5G Korea-
Link Level Simulator v2.1
User Manual
30. Dec. 2018.

Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea,
Chungnam National Univ., Daejeon, Korea,
Hanyang Univ.(ERICA), Ansan, Gyeonggi, Korea,
Hankyong National Univ., Anseong, Gyeonggi, Korea,
Korea Univ., Seoul, Korea,
Dongguk Univ., Seoul, Korea,
Seoul National Univ., Seoul, Korea,
Yonsei Univ. Seoul, Korea,

Web: http://5gopenplatform.org

Abstract

This document contains an overall description of 5G K-SimLink (5G-Korea link level simulator) and how to use it for users. There are list of features of 5G K-SimLink, a structure, and parameter descriptions in this document. 5G K-SimLink has been developed with a modular and flexible architecture, users can easily adjust the simulator to evaluate their algorithms or functions. This simulator is released under an academic, non-commercial use license.

Contents

1. Introduction

This document describes the structure and function of SU-SISO, SU / MU-MIMO based simulator, its main functions and requirements, the definition of functional blocks and modules, functional blocks and modules correlation, functional block and module calibration.

1.1 Purpose

This document defines the interrelationships, functional blocks and modules required for link level simulator development for effective interoperability of link level / system level / network level for system performance verification in 5G standard model development project/

1.2 License

TBD

1.3 Contact points

Jimin Bae

Korea Advanced Institute of Science and Technology (KAIST)

335 Gwahangno, Yuseong-gu, Daejeon, 305-701, Republic of Korea

Office: Room 717, IT Convergence Building (N1)

Tel: +82-42-350-5472

E-mail: jimin1203@kaist.ac.kr

1.4 Development environment

- OS: Windows 10
- Language: Matlab
- Compiler: Matlab 2017a

1.5 Download

http://5gopenplatform.org

1.6 Definition, symbols, and abbreviations

1.6.1 Definition

Word	Definitions

1.6.2 Symbol

Symbols	Descriptions
μ	Numerology parameter

1.6.3 Abbreviations

Abbreviations	Descriptions
BER	Bit error rate
BLER	Block error rate
CDL	Clustered Delay Line
CLSM	Closed-loop spatial multiplexing
CQI	Channel quality indicator
CRC	Cyclic redundancy check
CSI-RS	Channel-state information reference signal
CSI-IM	Channel-state information interference measurement
DM-RS	Demodulation reference signal
H-ARQ	Hybrid automatic repeat request
LDPC	Low-density parity-check code
MBSFN	Multicast-broadcast single-frequency network
MIMO	Multiple-input multiple-output
MU-MIMO	Multi-user multiple-input multiple-output
OCC	Orthogonal cover codes

OFDMA	Orthogonal frequency division multiple access
OLSM	Open-loop spatial multiplexing
OOP	object-oriented programming
PAPR	Peak to average power ratio
PBCH	Physical broadcast channel
PDCCH	Physical downlink control channel
PDP	Power-Delay Profile
PDSCH	Physical downlink shared channel
QAM	Quadrature amplitude modulation
RI	Rank indicator
PBCH	Physical broadcasting channel
PDCCH	Physical downlink control channel
PDSCH	Physical downlink shared channel
PMI	Precoding matrix indicator
RS	Reference signal
PSS	Primary synchronization signal
SC-FDMA	Single-carrier-Frequency division multiple access
SSS	Secondary synchronization signal
SU-MIMO	Single-user multiple-input multiple-output
TDD	Time division duplexing
TDL	Tapped Delay Line

2. Related works

2.1 Standardization

- **SU-SISO (Single-user Single Input & Single Output)**
  - Release 15 introduces various numerology depending on subcarrier spacing.
  - Release 15 adopts CP-OFDM as the basic waveform.
  - Release 15 introduces LDPC for channel coding especially for data channels.
- **SU-MIMO (Single-user Multi Input & Multi Output)**
  - Release 10 defines a non-codebook-based transmission mode 8 that can support up to 8 layers using DM-RS and CSI-RS.
  - Massive MIMO up to 16 layers is introduced in Release 13.
  - Massive MIMO up to 32 layers is introduced in Release 14.

- 5G massive MIMO for 64 or more layers is introduced in Release 15.

2.2 Trends in Korea

- **Link-level simulator:**
  - Currently, open simulator development through open-source does not exist in Korea.
  - As the closed simulator, Korea Electronics and Telecommunications Research Institute is developing the system that has a 1GHz bandwidth with the center frequency of millimeter-wave band 28 GHz. The base station has a maximum transmission rate of 20 Gbps, and the terminal has a maximum transmission rate of 1.5 Gbps considering the cell radius of 300m with 64QAM modulation for downlink and 16QAM modulation for uplink.
  - In addition, Samsung has developed a system that has 800MHz bandwidth with a baseband beam-width of 10 degrees, and a terminal having a beam-width as 140 degrees in the vertical direction and 20 degrees in the horizontal direction in the millimeter wave 28 GHz frequency band. In October 2014, the system recorded a transmission rate of 1.2Gbps at a moving speed of 100km / h or more in October 2014, and a maximum transmission rate of 7.5Gbps when stopped.

2.3 Overseas trends

- **Link-level simulator**:
  - As an open simulator, Vienna LTE-A downlink link level simulator was developed by the Vienna University of Technology in Austria. The simulator considers AWGN, flat Rayleigh fading, Winner Phase II+ as the channel environment, and the performance of BLER and throughput can be verified in the LTE system considering various channel estimation methods by using MIMO transmission mode such as TxD (Tx diversity), OLSM (open-loop spatial multiplexing), CLSM (closed-loop spatial multiplexing). It will continue to be supplemented and revised until 2016-Q2, and in March 2016, a book on simulators were published under the title "The Vienna LTE-Advanced Simulators: Up and Downlink, Link and System Level Simulation".
  - Vienna LTE-A downlink link level simulator is extended to be suite and to evolve to the next generation of mobile communication by introduction new 5G simulators, named Vienna 5G Link Level Simulator. The Vienna 5G Link Level Simulator is the newest member of the family of Vienna cellular Communications Simulators. Although, there exists no definite 5G specification, there are several hot candidates for physical layer waveforms and channel coding schemes. This simulator is expected to offer a unifying platform for performance

evaluation as well as co-existence investigation of candidate 5G physical layer schemes. This simulator provides flexibility by supporting a broad range of simulation parameters.

- As typical closed simulators, there are LTE / LTE-A based link-level simulators called LTE PHY Lab developed by IS-Wireless, a Polish telecommunication company. It includes various LTE/LTE-A features such as OFDMA / SC-FDMA, PARP reduction, channel coding / decoding, MIMO receiver, time and frequency synchronization, channel estimation, and baseband model of base station and terminal (transmission / reception, uplink / downlink, TDD / FDD). Also simulator generates the reference waveforms of all PHY channels and signals.
- As a closed simulator, in order to meet the requirements of Nokia 5G, there is an experiment with the transmission rate using 16QAM in a 200m cell radius using a frequency of 73GHz to 1GHz with a 3 degree antenna beam width and a horizontal 34 degree and 8 degree vertical beam. Ericsson tested 5Gbps data rate using 4 streams in 400MHz bandwidth of 15GHz operating frequency considering 10Gbps target system. NTT DoCoMo experimented with 2Gbps data transmission through beamforming technique with beam tracking according to the position of mobile device considering the millimeter wave system of 70GHz band for 5G commercialization and in cooperation with Samsung, they achieved 2.5Gbps reception rate at 60km speed with 28GHz frequency band signal. Also Qualcomm considered OFDM-based systems at sub-6GHz for the 5G NR (New Radio) system, and also considered beamforming and beam steering systems in non-line-of-sight conditions in the millimeter wave 28GHz band.

2.4 Differentiations from existing simulators

- Openness
  - This simulator is based on object-oriented programming (OOP) to improve speed, development possibility, and modularity and to build user interface and make user manual.
- Modularization:
  - This simulator is an open source software based on standard model that can modularize detailed functions according to standard specification and system model, and interoperate and integrate link level / system level / network simulator.
  - First, each module is implemented to flexibly support the required system parameters by reflecting the channel coding, modulation, layer mapping and precoding of the multi-antenna system, OFDMA in the transmitting end structure of the 5G system(Rel. 15).
  - Develop the detailed components by modularization. In particular, through modularization, by creating interworking module between link level / system level /network simulator, interworking and integration will be easy, and it will be possible to develop highly accurate simulator through module level

verification.

- Modularization makes the development of simulators to be applied to various environments. Specifically, in the single cell link level simulator, various users in a single cell require different quality of service. Accordingly, the base station has to enable the implementation and analysis of channel quality information (CQI) and traffic measurement based techniques to provide appropriate quality of service to different users.

3. Link level simulator structure

3.1 Overview

Here is the block diagram which represents modules in the simulator. The blue box is the module which is changed for NR. The Red box means added modules for NR.

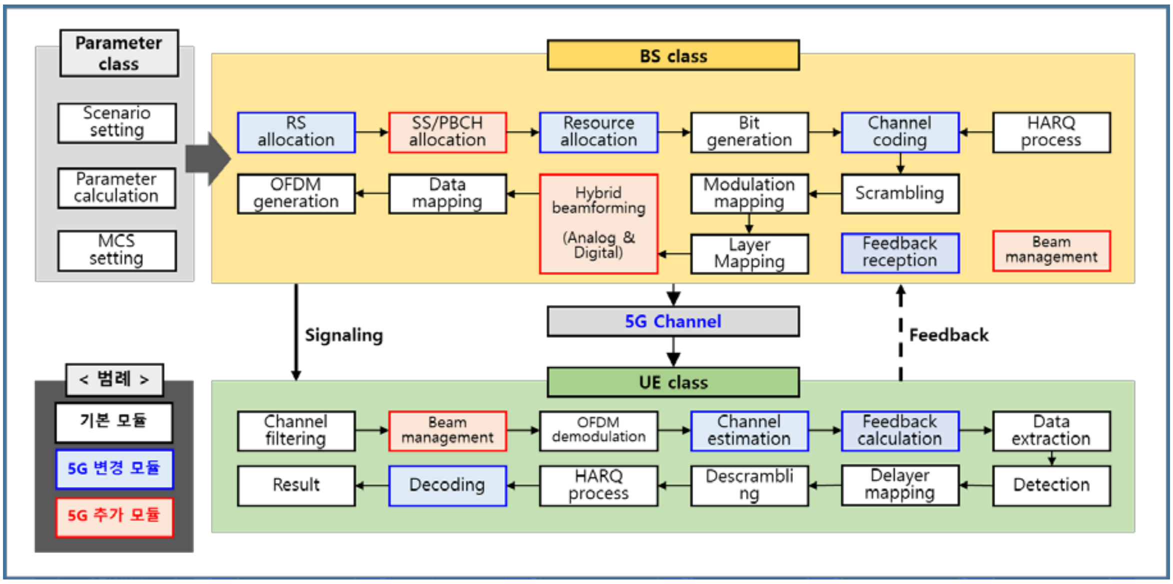

Figure 1 Block diagram for 5G K-SimLink

• Function Block: It is a practical unit that performs the functions defined in the subsystem and usually consists of one executable file. A function block contains one or more modules. However, functions commonly used in various functional blocks are defined as a common block (Utility Block), which is configured as a library type instead of an executable file type.

• Module: It is a S/W functional unit. When the classified functional units are large, they can be classified as sub-modules again. Each module is a collection of several units that support the function and can have different file names. Functions common to the various modules are defined as a common module (Utility Module).

(Reference) Functional Analysis Functional Decomposition and Allocation
Defining the top level functions required for system development and decomposing them into detailed functions to satisfy

them. Each detail function is assigned and interacted with the existing upper functional element. It also defines the interface with the external system. The S / W functional elements must be physically or logically assigned to the H / W components in the network in the operating environment.

3.2 Major techniques and requirements

- **SU/MU-MIMO**
  - For single user MIMO (SU-MIMO), the transmission is performed at the transmitting and receiving end, which have multiple antennas. Release 15 considers up to 32 × 32 MIMO for CSI-RS, 8×8 MIMO for DM-RS.
  - For multi-user MIMO (MU-MIMO), even with the same number of users and same number of Tx/Rx antenna, there can be different mode of antenna allocation. There can be several factors to be considered in MU-MIMO implementation as follows: the number of covered UEs, the number of used antenna for Tx and Rx, the receiver design, precoding algorithm, and so on, which are not concretely defined yet.
  -
- **3D Spatial channel model**
  - Previously, various 2D spatial channel models including large scale considering delay spread, angular spread, cluster, AOA / AOD, and pathloss and shadowing effects have been proposed in accordance with NLOS / LOS environment.
  - For a full-dimensional MIMO environment, one of the technologies considered in this simulator, a 3D spatial channel model including the azimuth direction as well as the elevation direction of the 2D antenna pattern is required.

- **Frame structure**
  - For multi-carrier transmission of 5G, new frame structure is provided in Release 15. In NR, new parameter 'numerology' is defined which determines subcarrier spacing in the resource grid. It is for generalization from subcarrier in 4G LTE. Numerology parameter 'μ' can be set from 0 to 5. $\mu = 0$ makes 15kHz subcarrier spacing as LTE. As mu added by 1, subcarrier spacing be twice, and symbol time length is reduced half.

- **Beam management**
  - Actually, beam management is not clearly defined in the standard. Thus, we refer some other documents. We search two steps. First, wide beam search, then, narrow beam search using CSI-RS. Wide beam is made by Woodward-Lawson method of antenna pattern synthesis. For narrow beam, DFT and its oversampled beam is used. DFT beam generation refers [4].

- **Channel model for LLS**

  - In NR, there is a channel model for LLS. Clustered Delay Line (CDL) and Tapped Delay Line (TDL). CDL is a channel model which has many clusters. Many parameters are given: delay, power, degrees (AoD: Angle Of Departure, ZoD: Zenith angle Of Departure, AoA: Angle Of Arrival, ZoA: Zenith angle Of Arrival) and other parameters. There are five scenarios in CDL model. CDL-A, CDL-B, and CDL-C have only Non-line-of-sight (NLOS). Whereas CDL-D and CDL-E for line-of-sight (LOS).

3.3 Design on the structure of simulator (Downlink and Uplink)

3.3.1 Parameter function block

3.3.1.1 Definitions of Parameter function block and its interface linked with other blocks

- **Definition of Parameter function block**
  - Sets the parameters used throughout the simulation.
- **Definition of the interface**
  - Simulation scenarios are interlocked with all functional blocks of BS, Channel, and UE. First, parameters are computed according to the simulation scenario and used for BS, Channel, and UE.

3.3.1.2 Definitions of modules in Parameter function block

- Determine the number of ports according to the simulation scenario and set the PDP according to the channel. Then calculate the parameters that depend on the scenario.
- Determine the number of slots and symbols per subframe, subcarrier spacing, bandwidth set and resource block set according to the numerology.

3.3.2-a BS_SUSISO function block for Downlink

3.3.2-a.1 Definitions of BS_SUSISO function block and its interface linked with other blocks

- **Definition of BS_SUSISO function block**
  - A function block for processing the BS function. It contains module for receiving feedback from the user, an RS generation and allocation module, a synchronization signal generation and allocation module, a HARQ process processing module, a random bit generation module, a channel coding module, a scrambling module, a power allocation module, an OFDM signal generation module, and the like.
- **Definition of the interface**
  - Parameter function block, UE_SISO function block, Channel SISO function block are

interlocked. First, the initial value is obtained through the parameter function block, and the following process is performed. The initial value includes information related to the number of resource blocks (RBs), OFDM subcarriers per RB, number of symbols per slot, OFDM FFT size, and antenna port. Thereafter, the OFDM signal generated after executing the internal module of the BS functional block is transmitted through the channel module and the control signal information is transmitted to the UE.

3.3.2-a.2 Definitions of modules in BS_SUSISO function block

- **Feedback reception module**
  - Module that receives feedback from the UE functional block and processes the CQI related information.
- **Reference signal allocation module**
  - Module that generates DMRS for channel estimation and allocates it according to TS 38.211.
- **Resource allocation module**
  - Module that allocates SS/PBCH and data according to resource grid.
- **Sync generation module**
  - Generate synchronization signal and allocate it to antenna port. All the PSS and SSS are created and assigned.
- **HARQ process module**
  - Module that stores and processes data for HARQ process.
- **Bit generation module**
  - Module that generates as many random bits as there are data bits to be allocated to the Resource grid.
- **Channel coding module**
  - Module that performs LDPC coding according to the coding rate of the generated data bit sequence.
- **Scrambling module**
  - Module that scrambles bit sequence in the way defined in standard [1] after channel coding.
- **Modulation mapping module**
  - Module that maps scrambled bit sequence to modulation symbol according to modulation order.
- **Data mapping module**
  - Module that assigns the modulated signal sequence to a position corresponding to the PDSCH of the physical antenna port.

- **Power allocation module**
  - Module that allocates power to reference signal and data signal mapped to antenna port.
- **OFDM generation module**
  - Module that converts the signals assigned to the resource grid into OFDM signals. Generates an OFDM signal depending on the fft size matching the Bandwidth.

3.3.2-b UE_SISO function block for Uplink

3.3.2-b.1 Definitions of UE_SISO function block and its interface linked with other blocks

- **Definition of UE_SISO function block**
  - A function block for processing the UE function. It contains module for receiving feedback from the BS, an RS generation and allocation module, resource allocation module, a HARQ process processing module, a random bit generation module, a channel coding module, a scrambling module, a modulation module, a transform precoding module, a data mapping module, a power allocation module, an OFDM signal generation module, and the like.
- **Definition of the interface**
  - Parameter function block, UE_SISO function block, Channel SISO function block are interlocked. First, the initial value is obtained through the parameter function block, and the following process is performed. The initial value includes information related to the number of resource blocks (RBs), OFDM subcarriers per RB, number of symbols per slot, OFDM FFT size, and antenna port. Thereafter, the OFDM signal generated after executing the internal module of the UE functional block is transmitted through the channel module and the control signal information is transmitted to the BS.

3.3.2-b.2 Definitions of modules in UE_SISO function block

- **Feedback reception module**
  - Module that receives feedback from the BS functional block and processes the CQI related information.
- **Reference signal allocation module**
  - Module that generates DMRS for channel estimation and allocates it according to TS 38.211.
- **Resource allocation module**
  - Module that allocates data according to resource grid.
- **HARQ process module**
  - Module that stores and processes data for HARQ process.

- **Bit generation module**
  - Module that generates as many random bits as there are data bits to be allocated to the Resource grid.
- **Channel coding module**
  - Module that performs LDPC coding according to the coding rate of the generated data bit sequence.
- **Scrambling module**
  - Module that scrambles bit sequence in the way defined in standard [1] after channel coding.
- **Modulation mapping module**
  - Module that maps scrambled bit sequence to modulation symbol according to modulation order.
- **Transform precoding module**
  - Module that maps modulated signal sequence to transform precoded modulated signal sequence if transform precoding is enabled.
- **Data mapping module**
  - Module that assigns the modulated signal sequence after transform precoding module to a position corresponding to the PUSCH of the physical antenna port.
- **Power allocation module**
  - Module that allocates power to reference signal and data signal mapped to antenna port.
- **OFDM generation module**
  - Module that converts the signals assigned to the resource grid into OFDM signals. Generates an OFDM signal depending on the fft size matching the Bandwidth.

3.3.3-a BS_MIMO function block for Downlink

3.3.3-a.1 Definitions of BS_SUMIMO function block and its interface linked with other blocks

- **Definition of BS_SUMIMO function block**
  - As functional block that processes BS function, it is a form in which a module used in the SU-MIMO is added to the BS_SUSISO functional block. It includes a module that extends the BS_SUSISO functional block module to a multi-antenna situation, and additionally includes functions such as layer mapping module and precoding module.

- **Definition of the interface**
  - Parameter function block, UE_MIMO function block, Channel MIMO function block are interlocked. The interface used in BS_SUSISO is extended to multi-antenna situations.

3.3.3-a.2 Definitions of modules in BS_SUMIMO function block

- **Feedback reception module**
  - A module that receives feedback from the UE functional block and processes the channel feedback related information for CQI related information, PMI and RI related information for SUMIMO. Also process the channel feedback related information for non-codebook case, and generates a precoder. It is assumed that feedback data has already been received from receiver by the uplink channel and we use that data in each module.
- **Beam management module**
  - Wide beam and narrow beam can be selected. SS / PBCH to search for widebeam and then consider narrow beam selection via CSI-RS. In addition, the role of the beam management block of UE is also part of this module.
- **RS allocation module**
  - A module that generates a reference signal and assigns it to an antenna port. It allocates a physical antenna port using CSI-RS and a DM-RS to a virtual antenna port.
- **Resource allocation module**
  - A module that generates a reference signal and assigns it to an antenna port. The SS/PBCH allocation block is also in this module. It allocates a physical antenna port using SS/PBCH to a virtual antenna port.
- **Data allocation module**
  - It is acting as data allocation in the resource allocation block. Sets the data position after storing the resource grid position.
- **Sync generation module**
  - Generate Synchronization signal and assign it to antenna port. All PSS and SSS are generated and assigned. It can be used for SS/PBCH allocation block.
- **HARQ process module**
  - Module that stores and processes data for HARQ process. Spatial multiplexing is considered and extended.
- **Bit generation module**
  - Module that generates as many random bits as there are data bits to be allocated to the Resource grid. For SUMIMO, the number of data bits is calculated considering spatial multiplexing and the number of transport blocks.
- **Channel coding module**
  - Module that generates turbo codes for each transport block considering additional transport blocks in BS_SUSISO.
- **Scrambling module**

- Module that scrambles bit sequence in the way defined in standard [1] after channel coding. Apply scrambling to each transport block.

- **Modulation mapping module**
  - Module that maps scrambled bit sequence to modulation symbol according to modulation order. For each transport block, mapping is performed using different modulation orders according to CQI.
- **Layer mapping module**
  - Module that maps modulated symbol sequences to each layer according to the number of transport blocks and the number of data streams for spatial multiplexing.
- **Precoding module**
  - The Hybrid beamforming block and the Data mapping block are executed in this module. module that precodes the data symbol sequence and the DM-RS mapped by layer and maps them to the positions assigned to the PDSCH and DM-RS of the physical antenna port, respectively. It replaces SUSISO's data_mapping module.
- **OFDM generation module**
  - Module that converts the signals assigned to the resource grid into OFDM signals. Generates an OFDM signal depending on the fft size matching the Bandwidth and Extensively adapt to multiple antenna situations

3.3.3-b UE_MIMO function block for Uplink

3.3.3-b.1 Definitions of UE_SUMIMO function block and its interface linked with other blocks

- **Definition of UE_MIMO function block**

  As functional block that processes UE function, it includes a module that extends the UE_SUSISO functional block module to a multi-antenna situation.

- **Definition of the interface**

- Parameter function block, UE_MIMO function block, Channel MIMO function block are interlocked. The interface used in BS_SUSISO is extended to multi-antenna situations.

3.3.3-b.2 Definitions of modules in UE_SUMIMO function block

- **Feedback reception module**
  - Module that receives feedback from the BS functional block and processes the CQI related information.
- **Reference signal allocation module**

- Module that generates DMRS for channel estimation and allocates it according to TS 38.211.

- **Resource allocation module**
  - Module that allocates data according to resource grid.
- **HARQ process module**
  - Module that stores and processes data for HARQ process.
- **Bit generation module**
  - Module that generates as many random bits as there are data bits to be allocated to the Resource grid.
- **Channel coding module**
  - Module that performs LDPC coding according to the coding rate of the generated data bit sequence.
- **Scrambling module**
  - Module that scrambles bit sequence in the way defined in standard [1] after channel coding.
- **Modulation mapping module**
  - Module that maps scrambled bit sequence to modulation symbol according to modulation order.
- **Transform precoding module**
  - Module that maps modulated signal sequence to transform precoded modulated signal sequence if transform precoding is enabled.
- **Data mapping module**
  - Module that assigns the modulated signal sequence after transform precoding module to a position corresponding to the PUSCH of the physical antenna port.
- **Power allocation module**
  - Module that allocates power to reference signal and data signal mapped to antenna port.
- **Precoding module**
  - The beamforming block are executed in this module. It is assumed that the precoder is already determined prior transmission procedure. Thus the determined precoder is used for uplink.
- **OFDM generation module**
  - Module that converts the signals assigned to the resource grid into OFDM signals. Generates an OFDM signal depending on the fft size matching the Bandwidth.

3.3.4 Channel_5G function block

3.3.4.1 Definitions of Channel_5G function block and its interface linked with other blocks

- **Definition of Channel_5G function block**
  - Channel_5G generates channel coefficient matrix in channel according to the kinds of the channel.
- **Definition of the interface**
  - Channel_5G is linked with Parameter function block. It defines and calculates the needed values using the predefined values in Parameter function block. Also, it generates different result according to whether the channel is block fading or fast fading, whether 'time_correlation' is correlated or independent, whether antenna pattern is 'Omni-directional' or 'Pattern'.

3.3.4.2 Definitions of modules in Channel_5G function block

- **Chan_Matrix module**
  - Chan_Matrix generates channel coefficient matrix according to the interpolation method.
- **Chan_H_fft module**
  - Chan_H_fft generates the value of the channel w.r.t. frequency applying Fast Fourier Transform.

3.3.5 ChannelOutput_CDL function block

3.3.5.1 Definitions of ChannelOutput_CDL function block and its interface linked with other blocks

- **Definition of ChannelOutput_CDL function block**
  - ChannelOutput_CDL generates CDL channel used in 5G.
- **Definition of the interface**
  - ChannelOutput_CDL is linked with Channel _5G function block. CDL channel is generated from ChannelOutput_CDL to be used for Channel_5G function block.

3.3.6-a UE_SISO function block for Downlink

3.3.6-a.1 Definitions of UE_SISO function block and its interface linked with other blocks

- **Definition of UE_SISO function block**
  - As a functional block that processes the functions of UE, it includes the module for receiving data from the BS and the channel, the OFDM demodulation module, the frequency offset estimation module, the channel estimation module, the HARQ process

module, the detection module, the descrambling module, and the channel decoding module.

- **Definition of the interface**
  - Parameter function block, BS_SISO function block, Channel SISO function block are interlocked. First, the initial value is received through the parameter function block to initialize the user information. The initial value includes information related to the number of resource blocks (RBs), the number of OFDM subcarriers per RB, the number of symbols per slot, and the OFDM FFT size. Then, the OFDM signal and the channel_SISO generated from the BS_SUSISO functional block are received through the channel generated from the functional block. Then, the UE internal module is executed.

3.3.6-a.2 Definitions of modules in UE_SISO function block for Downlink

- **Channel filtering module**
  - Module that plays a role of passing the OFDM signal generated by the BS through the channel and adding a virtual frequency offset.
- **Synchronization module**
  - Module that compensates the received signal by estimating the frequency offset using the PSS and SSS transmitted by the BS.
- **OFDM demodulation module**
  - Module that removes the cyclic prefix from the synchronized received signal stream and then demodulates the OFDM signal through IFFT.
- **Channel estimation module**
  - Module that performs an OFDM channel estimation function. Performs estimation and interpolation of the actual physical channel according to the used RS signal.
- **Feedback calculation module**
  - Module for calculating a CQI through a channel estimation value.
- **Data extraction module**
  - Module that extracts the data in the received resource grid in the order in which they are inserted in the BS.
- **Detection module**
  - Module that calculates the LLR using statistical information of the channel estimation value and noise from the extracted data signal.
- **Descrambling module**
  - Module that performs descrambling LLR sequence contrary to BS scramble.
- **HARQ process**
  - Module that executes the HARQ process. After storing the NACK-transmitted signal, the LLR is recalculated using a chase combining technique and a retransmitted signal.

- **Channel decoding module**
  - Module that extracts data bit using the calculated LLR through LDPC decoding.

3.3.6-b BS_SISO function block for Uplink

3.3.6-b.1 Definitions of BS_SUSISO function block and its interface linked with other blocks

- **Definition of BS_SUSISO function block**
  - As a functional block that processes the functions of BS, it includes the module for receiving data from the UE and the channel, the OFDM demodulation module, the frequency offset estimation module, the channel estimation module, the HARQ process module, the detection module, the descrambling module, and the channel decoding module.

- **Definition of the interface**
  - Parameter function block, UE_SUSISO function block, Channel SISO function block are interlocked. First, the initial value is received through the parameter function block to initialize the user information. The initial value includes information related to the number of resource blocks (RBs), the number of OFDM subcarriers per RB, the number of symbols per slot, and the OFDM FFT size. Then, the OFDM signal and the channel_SISO generated from the UE_SUSISO functional block are received through the channel generated from the functional block. Then, the BS internal module is executed.

3.3.6-b.2 Definitions of modules in BS_SUSISO function block for Uplink

- **Channel filtering module**
  - Module that plays a role of passing the OFDM signal generated by the UE through the channel and adding a virtual frequency offset.
- **Synchronization module**
  - Module that compensates the received signal by estimating the frequency offset.
- **OFDM demodulation module**
  - Module that removes the cyclic prefix from the synchronized received signal stream and then demodulates the OFDM signal through IFFT.
- **Channel estimation module**
  - Module that performs an OFDM channel estimation function. Performs estimation and interpolation of the actual physical channel according to the used RS signal.
- **Data extraction module**
  - Module that extracts the data in the received resource grid in the order in which they are inserted in the UE.

- **Detection module**
  - Module that calculates the LLR using statistical information of the channel estimation value and noise from the extracted data signal.
- **Descrambling module**
  - Module that performs descrambling LLR sequence contrary to UE scramble.
- **HARQ process**
  - Module that executes the HARQ process. After storing the NACK-transmitted signal, the LLR is recalculated using a chase combining technique and a retransmitted signal.
- **Channel decoding module**
  - Module that extracts data bit using the calculated LLR through LDPC decoding.
  -

3.3.7-a UE_MIMO function block for Downlink

3.3.7-a.1 Definitions of UE_MIMO function block and its interface linked with other blocks

- **Definition of UE_MIMO function block**
  - Functional block that handles the functions of the UE, basically performs the same function as UE_SISO. In addition, a receive beamforming function and delayer mapping according to multiple antennas are performed.
- **Definition of the interface**
  - Parameter function block, BS_SUMIMO/BS_MUMIMO function block, Channel MIMO function block are interlocked. First, the initial value is received through the parameter function block to initialize the user information. Thereafter, the OFDM signal generated from the BS_SUMIMO / BS_MIMIMO functional block and the channel generated from the Channel_MIMO functional block are received. Then, UE internal module is executed.

3.3.7-a.2 Definitions of modules in UE_MIMO function block for Downlink

- **Channel filtering module**
  - Module that plays a role of passing the OFDM signal generated by the BS through the channel and adding a virtual frequency offset. It is extended to multiple antenna situations and channel filtering is performed for each pair of transmitting antenna port and receiving antenna.
- **Synchronization module**
  - Module that compensates the received signal by estimating the frequency offset using the PSS and SSS transmitted by the BS.
- **OFDM demodulation module**
  - Module that removes the cyclic prefix from the synchronized received signal stream and

then demodulates the OFDM signal through IFFT. It is extended to multiple antennas and applied to each receiving antenna.

- **Channel estimation module**
  - Module that performs an OFDM channel estimation function. Performs estimation and interpolation of the actual physical channel according to the used RS signal. In the case of multiple antennas, channel estimation and interpolation are performed for each antenna pair or for each antenna pair and for each layer pair for spatial multiplexing.
- **Data extraction module**
  - Module that extracts the data in the received resource grid in the order in which they are inserted in the BS. In the case of multiple antennas, data is extracted for each antenna.
- **Detection module**
  - Module that calculates the LLR using statistical information of the channel estimation value and noise from the extracted data signal. In the case of multiple antennas, the reception beamforming function is included, and in the case of multiple layers, the LLR is calculated for each layer.
- **Delayer mapping module**
  - Module that performs the delayer mapping of the extracted LLR contrary to BS layer mapping.
- **Descrambling module**
  - Module that performs descrambling LLR sequence contrary to BS scramble.
- **HARQ process**
  - Module that executes the HARQ process. After storing the NACK-transmitted signal, the LLR is recalculated using a chase combining technique and a retransmitted signal.
- **Channel decoding module**
  - Module that extracts data bit using the calculated LLR through LDPC decoding.

3.3.7-b BS_MIMO function block for Uplink

3.3.7-b.1 Definitions of BS_MIMO function block and its interface linked with other blocks

- **Definition of BS_SUMIMO function block**
  - As a functional block that processes the functions of BS, it includes the module for receiving data from the UE and the channel, the OFDM demodulation module, the frequency offset estimation module, the channel estimation module, the HARQ process module, the detection module, the descrambling module, and the channel decoding module.

- **Definition of the interface**

- Parameter function block, UE_SUSISO function block, Channel SISO function block are interlocked. First, the initial value is received through the parameter function block to initialize the user information. The initial value includes information related to the number of resource blocks (RBs), the number of OFDM subcarriers per RB, the number of symbols per slot, and the OFDM FFT size. Then, the OFDM signal and the channel_SISO generated from the UE_SUSISO functional block are received through the channel generated from the functional block. Then, the BS internal module is executed.

3.3.7-b.2 Definitions of modules in BS_MIMO function block for Uplink

- **Channel filtering module**
  - Module that plays a role of passing the OFDM signal generated by the UE through the channel and adding a virtual frequency offset.
- **Synchronization module**
  - Module that compensates the received signal by estimating the frequency offset. Perfect synchronization is assumed.
- **OFDM demodulation module**
  - Module that removes the cyclic prefix from the synchronized received signal stream and then demodulates the OFDM signal through IFFT.
- **Channel estimation module**
  - Module that performs an OFDM channel estimation function. Performs estimation and interpolation of the actual physical channel according to the used RS signal. Also, calculates effective channel (channel multiplied by precoder) as well.
- **Data extraction module**
  - Module that extracts the data in the received resource grid in the order in which they are inserted in the UE.
- **Detection module**
  - Module that calculates the LLR using statistical information of the channel estimation value and noise from the extracted data signal.
- **Descrambling module**
  - Module that performs descrambling LLR sequence contrary to UE scramble.
- **HARQ process**
  - Module that executes the HARQ process. After storing the NACK-transmitted signal, the LLR is recalculated using a chase combining technique and a retransmitted signal.
- **Channel decoding module**
  - Module that extracts data bit using the calculated LLR through LDPC decoding.

3.4 Simulator key performance index

- **Block Error Rate**
  - This represents a ratio of the number of erroneous blocks to the total number of blocks transmitted. In this simulator, we measure the Block Error Rate for each SNR and CQI.
- **Throughput**
  - This is the maximum rate of production or the maximum rate at which transmission can be processed. It is measured for each CQI and SNR.

3.5 Environment of simulator development and test plan

3.5.1 Environment of simulator development

- **Simulator development environment (cloud, server)**
  - server environment
- **Development Language and Development Tools**
  - Matlab
  - C++

3.5.2 Test plan

- **Build test environment (test equipment and tools)**
  1. After initial simulator development using Matlab, conversion to object oriented program (C ++)
- **Test Spec.**
  - 5G (NR) based
- **Test Scenarios**
  - Single-cell SU-SISO
  - Single-cell SU-MIMO
  - Single-cell SU-MIMO Calibration 1
  - Single-cell SU-MIMO Calibration 2
  - Scenario using 5G (NR) technology

4. Detailed description of modules

4.1 Detailed explanation of modules

4.1.1-a Parameter function block for Downlink

- Determine the number of slots and symbols per subframe, subcarrier spacing, bandwidth set and resource block set according to the numerology. That parameters are defined in 4.2 and 4.3 of TS 38.211[1] and the below table shows subcarrier spacing and the number of slots and symbols corresponding to numerology.

μ	$\Delta f = 2^{\mu} \cdot 15\,[\text{kHz}]$	**Cyclic prefix**
0	15	Normal
1	30	Normal
2	60	Normal, Extended
3	120	Normal
4	240	Normal

Table 1 Supported transmission numerologies in [1]

μ	$N_{\text{symb}}^{\text{slot}}$	$N_{\text{slot}}^{\text{frame},\mu}$	$N_{\text{slot}}^{\text{subframe},\mu}$
0	14	10	1
1	14	20	2
2	14	40	4
3	14	80	8
4	14	160	16
5	14	320	32

Table 2 Number of OFDM symbols per slot, slots per frame, and slots per subframe in [1]

- Parameters can be organized as follows:

1. Parameters for frame structure

Role	Parameter name	Default	Options	Explanation
Frame structure	mu	0	0,1,2,3	μ
	subcarrier_spacing	15	Calculated ((2^obj.mu) * 15e3)	Subcarrier spacing
	num_sc	12	fixed	Number of subcarriers in

				one resource block
	num_symb	14	fixed	Number of OFDM symbols in one slot
	num_subframe	init	Any natural number	Number of subframes
	num_slot_in_subframe	1	Calculated (2^(obj.mu))	Number of slots in one subframe
	num_symb_in_subframe	14	Calculated (2^(obj.mu)*14;)	Number of OFDM symbols in one subframe
	Ts	1/(15e3 *2048)	Fixed	Reference time unit (T_s)
	size_fft	4096	Fixed	4096
	Fs	61440 [kHz]	Calculated (obj.size_fft * obj.subcarrier_spacing;)	Sampling frequency
	num_symb_in_slot	14	Fixed	Number of OFDM symobls in one slot

2. Parameters for reference signal

Reference signal	num_port_DMRS;	1	Calculated	Number of DMRS antenna ports
	num_port_CSIRS;	1	Calculated	Number of CSIRS antenna ports
	port_set_DMRS;	1000	1000 ~ 1011	DMRS antenna port p
	port_set_CSIRS;	3000	3000 ~ 3031	CSIRS antenna port p
	CSIRS_density;	1	0.5, 1, 3	Density of CSIRS
	CSIRS_CDMType;	'no CDM'	'no CDM', 'FD-CDM2', 'CDM4', 'CDM8'	CDM type of CSIRS
	CSIRS_bitmap_value	[1, 2, 3, 4, 5, 6]	Higher layer parameter	bitmap provided by the higher-layer parameter CSI-RS-ResourceMapping
	CSIRS_symbol_start_idx	[0, 5]	Higher layer parameter	Starting positions of a CSIRS provided by higher-layer parameter CSI-RS-ResourceMapping.
	row_CSIRS;	1	1~19	Row number in Table 7.4.1.5.2-1
	PDSCH_mapping_type;	‘A’	‘A’, ‘B’	PDSCH mapping type
	DMRS_type;	1	1, 2	DMRS configuration type
	DL_DMRS_add_pos;	0	0, 1, 2, 3	DL-DMRS-add-pos
	DL_DMRS_typeA_pos;	2	Higher layer parameter	DL_DMRS_typeA_pos
	dur_PDSCH_transmission;	14	1~14	Duration of PDSCH transmission
	CSIRS_periodicity;	5	Any natural number	Periodicity of CSIRS

3. Parameters for resource allocation

Resource allocation	use_PDCCH	false	false, true	Whether use PDCCH
	use_PBCH	false	false, true	Whether use PBCH
	use_Power_allocation	false	false, true	Whether use power allocation
	use_Sync	false	false, true	Whether use synchronization
	CFI	1	1	Control Format Indicator

4. Parameters for channel model

Channel model	DS	100* 10e-9	10 * 10e-9, 30 * 10e-9, 100* 10e-9, 300 * 10e-9, 1000 * 10e-9	Wanted delay spread in ns
	CDL;	Depends on channel type	Depends on channel type	Cluster delay line
	Channel;	Contains channel information depending on initial value	Contains channel information depending on initial value	Channel object
	Cov_Matrix_time;	Depends on channel type	Depends on channel type	Time covariance matrix
	Cov_Matrix_freq;	Depends on channel type	Depends on channel type	Frequency covariance matrix
	method_interpolation	nearest_neighbor	nearest_neighbor, sinc_interpolation	Channel interpolation type when generating channel model
	sin_num	10	10	Number of sin realizations
	w_d;	Calculated from 'user_speed', 'f', 'light_speed'.	Calculated from 'user_speed', 'f', 'light_speed'.	Implicit variable for channel modeling
	t;	0	0 if init. ch_mode = 'Block_Fading' , obj.t = 0:1:obj.num_symb-1; obj.t = obj.t*(1e-3/(2^(obj.mu))); if init.ch_mode = 'Fast_Fading'	Time for 14 OFDM slots of one subframe(1ms)
	AS_ratio	1	1	Channel realization fro MIMO 5G channel model
	c_ASD ;	Depends on channel	Depends on channel type	Degrees (ASD)

		type		
	c_ASA ;	Depends on channel type	Depends on channel type	Degrees (ASA)
	c_ZSD ;	Depends on channel type	Depends on channel type	Degrees (ZSD)
	c_ZSA ;	Depends on channel type	Depends on channel type	Degrees (ZSA)
	XPR_dB;	Depends on channel type	Depends on channel type	Cross polarization ratio in dB
	user_speed	5/6 [m/s]	Any positive number	Speed of user
	theta_v	45	Fixed	travel elevation angle
	phi_v	45	fixed	travel azimuth angle
	light_speed	299792458	Fixed	Speed of light

5. Parameters for channel estimation

Channel estimation	ch_est_mode	'MMSE'	'Perfect', 'LS', 'MMSE'	Channel estimation method with CSIRS
	ch_est_mode_DMRS	'MMSE'	'Perfect', 'LS', 'MMSE'	Channel estimation method with DMRS
	ch_interp_mode	'linear'	'linear', 'spline', 'TDI'	Channel estimation interpolation method with CSIRS
	ch_interp_mode_DMRS	'linear'	'linear', 'spline'	Channel estimation interpolation method with DMRS

6. Parameters for channel coding

Channel coding	num_HARQ_retransmission	0	0,1,2,3	Number of HARQ retransmittions
	num_HARQ_process	8	8	Number of HARQ processes
	hard_decision	False	True, false	Whether perform hard decision

7. Parameters for synchronization

Synchronization	sync_freq_offset	0	Any value within [-1, 1]	Normailized frequency offset
	sync_freq_mode	Perfect	'perfect', 'estimated'	Whether frequency synchronization is perferct or estimated
	num_SS_block	4	Calculated	Number of SS blocks
	slotset_SS_block	[1,2]	Calculated	Slot set for SS blocks

8. Parameters for precoding

	zero_TTI_feedback = true;	True	True, false	Whether perform feedback in zero TTI

				(instantaneously)

9. Parameters for detection

Detection	Nothing for SISO	Null	Null	null

10. Parameters for transmission

Transmission	num_codewords	1	1 for SUSISO	Number of codewords
	num_data_bits	Calculated	Depends on number of available resource element for data transmission	Length of data bits
	num_coded_bits	calculated	Depends on number of available resource element for data transmission	Length of coded bits
	M_order	2	Depends on CQI (2 if modulation order does not change)	Modulation order
	Coding_rate	1/3	Depends on CQI (1/3 if coding rate does not change)	Coding rate

11. Parameters for simulation scenario

Simulation scenario	Sim_Case	init	SUSISO, SUMIMO	Determines whether simulation is single user SISO or single user MIMO
	f	Init	Any positive value	Carrier frequency
	BW	init	TS 38.104 table 5.3.2	Carrier bandwidth
	BW_set	[5 10 15 20 25 30 40 50] * 1e6	TS 38.104 table 5.3.2 Depend on f and mu	Bandwidth set
	RB_set	RB_set = [25 52 79 106 133 160 216 270]	TS 38.104 table 5.3.2 Depend on f and mu	Number of resource blocks set
	num_RB_maximum	25	Calculated (obj.RB_set(obj.BW_set==obj.BW);)	Allowed maximum number of resource blocks
	num_RB	25	Calculated (obj.RB_set(obj.BW_set==obj.BW))	Used number of resource blocks, here only consider full bandwidth usage for PDSCH
	cal_scenario	init	true, false	Whether simulate for calibration
	scenario_beam	1	Fixed	Beam number (1 for SISO)
	num_Tx_antenna	1	1 for SUSISO	Number of Tx antenna
		4	Interger for MIMO	
	num_Rx_antenna	1	1 for SUSISO	Number of Rx antenna

		1	Interger for MIMO	
	Tx_pol	1	1 for SUSISO	Tx polarization
		1	1 for co pol, 2 for cross pol in MIMO	
	Rx_pol	1	1 for SUSISO	Rx polarization
		1	1 for co pol, 2 for cross pol in MIMO	
	N1	1	1 for SUSISO	Number of horizontal Tx antennas
		2	Interger for MIMO	
	N2	1	1 for SUSISO	Number of vertical Tx antennas
		2	Interger for MIMO	
	M1	1	1 for SUSISO	Number of horizontal Rx antennas
		1	Interger for MIMO	
	M2	1	1 for SUSISO	Number of vertical Rx antennas
		1	Interger for MIMO	
	O1	2	2 for SUSISO	Number of DFT oversampling of horizontal axis
		2	Calculated for MIMO	
	O2	1	1 for SUSISO	Number of DFT oversampling of vertical axis
		2	Calculated for MIMO	
	L1	4	Calculated (obj.L1 = obj.num_SS_block)	Number of horizontal wide beams
		4	Calculated for MIMO	
	L2	1	1 for SUSISO	Number of vertical wide beams,
		2	Calculated for MIMO	
	S1	0.5	Calculated (obj.N1*obj.O1/obj.L1)	Narrow beam interval that composed of diffrent wide beam (horizontal)
		1		
	S2	1	Calculated (obj.N2*obj.O2/obj.L2)	Narrow beam interval that composed of diffrent wide beam (vertical)
		1		
	P1	4	Calculated (max(obj.S1,2*obj.O1))	Number of horizontal narrow beams per a wide beam
		4		
	P2	2	Calculated (max(obj.S2,2*obj.O2))	Number of vertical narrow beams per a wide beam
		2		
	pol_slant_angle	[-45 45]	fixed	polarization slant angle
	TxArrayType	URA	URA, ULA	Transmit array type
	RxArrayType	URA	URA, ULA	Receive array type
	Tx_d_lambda	0.5	Fixed	Tx antenna spacing
	Rx_d_lambda	0.5	fixed	Rx antenna spacing
	Tx_downtilt	0	Fixed	Downtilt angle of transmit antenna
	Tx_pattern_type	Pattern	'Omni-directional', 'Pattern'	Pattern of transmit antenna
	Rx_pattern_type	Omni-	'Omni-directional',	Pattern of receive antenna

		directional	'Pattern'	
	SNR_range	init	Calculated from main file	SNR values for simulation
	SNR	init	Calculated from SNR_range	SNR value for current simulation
	ind_SNR	init	Calculated from ind_SNR	Index of SNR in SNR range for simulation
	sigma_n_time	init	Calculated (1/(10^(obj.SNR/10)))	Noise variance in time
	sigma_n_freq	init	Calculated (obj.size_fft/(obj.num_RB*obj.num_sc) * 1/(10^(obj.SNR/10)))	Noise variance in frequency
	cell_ID	0	Fixed	Cell ID
	use_fullband	true	Fixed	Whether use fullband

12. Parameters for power allocation

Power allocation	power_DMRS	1	Any positive value	Power of DMRS
	power_CSIRS	1	Any positive value	Power of CSIRS
	power_data	1	Any positive value	Power of data

13. Parameters for OFDM

OFDM	kappa	64	Fixed	κ
	Tc	1/(960e3*2048)	Calculated (obj.Ts / obj.kappa;)	T_c
	time_CP(1)	10240/(960e3*2048)	Calculated (obj.time_CP(1) = (144 * obj.kappa * 2^(-obj.mu) + 16 * obj.kappa) * obj.Tc;)	For first OFDM symbol
	time_CP(2)	9216/(960e3*2048)	Calculated (obj.time_CP(2) = (144 * obj.kappa * 2^(-obj.mu)) * obj.Tc)	For 2^{nd}~14^{th} OFDM symbol
	length_CP(1)	320	Calculated (obj.length_CP(1) = int16(obj.time_CP(1)*obj.Fs);)	For first OFDM symbol
	length_CP(2)	288	Calculated (obj.length_CP(2) = int16(obj.time_CP(2)*obj.Fs);)	For 2^{nd}~14^{th} OFDM symbol

- Determine antenna port set and transmission mode according to simulation case. For each simulation case, the antenna port set is defined in TS 38.211 [1].

- The parameters required for the simulation are calculated using the given parameters. Once the bandwidth is determined, the sampling frequency and the cyclic prefix time are calculated and the codebook is stored according to the codebook mode.
- Each channel may be defined by a PDP (Power-delay Profile), such as PedA, PedB, VehA, VehB, EPA, EVA, and ETU, and 3D spatial channel model(CDL, TDL). AWGN is not defined as PDPs. The following table shows the PDP for each channel except 3D spatial channel model.

Tap	Channel A		Channel B		Doppler spectrum
	Relative delay (ns)	Average power (dB)	Relative delay (ns)	Average power (dB)	
1	0	0	0	0	Classic
2	110	–9.7	200	–0.9	Classic
3	190	–19.2	800	–4.9	Classic
4	410	–22.8	1 200	–8.0	Classic
5	–	–	2 300	–7.8	Classic
6	–	–	3 700	–23.9	Classic

Table 3. Pedestrian test environment tapped-delay-line parameters in [11]

Tap	Channel A		Channel B		Doppler spectrum
	Relative delay (ns)	Average power (dB)	Relative delay (ns)	Average power (dB)	
1	0	0.0	0	–2.5	Classic
2	310	–1.0	300	0	Classic
3	710	–9.0	8 900	–12.8	Classic
4	1 090	–10.0	12 900	–10.0	Classic
5	1 730	–15.0	17 100	–25.2	Classic
6	2 510	–20.0	20 000	–16.0	Classic

Table 4. Vehicular test environment, high antenna, tapped-delay-line parameters in [11]

Excess tap delay [ns]	Relative power [dB]
0	0.0
30	-1.0
70	-2.0
90	-3.0
110	-8.0
190	-17.2
410	-20.8

Table 5. Extended Pedestrian A model (EPA) in [10]

Excess tap delay [ns]	Relative power [dB]
0	-1.0
50	-1.0
120	-1.0
200	0.0
230	0.0
500	0.0
1600	-3.0
2300	-5.0
5000	-7.0

Table 6. Extended Vehicle A model (EVA) in [10]

Excess tap delay [ns]	Relative power [dB]
0	0.0
30	-1.5
150	-1.4
310	-3.6
370	-0.6
710	-9.1
1090	-7.0
1730	-12.0
2510	-16.9

Table 7. Extended Typical Urban model (ETU) in [10]

- **Codebook_generation module**

- According to the antenna set-up and parameters for the codebook mode, generate a codebook for the CSI feedback as in 3GPP TS 38.214 V1.2.0 (2017-11)
- Ng, N1 and N2 are the number of antenna panels, the number of horizontal antennas in a panel and the number of vertical antennas in a panel, respectively. There are two codebook types, 1,2. From the codebook type decided by the upper layer, generate Type I Single-Panel Codebook, Type I Multi-Panel Codebook. The oversampling factor, O1 and O2 are decided by the number of horizontal antennas and the number of vertical antennas.

- The generated codebook is a cell-type variable that the size is the number of allowed layers in the configured set-up. Each cell type variable is composed of a cell-type variable for the codebook for each corresponding layer. The size of each of the cell-type variables is decided by the configured parameters for the analog beamforming $\mathrm{i}_{1,\mathrm{k}}$ and other parameters for the digital beamforming.
- Codebook matrix is made of the combinations of the DFT vectors of vertical and horizontal antennas. The DFT vector of the vertical antennas,

$$\boldsymbol{u}_m = \begin{cases} \begin{bmatrix} 1 & e^{j\frac{2\pi m}{O_2 N_2}} & \cdots & e^{j\frac{2\pi m(N_2-1)}{O_2 N_2}} \end{bmatrix}, & N_2 > 1 \\ 1, & N_2 = 1 \end{cases}$$

and the DFT matrix for the vertical and horizontal antennas,

$$\boldsymbol{v}_{l,m} = \begin{bmatrix} \boldsymbol{u}_m & e^{j\frac{2\pi l}{O_1 N_1}} \boldsymbol{u}_m & \cdots & e^{j\frac{2\pi l(N_1-1)}{O_1 N_1}} \boldsymbol{u}_m \end{bmatrix}^T$$

is made by some functions.

4.1.1-b Parameter function block for Uplink

- Basically, same with Downlink case except for detailed parameters for RS and transform precoding.
- Detailed parameters can be organized as follows:

1. Parameters for frame structure

Role	Parameter name	Default	Options	Explanation
Frame structure	mu	0	0,1,2,3	μ
	subcarrier_spacing	15	Calculated ((2^obj.mu) * 15e3)	Subcarrier spacing
	num_sc	12	fixed	Number of subcarriers in one resource block
	num_symb	14	fixed	Number of OFDM symbols in one slot
	num_subframe	init	Any natural number	Number of subframes
	num_slot_in_subframe	1	Calculated (2^(obj.mu))	Number of slots in one subframe
	num_symb_in_subframe	14	Calculated (2^(obj.mu)*14;)	Number of OFDM symbols in one subframe
	Ts	1/(15e3 *2048)	Fixed	Reference time unit (T_s)
	size_fft	4096	Fixed	4096
	Fs	61440 [kHz]	Calculated (obj.size_fft * obj.subcarrier_spacing;)	Sampling frequency
	num_symb_in_slot	14	Fixed	Number of OFDM symobls in one slot

2. Parameters for reference signal

Role	Parameter name	Default	Options	Explanation
Reference signal	num_port_DMRS;	1	Calculated	Number of DMRS antenna ports
	num_port_SRS;	1	Calculated	Number of SRS antenna ports
	port_set_DMRS;	1000	1000 ~	DMRS antenna port p
	port_set_SRS;	1000	3000~	SRS antenna port p
	PUSCH_mapping_type;	'A'	'A', 'B'	PUSCH mapping type
	DMRS_type;	1	1, 2	DMRS configuration type
	UL_DMRS_add_pos;	0	0, 1, 2, 3	DL-DMRS-add-pos
	dur_PUSCH_transmission;	14	1~14	Duration of PUSCH transmission

3. Parameters for resource allocation

Role	Parameter name	Default	Options	Explanation
Resource allocation	use_Power_allocation	false	false, true	Whether use power allocation
	use_Sync	false	false	Whether use synchronization
	CFI	1	1	Control Format Indicator

4. Parameters for channel model

Role	Parameter name	Default	Options	Explanation
Channel model	DS	100* 10e-9	10 * 10e-9, 30 * 10e-9, 100* 10e-9, 300 * 10e-9, 1000 * 10e-9	Wanted delay spread in ns
	CDL;	Depends on channel type	Depends on channel type	Cluster delay line
	Channel;	Contains channel information depending on initial value	Contains channel information depending on initial value	Channel object
	Cov_Matrix_time;	Depends on channel type	Depends on channel type	Time covariance matrix
	Cov_Matrix_freq;	Depends on channel type	Depends on channel type	Frequency covariance matrix
	method_interpolation	nearest_neighbor	nearest_neighbor, sinc_interpolation	Channel interpolation type when generating channel model
	sin_num	10	10	Number of sin realizations
	w_d;	Calculated	Calculated from 'user_speed', 'f',	Implicit variable for channel modeling

5G Wireless Innovative Systems Engineering Research Center

		from ‘user_speed’, ‘f’, ‘light_speed’.	‘light_speed’.	
	t;	0	0 if init. ch_mode = ‘Block_Fading' , obj.t = 0:1:obj.num_symb-1; obj.t = obj.t*(1e-3/(2^(obj.mu))); if init.ch_mode = 'Fast_Fading'	Time for 14 OFDM slots of one subframe(1ms)
	AS_ratio	1	1	Channel realization fro MIMO 5G channel model
	c_ASD ;	Depends on channel type	Depends on channel type	Degrees (ASD)
	c_ASA ;	Depends on channel type	Depends on channel type	Degrees (ASA)
	c_ZSD ;	Depends on channel type	Depends on channel type	Degrees (ZSD)
	c_ZSA ;	Depends on channel type	Depends on channel type	Degrees (ZSA)
	XPR_dB;	Depends on channel type	Depends on channel type	Cross polarization ratio in dB
	ch_type	CDL_A	‘CDL_A’, ‘CDL_B’, ‘CDL_C’, ‘CDL_D’ ‘CDL_E’	
	ch_mode	Block_Fading	‘Block_Fading’, ‘Fast_Fading’	
	user_speed	5/6 [m/s]	Any positive number	Speed of user
	theta_v	45	Fixed	travel elevation angle
	phi_v	45	fixed	travel azimuth angle
	light_speed	299792458	Fixed	Speed of light

5. Parameters for channel estimation:

Role	Parameter name	Default	Options	Explanation
	ch_est_mode_DMRS	‘MMS	‘Perfect’, ‘LS’,	Channel estimation method

		E'	'MMSE'	with DMRS
	ch_interp_mode_DMRS	'linear'	'linear', 'spline'	Channel estimation interpolation method with DMRS

6. Parameters for channel coding

Role	Parameter name	Default	Options	Explanation
Channel coding	num_HARQ_retransmission	0	0,1,2,3	Number of HARQ retransmittions
	num_HARQ_process	8	8	Number of HARQ processes
	hard_decision	False	True, false	Whether perform hard decision

7. Parameters for synchronization

Role	Parameter name	Default	Options	Explanation
Synchronization	sync_freq_offset	0	Any value within [-1, 1]	Normailized frequency offset
	sync_freq_mode	Perfect	'perfect', 'estimated'	Whether frequency synchronization is perferct or estimated

8. Parameters for precoding

Role	Parameter name	Default	Options	Explanation
Precoding	zero_TTI_feedback = true;	True	True, false	Whether perform feedback in zero TTI (instantaneously)

9. Parameters for transmission

Role	Parameter name	Default	Options	Explanation
Transmission	Transform precoding	true	true, false	Whether transform precoding is enabled or not
	num_codewords	1	1 for SUSISO	Number of codewords
	num_data_bits	Calculated	Depends on number of available resource element for data transmission	Length of data bits
	num_coded_bits	calculated	Depends on number of available resource element for data transmission	Length of coded bits
	M_order	2	Depends on CQI (2 if modulation order does not change)	Modulation order
	Coding_rate	1/3	Depends on CQI (1/3 if coding rate does not change)	Coding rate

10. Parameters for simulation scenario

Role	Parameter name	Default	Options	Explanation
Simulation scenario	Sim_Case	init	SUSISO, SUMIMO	Determines whether simulation is single user SISO or single user MIMO
	f	Init	Any positive value	Carrier frequency
	BW	init	TS 38.104 table 5.3.2	Carrier bandwidth

	BW_set	[5 10 15 20 25 30 40 50] * 1e6	TS 38.104 table 5.3.2	Bandwidth set
	RB_set	RB_set = [25 52 79 106 133 160 216 270]	TS 38.104 table 5.3.2	Number of resource blocks set
	num_RB_maximum	25	Calculated (obj.RB_set(obj.BW_set==obj.BW);)	Allowed maximum number of resource blocks
	num_RB	25	Calculated (obj.RB_set(obj.BW_set==obj.BW))	Used number of resource blocks, here only consider full bandwidth usage for PUSCH
	cal_scenario	init	true, false	Whether simulate for calibration
	scenario_beam	1	Fixed	Beam number (1 for SISO)
	num_Tx_antenna	1	1 for SUSISO	Number of Tx antenna
		4	Interger for MIMO	
	num_Rx_antenna	1	1 for SUSISO	Number of Rx antenna
		1	Interger for MIMO	
	Tx_pol	1	1 for SUSISO	Tx polarization
		1	1 for co pol, 2 for cross pol in MIMO	
	Rx_pol	1	1 for SUSISO	Rx polarization
		1	1 for co pol, 2 for cross pol in MIMO	
	N1	1	1 for SUSISO	Number of horizontal Tx antennas
		2	Interger for MIMO	
	N2	1	1 for SUSISO	Number of vertical Tx antennas
		2	Interger for MIMO	
	M1	1	1 for SUSISO	Number of horizontal Rx antennas
		1	Interger for MIMO	
	M2	1	1 for SUSISO	Number of vertical Rx antennas
		1	Interger for MIMO	
	O1	2	2 for SUSISO	Number of DFT oversampling of horizontal axis
		2	Calculated for MIMO	
	O2	1	1 for SUSISO	Number of DFT oversampling of vertical axis
		2	Calculated for MIMO	
	L1	4	Calculated (obj.L1 = obj.num_SS_block)	Number of horizontal wide beams
		4	Calculated for MIMO	
	L2	1	1 for SUSISO	Number of vertical wide beams,
		2	Calculated for	

			MIMO	
	S1	0.5 1	Calculated (obj.N1*obj.O1/obj.L1)	Narrow beam interval that composed of diffrent wide beam (horizontal)
	S2	1 1	Calculated (obj.N2*obj.O2/obj.L2)	Narrow beam interval that composed of diffrent wide beam (vertical)
	P1	4 4	Calculated (max(obj.S1,2*obj.O1))	Number of horizontal narrow beams per a wide beam
	P2	2 2	Calculated (max(obj.S2,2*obj.O2))	Number of vertical narrow beams per a wide beam
	pol_slant_angle	[-45 45]	fixed	polarization slant angle
	TxArrayType	URA	URA, ULA	Transmit array type
	RxArrayType	URA	URA, ULA	Receive array type
	Tx_d_lambda	0.5	Fixed	Tx antenna spacing
	Rx_d_lambda	0.5	fixed	Rx antenna spacing
	Tx_downtilt	0	Fixed	Downtilt angle of transmit antenna
	Tx_pattern_type	Pattern	‘Omni-directional', 'Pattern'	Pattern of transmit antenna
	Rx_pattern_type	Omni-directional	‘Omni-directional', 'Pattern'	Pattern of receive antenna
	SNR_range	init	Calculated from main file	SNR values for simulation
	SNR	init	Calculated from SNR_range	SNR value for current simulation
	ind_SNR	init	Calculated from ind_SNR	Index of SNR in SNR range for simulation
	sigma_n_time	init	Calculated (1/(10^(obj.SNR/10)))	Noise variance in time
	sigma_n_freq	init	Calculated (obj.size_fft/(obj.num_RB*obj.num_sc) * 1/(10^(obj.SNR/10)))	Noise variance in frequency
	cell_ID	0	Fixed	Cell ID
	use_fullband	true	Fixed	Whether use fullband

11. Parameters for power allocation

Role	Parameter name	Default	Options	Explanation
Power allocation	power_DMRS	1	Any positive value	Power of DMRS
	power_SRS	1	Any positive value	Power of SRS
	power_data	1	Any positive value	Power of data

12. Parameters for OFDM

Role	Parameter name	Default	Options	Explanation

OFDM	kappa	64	Fixed	κ
	Tc	1/(960e3*2048)	Calculated (obj.Ts / obj.kappa;)	T_c
	time_CP(1)	10240/(960e3*2048)	Calculated (obj.time_CP(1) = (144 * obj.kappa * 2^(-obj.mu) + 16 * obj.kappa) * obj.Tc;)	For first OFDM symbol
	time_CP(2)	9216/(960e3*2048)	Calculated (obj.time_CP(2) = (144 * obj.kappa * 2^(-obj.mu)) * obj.Tc)	For 2^{nd}~14^{th} OFDM symbol
	length_CP(1)	320	Calculated (obj.length_CP(1) = int16(obj.time_CP(1)*obj.Fs);)	For first OFDM symbol
	length_CP(2)	288	Calculated (obj.length_CP(2) = int16(obj.time_CP(2)*obj.Fs);)	For 2^{nd}~14^{th} OFDM symbol

4.1.2-a BS_SISO function block submodules for Downlink

- **Feedback reception module**

- The module for the feedback reception: The module received CQI from an UE and decide modulation order and coding rate, and so on. The CQI table is generated from Table 5.2.2.1-1 in 3GPP TS 38.214 V1.2.0 (2017-11) [4].
- The precoding matrix for the precoding is decided from the feedback from an UE. UE feedback codebook indexes for the analog precoding and the digital precoding. The feedback for the index for the analog precoding is composed of a single parameter for the whole band. The feedback for the indexes for the digital precoding are composed of multiple parameters that each of them represents the digital precoding index for the corresponding sub-band.
- From the precoding feedback indexes, the module decides the precoder for each of the RB.
- In the case of zero-TTI feedback mode is assumed, the calculation for the precoding indexes is performed at this module, instead of the feedback calculation module in the UE_MIMO module.

method_feedback_reception(obj, para, feedback, ch_output)
Input : feedback.CQI, para.zero_TTI_feedback, para.Channel_type, para.sigma_n_freq, para.M_order, para.Coding_rate Output : obj.CQI

- **RS_allocation module**

- Generates CSI-RS and allocated the CSI-RS to the resource grid. CSI-RS is generated as follows:
  - $$r(m)=\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m)\right)+j\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m+1)\right).$$
- Here, $c(i)$ is the pseudo random sequence defined in 5.2.1 [1], and is generated from the initial value as follows. Here, $n_{s,f}$ is the slot number in radio frame, l is the OFDM symbol number in slot, and n_{ID} indicated higher layer parameter, Scrambling ID.

$$c_{\mathrm{init}}=\left(2^{10}\cdot\left(14n_{\mathrm{s,f}}+l+1\right)\left(2n_{\mathrm{ID}}+1\right)+n_{\mathrm{ID}}\right)\bmod 2^{31}$$

- The generated CSI-RS is allocated to the resource grid with following rule,

$$a_{k,l}^{(p,\mu)}=\beta_{\mathrm{CSIRS}}w_{\mathrm{f}}(k')\cdot w_{\mathrm{t}}(l')\cdot r(m)$$
$$k=\bar{k}+k'$$
$$l=\bar{l}+l'$$

where

$$\bar{l}\in\{0,1,\ldots,13\}$$

$[b_3\cdots b_0]$, $k_i=f(i)$ for row 1 of Table 7.4.1.5.2-1

$[b_{11}\cdots b_0]$, $k_i=f(i)$ for row 2 of Table 7.4.1.5.2-1

$[b_2\cdots b_0]$, $k_i=4f(i)$ for row 4 of Table 7.4.1.5.2-1

$[b_5\cdots b_0]$, $k_i=2f(i)$ for all other cases

while other detailed parameters are indicated in Table 7.4.1.5.2-1~5 [1].

- Here, k and l indicated OFDM subcarrier place and OFDM symbol place, respectively.
- For the CSI-RS case, due to density and CDMtype, the number of RSs per RBs varies.
- Generate DMRS and allocated the DMRS to the resource gird. DMRS is generated as follows:

$$r(m)=\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m)\right)+j\frac{1}{\sqrt{2}}\left(1-2\cdot c(2m+1)\right)$$

- Here, $c(i)$ is the pseudo random sequence defined in 5.2.1 [1], and is generated from the initial value as follows. Here, n_s is the slot number in radio frame, l is the OFDM symbol number in slot, $N_{\mathrm{ID}}^{n_{SCID}}$, n_{SCID}, and n_{ID}^{cell} indicated higher layer parameters.

$$c_{\mathrm{init}}=\left(2^{17}\left(14n_{\mathrm{s}}+l+1\right)\left(2N_{\mathrm{ID}}^{n_{\mathrm{SCID}}}+1\right)+2N_{\mathrm{ID}}^{n_{\mathrm{SCID}}}+n_{\mathrm{SCID}}\right)\bmod 2^{31}$$

- The generated DMRS is allocated to the resource gird with following rule,

$$
\begin{aligned}
a_{k,l}^{(p,\mu)} &= \beta_{\mathrm{DMRS}} w_{\mathrm{f}}(k') \cdot w_{\mathrm{t}}(l') \cdot r(2n+k') \\
k &= \begin{cases} 4n+2k'+\Delta & \text{Configuration type 1} \\ 6n+k'+\Delta & \text{Configuration type 2} \end{cases} \\
k' &= 0,1 \\
l &= \bar{l} + l' \\
n &= 0,1,...
\end{aligned}
$$

where each parameter is defined in Table 7.4.1.1.2-1~ 7.4.1.1.2-5 [1].

- Here, k and l indicated OFDM subcarrier place and OFDM symbol place, respectively.
- For DMRS case, the number of RSs per RB is 4, but for the case of additional DMRS to be used, the number of RSs per RB gets larger.

```
method_RS_allocation(obj, para, ind_slot)
Input : ind_slot, parameters for reference signal (4.1.1-a.2)
Output : obj.DMRS_position_total,
         obj.DMRS_position,
         obj.resource_grid_DMRS,
         obj.resource_grid_CSIRS,
         obj.CSIRS_position
Function : func_pseudo_sequence_generation

  <DMRS part>
    1. Initialization of output matrix
    2. Define location of starting subcarrier according to DMRS configuration type
for  (from 1 to length of DMRS port set)
% In SISO case, length of DMRS port set is always '1'.
    if (single-symbol DMRS)
      Set l̄ according to Table 7.4.1.1.2-3 in TS 38.211 (switch)
    elseif (double-symbol DMRS)
      Set l̄ according to Table 7.4.1.1.2-4 in TS 38.211 (switch)
    else    return error;
    end

    if (DL DMRS configuration type 1),
      Set Δ, w_f and w_t according to Table 7.4.1.1.2-1 in TS 38.211 (switch)
      pseudo sequence generation using function func_pseudo_sequence_generation
      extract pseudo sequence actually used

      if (PDSCH mapping type is A),
        l_0 setting according to 'DL-DMRS-typeA-pos'

        for (from 1 to actual pseudo sequence length)
          % Generate of DM-RS position according to 7.4.1.1.2 of TS 38.211
          generate a vaule that matches 'n' in 7.4.1.1.2 of TS 38.211
          if (k' = 0)
            Calculate 'k'
            Input the α_{k,l}^{(p,μ)} to corresponding position (k, l_0 + 1) and (k, l_0 + 2)
            Save that the positions are used : 'true'
          else (k' = 1)
            Performs the same operation as when k' = 0.
          end
        end

        if (l̄ consists of l_0 and other components)
          if (single-symbol DM-RS)
            for (1 to the number of additional components)
              Insert single-symbol to corresponding position (See the figure in 4.1.2)
              Save that the positions are used : 'true'
```

```
            end
          else if (double-symbol DM-RS)
            for (1 to the number of additional components)
              Insert double-symbol to corresponding position (See the figure in 4.1.2)
              Save that the positions are used : 'true'
            end
          end
        end

        for (slot index is 2 to the number of slots in a subframe)
          for (OFDM symbol index is 1 to the number of symbols in a slot)
          end
        end
      else if (PDSCH mapping type is B)
        Setting l_0 to '0'
        for (from 1 to actual pseudo sequence length)
          See the case of 'PDSCH mapping type is A'
        end

        if (Single-symbol DM-RS & Duration of PDSCH transmission is '7')
          % PDSCH mapping type B, Table 7.4.1.1.2-3 in TS 38.211
          Insert single-symbol to corresponding position
          Save that the positions are used : 'true'
        end
      end
    else if (DL DMRS configuration type 2),
      Set Δ, w_f and w_t according to Table 7.4.1.1.2-2 in TS 38.211 (switch)
      pseudo sequence generation
      Same procedure as DL DMRS configuration type1
    end
    save to obj.resource_grid_DMRS_subframe, obj.DMRS_position_subframe
end

Convert the unit of resource grid and DMRS position matrix from subframe to slot.
```

<CSI-RS part>

1. Initialization of output matrix
2. Set w_f and w_t according to CDMtyp. See the Table 7.4.1.5.2-2 to 7.4.1.5.2-5 in TS 38.211 (switch)
3. Generate c_{init} (7.5.1.5.2 in TS 38.211).
4. Pseudo sequence generation using function func_pseudo_sequence_generation
5. Extract pseudo sequence actually used.

```
switch (the number of CSI-RS ports)
% CSI-RS locations within a slot, Table 7.4.1.5.2-1 in TS 38.211
% In SISO case, the number of CSI-RS port is always '1'.
```

```
  case 1
    switch (CSI-RS row)
      case 1
        Define (k_0, l_0)  % k_i = f(i), f(i) is defined in Parameter class (temporal values)
        Set l' = 0, k' = 0, l̄ = l_0 and k̄ = [k_0, k_0 + 4, k_0 + 8].

        for (i = 1: length of actual pseudo sequence / CSI-RS density)
            k = k̄ + k', for 3 of 'k'
            l = l̄ + l'
            Input the a_{k,l}^{(p,μ)} to corresponding positions
            Save that the positions are used : 'true'
            Position TEST (superposition of DM-RS and CSI-RS)
        end
        save to obj.resource_grid_CSIRS, obj.CSIRS_position
      case 2
        Define (k_0, l_0)
        Set l' = 0, k' = 0, l̄ = l_0 and , k̄ = k_0.
        for (i = 1: length of actual pseudo sequence)
            k = k̄ + k' with CSI-RS density
            l = l̄ + l'
            Input the a_{k,l}^{(p,μ)} to corresponding positions
            Save that the positions are used : 'true'
            Position TEST (superposition of DM-RS and CSI-RS)
        end
        save to obj.resource_grid_CSIRS, obj.CSIRS_position
      end
    end
end
```

- **Resource allocation module**

- This module stores PSS, SSS, PBCH, PDCCH, PDSCH positions on the Resource grid as a matrix with the number of subcarriers X the number of OFDM symbols.

- PSS, SSS and PBCH are located in each SS / PBCH block consisting of 4 consecutive OFDM symbols. The location of the SS / PBCH on the resource grid is defined in TS 38.213 [3], and the location of PSS, SSS and PBCH in each SS / PBCH block is defined in TS 38.211 [1].

- As the modulation scheme of PBCH is QPSK, it generates QPSK symbol arbitrarily and puts it into resource grid corresponding to PBCH position

- PDCCH is located in slot 0 of slots. Since the modulation scheme of the PDCCH is QPSK, a QPSK symbol is arbitrarily generated and put into a resource grid corresponding to the PDCCH position.

- PDSCH position is allocated to the remaining positions after all the positions of CRS, PBCH, PDCCH, PSS, and SSS are allocated and it is stored as Data_position.

- After determining the number of coded bits using the data position, determine the number of data bits by considering coding rate and crc for each codeword.

method_resource_allocation(obj,para,ind_slot)
Input : ind_slot, parameters for frame structure, resource allocation, transmission (4.1.1-a.1, 3, 10) Output : obj.M_order, obj.Coding_rate, obj.Control_position_SISO, obj.RS_position_SISO, obj.Data_position

- **Sync generation module**

- Physical-layer cell identities: there are 1008 unique physical-layer cell identities given by

$$N_{\mathrm{ID}}^{\mathrm{cell}} = 3N_{\mathrm{ID}}^{(1)} + N_{\mathrm{ID}}^{(2)}$$

where $N_{\mathrm{ID}}^{(1)} \in \{0,1,\ldots,335\}$ and $N_{\mathrm{ID}}^{(2)} \in \{0,1,2\}$.

- There are two types of synchronization signals: Primary synchronization signal (PSS) and Secondary synchronization signal (SSS).
- The sequence $d_{\mathrm{PSS}}(n)$ for the primary synchronization signal is defined by

$$d_{\mathrm{PSS}}(n) = 1 - 2x(m)$$
$$m = \left(n + 43N_{\mathrm{ID}}^{(2)}\right) \bmod 127$$
$$0 \le n < 127$$

where

$$x(i+7) = \left(x(i+4) + x(i)\right) \bmod 2$$

and

$$[x(6) \quad x(5) \quad x(4) \quad x(3) \quad x(2) \quad x(1) \quad x(0)] = [1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0].$$

- The sequence $d_{\mathrm{SSS}}(n)$ for the secondary synchronization signal is defined by

$$d_{\mathrm{SSS}}(n) = \left[1 - 2x_0\left((n + m_0) \bmod 127\right)\right]\left[1 - 2x_1\left((n + m_1) \bmod 127\right)\right]$$
$$m_0 = 15\left\lfloor \frac{N_{\mathrm{ID}}^{(1)}}{112} \right\rfloor + 5N_{\mathrm{ID}}^{(2)}$$
$$m_1 = N_{\mathrm{ID}}^{(1)} \bmod 112$$
$$0 \le n < 127$$

where

$$x_0(i+7) = \left(x_0(i+4) + x_0(i)\right) \bmod 2$$
$$x_1(i+7) = \left(x_1(i+1) + x_1(i)\right) \bmod 2$$

and

$$\left[x_0(6) \quad x_0(5) \quad x_0(4) \quad x_0(3) \quad x_0(2) \quad x_0(1) \quad x_0(0)\right] = \left[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1\right]$$
$$\left[x_1(6) \quad x_1(5) \quad x_1(4) \quad x_1(3) \quad x_1(2) \quad x_1(1) \quad x_1(0)\right] = \left[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1\right]$$

- Mapping to physical resources of PSS and SSSS are described in clause 7.4.3 of TS 38.211 [1].

method_sync_generation(obj,para,ind_slot)
Input : ind_slot, parameters for synchronization (4.1.1-a.7), para.cell_ID Output : obj.SSS_signal, obj.PSS_signal,

- **HARQ process**

- Function module in the BS for HARQ. It is possible to set up the maximum number of HARQ processes and retransmissions on the simulator, and the HARQ process proceeds according to those values.
- When a NACK is received from UE, retransmission is performed 8ms after as default value. The information necessary for retransmission is stored in the HARQ_buffer and signaled during retransmission to inform UE of whether retransmission is needed through the new data indicator.

- When an ACK is received from UE, all the values stored in the buffer of the corresponding HARQ process are initialized. It is initialized as well even in the case that the number of retransmission exceeds the maximum.

method_HARQ_process (obj, ind_slot, feedback)
Input : ind_slot, feedback, obj.HARQ_process_index Output : obj.HARQ_buffer

- **Bit generation module**

- Generates a bit stream for the number of data bits determined in the Resource allocation module for each codeword.

method_bit_generation(obj, para)
Input : **para.num_codewords** Output : **obj.bit_stream**

- **Channel encoding module**

- Figure 2 is the structure and procedure of channel encoding module based on section 6.2 and 7.2 in 3GPP TS 38.212 [2]. The encoding module consists of detail modules – CRC calculation, code block segmentation, LDPC encoding, rate matching, and code block concatenation

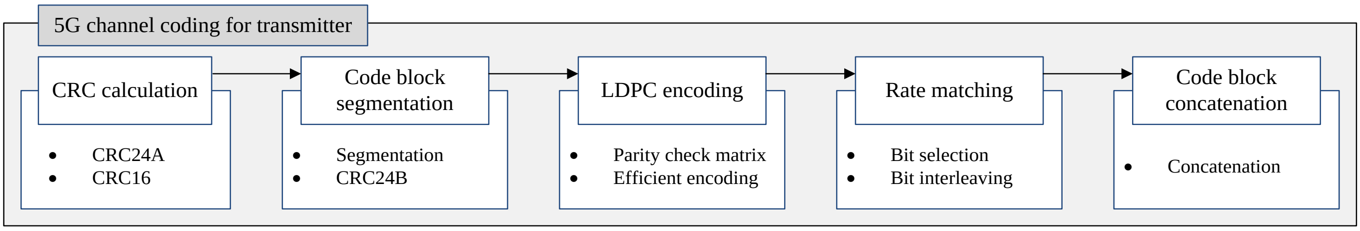

Figure 3. Structure and procedure of channel encoding module.

- In CRC calculation, the input bits and the parity bits are denoted by $a_0, a_1, a_2, a_3, ..., a_{A-1}$ and $p_0, p_1, p_2, p_3, ..., p_{L-1}$, respectively, where A and L are the size of the input bits and output bits, respectively. One of the following cyclic generator polynomials generator the parity bits for CRC:

  - $g_{\mathrm{CRC24A}}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^{7} + D^{6} + D^{5} + D^{4} + D^{3} + D + 1]$ for a CRC length $L = 24$
  - $g_{\mathrm{CRC24B}}(D) = [D^{24} + D^{23} + D^{6} + D^{5} + D + 1]$ for a CRC length $L = 24$
  - $g_{\mathrm{CRC24C}}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^{8} + D^{4} + D^{2} + D + 1]$ for a CRC length $L = 24$
  - $g_{\mathrm{CRC16}}(D) = [D^{16} + D^{12} + D^{5} + D + 1]$ for a CRC length $L = 16$

The input bits are systematically encoded in GF(2), and the polynomial is represented as below

$$a_0D^{A+L-1} + a_1D^{A+L-2} + ... + a_{A-1}D^L + p_0D^{L-1} + p_1D^{L-2} + ... + p_{L-2}D^1 + p_{L-1}$$

which is divided by the corresponding CRC generator polynomial (a remainder equals 0).
The bits attached by CRC are denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$, where $B = A + L$. The bits are represented as below

$$b_k = a_k \quad \text{for} \quad k = 0,1,2,..., A-1$$

$$b_k = p_{k-A} \text{ for } \quad k = A, A+1, A+2,..., A+L-1.$$

The LDPC base graph is selected as following:

- LDPC base graph 2, if $A \le 292$, or if $A \le 3824$ and $R \le 0.67$, or if $R \le 0.25$
- LDPC base graph 1, otherwise

- In code block segmentation, for the input bits denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$, if the size B is larger than the maximum code block size K_{cb}, the input bits are segmented as several code blocks. Then CRC bits of size $L = 24$ are added to each code block. The maximum code block size K_{cb} is determined by

LDPC base graph type as following:

- $K_{cb} = 8448$ for LDPC base graph 1
- $K_{cb} = 3840$ for LDPC base graph 2

Total number of code blocks C is determined as below:

- if $B \le K_{cb}$, the number of code blocks is one ($C = 1$), CRC bits are not added ($L = 0$) and $B' = B$,
- otherwise the number of code blocks is more than one ($C = \lceil B/(K_{cb} - L) \rceil$),CRC bits are added ($L = 24$) and $B' = B + C \cdot L$

The output bits of code block r after code block segmentation is $c_{r0}, c_{r1}, c_{r2}, c_{r3}, ..., c_{r(K_r-1)}$, where $r = 0,1,..., C-1$ is the code block number, and K_r is the size of bits for the code block r.
For each code block, the number of bits K is determined as following:

- $K' = B'/C$
- For LDPC base graph 1, $K_b = 22$.
- For LDPC base graph 2, if $B > 640$, then $K_b = 10$, or if $B > 560$, then $K_b = 9$, or if $B > 192$, then $K_b = 8$, or otherwise, $K_b = 6$

Z_c is determined by finding the minimum value in all sets of lifting sizes in Table 1 on condition that

$K_b \cdot Z_c \geq K'$. K is denoted by $22Z_c$ for LDPC based graph 1 and $10Z_c$ for LDPC base graph 2, respectively.

Table 1. Sets of LDPC lifting size Z.

Set index (i_{LS})	Set of lifting sizes (Z)
1	{2, 4, 8, 16, 32, 64, 128, 256}
2	{3, 6, 12, 24, 48, 96, 192, 384}
3	{5, 10, 20, 40, 80, 160, 320}
4	{7, 14, 28, 56, 112, 224}
5	{9, 18, 36, 72, 144, 288}
6	{11, 22, 44, 88, 176, 352}
7	{13, 26, 52, 104, 208}
8	{15, 30, 60, 120, 240}

For each code block, the output bits for the code block r are determined as following procedure:

$s = 0$

for $r = 0$ to $C - 1$

  for $k = 0$ to $K' - L - 1$

    $c_{rk} = b_s$

    $s = s + 1$

  end for

  if $C > 1$

    The sequence $c_{r0}, c_{r1}, c_{r2}, c_{r3}, \ldots, c_{r(K'-L-1)}$ is used to calculate the CRC parity bits $p_{r0}, p_{r1}, p_{r2}, \ldots, p_{r(L-1)}$ according to section 5.1 with the generator polynomial $g_{\mathrm{CRC24B}}(D)$.

    for $k = K' - L$ to $K' - 1$

      $c_{rk} = p_{r(k+L-K')}$

    end for

  end if

  for $k = K'$ to $K - 1$ -- Insertion of filler bits

    $c_{rk} = <NULL>$

  end for

end for

To the deal with *NULL* bits, -1 is inserted for simulations.

- In LDPC encoding, the input bits for any code block r $(0 \le r < C)$ to channel coding is denoted by $c_0, c_1, c_2, c_3, ..., c_{K-1}$, where K is the size of bits to encode. The output bits are denoted by $d_0, d_1, d_2, ..., d_{N-1}$, where N is the size of the encoded bits, which is $66Z_c$ for LDPC base graph 1 and $50Z_c$ for LDPC base graph 2.
  For each code block, the encoding procedure applies:

1) Find the set index i_{LS} in Table 1
 	
2) for $k = 2Z_c$ to $K-1$
 	
if $c_k \ne$ < $NULL$ >
 	
$d_{k-2Z_c} = c_k$
 	
else
 	
$c_k = 0$
 	
$d_{k-2Z_c} =$ < $NULL$ >
 	
end if
end for
3) Generate $N+2Z_c-K$ parity bits $\mathbf{w} = [w_0, w_1, w_2, ..., w_{N+2Z_c-K-1}]^T$ such that $\mathbf{H} \times \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \mathbf{0}$, where
 	
$\mathbf{c} = [c_0, c_1, c_2, ..., c_{K-1}]^T$. $\mathbf{0}$ is a column vector of all elements equal to 0.
The encoding is performed in GF(2).
4) for $k = K$ to $N+2Z_c-1$
 	
$d_{k-2Z_c} = w_{k-K}$
 	
end for

For LDPC, $\mathbf{H}_{BG}$ is LDPC base graph consisting of zero matrices and circular permutation matrices. For base graph 1, $\mathbf{H}_{BG}$ has 46 rows and 68 columns with row indices $i = 0,1,...,45$ and column indices $j = 0,1,...,67$. For LDPC base graph 2, a matrix of $\mathbf{H}_{BG}$ has 42 rows and 52 columns with row indices $i = 0,1,...,41$ and column indices $j = 0,1,...,51$. The (i, j) th element of $\mathbf{H}_{BG}$ has 1, where row and column indices are given in [Table 5.3.2-2, x1] (for LDPC base graph 1) and [Table 5.3.2-3, x1] (for LDPC base graph 2), and all the others are zero.

The parity check matrix $\mathbf{H}$ is obtained by replacing each element of $\mathbf{H}_{BG}$ with a $Z_c \times Z_c$ matrix, according to the following:

1) Each element of value 0 in $\mathbf{H}_{BG}$ is replaced by a $Z_c \times Z_c$ zero matrix $\mathbf{0}$ 2) Each element of value 1 in $\mathbf{H}_{BG}$ is replaced by a $Z_c \times Z_c$ circular permutation matrix $\mathbf{I}(P_{i,j})$, and $\mathbf{I}(P_{i,j})$ is obtained by circularly shifting the $Z_c \times Z_c$ identity matrix $\mathbf{I}$ to the right $P_{i,j}$ times. The

value of $P_{i,j}$ is given by $P_{i,j} = \mathrm{mod}\left(V_{i,j}, Z_c\right)$. The value of $V_{i,j}$ is given by Tables 2 and 3 according to the set index i_{LS} and base graph.

Specifically, the input bits can be encoded by efficient encoding method from the structure of parity check matrix.

From the equation $\mathbf{H} \times \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \mathbf{0}$, the parity bits sequence $\mathbf{W}$ is obtained, which is separated by $\mathbf{w}_1$ and $\mathbf{w}_2$, where the length of $\mathbf{w}_1$ corresponds to the number of columns of $\mathbf{B}$, and the length of $\mathbf{w}_2$ corresponds to the number of columns of $\mathbf{O}$. From the figure 2, $\mathbf{H} \times \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \mathbf{0}$ is rewritten as below:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{O} \\ & \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix} = \mathbf{0}$$

It is note that $\mathbf{O}$ is zero matrix and $\mathbf{I}$ is identity matrix. From the structure of parity check matrix, the efficient encoding method can be performed as following:

$$\begin{bmatrix} \mathbf{Ac} + \mathbf{Bw}_1 \\ \mathbf{C} \times [\mathbf{c}^{\mathrm{T}} \quad \mathbf{w}_1^{\mathrm{T}}]^{\mathrm{T}} + \mathbf{w}_2 \end{bmatrix} = \mathbf{0}$$

$$\mathbf{w}_1 = -\mathbf{B}^{-1}\mathbf{Ac}$$

$$\mathbf{w}_2 = -\mathbf{C} \times [\mathbf{c}^{\mathrm{T}} \quad \mathbf{w}_1^{\mathrm{T}}]^{\mathrm{T}}$$

All operations are performed in GF(2).

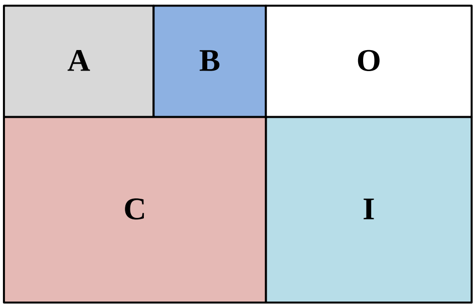

Figure 4. Structure of parity check matrix for LDPC.

- In rate matching, the rate matching is applied per coded block and consists of 2 steps - bit selection and bit interleaving. The input bit sequence is denoted by $d_0, d_1, d_2, \ldots, d_{N-1}$, and the output bit sequence is denoted by $f_0, f_1, f_2, \ldots, f_{E-1}$.

  In bit selection, the bit sequence $d_0, d_1, d_2, \ldots, d_{N-1}$ is inserted into a circular buffer of length N for coded block r. For coded block r, if $I_{LBRM} = 0$, then $N_{cb} = N$, and otherwise, $N_{cb} = \min\left(N, N_{ref}\right)$, where $N_{ref} = \left\lfloor \frac{TBS_{\mathrm{LBRM}}}{C \cdot R_{\mathrm{LBRM}}} \right\rfloor$, $R_{\mathrm{LBRM}} = 2/3$, and TBS_{LBRM} is determined according to [4] assuming the following, referring [2]:
  - maximum number of layers for one transport block supported by the UE for the serving cell, which for UL-SCH is according to higher layer parameter *ULmaxRank* if the parameter is

configured

- maximum modulation order configured for the serving cell, if configured by higher layers; otherwise a maximum modulation order is assumed for DL-SCH
- maximum coding rate of 948/1024
- $n_{PRB} = n_{PRB,LBRM}$ is given by Table 2, where the value of $n_{PRB,LBRM}$ for DL-SCH is determined according to the initial bandwidth part if there is no other bandwidth part configured to the UE;
- $N_{RE} = 156 n_{PRB}$

Table 2. Value of $n_{PRB,LBRM}$.

Maximum number of PRBs across all configured BWPs of a carrier	$n_{PRB,LBRM}$
Less than 33	32
33 to 66	66
67 to 107	107
108 to 135	135
136 to 162	162
163 to 217	217
Larger than 217	273

The rate matching output sequence length for the coded block r is denoted by E_r and determined as below

Set $j = 0$
for $r = 0$ to $C-1$
if the coded block r is not for transmission as indicated by CBGTI according to [4]
$E_r = 0$
else
if $j \le C' - \mathrm{mod}(G/(N_L \cdot Q_m), C') - 1$
$$E_r = N_L \cdot Q_m \cdot \left\lfloor \frac{G}{N_L \cdot Q_m \cdot C'} \right\rfloor$$
else
$$E_r = N_L \cdot Q_m \cdot \left\lceil \frac{G}{N_L \cdot Q_m \cdot C'} \right\rceil$$
end if
$j = j + 1$
end if
end for

where , referring [2],

- N_L is the number of transmission layers that the transport block is mapped onto
- Q_m is the modulation order
- G is the total number of coded bits available for transmission of the transport block
- $C' = C$ if CBGTI is not present in the DCI scheduling the transport block and C' is the number of scheduled code blocks of the transport block if CBGTI is present in the DCI scheduling the transport block

The rate matching output bit sequence e_k is generated as follows, where k_0 is given by Table 3 according to the value of the redundancy version number for this transmission rv_{id} = 0, 1, 2 or 3

```
k = 0, j = 0
while k < E
      if d_{(k_0+j) mod N_cb} ≠ <NULL>
            e_k = d_{(k_0+j) mod N_cb}
            k = k + 1;
      end if
      j = j + 1
end while
```

Table 3. Starting position of different redundancy versions, k_0.

rv_{id}	k_0	
	Base graph 1	Base graph 2
0	0	0
1	$\left\lfloor \frac{17N_{cb}}{66Z_c} \right\rfloor Z_c$	$\left\lfloor \frac{13N_{cb}}{50Z_c} \right\rfloor Z_c$
2	$\left\lfloor \frac{33N_{cb}}{66Z_c} \right\rfloor Z_c$	$\left\lfloor \frac{25N_{cb}}{50Z_c} \right\rfloor Z_c$
3	$\left\lfloor \frac{56N_{cb}}{66Z_c} \right\rfloor Z_c$	$\left\lfloor \frac{43N_{cb}}{50Z_c} \right\rfloor Z_c$

In bit interleaving, the bit sequence $e_0, e_1, e_2, ..., e_{E-1}$ is interleaved to bit sequence $f_0, f_1, f_2, ..., f_{E-1}$, as follows, where Q_m is the number of coded bits per QAM symbol given by Table 4.

```
for j = 0 to E/Q_m - 1
      for i = 0 to Q_m - 1
            f_{i+j·Q_m} = e_{i·E/Q_m+j}
      end for
end for
```

- In code block concatenation, the input sequences is f_{rk} for $r = 0,...,C-1$ and $k = 0,..., E_r - 1)$ and the output bit sequence is g_k for $k = 0,...,G-1$. The rate matching outputs are concatenated as below:

```
k = 0
for r = 0 to C - 1
      for j = 0 to E_r - 1
            g_k = f_rj
            k = k + 1
      end for
end for
```

method_channel_coding(obj,para)
Input : **obj.bit_stream,** obj.M_order,obj.Coding_rate, parameters for transmission (4.1.1-a.10) Output : **obj. coded_bit_stream,** obj.ch_coding

- **Scrambling module**

- For each codeword q, a pseudo-random sequence defined in TS.38.211 [1] is generated using c_{init} and codeword length satisfying the following equation.

$$c_{\text{init}} = n_{\text{RNTI}} \cdot 2^{15} + q \cdot 2^{14} + n_{\text{ID}}$$

- Creates scrambled_bit_stream by performing mod2 operation on codeword input value Coded_bit_stream and pseudo-random sequence

method_scrambling(obj, para, ind_slot)
Input : **obj.coded_bit_stream**, obj.n_RNTI, parameters for transmission (4.1.1-a.10) Output : **obj.scrambled_bit_stream**

- **Modulation mapping module**

- A module that maps scrambled bit sequences generated according to the standard document [1,3] to modulation symbols according to modulation order determined by CQI. Modulation mapping is realized in the case that the modulation order is set to 2, 4, 6, 8 in CQI table of 5.2.2.1-2 of [4]. In the case of mapping to modulation symbols according to modulation order, mapping is performed according to 7.1.2 ~ 5-1 of [1].

method_modulation_mapping(obj)
Input : **obj.scrambled_bit_stream,** obj.M_order, obj.bit_stream Output : **obj.modulated_signal_stream**

- **Data mapping module**

- Assign Modulated_signal_stream to the resource grid. Data is allocated in the order of the slots.

method_Data_mapping(obj,para)
Input : **obj.modulated_signal_stream, obj.Data_position,** Output : **obj.resource_grid**

- **Power allocation module**

- Assigns powers to the RSs and data signals allocated in the resource grid following the rule in 4.1 [4]. Basically, power allocation is determined from MAC layer parameters, so our simulator receives power of RSs and data signals from the user.

method_power_allocation(obj, para)
Input : parameters for power allocation (4.1.1-a.12), obj.Data_position, **obj.resource_grid** Output : **obj.resource_grid,** obj.resource_grid_DMRS, obj.resource_grid_CSIRS

- **OFDM generation module**

- Converts mapped signals to the OFDM signals and add cyclic prefix. Zero padding and DC carrier is also performed during converting. From 5.3 in [1], subcarrier spacing is μ and lth OFDM signal from antenna port p, $s_l^{(p,\mu)}(t)$ is defined as follows

$$s_l^{(p,\mu)}(t) = \sum_{k=0}^{N_{\mathrm{grid}}^{\mathrm{size},\mu} N_{\mathrm{sc}}^{\mathrm{RB}} - 1} a_{k,l}^{(p,\mu)} \cdot e^{j2\pi\left(k + k_0^{\mu} - N_{\mathrm{grid}}^{\mathrm{size},\mu} N_{\mathrm{sc}}^{\mathrm{RB}}/2\right)\Delta f\left(t - N_{\mathrm{CP},l}^{\mu} T_{\mathrm{c}}\right)}$$

where $0 \le t < \left(N_{\mathrm{u}}^{\mu} + N_{\mathrm{CP},l}^{\mu}\right) T_{\mathrm{c}}$.

- $a_{k,l}^{(p,\mu)}$ indicates value in the position of (k, l) in the resource grid.
- The position of the lth OFDM signal is determined as follows:

$$t_{\mathrm{start},l}^{\mu} = \begin{cases} 0 & l = 0 \\ t_{\mathrm{start},l-1}^{\mu} + \left(N_{\mathrm{u}}^{\mu} + N_{\mathrm{CP},l-1}^{\mu}\right) \cdot T_{\mathrm{c}} & \text{otherwise} \end{cases}$$

where,

$$N_{\mathrm{u}}^{\mu} = 2048\kappa \cdot 2^{-\mu}$$
$$N_{\mathrm{CP},l}^{\mu} = \begin{cases} 512\kappa \cdot 2^{-\mu} & \text{extended cyclic prefix} \\ 144\kappa \cdot 2^{-\mu} + 16\kappa & \text{normal cyclic prefix, } l = 0 \text{ or } l = 7 \cdot 2^{\mu} \\ 144\kappa \cdot 2^{-\mu} & \text{normal cyclic prefix, } l \ne 0 \text{ and } l \ne 7 \cdot 2^{\mu} \end{cases}.$$

method_OFDM_generation(obj, para)
Input : parameters for frame structure (4.1.1-a.1), parameters for OFDM (4.1.1-a.13), **obj.resource_grid** Output : **obj.OFDM_signal_stream**

4.1.2-b UE_SISO function block submodules for Uplink (common parts with downlink case are omitted)

- **Feedback reception module**

- The module for the feedback reception: The module received CQI from an BS and decide modulation order and coding rate, and so on. The CQI table is generated from Table 5.2.2.1-1 in 3GPP TS 38.214 V1.2.0 (2017-11) .

method_feedback_reception(obj, para, feedback, ch_output)
Input : feedback.CQI, para.zero_TTI_feedback, para.Channel_type, para.sigma_n_freq, para.M_order, para.Coding_rate Output : obj.CQI

- **RS_allocation module**

- Generates DM-RS and allocated the DM-RS to the resource grid according to clause 6.4.1.1 in TS 38.211. If transform precoding is enabled, Low PAPR sequence generation is performed instead of pseudo sequence generation as clause 5.2.2 in TS 38.211.
- Generate SRS and allocated the SRS to the resource grid according to clause 6.4.1.4 in TS 38.211.

method_RS_allocation(obj, para, ind_slot)
Input : ind_slot, parameters for reference signal of parameter class (4.1.1-a.2) Output : obj.DMRS_position_total, obj.DMRS_position, obj.resource_grid_DMRS, obj.SRS_position_total, obj.resource_grid_SRS, obj.SRS_position Function : func_pseudo_sequence_generation, func_Low_PAPR_sequence_generation

<DMRS part>
1. Initialization of output matrix
2. Define location of starting subcarrier according to DMRS configuration type
for (from 1 to length of DMRS port set)
% In SISO case, length of DMRS port set is always ‘1’.
if (transform precoding disabled)
 	
if (single-symbol DMRS)
Set $\bar{l}$ according to Table 6.4.1.1.3 in TS 38.211 (switch)
elseif (double-symbol DMRS)
Set $\bar{l}$ according to Table 6.4.1.1.3 in TS 38.211 (switch)
else return error;
end
 	
if (UL DMRS configuration type 1),
Set Δ, w_f and w_t according to Table 6.4.1.1.3 in TS 38.211 (switch)
pseudo sequence generation using function func_pseudo_sequence_generation (if transform precoding disabled)
extract pseudo sequence actually used or function func_Low_PAPR_sequence_generation (if transform precoding enabled)
 	
if (PUSCH mapping type is A),
 	
for (from 1 to actual pseudo sequence length)
% Generate of DM-RS position according to 7.4.1.1.2 of TS 38.211
generate a vaule that matches ‘n’ in 6.4.1.1.3 of TS 38.211
if ($k' = 0$)
Calculate ‘k’
Input the $\alpha_{k,l}^{(p,\mu)}$ to corresponding position $(k, l_0 + 1)$ and $(k, l_0 + 2)$
Save that the positions are used : ‘true’
else ($k' = 1$)
Performs the same operation as when $k' = 0$.
end
end
 	
if ($\bar{l}$ consists of l_0 and other components)

```
        if (single-symbol DM-RS)
          for (1 to the number of additional components)
            Insert single-symbol to corresponding position
            Save that the positions are used : 'true'
          end
        else if (double-symbol DM-RS)
          for (1 to the number of additional components)
            Insert double-symbol to corresponding position
            Save that the positions are used : 'true'
          end
        end
      end

      for (slot index is 2 to the number of slots in a subframe)
        for (OFDM symbol index is 1 to the number of symbols in a slot)
        end
      end
    else if (PUSCH mapping type is B)
      Setting l_0 to '0'
      for (from 1 to actual pseudo sequence length)
        See the case of 'PUSCH mapping type is A'
      end

      if (Single-symbol DM-RS & Duration of PUSCH transmission is '7')
        % PUSCH mapping type B, Table 6.4.1.1.3 in TS 38.211
        Insert single-symbol to corresponding position
        Save that the positions are used : 'true'
      end
    end
  else if (UL DMRS configuration type 2),
    Set Δ, w_f and w_t according to Table 6.4.1.1.3 in TS 38.211 (switch)
    pseudo sequence generation
    Same procedure as UL DMRS configuration type1
  end
  save to obj.resource_grid_DMRS_subframe, obj.DMRS_position_subframe
  elseif (transform precoding enabled)
    similar procedure with above except for detailed values following 6.4.1.1.3 in TS
  38.211 [1]

end

Convert the unit of resource grid and DMRS position matrix from subframe to slot.
```

- **Resource allocation module**

- This module stores PUSCH positions on the Resource grid as a matrix with the number of subcarriers X the number of OFDM symbols.

- PUSCH position is allocated to the remaining positions after all the positions of SRS and DMRS are allocated and it is stored as Data_position.

- After determining the number of coded bits using the data position, determine the number of data bits by considering coding rate and crc for each codeword.

method_resource_allocation(obj,para,ind_slot)
Input : ind_slot, parameters for frame structure, resource allocation, transmission (4.1.1-b.1, 3, 10) Output : obj.M_order, obj.Coding_rate, obj.RS_position_SISO, obj.Data_position

- **HARQ process**

- Function module in the UE class for HARQ. It is possible to set up the maximum number of HARQ processes and retransmissions on the simulator, and the HARQ process proceeds according to those values.
- When a NACK is received from BS, retransmission is performed 8ms after as default value. The information necessary for retransmission is stored in the HARQ_buffer and signaled during retransmission to inform BS of whether retransmission is needed through the new data indicator.
- When an ACK is received from BS, all the values stored in the buffer of the corresponding HARQ process are initialized. It is initialized as well even in the case that the number of retransmission exceeds the maximum.
-

method_HARQ_process (obj, ind_slot, feedback)
Input : ind_slot, feedback, obj.HARQ_process_index Output : obj.HARQ_buffer

- **Bit generation module**

- Generates a bit stream for the number of data bits determined in the Resource allocation module for each codeword.

method_bit_generation(obj, para)
Input : **para.num_codewords** Output : **obj.bit_stream**

- **Channel encoding module**

- Figure 5 is the structure and procedure of channel encoding module based on section 6.2 and 7.2 in 3GPP TS 38.212 [2]. The encoding module consists of detail modules – CRC calculation, code block segmentation, LDPC encoding, rate matching, and code block concatenation

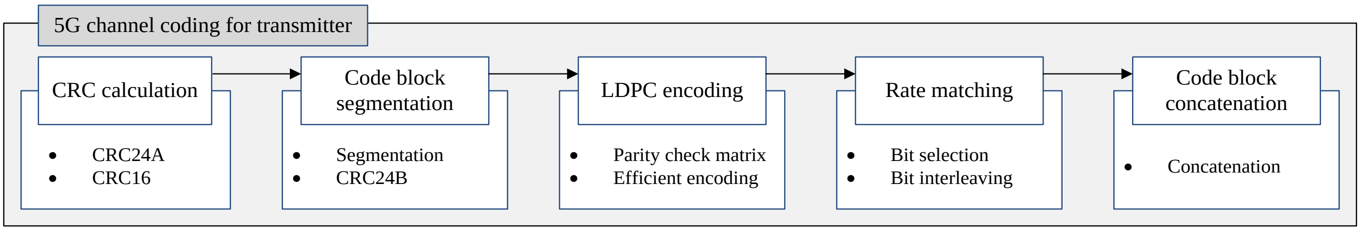

Figure 6. Structure and procedure of channel encoding module.

- In CRC calculation, the input bits and the parity bits are denoted by $a_0, a_1, a_2, a_3, ..., a_{A-1}$ and $p_0, p_1, p_2, p_3, ..., p_{L-1}$, respectively, where A and L are the size of the input bits and output bits, respectively. One of the following cyclic generator polynomials generator the parity bits for CRC:

  - $g_{\mathrm{CRC24A}}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^{7} + D^{6} + D^{5} + D^{4} + D^{3} + D + 1]$ for a CRC length $L = 24$
  - $g_{\mathrm{CRC24B}}(D) = [D^{24} + D^{23} + D^{6} + D^{5} + D + 1]$ for a CRC length $L = 24$
  - $g_{\mathrm{CRC24C}}(D) = [D^{24} + D^{23} + D^{21} + D^{20} + D^{17} + D^{15} + D^{13} + D^{12} + D^{8} + D^{4} + D^{2} + D + 1]$ for a CRC length $L = 24$
  - $g_{\mathrm{CRC16}}(D) = [D^{16} + D^{12} + D^{5} + D + 1]$ for a CRC length $L = 16$

The input bits are systematically encoded in GF(2), and the polynomial is represented as below

$$a_0 D^{A+L-1} + a_1 D^{A+L-2} + ... + a_{A-1} D^{L} + p_0 D^{L-1} + p_1 D^{L-2} + ... + p_{L-2} D^{1} + p_{L-1}$$

which is divided by the corresponding CRC generator polynomial (a remainder equals 0).
The bits attached by CRC are denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$, where $B = A + L$. The bits are represented as below

$$b_k = a_k \quad \text{for} \quad k = 0,1,2,...,A-1$$

$$b_k = p_{k-A} \text{ for } k = A, A+1, A+2, ..., A+L-1.$$

The LDPC base graph is selected as following:

- LDPC base graph 2, if $A \le 292$, or if $A \le 3824$ and $R \le 0.67$, or if $R \le 0.25$
- LDPC base graph 1, otherwise

- In code block segmentation, for the input bits denoted by $b_0, b_1, b_2, b_3, ..., b_{B-1}$, if the size B is larger than the maximum code block size K_{cb}, the input bits are segmented as several code blocks. Then CRC bits of size $L = 24$ are added to each code block. The maximum code block size K_{cb} is determined by

  LDPC base graph type as following:

  - $K_{cb} = 8448$ for LDPC base graph 1
  - $K_{cb} = 3840$ for LDPC base graph 2

  Total number of code blocks C is determined as below:

  - if $B \le K_{cb}$, the number of code blocks is one ($C = 1$), CRC bits are not added ($L = 0$) and $B' = B$,
  - otherwise the number of code blocks is more than one ($C = \lceil B/(K_{cb} - L) \rceil$),CRC bits are added ($L = 24$) and $B' = B + C \cdot L$

The output bits of code block r after code block segmentation is $c_{r0}, c_{r1}, c_{r2}, c_{r3}, ..., c_{r(K_r-1)}$, where $r = 0, 1, ..., C-1$ is the code block number, and K_r is the size of bits for the code block r.

For each code block, the number of bits K is determined as following:

- $K' = B'/C$
- For LDPC base graph 1, $K_b = 22$.
- For LDPC base graph 2, if $B > 640$, then $K_b = 10$, or if $B > 560$, then $K_b = 9$, or if $B > 192$, then $K_b = 8$, or otherwise, $K_b = 6$

Z_c is determined by finding the minimum value in all sets of lifting sizes in Table 1 on condition that $K_b \cdot Z_c \ge K'$. K is denoted by $22Z_c$ for LDPC based graph 1 and $10Z_c$ for LDPC base graph 2, respectively.

Table 1. Sets of LDPC lifting size Z.

Set index (i_{LS})	Set of lifting sizes (Z)
1	{2, 4, 8, 16, 32, 64, 128, 256}
2	{3, 6, 12, 24, 48, 96, 192, 384}
3	{5, 10, 20, 40, 80, 160, 320}
4	{7, 14, 28, 56, 112, 224}
5	{9, 18, 36, 72, 144, 288}
6	{11, 22, 44, 88, 176, 352}
7	{13, 26, 52, 104, 208}
8	{15, 30, 60, 120, 240}

For each code block, the output bits for the code block r are determined as following procedure:

$s=0$

for $r=0$ to $C-1$

for $k=0$ to $K'-L-1$

$c_{rk}=b_s$

$s=s+1$

end for

if $C>1$

The sequence $c_{r0},c_{r1},c_{r2},c_{r3},...,c_{r(K'-L-1)}$ is used to calculate the CRC parity bits $p_{r0},p_{r1},p_{r2},...,p_{r(L-1)}$ according to section 5.1 with the generator polynomial $g_{\text{CRC24B}}(D)$.

for $k=K'-L$ to $K'-1$

$c_{rk}=p_{r(k+L-K')}$

end for

end if

for $k=K'$ to $K-1$ -- Insertion of filler bits

$c_{rk}=\text{<}NULL\text{>}$

end for

end for

To the deal with *NULL* bits, -1 is inserted for simulations.

- In LDPC encoding, the input bits for any code block r $(0\le r<C)$ to channel coding is denoted by $c_0,c_1,c_2,c_3,...,c_{K-1}$, where K is the size of bits to encode. The output bits are denoted by $d_0,d_1,d_2,...,d_{N-1}$, where N is the size of the encoded bits, which is $66Z_c$ for LDPC base graph 1 and $50Z_c$ for LDPC base graph 2.

For each code block, the encoding procedure applies:

1) Find the set index i_{LS} in Table 1

2) for $k=2Z_c$ to $K-1$

if $c_k\ne\text{<}NULL\text{>}$

$d_{k-2Z_c}=c_k$

else

$c_k = 0$

$d_{k-2Z_c} = <NULL>$

end if

end for

3) Generate $N + 2Z_c - K$ parity bits $\mathbf{w} = \left[w_0, w_1, w_2, ..., w_{N+2Z_c-K-1}\right]^T$ such that $\mathbf{H} \times \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \mathbf{0}$, where $\mathbf{c} = [c_0, c_1, c_2, ..., c_{K-1}]^T$. $\mathbf{0}$ is a column vector of all elements equal to 0. The encoding is performed in GF(2).

4) for $k = K$ to $N + 2Z_c - 1$

$d_{k-2Z_c} = w_{k-K}$

end for

For LDPC, $\mathbf{H}_{\text{BG}}$ is LDPC base graph consisting of zero matrices and circular permutation matrices. For base graph 1, $\mathbf{H}_{\text{BG}}$ has 46 rows and 68 columns with row indices $i = 0,1,...,45$ and column indices $j = 0,1,...,67$. For LDPC base graph 2, a matrix of $\mathbf{H}_{\text{BG}}$ has 42 rows and 52 columns with row indices $i = 0,1,...,41$ and column indices $j = 0,1,...,51$. The (i, j) th element of $\mathbf{H}_{\text{BG}}$ has 1, where row and column indices are given in [Table 5.3.2-2, 2] (for LDPC base graph 1) and [Table 5.3.2-3, 4] (for LDPC base graph 2), and all the others are zero.

The parity check matrix $\mathbf{H}$ is obtained by replacing each element of $\mathbf{H}_{\text{BG}}$ with a $Z_c \times Z_c$ matrix, according to the following:

1) Each element of value 0 in $\mathbf{H}_{\text{BG}}$ is replaced by a $Z_c \times Z_c$ zero matrix $\mathbf{0}$

2) Each element of value 1 in $\mathbf{H}_{\text{BG}}$ is replaced by a $Z_c \times Z_c$ circular permutation matrix $\mathbf{I}(P_{i,j})$, and $\mathbf{I}(P_{i,j})$ is obtained by circularly shifting the $Z_c \times Z_c$ identity matrix $\mathbf{I}$ to the right $P_{i,j}$ times. The value of $P_{i,j}$ is given by $P_{i,j} = \text{mod}(V_{i,j}, Z_c)$. The value of $V_{i,j}$ is given by Tables 2 and 3 according to the set index i_{LS} and base graph.

Specifically, the input bits can be encoded by efficient encoding method from the structure of parity check matrix.

From the equation $\mathbf{H} \times \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \mathbf{0}$, the parity bits sequence $\mathbf{W}$ is obtained, which is separated by $\mathbf{w}_1$ and $\mathbf{w}_2$, where the length of $\mathbf{w}_1$ corresponds to the number of columns of $\mathbf{B}$, and the length of $\mathbf{w}_2$ corresponds to the number of columns of $\mathbf{O}$. From the figure 2, $\mathbf{H} \times \begin{bmatrix} \mathbf{c} \\ \mathbf{w} \end{bmatrix} = \mathbf{0}$ is rewritten as below:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{O} \\ & \mathbf{C} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{c} \\ \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix} = \mathbf{0}$$

It is note that $\mathbf{O}$ is zero matrix and $\mathbf{I}$ is identity matrix. From the structure of parity check matrix, the efficient encoding method can be performed as following:

$$\begin{bmatrix} \mathbf{Ac}+\mathbf{Bw}_1 \\ \mathbf{C}\times[\mathbf{c}^{\mathrm{T}} \quad \mathbf{w}_1^{\mathrm{T}}]^{\mathrm{T}}+\mathbf{w}_2 \end{bmatrix} = \mathbf{0}$$

$$\mathbf{w}_1 = -\mathbf{B}^{-1}\mathbf{Ac}$$

$$\mathbf{w}_2 = -\mathbf{C}\times[\mathbf{c}^{\mathrm{T}} \quad \mathbf{w}_1^{\mathrm{T}}]^{\mathrm{T}}$$

All operations are performed in GF(2).

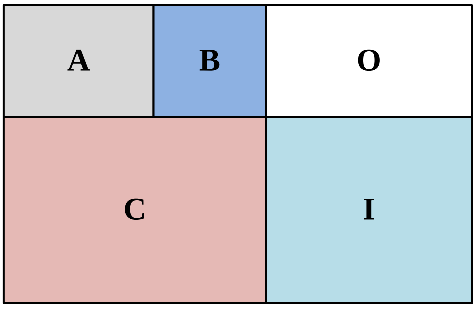

Figure 7. Structure of parity check matrix for LDPC.

- In rate matching, the rate matching is applied per coded block and consists of 2 steps - bit selection and bit interleaving. The input bit sequence is denoted by $d_0, d_1, d_2, \ldots, d_{N-1}$, and the output bit sequence is denoted by $f_0, f_1, f_2, \ldots, f_{E-1}$.

  In bit selection, the bit sequence $d_0, d_1, d_2, \ldots, d_{N-1}$ is inserted into a circular buffer of length N for coded block r. For coded block r, if $I_{LBRM} = 0$, then $N_{cb} = N$, and otherwise, $N_{cb} = \min(N, N_{ref})$, where $N_{ref} = \left\lfloor \frac{TBS_{\mathrm{LBRM}}}{C \cdot R_{\mathrm{LBRM}}} \right\rfloor$, $R_{\mathrm{LBRM}} = 2/3$, and TBS_{LBRM} is determined according to [4] assuming the following, referring [2]:

  - maximum number of layers for one transport block supported by the UE for the serving cell, which for UL-SCH is according to higher layer parameter *ULmaxRank* if the parameter is configured
  - maximum modulation order configured for the serving cell, if configured by higher layers; otherwise a maximum modulation order is assumed for DL-SCH
  - maximum coding rate of 948/1024
  - $n_{PRB} = n_{PRB,LBRM}$ is given by Table 2, where the value of $n_{PRB,LBRM}$ for DL-SCH is determined according to the initial bandwidth part if there is no other bandwidth part configured to the UE;
  - $N_{RE} = 156 n_{PRB}$

Table 2. Value of $n_{PRB,LBRM}$.

Maximum number of PRBs across all configured BWPs of a carrier	$n_{PRB,LBRM}$
Less than 33	32

33 to 66	66
67 to 107	107
108 to 135	135
136 to 162	162
163 to 217	217
Larger than 217	273

The rate matching output sequence length for the coded block r is denoted by E_r and determined as below

Set $j = 0$
for $r = 0$ to $C - 1$
if the coded block r is not for transmission as indicated by CBGTI according to [4]
$E_r = 0$
else
if $j \le C' - \mathrm{mod}(G/(N_L \cdot Q_m), C') - 1$
$$E_r = N_L \cdot Q_m \cdot \left\lfloor \frac{G}{N_L \cdot Q_m \cdot C'} \right\rfloor$$
else
$$E_r = N_L \cdot Q_m \cdot \left\lceil \frac{G}{N_L \cdot Q_m \cdot C'} \right\rceil$$
end if
$j = j + 1$
end if
end for
where, referring [2],

- N_L is the number of transmission layers that the transport block is mapped onto
- Q_m is the modulation order
- G is the total number of coded bits available for transmission of the transport block
- $C' = C$ if CBGTI is not present in the DCI scheduling the transport block and C' is the number of scheduled code blocks of the transport block if CBGTI is present in the DCI scheduling the transport block

The rate matching output bit sequence e_k is generated as follows, where k_0 is given by Table 3 according to the value of the redundancy version number for this transmission rv_{id} = 0, 1, 2 or 3

$k = 0$, $j = 0$
while $k < E$
if $d_{(k_0 + j) \bmod N_{cb}} \neq \text{<NULL>}$
$e_k = d_{(k_0 + j) \bmod N_{cb}}$
$k = k + 1;$
end if
$j = j + 1$
end while

Table 3. Starting position of different redundancy versions, k_0.

rv_{id}	k_0

	Base graph 1	Base graph 2
0	0	0
1	$\left\lfloor \frac{17N_{cb}}{66Z_c} \right\rfloor Z_c$	$\left\lfloor \frac{13N_{cb}}{50Z_c} \right\rfloor Z_c$
2	$\left\lfloor \frac{33N_{cb}}{66Z_c} \right\rfloor Z_c$	$\left\lfloor \frac{25N_{cb}}{50Z_c} \right\rfloor Z_c$
3	$\left\lfloor \frac{56N_{cb}}{66Z_c} \right\rfloor Z_c$	$\left\lfloor \frac{43N_{cb}}{50Z_c} \right\rfloor Z_c$

In bit interleaving, the bit sequence $e_0, e_1, e_2, ..., e_{E-1}$ is interleaved to bit sequence $f_0, f_1, f_2, ..., f_{E-1}$, as follows, where Q_m is the number of coded bits per QAM symbol given by Table 4.

```
for  j = 0  to  E / Q_m − 1
        for  i = 0  to  Q_m − 1
                f_{i+j·Q_m} = e_{i·E/Q_m+j}
        end for
end for
```

- In code block concatenation, the input sequences is f_{rk} for $r = 0,...,C-1$ and $k = 0,..., E_r - 1$) and the output bit sequence is g_k for $k = 0,...,G-1$. The rate matching outputs are concatenated as below:

```
k = 0
for  r = 0  to  C − 1
        for   j = 0  to  E_r − 1
                g_k = f_rj
                k = k + 1
        end for
end for
```

method_channel_coding(obj,para)
Input : **obj.bit_stream,** obj.M_order,obj.Coding_rate, parameters for transmission (4.1.1-a.10) Output : **obj. coded_bit_stream,** obj.ch_coding

- **Scrambling module**

- For each codeword q, a pseudo-random sequence defined in TS.38.211 is generated using c_{init} and codeword length satisfying the following equation.

$$c_{\text{init}} = n_{\text{RNTI}} \cdot 2^{15} + n_{\text{ID}}$$

- Creates scrambled_bit_stream by performing mod2 operation on codeword input value Coded_bit_stream and pseudo-random sequence

method_scrambling(obj, para, ind_slot)
Input : **obj.coded_bit_stream**, obj.n_RNTI, parameters for transmission of parameter class (4.1.1-b.10) Output : **obj.scrambled_bit_stream**

- **Modulation mapping module**

- A module that maps scrambled bit sequences generated according to TS 38.211 and TS 38.213 to modulation symbols according to modulation order determined by CQI. Modulation mapping is realized in the case that the modulation order is set to 2, 4, 6, 8 in CQI table of 5.2.2.1-2 of TS 38.214. In the case of mapping to modulation symbols according to modulation order, mapping is performed according to 7.1.2 ~ 5-1 of TS 38.211.

method_modulation_mapping(obj)
Input : **obj.scrambled_bit_stream,** obj.M_order, obj.bit_stream Output : **obj.modulated_signal_stream**

- **Transform precoding module**

- If transform precoding is not enabled, output signal of transform precoding module is same with modulated signal.
- If transform precoding is enabled, transform precoding is performed according to clause 6.3.1.4 in TS 38.211.

method_Transform_precoding(obj,para)
Input : **obj.modulated_signal_stream** Output : **obj.resource_grid,** obj.zero_length_for_tp

- **Data mapping module**

- Assign Modulated_signal_stream to the resource grid. Data is allocated in the order of the slots.

method_Data_mapping(obj,para)
Input : **obj.modulated_signal_stream, obj.Data_position,** Output : **obj.resource_grid**

- **Power allocation module**

- Assigns powers to the RSs and data signals as user's demand. Therefore, our simulator receives power of RSs and data signals from the user.

method_power_allocation(obj, para)
Input : parameters for power allocation (4.1.1-b.12), obj.Data_position, **obj.resource_grid** Output : **obj.resource_grid,** obj.resource_grid_DMRS, obj.resource_grid_SRS

-

- **OFDM generation module**

- Converts mapped signals to the OFDM signals and add cyclic prefix. Zero padding and DC carrier is also performed during converting. From 5.3 in [1], subcarrier spacing is μ and lth OFDM signal from antenna port p, $s_l^{(p,\mu)}(t)$ is defined as follows

$$s_l^{(p,\mu)}(t) = \sum_{k=0}^{N_{\text{grid}}^{\text{size},\mu} N_{\text{sc}}^{\text{RB}} - 1} a_{k,l}^{(p,\mu)} \cdot e^{j2\pi\left(k + k_0^{\mu} - N_{\text{grid}}^{\text{size},\mu} N_{\text{sc}}^{\text{RB}}/2\right)\Delta f\left(t - N_{\text{CP},l}^{\mu} T_c\right)}$$

where $0 \le t < \left(N_u^{\mu} + N_{\text{CP},l}^{\mu}\right) T_c$.

- $a_{k,l}^{(p,\mu)}$ indicates value in the position of (k,l) in the resource grid.
- The position of the lth OFDM signal is determined as follows:

$$t_{\text{start},l}^{\mu} = \begin{cases} 0 & l = 0 \\ t_{\text{start},l-1}^{\mu} + \left(N_u^{\mu} + N_{\text{CP},l-1}^{\mu}\right) \cdot T_c & \text{otherwise} \end{cases}$$

where,

$$N_u^{\mu} = 2048\kappa \cdot 2^{-\mu}$$

$$N_{\text{CP},l}^{\mu} = \begin{cases} 512\kappa \cdot 2^{-\mu} & \text{extended cyclic prefix} \\ 144\kappa \cdot 2^{-\mu} + 16\kappa & \text{normal cyclic prefix, } l = 0 \text{ or } l = 7 \cdot 2^{\mu} \\ 144\kappa \cdot 2^{-\mu} & \text{normal cyclic prefix, } l \neq 0 \text{ and } l \neq 7 \cdot 2^{\mu} \end{cases}.$$

method_OFDM_generation(obj, para)
Input : parameters for frame structure (4.1.1-b.1), parameters for OFDM (4.1.1-b.13), **obj.resource_grid** Output : **obj.OFDM_signal_stream**

4.1.3-a BS_MIMO function block submodules for Downlink

- **Feedback reception module**

- A module that receives feedback from the UE functional block and processes the channel feedback related information for CQI related information, PMI and RI related information for SUMIMO. Also process the channel feedback related information for non-codebook case, and generates a precoder. It is assumed that feedback data has already been received from receiver by the uplink channel and we use that data in each module.

- **Beam management module**

- Wide beam and narrow beam can be selected. The SS / PBCH is used to search for wide beam and then consider narrow beam selection via CSI-RS [20]. We considered that wide beam combines DFT beam and narrow beam through oversampling using the 'Woodward-Lawson method of antenna pattern synthesis' [21]. The DFT beam and its oversampling beam are considered as narrow beam [4]. In addition, the role of the beam management block of UE is also part of this module.

method_beam_management(obj, para, ch_output, ind_slot)
Input : para.cal_scenario, para.scenario_beam, para.N2, para.num_Rx_antenna Output : obj.Narrowbeam, obj.Precoder, obj.RI, obj.Precoder_CSIRS, obj.CSIRS_position_total
Find wide beam Find narrow beam Virtual reception of CSI-RS signal Beam channel estimation Beam selection & feedback

- **RS allocation module**

- DMRS and CSI-RS are generated and mapped with same rule as SU-SISO. The only difference comes from usage of OCC (orthogonal cover code) in the case of allocating same RS to same RE when the number of antenna ports gets larger.

method_RS_allocation(obj, para, ch_output)
Input : para.num_port_CSIRS, para.row_CSIRS, para.CSIRS_density ind_slot, parameters for reference signal (4.1.1-a.2) Output : obj.CSIRS_position, obj.CSIRS_position_total , obj.resource_grid_CSIRS obj.DMRS_position_total, obj.DMRS_position, obj.resource_grid_DMRS, obj.resource_grid_CSIRS, obj.CSIRS_position

- **Resource allocation module**

- It is an extended module used in BS_SUSISO. The SS/PBCH allocation block is also in this module as mentioned in SISO module of 4.1.2.

method_resource_allocation(obj,para,ind_slot)
Input : ind_slot, parameters for frame structure, resource allocation, transmission (4.1.1-a.1, 3, 10) Output : obj.M_order, obj.Coding_rate, obj.Control_position_SISO, obj.RS_position_SISO, obj.Data_position

- **Data allocation module**

- It is acting as data allocation in the resource allocation block of the block diagram. Sets the data position after storing the resource grid position of PDCCH, SS/PBCH and RS as in BS_SUSISO. Since the type of reference signal varies according to the transmission mode, how the data position is determined also changes.

method_data_allocation (obj, para)
Input : para.cal_scenario, obj.CQI, obj.RI Output : obj.M_order, obj.Coding_rate, obj.Data_position, para.num_coded_bits, para.num_data_bits

- **Sync generation module**

- Performs the same as Synchronization module of BS_SISO function block. It can be used for SS/PBCH. However, if there are more than two antennas of BS, there are two methods: allocating a synchronization signal to only the first antenna port, leaving the remaining antenna port empty; and allocating a synchronization signal by redistributing power to all antenna ports. In this simulator, a synchronization signal is assigned to only the first antenna port.

method_sync_generation(obj, para)
Input : para.cell_ID Output : obj.PSS_signal, obj.SSS_signal

- **HARQ process module**

- The basic operation is the same as the HARQ process module in SISO. In MIMO environment in which two transport blocks are transmitted in only one subframe, two HARQ processes are performed for one HARQ process.

method_HARQ_process(obj, para, feedback)
Input : obj.num_codewords, obj.HARQ_process_index, obj.HARQ_buffer Output : obj.HARQ_newdata_indicator, obj.HARQ_process_index, obj.HARQ_buffer

- **Bit generation module**

- Extended according to the number of transport blocks using the module used in BS_SUSISO.

method_bit_generation(obj, para)
Input : para.num_data_bits, obj.num_codewords Output : obj.bit_stream

- **Channel coding module**

- Extended according to the number of transport blocks using the module used in BS_SUSISO.

method_channel_coding(obj, para)

Input : para.num_data_bits, para.num_coded_bits, obj.bit_stream Output : obj.ch_coding, obj.coded_bit_stream

- **Scrambling module**

- Extended according to the number of transport blocks using the module used in BS_SUSISO.

method_scrambling(obj, para)
Input : obj.num_codewords, obj.num_codewords, obj.coded_bit_stream Output : obj.scrambled_bit_stream

- **Modulation mapping module**

- Extended according to the number of transport blocks using the module used in BS_SUSISO.

method_modulation_mapping(obj, para)
Input : obj.M_order, obj.num_codewords, obj.scrambled_bit_stream Output : obj.modulated_signal_stream

- **Layer mapping module**

- PDDCH and PBCH are stored separately in two parts of Real and Imaginary.
- Distributes Modulated_signal_stream to each layer as defined in the standard as follows.

method_layer_mapping(obj, para)
Input : obj.num_codewords, obj.modulated_signal_stream, obj.num_layers Output : obj.layered_signal

Number of layers	**Number of codewords**	**Codeword-to-layer mapping** $i = 0,1,\ldots,M_{symb}^{layer} - 1$	
1	1	$x^{(0)}(i) = d^{(0)}(i)$	$M_{symb}^{layer} = M_{symb}^{(0)}$
2	1	$x^{(0)}(i) = d^{(0)}(2i)$ $x^{(1)}(i) = d^{(0)}(2i+1)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 2$
3	1	$x^{(0)}(i) = d^{(0)}(3i)$ $x^{(1)}(i) = d^{(0)}(3i+1)$ $x^{(2)}(i) = d^{(0)}(3i+2)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 3$
4	1	$x^{(0)}(i) = d^{(0)}(4i)$ $x^{(1)}(i) = d^{(0)}(4i+1)$ $x^{(2)}(i) = d^{(0)}(4i+2)$ $x^{(3)}(i) = d^{(0)}(4i+3)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 4$
5	2	$x^{(0)}(i) = d^{(0)}(2i)$ $x^{(1)}(i) = d^{(0)}(2i+1)$ $x^{(2)}(i) = d^{(1)}(3i)$ $x^{(3)}(i) = d^{(1)}(3i+1)$ $x^{(4)}(i) = d^{(1)}(3i+2)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 2 = M_{symb}^{(1)} / 3$
6	2	$x^{(0)}(i) = d^{(0)}(3i)$ $x^{(1)}(i) = d^{(0)}(3i+1)$ $x^{(2)}(i) = d^{(0)}(3i+2)$ $x^{(3)}(i) = d^{(1)}(3i)$ $x^{(4)}(i) = d^{(1)}(3i+1)$ $x^{(5)}(i) = d^{(1)}(3i+2)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 3 = M_{symb}^{(1)} / 3$
7	2	$x^{(0)}(i) = d^{(0)}(3i)$ $x^{(1)}(i) = d^{(0)}(3i+1)$ $x^{(2)}(i) = d^{(0)}(3i+2)$ $x^{(3)}(i) = d^{(1)}(4i)$ $x^{(4)}(i) = d^{(1)}(4i+1)$ $x^{(5)}(i) = d^{(1)}(4i+2)$ $x^{(6)}(i) = d^{(1)}(4i+3)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 3 = M_{symb}^{(1)} / 4$
8	2	$x^{(0)}(i) = d^{(0)}(4i)$ $x^{(1)}(i) = d^{(0)}(4i+1)$ $x^{(2)}(i) = d^{(0)}(4i+2)$ $x^{(3)}(i) = d^{(0)}(4i+3)$ $x^{(4)}(i) = d^{(1)}(4i)$ $x^{(5)}(i) = d^{(1)}(4i+1)$ $x^{(6)}(i) = d^{(1)}(4i+2)$ $x^{(7)}(i) = d^{(1)}(4i+3)$	$M_{symb}^{layer} = M_{symb}^{(0)} / 4 = M_{symb}^{(1)} / 4$

Table 8. Table 7.3.1.3-1 in [1]

- **Precoding module**

- The Hybrid beamforming block and the Data mapping block are executed in this module. The signal stream data from the layer mapping is passed to the given precoder of each RB and is input to the data of each antenna port. The precoder can be generated using Type I CSI-RS codebook of 5.2.2.2.1 [4]. The precoder is 'obj.Precoder', and algorithm is written in 'func_beam_selection.m'. Then the power allocation method is also progressed and Data is allocated in the order of the slots.

method_precoding(obj, para)
Input : para.use_Power_allocation, para.power_DMRS, para.power_CSIRS, para.power_data, para.num_codewords, para.num_RB, para.num_sc, para.num_symb, para.num_Tx_antenna Output : obj.precoded_signal_stream, obj.resource_grid, obj.data_position_total

- **OFDM generation module**

- The same as SISO module.

method_OFDM_generation(obj, para)
Input : parameters for frame structure (4.1.1-a.1), parameters for OFDM (4.1.1-a.13), **obj.resource_grid** Output : **obj.OFDM_signal_stream**

-

4.1.3-b UE_MIMO function block submodules for Uplink (common parts with downlink case are omitted)

- **Feedback reception module**

- A module that receives feedback from the UE functional block and processes the channel feedback related information for CQI related information, PMI and RI related information for SUMIMO. Also process the channel feedback related information for non-codebook case, and generates a precoder. It is assumed that feedback data has already been received from receiver and we use that data in each module.

method_feedback_reception (obj, para, feedback)
Input : feedback.CQI, para.M_order, para.Coding_rate Output : obj.CQI, obj.M_order, obj.Coding_rate

- **Precoding module**

- The beamforming block are executed in this module. It is assumed that the precoder is already determined prior transmission procedure. Thus the determined precoder is used for uplink.

method_precoding(obj, para)
Input : para.num_RB, para.num_Tx_antenna Output : obj.Precoder

- **Transform precoding module**

- If transform precoding is not enabled, output signal of transform precoding module is same with modulated signal.
- If transform precoding is enabled, transform precoding is performed according to clause 6.3.1.4 in TS 38.211.

method_Transform_precoding(obj,para)
Input : **obj.modulated_signal_stream** Output : **obj.resource_grid,** obj.zero_length_for_tp

4.1.4 Channel_5G function block submodules

- **Chan_Matrix module**

- Chan_Matrix generates channel matrix based on the interpolation method. The result is different according to what kind of channel it is: AWGN, PedA, PedB, VehA, VehB, Rayleigh, EPA(Extended Pedestrian A model), EVA(Extended Vehicle A model), and ETB(Extended Typical Urban model).
- It defines and calculates the needed values using PDP per channel predefined in Parameter function block.
- It generates different result according to whether 'time_correlation' is correlated or independent. Besides, we applied the following formula in [8] 'III. New Simulation Models.'

$$X(t) = X_c(t) + jX_s(t), \quad X_c(t) = \frac{2}{\sqrt{M}} \sum_{n=1}^{M} cos(\Psi n)\, cos(w_d t cos\alpha_n + \varphi),$$

$$X_s(t) = \frac{2}{\sqrt{M}} \sum_{n=1}^{M} sin(\Psi n)\, cos(w_d t cos\alpha_n + \varphi)$$

- θ, φ, Ψn are uniformly distributed on $[-\pi, \pi)$, w_d is the maximum Doppler shift frequency, and α_n is given as:

$$\alpha_n = \frac{2\pi n - \pi + \theta}{4M}, \qquad n = 1, 2, \ldots, M$$

- Chan_Matrix generates different result according to whether the channel is block fading or fast fading.

- There are two interpolation methods: ‘nearest_neighbor’ and ‘sinc_interpolation.’ First, ‘nearest_neighbor’ interpolates the nearest value, while ‘sinc_interpolation’ applies sinc function for reconstruction. Especially when the channel is block fading, we apply correlation matrices to channel matrix for ‘nearest_neighbor’ method.
- Finally, we normalize the resulted matrix.
- Chan_Matrix generates channel matrix based on the interpolation method.
- 3D spatial channel model – CDL, TDL is newly added in MIMO channel. 3D spatial channel model generates channel matrix according to scenarios(CDL-A, CDL-B … CDL-E; TDL-A, … TDL-E). PDP for each scenario is defined in TR 38.901[19], saved in module_Parameter. PDP of TDL-A is exemplified as the following table:

• **Chan_H_fft module**

- Chan_H_fft generates the value of the channel w.r.t. frequency applying Fast Fourier transform.
- The result is different according to whether the channel is block fading or fast fading.

Clusters						
Cluster #	Normalized delay	Power dB	AoD °	AoA °	ZoD °	ZoA °
1	0.0000	-13.4	-178.1	51.3	50.2	125.4
2	0.3819	0	-4.2	-152.7	93.2	91.3
3	0.4025	-2.2	-4.2	-152.7	93.2	91.3
4	0.5868	-4	-4.2	-152.7	93.2	91.3
5	0.4610	-6	90.2	76.6	122	94
6	0.5375	-8.2	90.2	76.6	122	94
7	0.6708	-9.9	90.2	76.6	122	94
8	0.5750	-10.5	121.5	-1.8	150.2	47.1
9	0.7618	-7.5	-81.7	-41.9	55.2	56
10	1.5375	-15.9	158.4	94.2	26.4	30.1
11	1.8978	-6.6	-83	51.9	126.4	58.8
12	2.2242	-16.7	134.8	-115.9	171.6	26
13	2.1718	-12.4	-153	26.6	151.4	49.2
14	2.4942	-15.2	-172	76.6	157.2	143.1
15	2.5119	-10.8	-129.9	-7	47.2	117.4
16	3.0582	-11.3	-136	-23	40.4	122.7
17	4.0810	-12.7	165.4	-47.2	43.3	123.2
18	4.4579	-16.2	148.4	110.4	161.8	32.6
19	4.5695	-18.3	132.7	144.5	10.8	27.2
20	4.7966	-18.9	-118.6	155.3	16.7	15.2
21	5.0066	-16.6	-154.1	102	171.7	146
22	5.3043	-19.9	126.5	-151.8	22.7	150.7
23	9.6586	-29.7	-56.2	55.2	144.9	156.1
Per-Cluster Parameters						
Parameter	c_{ASD}	c_{ASA}	c_{ZSD}	c_{ZSA}	XPR	
Unit	°	°	°	°	dB	
Value	5	11	3	3	10	

Table 13. CDL-A in [19]

4.1.5-a UE_SISO function block submodules for Downlink

- **Channel filtering module**

- Filters the resampled channel impulse function with transmitted signal in time domain.
- For Block fading case, channel impulse function is identical during one subframe so that filtering conducts convolution with transmitted signal as follows:

$$\mathrm{y[n]} = x[n] * h[n]$$

- Here, $y[n]$ is the filtered output sequence with CP, $x[n]$ is the OFDM signal sequence, and $h[n]$ is the channel impulse function.

method_channel_filtering(obj, para, genie, ch_Output)
Input : **genie.OFDM_signal_stream,** parameters for scenario (4.1.1-a.11), parameters for frame structure (4.1.1-a.1) Output : **obj.received_signal_stream,** obj.sigma_n_freq, obj.sigma_n_time

- **Synchronization module**

- This function module carries out carrier offset estimation process for carrier frequency offset compensation. It is based on [5] and consists of fractional frequency offset (FFO) estimation, integer frequency offset (IFO) estimation and residual frequency offset (RFO) estimation.
- FFO estimation is the maximum likelihood estimator using cyclic prefix signal [5].
- IFO estimation estimates the frequency offset of integer multiples of subcarrier spacing using PSS and SSS signal [5].
- In the case of RFO estimation, CRS signal is used to estimate carrier frequency offset due to FFO estimation error [5].
- After removing the cyclic prefix from the synchronized received signal stream, it restores the existing $a_{k,l}^{(p)}$ through IFFT. At this time, after passing FFT, zero padding and DC carrier should be removed.

method_synchronization(obj, para, genie, ind_slot)
Input : **obj.received_signal_stream,** parameters for scenario (4.1.1-a.11), parameters for frame structure (4.1.1-a.1), parameters for OFDM (4.1.1-a.13) Output : **obj.received_signal_stream_sync**

- **OFDM demodulation module**

- Remove cyclic prefix from synchronized received single stream and then IFFT to reconstruct $a_{k,l}^{(p)}$. Here zero padding and DC carrier should be removed.

method_OFDM_demodulation(obj,para)
Input : **obj.received_signal_stream_sync,** parameters for OFDM (4.1.1-a.13), parameters for frame structure (4.1.1-a.1)

Output : **obj.received_resource_grid**

- **Channel estimation module**

- For SISO, channel estimation is conducted through CSI-RS and DMRS. Both CSI-RS and DMRS can perform same channel estimation since estimated channel with DMRS is effective channel where precoding matrix is identity matrix which leads to the effective channel to be same with channel. There is two methods for channel estimation. First, LS (least square) method [13]. LS method is performed with following equation.

$$argmin_h |y - hx|^2$$

Here, y indicates received signal, x indicates RS, and h indicates channel. For this case, simple division as follows estimates channel.

$$\hat{h} = y/x$$

For block fading case, we can perform better estimation since multiple RSs are transmitted for the same channel as follows:

$$\hat{h} = (x^H y)/(x^H x)$$

This estimated channel value is the channel for the REs where RSs have been allocated. In order to get the channel for the REs that RSs have not been allocated, we need interpolation method. We used linear, spline, and time domain interpolation method. Time domain interpolation method converts estimated channel to time domain through IDFT, then after zero-padding, DFT again to convert back to frequency domain.

- Another channel estimation method is from MMSE estimation. For MMSE estimation, we need channel covariance matrix to estimate channel as follows:

$$\hat{h} = R_h \left(\bar{R}_h + \frac{\sigma^2}{P} I \right)^{-1} \hat{h}_{LS}$$

where P indicates power of RS, $\hat{h}_{LS}$ indicates estimated channel through LS, $\bar{R}_h$ indicates covariance matrix for the estimated channel through LS in the position of RSs, and R_h indicates channel covariance matrix for the position of RSs. With this method, we do not need additional interpolation method.

method_channel_estimation(obj, para, genie, ch_Output,ind_slot)
Input : **obj.received_resource_grid,** obj.channel_buffer Output : obj.channel_buffer, **obj.H_estimated_power,** obj.HV_estimated_error, obj.HV_estimated_error_ave, obj.H_estimated_error, obj.H_estimated_error_ave, obj.HV_estimated, obj.H_estimated, obj.HV_perfect

- **Data extraction module**

- Data in the received resource gird is extracted in the order in which they are inserted in the BS. (Data is extracted in order of the slots, and extracted in the order of the RBs in the same slot.)

5G Wireless Innovative Systems Engineering Research Center

method_Data_extraction(obj,para, genie)
Input : **obj.H_estimated_power, obj.received_resource_grid,** parameters for frame structure (4.1.1-a.1) Output : **obj.Data_stream, obj.H_estimated_data, obj.Data_stream_indexf**

-

- **Detection module**

- Using estimated channel and received signal, calculates soft output for channel decoding. Soft output is the log likelihood ratio (LLR) in bit level which is calculated as follows:

$$L(s_i|y) = log\left(\frac{\sum_{s:s_i(s)=+1} p(y|s)}{\sum_{s:s_i(s)=-1} p(y|s)}\right) = log\left(\frac{\sum_{s:s_i(s)=+1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}{\sum_{s:s_i(s)=+1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}\right)$$

Here, s_i is the i-th bit of the received symbol s. The above equation is log-MAP soft output which can be replaced with following equation of Max-log, for low complexity.

$$L(s_i|y) \approx log\left(\frac{\max_{s:s_i(s)=+1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}{\max_{s:s_i(s)=-1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}\right)$$

method_detection(obj,para,genie)
Input : **obj.Data_stream, obj.H_estimated_data, obj.Data_stream_indexf,** obj.sigma_n_freq Output : **obj.LLR**

- **Descrambling module**

- For each codeword q, a pseudo-random sequence defined in TS.38.211 [1] is generated using c_{init} and codeword length satisfying the following equation.

$$c_{init} = n_{RNTI} \cdot 2^{14} + q \cdot 2^{13} + \frac{n_s}{2} \cdot 2^9 + N_{ID}^{cell}$$

- '0' is changed to '-1' in the generated pseudo-random sequence, and then multiplied by LLR to create a descrambled one.

method_descrambling(obj,para,ind_slot)
Input : **obj.LLR,** parameters for transmission (4.1.1-a.10) Output : **obj.descrambled_LLR**

-

- **HARQ process**

- Module that performs HARQ function in BS. When the received signal is NACK, the received signal and related information are stored in the corresponding HARQ_buffer. If the received signal is ACK, HARQ_buffer is initialized. If the new data indicator from the BS is 0 (in the case of retransmission), decoding is performed using the information in the previous buffer and the current retransmitted information.
- In this simulator, Chase combining and incremental redundancy based HARQ processes are basically implemented. Both methods are performed in rate matching block, refering the redundancy versions.

method_HARQ_process(obj,genie)
Input : **obj.descrambled_LLR,** genie.HARQ_process_index Output : **obj.descrambled_LLR_HARQ,** obj.HARQ_buffer

-

- **Channel decoding module**

- Figure 8 is the structure and procedure of channel encoding module based on corresponding to channel encoding module. The decoding module consists of detail modules – code block deconcatenation, rate dematching, LDPC decoding, code block desegmentation, and CRC check.

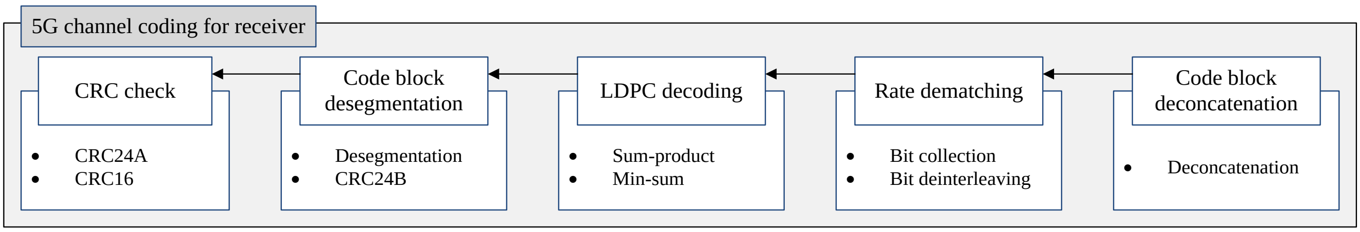

Figure 9. Structure and procedure of channel decoding module.

- In code block deconcatenation, the LLR sequence is received from the previous block. Assume that the input sequence and the output sequence of code block deconcatenation are denoted as g_k^{rx} and f_{rj}^{rx}. Then, the LLR sequence is corresponding to g_k^{rx}, and thus the f_{rj}^{rx} is deconcatenated as below:

$k = 0$
for $k = 0$ $r = 0$ to $C-1$
  for $j = 0$ to $E_r - 1$
    $f_{r,j}^{rx} = g_k^{rx}$
    $k = k + 1$
  end for
end for

- In rate dematching, the LLR sequence f_{rj}^{rx} is firstly deinterleaved as below:

for $r = 0$ to $C-1$
  for $j = 0$ to $E_r / Q_m - 1$

```
                    for  i = 0  to  Q_m − 1
                              e^rx_{r, j+i×E_r/Q_m} = f^rx_{r,i+j×Q_m}
                    end for
          end for
end for
```

The deinterleaved sequence $e^{\mathrm{rx}}_{r,j+i\times E_r/Q_m}$ is recovered as below, where $2Z_c$ punctured bits in LDPC encoding and *NULL* bits removed in bit selection step of rate matching are considered.

```
for  r = 0  to  C − 1
          j = 0,  k = 0
          while (1)
                    if  (mod(k_{0,rv_id} + j, N_cb) + 2Z_c < null_0) or (mod(k_{0,rv_id} + j, N_cb) + 2Z_c > null_{K−K'−1})
                              if ( k == E_r )
                                        break;
                              else if ( E_r < N_cb )
                                        d^rx_{r,mod(k_{0,rv_id} + j, N_cb) + 2×Z_c} = e^rx_{r,k}
                                        k = k + 1
                              else if ( E_r > N_cb )
                                        if ( j < N_cb )
                                                  d^rx_{r,mod(k_{0,rv_id} + j, N_cb) + 2×Z_c} = e^rx_{r,k}
                                        else
                                                  d^rx_{r,mod(k_{0,rv_id} + j, N_cb) + 2×Z_c} = d^rx_{r,mod(k_{0,rv_id} + j, N_cb) + 2×Z_c} + e^rx_{r,k}
                                        end if
                                        k = k + 1
                              end if
                    end if
                    j = j + 1
          end while
end for
```

- $N_{NULL} = K - K'$ is the number of *NULL* bits generated in block segmentation
- $null_k = K' + k,\ k = 0,\ldots,K-K'$
- $k_{0,rv_{id}}$ is rv_{id} -th starting point in Table 3 (Starting position of different redundancy versions, k_0)

- In LDPC decoding, a bipartite graph is considered from corresponding to parity check matrix of LDPC. The bipartite graph is a graph whose nodes are separated into two parts, variable nodes and check nodes. There are N_{H} variable nodes and $M_{\mathrm{H}} = N_{\mathrm{H}} - K$ check nodes. N_{H} and M_{H} are the number of columns and rows of $\mathbf{H}$, respectively. Also, the M_{H} rows and N_{H} columns specify the M_{H} check nodes and the N_{H} variable nodes. The bipartite graph of an LDPC code is drown if the check node m is connected to variable node n whenever h_{mn} as element of $\mathbf{H}$ is 1, otherwise 0.
For the convenience, the notation is presented as below:

- $N(m)$: Variable nodes connected to check node m

- $N(m)\backslash n$: Variable nodes connected to check node m except variable node n
- $M(n)$: Check nodes connected to variable node n
- $M(n)\backslash m$: Check nodes connected to variable node n except check node m
- $bit \in \{0,1\}$
- $f_n^{bit} = \Pr(y_n | d_n = bit)$
- $L(f_n) = \log \frac{f_n^0}{f_n^1} = \frac{2y_n}{\sigma^2}$: LLR of $\frac{f_n^0}{f_n^1}$
- S_m : Event that parity check equation of check node m is satisfied
- q_{mn}^{bit} : Message from bit node n to check node m, $q_{mn}^{bit} = \Pr(d_n = bit | y_n, S_{m' \in M(n)\backslash m})$
- r_{mn}^{bit} : Message from check node m to variable node n, $r_{mn}^{bit} = \Pr(S_m | d_n = bit, \mathbf{y})$
- $L(q_{mn}) = \log \frac{q_{mn}^0}{q_{mn}^1}$: LLR of $\frac{q_{mn}^0}{q_{mn}^1}$
- $L(r_{mn}) = \log \frac{r_{mn}^0}{r_{mn}^1}$: LLR of $\frac{r_{mn}^0}{r_{mn}^1}$

Basically, the 2 LDPC decoding methods are considered. The one is the sum-product method and the other is the min-sum method.

Firstly, the sum-product method is described as in the following.

The input LLR of LDPC decoding block is $d_{r,n}^{\mathrm{rx}}$ and $L(f_n) = d_{r,n}^{\mathrm{rx}}$. Before initialization step, the LLRs $L(f_n)$ of $2Z_c$ punctured bits in LDPC encoding are set as 0 due to the uncertainty of them. Also, the LLRs $L(f_n)$ of *NULL* bits are set as $-\infty$ because -1 is inserted to *NULL* bits for simulations.

In initialization step, the LLR $L(q_{mn})$ is initialized by $L(q_{mn}) = L(f_n)$.

In check node update step, the LLR $L(r_{mn})$ is updated as below:

$$L(r_{mn}) = -\left[\prod_{n' \in N(m)\backslash n} \mathrm{sgn}\left(L(q_{mn'})\right)\right] \Phi\left(\sum_{n' \in N(m)\backslash n} \Phi(|L(q_{mn'})|)\right)$$

$$\mathrm{sgn}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases}$$

$$\Phi(x) = \log \tanh(\frac{x}{2})$$

In variable node update step, the LLR $L(r_{mn})$ is updated as below:

$$L(q_{mn}) = L(f_n) + \sum_{m' \in M(n)\backslash m} L(r_{m'n})$$

LLR of a posteriori probability of coded bit n, $L(q_n)$ is calculated as below:

$$L(q_n) = L(f_n) + \sum_{m \in M(n)} L(r_{mn})$$

The decoding procedure is repeated for per coded block $r = 0, ..., C-1$. For coded block r, the iterative decoding is performed. In exactly, the check node update step and the variable node update step are iteratively calculated until the maximum iteration. Therefore, $L(q_{mn})$ is calculated in variable node update step, and then fed back to check node update step in order to calculate $L(r_{mn})$.

Secondly, the min-sum method is described as in the following.

The all procedures are same except for check node update step. The function $\Phi(x)$ is positive and monotonically decreasing function for $x > 0$, and it is satisfied that $\Phi(\Phi(x)) = x$, since the inverse of $\Phi(x)$ is $\Phi(x)$ for $x > 0$. From the properties of $\Phi(x)$, $\Phi\left(\sum_{n'\in N(m)\setminus n} \Phi(|L(q_{mn'})|)\right)$ can be approximated by

$$\Phi\left(\sum_{n'\in N(m)\setminus n} \Phi(|L(q_{mn'})|)\right) \approx \Phi\left(\Phi(\min_{n'\in N(m)\setminus n} |L(q_{mn'})|)\right) = \min_{n'\in N(m)\setminus n} |L(q_{mn'})| .$$

Thus, the LLR $L(r_{mn})$ is updated as below:

$$L(r_{mn}) = -\left[\prod_{n'\in N(m)\setminus n} \operatorname{sgn}\left(L(q_{mn'})\right)\right] \min_{n'\in N(m)\setminus n} |L(q_{mn'})|$$

The output LLR sequence $L(q_n)$ denoted as $c_{r,k}^{\mathrm{rx}}$, which is transferred to code block desegmentation.

- In code block desegmentation, the input sequence and output sequence is denoted as b_s^{rx} and $c_{r,k}^{\mathrm{rx}}$, respectively. The desegmentation is performed as below:

```
s = 0
for r = 0 to E_r
        for k = 0 to K' - L - 1
                b_s^rx = c_{r,k}^rx
                s = s + 1
        end for
end for
```

- In CRC check, based on CRC generator polynomials, a decoding error is detected from b_s^{rx} as the input sequence of CRC check. CRC generator polynomials is dependent on A. Thus, if $A > 3824$, $g_{\mathrm{CRC24A}}(D)$ is used, and otherwise, $g_{\mathrm{CRC16}}(D)$ is used. If the decoding error is detected, then NACK is generated. Otherwise, ACK is generated.

method_decoding(obj,para,genie,ind_slot)
Input : **obj.descrambled_LLR** Output : **obj.decoded_bit, obj.feedback, obj.Result, obj.HARQ_buffer**

4.1.5-b BS_SISO function block submodules for Uplink

- **Channel filtering module**

- Filters the resampled channel impulse function with transmitted signal in time domain.
- For Block fading case, channel impulse function is identical during one subframe so that filtering conducts convolution with transmitted signal as follows:

$$\mathrm{y[n]} = x[n] * h[n]$$

- Here, $y[n]$ is the filtered output sequence with CP, $x[n]$ is the OFDM signal sequence, and $h[n]$ is the channel impulse function.

method_channel_filtering(obj, para, genie, ch_Output)

Input : **genie.OFDM_signal_stream,** parameters for scenario (4.1.1-b.11), parameters for frame structure (4.1.1-b.1) Output : **obj.received_signal_stream,** obj.sigma_n_freq, obj.sigma_n_time

- **Synchronization module**

- This function module carries out carrier offset estimation process for carrier frequency offset compensation. It is based on [5] and consists of fractional frequency offset (FFO) estimation, integer frequency offset (IFO) estimation and residual frequency offset (RFO) estimation.
- FFO estimation is the maximum likelihood estimator using cyclic prefix signal [5].
- IFO estimation estimates the frequency offset of integer multiples of subcarrier spacing using PSS and SSS signal [5].
- In the case of RFO estimation, CRS signal is used to estimate carrier frequency offset due to FFO estimation error [5].
- After removing the cyclic prefix from the synchronized received signal stream, it restores the existing $a_{k,l}^{(p)}$ through IFFT. At this time, after passing FFT, zero padding and DC carrier should be removed.

method_synchronization(obj, para, genie, ind_slot)
Input : **obj.received_signal_stream,** parameters for scenario (4.1.1-b.11), parameters for frame structure (4.1.1-b.1), parameters for OFDM (4.1.1-b.13) Output : **obj.received_signal_stream_sync**

- **OFDM demodulation module**

- Remove cyclic prefix from synchronized received single stream and then IFFT to reconstruct $a_{k,l}^{(p)}$. Here zero padding and DC carrier should be removed.

method_OFDM_demodulation(obj,para)
Input : **obj.received_signal_stream_sync,** parameters for OFDM (4.1.1-b.13), parameters for frame structure (4.1.1-b.1) Output : **obj.received_resource_grid**

-

- **Channel estimation module**

- For SISO, channel estimation is conducted through DMRS. There is two methods for channel estimation. First, LS (least square) method [13]. LS method is performed with following equation.

$$argmin_h |y - hx|^2$$

Here, y indicates received signal, x indicates RS, and h indicates channel. For this case, simple division as follows estimates channel.

$$\hat{h} = y/x$$

For block fading case, we can perform better estimation since multiple RSs are transmitted for the same channel as follows:

$$\hat{h} = (x^H y)/(x^H x)$$

This estimated channel value is the channel for the REs where RSs have been allocated. In order to get the channel for the REs that RSs have not been allocated, we need interpolation method. We used linear, spline, and time domain interpolation method. Time domain interpolation method converts estimated channel to time domain through IDFT, then after zero-padding, DFT again to convert back to frequency domain.

- Another channel estimation method is from MMSE estimation. For MMSE estimation, we need channel covariance matrix to estimate channel as follows:

$$\hat{h} = R_h \left(\bar{R}_h + \frac{\sigma^2}{P} I \right)^{-1} \hat{h}_{LS}$$

where P indicates power of RS, $\hat{h}_{LS}$ indicates estimated channel through LS, $\bar{R}_h$ indicates covariance matrix for the estimated channel through LS in the position of RSs, and R_h indicates channel covariance matrix for the position of RSs. With this method, we do not need additional interpolation method.

method_channel_estimation(obj, para, genie, ch_Output,ind_slot)
Input : **obj.received_resource_grid,** obj.channel_buffer Output : obj.channel_buffer, **obj.H_estimated_power,** obj.HV_estimated_error, obj.HV_estimated_error_ave, obj.HV_estimated, obj.HV_perfect

- **Data extraction module**

- Data in the received resource gird is extracted in the order in which they are inserted in the UE. (Data is extracted in order of the slots, and extracted in the order of the RBs in the same slot.)

method_Data_extraction(obj,para, genie)
Input : **obj.H_estimated_power, obj.received_resource_grid,** parameters for frame structure (4.1.1-a.1) Output : **obj.Data_stream, obj.H_estimated_data, obj.Data_stream_indexf**

-

- **Detection module**

- Using estimated channel and received signal, calculates soft output for channel decoding. Soft output is the log likelihood ratio (LLR) in bit level which is calculated as follows:

$$L(s_i|y) = log\left(\frac{\sum_{s:s_i(s)=+1} p(y|s)}{\sum_{s:s_i(s)=-1} p(y|s)}\right) = \ log\left(\frac{\sum_{s:s_i(s)=+1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}{\sum_{s:s_i(s)=+1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}\right)$$

Here, s_i is the i-th bit of the received symbol s. The above equation is log-MAP soft output which can be replaced with following equation of Max-log, for low complexity.

$$L(s_i|y) \approx log\left(\frac{\max\limits_{s:s_i(s)=+1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}{\max\limits_{s:s_i(s)=-1} exp\left(-\frac{1}{N_0}\left|\|y-hs\|\right)^2}\right)$$

- If transform precoding is enabled, inverse procedure of transform precoding should be considered.

method_detection(obj,para,genie)
Input : **obj.Data_stream, obj.H_estimated_data, obj.Data_stream_indexf,** obj.sigma_n_freq Output : **obj.LLR**

• **Descrambling module**

- For each codeword q, a pseudo-random sequence defined in TS.38.211 [1] is generated using c_{init} and codeword length satisfying the following equation.

$$c_{init} = n_{RNTI} \cdot 2^{15} + q \cdot 2^{13} + n_s \cdot 2^9 + N_{ID}^{cell}$$

- '0' is changed to '-1' in the generated pseudo-random sequence, and then multiplied by LLR to create a descrambled one.

method_descrambling(obj,para,ind_slot)
Input : **obj.LLR,** parameters for transmission (4.1.1-b.10) Output : **obj.descrambled_LLR**

• **HARQ process**

- Module that performs HARQ function in UE. When the received signal is NACK, the received signal and related information are stored in the corresponding HARQ_buffer. If the received signal is ACK, HARQ_buffer is initialized. If the new data indicator from the UE is 0 (in the case of retransmission), decoding is performed using the information in the previous buffer and the current retransmitted information.
- In this simulator, Chase combining and incremental redundancy based HARQ processes are basically implemented. Both methods are performed in rate matching block, refering the redundancy versions.

method_HARQ_process(obj,genie)
Input : **obj.descrambled_LLR,** genie.HARQ_process_index Output : **obj.descrambled_LLR_HARQ,** obj.HARQ_buffer

- **Channel decoding module**

- Figure 10 is the structure and procedure of channel encoding module based on corresponding to channel encoding module. The decoding module consists of detail modules – code block deconcatenation, rate dematching, LDPC decoding, code block desegmentation, and CRC check.

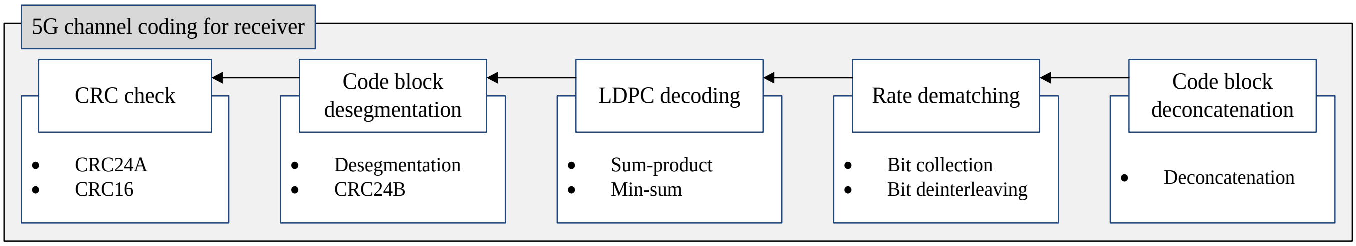

Figure 11. Structure and procedure of channel decoding module.

- In code block deconcatenation, the LLR sequence is received from the previous block. Assume that the input sequence and the output sequence of code block deconcatenation are denoted as g_k^{rx} and f_{rj}^{rx}. Then, the LLR sequence is corresponding to g_k^{rx}, and thus the f_{rj}^{rx} is deconcatenated as below:

$k=0$
for $k=0$ $r=0$ to $C-1$
    for $j=0$ to E_r-1
        $f_{r,j}^{\mathrm{rx}}=g_k^{\mathrm{rx}}$
        $k=k+1$
    end for
end for

- In rate dematching, the LLR sequence f_{rj}^{rx} is firstly deinterleaved as below:

for $r=0$ to $C-1$
    for $j=0$ to E_r/Q_m-1
        for $i=0$ to Q_m-1
            $e_{r,j+i\times E_r/Q_m}^{\mathrm{rx}}=f_{r,i+j\times Q_m}^{\mathrm{rx}}$
        end for
    end for
end for

The deinterleaved sequence $e_{r,j+i\times E_r/Q_m}^{\mathrm{rx}}$ is recovered as below, where $2Z_c$ punctured bits in LDPC encoding and *NULL* bits removed in bit selection step of rate matching are considered.

for $r=0$ to $C-1$

$j=0$, $k=0$

while (1)

if $(\mathrm{mod}(k_{0,rv_{id}}+j, N_{cb})+2Z_c < null_0)$ or $(\mathrm{mod}(k_{0,rv_{id}}+j, N_{cb})+2Z_c > null_{K-K'-1})$

if ($k == E_r$)

break;

else if ($E_r < N_{cb}$)

$d^{\mathrm{rx}}_{r,\mathrm{mod}(k_{0,rv_{id}}+j,N_{cb})+2\times Z_c} = e^{\mathrm{rx}}_{r,k}$

$k=k+1$

else if ($E_r > N_{cb}$)

if ($j < N_{cb}$)

$d^{\mathrm{rx}}_{r,\mathrm{mod}(k_{0,rv_{id}}+j,N_{cb})+2\times Z_c} = e^{\mathrm{rx}}_{r,k}$

else

$d^{\mathrm{rx}}_{r,\mathrm{mod}(k_{0,rv_{id}}+j,N_{cb})+2\times Z_c} = d^{\mathrm{rx}}_{r,\mathrm{mod}(k_{0,rv_{id}}+j,N_{cb})+2\times Z_c} + e^{\mathrm{rx}}_{r,k}$

end if

$k=k+1$

end if

end if

$j=j+1$

end while

end for

- $N_{NULL} = K-K'$ is the number of $NULL$ bits generated in block segmentation
- $null_k = K'+k,\ k=0,\ldots,K-K'$
- $k_{0,rv_{id}}$ is rv_{id} -th starting point in Table 3 (Starting position of different redundancy versions, k_0)

- In LDPC decoding, a bipartite graph is considered from corresponding to parity check matrix of LDPC. The bipartite graph is a graph whose nodes are separated into two parts, variable nodes and check nodes. There are N_{H} variable nodes and $M_{\mathrm{H}} = N_{\mathrm{H}} - K$ check nodes. N_{H} and M_{H} are the number of columns and rows of $\mathbf{H}$, respectively. Also, the M_{H} rows and N_{H} columns specify the M_{H} check nodes and the N_{H} variable nodes. The bipartite graph of an LDPC code is drown if the check node m is connected to variable node n whenever h_{mn} as element of $\mathbf{H}$ is 1, otherwise 0.
  For the convenience, the notation is presented as below:

  - $N(m)$: Variable nodes connected to check node m
  - $N(m)\backslash n$: Variable nodes connected to check node m except variable node n
  - $M(n)$: Check nodes connected to variable node n
  - $M(n)\backslash m$: Check nodes connected to variable node n except check node m
  - $bit \in \{0,1\}$
  - $f_n^{bit} = \Pr(y_n | d_n = bit)$
  - $L(f_n) = \log\frac{f_n^0}{f_n^1} = \frac{2y_n}{\sigma^2}$: LLR of $\frac{f_n^0}{f_n^1}$

- S_m : Event that parity check equation of check node m is satisfied
- q_{mn}^{bit} : Message from bit node n to check node m, $q_{mn}^{bit} = \Pr(d_n = bit \,|\, y_n, S_{m' \in M(n) \backslash m}|)$
- r_{mn}^{bit} : Message from check node m to variable node n, $r_{mn}^{bit} = \Pr(S_m \,|\, d_n = bit, \mathbf{y}|)$
- $L(q_{mn}) = \log \frac{q_{mn}^0}{q_{mn}^1}$: LLR of $\frac{q_{mn}^0}{q_{mn}^1}$
- $L(r_{mn}) = \log \frac{r_{mn}^0}{r_{mn}^1}$: LLR of $\frac{r_{mn}^0}{r_{mn}^1}$

Basically, the 2 LDPC decoding methods are considered. The one is the sum-product method and the other is the min-sum method.

Firstly, the sum-product method is described as in the following.

The input LLR of LDPC decoding block is $d_{r,n}^{\mathrm{rx}}$ and $L(f_n) = d_{r,n}^{\mathrm{rx}}$. Before initialization step, the LLRs $L(f_n)$ of $2Z_c$ punctured bits in LDPC encoding are set as 0 due to the uncertainty of them. Also, the LLRs $L(f_n)$ of *NULL* bits are set as $-\infty$ because -1 is inserted to *NULL* bits for simulations.

In initialization step, the LLR $L(q_{mn})$ is initialized by $L(q_{mn}) = L(f_n)$.

In check node update step, the LLR $L(r_{mn})$ is updated as below:

$$L(r_{mn}) = -\left[\prod_{n' \in N(m) \backslash n} \operatorname{sgn}\left(L(q_{mn'})\right)\right] \Phi\left(\sum_{n' \in N(m) \backslash n} \Phi(|L(q_{mn'})|)\right)$$

$$\operatorname{sgn}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases}$$

$$\Phi(x) = \log \tanh(\frac{x}{2})$$

In variable node update step, the LLR $L(r_{mn})$ is updated as below:

$$L(q_{mn}) = L(f_n) + \sum_{m' \in M(n) \backslash m} L(r_{m'n})$$

LLR of a posteriori probability of coded bit n, $L(q_n)$ is calculated as below:

$$L(q_n) = L(f_n) + \sum_{m \in M(n)} L(r_{mn})$$

The decoding procedure is repeated for per coded block $r = 0, \ldots, C-1$. For coded block r, the iterative decoding is performed. In exactly, the check node update step and the variable node update step are iteratively calculated until the maximum iteration. Therefore, $L(q_{mn})$ is calculated in variable node update step, and then fed back to check node update step in order to calculate $L(r_{mn})$.

Secondly, the min-sum method is described as in the following.

The all procedures are same except for check node update step. The function $\Phi(x)$ is positive and monotonically decreasing function for $x > 0$, and it is satisfied that $\Phi(\Phi(x)) = x$, since the inverse of $\Phi(x)$ is $\Phi(x)$ for $x > 0$. From the properties of $\Phi(x)$, $\Phi\left(\sum_{n' \in N(m) \backslash n} \Phi(|L(q_{mn'})|)\right)$ can be approximated by

$$\Phi\left(\sum_{n' \in N(m) \backslash n} \Phi(|L(q_{mn'})|)\right) \approx \Phi\left(\Phi(\min_{n' \in N(m) \backslash n} |L(q_{mn'})|)\right) = \min_{n' \in N(m) \backslash n} |L(q_{mn'})| \,.$$

Thus, the LLR $L(r_{mn})$ is updated as below:

$$L(r_{mn}) = -\left[\prod_{n'\in N(m)\backslash n} \operatorname{sgn}\left(L(q_{mn'})\right)\right] \min_{n'\in N(m)\backslash n} \left|L(q_{mn'})\right|$$

The output LLR sequence $L(q_n)$ denoted as $c_{r,k}^{\text{rx}}$, which is transferred to code block desegmentation.

- In code block desegmentation, the input sequence and output sequence is denoted as b_s^{rx} and $c_{r,k}^{\text{rx}}$, respectively. The desegmentation is performed as below:

```
s = 0
for r = 0 to E_r
        for k = 0 to K'-L-1
                b_s^rx = c_{r,k}^rx
                s = s+1
        end for
end for
```

- In CRC check, based on CRC generator polynomials, a decoding error is detected from b_s^{rx} as the input sequence of CRC check. CRC generator polynomials is dependent on A. Thus, if $A > 3824$, $g_{\text{CRC24A}}(D)$ is used, and otherwise, $g_{\text{CRC16}}(D)$ is used. If the decoding error is detected, then NACK is generated. Otherwise, ACK is generated.

method_decoding(obj,para,genie,ind_slot)
Input : **obj.descrambled_LLR** Output : **obj.decoded_bit, obj.feedback, obj.Result, obj.HARQ_buffer**

4.1.6-a UE_MIMO function block submodules for Downlink

- **Channel filtering module**

- Same with SISO case but to expand through multiple antenna case. For each of the transmitted antenna port and received antenna, perform same channel filtering as SISO case.

method_channel_filtering(obj, para, genie, ch_Output)
Input : **genie.OFDM_signal_stream,** parameters for scenario (4.1.1-a.11), parameters for

frame structure (4.1.1-a.1) Output : **obj.received_signal_stream,** obj.sigma_n_freq, obj.sigma_n_time

-

- **Synchronization module**

- Same as in SISO.

method_synchronization(obj, para, genie, ind_slot)
Input : **obj.received_signal_stream,** parameters for scenario (4.1.1-a.11), parameters for frame structure (4.1.1-a.1), parameters for OFDM (4.1.1-a.13) Output : **obj.received_signal_stream_sync**

-

- **OFDM demodulation module**

- Extended based on the number of transport blocks using the module used in UE_SISO.

method_OFDM_demodulation(obj,para)
Input : **obj.received_signal_stream_sync,** parameters for OFDM (4.1.1-a.13), parameters for frame structure (4.1.1-a.1) Output : **obj.received_resource_grid**

-

- **Channel_estimation module**

- Expands channel estimation method in SISO case for multiple antennas. If CSI-RSs do not use same REs, perform same channel estimation method as SISO case per antenna port. If CSI-RSs share same REs, use CDM (code division multiplexing) to maintain the orthogonality. For CDM, RSs several REs in the time domain which leads to the imperfect channel estimation if channel varies during that time duration. Therefore, we consider only block fading case. After removing interference with OCC, we estimate channel through LS.

- Effective channel estimation through DM-RS is the estimation of channel that data signal is actually served through. In this case, covariance matrix of the effective channel varies due to the precoder, so we only consider LS case. Other estimation method is same with CSI-RS case.

method_channel_estimation(obj, para, genie, ch_Output,ind_slot)
Input : **obj.received_resource_grid,** obj.channel_buffer Output : obj.channel_buffer, **obj.H_estimated_power,** obj.HV_estimated_error, obj.HV_estimated_error_ave, obj.H_estimated_error, obj.H_estimated_error_ave, obj.HV_estimated, obj.H_estimated, obj.HV_perfect

-

- **Data extraction module**

- Extended based on the number of transport blocks using the module used in UE_SISO.

method_Data_extraction(obj,para, genie)
Input : **obj.H_estimated_power, obj.received_resource_grid,** parameters for frame

structure (4.1.1-a.1) Output : **obj.Data_stream, obj.H_estimated_data, obj.Data_stream_indexf**

-

- **Detection module**

- For MIMO case, the detection method is same with SISO case in layer perspective.

method_detection(obj,para,genie)
Input : **obj.Data_stream, obj.H_estimated_data, obj.Data_stream_indexf,** obj.sigma_n_freq Output : **obj.LLR**

- **Delayer mapping module**

- Module that performs the delayer mapping of the extracted LLR contrary to BS layer mapping. It was created with reference to 6.3.3.2 in [1].

method_delayer_mapping(obj, genie)
Input : genie.num_codewords, genie.num_layers, genie.m_order Output : obj.delayered_LLR

-

- **Descrambling module**

- Extended based on the number of transport blocks using the module used in UE_SISO.

method_descrambling(obj,para,ind_slot)
Input : **obj.LLR,** parameters for transmission (4.1.1-a.10) Output : **obj.descrambled_LLR**

-

- **HARQ process module**

- The basic operation is the same as the HARQ process module in SISO. In MIMO environment in which two transport blocks are transmitted in a single subframe, the 2-bit new data indicator is transmitted from the BS through signaling, and the HARQ process proceeds based on the signal.

method_HARQ_process(obj,genie)
Input : **obj.descrambled_LLR,** genie.HARQ_process_index Output : **obj.descrambled_LLR_HARQ,** obj.HARQ_buffer

-

- **Channel decoding module**

- Extended based on the number of transport blocks using the module used in UE_SISO.

method_decoding(obj,para,genie,ind_slot)

Input : **obj.descrambled_LLR** Output : **obj.decoded_bit, obj.feedback, obj.Result, obj.HARQ_buffer**

4.1.6-b BS_MIMO function block submodules for Uplink (common parts with downlink case and SISO case are omitted)

- **Channel filtering module**

- Same with SISO case but to expand through multiple antenna case. For each of the transmitted antenna port and received antenna, perform same channel filtering as SISO case.

method_channel_filtering(obj, para, genie, ch_Output)
Input : **genie.OFDM_signal_stream,** parameters for scenario (4.1.1-a.11), parameters for frame structure (4.1.1-a.1) Output : **obj.received_signal_stream,** obj.sigma_n_freq, obj.sigma_n_time

- **Synchronization module**

- Module that compensates the received signal by estimating the frequency offset. Perfect synchronization is assumed.

method_synchronization(obj, para, genie, ind_slot)
Input : **obj.received_signal_stream,** parameters for scenario (4.1.1-a.11), parameters for frame structure (4.1.1-a.1), parameters for OFDM (4.1.1-a.13) Output : **obj.received_signal_stream_sync**

4.2 Input/Output design for modules

Module type	Input	Output
Feedback_reception	PMI, RI, CQI, CDI (channel direction information)	Modulation order, coding rate, precoder
RS_allocation	Antenna port, transmission mode, subframe index	CRS, DM-RS, CSI-RS
Resource_allocation	CRS position, subframe index	Position of PDCCH, PBCH, PSS, SSS, and PDSCH on Resource grid
Sync_generation	Subframe index	Resource grid containing sync signal
HARQ_process	Subframe index, feedback	HARQ buffer

Bit_generation	Bit number generated for each codeword	Bit stream arbitrarily generated as given bit number
Channel_encoding	Bit stream, coding rate, LDPC iteration generated for each codeword	Channel encoded bit stream
Scrambling	Channel encoded bit stream	Scrambled bit stream
Modulation_mapping	Scrambled bit sequence, modulation order	Modulated symbol sequence
Data_mapping	Modulated signal stream	Resource grid after data mapping
Power allocation module	RS, data signal	Power allocated RS, power allocated data signal
OFDM_generation	All the information of resource grid	OFDM signal stream
Layer mapping	Modulated signal stream	Layered signal stream
Precoding	Layered signal stream	Precoded signal stream
Chan_Matrix	Parameter module, Subframe index	channel coefficient matrix
Chan_H_fft	Parameter module	FFT form of the channel matrix
Channel filtering	Received OFDM signal stream	Signal stream after channel convolution and noise addition
Synchronization	Subframe index, received signal stream	Synchronized received signal stream
OFDM_demodulation	Synchronized received signal stream, SSS position	received_resource_grid through FFT
Data_extraction	Received resource grid	Data stream
Detection	Estimated channel, estimated effective channel	LLR for each codeword
descrambling	LLR for each codeword	Descrambled LLR for each codeword
Channel_estimation	CRS, DM-RS, CSI-RS, OFDM demodulated resource grid, transmission mode	Estimated channel, estimated effective channel

Feedback_calculation	Estimated channel, estimated effective channel, codebook, CDI codebook	PMI, RI, CQI, CDI
Channel decoding	Descrambled LLR, LDPC iteration	ACK information

5. Calibration methods

- **Link level calibrations**

- There are some evaluation scenarios for NR link level simulation in 3GPP. In R1-1701823, "Evaluation assumptions for Phase 1 NR MIMO link level calibration." is for calibration assumptions for NR channel modeling. And in R1-1703535, "Evaluation assumptions for Phase 2 NR MIMO link level calibration." is for calibration assumptions for NR data transmission.

- **Calibration method 1**

- For R1-1701823, assumptions are set in the table below.
- 'R1-1715252' is used for calibration. There are some calibration results for 'Phase 1 NR MIMO link level calibration' by many companies.

	Below 6GHz	**Above 6GHz**
Carrier Frequency	4 GHz	30 GHz
Subcarrier Spacing	15kHz	60kHz
bandwidth	20MHz	80MHz
Channel Model	CDL-A and CDL-B (delay spread =100ns, UE speed=3km/h)	
TXRU mapping	1D sub-array partition model	2D sub-array partition model
Beam Selection Method	- 1: DFT beam for the strongest cluster - 2: Best beam for the maximizing receive power	
Antenna configurations	BS: (M,N,P) = (4,4,2). UE: (M,N,P) = (1,1,2)	BS: (M,N,P) = (4,8,1). UE: (M, N, P) = (2,4,1);
Antenna element radiation pattern	BS: TR36.873 UE: Omni-directional	BS: Table A.2.1-6 in TR 38.802 UE: Table A.2.1-8 in TR 38.802
Metrics	CDF of receive SNR w/ beamforming at SNR=0dB	

- Simulation result for below 6GHz, CDL-A, and beam selection method 2 is below. The metric is CDF of receive SNR.

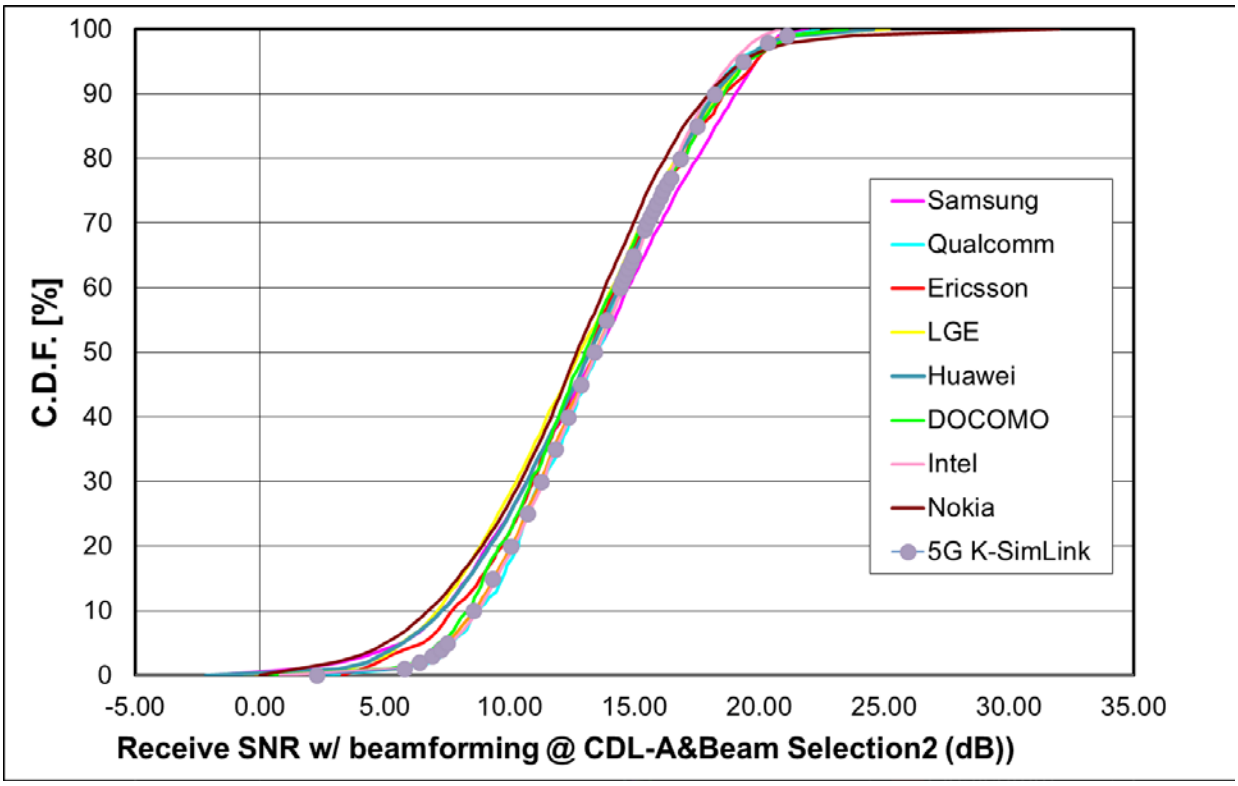

Figure 12. Calibration result for NR channel.

- **Calibration method 2**

- For R1-1703535, assumptions are set in the table below.
- 'R1-1715254is used for calibration. There are some calibration results for 'Phase 2 NR MIMO link level calibration' by many companies.

Carrier Frequency	4 GHz
Subcarrier Spacing	15kHz
Data allocation	8 Resource Blocks
Channel Model	CDL-A and CDL-B (delay spread =100ns, UE speed=3km/h)
TXRU mapping	1D sub-array partition (TR36.897)
Beam selection	- 1: DFT beam for the strongest cluster - 2: Best beam for the maximizing receive power
Antenna configurations	BS: (M,N,P) = (4,4,2), UE: (M,N,P) = (1,1,2)

Antenna element radiation pattern	BS: TR36.873 UE: Omnidirectional
BF scheme	Analog BF based on beam selection + Digital BF based on ideal SVD
MIMO mode	SU-MIMO with rank=1
UE receiver type	MMSE-IRC
MCS	Coding rate: 0.1354, Modulation: QPSK
DMRS channel estimation	2D MMSE (frequency – time correlation)
Metrics	BLER w/ beamforming as a function of transmission SNR ranging from -20 dB to 0dB

- Simulation result for below 6GHz, CDL-B, and beam selection method 2 is below. The metric is BLER.

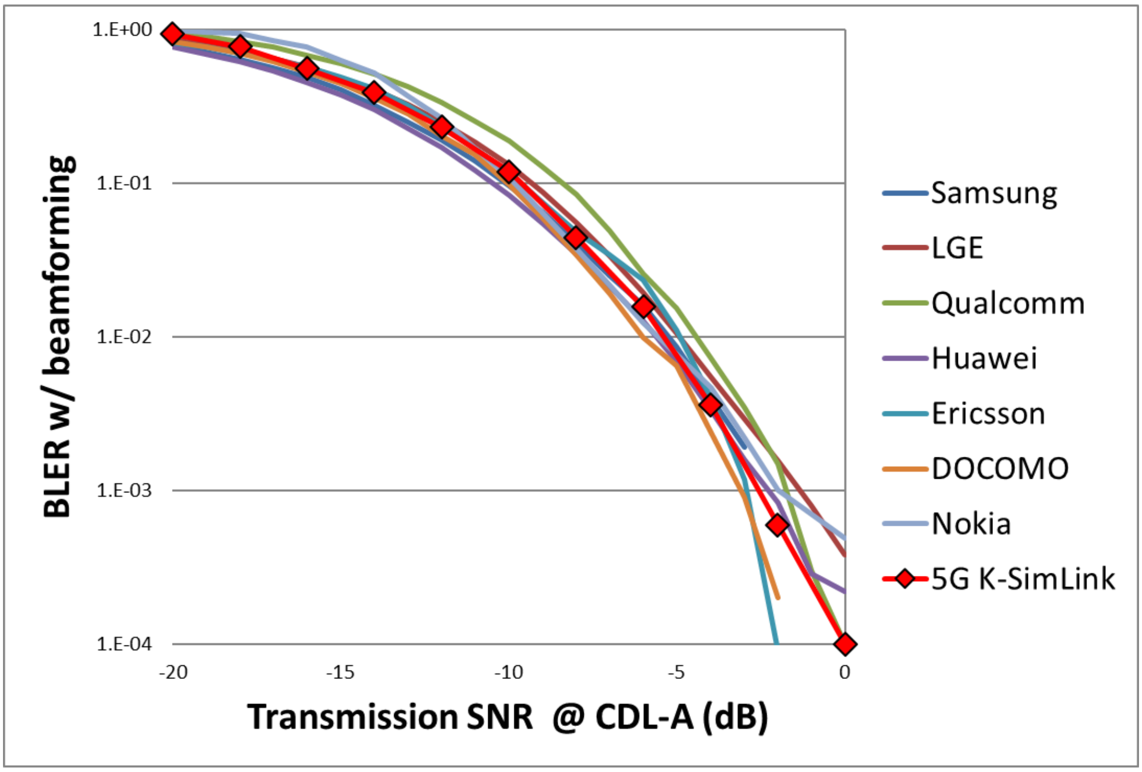

Figure 13. Calibration result for NR data transmission.

6. How to use

• **C++ guide**

- **Quick Installation & Execution Guide**
- **1. installation**
- **- Unzip LLS_DownLink_SUSISO_0912.zip**
- **- Below directories and files would appear**
-
- **o LLS_DownLink_SUSISO_0912**
- **2018-09-12 오전 12:52 <DIR> .**
- **2018-09-12 오전 12:52 <DIR> ..**
- **2018-09-12 오전 12:51 <DIR> bin**
- **2018-09-12 오전 12:53 <DIR> doc**
- **2018-09-12 오전 12:52 <DIR> MATLAB script code**
- **2018-09-12 오전 12:51 <DIR> src**
-
- **- Now, the installation is done.**
-
-
- **2. execution**
- **- move to bin directory as shown below.**
- **o LLS_DownLink_SUSISO_0912\bin**
-
- **2018-09-13 오전 12:39 <DIR> .**
- **2018-09-13 오전 12:39 <DIR> ..**
- **2016-06-17 오전 01:18 1,562,112 blas_win64_MT.dll**
- **2016-06-17 오전 01:18 26,180 blas_win64_MT.lib**
- **2018-09-12 오후 10:30 4,566,016 DL_SISOv0.41.exe**
- **2016-06-17 오전 01:18 8,342,016 lapack_win64_MT.dll**

- 2016-06-17 오전 01:18 246,138 lapack_win64_MT.lib
- 2018-07-18 오전 11:28 12,096 MATLAB_fixed_bit_stream_CQI1.csv
- 2018-07-18 오전 11:29 18,648 MATLAB_fixed_bit_stream_CQI2.csv
- 2018-07-18 오전 11:38 29,736 MATLAB_fixed_bit_stream_CQI3.csv
- 2018-07-18 오전 11:39 47,880 MATLAB_fixed_bit_stream_CQI4.csv
- 2018-07-18 오전 11:42 70,728 MATLAB_fixed_bit_stream_CQI5.csv
- 2018-07-18 오전 11:48 91,392 MATLAB_fixed_bit_stream_CQI6.csv
- 2018-07-18 오전 11:52 115,584 MATLAB_fixed_bit_stream_CQI7.csv
- 2018-07-18 오전 11:56 150,528 MATLAB_fixed_bit_stream_CQI8.csv
- 2018-06-13 오후 04:31 165,310 MATLAB_UE_Received_Signal_Stream_fixedRandom.csv
- 2018-09-12 오후 04:11 1,565 Plot_SUSISO.m
- 15개 파일 15,445,929 바이트
-
- - Type DL_SISOv0.41 <enter> to execute. (execution is preceeded by printing through the screen. Refer to end of this section.)
- - After 10~20 minutes, BLER and Throughput value will appear in the screen and two files(
- OUTPUT_CPP_BLER.csv, OUTPUT_CPP_Throughput.csv) would be generated
- - executaion option can be varied as.
- <command prompt> DL_SISOv0.4 -L [MATLAB | CPP] -R [Random | Fix] -S <positive integer> <Enter Key> (case insensitive)
- -L : chooce LDPC module
- CPP indicates loading C++LDPC module (default & only option: CPP)
- -R : Choose random number generation. Random indicates enable, Fix indicates disable (default: Random)
- -S : Set num_subfram. (default: 3)
-
-
- 3. Visualizing result through MATLAB plot
- (1) Execute MATLAB
- (2) Move to directory containing Plot_SUSISO.m
- (3) Through MATLAB, type Plot_SUSISO <enter> to execute
- (4) The result, Figure1 and Figure2 , will pop-up.
-
- Below is the execuation example.

- ++++++++++++++++++++++++++++++++[Execution Message]+++++++++++++++++++++++++++++++++++
-
-
- K:\Work\진척도 (주간보고)\LLS_DownLink_SUSISO_0912\bin>DL_SISOv0.41
-
- ┌──────────────────────────────────────┐
- │ DL_SISOv0.41.exe program started with the following options │
- │ - LDPC Module : orignal C++ LDPC CPP │
- │ - Random Fixation : UNFIXED │
- │ - Number of Subframe : 3 │
- └──────────────────────────────────────┘
-
-
-
- ★★★★★★★★★★★★【 Simulation Starts 】★★★★★★★★★★★★
- Simulation case : SUSISO
- Bandwidth : 10.00 [MHz]
- Channel type : CDL_A
- Channel mode : Block_Fading
- Subframe : 3
- Carrier frequency : 4.00[GHz]
- ★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
-
-
-
- ■░░░░░░░░░░░░░░░░░░【 cqi = 1, (1 of 8) 】░░░░░░░░░░░░■ CQI EXEC TIME: 0 ms (0.000 sec)
-
- ▶▷▷▶▷▷▶▷▷【 cqi = 1, SNR = -9.00 dB, (1 of 4) 】▶▷▷▶▷▷▶▷▷ 0 ms (0.000 sec)

- ——————————————【 cqi = 1, SNR = 1 (-9.00 dB), slot = 1 】——————————

  0 ms (0.000 sec)

- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 345, diffBitCnt = 231, ACK = 0
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 345, diffBitCnt = 231
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 4409, diffBitCnt = 3079
- ——————————————【 cqi = 1, SNR = 1 (-9.00 dB), slot = 2 】——————————

  1664 ms (1.664 sec)

- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 576, diffBitCnt = 0, ACK = 1
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 576, diffBitCnt = 0
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 5511, diffBitCnt = 1977
- ——————————————【 cqi = 1, SNR = 1 (-9.00 dB), slot = 3 】——————————

  981 ms (0.981 sec)

- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 576, diffBitCnt = 0, ACK = 1
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 576, diffBitCnt = 0
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 5742, diffBitCnt = 1746
- ——————————————【 cqi = 1, SNR = 1 (-9.00 dB), slot = 4 】——————————

  967 ms (0.967 sec)

- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 337, diffBitCnt = 239, ACK = 0
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 337, diffBitCnt = 239
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 4209, diffBitCnt = 3279
- ——————————————【 cqi = 1, SNR = 1 (-9.00 dB), slot = 5 】——————————

  1500 ms (1.500 sec)

- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 329, diffBitCnt = 247, ACK = 0
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 329, diffBitCnt = 247
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 4242,

diffBitCnt = 3246

- ────────────────────【 cqi = 1, SNR = 1 (-9.00 dB), slot = 6 】────────────────────
1580 ms (1.580 sec)
- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 464, diffBitCnt = 112, ACK = 0
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 464, diffBitCnt = 112
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 4751, diffBitCnt = 2737
-
- ▶▷▷▶▷▷▶▷▷【 cqi = 1, SNR = -4.67 dB, (2 of 4) 】▶▷▷▶▷▷▶▷▷ 8273 ms (8.273 sec)
- ────────────────────【 cqi = 1, SNR = 2 (-4.67 dB), slot = 1 】────────────────────
1539 ms (1.539 sec)
- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 304, diffBitCnt = 272, ACK = 0
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 304, diffBitCnt = 272
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 3861, diffBitCnt = 3627
- ────────────────────【 cqi = 1, SNR = 2 (-4.67 dB), slot = 2 】────────────────────
1510 ms (1.510 sec)
- LDPC_rx(): ch_coding.tx_a.n_elem = 576, ch_coding.rx_a.n_elem = 576 ⇒ sameBitCnt = 576, diffBitCnt = 0, ACK = 1
- UE::genie.bit_stream.n_elem = 576, UE.decoded_bit.n_elem = 576 ⇒ sameBitCnt = 576, diffBitCnt = 0
- UE::genie.coded_bit_stream.n_elem = 7488, UE.coded_bit.n_elem = 7488 ⇒ sameBitCnt = 5765, diffBitCnt = 1723
- ────────────────────【 cqi = 1, SNR = 2 (-4.67 dB), slot = 3 】────────────────────
969 ms (0.969 sec)
-
- --- 이하 생략 --

• **C++ guide (cont.)**

- Quick Source code compile Guide
- ======================
-

-
- 1. Install armadillo library
-
- - Download armadillo library and unzip to some directory as shown below. (http://arma.sourceforge.net/download.html)
-
- Directory & File Listing
- +++++++++++++++
-
- 2018-09-12 오후 10:47 <DIR> .
- 2018-09-12 오후 10:47 <DIR> ..
- 2016-06-17 오전 01:19 432 armadillo_icon.png
- 2016-06-17 오전 01:19 206,810 armadillo_joss_2016.pdf
- 2016-06-17 오전 01:19 225,088 armadillo_nicta_2010.pdf
- 2016-06-17 오전 01:19 142,613 arma_gmm_joss_2017.pdf
- 2016-06-17 오전 01:19 420,075 arma_gmm_spcs_2017.pdf
- 2016-06-17 오전 01:19 171,764 arma_spmat_icms_2018.pdf
- 2016-06-17 오전 01:19 19,810 CMakeLists.txt
- 2018-09-12 오후 10:47 <DIR> cmake_aux
- 2016-06-17 오전 01:19 428 configure
- 2016-06-17 오전 01:19 562,766 docs.html
- 2018-09-12 오후 10:47 <DIR> examples
- 2018-09-12 오후 10:47 <DIR> include
- 2016-06-17 오전 01:19 2,107 index.html
- 2016-06-17 오전 01:19 11,560 LICENSE.txt
- 2018-09-12 오후 10:47 <DIR> mex_interface
- 2018-09-12 오후 10:47 <DIR> misc
- 2016-06-17 오전 01:19 526 NOTICE.txt
- 2016-06-17 오전 01:19 234,439 rcpp_armadillo_csda_2014.pdf
- 2016-06-17 오전 01:19 19,243 README.md
- 2018-09-12 오후 10:47 <DIR> src
- 2018-09-12 오후 10:47 <DIR> tests

- 14개 파일 2,017,661 바이트
-
- - make dir named include and lib (ex: D:\Project\usr\include, D:\Project\usr\lib)
- - Copy files in ‘include’ directory to D:\Project\usr\include
- - In directory examples\lib_win64, copy all files to D:\Project\usr\lib
-
-
- **2. Way to use armadillo in Microsoft Visual Studio C++**
-
- **(1) Download armadillo library**
- **Enterprise Edition, Professional Edition, Community Edition (any edition is fine)**
- **(https://visualstudio.microsoft.com/ko/downloads/)**
-
- **(2) Generate new project in Visual Studio C++**
-
- **(3) Copy all files in src directory in LLS_DownLink_SUSISO to the directory generated in (2)**
-
- **(4) Through solution finder, add source code and header file. Add as default items but to add all of the LDPC subdirectory files also.**
- **(5) Load main_SUSISO.cpp through editor.**
-
- **(6) In upper part in Visual Studio C++, [Debug | Release] , choose "Debug" to choose x64 among [x86 | x64]**
-
- **(7) In Visual Studio C++, click [project] --> [(project name) properties "(project name) property page " to pop up..**
-
- **(8) In the left part of properties page, activate “C/C++” and click [additional directory] to add include directory of armadillo library from (1)**
-
- **(9) Open [Linker] tab in the left part to activate [general] tab then click [additional library directory] in the right part to add lib directory of armadillo library from (1)**
- **(10) Activate [Linker] tab’s [Input] tab and click [additional dependent property] to open this tab.**
-
- **(11) Click the uppermost window to write blas_win64_MT.lib;lapack_win64_MT.lib , then click to finally click [Confirm] in the properties page**
- **(12) Visual Studio C++ [Build(B)] --> [(Project name) Build(U)] to compile.**

- (All the warning can be ignored.)
-
- (13) After complie has done, at "%root of the project%\x64\Debug" (project name).exe file will be generated.

7. Change Log

- v0.1, 2017-11-20
  - Block diagrams has been added
- v0.2, 2017-12-10
  - Calibration scenario has been added
- v0.3, 2018-02-01
  - MIMO part for downlink has been added
- v0.4, 2018-03-03
  - Uplink part has been added
- v0.5, 2018-04-10
  - MIMO part for uplink has been added
- v1.0, 2018-05-20
  - Update the document with the latest code for beta test
- V2.0, 2018-12-30
  - Update code for the current standard and add C++ description.

8. Reference

[1] TS 38.211 : "NR; Physical channels and modulation (Release 15)"
[2] TS 38.212 : "NR; Multiplexing and channel coding (Release 15)"
[3] TS 38.213 : "NR; Physical layer procedures for control (Release 13)"
[4] TS 38.214 : "NR; Physical layer procedures for data (Release 13)"
[5] Q Wang, C Mehlführer, M Rupp, "Carrier frequency synchronization in the downlink of 3GPP LTE", in Proc. 21st Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2010), Istanbul, Turkey (Sep. 2010)
[6] Edward W.Jang, Jungwon Lee, Hui-Ling Lou and John M. Cioffi "On the Combining Schemes for MIMO Systems with Hybrid ARQ", IEEE Transactions on wireless communication, Feb. 2009.
[7] Christoph Stude, Andreas Burg, and Helmut Bolcskei "Soft-Output Sphere decoding: Algorithms and VLSI Implementation" IEEE Journal on Selected areas in communications, Vol. 26, No. 2, Feb. 2008.
[8] Yahong Rosa Zheng and Chengshan Xiao, Simulation Models With Correct Statistical Properties for Rayleigh Fading Channels, IEEE Transactions on Communications, Jun. 2003.
[9] IST-WINNER D1.1.2 P. Kyosti, et al., "WINNER II Channel Models", ver 1.1, Sep. 2007.
[10] 3GPP, 3rd generation partnership project; Technical specification group radio access network; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) radio transmission and reception (Release 8), 3GPP TS 36.104 V8.2.0 (2008-05)
[11] "Gudelines for evaluation of radio transmission technologies for IMT-2000", Rec. ITU-R M.1225 1
[12] M. Trivellato, F. Boccardi and H. Huang, "On transceiver design and channel quantization for downlink multiuser MIMO systems with limited feedback", IEEE Journal of Sel. Areas in Commun., vol. 26, no. 8, Oct. 2008.
[13] Y. Shen and E. Martinez, "Channel estimation in OFDM systems", Application Note, Freescale Semiconductor, 2006.
[14] Mobile and wireless communications Enabler for the saTwenty-twenty information (METIS) ICT-317669-METIS/D6.1 "Simulation guidelines"
[15] Proakis, J. (2005). Digital Communications.
[16] Nokia. (2007). R4-071640, Ideal simulation results for PDSCH in AWGN. TSG RAN WG4 Meeting #44bis. http://www.3gpp1.org/ftp/specs/html-info/TDocExMtg--R4-44b--26624.htm
[17] Ericsson. (2008). R4-080538, Collection of PDSCH ideal results with practical channel estimation. TSG WG4 Meeting #46. http://ftp.3gpp.org/Specs/html-info/TDocExMtg--R4-46--26787.htm
[18] Motorola (2007). R4-07180, Agreed UE demodulation simulation assumption TSG WG4 Meeting #44 bis.
[19] TR 38.901 : "Study on channel model for frequencies from 0.5 to 100 GHz"
[20] R1-1707604 "Discussion on DL beam management," LG Electronics, May. 2017.
[21] P. M. Woodward and J. D. Lawson, "The Theoretical Precision with Which an Arbitrary Radiation Pattern may be Obtained from a Source of Finite Size," J IEE-Part III: Radio and Communication Engineering, vol. 95, pp. 363–370, Sep. 1948.