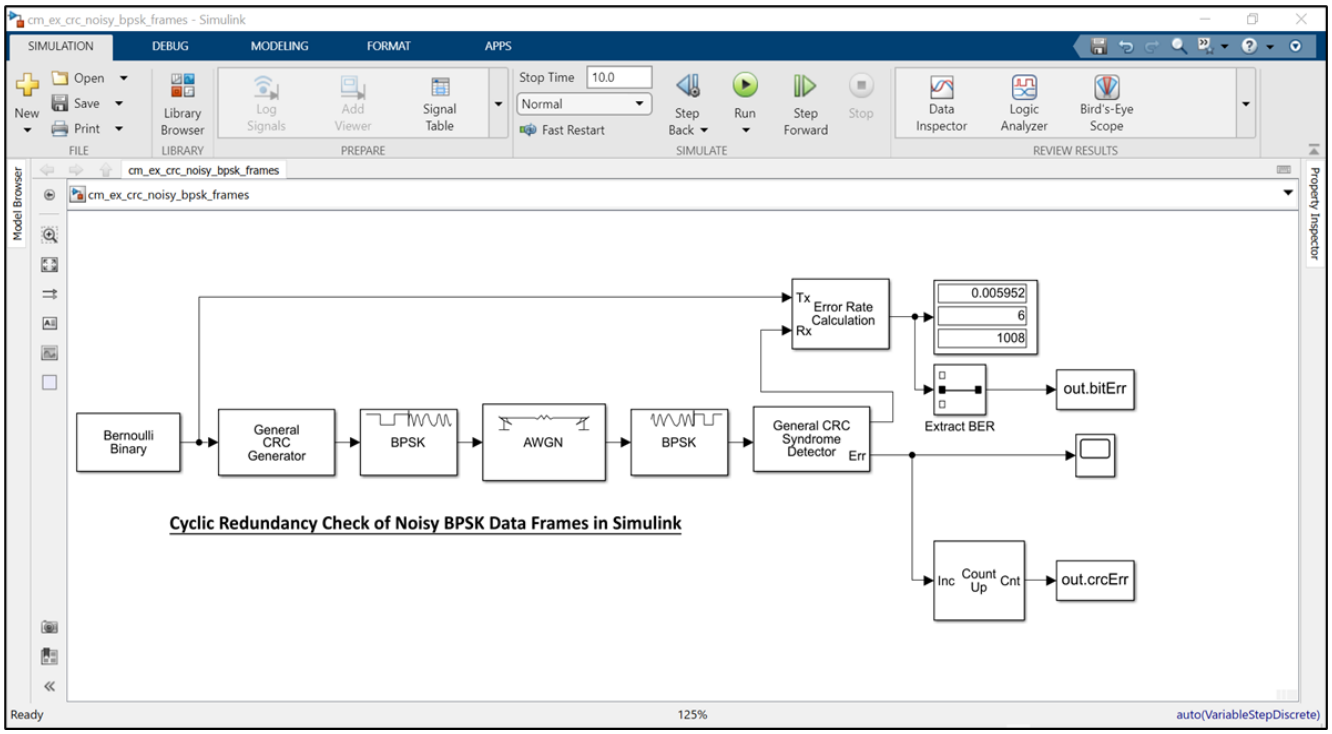
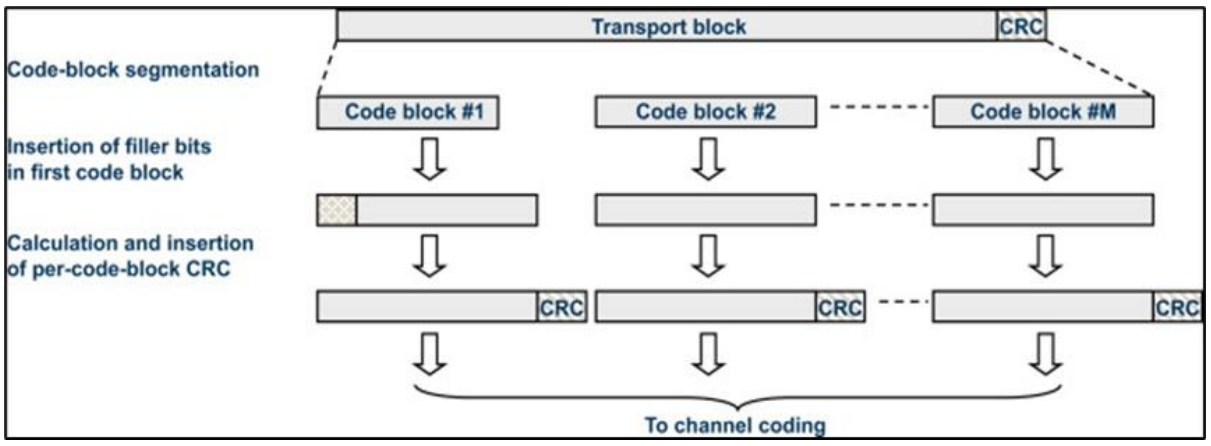


Figure 2-2:2.2 "Simulink Design Noisy BPSK with CRC".

The results of the CRC detection are compared to a BER calculation.

Number of bit errors detected: 6
Number of crc errors detected: 7

2.3 Segmentation and Memory



Code block segmentation In LTE there are two sizes defined for code block i.e., minimum and maximum code block size. These block sizes are based on block sizes as supported by the turbo interleave module of CTC Encoder. They are as follows:

- 40 bits of min. code block size.
- 6144 bits of max code block size.

If input transport block length is greater than the maximum code block size as supported by encoder, then the input block is segmented into the one supported. This segmented block is referred as code blocks.

Each of these code blocks has a 24-bit CRC attached. This CRC is calculated similar to Transport Block CRC calculation. Filler bits are appended at the start of segment; this helps code block size to match a set of valid turbo interleaver block sizes.

In the case when input block (B) is less than min. size, filler bits as zeros are added to the beginning of the code block to get complete set of 40 bits. If B is larger than the maximum code block size Z, segmentation of the input bit sequence is performed and an additional CRC sequence of L =24 bits is attached to each code block.

The maximum code block size is Z = 6144. If the number of filler bits F calculated below is not 0, filler bits are added to the beginning of the first block.
All of those numbers were determined by 3GPP_36.212 std.

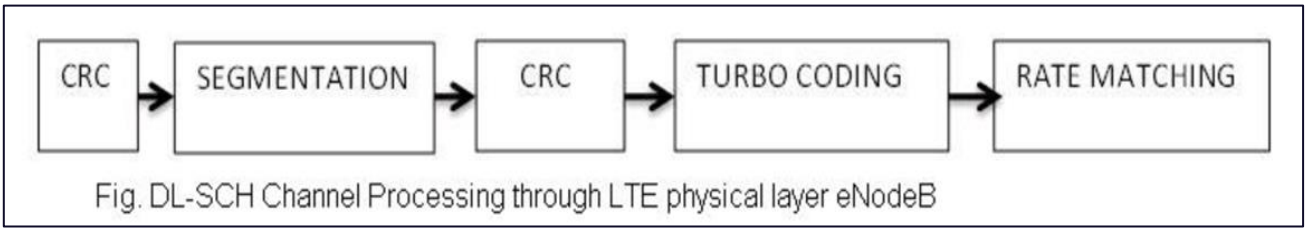
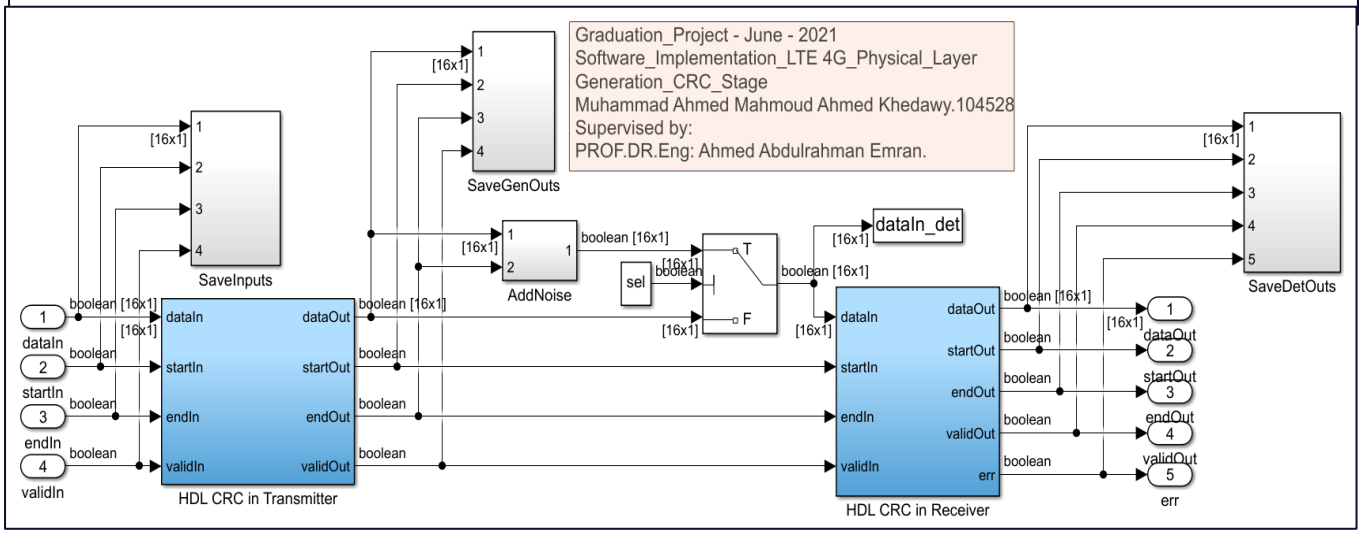


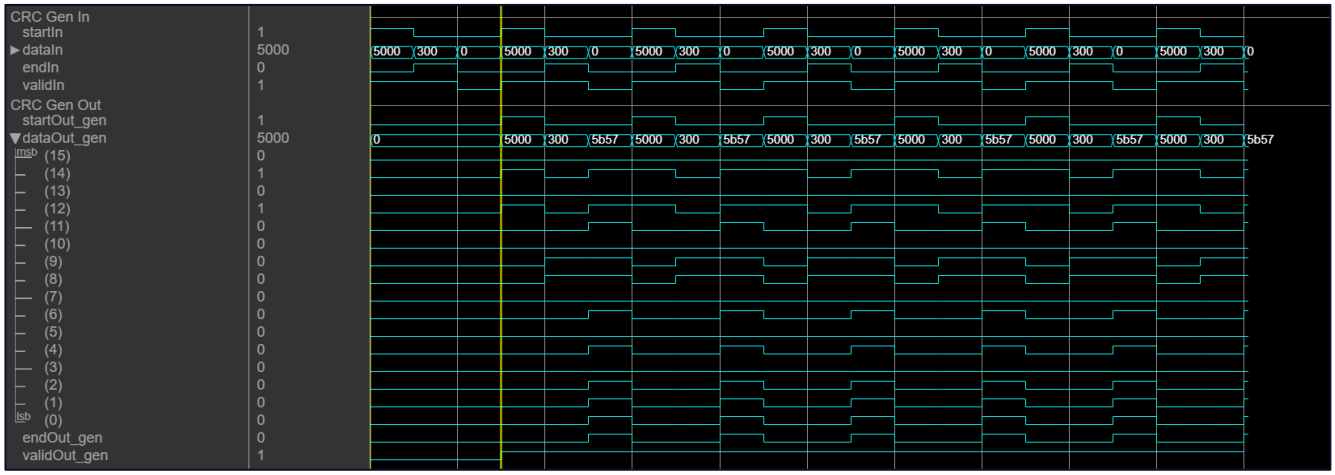
Fig. DL-SCH Channel Processing through LTE physical layer eNodeB

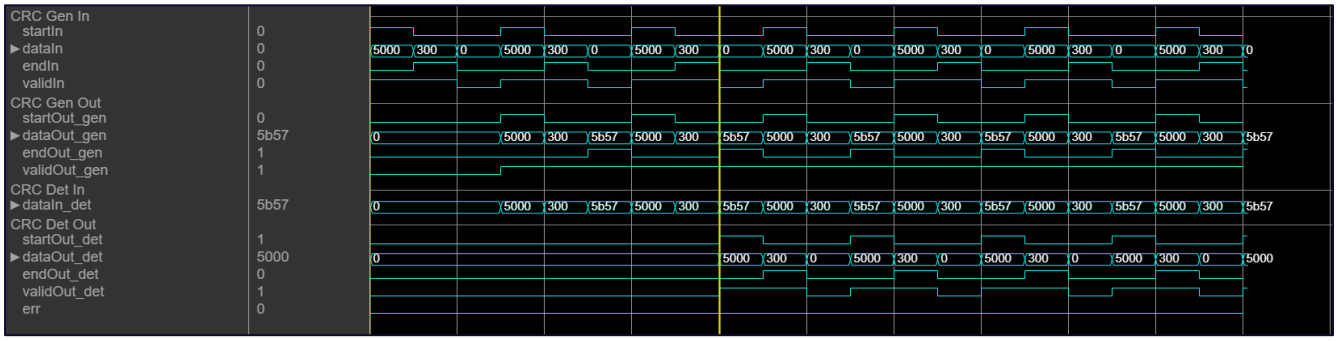
At first the transport block is passed through a CRC encoder; we will use 24-bit CRC method. If the number of bits is more than 6144 bits then it is broken into smaller blocks. It is then turbo coded. Turbo coding is a form of concatenated coding, consisting of two convolutional encoders with certain interleaving between them. Rate matching acts as rate coordinator between preceding and succeeding blocks, it uses a buffer. Modulation used is QAM. It is then passed through a OFDM modulator. The same is shown below in the DL-SCH channel processing figure.

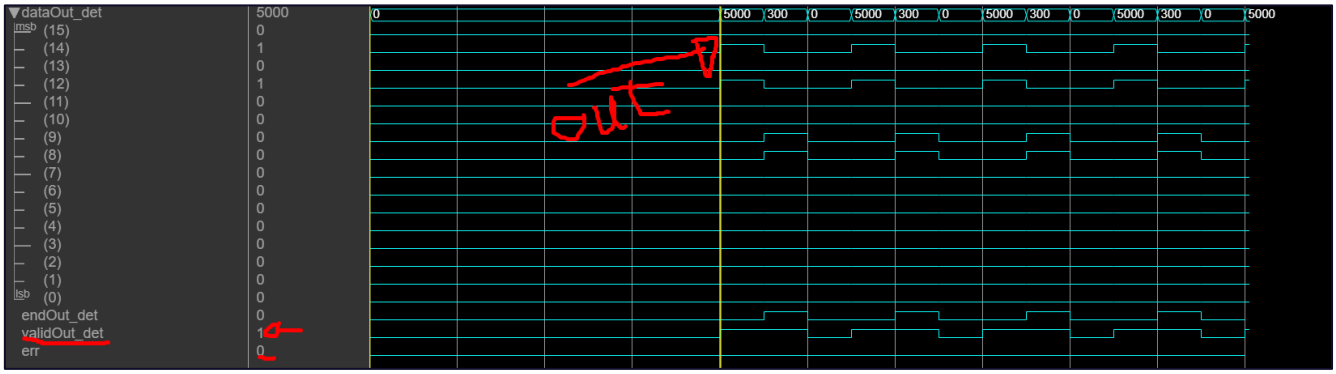
2.4 Synthesis and HDL Coder

```
%Graduation_Project - June - 2021
%Software_Implementation_LTE 4G_Physical_Layer
%Generation_CRC_Check_Error
%Edited by: Muhammad Ahmed Mahmoud Ahmed Khedawy - 104528
%Supervised by: PROF.DR.Eng: Ahmed Abdulrahman Emran.
%showing that how to use the HDL Optimized CRC Generator and CRC Decoding
modelname = 'commcrchdl';
open_system(modelname);
%%
systemname = [modelname '/CRC Subsystem'];
open_system(systemname);
data = [0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0];
crc_len = 16;
%Adding of crc_len zero
msg = [data zeros(1,crc_len)];
dataIn_width = 16;
mlen = length(msg)/dataIn_width;
dlen = length(data)/dataIn_width;
sim(modelname);
%%
h = commcrchdl_plot(dataIn,startIn,endIn,validIn,...
dataOut_gen,startOut_gen,endOut_gen,validOut_gen,...
dataIn_det,dataOut_det,startOut_det,endOut_det,validOut_det,err);
%%
initial_delay_gen =  crc_len/dataIn_width + 2;
initial_delay_det =  4*crc_len/dataIn_width + 4;
displayEndOfDemoMessage(mfilename)
```




CRC Gen In
startIn 1
dataIn 5000
endIn 0
validIn 1
CRC Gen Out
startOut_gen 1
dataOut_gen 5000
msb (15) 0
(14) 1
(13) 0
(12) 1
(11) 0
(10) 0
(9) 0
(8) 0
(7) 0
(6) 0
(5) 0
(4) 0
(3) 0
(2) 0
(1) 0
lsb (0) 0
endOut_gen 0
validOut_gen 1


CRC Gen In
startIn 0
dataIn 0
endIn 0
validIn 0
CRC Gen Out
startOut_gen 0
dataOut_gen 5b57
endOut_gen 1
validOut_gen 1
CRC Det In
dataIn_det 5b57
CRC Det Out
startOut_det 1
dataOut_det 5000
endOut_det 0
validOut_det 1
err 0


dataOut_det 5000
msb (15) 0
(14) 1
(13) 0
(12) 1
(11) 0
(10) 0
(9) 0
(8) 0
(7) 0
(6) 0
(5) 0
(4) 0
(3) 0
(2) 0
(1) 0
lsb (0) 0
endOut_det 0
validOut_det 1
err 0
OUT ?

3 Turbo Codes

Turbo codes is a coding scheme consisting of two parallel recursive systematic convolutional encoders first introduced by Berrou in 1993.

3.1 A Brief History of Turbo Codes

The invention of turbo codes is not the outcome of a mathematical development. In information theory, turbo codes (originally in French *Turbocodes*) are a class of high-performance forward error correction (FEC) codes developed around 1990–91, but first published in 1993. They were the first practical codes to closely approach the maximum channel capacity or Shannon limit, a theoretical maximum for the code rate at which reliable communication is still possible given a specific noise level. Turbo codes are used in 3G/4G mobile communications (e.g., in UMTS and LTE) and in (deep space) satellite communications as well as other applications where designers seek to achieve reliable information transfer over bandwidth- or latency-constrained communication links in the presence of data-corrupting noise. Turbo codes compete with low-density parity-check (LDPC) codes, which provide similar performance.[1]

The name "turbo code" arose from the feedback loop used during normal turbo code decoding, which was analogized to the exhaust feedback used for engine turbocharging. Hagenauer has argued the term turbo code is a misnomer since there is no feedback involved in the encoding process.[1]

It is the result of an intuitive experimental approach whose origin can be found in the work of several European researchers: Gerard Battail, Joachim Hagenauer and Peter Hoeher who, at the end of the 80s highlighted the interest of probabilistic processing in receivers. Others before them, mainly in the United States: Peter Elias Michael Tanner, Robert Gallager, etc. had earlier imagined procedures for coding and decoding that were the forerunners of turbo codes.

In a laboratory at cole Nationale Suprieure des Tlcommunications de Bretagne (Telecom Bretagne), Claude Berrou and Patrick Adde were attempting to transcribe the Viterbi algorithm with weighted input (SOVA: Soft-Output Viterbi Algorithm), into MOS transistors, in the simplest possible way.

A suitable solution was found after two years which enabled these researchers to form an opinion about probabilistic decoding. Claude Berrou, then Alain Glavieux, pursued the study and observed, after Gerard Battail, that a decoder with weighted input and output could be considered as a signal to noise ratio amplifier. This encouraged them to implement the concepts commonly used in amplifiers, mainly feedback. Perfecting turbo codes involved many very pragmatic stages and also the introduction of neologisms, like ”parallel concatenation” or ”extrinsic information”, nowadays common in information theory jargon. The publication in 1993 of the first results, with a performance 0,5 dB from the Shannon limit, shook the coding community, a gain of almost 3 dB, compared to solutions existing at that time.

3.2 Turbo Encoder

The turbo encoder gets a three-bit stream, every output bit stream with length of the input bit stream.
The scheme of turbo encoder is a *Parallel Concatenated Convolutional Code* (PCCC) with two 8-state constituent encoders and one turbo code internal interleaver. The coding rate of turbo encoder is 1/3. The structure of turbo encoder is illustrated in figure 3.1 The transfer function of the 8-state constituent code for the PCCC is:

$$G(D) = \left[1, \frac{g_0(D)}{g_1(D)}\right],$$

where

$$g_0(D) = 1 + D2 + D3\,,$$
$$g_1(D) = 1 + D + D3\,.$$

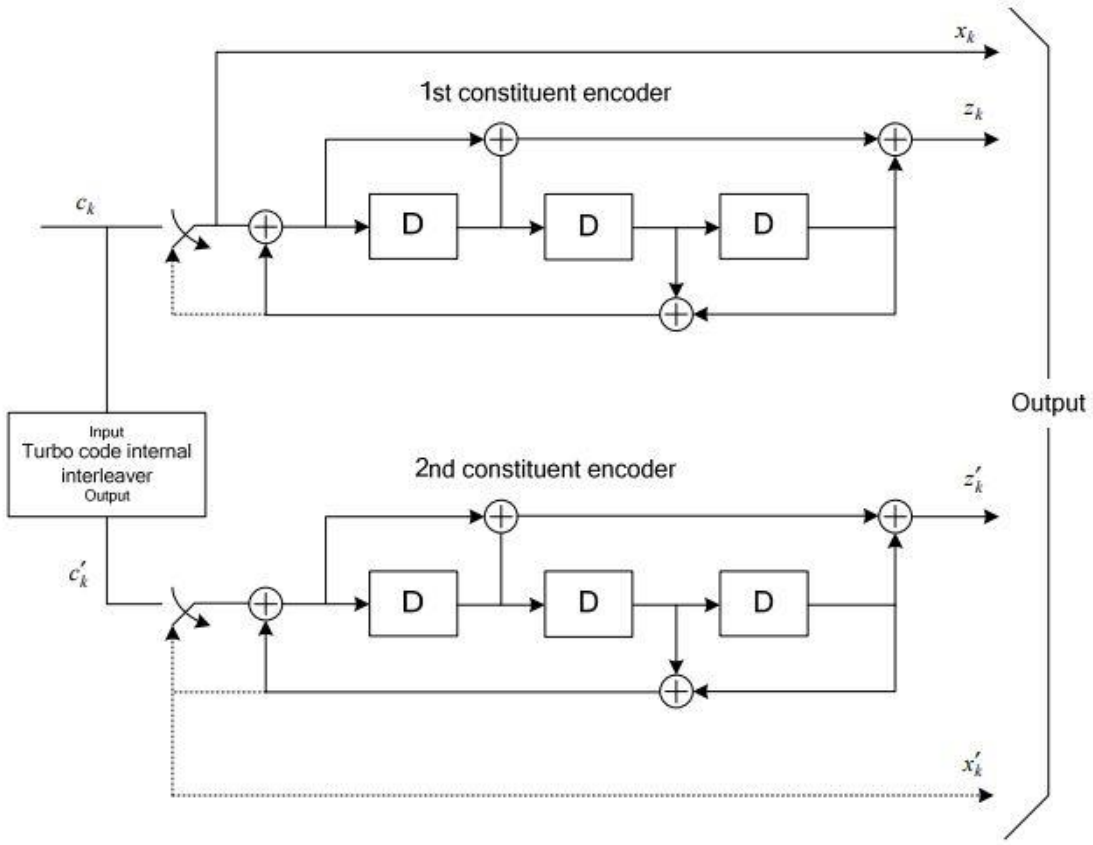


Figure 3-0-1:Structure of rate 1/3 turbo encoder (dotted lines apply for trellis termination only)

The initial value of the shift registers of the 8-state constituent encoders shall be all zeros when starting to encode the input bits. The output from the turbo encoder is

$d_k^{(0)} = x_k$

$d_k^{(1)} = z_k$

$d_k^{(2)} = z'_k$

for $k = 0,1,2, \ldots, K-1$. Where K is the total number of bit stream If the code block to be encoded is the 0^{th} code block and the number of filler bits is greater than zero, i.e., $F > 0$, then the encoder shall set $c_k = 0$, $k = 0,1, \ldots,(F-1)$ at its input where c_k is the input code block and shall set $d_k^{(0)} = \langle NULL \rangle$, $d_k^{(1)} = \langle NULL \rangle$ *for* $k = 0,\ldots,(F-1)$ at its output. The bits input to the turbo encoder are denoted by $c_0, c_1, c_2, c_3, \ldots, c_{k-1}$ and the bits output from the first and second 8- state constituent encoders are denoted by

$$z_0, z_1, z_2, z_3, \ldots, z_{k-1}$$

$$z'_0, z'_1, z'_2, z'_3, \ldots, z'_{k-1}$$

, respectively. The bits output from the turbo code internal interleaver are denoted by $c'_0, c'_1, c'_2, c'_3, \ldots, c'_{k-1}$ and these bits are to be the input to the second 8-state constituent encoder.[2]

3.3 Trellis termination for turbo encoder

The trellis is an algorithm that it encodes to get the minimum possible error that could happen in the decoder to get maximum efficiency to correct the error.
Trellis termination is performed by taking the tail bits from the shift register feedback after all information bits are encoded. Tail bits are padded after the encoding of information bits.

From examining the block diagram of the turbo encoder, we can see that this encoder has a constraint length of 4 (number of delays +1), a generator polynomial matrix of [13 15], (the numbers here is in octal representation) and a feedback connection polynomial of 13 (the numbers here is in octal representation). Therefore, in order to set the trellis structure, we need to use the poly2trellis(4, [13 15],13) function. To construct the LTE interleaver based on the QPP(Quadratic Permutation Polynomial) scheme.

The first three tail bits shall be used to terminate the first constituent encoder (upper switch of figure 3.1 in lower position) while the second constituent encoder is disabled. The last three tail bits shall be used to terminate the second constituent encoder (lower switch of figure 3.1 in lower position) while the first constituent encoder is disabled. The transmitted bits for trellis termination shall then be:

$$d_k^{(0)} = x_k \ , \ d_{k+1}^{(0)} = z_{k+1} \ , \ d_{k+2}^{(0)} = x'_k \ , \ d_{k+3}^{(0)} = z'_{k+1}$$
$$d_k^{(1)} = z_k \ , \ d_{k+1}^{(1)} = x_{k+2} \ , \ d_{k+2}^{(1)} = z'_k \ , \ d_{k+3}^{(1)} = x'_{k+2}$$
$$d_k^{(2)} = x_{k+1} \ , \ d_{k+1}^{(2)} = z_{k+2} \ , \ d_{k+2}^{(2)} = x'_{k+1} \ , \ d_{k+3}^{(2)} = z'_{k+2}$$

the trellis termination is explained by dotted lines in figure 3.1. the tail bits are multiplexed to the end of the three streams, whose lengths are hence increased to (K+4) bits each.

3.4 Turbo Code internal interleaver

The interleaver is a fundamental part of the turbo code design and plays a critical role in the performance of turbo coding.

A block interleaver accepts a set of symbols and rearranges them, without repeating or omitting any of the symbols in the set. The number of symbols in each set is fixed for a given interleaver. Interleaving and deinterleaving can be useful for reducing errors caused by burst errors in a communication system. The process of interleaver that explain the randomization of the error is illustrated in figure 3.2.

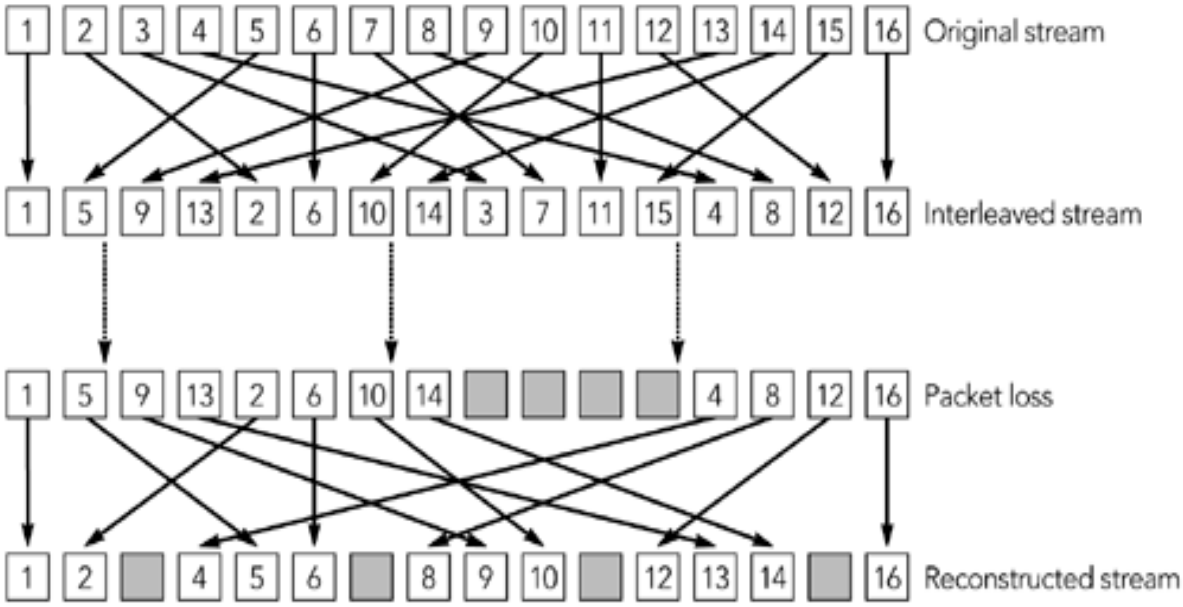


Figure 3-0-2: the process of the interleaver and randomizing the error to correct it.

The bits input to the turbo code internal interleaver are denoted by c_0 , c_1 , c_2 ,c_3 ,…..,c_{k-1} , where K is the number of input bits.The bits output from the turbo code internal interleaver are denoted by $c'_0, c'_1, c'_2, c'_3, \ldots\ldots, c'_{k-1}$. The relationship between the input and output bits is as follows:

$$c'_I = c_{\pi(i)} \qquad \text{where } I = 0, 1, 2, 3, \ldots\ldots, (K-1)$$

where the relationship between the output index I and the input index π_i satisfies the following quadratic form:

$$\pi_i = (f_1 . I + f_2 . i^2) \bmod K$$

The parameters f_1 and f_2 depend on the block size K and are summarized in Table 3.1.

Table 3-0-1: Turbo code internal interleaver parameters

i	K	f_1	f_2	i	K	f_1	f_2	i	K	f_1	f_2	i	K	f_1	f_2
1	40	3	10	48	416	25	52	95	1120	67	140	142	3200	111	240
2	48	7	12	49	424	51	106	96	1152	35	72	143	3264	443	204
3	56	19	42	50	432	47	72	97	1184	19	74	144	3328	51	104
4	64	7	16	51	440	91	110	98	1216	39	76	145	3392	51	212
5	72	7	18	52	448	29	168	99	1248	19	78	146	3456	451	192
6	80	11	20	53	456	29	114	100	1280	199	240	147	3520	257	220
7	88	5	22	54	464	247	58	101	1312	21	82	148	3584	57	336
8	96	11	24	55	472	29	118	102	1344	211	252	149	3648	313	228
9	104	7	26	56	480	89	180	103	1376	21	86	150	3712	271	232
10	112	41	84	57	488	91	122	104	1408	43	88	151	3776	179	236
11	120	103	90	58	496	157	62	105	1440	149	60	152	3840	331	120
12	128	15	32	59	504	55	84	106	1472	45	92	153	3904	363	244
13	136	9	34	60	512	31	64	107	1504	49	846	154	3968	375	248
14	144	17	108	61	528	17	66	108	1536	71	48	155	4032	127	168
15	152	9	38	62	544	35	68	109	1568	13	28	156	4096	31	64
16	160	21	120	63	560	227	420	110	1600	17	80	157	4160	33	130
17	168	101	84	64	576	65	96	111	1632	25	102	158	4224	43	264
18	176	21	44	65	592	19	74	112	1664	183	104	159	4288	33	134
19	184	57	46	66	608	37	76	113	1696	55	954	160	4352	477	408
20	192	23	48	67	624	41	234	114	1728	127	96	161	4416	35	138
21	200	13	50	68	640	39	80	115	1760	27	110	162	4480	233	280
22	208	27	52	69	656	185	82	116	1792	29	112	163	4544	357	142
23	216	11	36	70	672	43	252	117	1824	29	114	164	4608	337	480
24	224	27	56	71	688	21	86	118	1856	57	116	165	4672	37	146
25	232	85	58	72	704	155	44	119	1888	45	354	166	4736	71	444
26	240	29	60	73	720	79	120	120	1920	31	120	167	4800	71	120
27	248	33	62	74	736	139	92	121	1952	59	610	168	4864	37	152
28	256	15	32	75	752	23	94	122	1984	185	124	169	4928	39	462
29	264	17	198	76	768	217	48	123	2016	113	420	170	4992	127	234
30	272	33	68	77	784	25	98	124	2048	31	64	171	5056	39	158
31	280	103	210	78	800	17	80	125	2112	17	66	172	5120	39	80
32	288	19	36	79	816	127	102	126	2176	171	136	173	5184	31	96
33	296	19	74	80	832	25	52	127	2240	209	420	174	5248	113	902
34	304	37	76	81	848	239	106	128	2304	253	216	175	5312	41	166
35	312	19	78	82	864	17	48	129	2368	367	444	176	5376	251	336
36	320	21	120	83	880	137	110	130	2432	265	456	177	5440	43	170
37	328	21	82	84	896	215	112	131	2496	181	468	178	5504	21	86
38	336	115	84	85	912	29	114	132	2560	39	80	179	5568	43	174
39	344	193	86	86	928	15	58	133	2624	27	164	180	5632	45	176
40	352	21	44	87	944	147	118	134	2688	127	504	181	5696	45	178
41	360	133	90	88	960	29	60	135	2752	143	172	182	5760	161	120
42	368	81	46	89	976	59	122	136	2816	43	88	183	5824	89	182
43	376	45	94	90	992	65	124	137	2880	29	300	184	5888	323	184
44	384	23	48	91	1008	55	84	138	2944	45	92	185	5952	47	186
45	392	243	98	92	1024	31	64	139	3008	157	188	186	6016	23	94
46	400	151	40	93	1056	17	66	140	3072	47	96	187	6080	47	190
47	408	155	102	94	1088	171	204	141	3136	13	28	188	6144	263	480

3.5 *Rate Matching*

3.5.1 Rate Matching for transport block Channel

The task of rate matching is to extract from the blocks of code bits, delivered by the Turbo Encoding, the exact set of bits to be transmitted within a given transmission time interval, depending on the existing channel conditions.

The rate matching for turbo coded transport channels is defined per coded block and consists of interleaving the three information bit streams $d_k^{(0)}$, $d_k^{(1)}$, $d_k^{(2)}$,

followed by the collection of bits and the generation of a circular buffer as depicted in Figure 3.4. The output bits for each code block are transmitted as described in section 3.5.3.

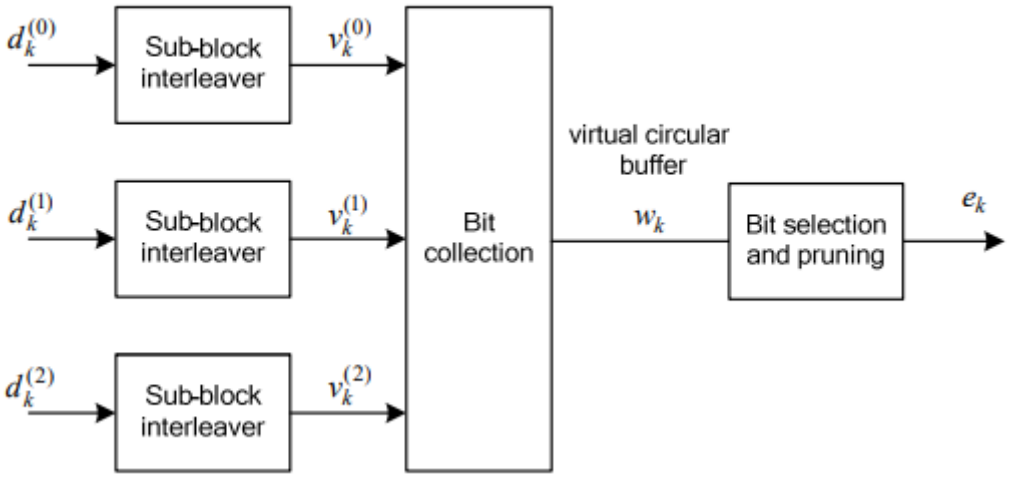


Figure 3-0-3: Rate matching for turbo coded transport channels

The bit stream $d_k^{(0)}$ is interleaved according to the sub-block interleaver defined in section 3.5.2 with an output sequence defined as $v_0^{(0)}$, $v_1^{(0)}$, $v_2^{(0)}$,…… , $v_{k\pi-1}^{(0)}$ and where K_π is defined in section 3.5.2. The bit stream $d_k^{(1)}$ is interleaved according to the sub-block interleaver defined in section 3.5.2with an output sequence defined as $v_0^{(1)}$, $v_1^{(1)}$, $v_2^{(1)}$,…… , $v_{k\pi-1}^{(1)}$. The bit stream $d_k^{(2)}$ is interleaved according to the sub-block interleaver defined in section 3.5.2 with an output sequence defined as $v_0^{(2)}$, $v_1^{(2)}$, $v_2^{(2)}$,…… , $v_{k\pi-1}^{(2)}$. The sequence of bits e_k for transmission is generated according to section 3.5.3.

3.5.2 Sub-block Interleaver

We studied Quadrature Permutation Polynomial (QPP) interleavers, which have several desirable properties such as the simple algebraic construction rule and the MCF property. In particular, the MCF property of interleavers is one of the key design requirements for the parallel decoding implementations of turbo codes, aiming at very high decoding throughput. We proposed the simple criterion to efficiently find good QPP interleavers for various block lengths. Our method finds QPP interleavers which maximize the randomness while certain minimum

spreading property is guaranteed. We also proposed the empirical value for the minimum spreading property constraint which is a function of the block length. The bits input to the block interleaver are denoted by $d^{(i)}_0$, $d^{(i)}_1$, $d^{(i)}_{2,\ldots\ldots,}$ $d^{(i)}_{D-1}$, where D is the number of bits. The output bit sequence from the block interleaver is derived as follows:

(1) Assign $C^{TC}_{subblock}$= 32 to be the number of columns of the matrix. The columns of the matrix are numbered 0, 1, 2, ……, $C^{TC}_{subblock}$ -1 from left to right.

(2) Determine the number of rows of the matrix $R^{TC}_{subblock}$ by finding minimum integer $R^{TC}_{subblock}$, such that:
$D \leq (R^{TC}_{subblock} \times C^{TC}_{subblock},)$ The rows of rectangular matrix are numbered 0, 1, 2,…, $R^{TC}_{subblock}$ -1 from top to bottom.

(3) if (RTCsubblock x CTCsubblock,) < D then ND= (RTCsubblock x CTCsubblock - D) dummy bits are padded such that yk = <NULL> for $K = 0, 1, 2, \ldots\ldots, ND - 1$. Then $y\,ND + k = d(i)k$ where $k = 0,1,2, \ldots, D - 1$ and the bit sequence yk is written into the (RTCsubblock x CTCsubblock,) matrix row by row starting with bit y0 in column 0 of row 0:

$$\begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_{C^{TC}_{subblock}-1} \\ y_{C^{TC}_{subblock}} & y_{C^{TC}_{subblock}+1} & y_{C^{TC}_{subblock}+2} & \cdots & y_{2C^{TC}_{subblock}-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}} & y_{(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}+1} & y_{(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}+2} & \cdots & y_{(R^{TC}_{subblock}\times C^{TC}_{subblock}-1)} \end{bmatrix}$$

For $d_k^{(0)}$, $d_k^{(1)}$

Perform the inter-column permutation for the matrix based on the pattern $\langle P\langle j\rangle\rangle$ where j = 0,1,…, $C^{TC}_{Subblock}$ that is shown in table 3.2 and P(j) is the original column position of the j-th permuted column.
After permutation of the columns, the inter-column permuted ($R^{TC}_{subblock}$ x $C^{TC}_{subblock}$,) matrix is equal to

$$\begin{bmatrix} y_{P(0)} & y_{P(1)} & y_{P(2)} & \cdots & y_{P(C^{TC}_{subblock}-1)} \\ y_{P(0)+C^{TC}_{subblock}} & y_{P(1)+C^{TC}_{subblock}} & y_{P(2)+C^{TC}_{subblock}} & \cdots & y_{P(C^{TC}_{subblock}-1)+C^{TC}_{subblock}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{P(0)+(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}} & y_{P(1)+(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}} & y_{P(2)+(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}} & \cdots & y_{P(C^{TC}_{subblock}-1)+(R^{TC}_{subblock}-1)\times C^{TC}_{subblock}} \end{bmatrix}$$

The output of the block interleaver is the bit sequence read out column by column from the inter-column permuted ($R^{TC}_{subblock}$ x $C^{TC}_{subblock}$,) matrix. The bits after sub-block interleaving are denoted by $v_0^{(0)}$, $v_1^{(0)}$, $v_2^{(0)}$,…… , $v_{k\pi-1}^{(0)}$

Where

$v_0^{(i)}$ corresponds to $y_{p(0)}$

$v_1^{(i)}$ corresponds to $y_{p(0)+ CTCsubblock}$ …etc

$K_\pi = (R^{TC}_{subblock}$ x $C^{TC}_{subblock},)$

For $d_k^{(2)}$

(4) The output of the sub-block interleaver is denoted by $v_0^{(2)}$, $v_1^{(2)}$, $v_2^{(2)}$,…… , $v_{k\pi-1}^{(2)}$, where $v_k^{(2)} = y_{\pi(k)}$ and where

$$\pi(k) = \left(P\left(\left\lfloor \frac{K}{R^{TC}_{Subblock}} \right\rfloor\right) + C^{TC}_{Subblock} * \left(K \bmod R^{TC}_{Subblock}\right) + 1\right) \bmod K_\pi$$

The permutation function P is defined in Table 3.1.

Number of columns $C^{TC}_{subblock}$	Inter-column permutation pattern $< P(0), P(1),..., P(C^{TC}_{subblock}-1) >$
32	< 0, 16, 8, 24, 4, 20, 12, 28, 2, 18, 10, 26, 6, 22, 14, 30, 1, 17, 9, 25, 5, 21, 13, 29, 3, 19, 11, 27, 7, 23, 15, 31 >

Table 3-2:Inter-column permutation pattern for sub-block interleaver

3.5.3 Bit collection, selection and transmission

In the circular buffer rate matching (CBRM) method for rate 1/3 turbo codes , each of the three output streams of the turbo coder is rearranged with its own sub-block interleaver. Then, a single output buffer is formed by placing the rearranged systematic bits in the beginning followed by bit-by-bit interlacing of the two rearranged parity streams (see Figure 3.4).

Interlacing allows equal levels of protection for each constituent code. Since each sub-block interleaver is typically based on row-column permutations, the CB can also be visualized as in 2-dimensional format (1-dimensional CB is obtained by reading bits column-by column).

Assuming 32 columns in each sub-block interleaver, the first 32 columns represent the 32 columns of the systematic bits after sub-block interleaving, while

the remaining 64 columns represent bit-by-bit interlaced columns of the permuted Parity 1 and Parity 2 streams.
For a desired code rate, the number of coded bits N_{data} to be selected for transmission is passed to the RM algorithm.

The Rate Monotonic scheduling algorithm is a simple rule that assigns priorities to different tasks according to their time period. That is task with smallest time period will have highest priority and a task with longest time period will have lowest priority for execution.
The bit selection step of the CBRM simply reads out the first N_{data} bits from the start of the buffer. In general, the bits to be selected for transmission can be read out starting from any point in the buffer. If the end of the buffer is reached, then the reading continues by wrapping around to the beginning of the buffer (hence the term circular buffer). Thus, puncturing and repetition is achieved using a unified method.
The CRBM algorithm has advantages in flexibility (in code rates achieved) and also granularity (in stream sizes). Typically, low complexity sub-block interleavers that facilitate uniform puncturing at any code rate are preferred. Figure 3.5 explaining the block diagram of circular buffer.

The circular buffer of length $K_w = 3K_\pi$ for the r^{th} coded block is generated as follows:
$W_k = v^{(0)}_k$ for $k = 0,\ldots, K_\pi - 1$
$W_{k\pi+2k} = v^{(1)}_k$ for $k = 0,\ldots, K_\pi - 1$
$W_{k\pi+2k+1} = v^{(2)}_k$ for $k = 0,\ldots, K_\pi - 1$
Denote the soft buffer size for the transport block by N_{IR} bits and the soft buffer size for the r^{th} code block by N_{cb} bits.
The size N_{cb} is obtained as follows, where C is the number of code blocks.
Where $N_{cb} = min\left(\frac{N_{IR}}{C}, K_W\right)$
Denoting by E the rate matching output sequence length, the rate matching output bit sequence is e_k , k = 0,1,..., E −1.
Set k = 0 and j = 0
while { k < E }
if $w_{j \bmod kw} \neq <$ NULL
$e_k = w_{j \bmod Kw}$
k = k +1
end if

```
        j = j +1
end while
```

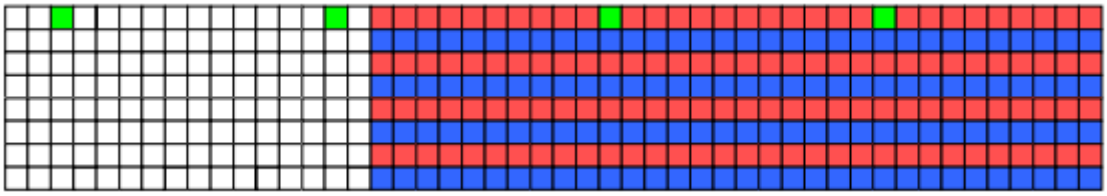


Figure 3.4: Conceptual composition of a circular buffer for LTE turbo code rate matching. White, red and blue cells contain bits from the Systematic, Parity 1 and Parity 2 streams, respectively. Green cells mark starting points of redundancy versions defined for RM. The number of columns for each stream is 32 (not shown to scale).

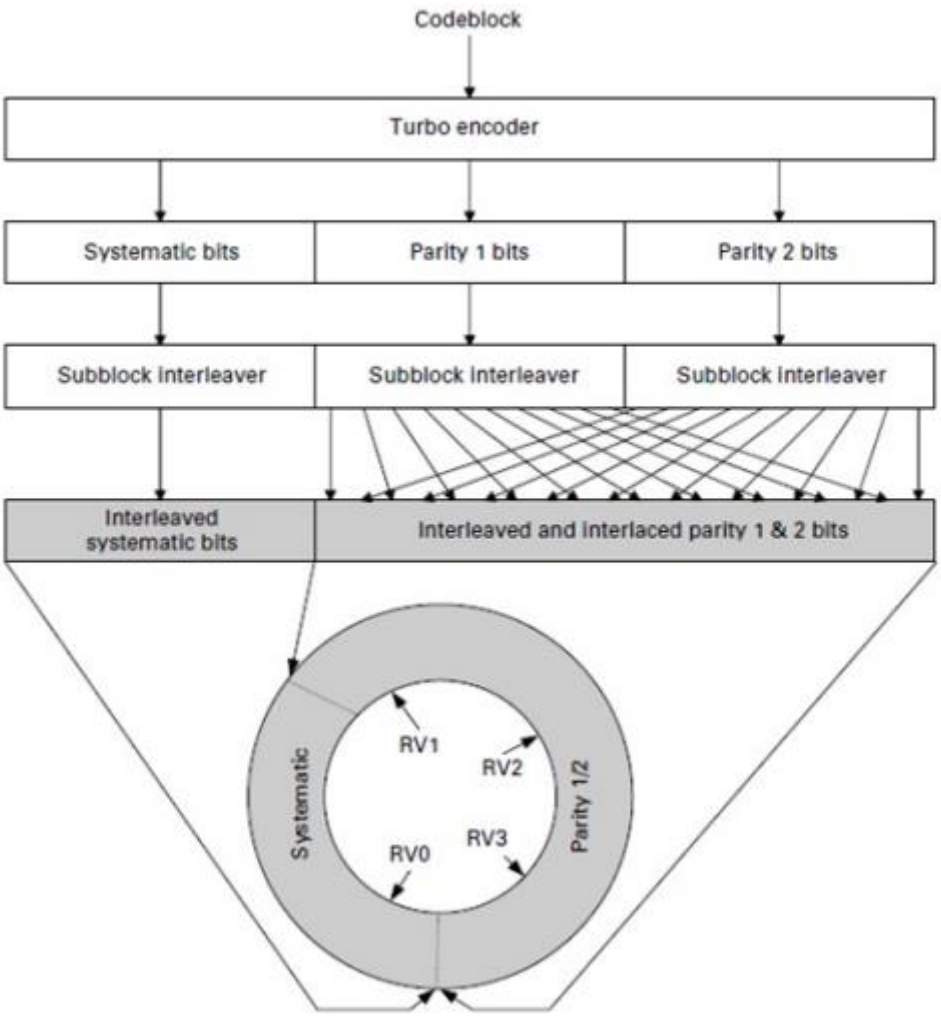


Figure 3.5: Circular-buffer rate matching for turbo

3.5.4 Puncturing

Puncturing is the process of removing certain symbols/positions from the codeword, thereby reducing the codeword length and increasing the overall code rate. In the original turbo code. punctured half of the bits from each constituent encoder. Puncturing half of the systematic bits from each constituent encoder corresponds to sending all the systematic bits once if the puncturing is properly performed.

The overall code rate is R = 1/2 . Furthermore, puncturing may have different effect for different choices of interleavers, and for different constituent encoders.

When puncturing is considered, for example, some output bits of v_0, v_1 and v_2 are deleted according to a chosen pattern defined by a puncturing matrix P.
For instance, a rate 1/2 turbo code can be obtained by puncturing a rate 1/3 turbo code. Commonly used puncturing matrix is given by

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

where the puncturing period is 2. According to the puncturing matrix, the parity check digits from the two component encoders are alternately deleted. The punctured turbo code symbol at a given time consists of an information digit followed by a parity check digit which is alternately obtained from the first and the second component encoder

3.5.5 Code block concatenation

The code block concatenation is putting the code from parallel representation to a serial representation.The input bit sequence for the code block concatenation block is the sequences e_{rk} , for r = 0,...,C −1 and k = 0,..., E_r −1.
The output bit sequence from the code block concatenation block is the sequence f_k for k = 0,...,G −1 .

The code block concatenation consists of sequentially concatenating the rate matching outputs for the different code blocks.

Therefore, Set k = 0 and r = 0
while r < C
Set j = 0
while j <E_r
$f_k = e_{rj}$
k = k +1
j = j +1
end while
r = r +1
end while

3.6 Turbo decoder:

In the receiver, the turbo decoder inverts the operations performed by the turbo encoder.A Turbo decoder consists of two single soft-in soft-out (SISO) decoders that work iteratively and concatenated serially via an interleaver, identical to the one in the encoder. The output of the first (upper decoder) feeds into the second to form a Turbo decoding iteration as shown in fig (3.4).

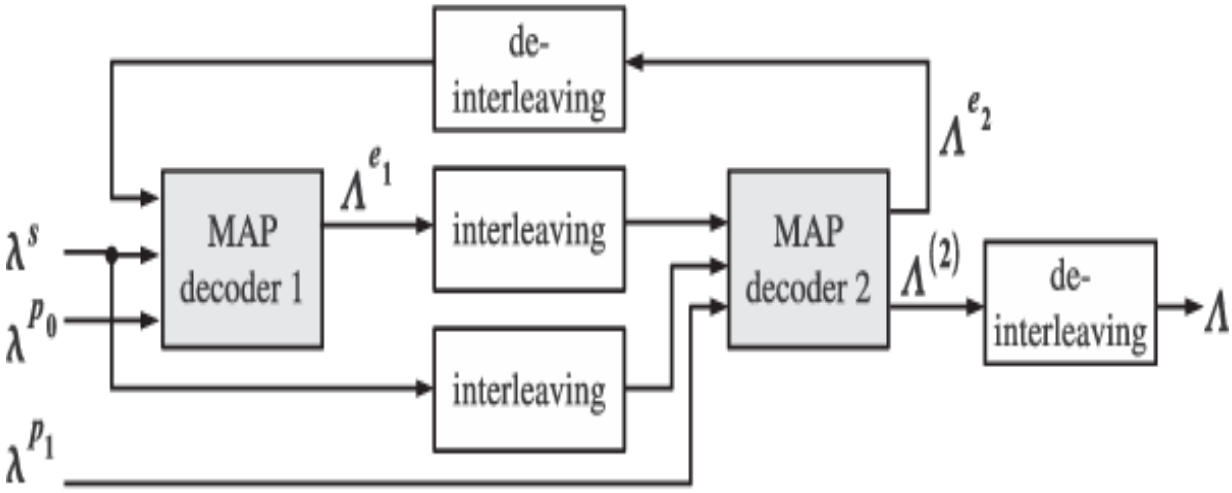


Figure 3-0-4: Turbo decoding block diagram

Where performance improves as more iterations are gone through The same trellis structure found in the turbo encoder is used in the APP decoder, as is the same interleaver.

The difference is that turbo decoding is an iterative operation. The performance and the computational complexity of a turbo decoder directly relate to the number of iterations performed. At the receiver, the turbo decoder performs the inverse operation of a turbo encoder. By processing its input signal, which is the output of a demodulator and descrambler, the turbo decoder will recover the best estimate of the TrCH transmitted bits. Note that the turbo decoder input needs to be expressed in LLRs. As we saw earlier, LLRs are generated by the demodulator if soft-decision demodulation is performed Similarly, the TurboDecoder function operates on its first input signal (u), which is the LLR output of the demodulator and descrambler. The turbo decoder will recover the best estimate of the transmitted bits. The function also takes as inputs the interleaving indices (intrlvrIndices) and the maximum number of iterations used in the decoder (maxIter).

To set the trellis structure, we use the ploy2trellis function of the Communications System Toolbox. This function transforms the encoder connection polynomials to a trellis structure. As the LTE trellis structure has both feed-forward and feedback connection polynomials, we first build a binary-number representation of the polynomials and then convert the binary representation into an octal representation. From examining the block diagram of the turbo encoder in Figure 4.3, we can see that this encoder has a constraint length of 4, a generator polynomial matrix of [13 15], and a feedback connection polynomial of 13.

Therefore, in order to set the trellis structure, we need to use the poly2trellis(4, [13 15],13) function. To construct the LTE interleaver based on the QPP scheme2

The same trellis structure found in the turbo encoder is used in the APP decoder, as is the same interleaver. The difference is that turbo decoding is an iterative operation. The performance and the computational complexity of a turbo decoder directly relate to the number of iterations performed. At the receiver, the turbo decoder performs the inverse operation of a turbo encoder. By processing its input signal, which is the output of a demodulator and descrambler, the turbo decoder will recover the best estimate of the TrCH transmitted bits. Note that the turbo decoder input needs to be expressed in LLRs. As we saw earlier, LLRs are generated by the demodulator if soft-decision demodulation is performed Similarly, the TurboDecoder function operates on its first input signal (u), which is the LLR output of the demodulator. The turbo decoder will recover the best estimate of the transmitted bits. The function also takes as inputs the interleaving indices (intrlvrIndices) and the maximum number of iterations used in the decoder (maxIter).

To set the trellis structure, we use the ploy2trellis function of the Communications System Toolbox. This function transforms the encoder connection polynomials to a trellis structure. As the LTE trellis structure has both feed-forward and feedback connection polynomials, we first build a binary-number representation of the polynomials and then convert the binary representation into an octal representation.

The comm.TurboDecoder stepimpl repeats a sequence of operations, including two APP decoders and interleavers, N times. The value of N corresponds to the maximum number of iterations in a turbo decoder. At the end of each processing iteration, the turbo decoder uses the results to update its best estimate.

The performance of any turbo coder depends on the number of iterations performed in the decoding operation. This means that for a given turbo encoder

(e.g., the one specified in the LTE standard), the BER performance becomes successively better with a greater number of iteration.

SISO (Soft Input/Soft Output) algorithms are well suited for iterative decoding because they accept a priori information at their input and produce a posteriori information at their output. In turbo decoding, trellis based decoding algorithms are used. These are recursive methods suitable for the estimation of the state sequence of a discrete time finite-state Markov process observed in memory less noise. With reference to decoding of noisy coded sequences, The MAP algorithm is used to estimate the most likely information bit to have been transmitted in a coded sequence. Here, we only discuss the iterative decoding of two dimensional turbo codes. The extension to the case of multidimensional concatenated codes is straightforward.

3.7 Iterative Decoding Principle

3.7.1 BCJR Algorithm:

The Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm, also known as the forward-backward or the a posteriori probability algorithm, or Maximum a posteriori algorithm, is the core component in many iterative detection and decoding schemes. BCJR algorithm is optimal for estimating the states or the outputs of a Markov process observed in white noise. It produces the sequence of A Posteriori Probabilities (APP), where is the APP of the data bit given all the received sequence.

The numerical representation of probabilities, non-linear functions and mixed multiplications and additions of these values perhaps make this algorithm too difficult to implement. As a result, different derivatives of this algorithm such as Log-MAP and Max-Log-MAP algorithm have been used in the decoding of turbo codes.

3.7.2 Tools for Iterative Decoding of Turbo Codes Log-likelihood Algebra.

The log-likelihood ratio of a binary random variable u_k, $L(u_k)$ is defined as

$$L(u_k) = \ln P(u_k = +1)\, P(u_k = -1) \qquad (3.1)$$

where u_k is information bit at time k Since

$$P(u_k = +1) = 1 - P(u_k = -1) \quad (3.2)$$

$$L(u_k) = \ln P(u_k) = +1)\ 1 - P(u_k = +1) \quad (3.3)$$

Simplifying we find

$$P(u_k = \pm 1) = ((\ e^{-L(uk)/2})/(\ 1 + e^{-L(uk)/2}\ )) \cdot e^{uk \cdot L(uk)/2} = A_k e^{uk \cdot L(uk)/2} \quad (3.4)$$

Where

$$A_k = (\ e^{-L(uk)/2}\ )/(1+e^{-L(uk)/2}\ )$$

is a common factor.

If the binary random variable u_k is conditioned on a different random variable or vector y_k then we have a conditioned log-likelihood $L(u_k|y_k)$ ratio with

$$L(u_k|y_k) = \ln P(u_k = +1|y_k)\ /P(u_k = -1|y_k)$$
$$= \ln P(yk|u_k = +1) \cdot P(u_k = +1)\ /P(y_k|u_k = -1) \cdot P(u_k = -1)$$
$$= \ln P(y_k|u_k = +1)\ P(y_k|u_k = -1) + \ln P(u_k = +1)\ P(u_k = -1)$$
$$= L(y_k|u_k) + L(u_k) \quad (3.5)$$

Soft Channel Outputs After transmission over a channel with a fading factor a and additive Gaussian noise,

$$L(u_k|y_k) = \frac{\ln P(yk|uk = +1) \cdot P(uk = +1)}{P(yk|uk = -1) \cdot P(uk = -1)}$$
$$= ln\ \frac{\exp(-\ Es\ N0\ (yk - a)^2)}{\exp(-\ Es\ N0\ (yk + a)^2)} + ln\ \frac{P(uk = +1)}{P(uk = -1)}$$
$$= 4 \cdot \frac{Es}{No} \cdot\ y_k + L(u_k)$$
$$= L_c \cdot y_k + L(u_k) \quad (3.6)$$

where $L_c = 4 \cdot \frac{Es}{No}$ For a fading channel, a denotes the fading amplitude whereas for a Gaussian channel , we set a = 1 Since

$$P(y_k) = P(y_k|u_k = +1) \cdot P(u_k = +1)$$

$$P(y_k) = P(y_k|u_k = -1) \cdot P(u_k = -1) \qquad (3.7)$$

and using the previous equations, we can prove that

$$p(y_k|u_k) = B_k \cdot e^{u_k \cdot Lc \cdot y_k/2} \qquad (3.8)$$

where

$$\mathbf{B_k} = \frac{\mathbf{P(y_k)\cdot(1+e^{-L(u_k)})\cdot e^{-Lc\cdot y_k/2}}}{\mathbf{1+e^{-L(u_k)+Lc\cdot y_k}}}$$

3.7.3 Principle of the Iterative Decoding Algorithm:

Assume that we have a '' soft-in/soft-out '' decoder available as shown in Figure 3.5

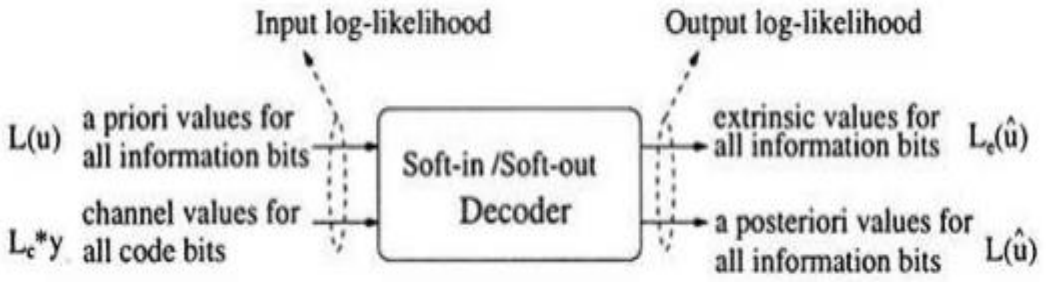


Figure 30-5: soft-in/soft out decoder

for decoding of the component codes. The output of symbol-by-symbol Maximum a posteriori Probability (MAP) decoder is defined as the a posteriori log-likelihood ratio, that is, the logarithm of the ratio of the probabilities of a given bit being +1 or -1 given the observation y.

$$L(\hat{u}) = L(u|y) = ln\ P(u = +1|y)\ P(u = -1|y) \qquad (3.9)$$

Such a decoder uses a priori values L(u) for all information bits u , if available, and the channel values Lc · y for all coded bits. It also delivers soft outputs L(ˆ(u)) on all information bits and an extrinsic information Le(ˆ(u)) which contains the soft output information from all the other coded bits in the code sequence and is not influenced by the L(u) and Lc · y values of the current bit.

For systematic codes, the soft output for the information bit u will be represented as the sum of three terms

$$L(\hat{u}) = Lc \cdot y + L(u) + Le(\hat{u}) \qquad (3.10)$$

This means that we have three independent estimates for the log-likelihood ratio of the information bits: the channel values the a priori values L(u) and the values by a third independent estimator utilizing the code constraint. The whole procedure of iterative decoding with two Softin/Softout decoders is shown in Figure 3.6

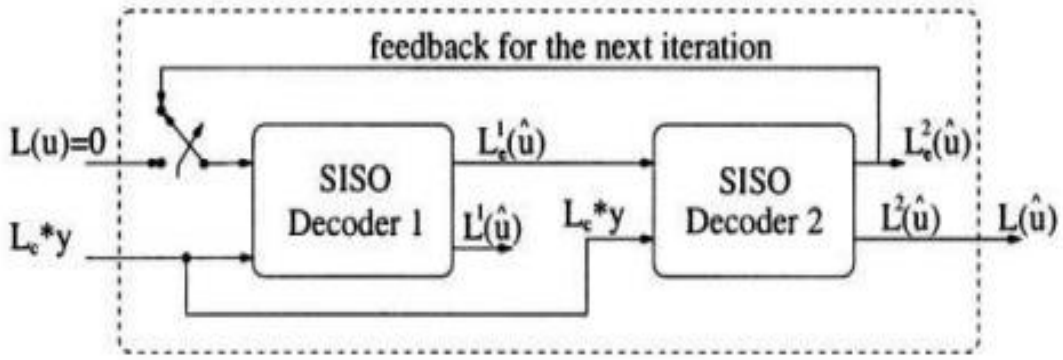


Figure 3-0-6: Iterative decoding procedure with two soft-in/soft out decoder

In the first iteration of the iterative decoding algorithm, Decoder 1 computes the extrinsic information

$$L\ 1\ e\ (\hat{u}) = L\ 1 - [Lc \cdot y + L(u)] \qquad (3.11)$$

We assume equally likely information bits: thus, we initialize L(u) = 0 for the first iteration. This extrinsic information from the first decoder, is passed to the Decoder 2, which uses L 1 e (ˆu) as the a priori value in place of L(u) to compute L 2 e (ˆu) Hence, the extrinsic information value computed by Decoder 2 is

$$L\ 2\ e\ (\hat{u}) = L\ 2\ (\hat{u}) - [Lc \cdot y + L\ 1\ e\ (u)] \quad (3.12)$$

Then, Decoder 1 will use the extrinsic information values L 2 e (ˆ(u)) as a priori information in the second iteration. The computation is repeated in each iteration. The iterative process is usually terminated after a predetermined number of iterations, when the soft-output value L 2 e (ˆ(u)) stabilizes and changes little

between successive iterations. In the final iteration, Decoder 2 combines both extrinsic information values in computing the soft-output values

$$L\,2\,(\hat{u}) = Lc \cdot y + L\,1\,e\,(\hat{u}) + L\,2\,e\,(\hat{u}) \qquad (3.13)$$

3.8 Stopping Criteria for Turbo Decoding:

Iterative decoding is a key feature of turbo codes. Each decoding iteration results in additional computations and decoding delay. As the decoding approaches the performance limit of a given turbo code, any further iteration results in very little improvement. Often, a fixed number M is chosen, and each frame is decoded for M iterations. Usually M is set with the worst corrupted frames in mind. Most frames need fewer iterations to converge. Therefore, it is important to devise an efficient criterion to stop the iteration process and prevent unnecessary computations and decoding delay.

HDA Although iterative decoding improves the LLR value for each information bit through iterations, the hard decision of the information bit is ultimately made based on the sign of its LLR value. The hard decisions of the information sequence at the end of each iteration provide information on the convergence of the iterative decoding process. At iteration (i − 1), we store the hard decisions of the information bits based on $L^{(i-1)}{}_2(\hat{u})$ and check the hard decisions based on $L^{(i)}{}_2(\hat{u})$ at iteration If they agree with each other for the entire block, we simply terminate the iterative process at iteration i This stopping criterion is called the hard-decision-aided (HDA) criterion.

IHDA Although iterative decoding improves the LLR value ($L(\hat{u}_k)$) for each information bit through iterations, the hard decision of the information bit is ultimately made based on the sign of its LLR value. From repeated simulations, it was observed that, as the number of iterations used increases, for a good (easy to decode) frame, the magnitudes of the LLRs gradually become larger. Since the term $L_c \cdot y$ is fixed for every iteration, the increase in the magnitudes of the LLRs is due to increases in the magnitudes of the extrinsic information. Since the extrinsic information keeps increasing as the number of iteration i increases, it is conceivable, as the decoding iteration converges to the final stage, the hard decision based on $Lc \cdot y + L^{(i)}{}_{e1}(\hat{u})$ from the first component decoder should

agree with the hard decision based on the LLR at the output of the second component decoder1 according to the following equation

$$\mathbf{L^{2}(\hat{u}) = L_c \cdot y + L^{1}_{e}(\hat{u}) + L^{2}_{e}(\hat{u})} \qquad \mathbf{(3.14)}$$

At iteration i, compare the hard decisions of the information bit based on $Lc \cdot y + L^{(i)}_{e1}(\hat{u})$ with the hard decision based on $L^{(i)}_{2}(\hat{u})$. If they agree with each other for the entire block, terminate the iterative process at iteration i.

4 Modulation

Why modulation?.....There are many reasons for modulation, but the most important reason is the length of antenna.

As we know length of antenna is inversely proportional to its frequency.

Digital Modulation Types:

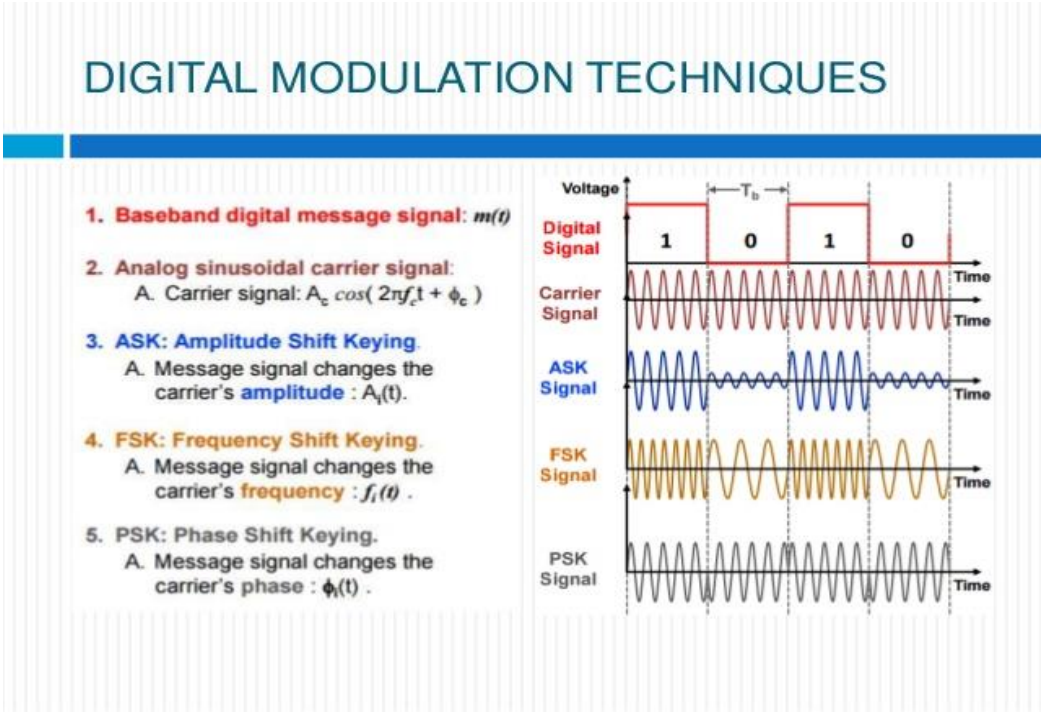


Figure 4-0-1:Digital Modulation Types.

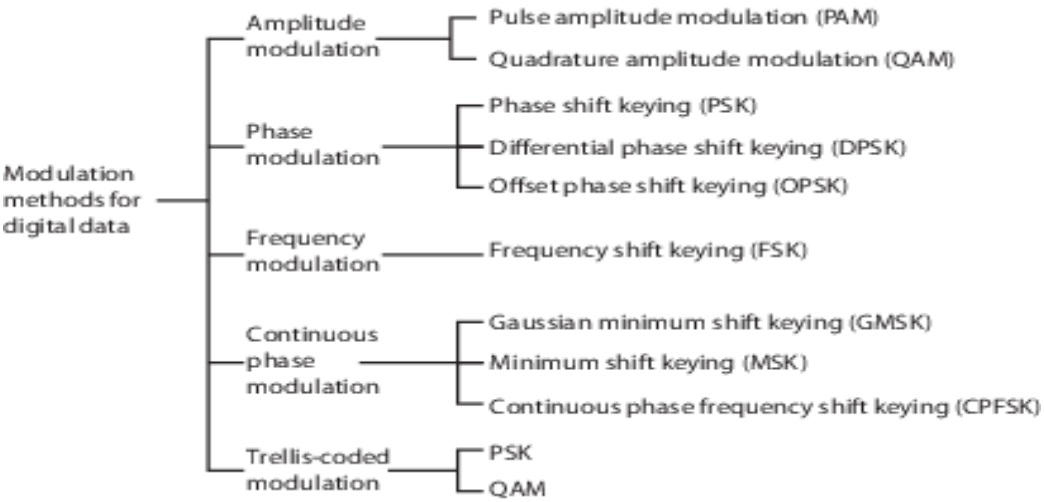


Figure 4-0-2 Digital Modulation Types

Modulation – Downlink Alphabet

Table below shows the standard type of modulation in LTE:

Modulation schemes

Physical channel	Modulation schemes
PUSCH	QPSK, 16QAM, 64QAM
PDSCH	QPSK, 16QAM, 64QAM
PMCH	QPSK, 16QAM, 64QAM
PBCH	QPSK
PCFICH	QPSK
PDCCH	QPSK
PHICH	BPSK

PUSCH	Physical uplink shared channel
PDSCH	Physical downlink shared channel
PMCH	Physical multicast channel
PBCH	Physical broadcast channel
PCFICH	Physical control format indicator channel
PDCCH	Physical downlink control channel
PHICH	Physical hybrid-ARQ indicator channel

Figure 4-0-3 Standard type of modulation in LTE

Physical Downlink Shared Channel

The PDSCH is utilized basically for data and multimedia transport. It therefore is designed for very high data rates.
Modulation options therefore include QPSK, 16QAM and 64QAM. Spatial multiplexing is also used in the PDSCH. In
fact, spatial multiplexing is exclusive to the PDSCH. It is not used on either the PDCCH or the CCPCH.

Physical Downlink Control Channel

The PDCCH conveys UE-specific control information. Robustness rather than maximum data rate is therefore the chief consideration QPSK is the only available modulation format. The PDCCH is mapped onto resource elements in up tothe first three OFDM symbols in the first slot of a subframe.

Common Control Physical Channel

The CCPCH carries cell-wide control information. Like the PDCCH, robustness rather than maximum data rate is the chief consideration. QPSK is therefore the only available modulation format. In addition, the CCPCH is transmitted as close to the center frequency as possible. CCPCH is transmitted exclusively on the 72 active subcarriers centered on the DC subcarrier.

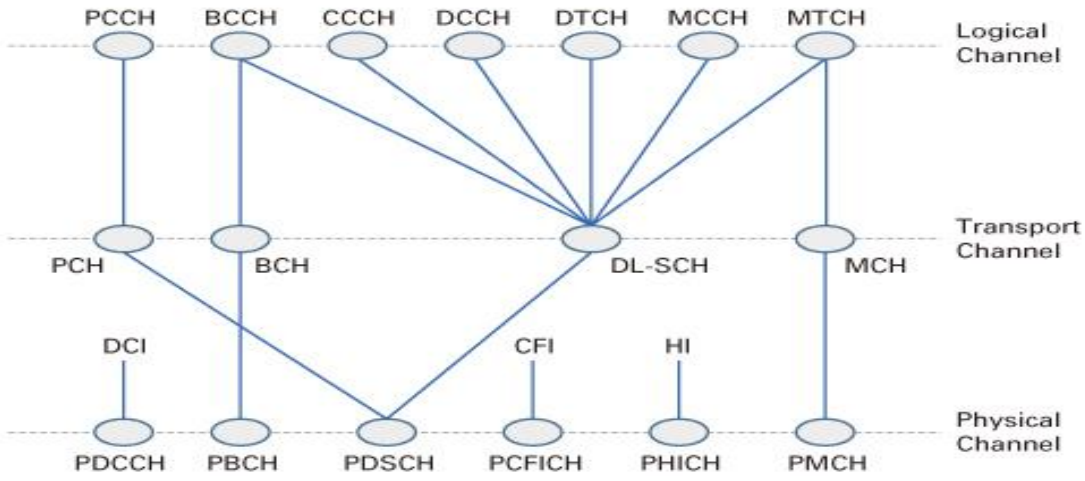
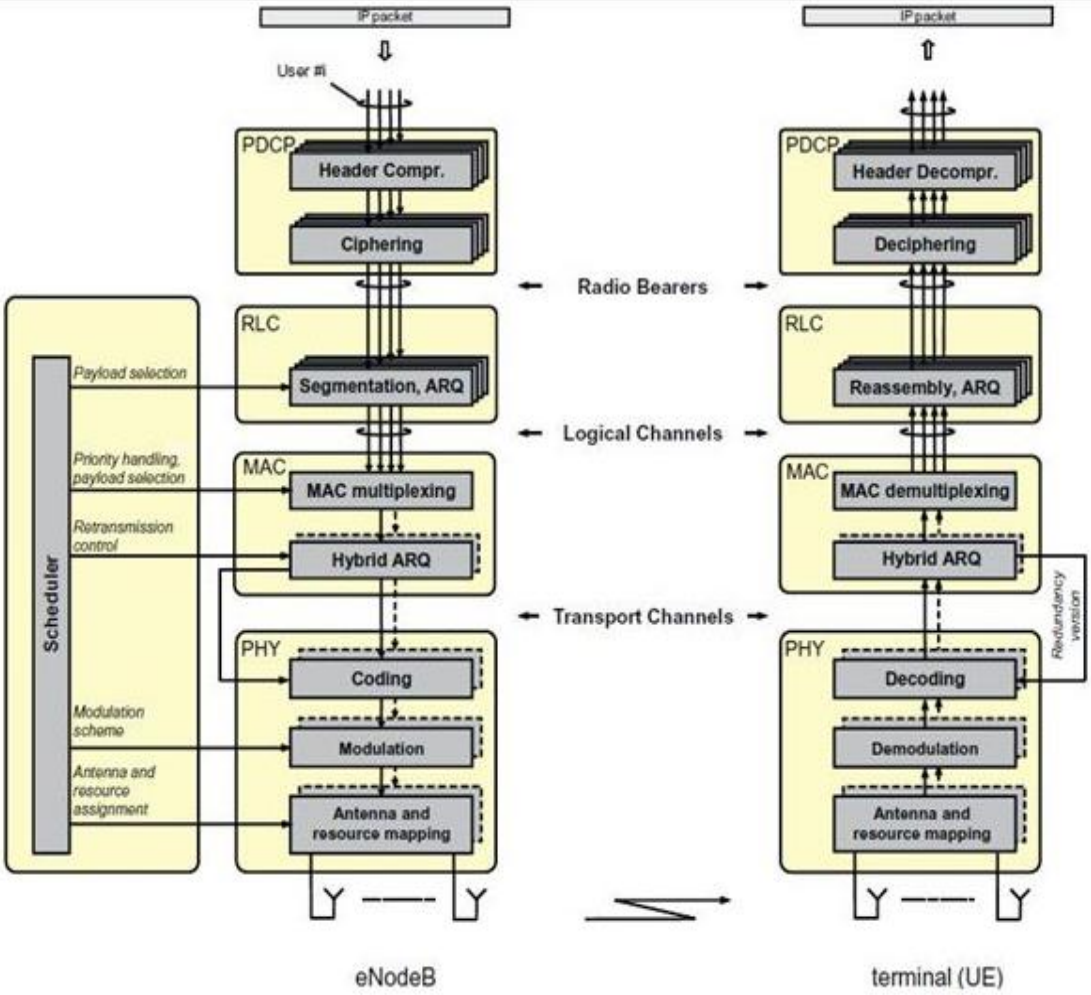


Figure 4-0-4 Channels on LTE

From above we notice that DLSCH uses 16 QAM or 64 QAM according to signal strength, however in weak signal or in CCPCH we use QPSK as it considers the most ROBUST Modulation scheme.

MAC Scheduler module: It supposed to dynamically decide modulation scheme should we use; The MAC scheduler gets information of the instantaneous Radio channel quality from users through channel state information message in which



it sends the CQI (channel quality report) based on channel quality it decides which modulation scheme is to be used.

Notice that: Higher order of modulation increases data rate, bandwidth, but also increasing probability of error and decrease power efficiency. So, we use lower order of modulation like (QPSK) to send the traffic to the user with poor signal strength, and for this reason we use lower order of modulation for users near cell edge, and near e-NODB we can use high order of modulation like (16-QAM,64-QAM).

In LTE this problem will be enhanced by increasing symbol duration then we can send large number of bits to cell edge user and this why data rate increased in LTE.

Modulation considers as the first step in OFDM as shown in OFDM block diagram below.

OFDM is the modulation scheme for the DL.

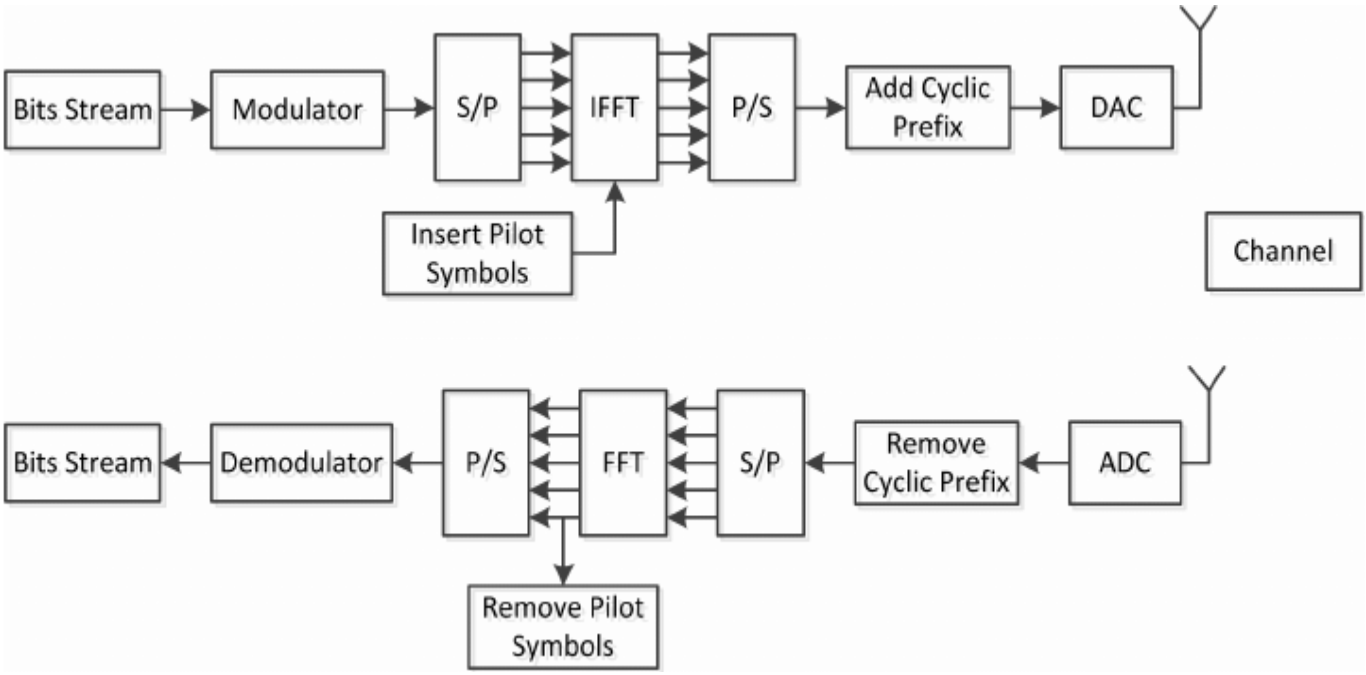


Figure 4-6 OFDM Block diagram

Modulation on OFDM consider as symbol mapping stage: On this stage we decide the number of bits used to map the symbol, depends on this equation: $M = log_2 N$

Table below show types of Modulation and bits/symbol on each type.

MODULATION	BITS PER SYMBOL	SYMBOL RATE
BPSK	1	1 x bit rate
QPSK	2	1/2 bit rate
8PSK	3	1/3 bit rate
16QAM	4	1/4 bit rate
32QAM	5	1/5 bit rate
64QAM	6	1/6 bit rate

Figure 4-7 Types of Modulation and Number of bits per symbol

Modulation techniques block diagram and simulation on MATLAB with constellation diagram, both frequency domain and time domain will be shown later:
Modulation is often followed by pulse shaping, and demodulation is often preceded by a filtering or an integrate and dump operation.
Notes for the MATLAB examples:

- Filtering Expression in Mat-Lab codes below means the way to avoid ISI.
- Verification for ideal case of modulations
- At ideal case we use raised cos and root raised cos with roll of factor to avoiding ISI.
- Using Noisy channel (default: AWGN) to know which type of modulation we should using.
- Testing all modulation technique by Simulink in practical noisy channel (AWGN+ SISO fading channels)

BER in different type of modulations effected with:

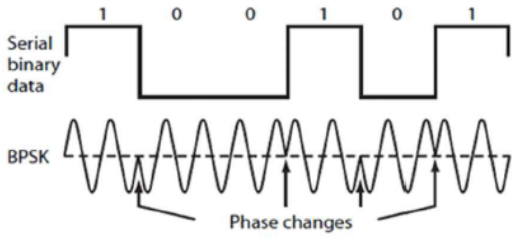
- channel coding before modulation
- Type of channel coding convolutional code or block code
- Hard or soft decision

4.1 BPSK

- **Binary phase shift keying (BPSK)**

$$s_i(t) = \sqrt{\frac{2E_b}{T_b}}\cos(\omega_c t + (i-1)\pi) \qquad 0 \le t \le T_b$$

$$i = 1,2$$



Therefore,

$$s_1(t) = \sqrt{\frac{2E_b}{T_b}}\cos(\omega_c t) \qquad , s_2(t) = \sqrt{\frac{2E_b}{T_b}}\cos(\omega_c t + \pi) = -\sqrt{\frac{2E_b}{T_b}}\cos(\omega_c t) = -s_1(t)$$

- Signal Space (SS) representation

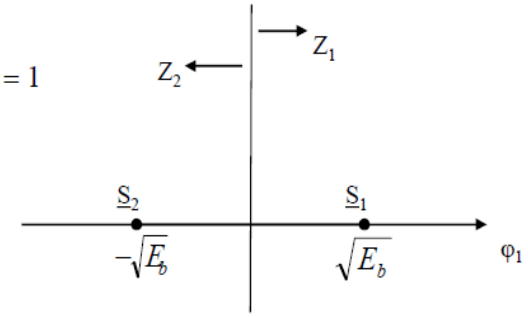
By inspection ,

$$\phi_1(t) = \sqrt{\frac{2}{T_b}}\cos(\omega_c t) \qquad N = 1$$

$$\therefore S_{11} = \sqrt{E_b}, \qquad S_{21} = -\sqrt{E_b}$$

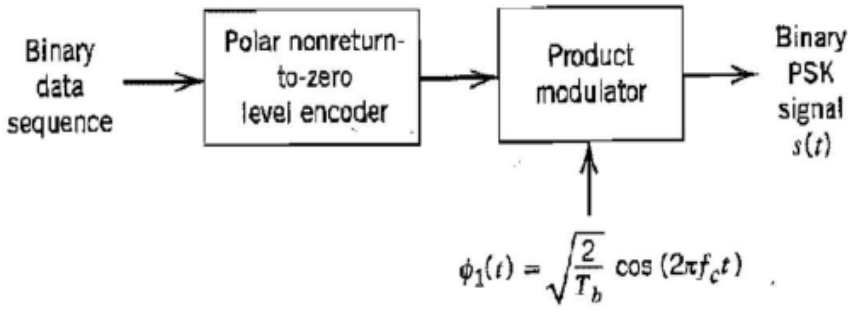
$$\underline{S}_1 = [\sqrt{E_b}], \qquad \underline{S}_2 = [-\sqrt{E_b}]$$

$$\rho_{12} = -1 \qquad d_{12} = \|\underline{S}_1 - \underline{S}_2\| = 2\sqrt{E_b}$$



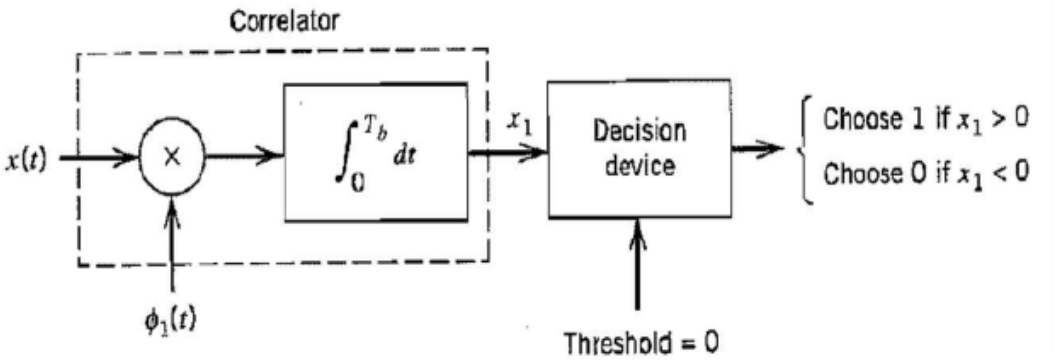
Block diagram:

- T_x



(a)

- R_x



Verification for ideal case:

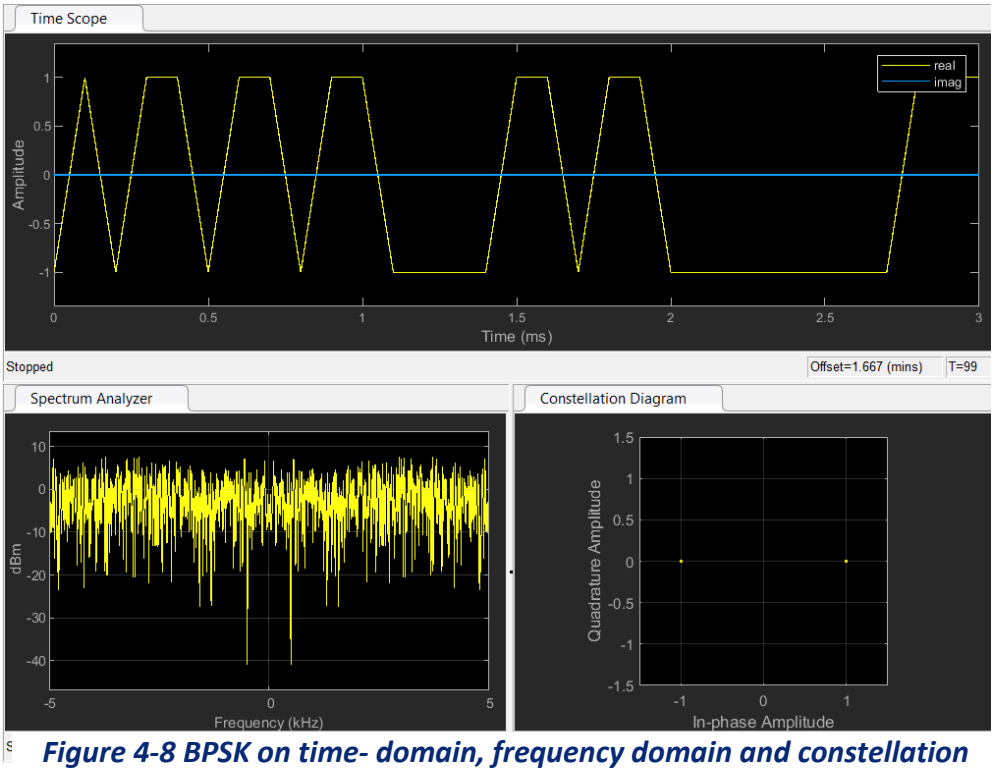


Figure 4-8 BPSK on time- domain, frequency domain and constellation diagram

Verification for ideal case with avoid of ISI:

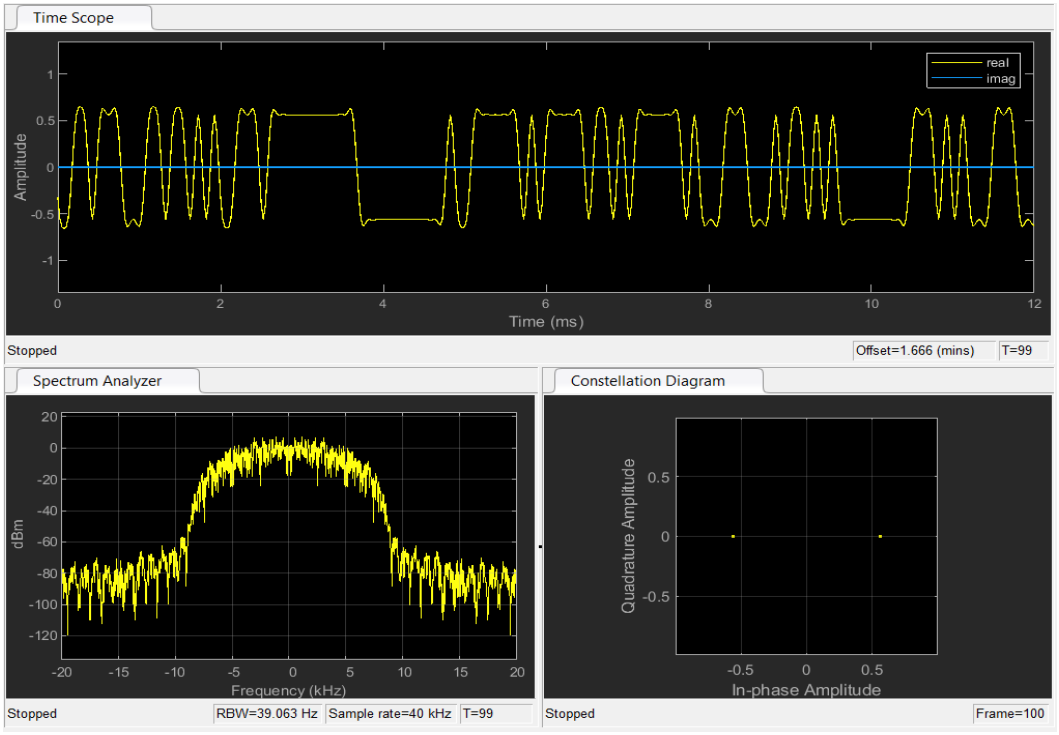
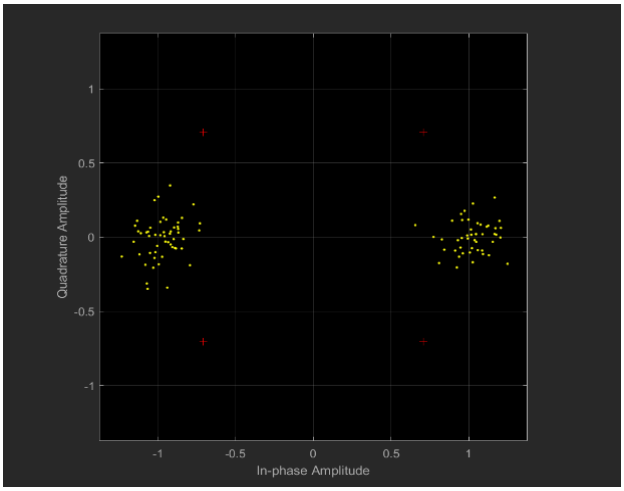


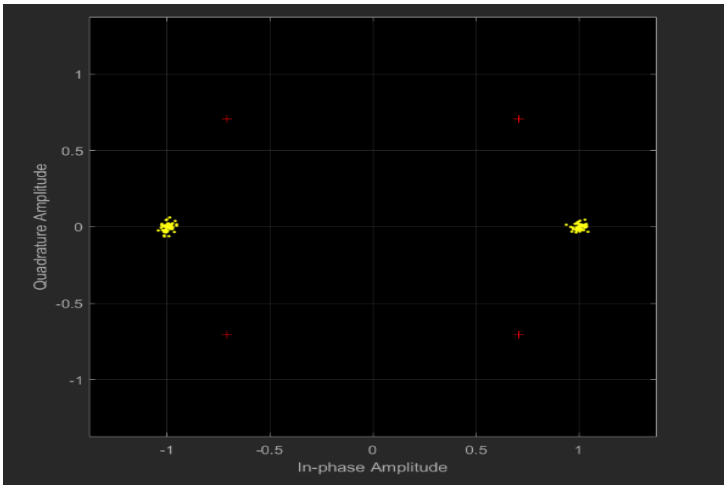
Figure 4-9 Use raised cos and root raised cos with roll of factor.

Test in practical only AWGN:

This Circuit Block is used in Simulink with Some Changes in parameters and Channels at (AWGN):



At SNR=15db



At SNR=30db

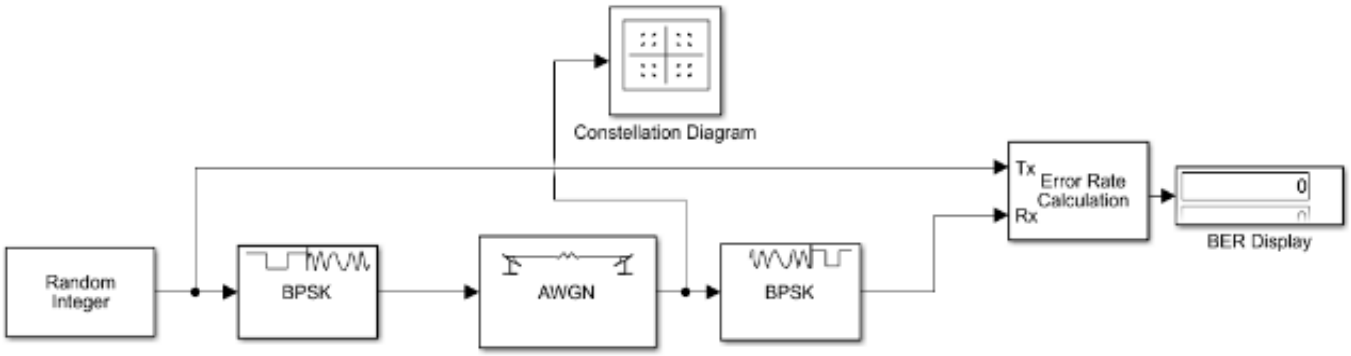


Figure 4-10 BPSK Block diagram

4.2 Quadrature Phase Shift Keying [QPSK]

is a variation of BPSK, and it is also a Double Side Band Suppressed Carrier DSBSC modulation scheme, which sends two bits of digital information at a time. Instead of the conversion of digital bits into a series of digital stream, it converts them into bit pairs. This decreases the data bit rate to half, which allows space for the other users.

4.2.1 QPSK Modulator

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

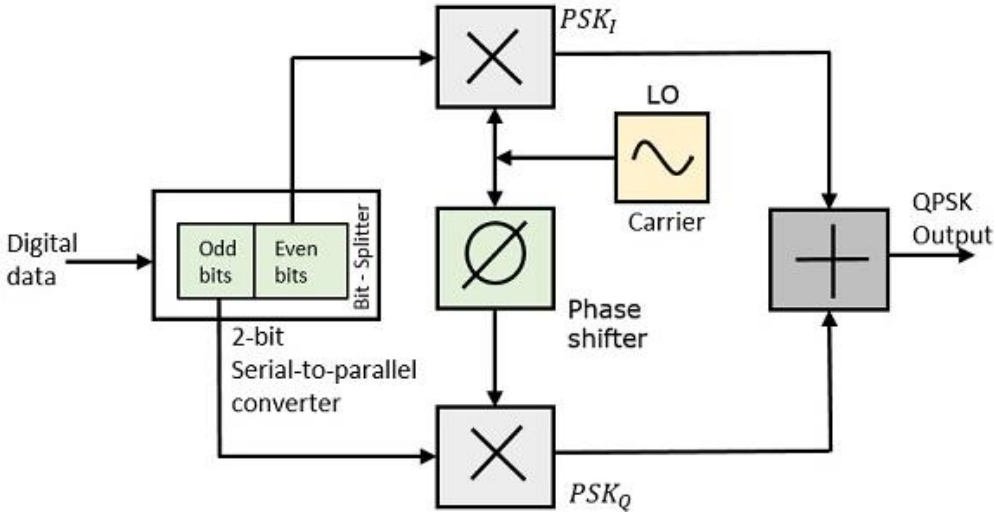


Figure 4-11 QPSK Modulator

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

The QPSK waveform for two-bits input is as follows, which shows the modulated result for different instances of binary inputs.

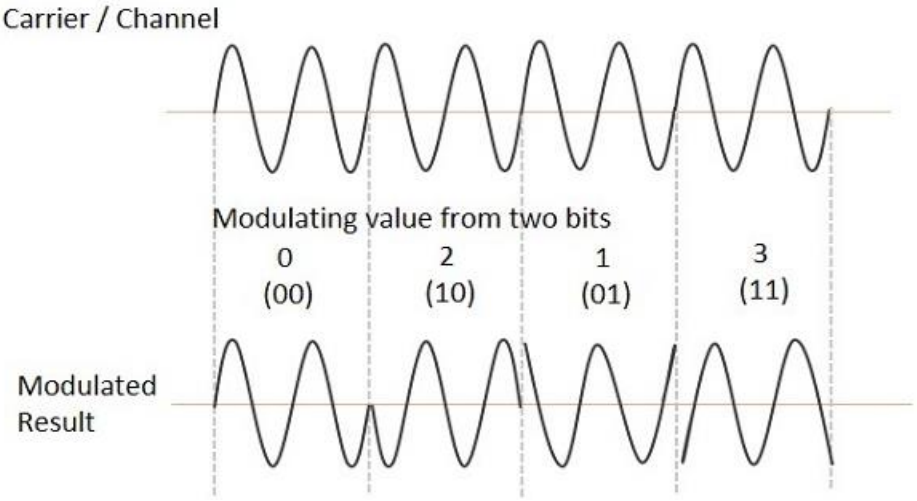


Figure 4-12 QPSK waveform

4.2.2 QPSK Demodulator

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

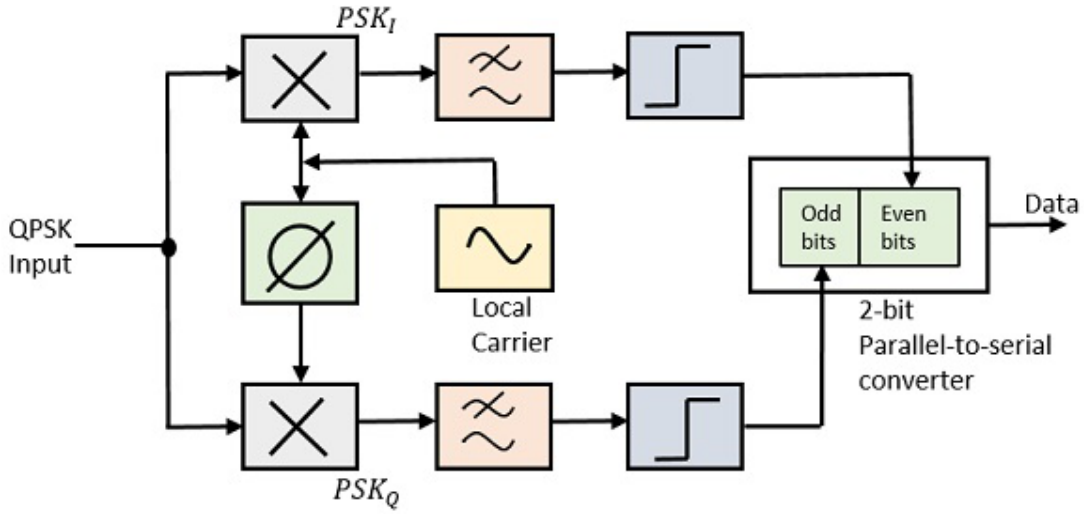
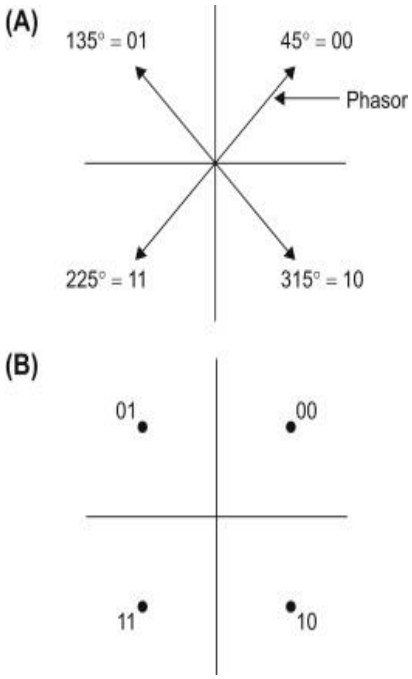
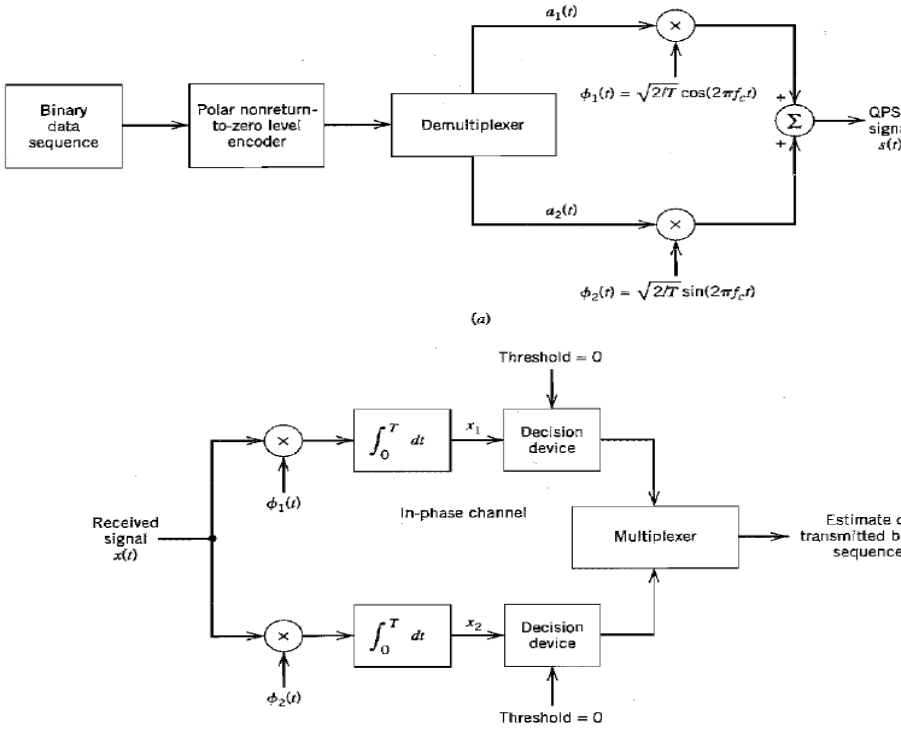


Figure 4-13 QPSK Demodulator

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





Simulation on MATLAB for QPSK

Figure 4-14 QPSK Modulator and demodulator

Verification for ideal case:

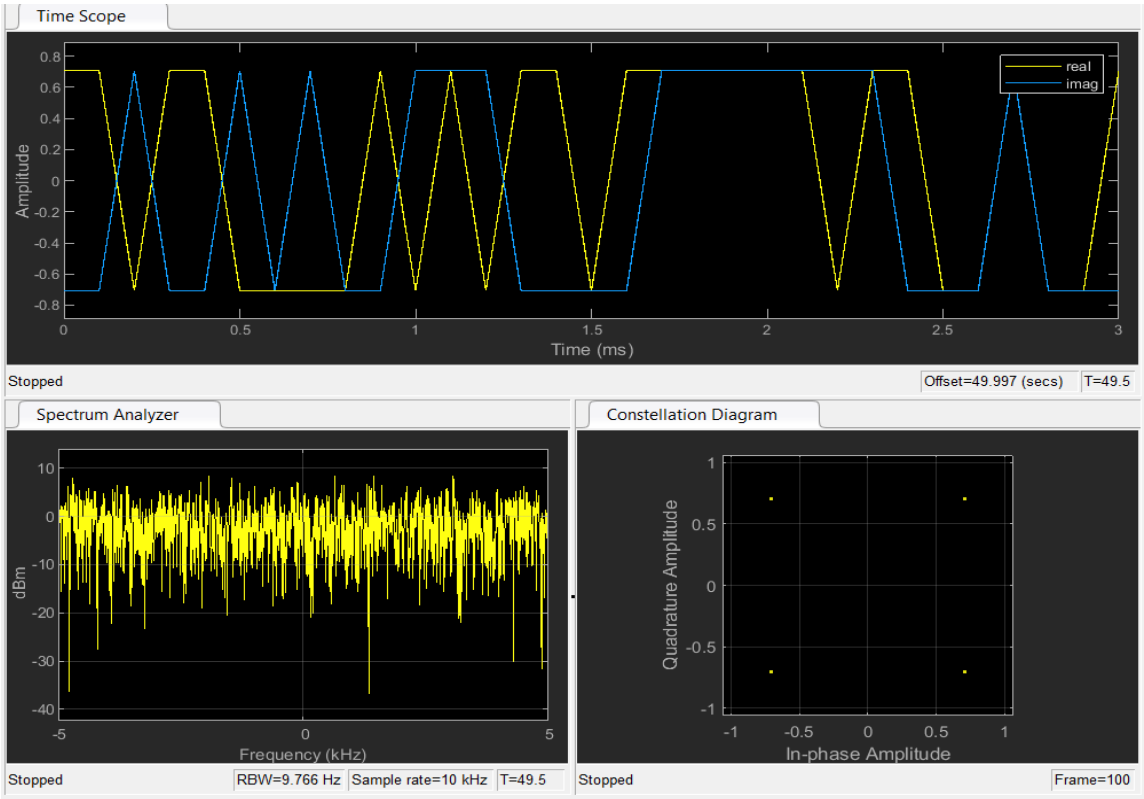
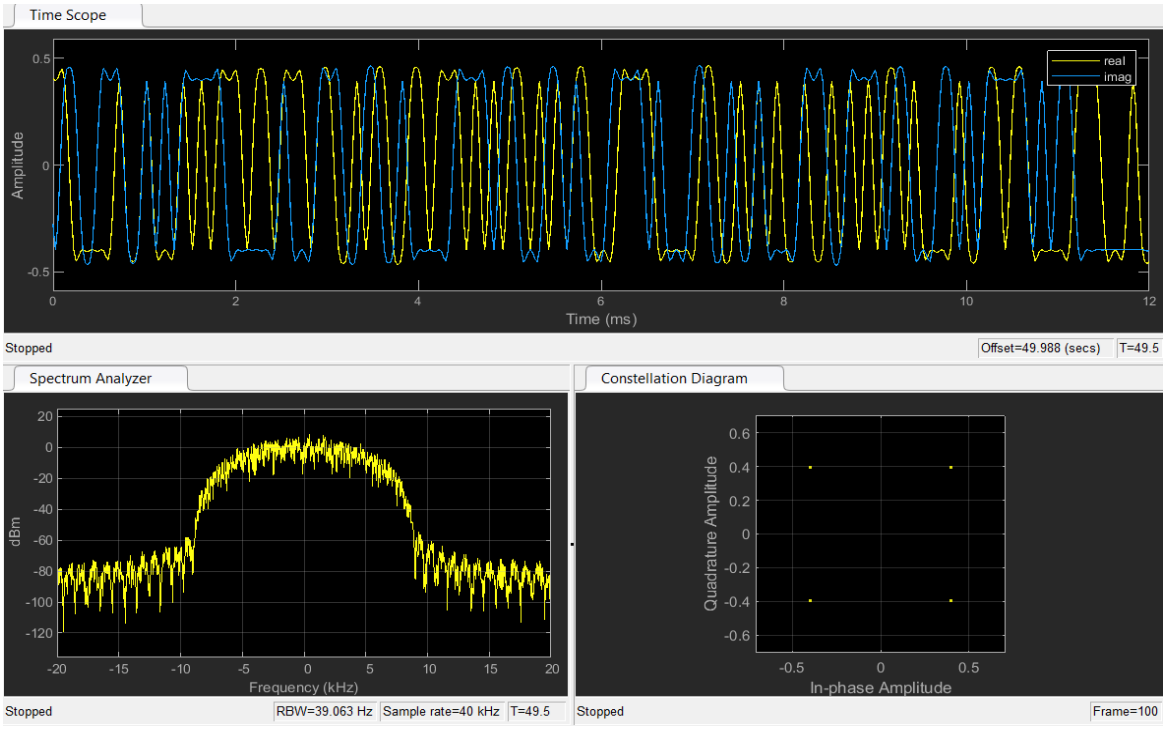


Figure 4-15 QPSK on time- domain, frequency domain and constellation diagram.

Verification for ideal case with avoid of ISI:



Test at Simulink with Changing in Circuit-Block with (15,30db):

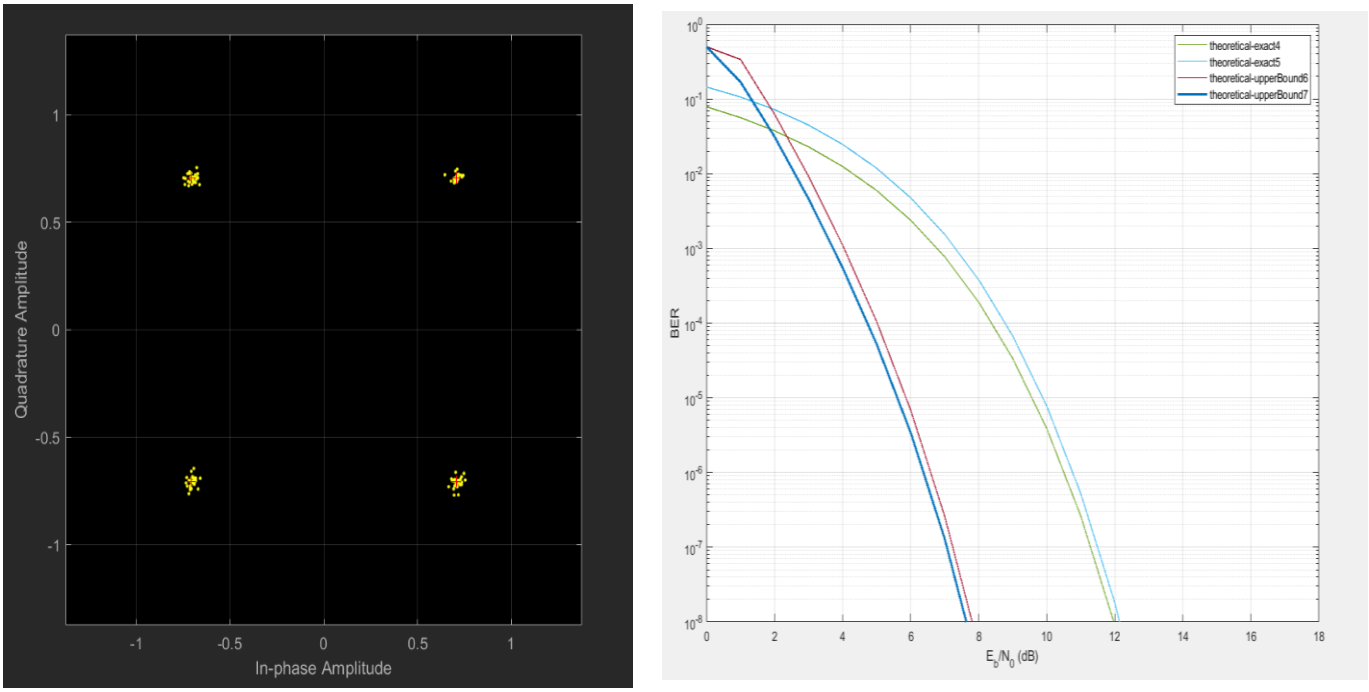


Figure 4-17 Test the "BER" with Noisy Channel "AWGN" at different cases as explained above:

Green: without pre-coding.
Sky-blue: with pre-coding. "Differential".
Red: with pre-coding and channel coding "Convolutional Coding" Hard-decision.
Blue: with pre-coding and channel coding "Convolutional Coding" Soft-decision.

4.3 16QAM

Can be implement by transmitting 4 bits per time interval or symbol causing the data rate to quadruple in the same bandwidth. The circuit for producing 16-QAM is shown below. The 4-bit shift register produces two groups of I and Q bits. These are converted into four levels by a two- to four-level converter that works like a digital-to-analog converter (DAC). The resulting constellation diagram for 16-QAM is shown. Other common versions are 64-QAM and 256-QAM, which transmit 6 and 8 bits per symbol, respectively, in the same bandwidth. A 1024QAM is also an option in some systems.

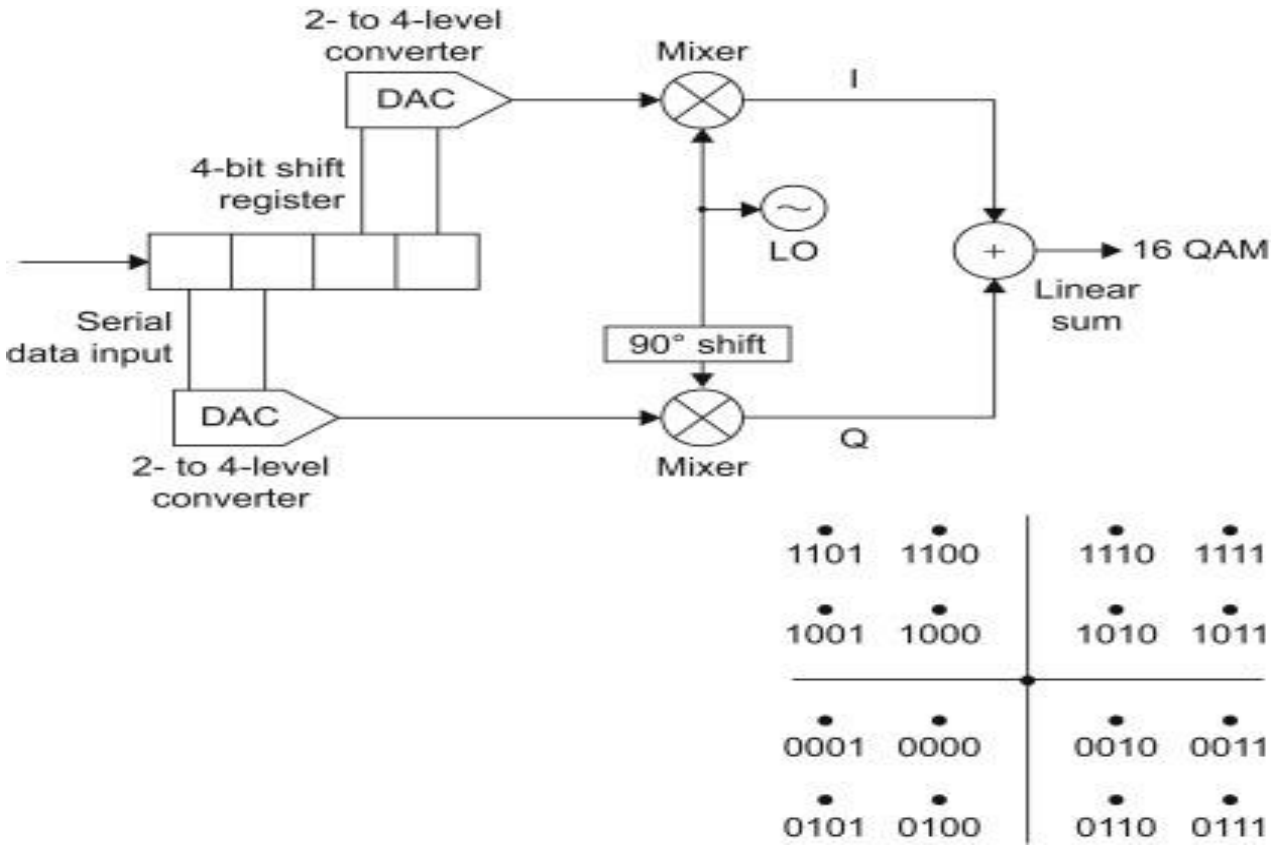


Figure 4-18 16QAM Modulator and constellation diagram

R_x

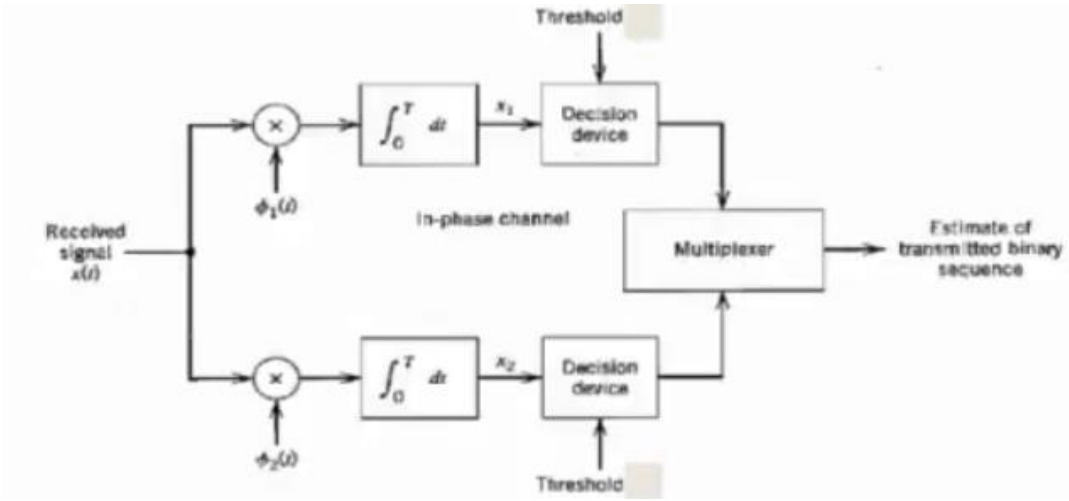
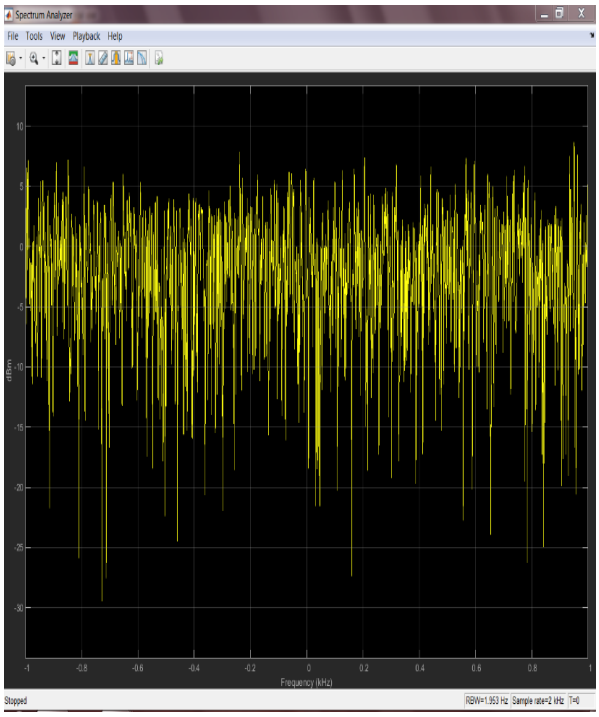


Figure 4-19 16QAM Demodulator

Verification for ideal case:



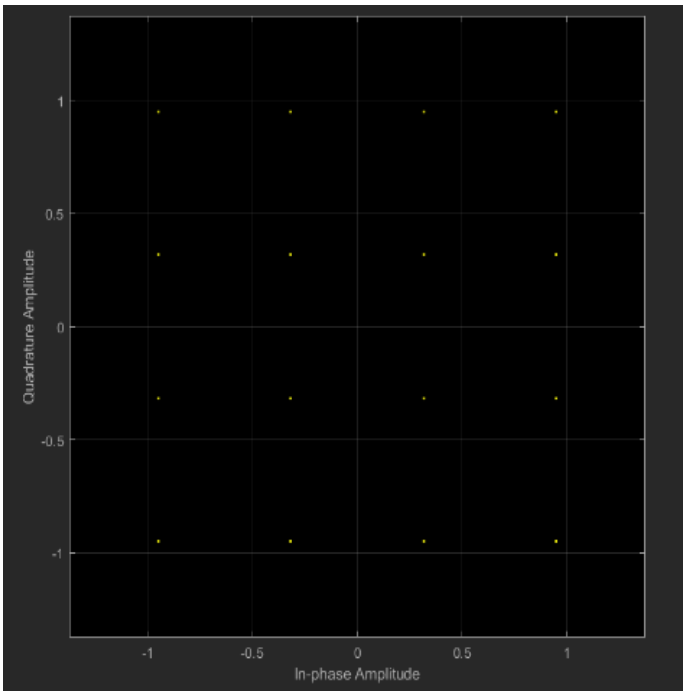


Figure 0-20 16QAM on time- domain, frequency domain and constellation diagram.

Verification for ideal case with avoid of ISI:

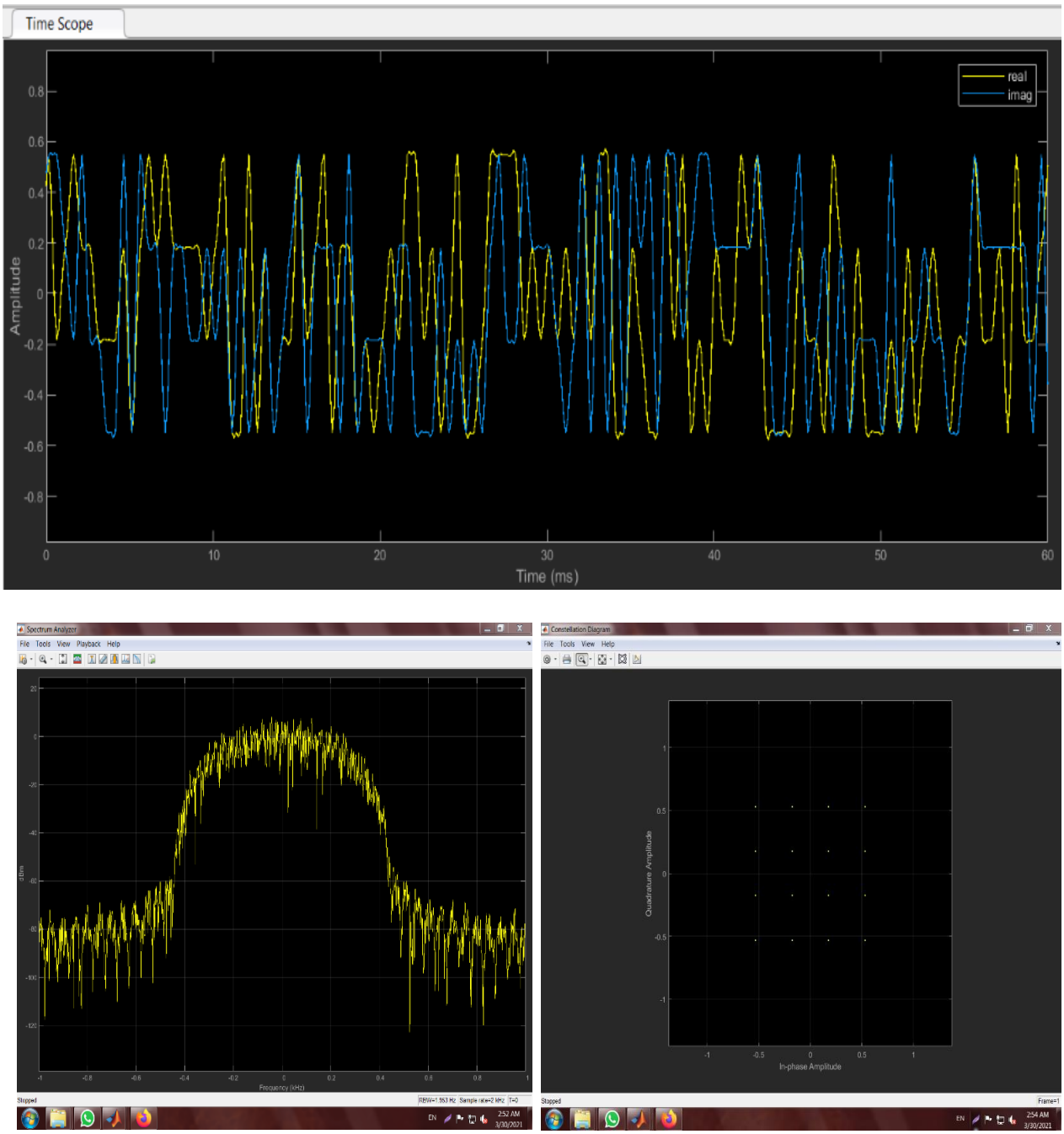
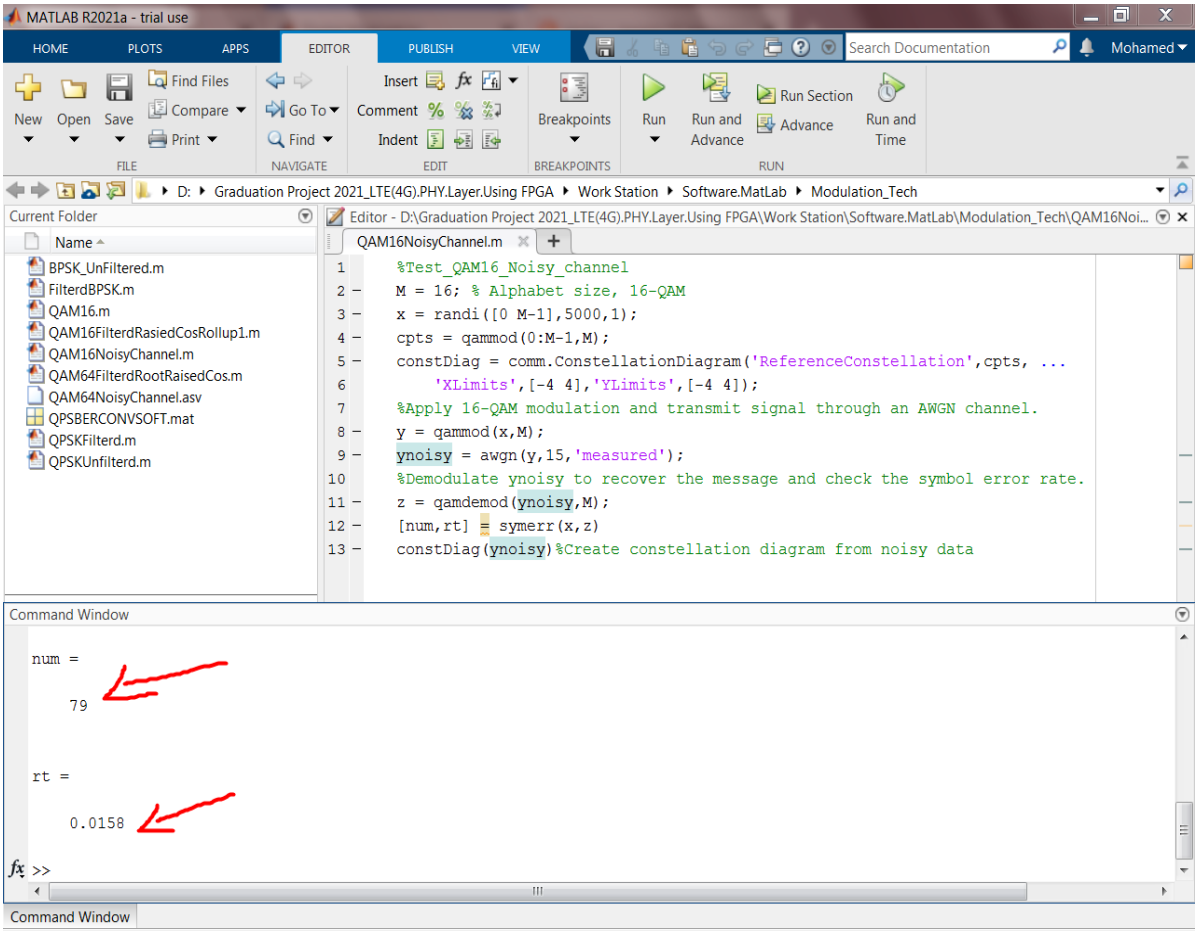


Figure 4-21 16QAM Use raised cos and root raised cos with roll of factor.



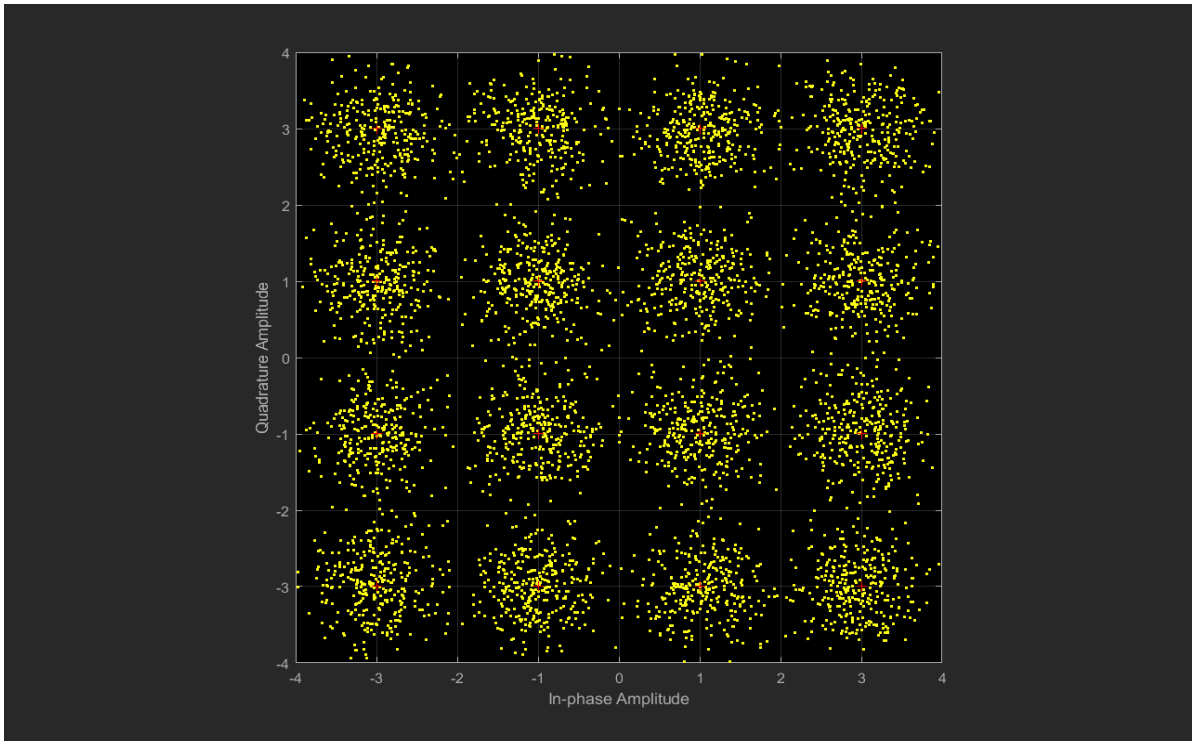
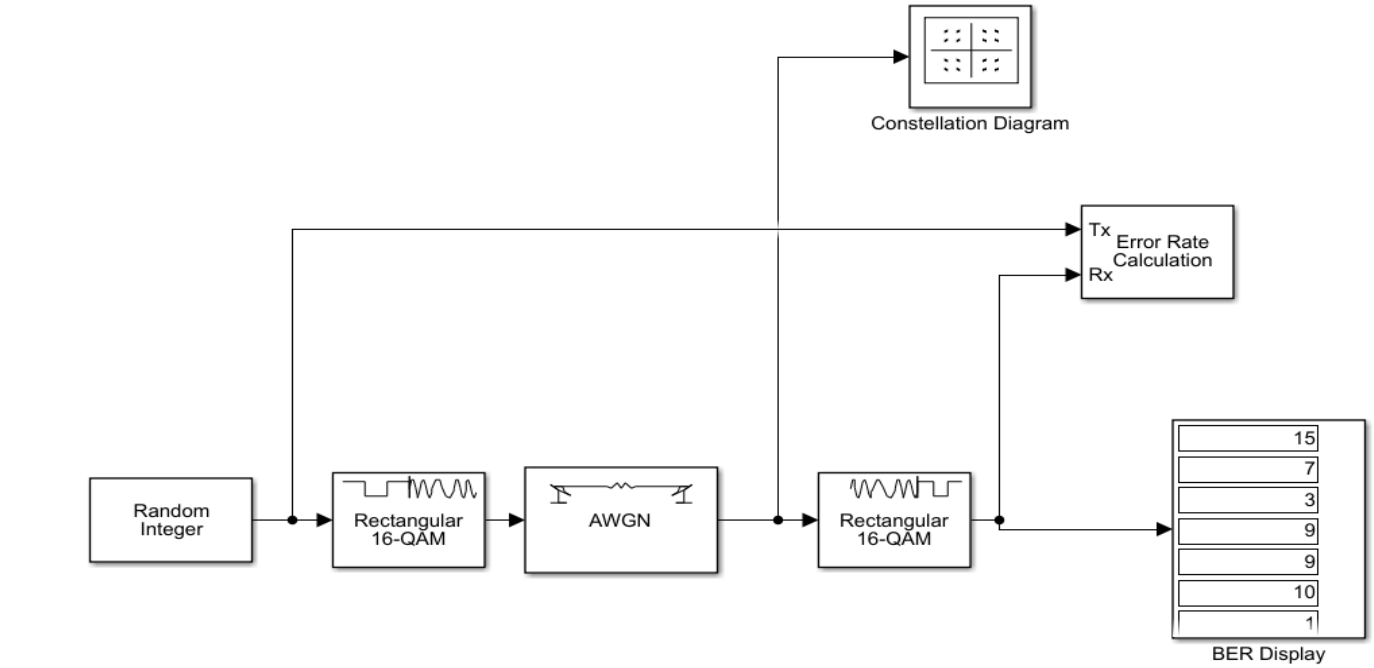
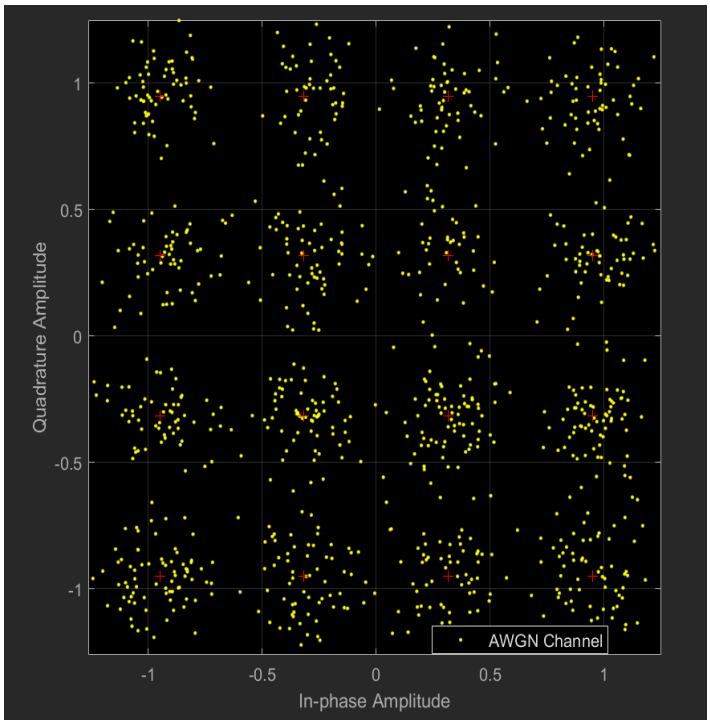


Figure 4-22 Test QAM16 at Noisy-Channel with Measured SNR and Finding the value of Signal Power with constellation diagram.





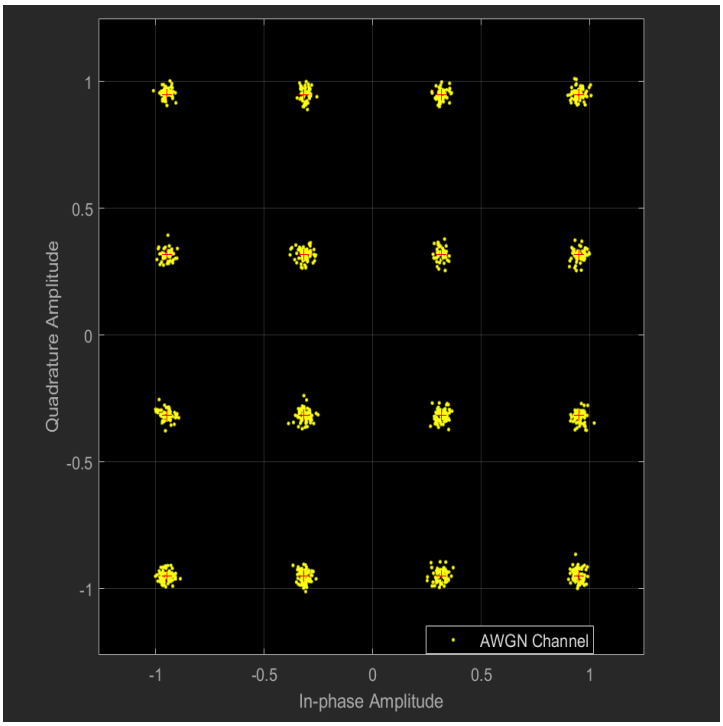


Figure 4-23 Test at Simulink with Changing in Circuit-Block with AWGN (15,30db)

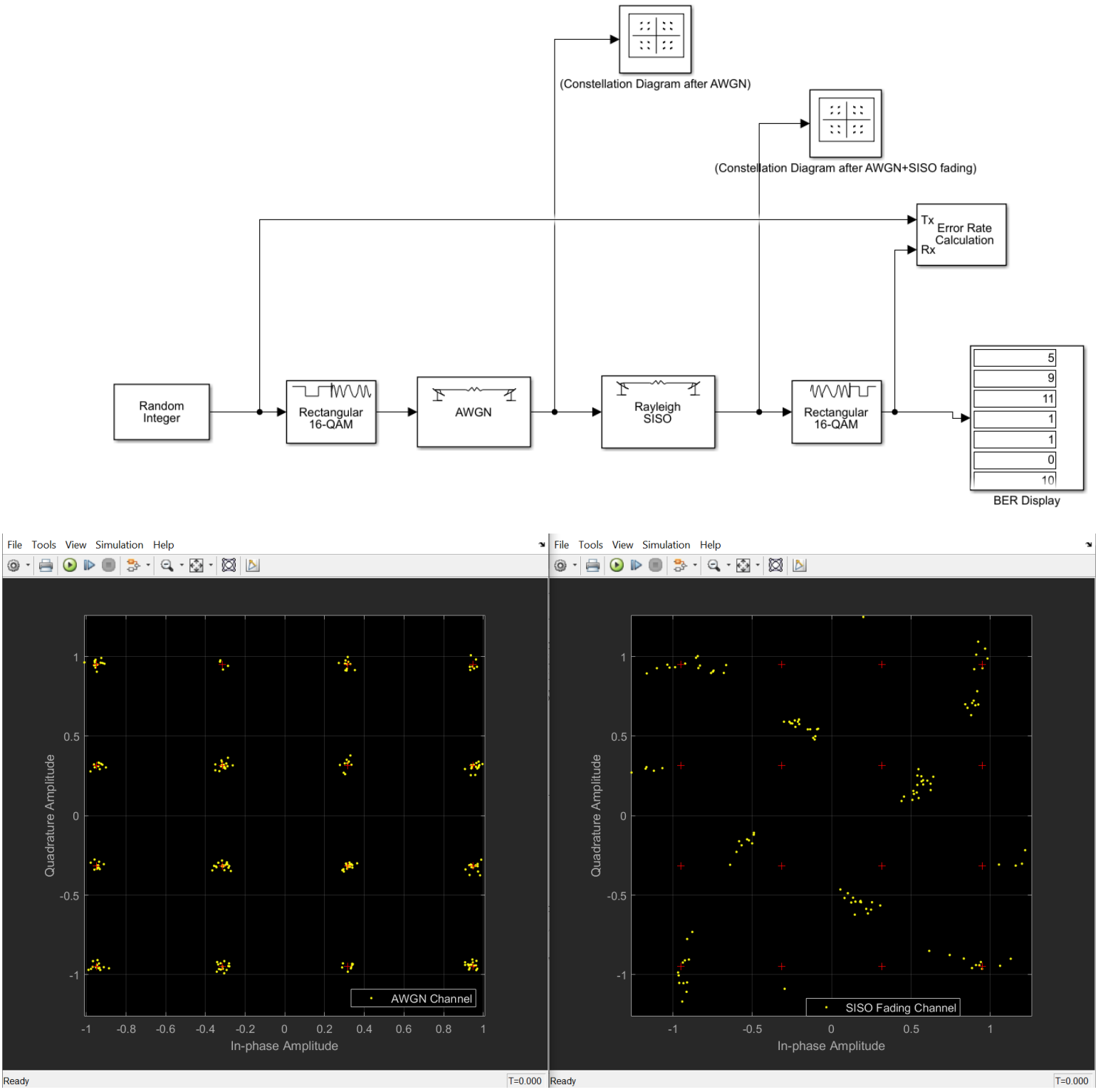


Figure 4-24 Testing QAM16 by Simulink in practical noisy channel (AWGN (30dB) and SISO fading channels)

4.4 64QAM

Verification for ideal case:

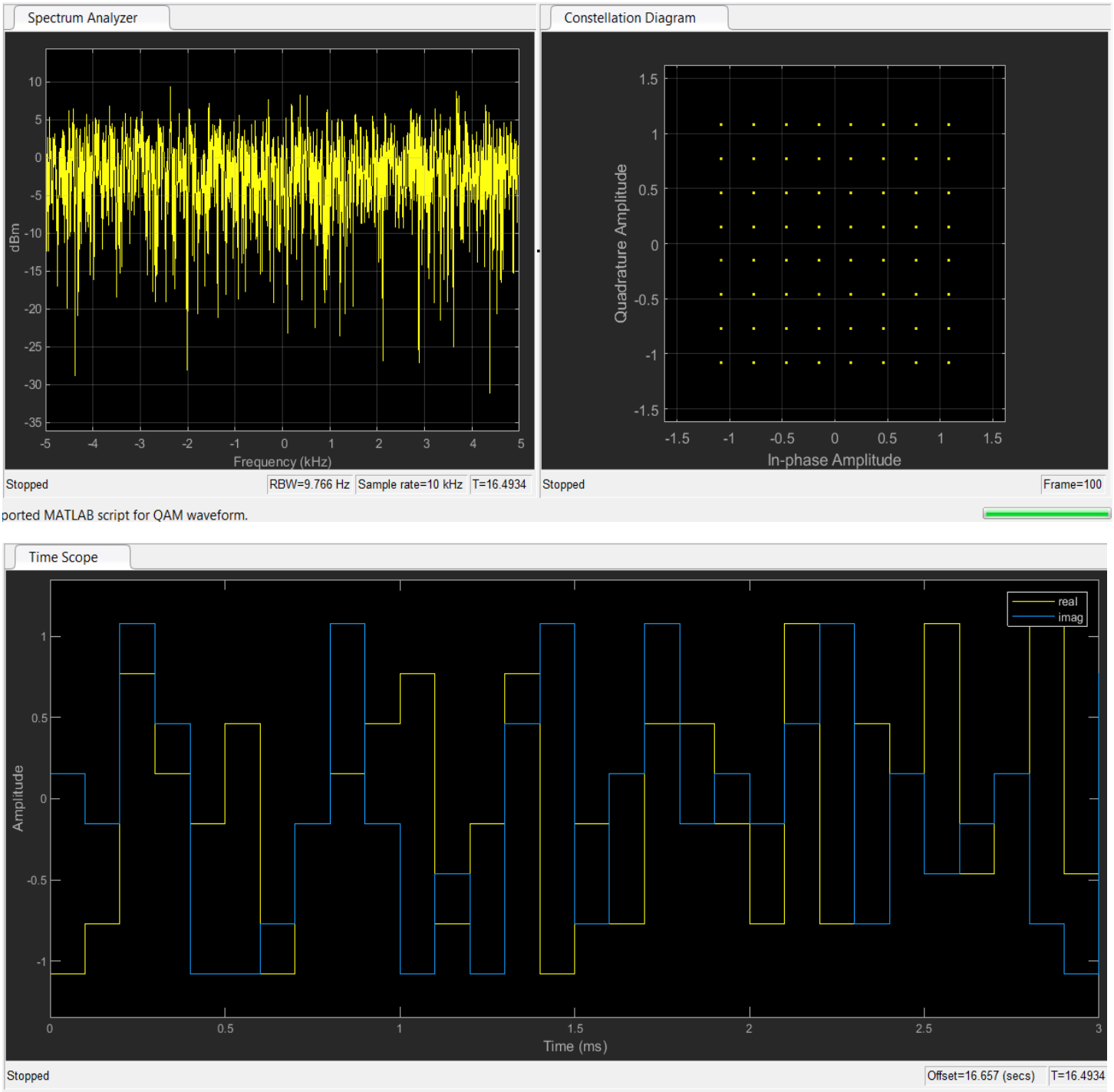


Figure 4-25 64QAM on time- domain, frequency domain and constellation diagram.

Verification for ideal case with avoid of ISI with raised cos:

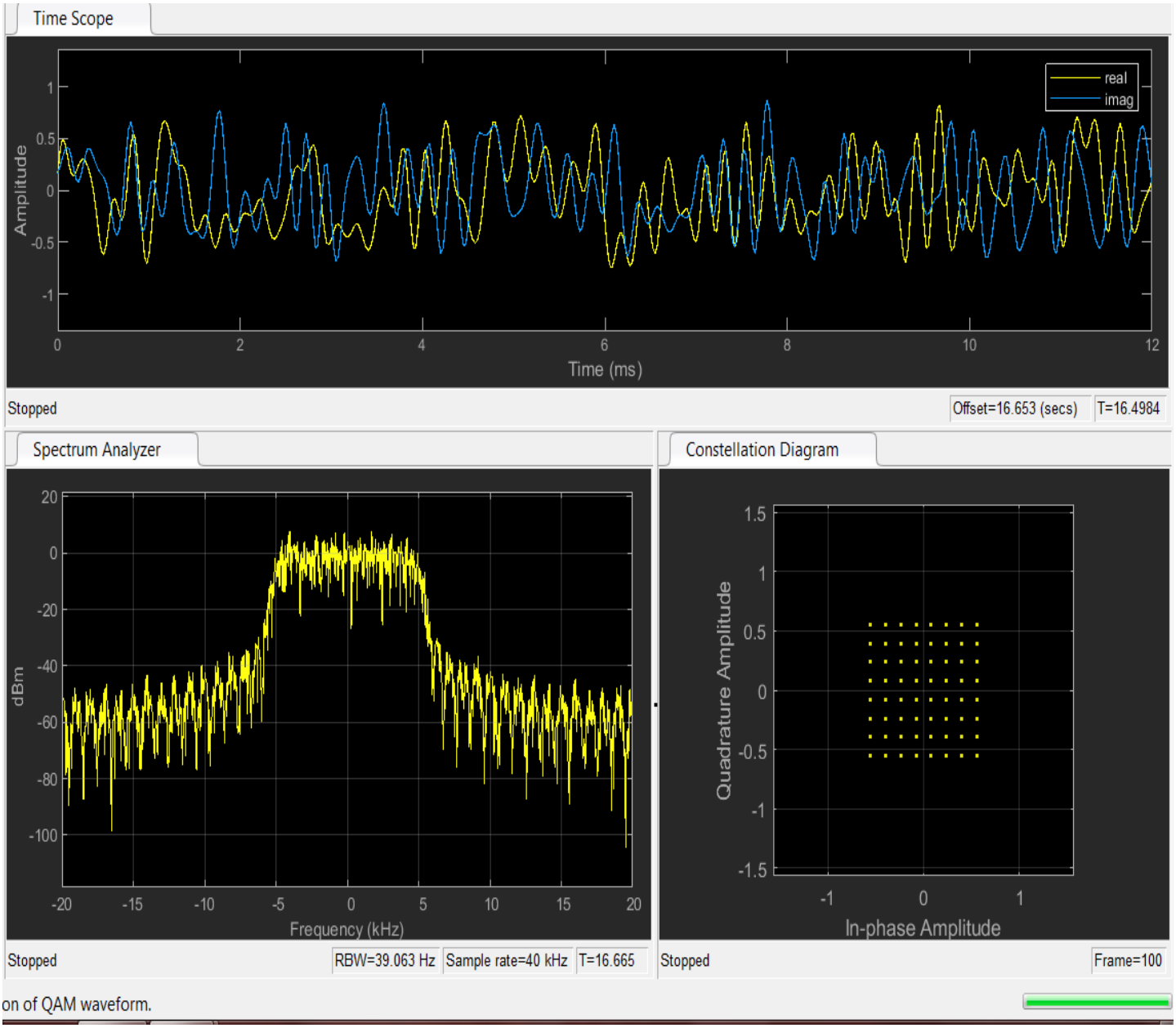


Figure 4-26 64QAM Use raised cos and root raised cos with roll of factor.

Verification for ideal case with avoid of ISI with root raised cos:

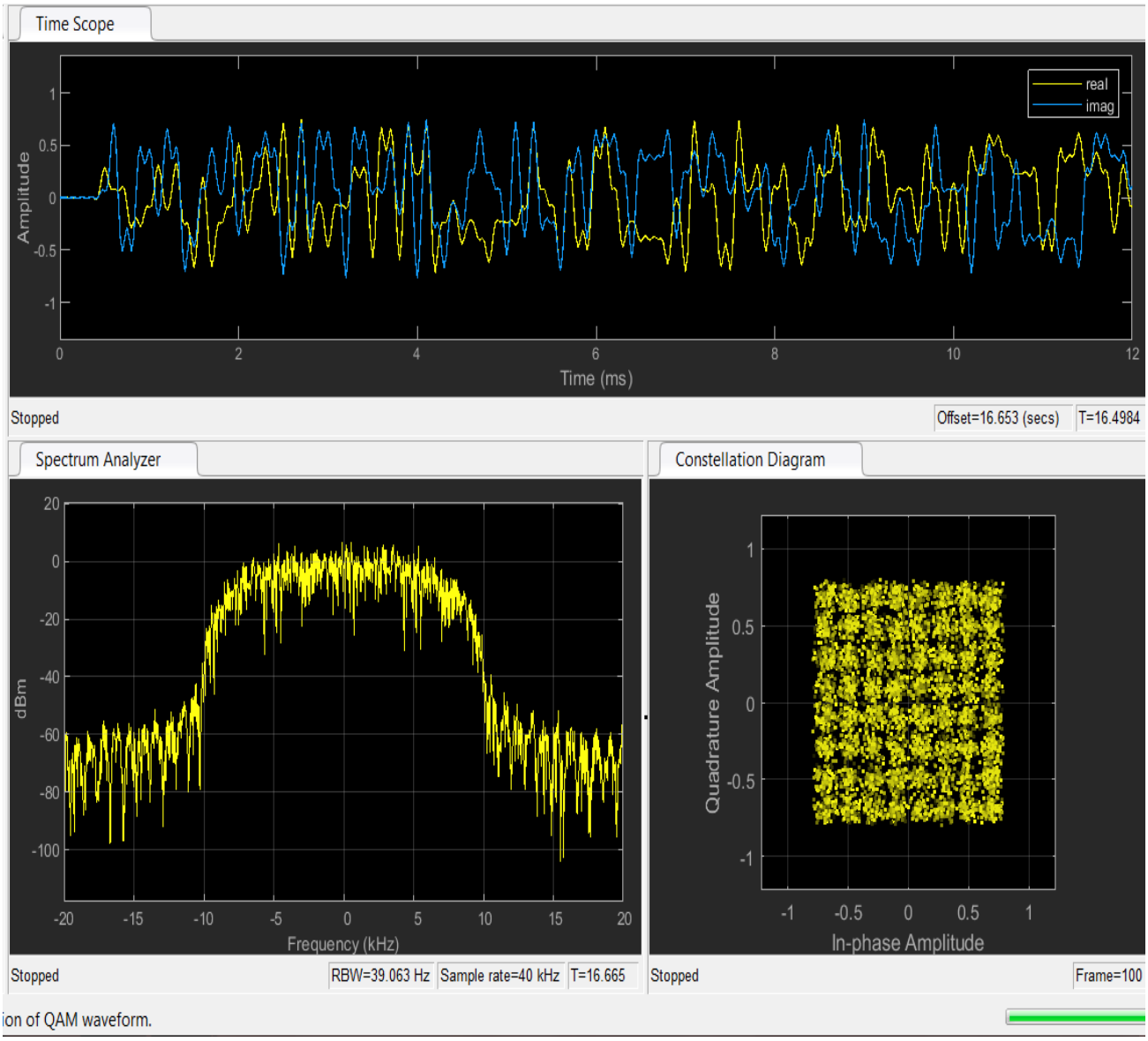


Figure 4-27 64QAM with root raised cos.

Test QAM64 at Noisy-Channel with Measured SNR (15) and finding the value of Signal Power:

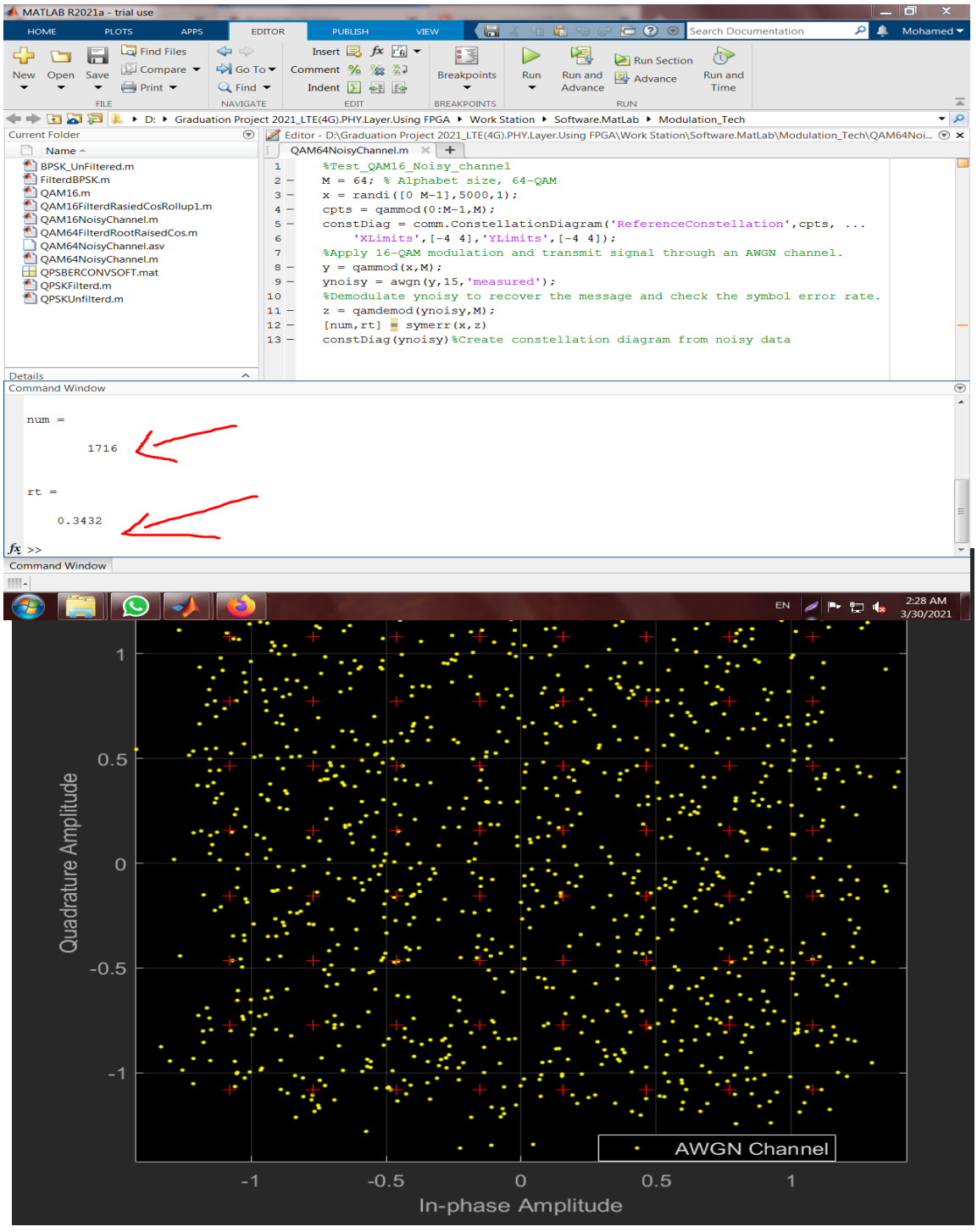


Figure 4-28 QAM64 at Noisy-Channel with SNR (15)

Test QAM64 at Noisy-Channel with Measured SNR (20) and finding the value of Signal Power:

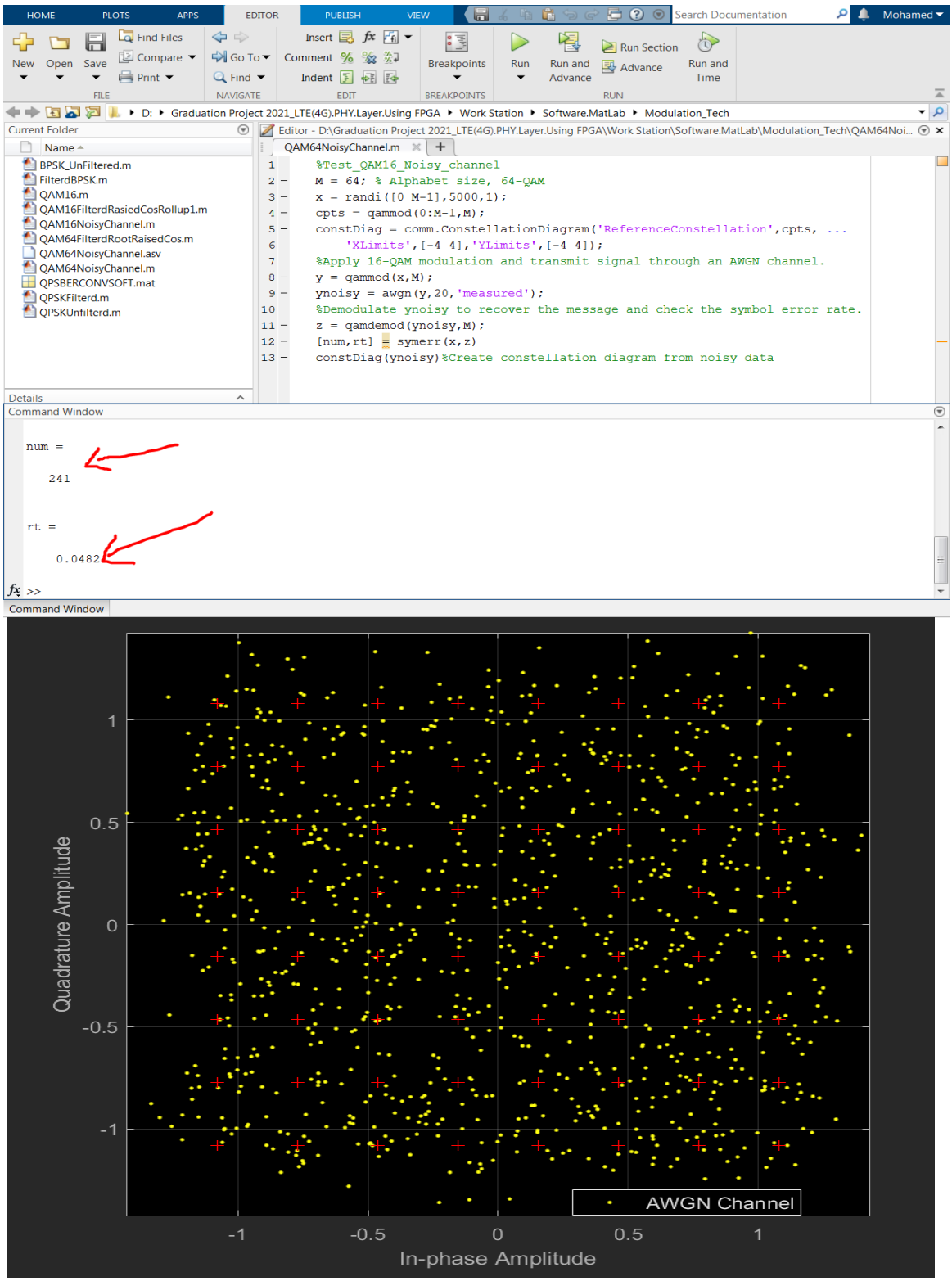


Figure 4-29 QAM64 at Noisy-Channel with SNR (20)

Test QAM64 at Noisy-Channel with Measured SNR (30) and finding the value of Signal Power:

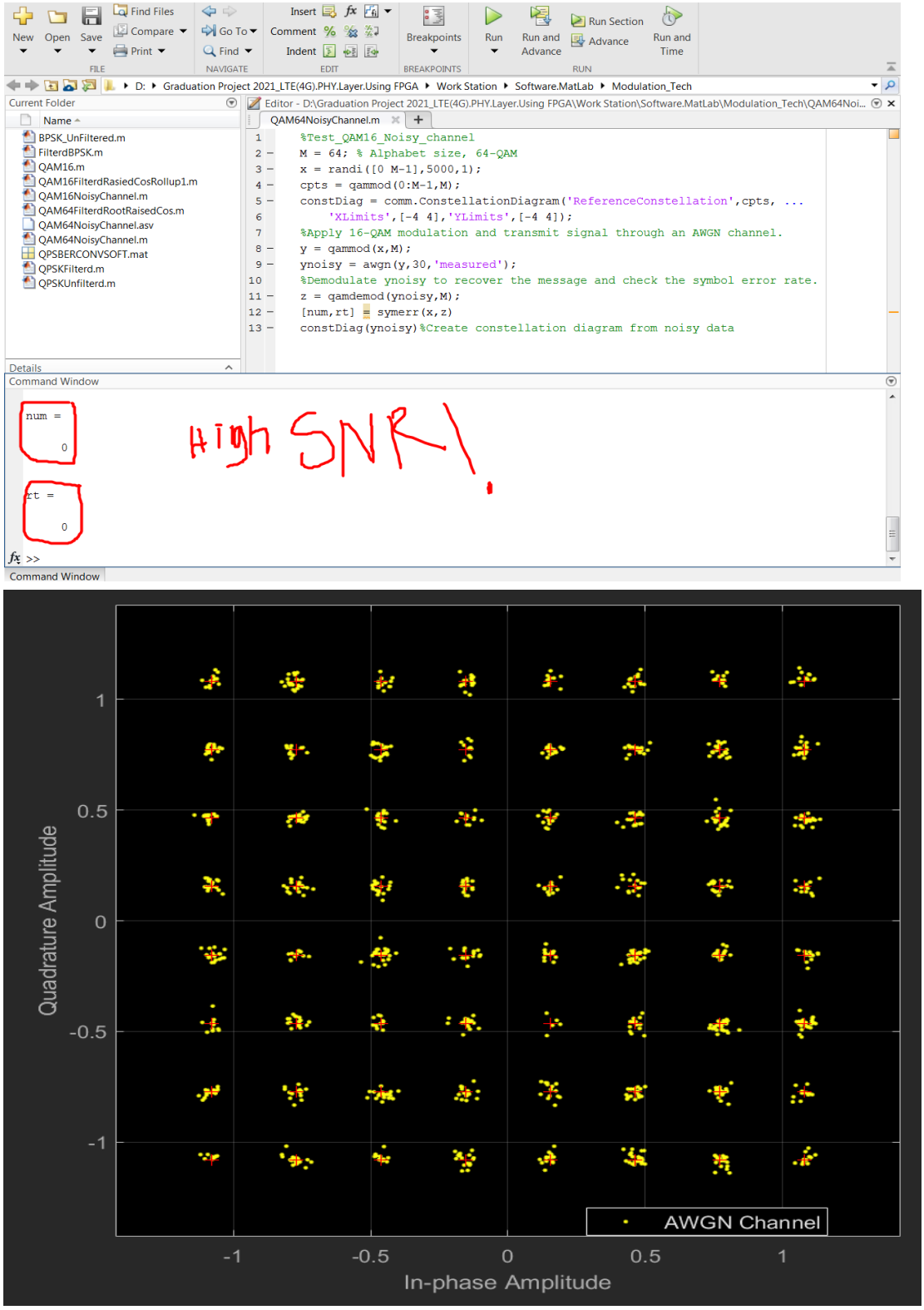
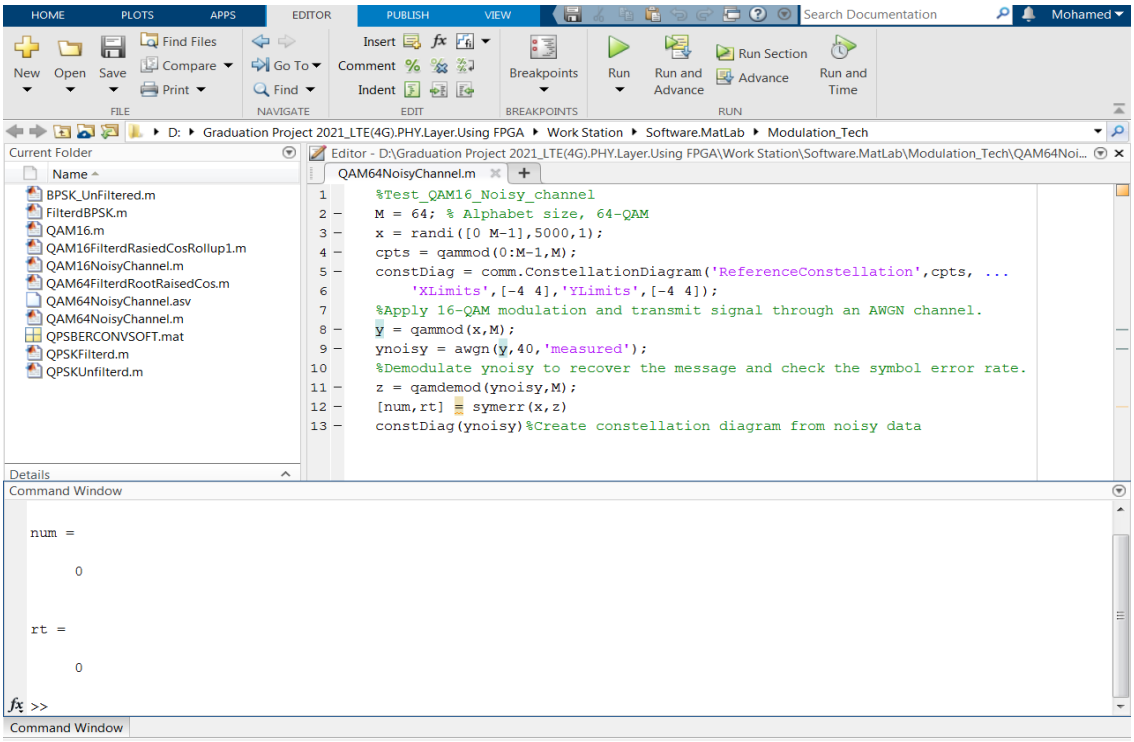


Figure 4-30 QAM64 at Noisy-Channel with SNR (30)

Test QAM64 at Noisy-Channel with Measured SNR (40) and finding the value of Signal Power:



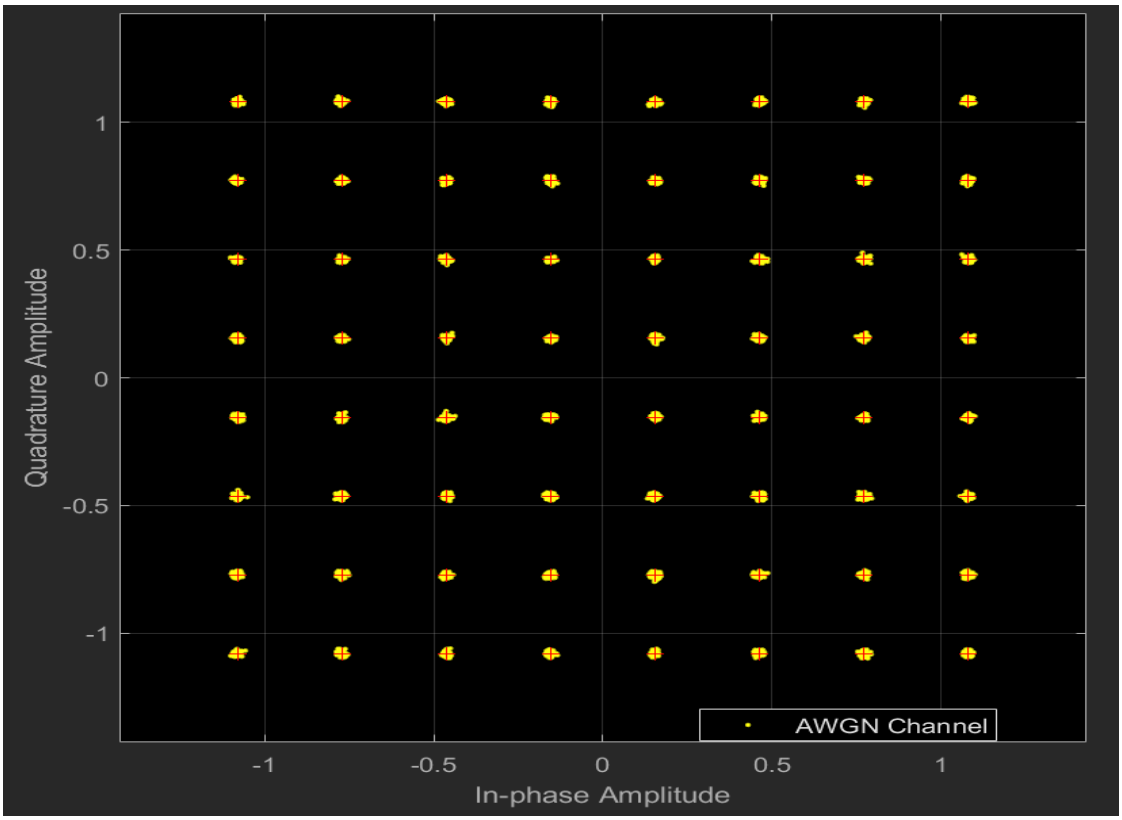


Figure 4-31 QAM64 at Noisy-Channel with SNR (40)

Test with Practical Noisy Channel (AWGN and SISO fading channel), using Simulink:

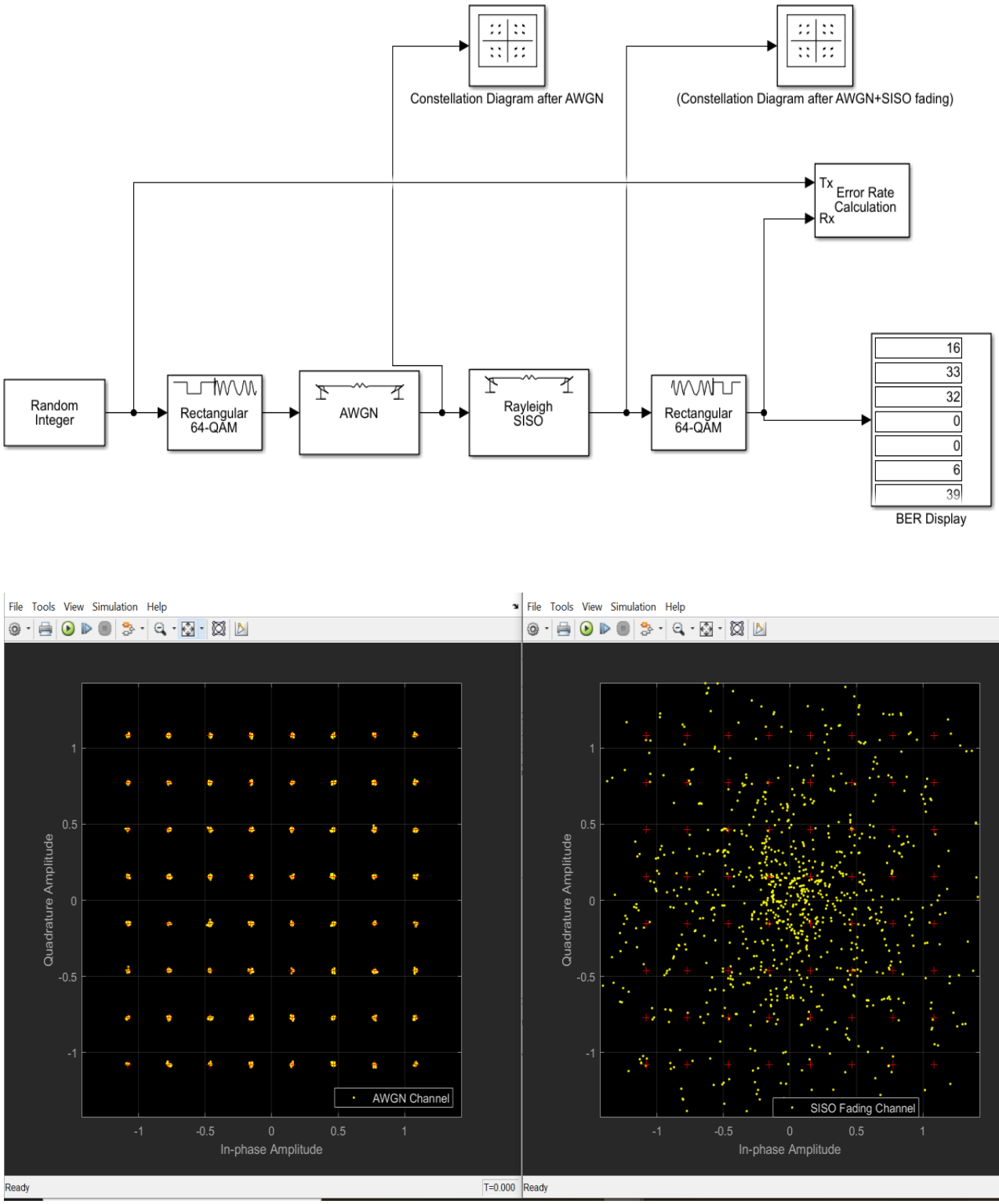


Figure 4-32 Test with Practical Noisy Channel (AWGN and SISO fading channel), using Simulink

4.5 Differential Phase Shift Keying

(DPSK below explained due to it had been used on CRC stage.)

Differential phase shift keying (DPSK) is a common type of phase modulation that conveys data by changing the phase of the carrier wave. In DPSK the phase of the modulated signal is shifted relative to the previous signal element. The signal phase follows the high or low state of the previous element. DPSK does not need a synchronous (coherent) carrier at the demodulator.

The input sequence of binary bits is modified such that the next bit depends upon the previous bit. Therefore, in the receiver, the previous received bits are used to detect the present bit. The following figure.1 shows the waveform of DPSK.

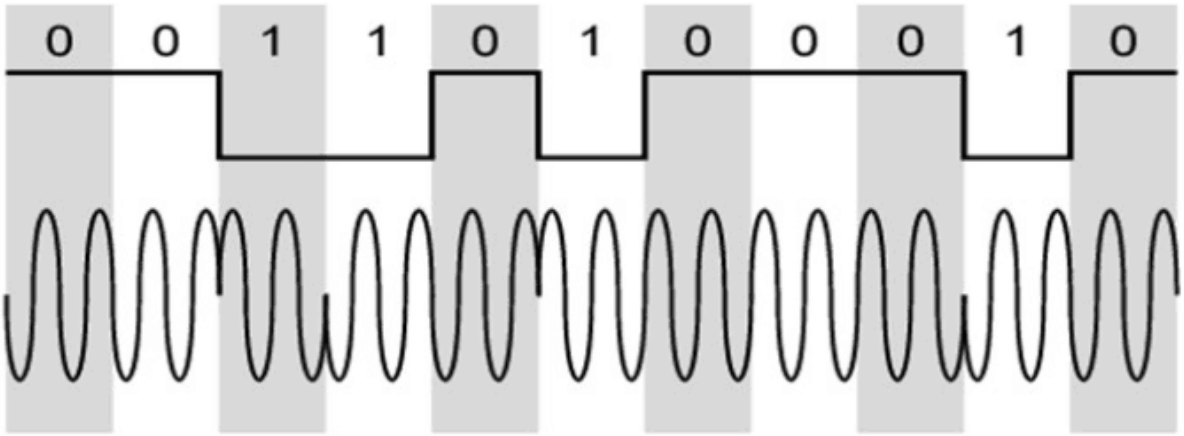


Figure 4-33 waveform of DPSK.

It cabs seen from the above figure that, when the data bit is Low i.e., '0', the phase of the signal is not reversed, and continued as it was. When the data is a High i.e., '1', the phase of the signal is reversed. If we observe the above waveform, we can say that the High state represents an M in the modulating signal and the Low state represents a W in the modulating signal.

Figure 4-34 DPSK Modulator.

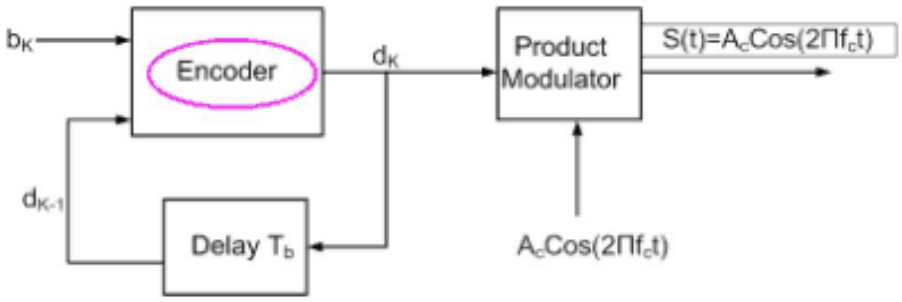


Figure 4-35 DPSK Modulator.

DPSK Modulator

DPSK is a technique of BPSK, in which there is no reference phase signal. In this case, the transmitted signal itself used as a reference signal. The following Fig.2 shows a DPSK Modulator.

The serial data input b(t) is applied to the input of the encoder. The output of the encoder is applied to one of the inputs of the product modulator. To the other input of this product modulator, a sinusoidal carrier of fixed amplitude and frequency is applied. The relationship between the binary sequence and its deferentially encoded version is shown in table.1 for an assumed data sequence 0 0 1 0 0 1 0 0 1 1 1. In this illustration it has been assumed that the encoding has been done in such a way that transition in the given binary sequence with respect to the previous encoded bit is represented by a symbol '0' and no transition by symbol '1'. It may be noted that an extra bit (symbol 1) has been arbitrarily added as an initial bit. This is essential to determine the encoded sequence. The phase of the generated DPSK signal has been shown in the third row of table.1.

DPSK Demodulator

In DPSK demodulator, the phase of the reversed bit is compared with the phase of the previous bit.

For detection of the differentially encoded PSK (DPSK), we can use the receiver arrangement as shown in Fig.3 below.

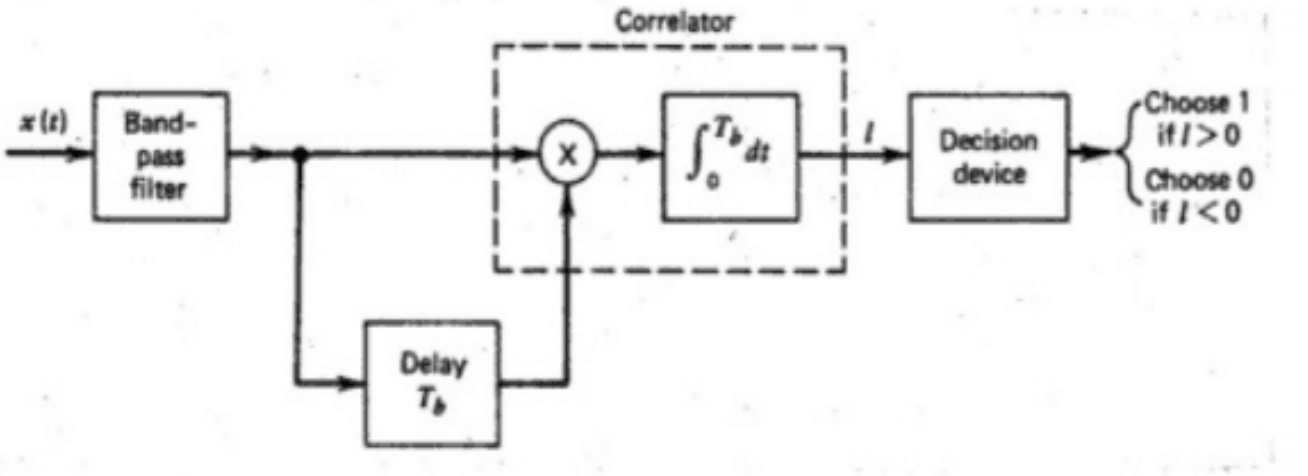


Figure 4-36 DPSK Demodulator

The received DPSK signal is applied to one input of the multiplier and a delayed version of the received DPSK signal by the time interval Tb is applied to the other input of the multiplier.The delayed version of the received DPSK signal (in the absence of noise) has been shown in the 4th row of table.1. The output of the difference is proportional to cos(Φ), here Φ is the difference between the carrier phase angle of the received DPSK signal and its delayed version, measured in the same bit interval. The phase angle of the DPSK signal and its delayed version have been shown in 3rd and 5th rows, respectively. The phase difference between the two sequences for each bit interval is used to determine the sign of the phase comparator output. When Φ =0, the integrator output is positive whereas when Φ=π, the integrator output is negative. By comparing the integrator output with a decision level of zero volt, the decision device can reconstruct the binary sequence by assigning a symbol '0' for negative output and a symbol '1' for positive output. The reconstructed binary data is shown in the last row of the table.1. It is thus seen that in the absence of noise, the receiver can reconstruct the transmitted data exactly.

Binary Data		0	0	1	0	0	1	0	0	1	1
Differentially Encoded Data	1	0	1	1	0	1	1	0	1	1	1
Phase of DPSK	0	π	0	0	π	0	0	π	0	0	0
Shifted Differentially encoded Data d_{k-1}		1	0	1	1	0	1	1	0	1	1
Phase of shifted Data		0	π	0	0	π	0	0	π	0	0
Phase Comparision Output		-	-	+	-	-	+	-	-	+	+
Detected Binary Seq.		0	0	1	0	0	1	0	0	1	1

Figure 4-37 Table.1 for DPSK process

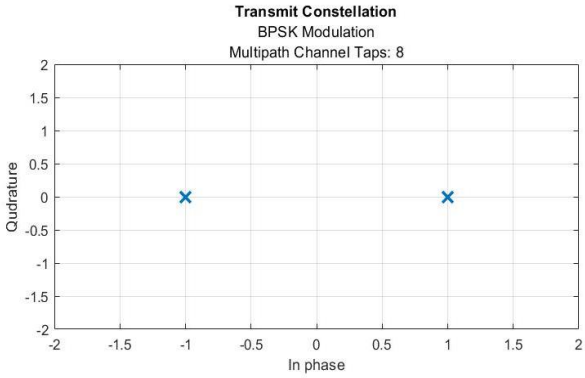
As we mentioned above: By increasing order of modulation BER will increase and probability of recovering data (image for example) will decrease.

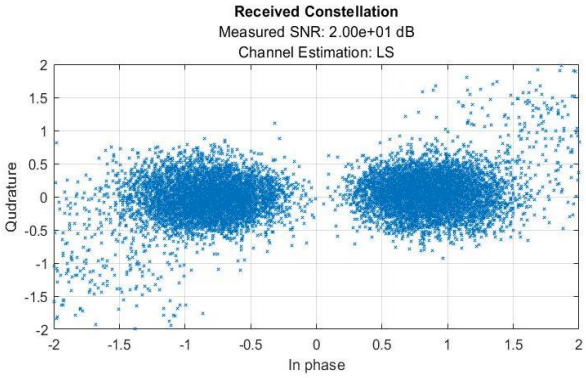
MATLAB code and pictures below explained this phenomenon using OFDM :

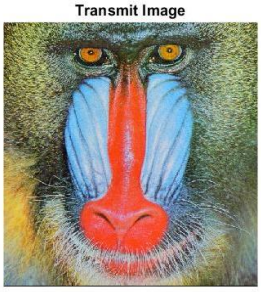
```
%MATLAB code to explain effect of different types of modulation on the same channel responce &
SNR and same data transmitted
clear all
close all
clc
%Modulation method: BPSK, QPSK, 8PSK, 16QAM, 32QAM, 64QAM,
mod_method = input('enter the order of modulation:');
%fft Size
nfft = 64;
n_fft = 64;
%Size of cycle prefix extension
n_cpe = 16;
snr = 20; % in dB
%number of channel taps.
n_taps = 8;
ch_est_method = 'LS';
% ch_est_method = 'none';
mod_methods = {'BPSK', 'QPSK', '8PSK', '16QAM', '32QAM', '64QAM'};
mod_order = find(ismember(mod_methods,mod_method));
im = imread('baboon.png');
im_bin = dec2bin(im(:))';
im_bin = im_bin(:);
sym_rem = mod(mod_order-mod(length(im_bin),mod_order),mod_order);
padding = repmat('0',sym_rem,1);
im_bin_padded = [im_bin;padding];
cons_data = reshape(im_bin_padded,mod_order,length(im_bin_padded)/mod_order)';
cons_sym_id = bin2dec(cons_data);
%Symbol modulatio
%BPSK
if mod_order == 1
   mod_ind = 2^(mod_order-1);
   n = 0:pi/mod_ind:2*pi-pi/mod_ind;
   in_phase = cos(n);
   quadrature = sin(n);
   symbol_book = (in_phase + quadrature*1i);
end
% Phase shift keying about unit circle
if mod_order == 2 || mod_order == 3
   mod_ind = 2^(mod_order-1);
   n = 0:pi/mod_ind:2*pi-pi/mod_ind;
   in_phase = cos(n+pi/4);
   quadrature = sin(n+pi/4);
   symbol_book = (in_phase + quadrature*1i);
end
%16QAM, 64QAM
if mod_order == 4 || mod_order == 6
   mod_ind = sqrt(2^mod_order);
   %n = 0:pi/mod_ind:2*pi-pi/mod_ind;
```

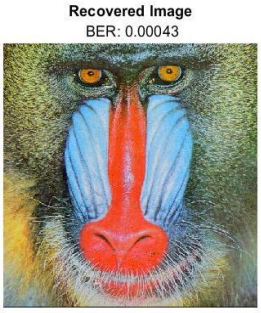
```
    in_phase = repmat(linspace(-1,1,mod_ind),mod_ind,1);
    quadrature = repmat(linspace(-1,1,mod_ind)',1,mod_ind);
    symbol_book = (in_phase(:) + quadrature(:)*1i);
end
%32QAM - Generates 6x6 constellation and removes corners
if mod_order == 5
    mod_ind = 6;
    %n = 0:pi/mod_ind:2*pi-pi/mod_ind;
    in_phase = repmat(linspace(-1,1,mod_ind),mod_ind,1);
    quadrature = repmat(linspace(-1,1,mod_ind)',1,mod_ind);
    symbol_book = (in_phase(:) + quadrature(:)*1i);
    symbol_book = symbol_book([2:5 7:30 32:35]); %corners are removed
end
%modulate data according to the symbol_book
X = symbol_book(cons_sym_id+1);
fft_rem = mod(n_fft-mod(length(X),n_fft),n_fft);
X_padded = [X;zeros(fft_rem,1)];
X_blocks = reshape(X_padded,nfft,length(X_padded)/nfft);
x = ifft(X_blocks);
%Add cyclic prefix entension and shift from parallel to serial
x_cpe = [x(end-n_cpe+1:end,:);x];
x_s   = x_cpe(:);
data_pwr = mean(abs(x_s.^2));
% Add noise to the channel
noise_pwr = data_pwr/10^(snr/10);
noise = normrnd(0,sqrt(noise_pwr/2),size(x_s))+normrnd(0,sqrt(noise_pwr/2),size(x_s))*1i;
x_s_noise = x_s + noise;
% Measure SNR
snr_meas = 10*log10(mean(abs(x_s.^2))/mean(abs(noise.^2)));
g = exp(-(0:n_taps-1));
g = g/norm(g);
x_s_noise_fading = conv(x_s_noise,g,'same');
x_p = reshape(x_s_noise_fading,nfft+n_cpe,length(x_s_noise_fading)/(nfft+n_cpe));
x_p_cpr = x_p(n_cpe+1:end,:);
% Move to frequency domain
X_hat_blocks = fft(x_p_cpr);
if n_taps > 1
    switch(ch_est_method)
        case 'none'
        case 'LS'
            G = X_hat_blocks(:,1)./X_blocks(:,1);
            X_hat_blocks = X_hat_blocks./repmat(G,1,size(X_hat_blocks,2));
    end
end
X_hat = X_hat_blocks(:);
X_hat = X_hat(1:end-fft_rem);
%Recover data from modulated symbols
A=[real(symbol_book) imag(symbol_book)];
if (size(A,2)>2)
    A=[real(symbol_book)' imag(symbol_book)'];
end
rec_syms = knnsearch(A,[real(X_hat) imag(X_hat)])-1;
%Parse to binary stream to remove symbol padding
rec_syms_cons = dec2bin(rec_syms);
```

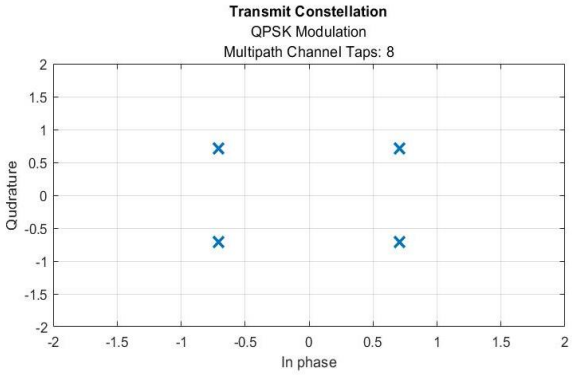
```
rec_im_bin = reshape(rec_syms_cons',numel(rec_syms_cons),1);
rec_im_bin = rec_im_bin(1:end-sym_rem);
ber = sum(abs(rec_im_bin-im_bin))/length(im_bin);
rec_im = reshape(rec_im_bin,8,numel(rec_im_bin)/8);
rec_im = uint8(bin2dec(rec_im'));
rec_im = reshape(rec_im,size(im));
subplot(2,2,1);
plot(X,'x','linewidth',2,'markersize',10);
xlim([-2 2]);
ylim([-2 2]);
xlabel('In phase')
ylabel('Qudrature')
if n_taps > 1
   title(sprintf('\\bfTransmit Constellation\n\\rm%s Modulation\nMultipath Channel Taps:
%d',mod_method,n_taps));
else
   title(sprintf('\\bfTransmit Constellation\n\\rm%s Modulation\nMultipath Channel Taps:
%d',mod_method));
end
grid on
% Recovered constellation
subplot(2,2,2);
plot(X_hat(1:500:end),'x','markersize',3);
xlim([-2 2]);
ylim([-2 2]);
xlabel('In phase')
ylabel('Qudrature')
if n_taps > 1
   title(sprintf('\\bfReceived Constellation\n\\rmMeasured SNR: %.2d dB\nChannel Estimation:
%s',snr_meas,ch_est_method));
else
   title(sprintf('\\bfReceived Constellation\n\\rmMeasured SNR: %.2d dB',snr_meas));
end
grid on
% Original image
subplot(2,2,3);
imshow(im);
title('\bfTransmit Image');
% Recovered image
subplot(2,2,4);
imshow(rec_im);
title(sprintf('\\bfRecovered Image\n \\rmBER: %.2g',ber));
```

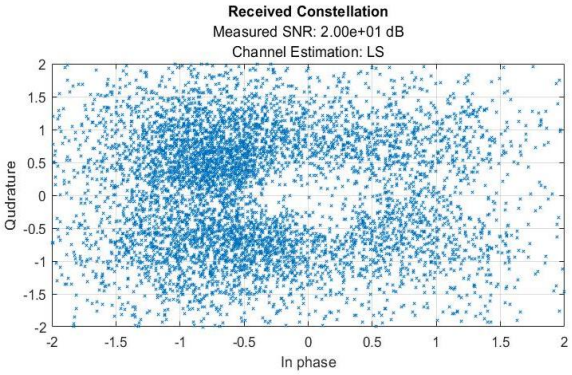

Transmit Constellation
BPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase

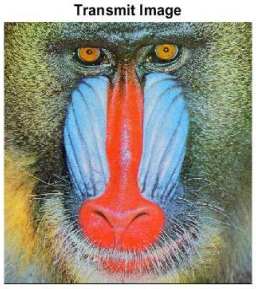

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase

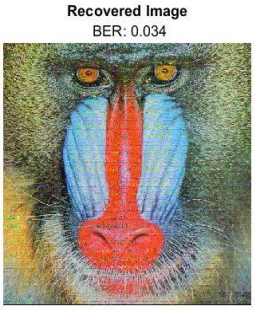

Transmit Image

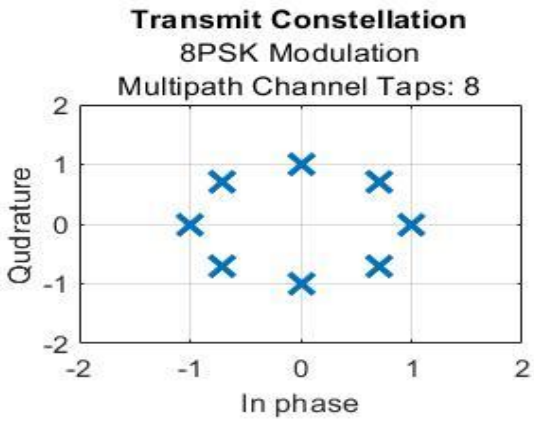

Recovered Image
BER: 0.00043

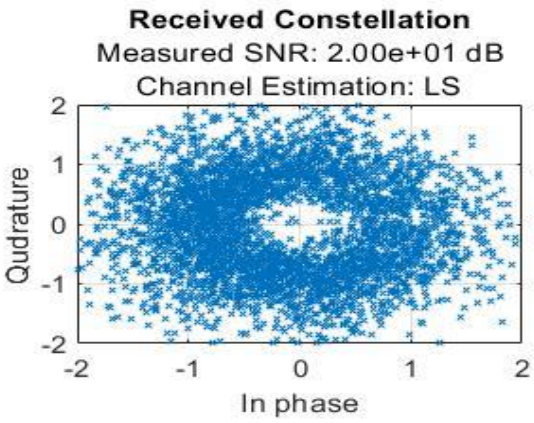

Transmit Constellation
QPSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase

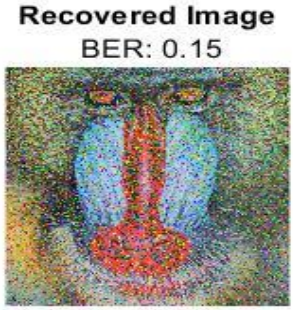

Transmit Image


Recovered Image
BER: 0.034

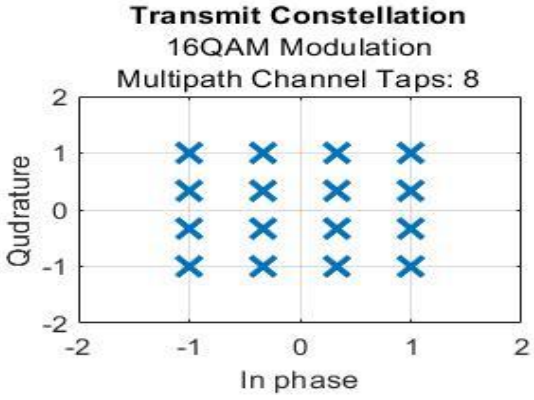

Transmit Constellation
8PSK Modulation
Multipath Channel Taps: 8
Qudrature
In phase

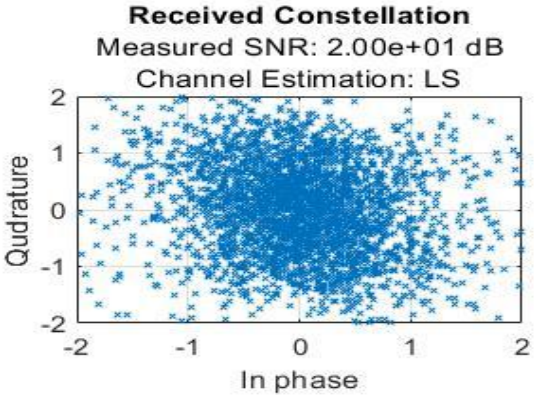

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase

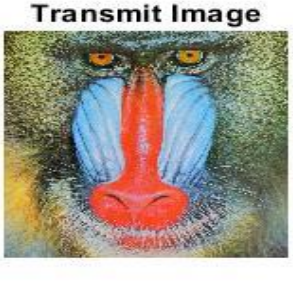

Transmit Image


Recovered Image
BER: 0.15

16QAM:

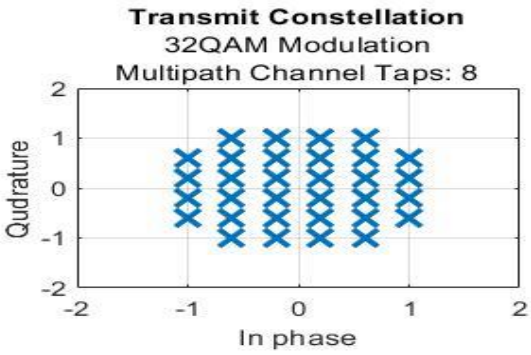

Transmit Constellation
16QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase

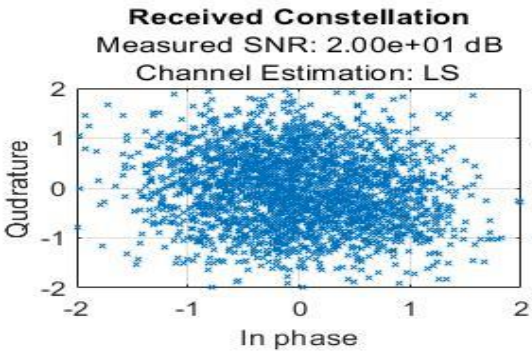

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image


Recovered Image
BER: 0.26

32QAM:

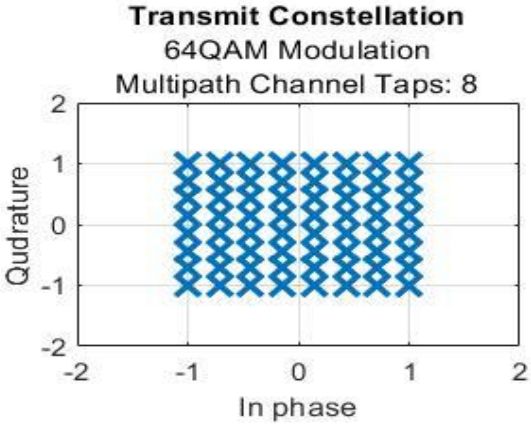

Transmit Constellation
32QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase

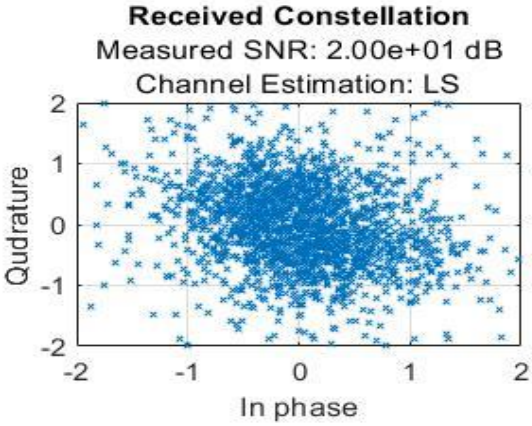

Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase

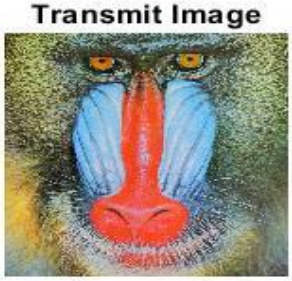

Transmit Image


Recovered Image
BER: 0.32

64QAM:


Transmit Constellation
64QAM Modulation
Multipath Channel Taps: 8
Qudrature
In phase


Received Constellation
Measured SNR: 2.00e+01 dB
Channel Estimation: LS
Qudrature
In phase


Transmit Image


Recovered Image
BER: 0.33

5 Orthogonal Frequency Division Multiplexing (OFDM)

5.1 Introduction

In general, multicarrier schemes subdivide the used channel bandwidth into a number of parallel subchannels as shown in Figure 1a. Ideally the bandwidth of each subchannel is such that they are, ideally, each non-frequency selective (i.e. having a spectrally flat gain); this has the advantage that the receiver can easily compensate for the subchannel gains individually in the frequency domain.

Orthogonal Frequency Division Multiplexing (OFDM) is a special case of Frequency Division Multiplexing (FDM) as shown in figure(5.1) where the non-frequency-selective narrowband subchannels, into which the frequency-selective wideband channel is divided, are overlapping but orthogonal, as shown in Figure 1b. This avoids the need to separate the carriers by means of guard-bands, and therefore makes OFDM highly spectrally efficient. The spacing between the subchannels in OFDM is such that they can be perfectly separated at the receiver. This allows for a low complexity receiver implementation, which makes OFDM attractive for high-rate mobile data transmission such as the LTE downlink.

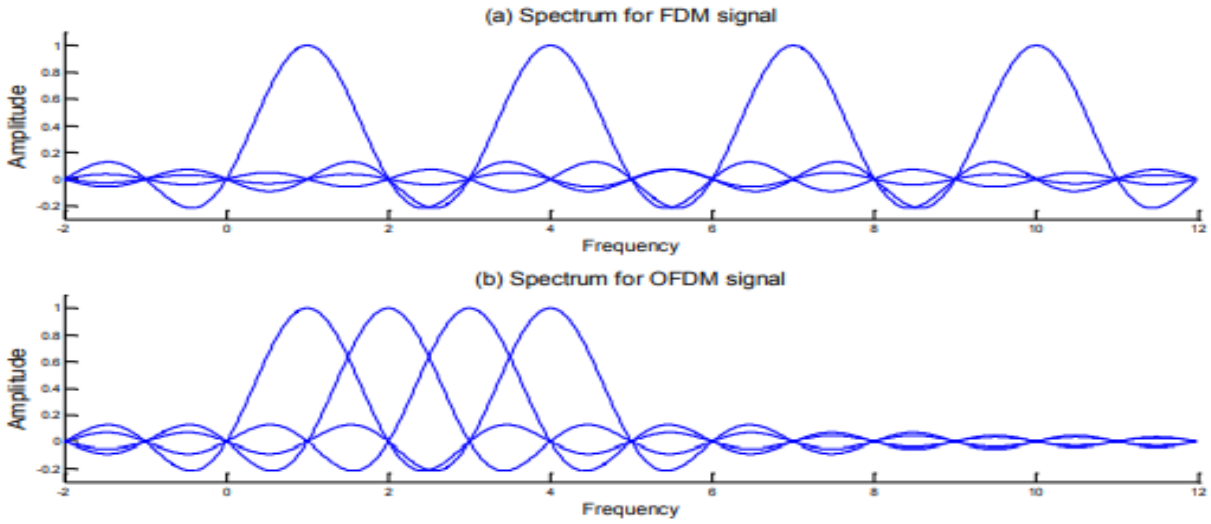


Figure 5-0-1: subcarrier in FDM and OFDM Systems

It is worth noting that the advantage of separating the transmission into multiple narrowband subchannels cannot itself translate into robustness against time-variant channels if no channel coding is employed. The LTE downlink combines OFDM with channel coding and Hybrid Automatic Repeat request (HARQ) to overcome the deep fading which may be encountered on the individual

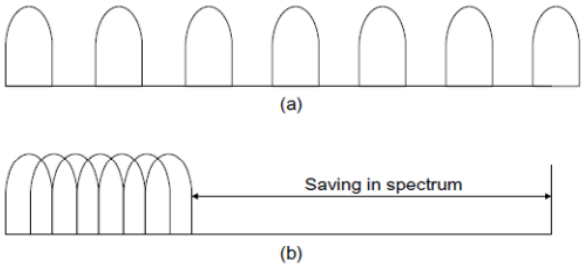


Figure 5-0-2: Spectral efficiency of OFDM compared to classical multicarrier modulation: (a) classical multicarrier system spectrum; (b) OFDM system spectrum.

subchannel

Why OFDM

Transmission by means of OFDM can be seen as a kind of multi-carrier transmission. The basic characteristics of OFDM transmission, which distinguish it from a straightforward multi-carrier extension of a more narrowband transmission scheme as outlined in Figure 3 are

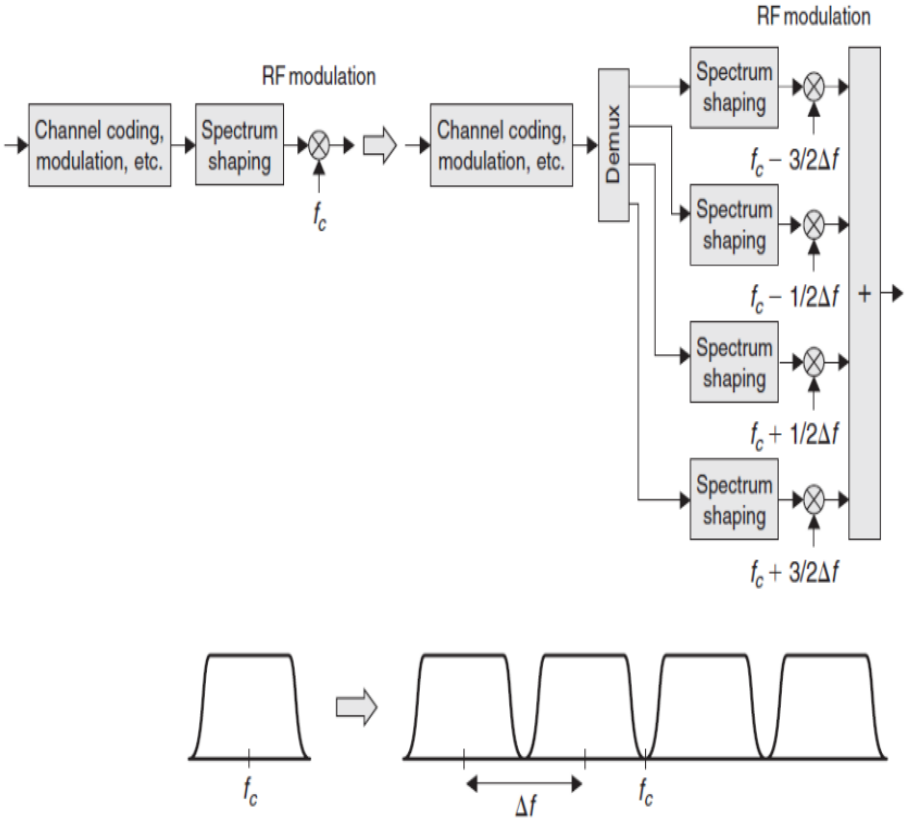


Figure 5-0-3: Extension to wider transmission bandwidth by means of multi-carrier transmission.

- The use of a relatively large number of narrowband subcarriers. In contrast, a straightforward multi-carrier extension as outlined in Figure 3 would typically consist of only a few subcarriers, each with a relatively wide bandwidth. As an example, a WCDMA multi-carrier evolution to a 20MHz overall transmission bandwidth could consist of four (sub)carriers, each with a bandwidth in the order of 5 MHz. In comparison, OFDM transmission may imply that several hundred subcarriers are transmitted over the same radio link to the same receiver.

- Simple rectangular pulse shaping as illustrated in Figure 4a. This corresponds to a sinc-square-shaped per-subcarrier spectrum, as illustrated in Figure 4b.

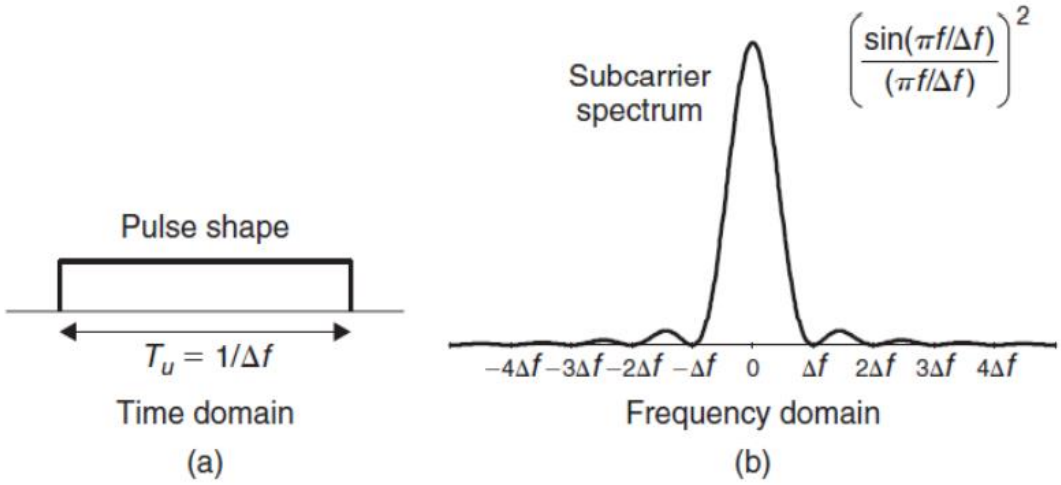


Figure 5-0-4: Per-subcarrier pulse shape and spectrum for basic OFDM transmission.

- Tight frequency-domain packing of the subcarriers with a subcarrier spacing $\Delta f = \frac{1}{T_u}$, where Tu is the per-subcarrier modulation-symbol time (see Figure5). The subcarrier spacing is thus equal to the per subcarrier modulation rate $\frac{1}{T_u}$. An illustrative description of a basic OFDM modulator is provided in

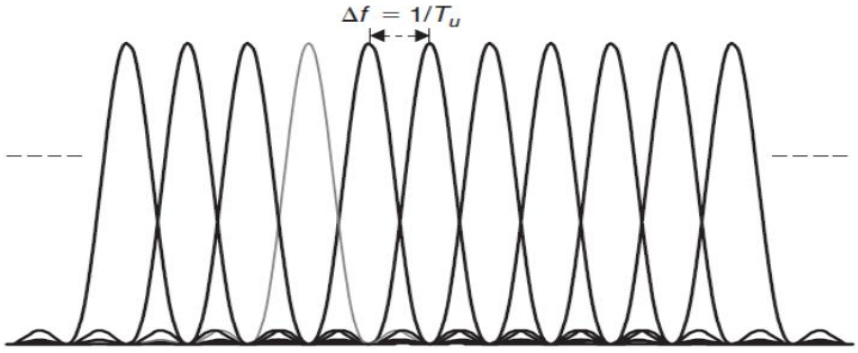


Figure 5-0-5: OFDM subcarrier spacing.

Figure 5.

It consists of a bank of N_c complex modulators, where each modulator corresponds to one OFDM subcarrier.

In complex baseband notation, a basic OFDM signal x(t) during the time interval $mT_u \leq t < (m+1)T_u$ can thus be expressed as :

$$\boldsymbol{x(t) = \sum_{K=1}^{N-1} X_k^m \, e^{j2\pi k \Delta f t}} \qquad (\mathbf{1})$$

where $x_k(t)$ is the k_{th} modulated subcarrier with frequency $f_k = k\,\Delta f$ and X_k^m is the, in general complex, modulation symbol applied to the k_{th} subcarrier during the m_{th} OFDM symbol interval, i.e. during the time interval $mT_u \leq t < (m+1)T_u$. OFDM transmission is thus block based, implying that, during each OFDM symbol interval, N_c modulation symbols are transmitted in parallel. The modulation symbols can be from any modulation alphabet, such as QPSK, 16QAM, or 64QAM.

The number of OFDM subcarriers can range from less than one hundred to several thousand, with the subcarrier spacing ranging from several hundred kHz down to a few kHz. What subcarrier spacing to use depends on what types of environments the system is to operate in, including such aspects as the maximum expected radio channel frequency selectivity (maximum expected time dispersion) and the maximum expected rate of channel variations (maximum expected Doppler spread). Once the subcarrier spacing has been selected, the number of subcarriers can be decided based on the assumed overall transmission bandwidth, taking into account acceptable out of-band emission, etc.

As an example, for 3GPP LTE the basic subcarrier spacing equals 15 kHz. On the other hand, the number of subcarriers depends on the transmission bandwidth, with in the order of 600 subcarriers in case of operation in a 10MHz spectrum allocation and correspondingly fewer/more subcarriers in case of smaller/larger overall transmission bandwidths.

5.2 Orthogonal Multiplexing Principle

Signals are orthogonal if they are mutually independent of each other Orthogonality is a property that allows multiple information signals to be transmitted perfectly over a common channel and detected, without interference. Mathematically, two functions are orthogonal if their product when integrated over certain interval gives zero. We note that although subcarriers overlap in time , we can separate them due to their orthogonality

$$\int_{mT_u}^{(m+1)T_u} x_{k1}(t)\; x_{k2}^{*}(t)$$

$$= \int_{mT_u}^{(m+1)T_u} a_{k1}(t)\; a_{k2}^{*}(t)\; e^{j2\pi k1\Delta ft}\; e^{-j2\pi k1\Delta ft} \qquad (2)$$

A high-rate data stream typically faces the problem of having a symbol period T_s much smaller than the channel delay spread T_d if it is transmitted serially. This generates Inter-Symbol Interference (ISI) which can only be undone by means of a complex equalization procedure. In general, the equalization complexity grows with the square of the channel impulse response length. In OFDM, the high-rate stream of data symbols is first Serial to-Parallel (S/P) converted for modulation onto M parallel subcarriers as shown in Figure 6. This increases the symbol duration on each subcarrier by a factor of approximately M, such that it becomes significantly longer than the channel delay spread. This operation has the important advantage.of requiring a much less complex equalization procedure in the receiver, under the assumption that the time-varying channel impulse response remains substantially constant during the transmission of each

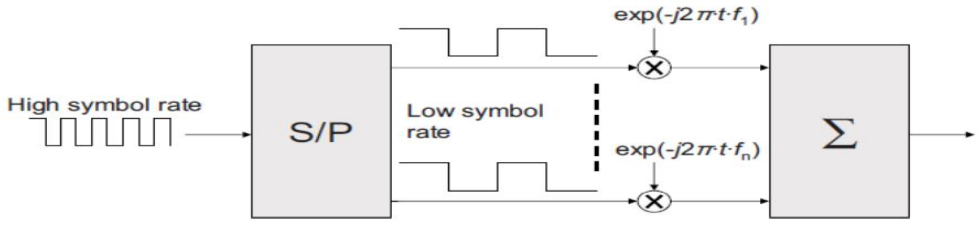


Figure 5-0-6: Serial-to-Parallel (S/P) conversion operation for OFDM.

modulated OFDM symbol. Figure 6 shows how the resulting long symbol

duration is virtually unaffected by ISI compared to the short symbol duration, which is highly corrupted.

Figure 7 shows the typical block diagram of an OFDM system. The signal to be transmitted is defined in the frequency domain.

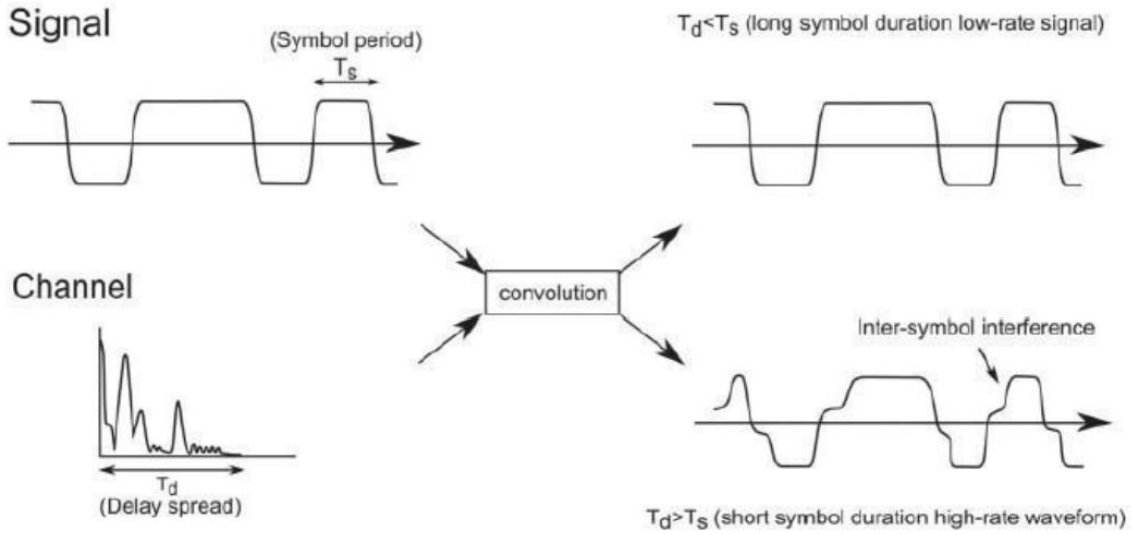


Figure 5-0-7:Effect of channel on signals with short and long symbol duration

An S/P converter collects serial data symbols into a data $block\ S[k] = [S_0[k], S_1[k], .., S$ of dimension M, where k is the index of an OFDM symbol (spanning the M subcarriers) . The M parallel data streams are first independently modulated resulting in the complex vector

$$X[k] = \left[X_0[k], X_1[k], \dots, X_{M-1}[k]\right]T$$

Note that in principle it is possible to use different modulations (e.g. QPSK or 16QAM) on each subcarrier; due to channel frequency selectivity, the channel gain may differ between subcarriers, and thus some subcarriers can carry higher data rates than others. The vector $X\ [k]$ is then used as input to an N-point Inverse FFT (IFFT) resulting in a set of N complex time domain samples

$$\mathrm{X}[k] = \left[x_0[k], \dots, x_{N-1}[k]\right]T\ .$$

In a practical OFDM system, the number of processed subcarriers is greater than the number of modulated subcarriers (i.e. N ≥ M), with the un-modulated subcarriers being padded with zeros.

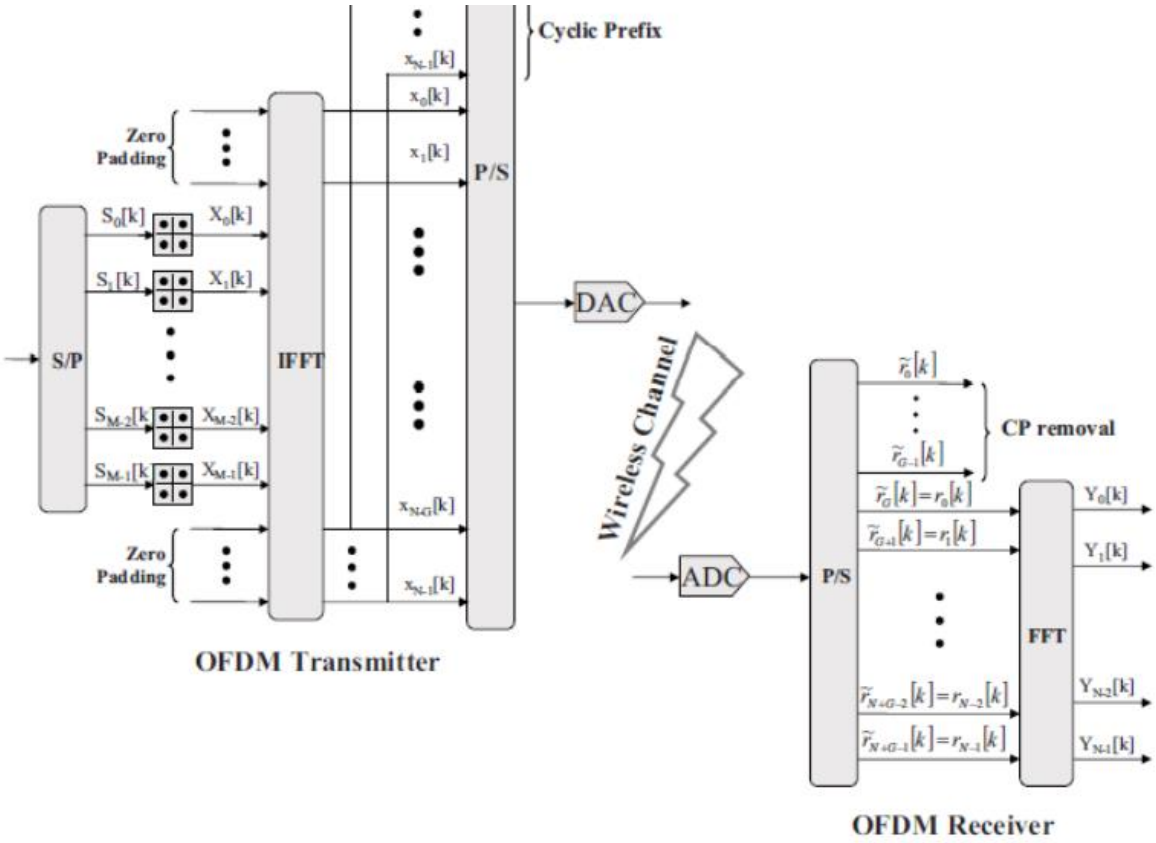


Figure 5-0-8: OFDM system model: (a) transmitter; (b) receiver.

The next key operation in the generation of an OFDM signal is the creation of a guard period at the beginning of each OFDM symbol x [k] by adding a Cyclic Prefix (CP), to eliminate the remaining impact of ISI caused by multipath propagation. The CP is generated by duplicating the last G samples of the IFFT output and appending them at the beginning of $x\ [k]$. This yields the time domain OFDM symbol $[x_{N-G}[k], \ldots, x_{N-1}[k], x0[k], \ldots, x_{N-1}[k]]T$, as shown in 9

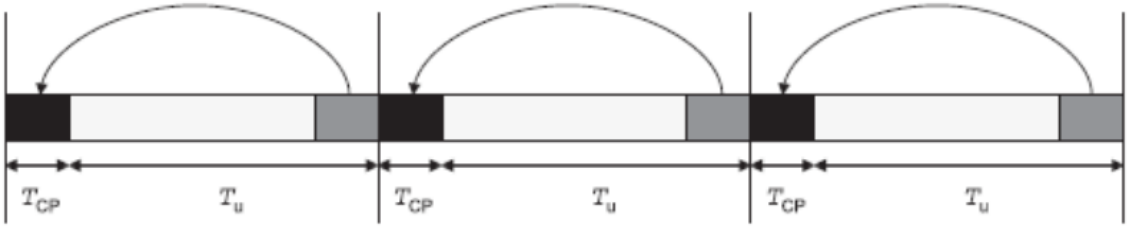


Figure 5-0-9:OFDM Cyclic Prefix (CP) insertion

mission through the frequency-selective channel. At the receiver, the reverse operations are performed to demodulate the OFDM signal. Assuming that time- and frequency-synchronization is achieved , a number of samples corresponding to the length of the CP are removed, such that only an ISI free block of samples is passed to the DFT. If the number of subcarriers N is designed to be a power of 2, a highly efficient FFT implementation may be used to transform the signal back to the frequency domain. Among the N parallel streams output from the FFT, the modulated subset of M subcarriers are selected and further processed by the receiver. Let x(t) be the symbol transmitted at time instant t. The received signal in a multipath environment is then given by

$$r(t) = x(t) * h(t) + z(t) \qquad (3)$$

where $h(t)$ is the continuous-time impulse response of the channel, $*$ represents the convolution operation and $z(t)$ is the additive noise. Assuming that $x(t)$ is band-limited to $[-\frac{1}{2T_s}, \frac{1}{2T_s}]$, the continuous-time signal $x(t)$ can be sampled at sampling rate Ts such that the Nyquist criterion is satisfied. As a result of the multipath propagation, several replicas of the transmitted signals arrive at the receiver at different delays.

5.2 OFDM advantage and disadvantages

OFDM advantages

- High data rate by transmitting on a number of subcarriers simultaneously.
- High spectral utilization by spacing the subcarriers very closely.
- High robustness against time dispersion of multipath channel by elongating symbol duration.
- High resistance to ISI and ICI by introducing cyclic prefix (CP).
- Less implementation complexity by using forward and inverse fast Fourier transform (FFT/IFFT) pair.
- Simple equalization to remove channel effects in the frequency domain.

OFDM disadvantages

- Intercarrier interference (ICI) due to phase noise and carrier frequency offset which destroy the orthogonality.
- Inter-symbol ISI due to channel delays and dispersion.

5.3 Cyclic Prefix Insertion

uncorrupted OFDM signal can be demodulated without any interference between subcarriers. One way to understand this subcarrier orthogonality is to recognize that a modulated subcarrier $X(t)$ consists of an integer number of periods of complex exponentials during the demodulator integration interval $T_u = 1/\Delta f$ However, in case of a time-dispersive channel the orthogonality between the subcarriers will, at least partly, be lost.

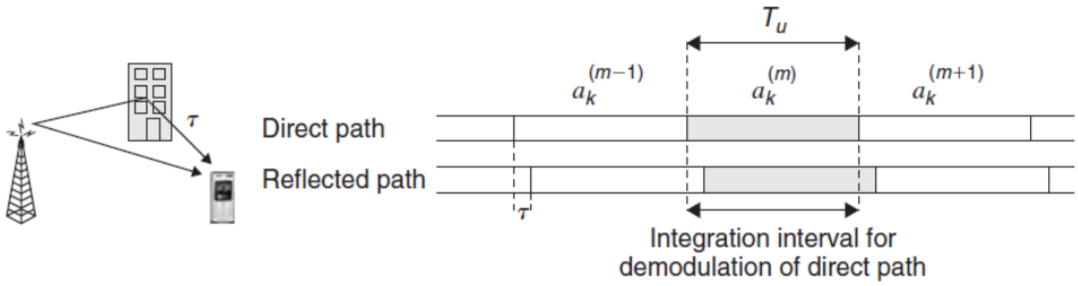


Figure 5-0-10 : Time dispersion and corresponding received-signal timing.

The reason for this loss of subcarrier orthogonality in case of a time-dispersive channel is that, in this case, the demodulator correlation interval for one path will overlap with the symbol boundary of a different path, as illustrated in Figure 10 Thus, the integration interval will not necessarily correspond to an integer number of periods of complex exponentials of that path as the modulation symbols $a_k$may differ between consecutive symbol intervals. As a consequence, in case of a time-dispersive channel there will not only be inter-symbol interference within a subcarrier but also interference between subcarriers.

Another way to explain the interference between subcarriers in case of a time dispersive channel is to have in mind that time dispersion on the radio channel is equivalent to a frequency-selective channel frequency response. Orthogonality between OFDM subcarriers is not simply due to frequency domain separation but due to the specific frequency-domain structure of each subcarrier. Even if the frequency-domain channel is constant over a bandwidth corresponding to the main lobe of an OFDM subcarrier and only the subcarrier side lobes are corrupted due to the radio-channel frequency selectivity, the orthogonality between subcarriers will be lost with inter subcarrier interference as a consequence. Due to the relatively large side lobes of each OFDM subcarrier, already a relatively limited amount of time dispersion or, equivalently, a relatively modest radio-

channel frequency selectivity may cause non-negligible interference between subcarrier.

To deal with this problem and to make an OFDM signal truly insensitive to time dispersion on the radio channel, so-called cyclic-prefix insertion is typically used in case of OFDM transmission. As illustrated in Figure 11, cyclic-prefix insertion implies that the last part of the OFDM symbol is copied and inserted at the beginning of the OFDM symbol. Cyclic-prefix insertion thus increases the length of the OFDM symbol from $T_u \; to \; T_u \; + \; T_{cp}$, where T_{cp} is the length of the cyclic prefix, with a corresponding reduction in the OFDM symbol rate as a consequence.

As illustrated in the lower part of Figure 11, if the correlation at the receiver side is still only carried out over a time interval $Tu \; = 1/\Delta f$, subcarrier orthogonality will then be preserved also in case of a time-dispersive channel, as long as the span of the time dispersion is shorter than the cyclic-prefix length.

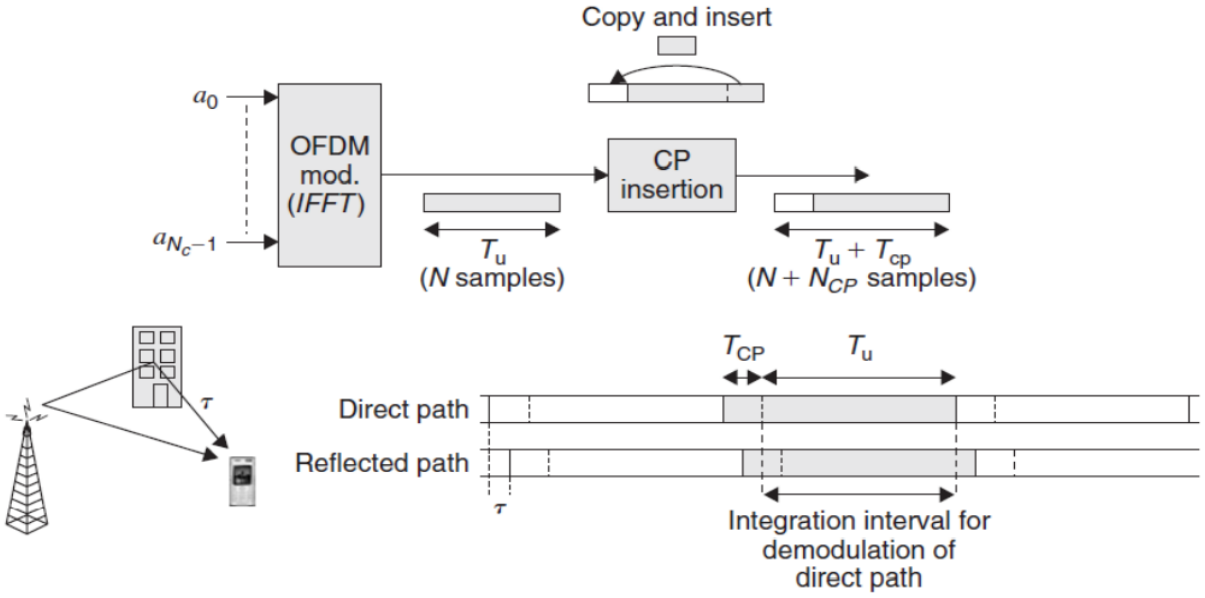


Figure 5-0-11:Cyclic-prefix insertion.

In practice, cyclic prefix insertion is carried out on the time discrete output of the transmitter IFFT. Cyclic-prefix insertion then implies that the last N_{cp} samples of the IFFT output block of length N is copied and inserted at the beginning of the block, increasing the block length from $N \; to \; N \; + \; N_{cp}$. At the receiver side, the

corresponding samples are discarded before OFDM demodulation by means of, for example, DFT/FFT processing.

Cyclic-prefix insertion is beneficial in the sense that it makes an OFDM signal insensitive to time dispersion as long as the span of the time dispersion does not exceed the length of the cyclic prefix. The drawback of cyclic prefix insertion is that only a fraction $T_u/(T_u + T_{cp})$ of the received signal power is actually utilized by the OFDM demodulator, implying a corresponding power loss in the demodulation. In addition to this power loss, cyclic prefix insertion also implies a corresponding loss in terms of bandwidth as the OFDM symbol rate is reduced without a corresponding reduction in the overall signal bandwidth.

One way to reduce the relative overhead due to cyclic-prefix insertion is to reduce the subcarrier spacing Δf , with a corresponding increase in the symbol time T_u as a consequence. However, this will increase the sensitivity of the OFDM transmission to fast channel variations, that is high Doppler spread, as well as different types of frequency errors.

It is also important to understand that the cyclic prefix does not necessarily have to cover the entire length of the channel time dispersion. In general, there is a trade-off between the power loss due to the cyclic prefix and the signal corruption (inter-symbol and inter-subcarrier interference) due to residual time dispersion not covered by the cyclic prefix and, at a certain point, further reduction of the signal corruption due to further increase of the cyclic-prefix length will not justify the corresponding additional power loss. This also means that, although the amount of time dispersion typically increases with the cell size, beyond a certain cell size there is often no reason to increase the cyclic prefix further as the corresponding power loss due to a further increase of the cyclic prefix would have a larger negative impact, compared to the signal corruption due to the residual time dispersion not covered by the cyclic prefix.

5.4 Circular convolution

- When an input data stream x[n] is sent through a linear time-invariant FIR channel h[n] the output is the linear convolution: $y[n] = x[n] * h[n]$.
- If the convolution is circular convolution, it is possible to take the DFT of the channel output y[n] to get: DFT$\{y[n]\} = DFT\{x[n] * h[n]\}$ Or in the frequency domain: $Y[m] = X[m]H[m]$.
- This formula describes an ISI-free channel in the frequency domain, where each input symbol X[m] is simply scaled by a complex-value H[m].
- For the convolution to be circular we need to add a cyclic prefix.
- If the maximum channel delay spread has a duration of $N + 1$ samples, then by adding a guard band of at least N samples between OFDM symbols, each OFDM symbol is made independent of those coming before and after it, and so the ISI between OFDM symbols is avoided.
- The channel output y is decomposed into a simple multiplication of the channel frequency response $H = DFT\{h\}$ and the channel frequency domain input,X = DF T{x}.
- The cyclic prefix is not entirely free. It comes with both a bandwidth and power penalty.
- Since N redundant symbols are sent, the required bandwidth for OFDM increases from B to $(L + N/L)B$.
- An additional v symbols must be counted against the transmit power budget.
- The use of cyclic prefix entails data rate and power losses that are both: Rate Loss = PowerLoss = $L/(L + V)$

5.5 Frequency-domain model of OFDM transmission

Assuming a sufficiently large cyclic prefix, the linear convolution of a time dispersive radio channel will appear as a circular convolution during the demodulator integration interval Tu. The combination of OFDM modulation (IFFT processing), a time-dispersive radio channel, and OFDM demodulation (FFT processing) can then be seen as a frequency-domain channel as illustrated in Figure12, where the frequency-domain channel taps $H_0, \ldots, H_{N_{c-1}}$ can be directly derived from the channel impulse response.

The demodulator output b_k in Figure 12 is the transmitted modulation symbol a_k scaled and phase rotated by the complex frequency-domain channel tap H_k and impaired by noise n_k. To properly recover the transmitted symbol for further processing, for example data demodulation and channel decoding, the receiver should multiply bk with the complex conjugate of H_k, as illustrated in Figure 13, This is often expressed as a one-tap equalizer being applied to each received subcarrier

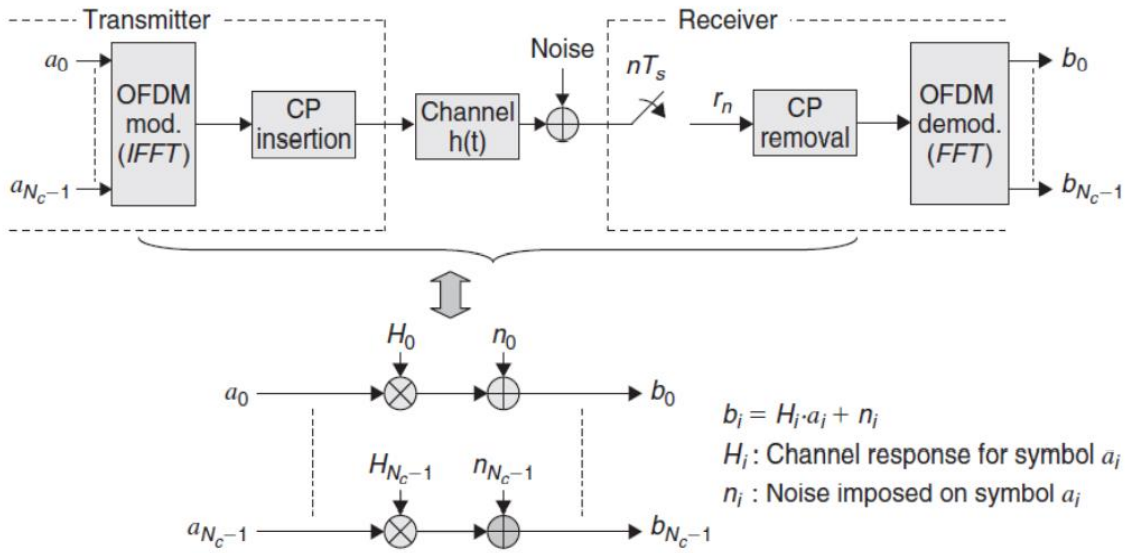


Figure 5-0-13: Frequency-domain model of OFDM transmission/reception.

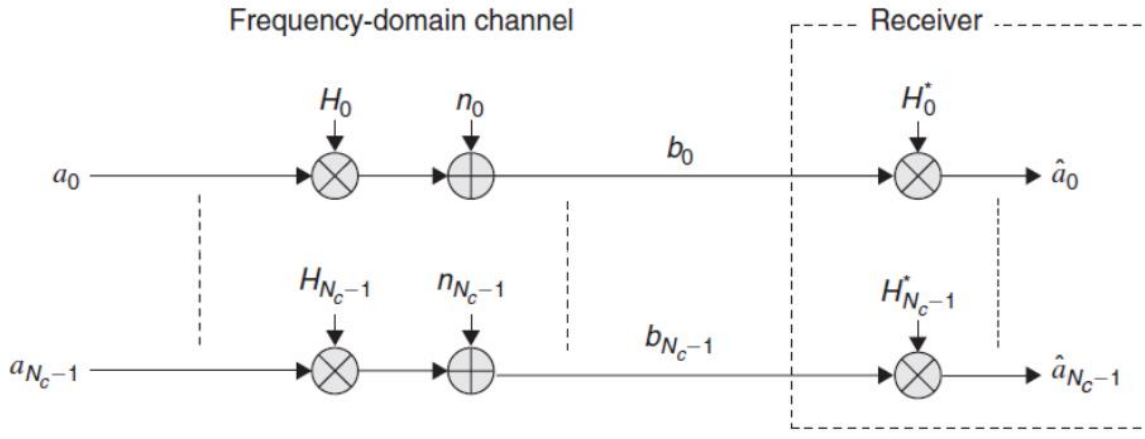


Figure 5-0-12 : frequency-domain model of OFDM transmission/reception with one-tap equalization at the receiver.

5.6 Channel estimation and reference symbols

As described above, to demodulate the transmitted modulation symbol a_k and allow for proper decoding of the transmitted information at the receiver side, scaling with the complex conjugate of the frequency-domain channel tap H_k should be applied after OFDM demodulation (FFT processing) (see Figure 13). To be able to do this, the receiver obviously needs an estimate of the frequency domain channel taps $H_0, \ldots, H_{N_{c-1}}$. The frequency domain channel taps can be estimated indirectly by first estimating the channel impulse response and, from that, calculate an estimate of H_k. However, a more straight forward approach is to estimate the frequency-domain channel taps directly. This can be done by inserting known reference symbols, sometimes also referred to as pilot symbols, at regular intervals within the OFDM time-frequency grid, as illustrated in Figure 14. Using knowledge about the reference symbols, the receiver can estimate the frequency domain channel around the location of the reference symbol. The reference symbols should have a sufficiently high density in both the time and the frequency domain to be able to provide estimates for the entire time/frequency grid also in case of radio channels subject to high frequency and/or time selectivity. Different more or less advanced algorithms can be used for the channel estimation, ranging from simple averaging in combination with linear interpolation to Minimum-Mean-Square-Error (MMSE) estimation relying on more detailed knowledge of the channel time/frequency-domain characteristics.

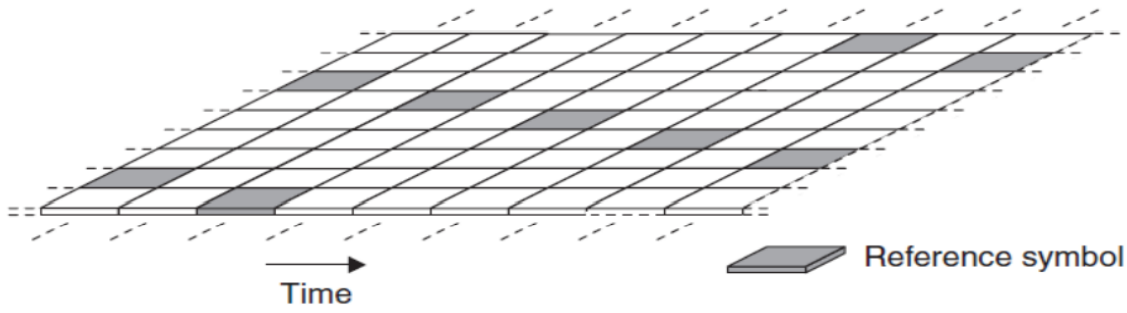


Figure 5-0-14: Time-frequency grid with known reference symbols.

5.7 OFDM as a user-multiplexing and multiple-access scheme

The discussion has, until now, implicitly assumed that all OFDM subcarriers are transmitted from the same transmitter to a certain receiver, i.e.:

- Downlink transmission of all subcarriers to a single mobile terminal.
- Uplink transmission of all subcarriers from a single mobile terminal.

However, OFDM can also be used as a user-multiplexing or multiple access scheme, allowing for simultaneous frequency-separated transmissions to/from multiple mobile terminals. See Figure 15

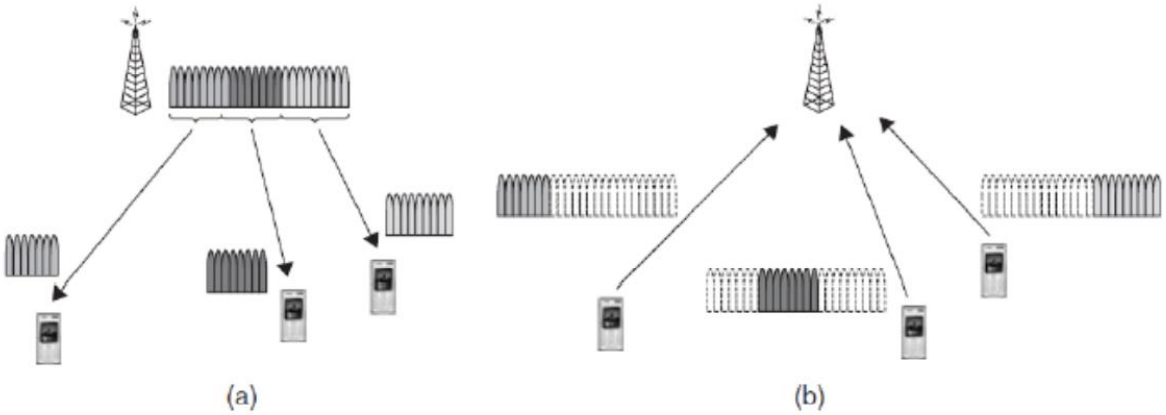


Figure 5-0-15: OFDM as a user-multiplexing/multiple-access scheme : (a) downlink and (b) uplink

In the downlink direction, OFDM as a user-multiplexing scheme implies that, in each OFDM symbol interval, different subsets of the overall set of available subcarriers are used for transmission to different mobile terminals (see Figure 15a).

Similarly, in the uplink direction, OFDM as a user-multiplexing or multiple access scheme implies that, in each OFDM symbol interval, different subsets of the overall set of subcarriers are used for data transmission from different mobile terminals Assumes that consecutive subcarriers are used for transmission to/from the same mobile terminal. However, distributing the subcarriers to/from a mobile terminal in the frequency domain is also possible as illustrated in Figure 16. The benefit of such distributed user multiplexing or distributed

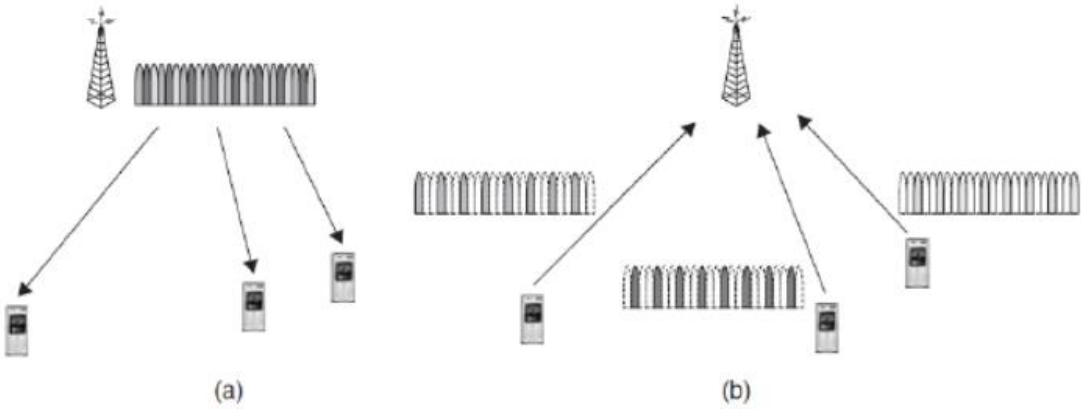


Figure 5-0-16:Distributed user multiplexing.

multiple access is a possibility for additional frequency diversity as each transmission is spread over a wider bandwidth. In the case when OFDMA is used as an uplink multiple-access scheme, i.e.
in case of frequency multiplexing of OFDM signals from multiple mobile terminals, it is critical that the transmissions from the different mobile terminals arrive approximately time aligned at the base station. More specifically, the transmissions from the different mobile terminals should arrive at the base station with a timing misalignment less than the length of the cyclic prefix to preserve orthogonality between subcarriers received from different mobile terminals and thus avoid inter-user interference.

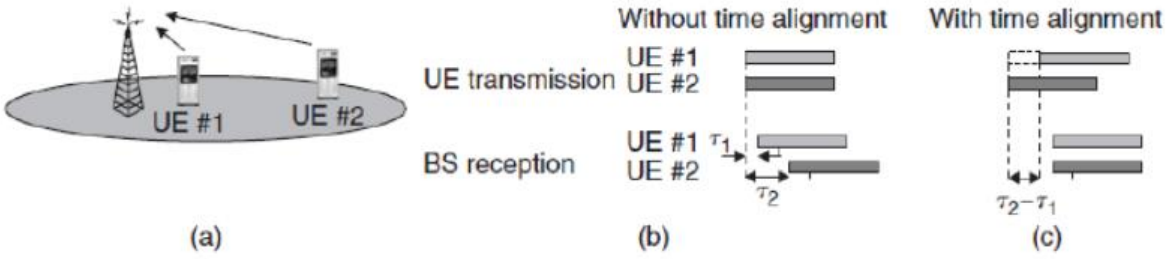


Figure 5-0-17:Uplink transmission-timing control.

Due to the differences in distance to the base station for different mobile terminals and the corresponding differences in the propagation time (which may far exceed the length of the cyclic prefix), it is therefore necessary to control the uplink transmission timing of each mobile terminal (see Figure 17). Such transmit timing control should adjust the transmit timing of each mobile terminal to ensure that uplink transmissions arrive approximately time aligned at the base station.

As the propagation time changes as the mobile terminal is moving within the cell, the transmit timing control should be an active process, continuously adjusting the exact transmit timing of each mobile terminal.

Furthermore, even in case of perfect transmit timing control, there will always be some interference between subcarriers e.g. due to frequency errors. Typically this interference is relatively low in case of reasonable frequency errors, Doppler spread, etc. However, this assumes that the different subcarriers are received with at least approximately the same power. In the uplink, the propagation distance and thus the path loss of the different mobile-terminal transmissions may differ significantly. If two terminals are transmitting with the same power, the received-signal strengths may thus differ significantly, implying a potentially significant interference from the stronger signal to the weaker signal unless the subcarrier orthogonality is perfectly retained. To avoid this, at least some degree of uplink transmit power control may need to be applied in case of uplink OFDMA, reducing the transmit power of user terminals close to the base station and ensuring that all received signals will be of approximately the same power.

5.8 The downlink physical resource:

LTE downlink transmission is based on OFDM. The basic LTE downlink physical resource can thus be seen as a time-frequency resource grid (Figure 18), where each resource element corresponds to one OFDM subcarrier during one OFDM symbol interval.

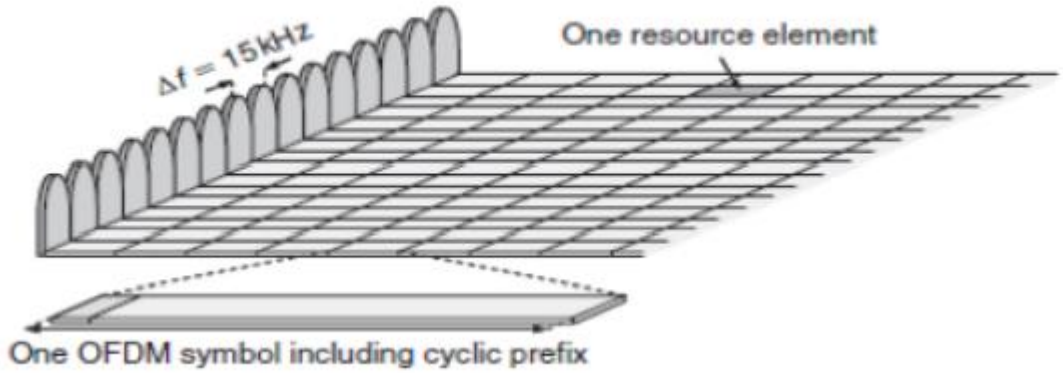


Figure 5-0-18:The LTE downlink physical resource.

For LTE, the OFDM subcarrier spacing has been chosen to Δf =15 kHz. Assuming an FFT-based transmitter/receiver implementation, this corresponds to a sampling rate $f_s = 15\,000 * NFFT$, where NFFT is the FFT size. The basic time unit T_s defined in the pre-vinous section can thus be seen as the sampling time of an FFT-based transmitter/receiver implementation with an FFT size equal to 2048.

It is important to understand though that the time unit Ts is introduced in the LTE radio-access specifications purely as a tool to define different time intervals and does not impose any specific transmitter and/or receiver implementation constraints (e.g. a certain sampling rate).

In practice, an FFT-based transmitter/receiver implementation with an FFT size equal to 2048 and a corresponding sampling rate of 30.72 MHz is suitable for the wider LTE transmission bandwidths, such as bandwidths in the order of 15 MHz and above. However, for smaller transmission bandwidths, a smaller FFT size and a correspondingly lower sampling rate can very well be used. As an example, for transmission bandwidths in the order of 5 MHz, an FFT size equal to 512 and

a corresponding sampling rate of 7.68 MHz may be sufficient.

Assuming a power-of-two FFT size and a subcarrier spacing of 15 kHz, the sampling rate $\Delta f * NFFT$ will be a multiple or submultiple of the WCDMA/HSPA chip rate (3.84 Mcps). This relation can be utilized when implementing multimode terminals supporting both WCDMA/HSPA and LTE.

In addition to the 15 kHz subcarrier spacing, a reduced subcarrier spacing Δ flow = 7.5 kHz with twice as long OFDM symbol time is also defined for LTE. The reduced subcarrier spacing specifically targets MBSFN-based multicast/broadcast transmissions.

As illustrated in Figure 19, in the frequency domain the downlink subcarriers are grouped into resource blocks, where each resource block consists of 12 consecutive sub-carriers. In addition, there is an unused DC-subcarrier in the center of the downlink band. The reason why the DC-subcarrier is not used for downlink transmission is that it may be subject to un-proportionally high interference, for example, due to local-oscillator leakage.

The LTE physical-layer specification allows for a downlink carrier to consist of any number of resource blocks, ranging from a minimum of 6 resource blocks up to a maximum of 110 resource blocks. This corresponds to an overall downlink transmission bandwidth ranging from roughly 1 MHz up to in the order of 20 MHz with very fine granularity and thus allows for

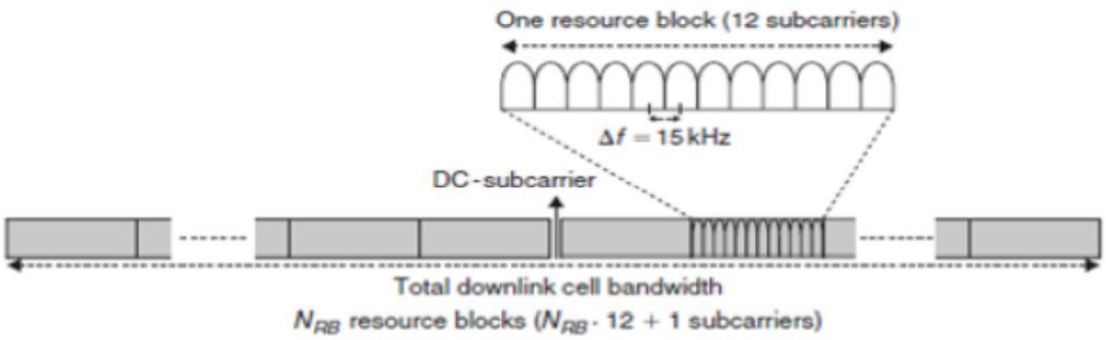


Figure 5-0-19: Frequency-domain structure for LTE downlink.

a very high degree of LTE bandwidth flexibility, at least from a physical layer specification point-of-view. However, LTE radio-frequency requirements are, at least initially, only specified for a limited set of transmission bandwidths,

corresponding to a limited set of possible values for the number of resource blocks within a carrier.

Figure 20 outlines the more detailed time-domain structure for LTE downlink transmission. Each 1 ms subframe consists of two equally sized slots of length $T_{slot} = 0.5\,ms\ (15\,360 * T_s)$. Each slot then consists of a number of OFDM symbols including cyclic prefix. A subcarrier spacing of 15 kHz corresponds to a useful symbol time of approximately 66.7 μ s. The overall OFDM symbol time is then the sum of the useful symbol time and the cyclic-prefix length.

As illustrated in Figure 20 , LTE defines two cyclic-prefix lengths, the normal cyclic prefix and an extended cyclic prefix, corresponding to seven and six OFDM symbols per slot, respectively. The exact cyclic-prefix lengths, expressed in the basic time unit Ts , are given in Figure 21. It can be noted that, in case of the normal cyclic prefix, the cyclic-prefix length for the first OFDM symbol of a slot is some what larger, compared to the remaining OFDM symbols. The reason for this is simply to fill the entire 0.5 ms slot as the number of basic time units T_s per slot (15 360) is not dividable by seven.

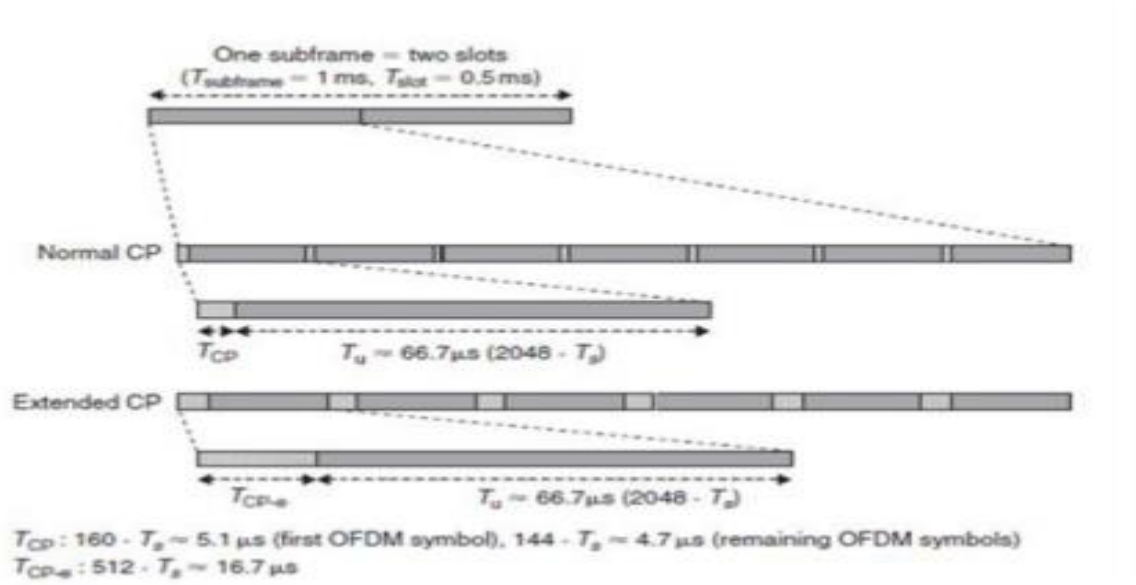


Figure 5-0-20:detailed time domain structure for LTE downlink transmission.

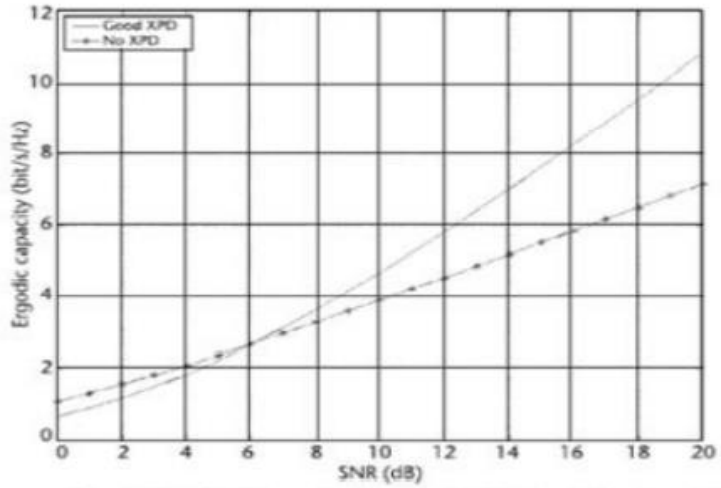


Figure 0-21 Ergodic capacity of a MIMO channel with good XPD ($\alpha = 0$) and no XPD ($\alpha = 1$).

The reasons for defining two cyclic-prefix lengths for LTE are twofold:
A longer cyclic prefix, although less efficient from a cyclic-prefix-overhead point-of-view, may be beneficial in specific environments with very extensive delay spread, for example in very large cells. It is important to have in mind, though, that a longer cyclic prefix is not necessarily beneficial in case of large cells, even if the delay spread is very extensive in such cases. If, in large cells, link performance is limited by noise rather than by signal corruption due to residual time dispersion not covered by the cyclic prefix, the additional robustness to radio-channel time dispersion, due to the use of a longer cyclic prefix, may not justify the corresponding loss in terms of reduced received signal energy.

In case of MBSFN-based multicast/ broadcast transmission, the cyclic prefix should not only cover the main part of the actual channel time dispersion but also the timing difference between the transmissions received from the cells involved in the MBSFN transmission. In case of MBSFN operation, the extended cyclic prefix is therefore often needed. Thus, the main use of the extended cyclic prefix can be expected to be MBSFN-based transmission. It should be noted that different cyclic prefix lengths may be used for different subframes within a frame. As an example, MBSFN-based multicast/broadcast transmission is typically confined to certain subframes in which case the use of the extended cyclic prefix, with its associated additional cyclic-prefix overhead, may only be applied to these subframes.

Taking into account also the downlink time-domain structure, the resource blocks mentioned above consist of 12 subcarriers during a 0.5 ms slot, as illustrated in Figure 22. Each resource block thus consists of 84 resource elements in case of normal cyclic prefix and 72 resource elements in case of extended cyclic prefix.

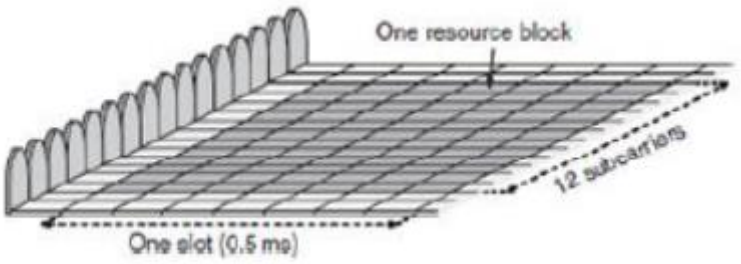


Figure 5-0-22:downlink resource block assuming normal cyclic prefix (i.e. 7 OFDM symbols per slot). with extended cyclic prefix there are six OFDM symbols per slot.

Although resource blocks are defined over one slot, the basic time-domain unit for dynamic scheduling in LTE is one subframe, consisting of two consecutive slots. The reason to define the resource blocks over one slot is that distributed downlink transmission is defined on a slot basis. The minimum scheduling unit consisting of two resource blocks within one subframe (one resource block per slot) is sometimes referred to as a resource block pair .

Appendix: A mat-lab code

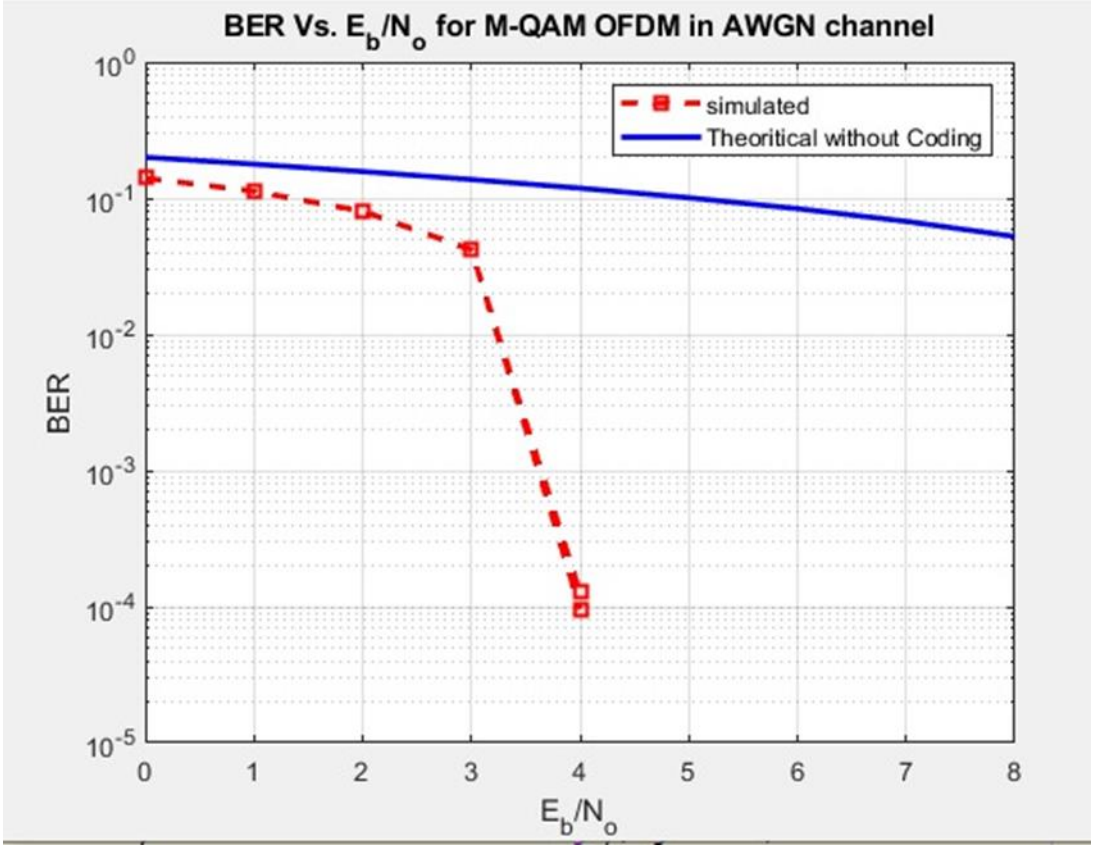
LTE. Downlink

```
% Graduation Project
% Downlink Physical Layer
% edit by LTE Team
% Setup
% Define parameters.
clear all; clc;
nFFT = 256; % fft size
nDSC = nFFT; % number of data subcarriers
CP = 1/4;
CP_start = (nFFT-(nFFT*CP))+1;
CP_end = (nFFT*CP)+1;
Total_Lingth = (nFFT*CP)+nFFT;
M = 64; % Size of signal constellation
k = log2(M); % Number of bits per symbol
n_FRM = 300;
FRM = 6144;
FRM = FRM-24;
K1 = FRM+24;
codeRate = 1/3;
f1=263; f2=480; inx=0:(K1-1);
Indices = rem((f1*inx)+(f2*(inx.^2)),K1)+1;
Trellis = poly2trellis(4, [13 15], 13);
TurboEnc = comm.TurboEncoder('TrellisStructure',Trellis,'InterleaverIndices',Indices);
TurboDec = comm.TurboDecoder('TrellisStructure',Trellis,'InterleaverIndices',Indices,'NumIterations',6);
zm = comm.RectangularQAMModulator(M, 'SymbolMapping', 'Gray', 'BitInput',
true,'NormalizationMethod','Average power','OutputDataType','double');
zd = comm.RectangularQAMDemodulator(M, 'SymbolMapping', 'Gray', 'BitOutput',
true,'NormalizationMethod','Average power','DecisionMethod','Log-likelihood ratio','VarianceSource','Input
port');
CRC_Tx = comm.CRCGenerator('Polynomial',[1 1 zeros(1, 16) 1 1 0 0 0 1 1]);
CRC_Rx = comm.CRCDetector('Polynomial', [1 1 zeros(1, 16) 1 1 0 0 0 1 1]);
if M==2
EbNo = [0:8];
elseif M==4
EbNo = [0:8];
elseif M==16
EbNo = [0:8];
elseif M==64
EbNo = [0:8];
elseif M==256
EbNo = [0:8];
end
bit_error_rate=[];
resource_block=12;
No_Code_Blocks=21;
for ii = 1:length(EbNo)
    ii
 Total_number_of_errors=0;
 for jj=1:n_FRM
%% Signal Source
t_data = randi([0 1], FRM,1); % Random binary data stream
%% CRC
t_data_crc = step(CRC_Tx,t_data);
%% Turbo Coding
```

```
cod_data=step(TurboEnc,t_data_crc);
%% Interleaving
D = K1 +4;
cod_data_Interleaver = reshape(cod_data,3,D);
colTcSb = 32;
rowTcSb = ceil(D/colTcSb);
Kpi = colTcSb * rowTcSb;
Nd = Kpi - D;
d =(1:D)';
colPermPat =[0,16,8,24,4,20,12,28,2,18,10,26,6,22,14,30,1,17,9,25,5,21,13,29,3,19,11,27,7,23,15,31];
y = [NaN*ones(Nd, 1); d];
inpMat = reshape(y, colTcSb, rowTcSb).';
permInpMat = inpMat(:, colPermPat+1);
Index1 = permInpMat(:);
% For 3rd stream only
pai = zeros(colTcSb*rowTcSb, 1);
for i = 1 : length(pai)
pai(i) = colPermPat(floor((i-1)/rowTcSb)+1) + colTcSb*(mod(i-1, rowTcSb)) + 1;
end
ytemp = inpMat.';
y2 = ytemp(:);
Index2 = y2(pai);
 for inv=1:3
 if inv==3
 Index=Index2;
 else
 Index=Index1;
 end
 IndexG=find(~isnan(Index)==1);
 IndexB=find(isnan(Index)==1);
 cod_data_Interleaver_temp=cod_data_Interleaver(inv,:);
 cod_data_Interleaver_temp1=zeros(size(Index));

cod_data_Interleaver_temp1(IndexG)=cod_data_Interleaver_temp(Index(IndexG));
 Nd=numel(IndexB);
 cod_data_Interleaver_temp1(IndexB)=nan*ones(Nd,1);

cod_data_Interleaver_temp2=cod_data_Interleaver_temp1(isfinite(cod_data_Interleaver_temp1));
cod_data_Interleaver_temp3(inv,:)= cod_data_Interleaver_temp2.';
 end
Interleaved_data =reshape(cod_data_Interleaver_temp3,D*3,1); % Buffer
%% Rate Matching
if codeRate==(1/3); ek_Length=length(Interleaved_data); else
ek_Length=k*ceil((D/k)/codeRate);end
Interleaved_data_matched=Interleaved_data(1:ek_Length);
%% Modulation
mod_data = step(zm, Interleaved_data_matched);
%% Data Mapping
allocated_resource_block=No_Code_Blocks*resource_block;
no_OFDM_Symbol=ceil((ek_Length/k)/allocated_resource_block);
Dummy_zeros= zeros((allocated_resource_block*no_OFDM_Symbol)-(ek_Length/k),1);
All_Data=[mod_data;Dummy_zeros];
%% serial to parallel conversion
par_data = reshape(All_Data,allocated_resource_block,no_OFDM_Symbol).';
clear mod_data;
%% DFT-precoding.
par_data_freq = fft(par_data,[],2);
%% Subcarrier mapping.
par_data_freq_LFDMA = zeros(no_OFDM_Symbol,nDSC);
subband=0;
```

```
par_data_freq_LFDMA(:,[1:allocated_resource_block]+(allocated_resource_block*subband)) = par_data_freq;
% Localized mapping.
%% fourier transform time doamain data and normalizing the data
IFFT_data = (nFFT/sqrt(nDSC))*ifft(par_data_freq_LFDMA,nFFT,2);
clear par_data;
%% addition cyclic prefix
cylic_add_data = [IFFT_data(:,[CP_start:nFFT]) IFFT_data];
clear IFFT_data;
%% parallel to serial coversion
ser_dataIN = reshape(cylic_add_data.',(Total_Lingth)*no_OFDM_Symbol,1);
clear cylic_add_data;
%% AWGN Channel
snr(ii) = EbNo(ii)+(10*log10(k))+ 10*log10(codeRate)-10*log10(nDSC/(allocated_resource_block));
chan_awgn = awgn(ser_dataIN,snr(ii),'measured');
clear ser_dataIN;
%% serial to parallel coversion
para_dataOUT= reshape(chan_awgn,Total_Lingth,no_OFDM_Symbol).';
%% removing cyclic prefix
cyclic_pre_rem = para_dataOUT(:,[CP_end:(nFFT+(nFFT*CP))]);
clear para_dataOUT;
%% converting to frequency domain
FFT_recdata= (sqrt(nDSC)/nFFT)*fft(cyclic_pre_rem,nFFT,2);
clear cyclic_pre_rem;
%% Subcarrier de-mapping.
FFT_recdata_LFDMA = FFT_recdata(:,[1:allocated_resource_block]+(allocated_resource_block*subband));
%% DFT-Despreading
FFT_recdata_time = ifft(FFT_recdata_LFDMA,[],2);
%% parallel to serial coversion
ser_data= reshape(FFT_recdata_time.',allocated_resource_block*no_OFDM_Symbol,1);
clear FFT_recdata;
%% Data de-mapping.
ser_data_demapping = ser_data(1:(ek_Length/k),:);%% Subcarrier de_mapping.
%% Demodulation
demod_Data = step(zd, ser_data_demapping,10^(-snr(ii)/10));
%% Rate Dematching
demod_Data_Dematched=zeros(3*D,1);
demod_Data_Dematched(1:numel(demod_Data))=demod_Data;
%% De-Interleaving
R_data_Denterleaver = reshape(demod_Data_Dematched,3,D); % Buffer
 for dnv=1:3
if dnv==3
 Index=Index2;
 else
 Index=Index1;
 end
 IndexG=find(~isnan(Index)==1);
 IndexOut=Index(IndexG);
 R_data_Denterleaver_temp=(R_data_Denterleaver(dnv,:));

R_data_Denterleaver_temp2=zeros(size(R_data_Denterleaver_temp));
 R_data_Denterleaver_temp2(IndexOut)=R_data_Denterleaver_temp;
 R_data_Denterleaver_temp3(dnv,:)= R_data_Denterleaver_temp2.';
 end
DeInterleaved_data =reshape(R_data_Denterleaver_temp3,D*3,1);
%% Turbo - Decoding
R_data=step(TurboDec,-DeInterleaved_data);
%% CRC
R_data_crc = step(CRC_Rx,R_data);
%% BER Computation
[number_of_errors,bit_error_rate_temp] = biterr(t_data,R_data_crc);
```

```
clear t_data ; clear demod_Data ;
Total_number_of_errors=Total_number_of_errors+number_of_errors;
 end
 bit_error_rate(ii)=Total_number_of_errors/(FRM*n_FRM);
 clear Total_number_of_errors
 bit_error_rate
end
%% ploating
semilogy(EbNo,bit_error_rate,'--sr','linewidth',2);
hold on;
if M==2 % Theoretical BER for
BPSK
BER_theory = berawgn(EbNo,'psk',M,'nondiff')
else % Theoretical BER for QAM
BER_theory = berawgn(EbNo,'qam',M)
end
semilogy(EbNo,BER_theory,'-b','linewidth',2);
legend('Theoretical without Coding'); grid on;
xlabel('E_b/N_o'); ylabel('BER')
title('BER Vs. E_b/N_o for M-QAM OFDM in AWGN channel');
axis([0 EbNo(end) 10^-5 1]);
```



Appendix: B Software Implementation

This design is defined as software implementation of LTE. Physical layer transceiver with all stages in details. it verifies the idea of our project, and show briefly output of OFDM symbol troughing in SISO-Fading channel and AWGN (Ideal – Channel) effect.

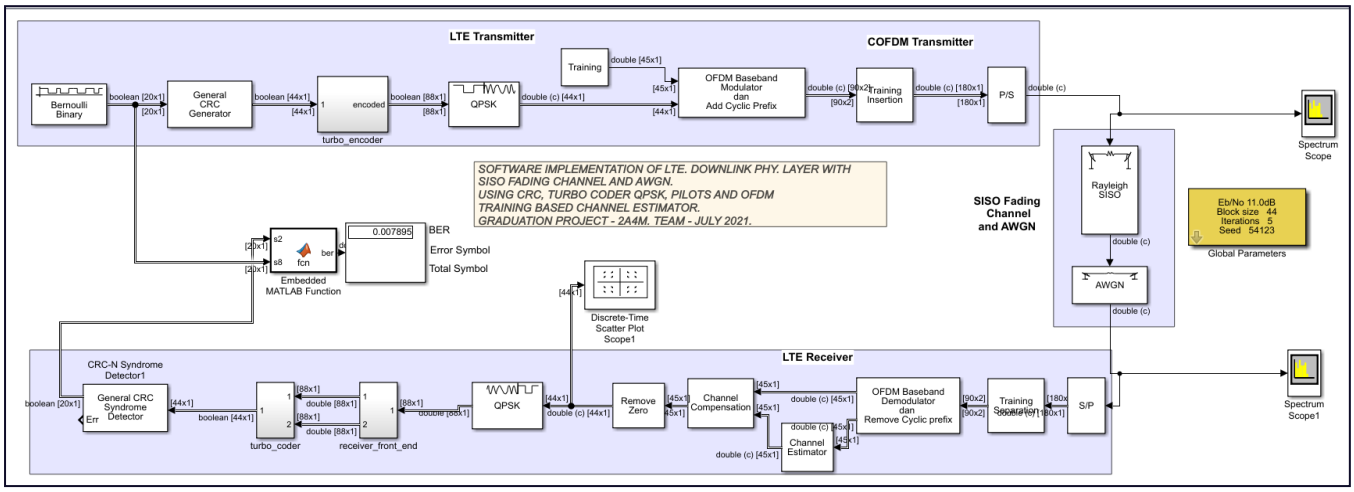
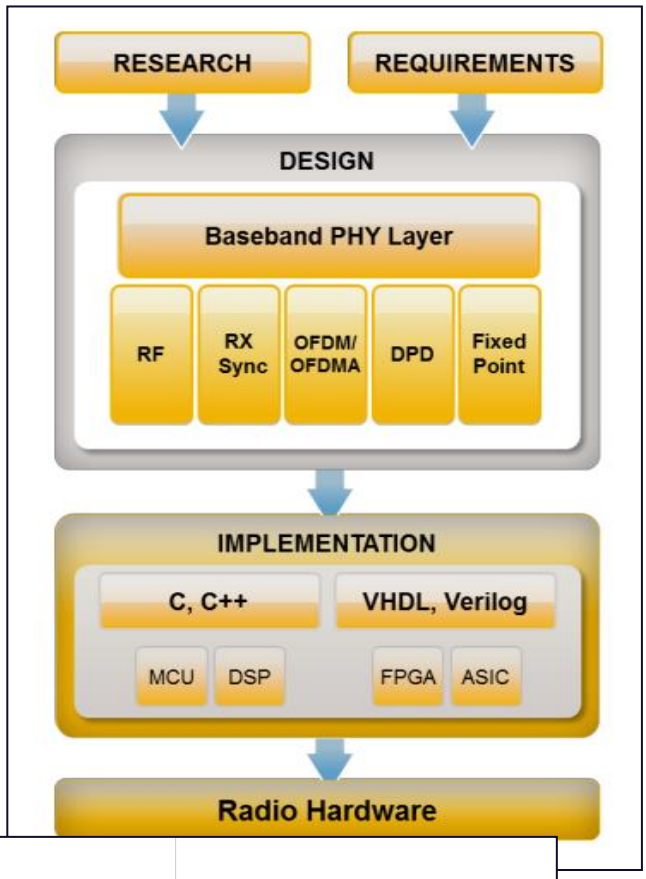


Figure B-0-1:"Full-Blocks and Subsystems of LTE. DL. Physical Layer"

ideal for LTE algorithm and system design MATLAB and Simulink provide an environment for dynamic & large-scale simulations Accelerate simulation with a variety of options in MATLAB Connect system design to implementation with

– C and HDL code generation
– Hardware-in-the-loop verification

Turbo stages are running with parameters file and uploading with workspace.




```
function turbo_code_punc_setup
% Get parameter names and values from mask
mask_ws_vars = get_param([gcs '/Global Parameters'],'maskwsvariables');

if ~isempty(mask_ws_vars)
    for i = 1:length(mask_ws_vars),
        curr_var = mask_ws_vars(i).Name;
        evalin('base',[curr_var ' = ' num2str(mask_ws_vars(i).Value) ';']);
    end

    % Set up other parameters in the MATLAB workspace as needed
    evalin('base', 'trellis = poly2trellis(5, [37 21],37);');  % code rate 1/2
    evalin('base', 'code_rate = 1/3;');                % Overall code rate = 1/3
    evalin('base', 'Ps = 1;');                         % Signal power is 1
    evalin('base', 'EbNo = 10.0.^(0.1*EbNodB);');      % Convert from dB to linear EbNo
    evalin('base', 'EsNo = EbNo/code_rate;')           %
    evalin('base', 'multiplier = 1/code_rate;')        % multiplier = symbol_period/sample_time
    evalin('base', 'Variance = Ps*multiplier/EsNo;'); % Calculate channel noise variance. See Help of AWGN
    evalin('base', 'clear code_rate Ps EbNo EsNo multiplier;');

else
    evalin('base','Len = 44;');
    evalin('base','Iter = 11;');
end
```

The main focus of our study is to measure the performance of LTE uplink and downlink physical layer based on Release 8 & 9. First and foremost, studies made on issues related to LTE and the fundamental of LTE need to understand. Next, the related simulator needs to find to running the simulation and the suitable simulator use to obtain the result is MATLAB. The purpose of choosing

MATLAB simulator is because it is widely used in data analysis. Furthermore, the result can be obtained by running a program and setting the parameter in the simulator and the comparison of OFDM performance can be measured and analyzed with different modulation techniques and different channel models. Simulink is a graphical extension to MATLAB for the modeling and simulation of systems. In Simulink, systems are drawn on screen as block diagrams. Many elements of block diagrams are available (such as transfer functions, summing junctions, etc.), as well as virtual input devices and output devices. Simulink is integrated with MATLAB and data can be easily transferred between the programs.

Simulink model consists of transmitter and receiver side. Transmitter section first block is Data source, its Contains random integer input and integer to binary bit converter, this integer is converted into binary bits then its gives input as a IQ Mapper, this IQ Mapper block contains binary bit into integer converter, QAM modulation, Math function ,here IQ mapper performs modulation for given inputs then its followed to OFDM Modulation its converts given input as 24 subcarrier then follows perform FFT and Add cyclic prefix after its goes to receiver section its perform operation of transmitter side.

After that each block output are taken from use of Go to tag then it's given to input of error rate calculation use of from tag.Similarly, between transmitter and receiver use different channel model and analyze the performance.

From the Simulink results, the packet loss and bit loss are zero for AWGN channel but its line of sight. By using Non line of sight channel Rayleigh and Rician there is some packet loss and bit loss. When compared to Rayleigh channel,

Rician channel performs better because it gives only 0.2618 for QPSK modulation.

To develop products that conform to the LTE and LTE-Advanced standards, it's important to understand the structure of LTE signals, physical layer algorithms such as OFDM and spatial multiplexing, and strategies for simulating and testing your product design. LTE tutorial videos, articles, and code examples introduce key LTE concepts and effective techniques and workflows for LTE physical layer development including:

1) Waveform generation
2) End-to-end simulation and conformance testing
3) Design verification
4) Signal analysis

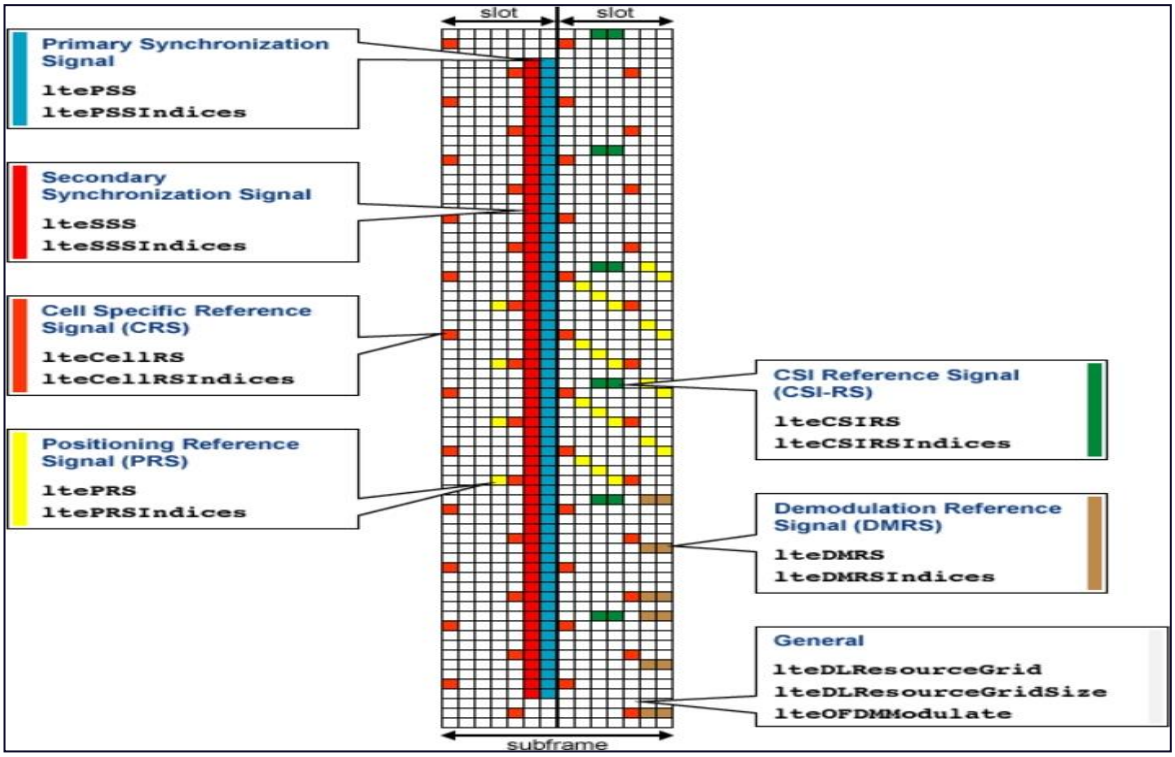
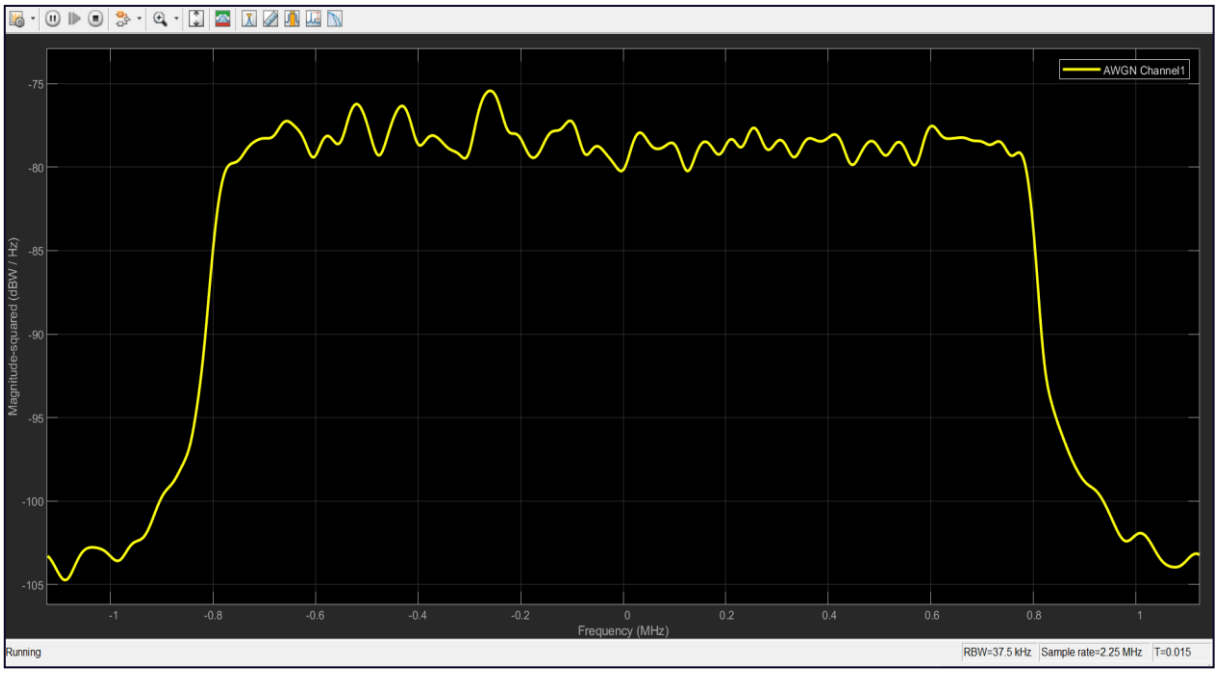
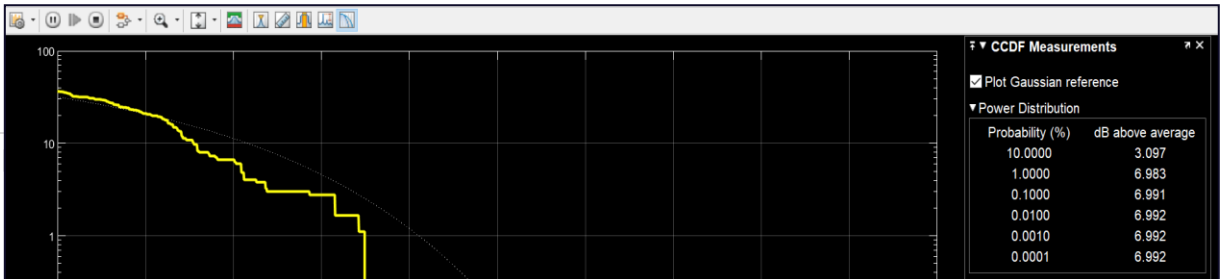
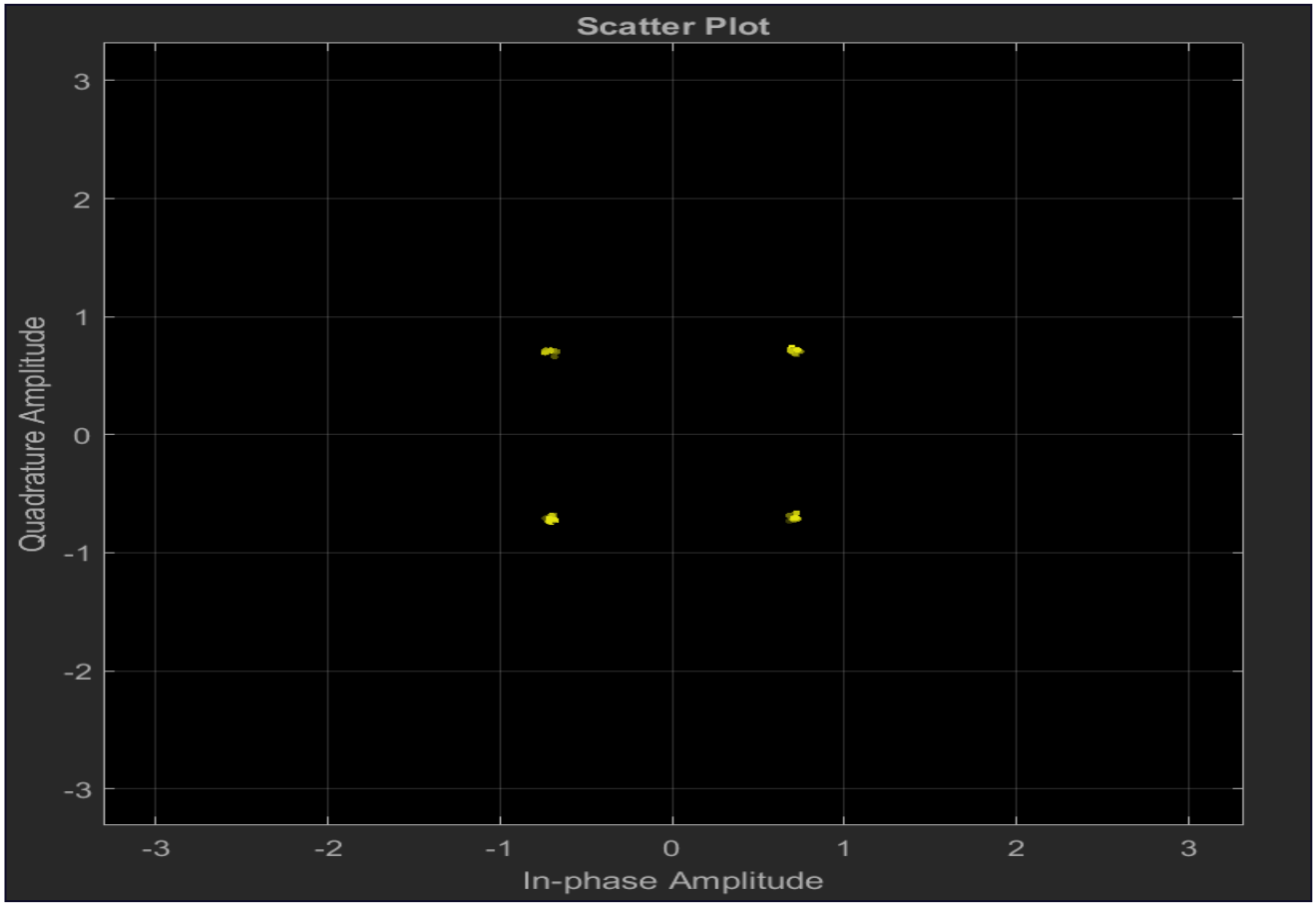


Figure B-0-2:
Frame structure of the LTE resource grid.

Results according to measurements connected:






Scatter Plot
Quadrature Amplitude
In-phase Amplitude
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

We can change any parameters or connect scopes and BER counter at any stage of these system, this design is causal and reliable.

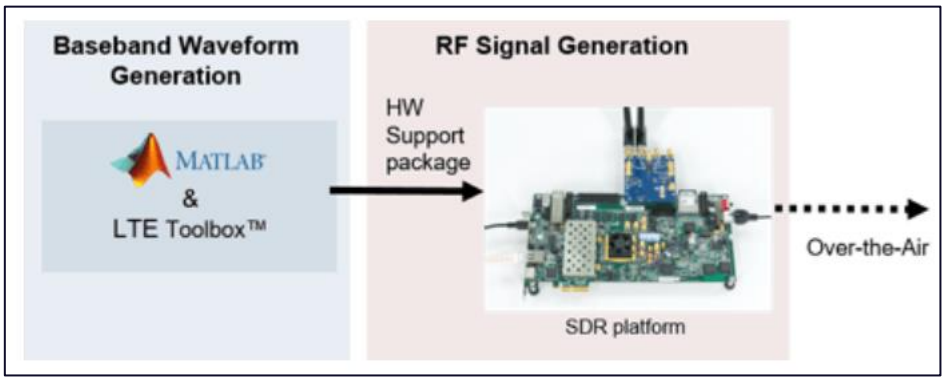
Future Work

The main goal of our project is making air – interface of 4G. using FPGA – Xilinx – ZYNQ and transmit with SISO – MODE of Images.
We made the transmitter with correct processing, but the receiver needs to kit connected with SDR – Platform and Antenna Port.
The next words showing that processing of TX and RX using Built-in ZYNQ Library**.**

LTE Transmitter using Zynq-based Software-Defined Radio (SDR):

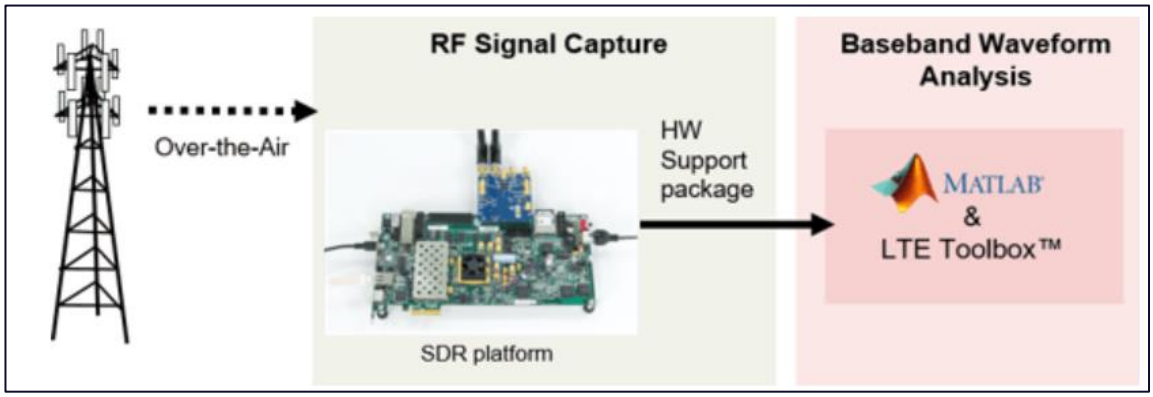
LTE Toolbox can be used to generate standard-compliant baseband IQ downlink and uplink reference measurement channel (RMC) waveforms and downlink test model (E-TM) waveforms. These baseband waveforms can be modulated for RF transmission using SDR radio hardware such as Xilinx Zynq-based radio.

In this example eight frames of a baseband RMC waveform are generated using the LTE Toolbox. A continuous RF LTE waveform is created by looping transmission of these eight frames with the Zynq radio hardware for a user-specified time period.



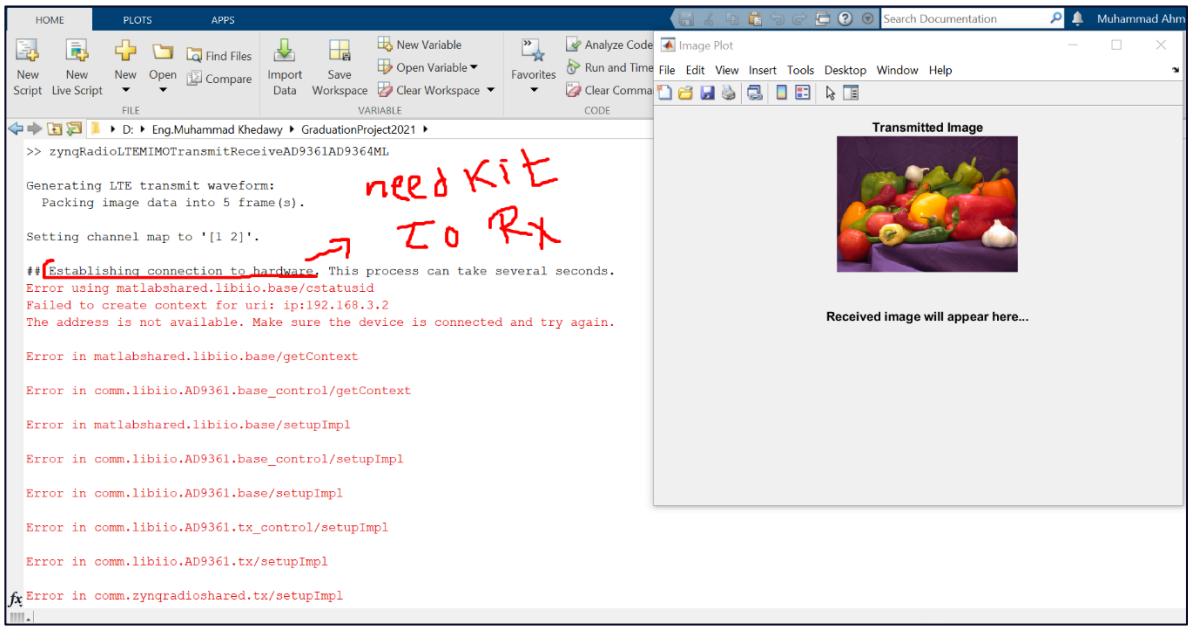
LTE Receiver Using Zynq-based Software-Defined Radio (SDR):

LTE Toolbox provides functions and tools to decode LTE waveforms. This example shows how an LTE waveform can be captured with SDR radio hardware such as Xilinx Zynq-based radio, and be decoded to recover basic system information.



This example decodes the MIB for a burst of captured frames, and then decodes the control format indicator (CFI) for each subframe, which informs the user equipment (UE) of the size of the control region. The physical downlink control channel (PDCCH) is then decoded.

The example outputs the detected cell identity as well as the decoded CFI value. Plots for the spectrum of the captured signal, the PDCCH constellation diagram and the channel estimate are also produced.



References

[1] Ericsson, ''LTE-an introduction'', White Paper, June 2009.

[2] 3GPP TS36.300, ''Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description''.

[3] Hyung G. Myung and David J.Goodman, ''Single Carrier FDMA, New Air Interface for LTE'', Wiley, 2008.

[4] S. Schwarz, C. Mehlfuhrer and M. Rupp, ''Low Complexity Approximate Maximum Throughput Scheduling for LTE'', 44th Annual Asilomar conference on Signals, Systems and Computers, California, 2010.

[5] Andrea Goldsmith, ''Wireless Communications'', Cambridge University Press 2005, pp.374-398.

[6] Jim Zyren, ''Overview of the 3GPP Long Term Evolution Physical Layer'',freescale semiconductor, July 2007.

[7] S. Parkvall and D. Astely, ''The Evolution of LTE towards IMT-Advanced'',Journal of Communications vol.4, N0.3, 2009.

[8] E.Dahlman, S.Parkvall, J.Skold, P.Beming, ''3G Evolution HSPA and LTE for Mobile Broadband'', Elsevier, 2008.

[9] Motorola, ''Long Term Evolution (LTE): overview of LTE Air –Interface'',Technical White Paper, January 2008.

[10] Farooq Khan, 'LTE for 4G Mobile Broadband, Air Interface Technologies and Performance, Cambridge University Press, 2009'.

[11] C.E. Shannon, ''A Mathematical Theory of Communication'', Bell System Technical Journal, Vol.27, July and October 1948, pp.379-423,623-656.

[12] C. Mehlfuhrer, M. Wrulich, J.C Ikuno, D.Bosanska, and M. Rupp, ''Simulating the long term evolution physical layer, '' in Proc. of the 17th European Signal processing conference (EUSIPCO 2009), Glasgow, Scotland, Aug. 2009.

[13] Using HDL Optimized CRC Library Blocks - Mathwork.

[14] ‘Two Decoding Algorithms for Tailbiting Codes “ by Rose Y. Shao, Member,
IEEE, Shu Lin, Life Fellow, IEEE, and Marc P. C. Fossorier, Senior Member,IEEE in 2003.

[15] Peterson, W. W. and Brown, D.T. “Cyclic Codes for Error Detection.” In Proceedings of the IRE, January 1961, 228–235.

[16] “CRC Implementation With MSP430 Application” Report SLAA221 November 2004, Texas Instrument.

[17] Joachim Hagenauer, Joachim; et al. "Iterative Decoding of Binary Block and Convolutional Codes" *(PDF)*. Archived from the original *(PDF)* on 11 June 2013. *Retrieved 20 March 2014*.

[18] LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 14.2.0 Release 14)

[19] Cheng, Jung-Fu, et al. "Analysis of circular buffer rate matching for LTE turbo code." *2008 IEEE 68th Vehicular Technology Conference*. IEEE, 2008.

جامعة الأزهر
كليـــة الهندســـة
قســـم الاتصـــالات و الإلكترونيـــات

دراسة وتنفيذ برنامج الطبقة الفيزيائية لشبكات الجيل الرابع .

إعـــــــداد:

محـمد مجـدى سليمـان حمـده
محـمد سليمـان السيـد سليمـان
محـمد سعيـد محـمد عثمـان

محمد احمد محمد احمد الخديوي
احمـد محمد عبد الرحمـن علـوان
احمد حمدي عبد الفتاح صابر العش

إشـــــراف
د. احمـد عبـدالرحمـن عمـران