2018

قسم هندسة الالكترونيات والاتصالات الكهربائية

كلية الهندسة - جامعة القاهرة

LOW POWER DESIGN OF THE PHYSICAL LAYER OF NARROW BAND IOT LTE

UPLINK DIGITAL TRANSMITTER

LOW POWER DESIGN OF THE PHYSICAL LAYER OF NARROW BAND IOT LTE UPLINK DIGITAL TRANSMITTER

Ahmed Shawky Ahmed Hassan

Ahmed Taha Fathy Ali Mohamed

Ali Mohamed Khaled Mohamed

Basma Hesham Mohamed Salah-Elden

Essraa Ahmed Magdy Ezz-Elden

Manar Mohsen Mostafa Abd El-Aziz

Under the supervision of

Associate prof. Hassan Mostafa

Si-Vision

A thesis Submitted to the Faculty of Engineering at Cairo University

in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

in

Electronics and Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

JULY 2018

Abstract

With the new age of technology, and the release of the Internet of Things (IoT) revolution, there is a need to connect a wide range of devices with varying throughput and performance requirements. In this thesis, a digital transmitter of Narrow Band Internet of Things (NB-IoT) is proposed which is targeted towards very low power, delay insensitive IoT applications with low throughput requirements. NB-IoT is a new cellular technology introduced in 3GPP Release 13 for providing wide-area coverage for the IoT. The low cost receivers for such devices will have very low complexity, consume very low power and hence will run for several years. In this thesis the implementation of the data path chain of digital uplink transmitter is presented. The standard specifications were studied carefully to determine the required design parameters for each block. Every block of the transmitter was modelled by using MATLAB and designed using HDL then verified by using test benches to compare the results of RTL simulations and MATLAB models. In addition, the blocks were synthesized with the 130 nm technology to produce the netlist equivalent and calculate the power and area of every block. Power and area performance were compared for various designs to decide which design achieves the better performance. Eventually, the whole chain blocks were integrated and verified to achieve the required performance, data rate, coverage, delay and throughput. The hardware implementation of the system is presented as well as simulation results. Results were compared with the MATLAB models to make sure that the functionality of the system is achieved.

Acknowledgments

This dissertation does not only hold the results of the year work, but also reflects the relationships with many generous and stimulating people. This is the time where we have the opportunity to present our appreciation to all of them.

First, to our advisor, Dr. Hassan Mostafa and our advisors at si-vision, Eng. Ahmed Ibrahim, Eng. Amr el Hossiny and Eng. Rodaina, we would like to express our sincere gratefulness and appreciation for their excellent guidance, caring, patience, and immense help in planning and executing the work in a timely manner. Their great personality and creativity provided us with an excellent atmosphere for work, while their technical insight and experience helped us a lot in our research.

Of course, we will never find words enough to express the gratitude and appreciation that we owe to our families. Their tender love and support have always been the cementing force for building what we achieved. The all-round support rendered by them provided the much needed stimulant to sail through the phases of stress and strain.

Finally, Thanks are due to previous GP members. Their thesis was supportive, informative and guided us in many critical issues.

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviation

$M_{\mathrm{sc}}^{\mathrm{NPUSCH}}$	Scheduled bandwidth for uplink transmission, expressed as a number of subcarriers
$M_{\mathrm{symb}}^{\mathrm{layer}}$	Number of modulation symbols to transmit per layer for a physical channel
$M_{\mathrm{identical}}^{\mathrm{NPUSCH}}$	Number of repetitions of identical slots for NPUSCH
$M_{\mathrm{symb}}^{\mathrm{ap}}$	Number of modulation symbols to transmit per antenna port for a physical channel
$N_{\mathrm{sc}}^{\mathrm{RU}}$	Number of consecutive subcarriers in an UL resource unit for NB-IoT
$N_{\mathrm{sc}}^{\mathrm{UL}}$	Number of subcarriers in the frequency domain for NB-IoT
$N_{\mathrm{slots}}^{\mathrm{UL}}$	Number of consecutive slots in an UL resource unit for NB-IoT
$N_{\mathrm{symb}}^{\mathrm{UL}}$	Number of SC-FDMA symbols in an uplink slot
$N_{\mathrm{RB}}^{\mathrm{UL}}$	Uplink bandwidth configuration, expressed in multiples of $N_{\mathrm{sc}}^{\mathrm{RB}}$
$N_{\mathrm{sc}}^{\mathrm{RB}}$	Resource block size in the frequency domain, expressed as a number of subcarriers
$N_{\mathrm{symb}}^{\mathrm{UL}}$	Number of SC-FDMA symbols in an uplink slot
$\mathrm{N}_{\mathrm{RB}}^{\mathrm{min,UL}}$	The minimum number of resource blocks in uplink
$\mathrm{N}_{\mathrm{RB}}^{\mathrm{max,UL}}$	The maximum number of resource blocks in uplink.
$M_{\mathrm{rep}}^{\mathrm{NPUSCH}}$	Scheduled number of repetitions of a NPUSCH transmission
$M_{\mathrm{bit}}^{(q)}$	Number of coded bits to transmit on a physical channel [for codeword q]
$M_{\mathrm{symb}}^{(q)}$	Number of modulation symbols to transmit on a physical channel [for codeword q]
$N_{\mathrm{ID}}^{\mathrm{Ncell}}$	Narrowband physical layer cell identity
n_{RNTI}	Radio network temporary identifier
n_{f}	System frame number
n_{s}	Slot number within a radio frame
q	Codeword number
T_{f}	Radio frame duration
Δf	Subcarrier spacing
T_s	Basic time unit
T_{slot}	Slot duration
N_L	Number of Layers
rv_{idx}	Redundancy Version
QAM	Quadrature amplitude Modulation
QPSK	Quadrature Phase Shift Keying

BPSK	Binary Phase Shift keying
FDD	Frequency division duplexing
TDD	Time division duplex
LAA	Licensed-Assisted Access
SC-FDMA	Single carrier frequency division multiple access
DCI	Downlink Control Information
NPUSCH	Narrowband Physical Uplink Shared Channel
PMI	Precoding Matrix Indicator
RI	Rank Indication
CRC	Cyclic Redundancy Check
CQI	Channel Quality information
HARQ-ACK	Hybrid Automatic Repetition Request Acknowledge
ASK	Amplitude Shift Keying
FSK	Frequency Shift Keying
PSK	Phase Shift Keying
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
OFDM	Orthogonal Frequency Division Multiplexing
DFT	Discrete Fourier Transform
PAPR	Peak to Average Power Ratio
LTE	Long Term Evolution
SNR	Signal to Noise Ratio
SQNR	Signal to Quantization Noise Ratio

Chapter 1

Introduction

1.1. Motivation

Over the past decade, there has been a tremendous growth in the number of wireless devices. As wireless technology matures, this number is expected to grow at an even higher rate with the goal being able to connect every physical device to the internet. This presents a huge opportunity for wireless system designers as 99.4 percent of the physical devices are still unconnected. The idea behind IoT is to have a network of computing devices, vehicles, embedded systems with sensors etc which generate, exchange and process data intelligently and continuously. About 50 billion devices are estimated to be connected with IoT by 2020, up from about 200 million in the year 2000 and 10 billion in 2013. With such a massive number of devices connected to each other, machine to machine (M2M) communication is needed where devices would communicate with each other without any human interaction. It is expected that the revenue from M2M devices will grow from $200 million in 2011 to $1.2 trillion in 2022. Some of the services which might need machine type communication (MTC) include consumer electronics, security and public safety, automotive, utilities, remote maintenance, payments, health and smart cities. Until now, most of the wireless technologies were focused on improving the quality and performance of consumer electronic devices intended for human communication. MTC, however, is different from human communication as the throughput/delay/power requirements vary to a great extent based on the application and thus, require different connectivity solutions.

While services involving consumer electronics, building security and maintenance can be served by a local area network (LAN), services such as remote maintenance (sensors, vending machines etc), utilities (power, gas, water, heating etc), smart cities etc need a wide area network (WAN). Since the cellular technology is pretty mature and is already deployed in most parts of the world, cellular based MTC systems can provide a solution for the IoT applications which need a WAN. LTE is one of the latest and widely accepted, high speed wireless communication standard by 3rd generation partnership project (3GPP). The standard is revised regularly, in order to accommodate additional channel conditions and provide better connectivity with higher data rates and efficient resource utilization.

However, LTE was designed for high speed communication and is not optimized for applications that need to support a potentially large number of low-rate, low power and delay tolerant devices. These low cost devices, typically used in applications such as sensors, remote maintenance and tracking, health-care, utilities etc, are expected to have very low complexity, low mobility and low power consumption with a very long battery life. Hence, it is desirable to develop LTE based wireless systems; suitable for low data-rate and low power IoT applications.

1.2. Background

Realizing the importance of IoT for low-power and low cost applications with extended coverage and very long battery life, 3GPP introduced narrowband IoT (NB-IoT), as a part of their LTE-Release-13. Although NB-IoT is based on LTE, but it's a new radio-access technology as it is not fully backward compatible with the existing LTE devices. However, it can be easily integrated in the existing LTE network by allocating some of the time and frequency resources to NB-IoT.

NB-IoT occupies 180 kHz of spectrum, which is substantially smaller than LTE bandwidths of 1.4–20 MHz.

Spectrum, NB-IoT is designed to support three different deployment scenarios:

1. Stand alone: utilizing a 200 kHz band used by global system for mobile communication (GSM) frequencies as shown in figure 1.2-1.
2. Guard band: occupying a 180 kHz wide physical resource block in the guard band of existing LTE carrier band as shown in figure 1.2-2.
3. In band: occupying one physical resource block within the LTE carrier bandwidth shown in figure 1.2-3.

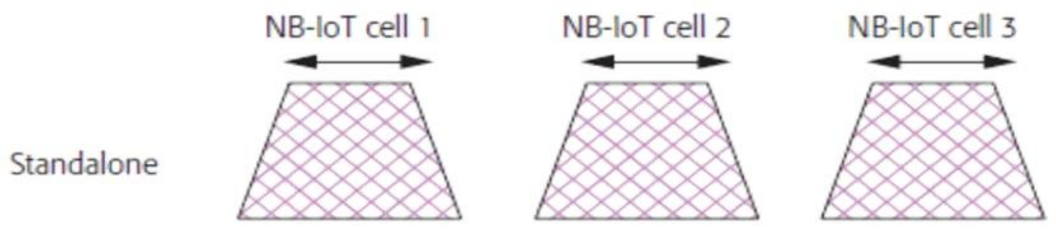


Figure 1.2-1: Standalone of NBIOT in GSM

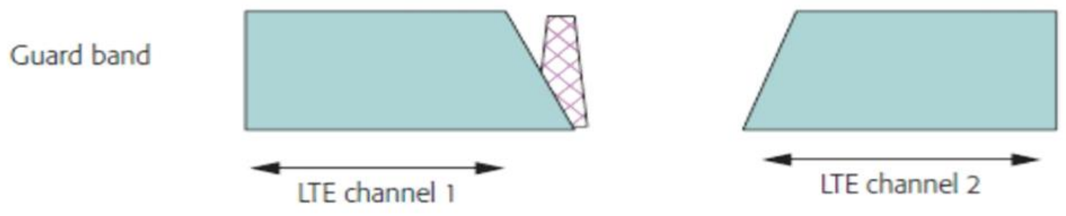


Figure 1.2-2: NBIOT in guard band

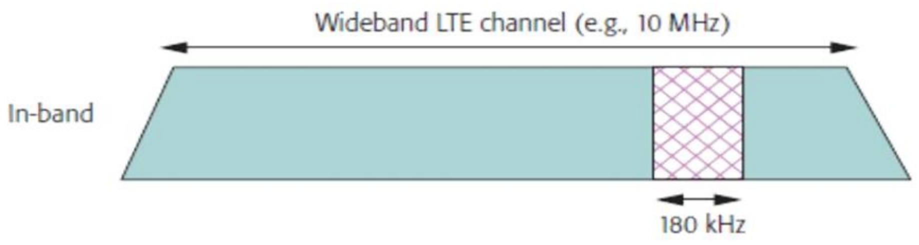


Figure 1.2-3: NBIOT in band

Thus, NB-IoT reuses the existing LTE design with respect to physical layer processing which reduces the time required to develop the NB-IoT devices to a great extent.

1.3. Purpose

As discussed in the previous section, NB-IoT provides a provision to allocate a single resource block (RB) to each user, instead of at-least six resource blocks in existing LTE. A single RB corresponds to a bandwidth of 180 kHz (and hence the name narrow-band) as compared to the minimum bandwidth of 1.4 MHz in conventional LTE systems. The resource block concept is discussed in later chapters. The purpose of this thesis is to develop a new, low power, wireless communication system targeted towards IoT applications, having very low throughput requirements with relaxed transmission delay requirements. The overall system, similar to NB-IoT, will be based on LTE and will have very low complexity, consume low power and have a long battery life. This system is able to allocate a very narrow bandwidth of 15 kHz to each user while retaining the existing LTE physical layer structure.

1.4. Problem Formulation

NB-IoT standard allocates one resource block per user irrespective of the throughput requirements of the application. A single resource block comprises of twelve subcarriers, each separated by 15kHz in frequency, giving a bandwidth of 180kHz. This might not be an efficient way of using the available spectrum when the throughput requirement is extremely low. In extreme cases, when the data-rate requirement is very low, we might as well allocate each subcarrier to a different user. This would enable a very efficient use of the available spectrum with a multiplexing gain of twelve. The transmitter in this case will be a NB-IoT transmitter with each subcarrier carrying data for a separate user.

1.5. Limitations

In this thesis, we study the physical layer properties of the transmitter architecture, with a focus on the uplink path of the system. Problems related to resource allocation and higher layer management is not discussed as a part of the thesis. Further, since the target data-rate and power requirements are low, only lower order modulation schemes, for instance, binary phase-shift keying (BPSK) and quadrature phase-shift keying (QPSK).

1.6. Frame Structure

Downlink, uplink and sidelink transmissions are organized into radio frames with T_f = T_s*307200 = 10ms duration where T_s is the number of time units =1 / (1500*2048) seconds.

<u>Three radio frame structures are supported:</u>

- Type 1, applicable to FDD only.

- Type 2, applicable to TDD only.

- Type 3, applicable to LAA secondary cell operation only.

The difference between FDD and TDD is that in TDD system (Time division duplex) same frequency band FC is used by both Transmit and receive path at different time instants while FDD system (Frequency division duplex) different frequency bands Fc1 and Fc2 are used by transmit and receive paths at same time instant as shown in figure 1.6-1.

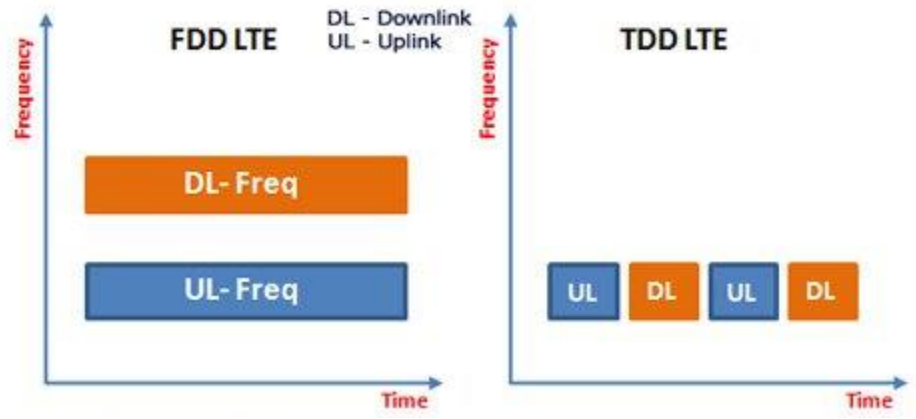


Figure 1.6-1: FDD and TDD differences

In NBIOT-LTE we use frame structure type 1 which is applicable to both full duplex and half duplex FDD only , Each radio frame is T_f= 307200*T_s = 10ms long and consists of 10 subframes of length 30720 * T_s= 1ms numbered from 0 to 9.

For subframes using Δf =15 kHz, subframe i is defined as two slots, 2i and 2i +1, of length T_{slot}= 15360 * T_s= 0.5 ms each.

For FDD, 10 subframes are available for downlink transmission and 10 subframes are available for uplink transmissions in each 10 ms interval. Uplink and downlink transmissions are separated in the frequency domain. In half-duplex FDD operation, the UE cannot transmit and receive at the same time while there are no such restrictions in full-duplex FDD.

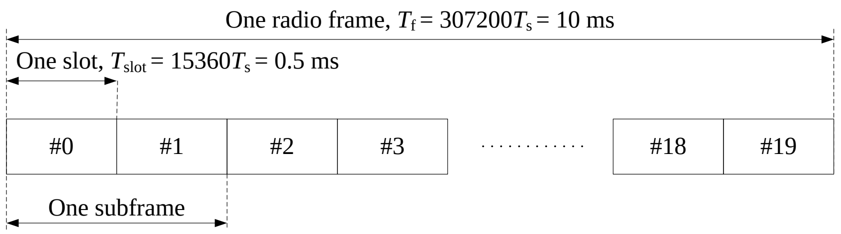

One radio frame, $T_f = 307200T_s = 10$ ms
One slot, $T_{slot} = 15360T_s = 0.5$ ms
#0
#1
#2
#3
#18
#19
One subframe

Figure 1.6-2: Frame structure type 1

1.7. Slot Structure

Uplink waveform in LTE-NB is the same as in legacy LTE Uplink. That is, LTE-NB Uplink is also using SC-FDMA. But there is some differences between LTE-NB and legacy LTE in terms of structure of uplink signal. In addition, there is a new unit called RU (Resource Unit) that exists in LTE-NB but not used in legacy LTE.

1.7.1. Subcarrier Spacing

There are two different types of sub-carrier spacing in LTE-NB. One is 15 KHz (this is same as in legacy LTE) and the other one is 3.75 KHz.

Table 1.7.1-1: Subcarrier spacing and corresponding number of subcarriers per UL

Subcarrier spacing	N_{sc}^{UL}	T_{slot}
$\Delta f = 3.75kHz$	48	61440*T_s
$\Delta f = 15$ kHz	12	15360*T_s

In this thesis we are using 15 KHz subcarrier spacing.

1.7.2. Resource Grid

The transmitted signal in each slot is described by one or several resource grids of N_{RB_UL} * N_{SC_RB} subcarriers and N_{symb_UL} SC-FDMA symbols. The quantity N_{RB_UL}depends on the uplink transmission bandwidth configured in the cell and shall fulfil.

$$N_{RB}^{min,UL} \leq N_{RB}^{UL} \leq N_{RB}^{max,UL}$$

Where $N_{RB}^{min,UL}$ and $N_{RB}^{max,UL}$ are the smallest and largest uplink bandwidths.

N_{RB_UL} is number of resource blocks in uplink, N_{SC_RB} is Number of subcarriers resource blocks, $N_{symb_UL}$is number of symbols per uplink, $N_{RB}^{min,UL}$ is the minimum number of resource blocks in uplink and $N_{RB}^{max,UL}$ is the maximum number of resource blocks in uplink.
Resource grid in a frame based on 15 KHz subcarrier spacing, Number of subcarriers is easily calculated to 12 sub-carriers within 180 KHz BW (LTE-NB System Bandwidth), there are 20 slots within a radio frame. Representing this case in a graphical format, it becomes as shown in figure 1.7.2-1. Basically this is the same as in legacy LTE uplink resource grid.

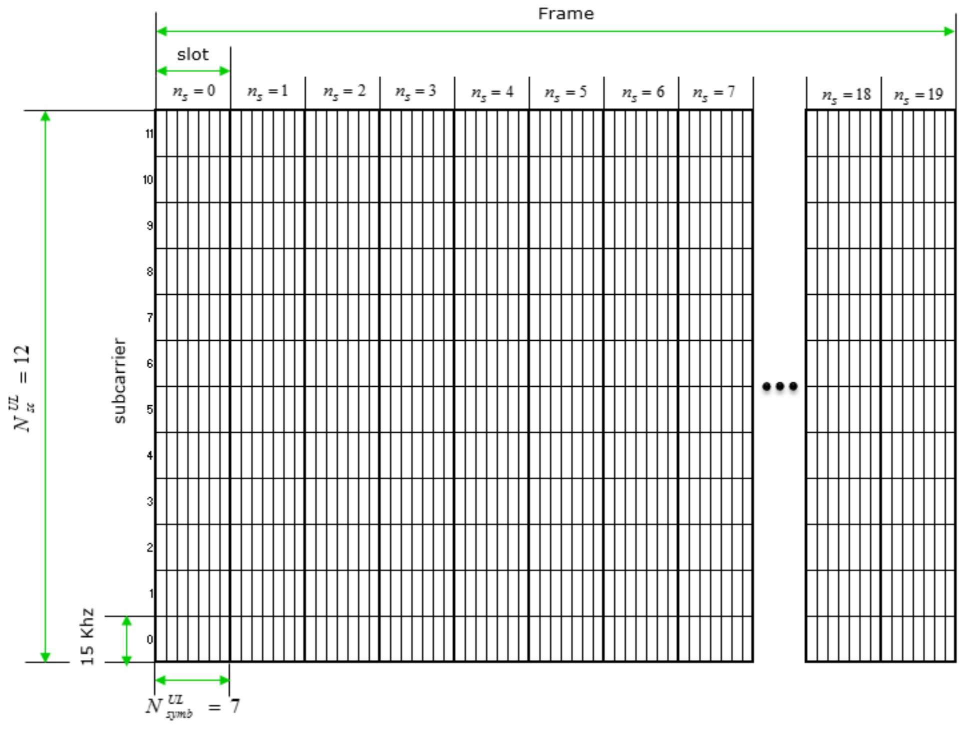


Figure 1.7.2-1: Resource grid with 15kHz spacing

If a 3.75 KHz subcarrier spacing resource grid frame is drawn, 48 subcarriers exist in the 180 KHz BW (LTE-NB System Bandwidth), with 5 slots within a radio frame. Translating this case into a graphical format, it would become as the following figure 1.7.2-2.

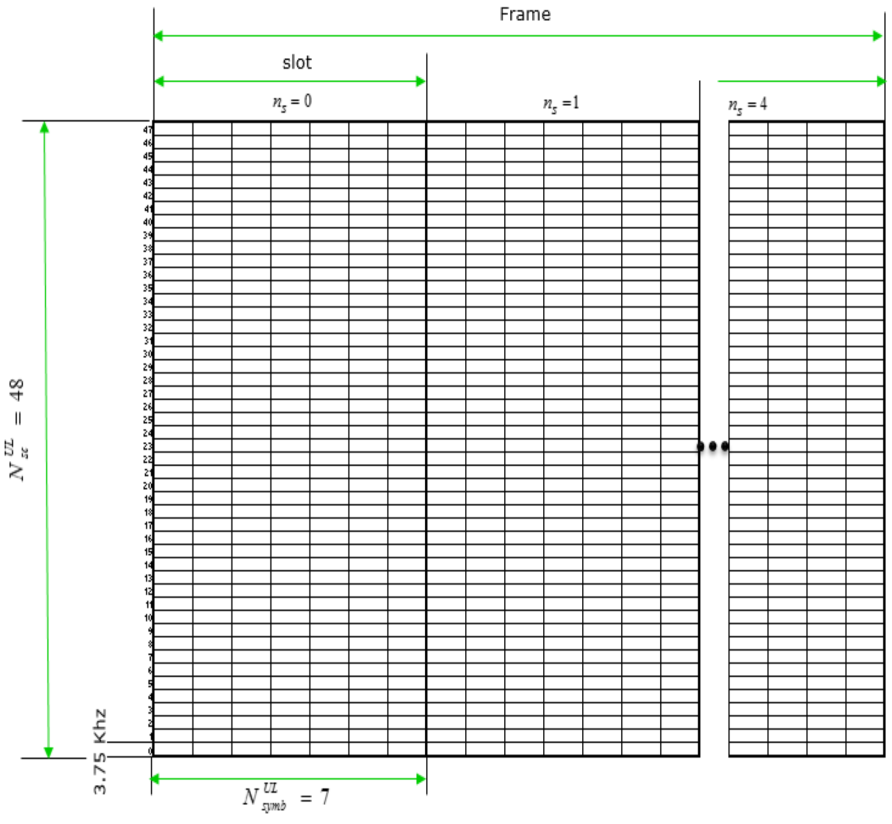


Figure 1.7.2-2: resource grid with 3.75kHz spacing

Comparing 3.75 KHz resource grid and 15 KHz resource grid in the two figures 1.7.2-1 and 1.7.2-2, some points can be noticed the followings:

- Sub-carrier spacing in 3.75 KHz became narrower by 4 times comparing to 15 KHz resource grid.
- Symbol length in 3.75 KHz resource grid became longer by 4 times comparing to 15 KHz resource grid (this is a basic property for OFDM. The narrower sub-carrier spacing is, the longer symbol length).
- Length of a Radio Frame is defined to be the same (10 ms) in both 3.75 KHz resource grid and 15 KHz resource grid.
- The number of slots within a radio frame in 3.75 KHz resource grid became smaller by 4 times comparing to 15 KHz resource grid.
- The number of OFDM symbols within a slot is the same (=7) in both 3.75 KHz resource grid and 15 KHz Grid.

1.7.3. Resource unit

A kind of new concept in LTE-NB UL resource assignment comparing to legacy LTE, LTE-NB introduce a new resource unit called RU (Resource Unit) as a basic unit for NPUSCH allocation. This unit can take several different types of configurations as defined in the table 1.7.3-1, noticing that NPUSCH format 1 and 15 KHz subcarrier spacing is assumed in this thesis.

Table 1.7.3-1: Number of subcarriers per RU in each carrier spacing

NPUSCH	Δf	N_{sc}^{RU}	N_{slots}^{UL}	N_{symb}^{UL}
1	15kHz	1	16	7
		3	8	
		6	4	
		12	2	

For NPUSCH format 1, Sub Carrier Spacing = 15 KHz, the number of sub carrier in one RU is 1 and the number of slots within the RU is 16. The graphical representation that I interpreted is as shown below the table the red slots will send to the eNodeB.

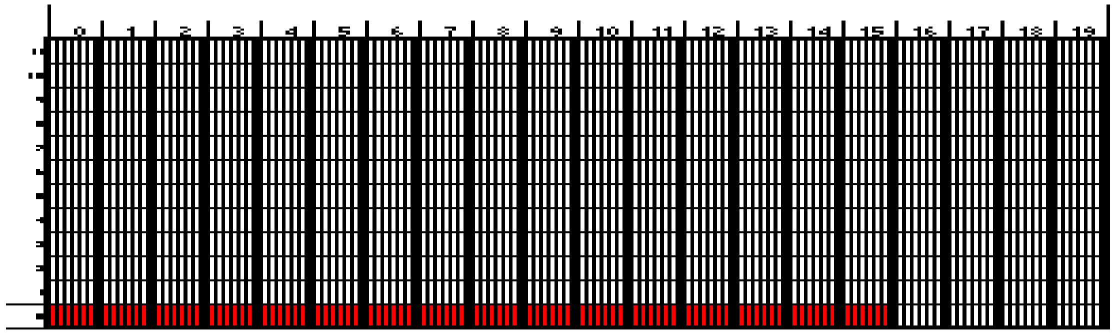


Figure 1.7.3-1: Example on NPUSH format1 15KHZ subcarrier spacing

1.7.4. Resource Allocation

In standard 213, the narrow band section describes The DCI, which is a Downlink Control Information. The resource allocation information in uplink, DCI format N0 for NPUSCH transmission indicates to a scheduled UE

- A set of contiguously allocated subcarriers of a resource unit determined by the Subcarrier indication field in the corresponding DCI.
- A number of resource units determined by the resource assignment field in the corresponding.
- A repetition number determined by the repetition number field in the corresponding DCI.

Table 1.7.4-1: DCI format N0 information

Field	# of Bits
Flag for format N0/format N1 differentiation	1
Subcarrier indication	6

Resource Assignment	3
Scheduling delay	2
Modulation and coding scheme	4
Redundancy version	1
Repetition number	3
New data indicator	1
DCI subframe repletion number	2
Total number of Bits	23

– <u>For the second field "subcarrier indication":</u>

Table 1.7.4-2: Allocated subcarriers for NPUSCH with delta F=15KHz

Subcarrier indication field (I_{sc})	**Set of Allocated subcarriers (n_{sc})**
0 – 11	I_{sc}
12-15	$3(I_{sc} - 12) + \{0,1,2\}$
16-17	$6(I_{sc} - 16) + \{0,1,2,3,4,5\}$
18	$\{0,1,2,3,4,5,6,7,8,9,10,11\}$
19-63	Reserved

It shows which subcarriers should be mapped in resource grid.

– <u>For the third field "Resource assignment":</u>

Table 1.7.4-3: Number of resource units

I_{RU}	N_{RU}
0	1
1	2
2	3
3	4
4	5
5	6
6	8
7	10

This allows us to know Number of resource units.

– <u>For the "Modulation and coding scheme number" field:</u>

Table 1.7.4-4: Modulation order and transport block index

MCS Index I_{MCS}	**Modulation Order** Q_m	**TBS Index** I_{TBS}
0	1	0
1	1	2

2	2	1
3	2	3
4	2	4
5	2	5
6	2	6
7	2	7
8	2	8
9	2	9
10	2	10

It indicates the modulation order whether it's QPSK or BPSK and from the Transport block index, the transport block size can be obtained from table 1.7.4-5.

Table 1.7.4-5: Transport block size.

I_{TBS}	I_{RU}							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	16	32	56	88	120	152	208	256
1	24	56	88	144	176	208	256	344
2	32	72	144	176	208	256	328	424
3	40	104	176	208	256	328	440	568
4	56	120	208	256	328	408	552	680
5	72	144	224	328	424	504	680	872
6	88	176	256	392	504	600	808	1000
7	104	224	328	472	584	712	1000	1224
8	120	256	392	536	680	808	1096	1384
9	136	296	456	616	776	936	1256	1544
10	144	328	504	680	872	1000	1384	1736
11	176	376	584	776	1000	1192	1608	2024
12	208	440	680	1000	1128	1352	1800	2280
13	224	488	744	1032	1256	1544	2024	2536

In our chain we made it on maximum transport block size 2536 in release14.

Chapter 2

Transmitter uplink chain and sub-blocks' function

2. Transmitter Uplink Chain

The function of each block in the full NarrowBand IOT LTE chain will be introduced in this chapter. This chain as shown in figure 2-1 consists of 9 Main blocks, a control unit and the upper layer role is to supply our chain with the necessary parameters.

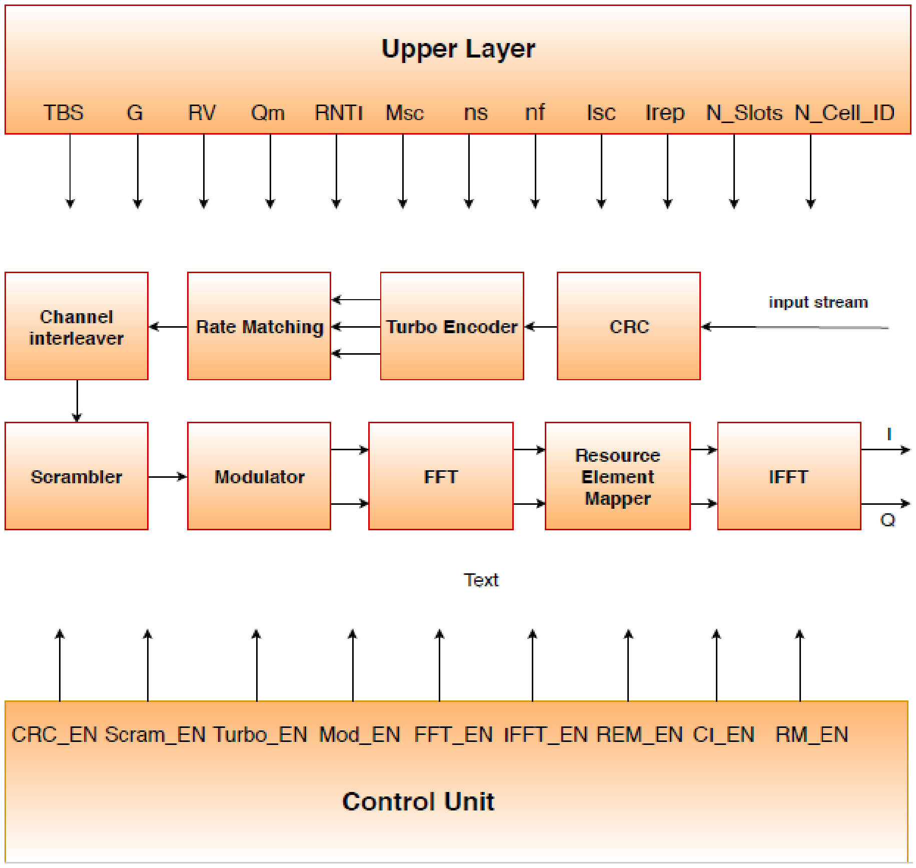


Figure 2-1: Full Chain Block Diagram

2.1. Channel coding, multiplexing and interleaving

2.1.1. Cyclic Redundancy Check (CRC)

Cyclic Redundancy Check is an error detection technique, it is used to calculate parity bits that added to the transmitted data and at the receiver it divides the received data by the same polynomial if result is zero then data is correct else there is an error in the received data.

2.1.2. Turbo Encoder

Turbo codes is the most exciting and potentially important development in coding theory in recent years and has opened a whole new way of looking at the problem of constructing good codes and decoding them with low complexity, turbo code has become the coding technique of choice in many communication and storage systems due to its near Shannon limit error correction capability, its function is to help the receiver to correct the received data.

2.1.3. Rate Matching

Rate matching in NPUSCH is a very important block in baseband processing. The basic function of rate matching module is to match the number of bits in transport block to the number of bits that can be transmitted in the given allocation, it also controls the rate as the turbo encoder gives 1/3 rate, we can increase or decrease rate by using this block according to DCI and channel quality information. By means of rate matching, any arbitrary code rate can be achieved from a fixed-rate mother code.

2.1.4. Data Multiplexing and Channel interleaver

The main function of this block is control and data multiplexing which is performed such that HARQ-ACK information is present on both slots and is mapped to resources around the demodulation reference signals.

In addition, the multiplexing ensures that control and data information are mapped to different modulation symbols, however in narrowband we have no control data multiplexed, so we keep multiplexing block and channel interleaver to overcome burst errors.

As for the channels with deep fading characteristics, errors often occur in bursts (affecting consecutive bits), these consecutive bits may be of the same symbol, code word , bit before coding so this burst noise will result in a high probability of error at the receiver. The channel interleaver overcomes this type of noise by changing this burst error into a group of discrete errors by reordering the bits so that the error happens in one bit only inside a code word.

2.2. Physical Channel and modulation

2.2.1. Scrambler

A scrambler is an algorithm that converts an input bits stream into a seemingly random output stream of the same length (e.g., by pseudo-randomly selecting bits to invert), thus avoiding long sequences of bits of the same value; randomizer is also referred to as a scrambler. It facilitates the work of a timing recovery circuit, automatic gain control and other adaptive circuits of the receiver (eliminating long sequences consisting of '0' or '1' only).

2.2.2. Modulation Mapper

Modulation is a process in which the incoming data stream is modulated onto a carrier, the modulation process involves switching the amplitude, frequency or phase of sinusoidal carrier in some fashion in accordance with the incoming data; there are three basic modulation schemes known as ASK, FSK and PSK.

2.2.3. FFT

FFT is a block added to generate SC-FDMA symbols instead of OFDM symbols which have high peak to average power ratio (PAPR) which require highly linear power amplifiers to avoid excessive intermodulation distortion, while SC-FDMA solves this problem by distributing given number of modulated symbols over all assigned subcarriers for transmission instead of distributing one symbol over one subcarrier as in OFDM, therefore the total PAPR decreased due to this fair distribution.

Uplink uses SC-FDMA which is a modified form of the OFDM with similar throughput performance and complexity, SC-FDMA is viewed as DFT-coded OFDM where time-domain symbols are transformed to frequency domain symbols and then go through the standard OFDM modulation as shown in Figure 2.2.3-1.

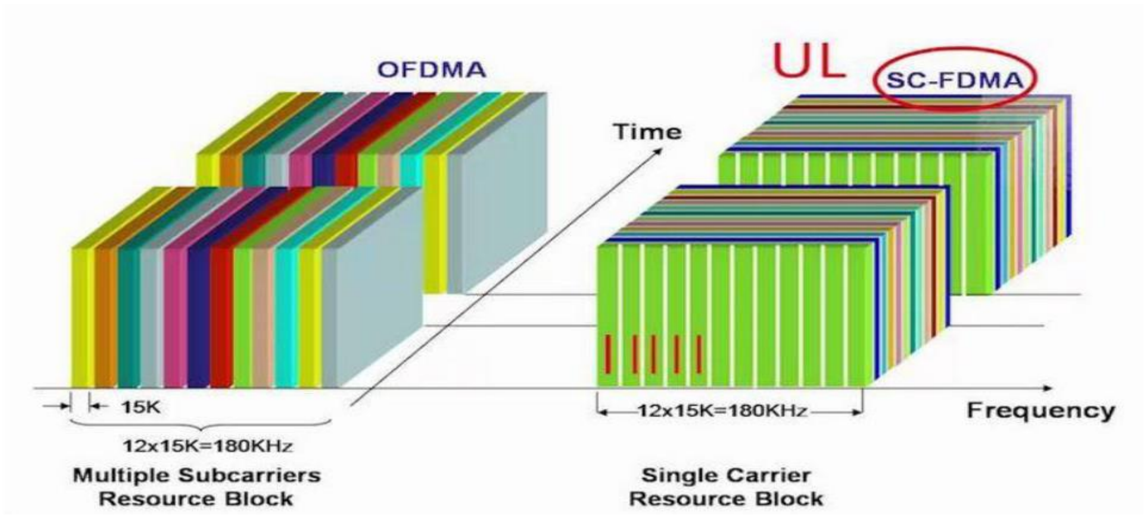


Figure 2.2.3-1: OFDMA vs SC-FDMA

SC-FDMA has all the advantages of OFDM like robustness against multi-path signal propagation, the block diagram for the SC-FDMA is shown in Figure 2.2.3-2.

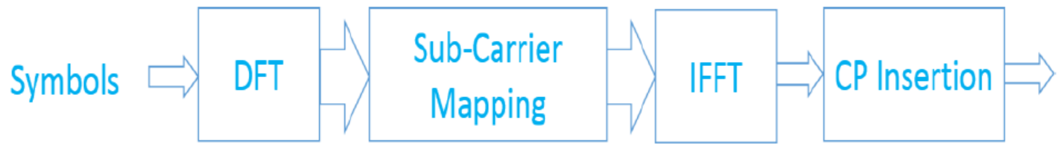


Figure 2.2.3-2: SC-FDMA block diagram

2.2.4. Resource element Mapper

Resource element mapper's function is to take the output symbols of the FFT and assign them to the proper subcarriers allocated for the NB-IoT bandwidth and this is was provided as information from the upper layer assigned number of subcarriers can be 1, 3, 6 or 12 subcarriers, in the case of 12 subcarriers; all the symbols are assigned to the 12 subcarriers. In the case of 6 point FFT, the output 6 symbols of the FFT will be assigned to a certain 6 subcarriers of the 12 assigned subcarriers and the rest are padded with zeros and the same will be done with 3 point FFT results, The location of the symbols within the 12 subcarriers in case of 3 and 6 subcarriers is calculated according to information from upper layer.

2.2.5. IFFT

IFFT is the block responsible for transforming frequency sub channels into time domain samples to be transmitted through the RF blocks, one challenging issue of the IFFT block at transmitter of SC-FDMA is being at the end the chain that's why it must satisfy the required output rate (i.e new SC-FDMA symbol each SC-FDMA symbol duration) not only produce this symbol in this duration but also with a constant rate without any bubbles.

Chapter 3

Standard specifications and Assumptions

3. Standard Specifications

In this Chapter, the standard description for each block in the chain will be stated and if any assumption is taken or optimization in the standard equations according to our thesis scope "NB-IOT".

3.1. Channel coding, multiplexing and interleaving

3.1.1. Cyclic Redundancy Check (CRC)

Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3,...,a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3,..., p_{L-1}$. A is equal to TBS and L is the number of parity bits at which L=24 bits for NB-IOT. The parity bits for NB-IOT are generated the following cyclic generator polynomial:

$$g_{CRC24A}(D)=[D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1] \quad (1)$$

The bits after CRC attachment are denoted by, where B = A+ L. The relation between a_k and b_k is:

$$b_k = a_k \qquad (2) \qquad \text{For } k = 0, 1, 2,\ldots, A\text{-}1$$

$$b_k = p_{k-A} \qquad (3) \qquad \text{For } k = A, A+1, A+2,..., A+L-1.$$

There is no segmentation as for NB-IOT maximum TBS=2536 bits it's less than Z=6144 bits so there is only one code block C=1.

3.1.2. Turbo Encoder

According to 3GPP narrowband LTE IOT Standard release 14, Turbo Encoder is consists of two recursive Convolutional Encoder and one Turbo internal interleaver as shown in the following figure 3.1.2-1.

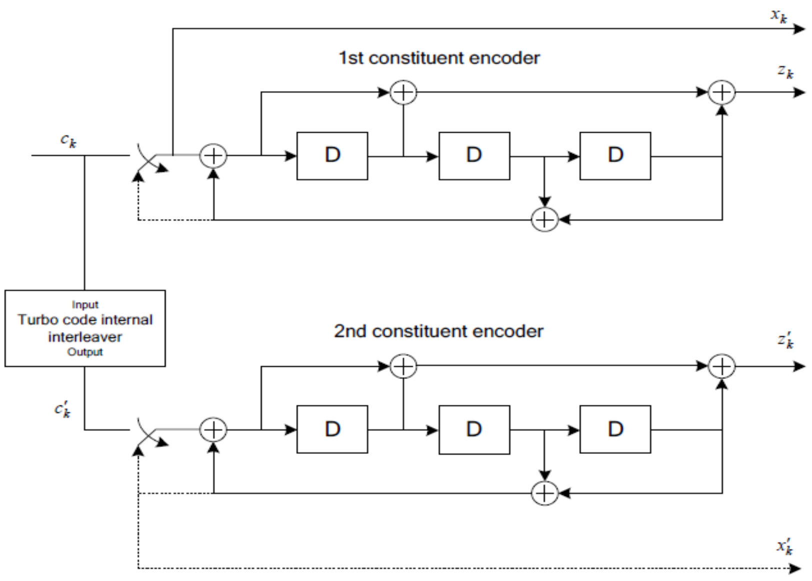


Figure 3.1.2-1: internal Structure of NB LTE Turbo Encoder of rate 1/3

3.1.2.1. Recursive Convolutional Encoder

The scheme of recursive Convolutional encoder is a Parallel Concatenated Convolutional Code (PCCC) with two 8-state constituent encoders, the coding rate of turbo encoder is 1/3, the recursive convolutional encoders are differ from ordinary convolutional encoder as it has both feedback and feedforward convolutional encoders as in figure 3.1.2.1-1.

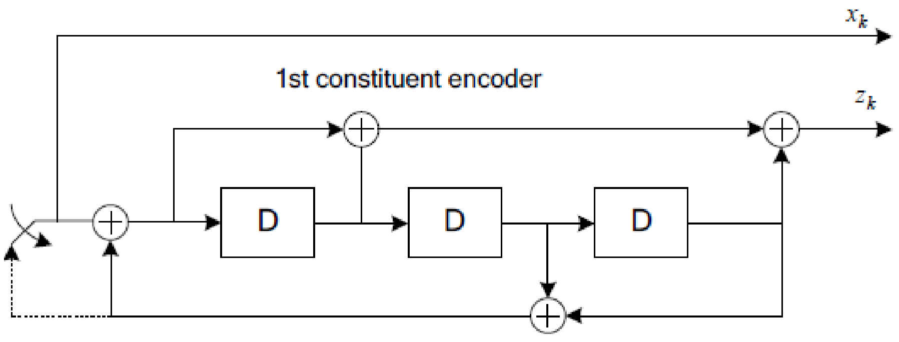


Figure 3.1.2.1-1: Recursive Convolutional Encoder

The transfer function of the 8-state constituent code for the PCCC is:

$$G(D) = \left[1, \frac{g_1(D)}{g_0(D)}\right] \qquad (4)$$

Where $g_0(D) = 1 + D^2 + D^3$, (5)

$g_1(D) = 1 + D + D^3$ (6)

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

The output from the turbo encoder is:

$$d_k^{(0)} = x_k \qquad (7)$$
$$d_k^{(1)} = z_k \qquad (8)$$
$$d_k^{(2)} = z'_k \qquad (9)$$

For k= 0, 1, 2, .., k-1.

The bits input to the turbo encoder are denoted by $c_0, c_{1,}\ c_{2,}\ c_{K-1}$, and the bits output from the first and second 8-state constituent encoders are denoted by $z_0, z_{1,}\ z_{2,}\ z_{K-1}$, and $z'_0, z'_1\ , \ldots, z'_{K-1}$ respectively. The bits output from the turbo code internal interleaver are denoted by $c'_0, c'_{1,}\ c'_{2,}\ c'_{K-1}$, and these bits are to be the input to the second 8-state constituent encoder.

3.1.2.2. Trellis termination of turbo encoder

Trellis termination is performed by taking the tail bits from the shift register feedback after all information bits are encoded. Tail bits are padded after the encoding of information bits.

The first three tail bits shall be used to terminate the first constituent encoder (upper switch of figure 3.1.2-1 in lower position) while the second constituent encoder is disabled. The last three tail bits shall be used to terminate the second constituent encoder (lower switch of figure 3.1.2-1 in lower position) while the first constituent encoder is disabled.

The transmitted bits for trellis termination shall then be:

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

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

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

3.1.2.3. Internal interleaver

The bits input to the turbo code internal interleaver are denoted by $C_0\ , C_1\ , C_2\ , \ldots C_{k-1}$, where K is the number of input bits = TBS + 24 CRC bits. The bits output from the turbo code internal interleaver are denoted by $C'_0\ , C'_1\ , C'_2\ , \ldots C'_{k-1}$.

The relationship between the input and output bits is as follows:

$$C'_i = C_{\pi(i)}\ , Where\ i = 0,1,2, \ldots (k-1) \qquad (10)$$

Where the relationship between the output index *i (ordered bits index)* and the input index $\pi(i) - interleaved\ index -$ satisfies the following quadratic form:

$$\pi(i) = (f_1 * i \ + f_2 * i^2) \ \% \ K \qquad (11)$$

The parameters f_1 and f_2 depend on the block size K and are summarized in Table 3.1.2.3-1 in the standard:

Table 3.1.2.3-1: Turbo code internal interleaver parameters

M	***K***	f_1	f_2	***M***	***K***	f_1	f_2	***M***	***K***	f_1	f_2
1	40	3	10	48	416	25	52	95	1120	67	140
2	48	7	12	49	424	51	106	96	1152	35	72
3	56	19	42	50	432	47	72	97	1184	19	74
4	64	7	16	51	440	91	110	98	1216	39	76
5	72	7	18	52	448	29	168	99	1248	19	78
6	80	11	20	53	456	29	114	100	1280	199	240
7	88	5	22	54	464	247	58	101	1312	21	82
8	96	11	24	55	472	29	118	102	1344	211	252
9	104	7	26	56	480	89	180	103	1376	21	86
10	112	41	84	57	488	91	122	104	1408	43	88
11	120	103	90	58	496	157	62	105	1440	149	60
12	128	15	32	59	504	55	84	106	1472	45	92
13	136	9	34	60	512	31	64	107	1504	49	846
14	144	17	108	61	528	17	66	108	1536	71	48
15	152	9	38	62	544	35	68	109	1568	13	28
16	160	21	120	63	560	227	420	110	1600	17	80
17	168	101	84	64	576	65	96	111	1632	25	102
18	176	21	44	65	592	19	74	112	1664	183	104
19	184	57	46	66	608	37	76	113	1696	55	954
20	192	23	48	67	624	41	234	114	1728	127	96
21	200	13	50	68	640	39	80	115	1760	27	110
22	208	27	52	69	656	185	82	116	1792	29	112
23	216	11	36	70	672	43	252	117	1824	29	114
24	224	27	56	71	688	21	86	118	1856	57	116
25	232	85	58	72	704	155	44	119	1888	45	354
26	240	29	60	73	720	79	120	120	1920	31	120
27	248	33	62	74	736	139	92	121	1952	59	610
28	256	15	32	75	752	23	94	122	1984	185	124
29	264	17	198	76	768	217	48	123	2016	113	420
30	272	33	68	77	784	25	98	124	2048	31	64
31	280	103	210	78	800	17	80	125	2112	17	66
32	288	19	36	79	816	127	102	126	2176	171	136
33	296	19	74	80	832	25	52	127	2240	209	420
34	304	37	76	81	848	239	106	128	2304	253	216
35	312	19	78	82	864	17	48	129	2368	367	444
36	320	21	120	83	880	137	110	130	2432	265	456
37	328	21	82	84	896	215	112	131	2496	181	468
38	336	115	84	85	912	29	114	132	2560	39	80

3.1.3. Rate Matching

According to standard rate matching mainly consists of three main parts:

1. First part is sub-block interleavers which are used to interleave the three information bit streams $d_k^{(0)}$, $d_k^{(1)}$ and $d_k^{(2)}$ coming from turbo-encoder as shown in figure 3.1.3-1.
2. Second part is bit collection block which concatenates the three output streams of the three sub-block interleavers $v_{k_\pi}^{(0)}, v_{k_\pi}^{(1)}, v_{k_\pi}^{(2)}$ which represent the systematic bit stream, parity bit stream and interleaved parity stream respectively as in figure 3.1.3-1.
3. Last sub-block is bit-selection block which select the start point inside buffer to get output according to upper layer parameters described in following sections.

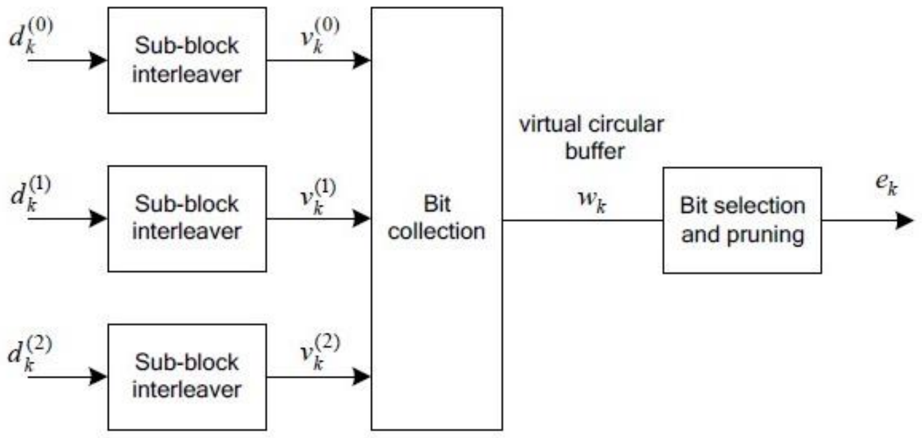


Figure 3.1.3-1: Standard block diagram for rate matching

3.1.3.1. Sub-block interleavers

The input bits to the sub-block interleaver are denoted by $d_0^{(i)}, d_1^{(i)}, d_2^{(i)}, \ldots, d_{D-1}^{(i)}$, where D is the number of bits $D = TBS + 24 + 4$ where the 24 bits are CRC bits and the other 4 bits are the trellis termination bits added by the encoder and i =0,1,2.The interleaving is done after putting the data in matrix form according to the following steps :

- Assign $C_{subblock}^{TC} = 32$ to be the number of columns of the matrix. The columns of the matrix are numbered 0, 1, 2,…, $C_{subblock}^{TC} - 1$ from left to right.
- Determine the number of rows of the matrix $R_{subblock}^{TC}$ at which the rows of rectangular matrix are numbered 0, 1, 2,…, $R_{subblock}^{TC} - 1$ from top to bottom. Number of rows can be found by calculating minimum integer $R_{subblock}^{TC}$ such that:

$$D \le \left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right) \qquad (12)$$

- The next step is to determine the number of dummy bits to complete the rectangular matrix according to next scenario :

    a. If $\left(R^{TC}_{subblock} \times C^{TC}_{subblock}\right) > D$, then $N_D = \left(R^{TC}_{subblock} \times C^{TC}_{subblock} - D\right)$ where N_D is the number of dummy bits.

    b. Dummy bits are padded at the beginning of the matrix such that y_k = <*NULL*> for $k = 0, 1, \ldots, N_D - 1$.

- Then, the input data bits entered and placed after dummy bits as follow $y_{N_D+k} = d_k^{(i)}$, $k = 0, 1, \ldots, D\text{-}1$, and the bit sequence y_k is written into the $\left(R^{TC}_{subblock} \times C^{TC}_{subblock}\right)$ matrix row by row starting with bit y_0 in column 0 of row 0 as in following matrix.

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

(13)

- After the write state ends reading state starts from the matrix according to permutation table which is the same table for $d_k^{(0)}$ and $d_k^{(1)}$ but for $d_k^{(2)}$ it's different.

    a. For $d_k^{(0)}$ and $d_k^{(1)}$:

Perform the inter-column permutation for the matrix based on the pattern $\langle p(j)\rangle_{j=\{0,1,\ldots,C^{TC}_{subblock}-1\}}$ that is shown in table 3.1.3.1-1, where P (j) is the original column position of the j-th permuted column. After permutation of the columns, the inter-column permuted $\left(R^{TC}_{subblock} \times C^{TC}_{subblock}\right)$ matrix is as in following matrix, while the output of the sub-block interleaver is the bit sequence read out column by column from the inter-column permuted $\left(R^{TC}_{subblock} \times C^{TC}_{subblock}\right)$matrix. The bits after sub-block interleaving are denoted by $v_0^{(i)}, v_1^{(i)}, v_2^{(i)}, \ldots, v_{K_\Pi-1}^{(i)}$, where $v_0^{(i)}$ corresponds to $y_{P(0)}$, $v_1^{(i)}$ to $y_{p(0)+C^{TC}_{subblock}}$ … and $K_\Pi = \left(R^{TC}_{subblock} \times C^{TC}_{subblock}\right)$.

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

(14)

Table 3.1.3.1-1: Inter-column permutation pattern for sub-block interleaver

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

- For $d_k^{(2)}$:

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

$$\pi(k) = \left(P\left(\left\lfloor \frac{k}{R_{subblock}^{TC}} \right\rfloor \right) + C_{subblock}^{TC} \times \left(k \bmod R_{subblock}^{TC} \right) + 1 \right) \bmod K_\Pi \qquad (15)$$

One of the simplifications reached after tracing the equation of inter-column permutation for the third stream columns that on substituting by k for each $R_{subblock}^{TC}$ the result that can be reached is that the table 3.1.3.1-1 can be used with adding one to each column index to be as in table 3.1.3.1-2.

Table 3.1.3.1-2: Inter-column permutation pattern for third sub-block interleaver

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

3.1.3.2. <u>Bit collection</u>

This block concatenate the three streams in circular buffer at first it adds the original interleaved stream which are known as systematic bits then interlacing between two other parity streams, the circular buffer of length $K_w = 3K_\Pi$ for the coded block is generated as following :

$$w_k = v_k^{(0)} \qquad \text{for } k = 0, \ldots, K_\Pi - 1$$

$$w_{K_\Pi + 2k} = v_k^{(1)} \qquad \text{for } k = 0, \ldots, K_\Pi - 1$$

$$w_{K_\Pi + 2k + 1} = v_k^{(2)} \qquad \text{for } k = 0, \ldots, K_\Pi - 1$$

3.1.3.3. <u>Bit selection</u>

Denote the soft buffer size for code block (only one code block for NB-IOT) by N_{cb} bits, where the size N_{cb} is obtained as follows:

$$N_{cb} = K_w, \qquad \text{Where } K_w = 3K_\Pi$$

Denoting by E the rate matching output sequence length for the coded block and the redundancy version number which used to determine the starting transmission point in the buffer for this transmission, where rvidx = 0 or 2 for NB-IOT and the rate matching output bit sequence indexes are $\mathrm{k} = 0,1,...,E-1$.

To determine the output length, define G which represents the total number of bits available for the transmission of one transport block, then set $G' = G/(N_L \cdot Q_m)$, where modulation index Q_m is 1 for BPSK and 2 for QPSK, while N_L is equal to the number of layers a transport block is mapped onto which is equal to 1 layer in NB-IOT, therefore $G' = G/(Q_m)$, then the output length can be determined as $E = Q_m \cdot \lceil G' \rceil$.

The last stage is to choose point to start from inside the buffer so define k_0 at which

$$k_0 = R_{subblock}^{TC} \cdot \left(2 \cdot \left\lceil \frac{N_{cb}}{8R_{subblock}^{TC}} \right\rceil \cdot rv_{idx} + 2 \right) \qquad (16),$$

Where $R_{subblock}^{TC}$ is the number of rows defined in previous section. However, this equation can be simplified in NB-IOT to be $k_0 = R_{subblock}^{TC} \cdot (24 \cdot rv_{idx} + 2)$, as
$N_{cb} = K_w = 3K_\Pi = 3\left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right) = 3\left(R_{subblock}^{TC} \times 32\right)$

Then output can be determined according the following pseudo code where k is bit index counter while j is loop counter to avoid reading NULL dummy bits and the modulus here just to represent the idea of circular buffer when $k_0 + j = N_{cb}$ it returns to index zero inside the buffer.

```
Set $k = 0$ and $j = 0$
While {k < $E$}
    If $w_{(k_0+j) mod\ N_{cb}} \neq < NULL >$
        $e_k = w_{(k_0+j) mod\ N_{cb}}$
        $k = k + 1$
    End if
    $j = j + 1$
End while
```

3.1.4. Data Multiplexing and Channel interleaver

After rate matching, interleaving is applied per resource unit without any control information multiplexing in order to apply a time-first rather than frequency-first mapping, where the input sequence is the portion of e for a resource unit (output of the Rate matching) and where maximum number of columns C_{max} is:

$$C_{max} = (N_{symb}^{UL} - 1) * N_{slot}^{UL} \qquad (17)$$

Where N_{symb}^{UL} is the number of SC-FDMA symbols for NPUSCH in a UL resource unit, $\mathrm{N}_{\mathrm{slot}}^{\mathrm{UL}}$ is number of subcarriers in the frequency domain for NB-IoT and $\mathrm{N}_{\mathrm{sc}}^{\mathrm{RU}}$ is number of consecutive subcarriers in an UL resource unit for NB-IoT as given in Table 1.7.3-1.

3.1.4.1. Data and Control Multiplexing

The control and data multiplexing is performed such that HARQ-ACK information is present on both slots and is mapped to resources around the demodulation reference signals. In addition, the multiplexing ensures that control and data information are mapped to different modulation symbols.

The inputs to the data and control multiplexing are the coded bits of the control information denoted by $q_0, q_1, q_2, \ldots. q_{N_L * Q_{CQI}-1}$ and the coded bits of the UL-SCH denoted by $f_0, f_1, f_2, \ldots\ldots, f_{G-1}$ The output of the data and control multiplexing operation is denoted by $g_0, g_1, g_2, \ldots. g_{H\prime-1}$

$$\text{Where } H = G + N_L * Q_{CQI} \qquad (18)$$

$$\text{And } H' = H / N_L * Q_m \qquad (19)$$

H is the total number of coded bits allocated for UL-SCH data and CQI/PMI information across the N_L transmission layers of the transport block.

In Narrowband IoT-LTE (Assumptions):

- $Q_m = 1\ for\ BPSK\ and\ 2\ for\ QPSK$
- $N_L = 1$
- $Q_{CQI} = 0$, As no control bits is multiplexed

f0, f1, f2, .. ,
Code bits
Multiplexer
g0, g1, g2, ...
Multiplexed output

Figure 3.1.4.1-1: Multiplexer block diagram

```
i = k = 0
While i < G -- then place the data
    $g_k = [f_i \dots . f_{i+Q_m*N_L-1}]^T$
    $i = i + Q_m * N_L$
    k=k+1
end while
```

Then for BPSK:

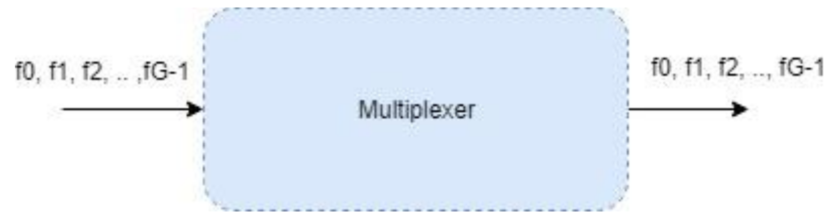


Figure 3.1.4.1-2: Multiplexing BPSK stream

For QPSK:

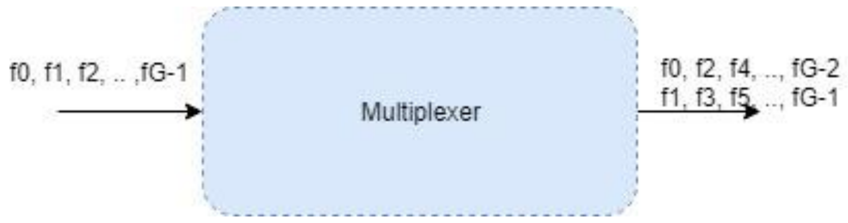


Figure 3.1.4.1-3: Multiplexing QPSK stream

3.1.4.2. <u>Channel Interleaver</u>

The channel interleaver described in this section in conjunction with the resource element mapping implements a time-first mapping of modulation symbols onto the transmit waveform while ensuring that the HARQ-ACK and RI information are present on both slots in the subframe. But in our project we take into consideration No control information, so code bits $g_0, g_1, g_2, \dots . g_{H'-1}$ are only interleaved in channel interleaver.

The output bit sequence from the channel interleaver is derived as follows:

1. Assign $C_{max}$by equation (17) to be the number of columns of the matrix. The columns of the matrix are numbered 0, 1, 2, … , $C_{max} - 1$ from left to right.
2. The number of rows of the matrix is $R_{max} = (H'_{total} . Q_m . N_L)/C_{max}$ and we define $R'_{\max} = R_{max}/(Q_m . N_L)$. The rows of the rectangular matrix are numbered 0, 1, 2, … , $R_{max} - 1$ from top to bottom.

3. Write the input vector sequence, for k = 0, 1,…, $H'-1$, into the ($R_{max}\ x\ C_{max}$) matrix by sets of ($Q_m.N_L$) rows starting with the vector y_0 in column 0 and rows 0 to ($Q_m.N_L$) -1. We filled ($R_{max}\ x\ C_{max} -$ H') with zeros in the last $Q_m.N_L$ rows. (Assumption)
   Where $y_0 = f_0\ for\ BPSK\ that\ occupies\ (Q_m.N_L = 1)\ rows$ and $y_0 = \begin{bmatrix} f_0 \\ f_1 \end{bmatrix}\ for\ QPSK\ that\ occupies\ (Q_m.N_L = 2)\ rows.$

$$\begin{bmatrix} \underline{y}_0 & \underline{y}_1 & \underline{y}_2 & \cdots & \underline{y}_{C_{mux}-1} \\ \underline{y}_{C_{mux}} & \underline{y}_{C_{mux}+1} & \underline{y}_{C_{mux}+2} & \cdots & \underline{y}_{2C_{mux}-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \underline{y}_{(R'_{mux}-1)\times C_{mux}} & \underline{y}_{(R'_{mux}-1)\times C_{mux}+1} & \underline{y}_{(R'_{mux}-1)\times C_{mux}+2} & \cdots & \underline{y}_{(R'_{mux}\times C_{mux}-1)} \end{bmatrix}$$

The output of the block interleaver is the bit sequence read out column by column from the ($R_{max}\ x\ C_{max}$) matrix. The bits after channel interleaving are denoted by $h_0, h_1, h_2 \dots h_{R_{max}\ x\ C_{max}-1}$.

3.2. Physical Channel and modulation

3.2.1. Scrambler

Scrambler mainly consists of two linear feedback shift registers which simply generating a L=31- Golden Sequence C(n) by 2 paths of flip flops which initialized by two different values as shown in figure 3.2.1-1, for input codeword, the block of bits $b^{(q)}(0),...,b^{(q)}(M_{\text{bit}}^{(q)}-1)$, where $M_{\text{bit}}^{(q)}$ is the number of bits transmitted in codeword on the physical uplink shared channel in one sub-frame, shall be scrambled with a UE-specific scrambling sequence prior to modulation, resulting in a block of scrambled bits $\tilde{b}^{(q)}(0),...,\tilde{b}^{(q)}(M_{\text{bit}}^{(q)}-1)$.

In order to get the required output, the first m-sequence shall be initialized with $x_1(0)=1, x_1(n)=0, n=1,2,...,30$. On the other hand the initialization of the second m-sequence is denoted by $c_{\text{init}}=\sum_{i=0}^{30} x_2(i)\cdot 2^i$ and c_{init} for NB-IOT is given by

$$c_{\text{init}} = n_{\text{RNTI}}\cdot 2^{14} + n_f \bmod 2\cdot 2^{13} + \lfloor n_s/2\rfloor\cdot 2^9 + N_{\text{ID}}^{\text{Ncell}} \qquad (20)$$

This equation calculates the initial decimal value for x_2 which can be converted to its binary value to initialize the register. To work properly, initial 1600 shift cycles should be taken in consideration to induce extra randomness for the initial state, so $N_C = 1600$ and the output sequence is given by:

$$\begin{aligned} c(n) &= (x_1(n+N_C)+x_2(n+N_C)) \bmod 2 \\ x_1(n+31) &= (x_1(n+3)+x_1(n)) \bmod 2 \\ x_2(n+31) &= (x_2(n+3)+x_2(n+2)+x_2(n+1)+x_2(n)) \bmod 2 \end{aligned} \qquad (21)$$

, so 1600 shift cycles should be performed directly after initialization and before enabling scrambler to be able to receive any input.

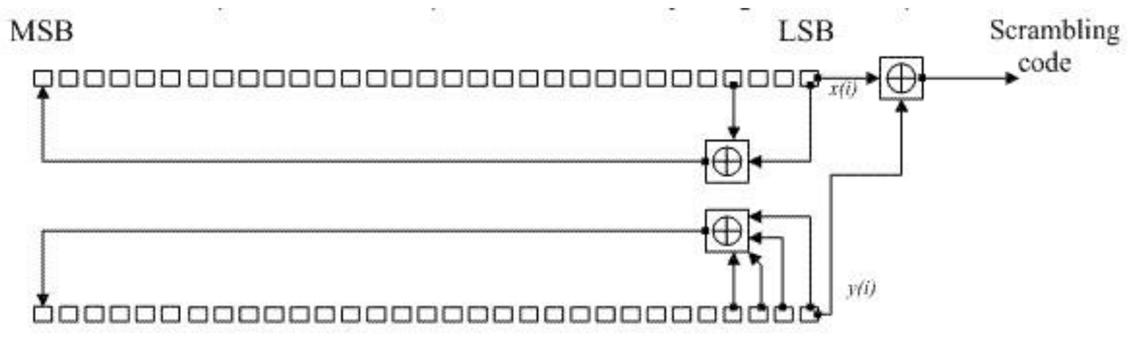


Figure 3.2.1-1: Scrambler standard internal structure

3.2.2. Modulation Mapper

According to 3GPP Narrowband LTE Standard Release 14; Table 3.2.2-1 specifies the modulation mapping applicable for the narrowband physical uplink shared channel

Table 3.2.2-1: BPSK Modulation Mapping

NPUSCH format	N_{sc}^{RU}	Modulation scheme
1	1	BPSK, QPSK
	>1	QPSK
2	1	BPSK

- **BPSK**

In case of BPSK modulation, a single bit b(i), is mapped to a complex-valued modulation symbol x =I + j Q according to Table 3.2.2-2 and figure 3.2.2-1.

Table 3.2.2-2: BPSK Modulation Mapping

b(i)	I	Q
0	$1/\sqrt{2}$	$1/\sqrt{2}$
1	$-1/\sqrt{2}$	$-1/\sqrt{2}$

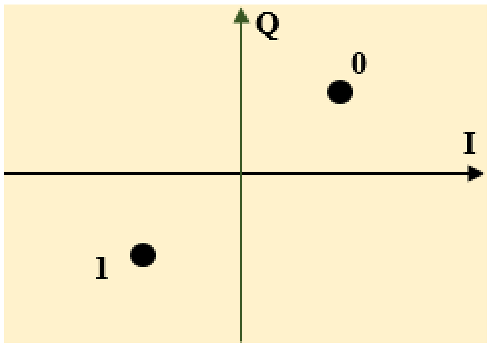


Figure 3.2.2-1: BPSK Constellation

- **QPSK**

In case of QPSK modulation, pairs of bits b(i),b(i+1) , are mapped to complex-valued modulation symbols x where x= I + jQ according to Table 3.2.2-3 and figure 3.2.2-2.

Table 3.2.2-3: QPSK Modulation Mapping

b(i),b(i+1)	I	Q
00	$1/\sqrt{2}$	$1/\sqrt{2}$
01	$1/\sqrt{2}$	$-1/\sqrt{2}$
10	$-1/\sqrt{2}$	$1/\sqrt{2}$
11	$-1/\sqrt{2}$	$-1/\sqrt{2}$

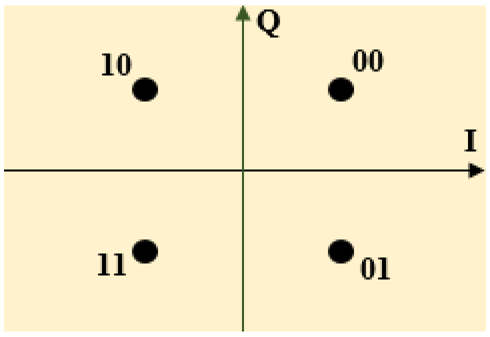


Figure 3.2.2-2: QPSK Constellation

3.2.3. FFT

The block of complex values x (0), x (1) ………. x($M_{symbol}^{Layer} - 1$) , generated from the modulation mapper, is divided into a set of $M_{symbol}^{Layer}/M_{sc}^{Npusch}$ sets, each corresponding to one SC-FDMA symbol. Transform precoding shall be applied according to

$$y\left(l.M_{sc}^{Npusch} + k\right) = \frac{1}{\sqrt{M_{sc}^{Npusch}}} \sum_{k=0}^{M_{sc}^{Npusch}-1} x(l.M_{sc}^{pusch} + i)e^{-j\frac{2\pi ik}{M_{sc}^{Npusch}}} \quad (22)$$

$$k = 0,1, \dots \dots\ M_{sc}^{Npusch} - 1$$

$$l = 0,1, \dots \dots . \frac{M_{symbol}^{Layer}}{M_{sc}^{Npusch}} - 1$$

Resulting in a block of complex-valued symbols y (0), y (1)….. y ($M_{symbol}^{Layer} - 1$). Where$M_{sc}^{Npusch} = N_{sc}^{RU}$. $N_{sc}^{RU}$is the number of allocated subcarriers and it could be determined for NB-IoT from table 1.7.3-1.

Resource units are used to describe the mapping of the NPUSCH to resource elements. A resource unit is defined as $N_{slots}^{UL} * N_{symbols}^{UL}$ consecutive SC-FDMA symbols in the time domain and N_{sc}^{RU}, consecutive subcarriers in the frequency domain, where N_{slots}^{UL}, $N_{symbols}^{UL}$and N_{sc}^{RU}, are given by table 1.7.3-1. In our design we assumed working with NPUSCH format 1 with 15 KHz spacing as mentioned before in chapter 1.

3.2.4. Resource element Mapper

3.2.4.1. Resource grid

A transmitted physical channel or signal in a slot is described by one or several resource grids of $N_{\mathrm{sc}}^{\mathrm{UL}}$ subcarriers and $N_{\mathrm{symb}}^{\mathrm{UL}}$ SC-FDMA symbols. The resource grid is illustrated in figure 3.2.4.1-1. The slot number within a radio frame is denoted n_{s} where $n_{\mathrm{s}} \in \{0,1,\ldots,19\}$ for $\Delta f = 15\,\mathrm{kHz}$ and $n_{\mathrm{s}} \in \{0,1,\ldots,4\}$ for $\Delta f = 3.75\,\mathrm{kHz}$.

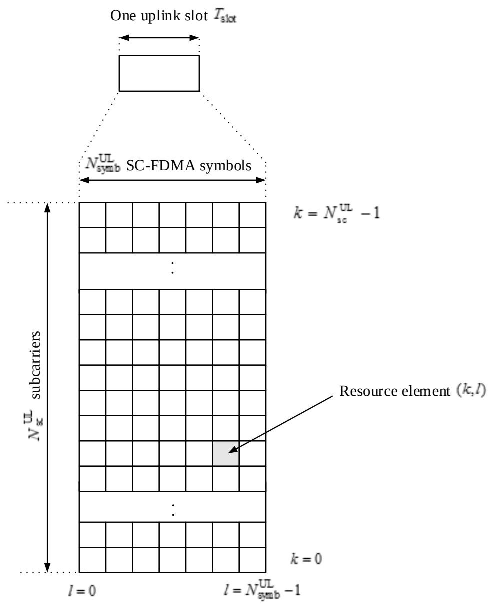

One uplink slot T_{slot}
$N_{\mathrm{symb}}^{\mathrm{UL}}$ SC-FDMA symbols
$k = N_{\mathrm{sc}}^{\mathrm{UL}} - 1$
$N_{\mathrm{sc}}^{\mathrm{UL}}$ subcarriers
Resource element (k,l)
$k = 0$
$l = 0$
$l = N_{\mathrm{symb}}^{\mathrm{UL}} - 1$

Figure 3.2.4.1-1: Uplink resource grid for NB-IoT

The uplink bandwidth in terms of subcarriers $N_{\mathrm{sc}}^{\mathrm{UL}}$, and the slot duration T_{slot} are given in table 1.7.1-1.

3.2.4.2. Resource element

Each element in the resource grid is called a resource element and is uniquely defined by the index pair (k,l) in a slot where $k = 0,\ldots,N_{\mathrm{sc}}^{\mathrm{UL}} - 1$ and $l = 0,\ldots,N_{\mathrm{symb}}^{\mathrm{UL}} - 1$ are the indices in the frequency and time domains, respectively. Resource element (k,l) corresponds to the complex value $a_{k,l}$. Quantities $a_{k,l}$ corresponding to resource elements not used for transmission of a physical channel or a physical signal in a slot shall be set to zero.

3.2.4.3. Resource unit

Resource units are used to describe the mapping of the NPUSCH to resource elements. A resource unit is defined as $N_{\text{symb}}^{\text{UL}} N_{\text{slots}}^{\text{UL}}$ consecutive SC-FDMA symbols in the time domain and $N_{\text{sc}}^{\text{RU}}$ consecutive subcarriers in the frequency domain, where $N_{\text{sc}}^{\text{RU}}$ and $N_{\text{symb}}^{\text{UL}}$ are given by table 1.7.3-1.

3.2.4.4. Mapping to physical resources

NPUSCH can be mapped to one or more than one resource units, N_{RU} each of which shall be transmitted $M_{\text{rep}}^{\text{NPUSCH}}$ times.

The block of complex-valued symbols $z(0),...,z(M_{\text{symb}}^{\text{ap}}-1)$ which are the output of the FFT block are mapped in sequence starting with $z(0)$ to subcarriers assigned for transmission of NPUSCH. The mapping to resource elements (k,l) corresponding to the subcarriers assigned for transmission and not used for transmission of reference signals shall be in increasing order of first the index k, then the index l, starting with the first slot in the assigned resource unit.

After mapping to N_{slots} slots, the N_{slots} slots shall be repeated $M_{\text{identical}}^{\text{NPUSCH}}-1$ additional times, before continuing the mapping of $z(\cdot)$ to the following slot, where

$$M_{\text{identical}}^{\text{NPUSCH}} = \begin{cases} \min\left(\left\lceil M_{\text{rep}}^{\text{NPUSCH}}/2 \right\rceil, 4\right) & N_{\text{sc}}^{\text{RU}} > 1 \\ 1 & N_{\text{sc}}^{\text{RU}} = 1 \end{cases} \qquad (23)$$

$$N_{\text{slots}} = \begin{cases} 1 & \Delta f = 3.75\,\text{kHz} \\ 2 & \Delta f = 15\,\text{kHz} \end{cases} \qquad (24)$$

The resource allocation information in uplink DCI format N0 for NPUSCH transmission indicates to a scheduled UE

- A set of contiguously allocated subcarriers (n_{sc}) of a resource unit determined by the Subcarrier indication field in the corresponding DCI,
- A number of resource units (N_{RU}) determined by the resource assignment field in the corresponding DCI according to table 3.2.4.4-1.
- A repetition number (N_{Rep}) determined by the repetition number field in the corresponding DCI according to table 3.2.4.4-2.

The subcarrier spacing Δf of NPUSCH transmission is assumed 15 KHz for our operation.

For NPUSCH transmission with subcarrier spacing $\Delta f = 15\,\text{kHz}$, the subcarrier indication field (I_{sc}) in the DCI determines the set of contiguously allocated subcarriers (n_{sc}) according to table 1.7.4-2.

I_{RU}	N_{RU}
0	1
1	2
2	3
3	4
4	5
5	6
6	8
7	10

Table 3.2.4.4-1 Number of resource units (N_{RU}) for NPUSCH

I_{Rep}	N_{Rep}
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Table 3.2.4.4-2 Number of repetitions (N_{Rep}) for NPUSCH.

3.2.5. IFFT and CP

IFFT is the block responsible for transforming frequency sub channels into time domain samples to be transmitted through the RF blocks. LTE uplink uses SC-FDMA which is a modified form of the OFDM with similar throughput performance and complexity, SC-FDMA is viewed as DFT-coded OFDM where time-domain symbols are transformed to frequency domain symbols and then go through the standard OFDM modulation, SC-FDMA has all the advantages of OFDM like robustness against multi-path signal propagation, the block diagram for the SC-FDMA is shown in 3.2.5-1.

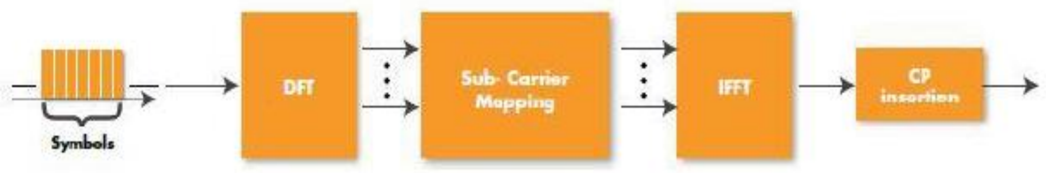


Figure 3.2.5-1: SCFDMA Block diagram

The main advantage of SC-FDMA is the low Peak Average Power Ratio (PAPR) of the transmit signal, PAPR is a big concern for user equipment, as PAPR relates to the power amplifier efficiency as low PAPR allows the power amplifier to operate close to the saturation region resulting in high efficiency that is why SC-FDMA is the preferred technology for user terminals.

The NB-IoT supported bandwidth is 180 KHz with number of subcarriers depending on the spacing between sub channels, 2 sub channels are defined in the standard 15 KHz and 3.75 KHz each with its own symbol duration. For more clear understanding of the symbol durations and frequency allocation of symbols we need to illustrate this on the resource grid, the resource grid represents time on the x-axis and frequency on the y-axis. In its time domain structure, each LTE frame is 10ms long and comprises of 10 sub-frames, each of 1ms duration. The sub-frames are further divided into two time slots of 0.5ms, each comprising 7 SC-FDMA symbols in normal cyclic prefix operation and 6 SC-FDMA symbols in extended cyclic prefix mode. For the sake of simplicity, we shall restrict our discussion only to normal cyclic prefix mode. A fixed subcarrier spacing of 15 KHz gives a SC-FDMA symbol duration of 66.67 us. A cyclic prefix of duration 5.2 us is used for the first SC-FDMA symbol and 4.68 us for the rest of the symbols in each time-slot. This gives a total symbol duration of 71.87 us for the first symbol and 71.35 us for the rest of the symbols in a time-slot.

The IFFT subcarriers are grouped into sets of 12 subcarriers with spacing of 15 KHz between each adjacent sub channels, each group is called a resource block NB-IoT has one resource block only per UE transmitter. To implement IFFT block which supports 12 subcarriers as the max number of allocated sub carriers we may use 16 point IFFT as a minimum size. The less the IFFT size, the less the output sample rate is as this number of samples in addition to cyclic prefix samples are distributed on the same defined SC-FDMA symbol duration. In our implementation we use 16 point IFFT to decrease latency and power dissipation of the block and output sample rate could be compensated by an up sample filter to upgrade the rate to a convenient rate for the digital to analog converter (ADC) block.

These settings are equivalent to cyclic prefix samples per Sc-fdma symbol which is 16 samples =2 cyclic prefix samples, peak data rate = 336 Kbit/s and output sample rate =36 Ksample/sec.

Chapter 4

Design Block diagram and interfaces

4. Transmitter Uplink Chain Design, operation and block interfaces

After discussing the Standard specification for each block in Chapter 3 and the Function of each one in Chapter 2, the Design implementation Idea and Block diagram will be introduced in this chapter for each individual block.

4.1. Channel coding, multiplexing and interleaving

4.1.1. Cyclic Redundancy Check (CRC)

4.1.1.1. Structure and Block interfaces

In RTL, CRC is implemented as shift registers with XOR gates depending on the polynomial introduced in the standard chapter as shown in figure 4.1.1.1-1.

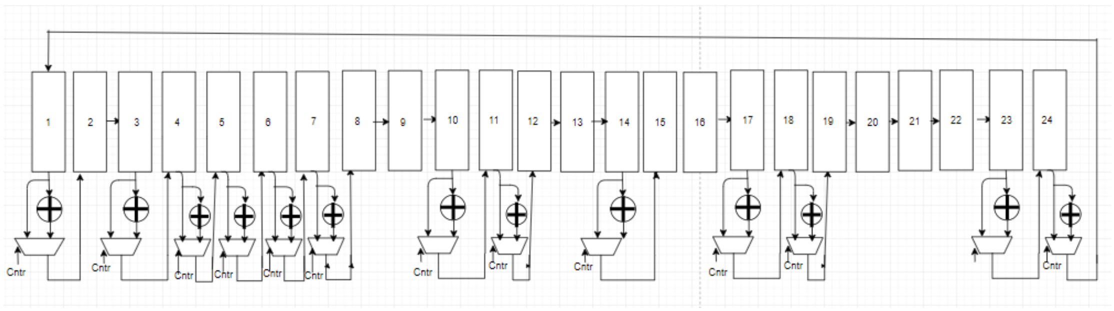


Figure 4.1.1.1-1: CRC shift registers

CRC in RTL is divided into 2 main blocks:

a. Linear-feedback shift register (LFSR) that contains the shift registers. Its interfaces as shown in figure 4.1.1.1-2 are:
   - in: input bits coming from upper layer.
   - cntr: control signal comes from crc_cu as a selection of multiplexer if it's equal 1 then shift only to get crc bits.
   - Y: output of crc

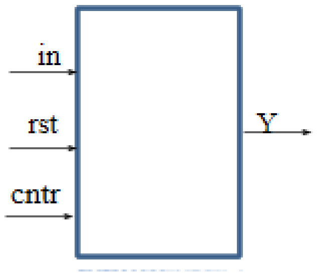


Figure 4.1.1.1-2: CRC LFSR block interfaces

b. Control Unit that contains 3 states
   - Idle state
   - Normal state: xoring the bits in the registers and shift.
   - Shift only state: doing shift without xor.

   Its interfaces are as shown in figure 4.1.1.1-3:

   - TBS: transport block size
   - Ready: indicates that crc is at normal state
   - Cntr: selection of multiplexer if it's equal 1 then shift only to get crc bits.

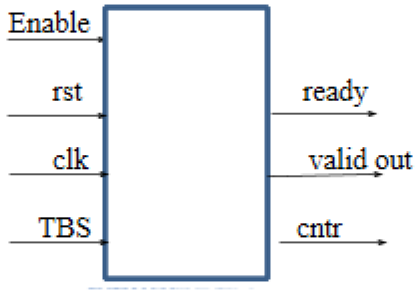


Figure 4.1.1.1-3: CRC Control Unit block interfaces

These 2 main blocks are the Architecture components of the CRC whose interfaces are shown in figure 4.1.1.1-4 and table 4.1.1.1-1:

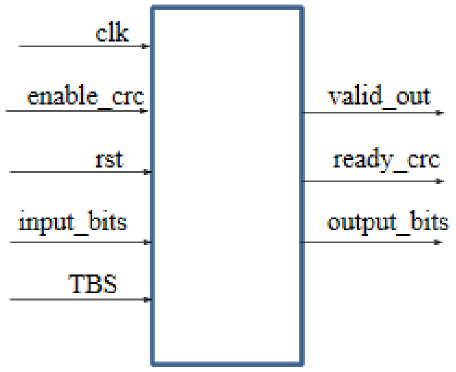


Figure 4.1.1.1-4: CRC block inputs and outputs

- **Block interfaces:**

Table 4.1.1.1-1: CRC block inputs and outputs

Port name	Direction	Width	Description
CRC_in	Input	1 bit	Input stream
CRC_clk	Input	1 bit	System clock
CRC_enable	Input	1 bit	Block Enable
CRC_RST	Input	1 bit	Block reset
TBS	Input	12 bits	Transport block size
CRC_OUT	Output	1 bit	CRC output bits
Valid_out	Output	1 bit	Flag signal to indicate the valid output
Ready_crc	Output	1 bit	indicates that CRC is at normal state
CRC_end	Output	1 bit	Flag to indicate the end of the valid output

4.1.1.2. Operation

CRC operation is started by entering input concatenated by 24 zeros and then taking the output. When the 24 CRC bits are ready in the shift registers we can’t take it through last register; as it will be xored together depending on polynomial, so the solution is to shift these bits only without "xoring" through multiplexers as seen in the Figure 4.1.1.1-1.

CRC enters each state depending on the transport block size, enable and reset. If we reset the Control Unit it will enter the idle state waiting for the enable signal and the input bits. After enabling it, it will enter the Normal state and after entering the whole transport block, it will enter shift only state to get the 24 CRC bits out without Xoring them.

4.1.2. Turbo Encoder

4.1.2.1. Structure and Block interfaces

Our proposed design as shown in figure 4.1.2.1-1 consists of:

- A LUT to store the values of f1 and f2
- Buffer for input stream storing, Restoring division unit for index calculation,
- Two main blocks:
  a. first block is responsible for shifting and xoring the input stream so it consists of three main elements:
     - Shift register to shift and xoring the input stream.
     - Multiplexer to release the input pin after K cycles to generate the trellis termination bits.
     - Counter to count of TBS, after that change the selection of the multiplexer to connect an internal signal instead of the input pin.

  b. Second Block is Output controller to control the positions of the 12 trellis terminations bits in the proper locations in the 3 streams which contain 6 multiplexers to help this operation to done in the specific mentioned manner.

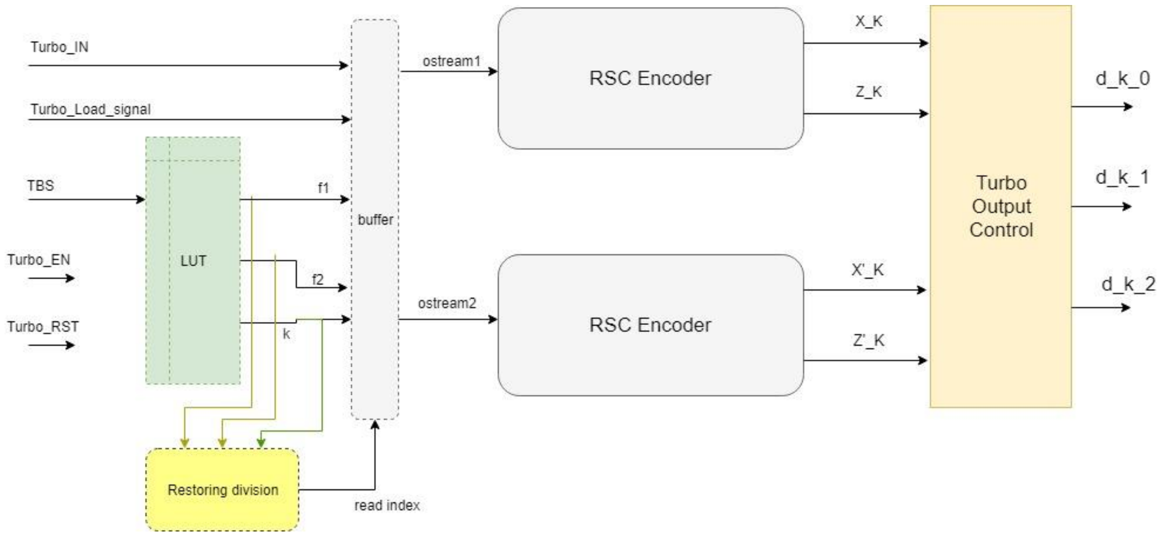


Figure 4.1.2.1-1: Turbo Encoder design Block Diagram

The top level block interfaces are shown in figure 4.1.2.1-2 and illustrated in table 4.1.2.1-1 as well.

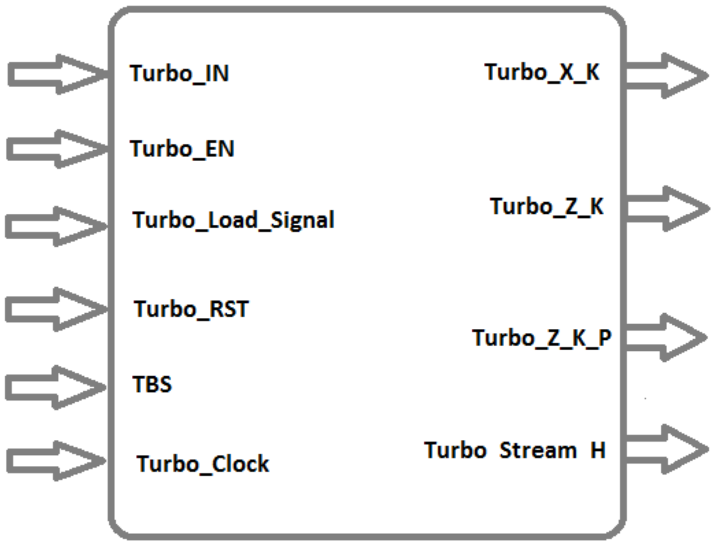


Figure 4.1.2.1-2: Turbo Encoder input and output ports

- **Block interfaces:**

Table 4.1.2.1-1: Turbo Encoder input and output ports

Port name	Direction	Width	Description
Turbo_in	Input	1 bit	Input stream
Turbo_clk	Input	1 bit	System clock
Turbo_enable	Input	1 bit	Block Enable
Turbo_Load_signal	Input	1 bit	Input Load signal
Turbo_RST	Input	1 bit	Block reset
TBS	Input	12 bits	Transport block size
Turbo_X_k	Output	1 bit	Output Stream 1
Turbo_Z_k	Output	1 bit	Output Stream 3
Turbo_Z_k_P	Output	1 bit	Output Stream 3
Turbo_Stream_H	Output	1 bit	Flag signal to indicate the valid output
Turbo_done	Output	1 bit	Flag to indicate the end of the valid output

4.1.2.2. Operation

- Internal interleaver:

According to a "New Low Area NB-IoT Turbo Encoder Interleaver by sharing resources" paper [5] which states for the interleaver that it receives the data block as a bit stream at the same time as the CRC receives the stream. Its function is to store the bits in a buffer and transmit them again but the bits should be interleaved for the Down Convolutional Encoder and the same input stream (in the same order) for the Up convolutional Encoder.

The index of the interleaved bits is derived from the polynomial (11) where K is the transport block size whose max value is 2536 bits adding to it the 24 CRC bits. f_1 And f_2 values depend on K and are implemented logically in the Look Up Table (LUT). The dividend $(f_1 * i + f_2 * i^2)$ is implemented using three behavioral RTL codes of multipliers.

The modulus function, which is responsible for providing reasonable diffusion of the indices, is implemented using Restoring division algorithm which takes n cycles for one index calculation, where n in our case = 33 bit.

- Recursive Convolutional:

According to 3GPP Narrowband LTE IOT standard, recursive convolutional encoder is a block from turbo encoder blocks which is different from the ordinary convolutional encoder by containing both feedback and feedforward signals ,it contain 3 shift registers and 4 XOR gates ,it's operation is shifting and xoring to the input stream bits and after the input stream is finished the input pin is released and the encoder continues shifting for 4 cycles to generate the trellis termination bits which are 12 bits divided on the 3 streams in a specific manner so every stream is ended by 4 trellis termination bits so if input stream size is K bits so every input stream length is K+4.

4.1.2.3. Challenges and enhancement

- We consider the input Transport block coming from the upper layer to enter the CRC and the Turbo blocks at the same time to same clock cycles and then the turbo will be disabled, waiting for the last 24 CRC bits to be ready to continue operating and start sending the bits to the two convolutional encoders.
- The modulus (Remainder) function, we have tried 3 different designs:
  a. Successive subtraction
     The input index *i* is generated by the input index counter and keep subtracting the dividend, which may reach tens of thousands, from the divisor k whose minimum number is 40. As expected that this algorithm will take thousands of cycles to

calculated one interleaved index which is refused and another Algorithm is implemented to reduce number of cycles.

b. Restoring Algorithm

For n-bit dividend it would take up to n cycles to calculate one interleaved index, adding to it some delay cycles. In our case, as k is up to 2560 then n = 33 bits then it would take 36 cycles for each bit index calculation.

As a comparison between the 2 design approaches (a) and (b) for the internal interleaver showing the area and power results:

Table 4.1.2.3-1: Area and Power comparison between the proposed 2 designs for the Turbo internal interleaver

Block name	Combinational Area (μm^2)	Non-Combinational Area (μm^2)	Total Area (μm^2)	Total Power (μW)
Buffer1 in (a)	83013.831681	70833.809622	153847.641303	411.8929
Buffer2 in (b)	74998.150985	72420.002228	147418.153213	32.6439
LUT	2625.217751	402.431385	3027.649136	0.5666004
Internal interleaver top	77709.267822	72814.196718	150523.464540	33.0996

4.1.2.4. Future Work

We have tried a Modulus (Remainder) Algorithm "An Improved Non-Restoring Algorithm" paper [6] which reduces number of shifts/cycles to the half of the dividend (n/2), where n is the number of bits the dividend is represented in. But unfortunately it doesn't pass completely, that’s may be due some assumption in the paper which we didn't take it into consideration.

4.1.3. Rate Matching

4.1.3.1. Structure and Block interfaces

- **Design:**

Two designs are introduced in this section the first design is the straight forward one as the standard mentioned containing all the blocks, while the other design is an optimized design in power, area and latency depending on tracing standard equations and optimizing these equations, it uses the idea of virtual circular buffer and that the data is already exists in sub-block interleavers to read the right data sequence according to specific manner so only sub-block interleavers are needed with some additional control.

Rate matching interfaces and description of each pin for both designs are illustrated in the following table 4.1.3.1-1.

Table 4.1.3.1-1: Rate Matching input and output ports

Port Name	Direction	Width	Description
RM_clk	Input	1 bit	Operating Clock
RM_reset	Input	1 bit	Reset the whole block
RM_Enable	Input	1 bit	Enable signal to enable the rate matching block to operate
RM_enable_ld	Input	1 bit	Enable block to read input streams
in_d0	Input	1 bit	First input stream which is d0 outp from turbo encoder of total Length K+4
in_d1	Input	1 bit	Second input stream which is d1 ou from turbo encoder of total Length K+4
in_d2	Input	1 bit	Third input stream which is d2 outp from turbo encoder of total Length K+4
TBS	Input	12 bit	Upper layer parameter to determine transport block size to use it in $\left(R_{subblock}^{TC}\right)$ calculations and dummy bits calculations
G	Input	12 bit	Upper layer parameter to determine output sequence length E
Rv	Input	1 bit	Redundancy version to choose the place we will start from while selec bits has only two values 0 equivale 0 in RTL and 2 equivalent to 1 in R
Qm	Input	1 bit	Modulation index used in rate matching to calculate the output ler E, it has two values 1 equivalent to

			RTL which indicates BPSK modulation scheme and 2 equivalent to 1 in RTL which indicates QPSK modulation scheme
E	output	12 bit	Output sequence length as it's need also by channel interleaver
RM_out	output	1 bit	Output bit stream from rate matchi block
RM_out_enable	output	1 bit	Valid output enable to inform next block to read rate matching output

a. *First Design structure using an actual Circular buffer is shown in figure 4.1.3.1-1.*

- 32-bit serial to parallel converters one for each stream.
- One port RAMs of size 128x32 bit as the maximum number of rows corresponding to TBS of 2536 bit is 81 rows.
- Circular buffer of maximum length 7692 bits to concatenate the three streams from the three arrays.
- Control unit.

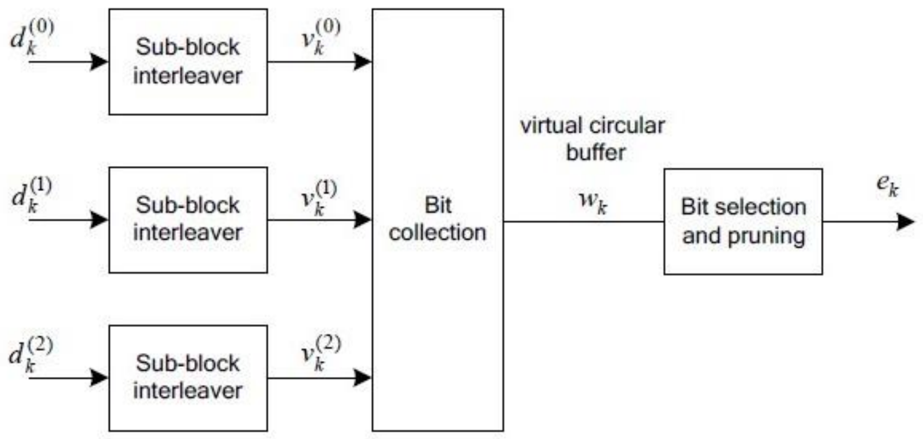


Figure 4.1.3.1-1: RM First Design using Direct Method Block Diagram

b. *Figure 4.1.3.1-2 shows the structure of the rate matching using the second Design Method and it's obvious that there is no buffer, where it composed of:*

- 32-bit serial to parallel converters one for each stream.
- One port RAMs of size 128x32 bit as the maximum number of rows corresponding to TBS of 2536 bit is 81 rows.
- Control unit.

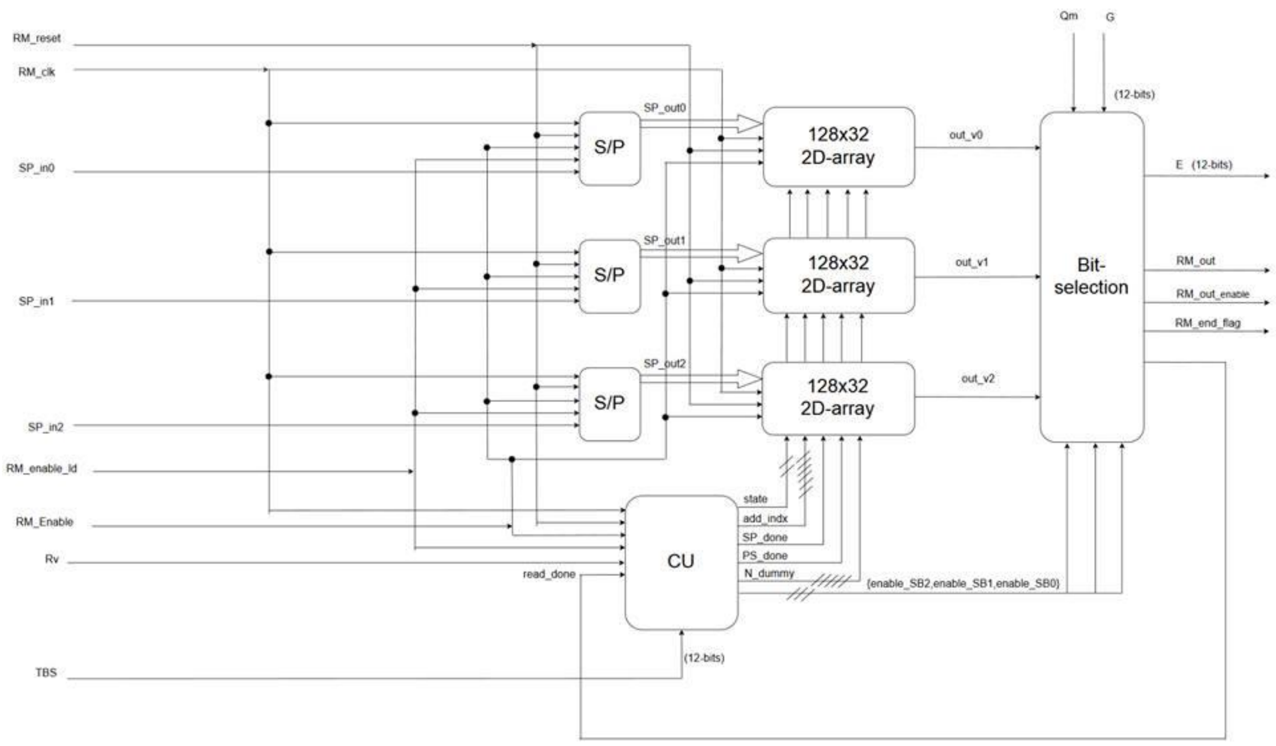


Figure 4.1.3.1-2: RM First Design using Virtual Circular buffer Block Diagram

4.1.3.2. Operation

a. *The Operation of the First Design is simply:*

1. Firstly, write state begins where the input streams are serially received so it needed to be converted to parallel to be stored in the 2-D array during write state that's why 32-bit serial to parallel converter is used to write data row by row.
2. After the end of write state read state begins at which data be read and stored in circular buffer according to manner mentioned by standard in the previous section.
3. The next step is reading data from the buffer at which start point determined by k_0 mentioned in previous section and output sequence starts with valid output enable of value 1.

b. *While the Operation of the second Design is:*

1. The write state begins where the input streams are serially received so it needed to be converted to parallel to be stored in the 2-D array during write state that's why 32-bit serial to parallel converter is used to write data row by row as the first design.
2. After tracing the equation $k_0 = R_{subblock}^{TC} \cdot (24 \cdot rv_{idx} + 2)$ when $rv_{idx} = 0$ then $k_0 = 2R_{subblock}^{TC}$ which indicates that for each $R_{subblock}^{TC}$ the output sequence always starts from the third permuted column of first sub-block interleaver, on the other hand when $rv_{idx} = 2$ then $k_0 = 50R_{subblock}^{TC}$ which indicates that for each $R_{subblock}^{TC}$ the output sequence always starts from the eighth permuted column of second sub-block interleaver so the start points are now

determined so the output sequence can be directly read from sub-block interleavers without any need for physical buffer which is the idea of virtual circular buffer.

4.1.3.3. Challenges and enhancement

We worked on some enhancement and optimization procedures to optimize in the hardware and power in the 2 designs as much as we could. In this section we will illustrate the Challenges we met and how we managed to solve it and ways of optimizations for the two designs.

a. *For the First Design:*

1. All equations are executed without making any multiplication or divisions only used operations are addition, subtraction and shifting.
2. The block has an enable to reduce power consumption by disabling block when it isn't needed.
3. One of the big challenges was the dummy bits, these bits used only to complete the rectangular 2-D array so it's difficult in RTL to differentiate between real data bit which is 0 or 1 and dummy bit which is 0, as a solution an additional counter is used to determine the location of dummy bits to avoid reading these bits from the buffer.
4. Another challenge that leads to the other design is the area and power at which the buffer and additional counter consume very large area and power.

b. *For the Second Design:*

1. All equations are executed without making any multiplication or divisions only used operations are addition, subtraction and shifting.
2. The block has an enable to reduce power consumption by disabling block when it isn't needed.
3. Here the dummy bits challenge is easier as their locations are known inside the interleavers so they can be easily avoided.
4. Another challenge is controlling the three 2-D arrays together this can be done using three enables one for each array (enable_SB0, enable_SB1 and enable_SB2).
5. Area, power and latency obviously enhanced as there is no additional buffers or counters.

4.1.4. Data Multiplexing and Channel interleaver

In our design we proposed that multiplexing is done in a 2D interleaver memory directly and not in a separate block to reduce power and time.

4.1.4.1. Structure and Block interfaces

Our proposed design consists of Four 2D memories for the stream to be stored row by row and read column by column, 2 serial to parallel blocks for writing , 1 parallel to serial block for reading, Multiplexers to select the proper Memory and gate the others according to an upper layer signal and a control unit to manage the reading and writing operation as shown in figure 4.1.4.1-1. The top level block interfaces are shown in table 4.1.4.1-1 as well.

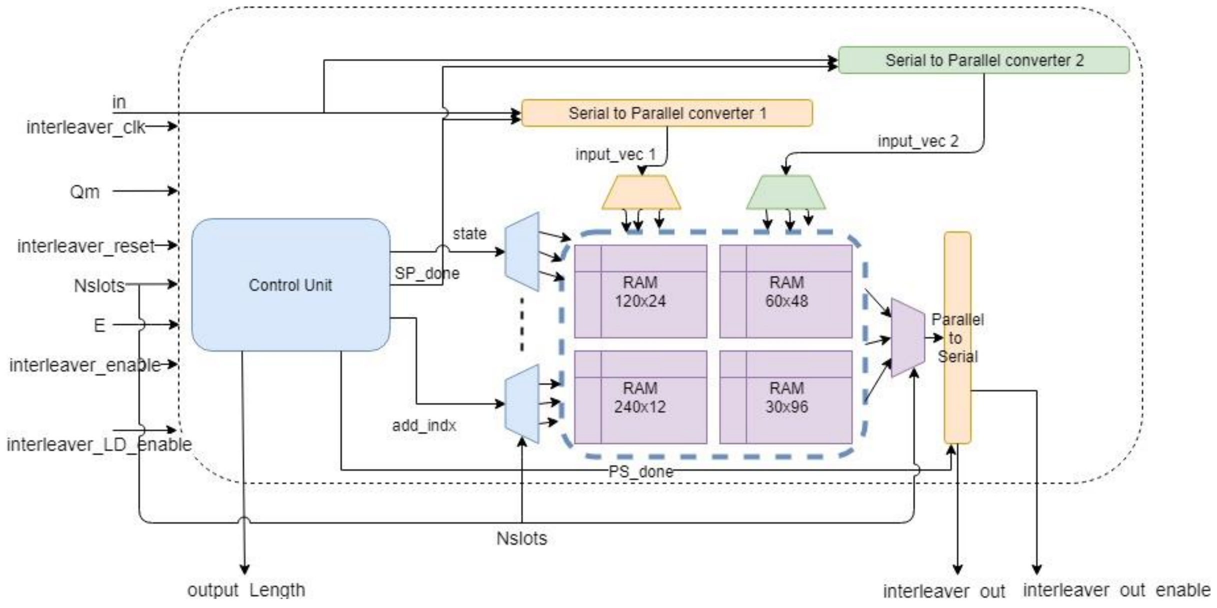


Figure 4.1.4.1-1: Data Multiplexer and Channel interleaver Block Diagram

- **Block interfaces:**

Table 4.1.4.1-1: Data multiplexing and channel interleaver input and output ports

Port name	Direction	Width	Description
in	Input	1 bit	Input stream
interleaver_clk	Input	1 bit	System clock
interleaver_enable	Input	1 bit	Block Enable
Interleaver_LD_enabler	Input	1 bit	Input Load signal
interleaver_reset	Input	1 bit	Block reset

G(=E)	Input	12 bits	Rate matching total output bits
N_slots	Input	5 bits	Number of subcarriers in the frequency domain for NB-IoT
Q_m	Input	1 bit	Modulation scheme, '0' for BPSK and '1' for QPSK
Interleaver_out_enable	Output	1 bit	Valid output signal, high at valid output is ready
Interleaver_out	Output	1 bit	Valid output from the interleaver
Output_Length	Output	12 bits	Output total number of bits (Rows*Columns)
CI_end_flag	Output	1 bit	Flag that indicates the end of the interleaver valid output

4.1.4.2. Operation

As shown in the above figure 4.1.4.1-1, in the write state, input stream is written in a serial to parallel 1/2. In BPSK case, then we write input bits in SP1 after calculating number of columns using "N_slots" parameter and then write this row in the selected memory according to the mentioned upper layer signal in the address index output from the control unit. In QPSK case, input bits is written in SP1 in the 1st cycle then in SP2 in the 2nd cycle and so on then these two rows is written in the 2D memory in address index and address index+1.

In the Read state, after writing all input bits and filling the uncomplete rows with Zeros, column by column is transferred to the parallel to serial block to read it bit by bit.

4.1.4.3. Challenges and enhancement

- First enhancement is including the multiplexing to be done inside the interleaver block to decrease power and area as mentioned before.
- Firstly, 2D memory is chosen with the maximum size instead of 1D memory to decrease the decoding complexity as shown in figure 4.1.4.3-1. Nevertheless, it consumes a significant amount of power so a Four Memories' Design is built, each for certain number of columns and the unused memory is gated to reduce the consumed power.

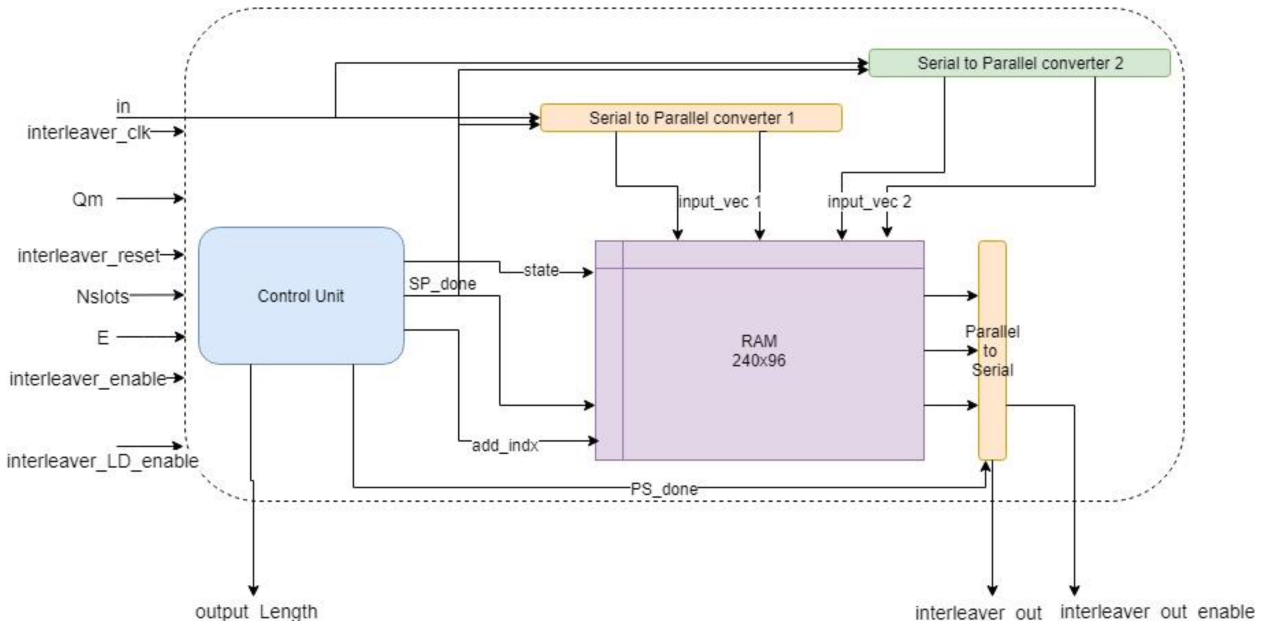


Figure 4.1.4.3-1: Data Multiplexing and Channel Interleaver Hardware Architecture for Design 1 with maximum memory size

4.2. Physical Channel and modulation

4.2.1. Scrambler

4.2.1.1. Structure and Block interfaces

The structure is as shown in figure 4.2.1.1 which consists of:

1. Two linear feedback shift registers of length 31 to generate the gold sequence.
2. Combinational logic to calculate linear feedback shift register initial value.

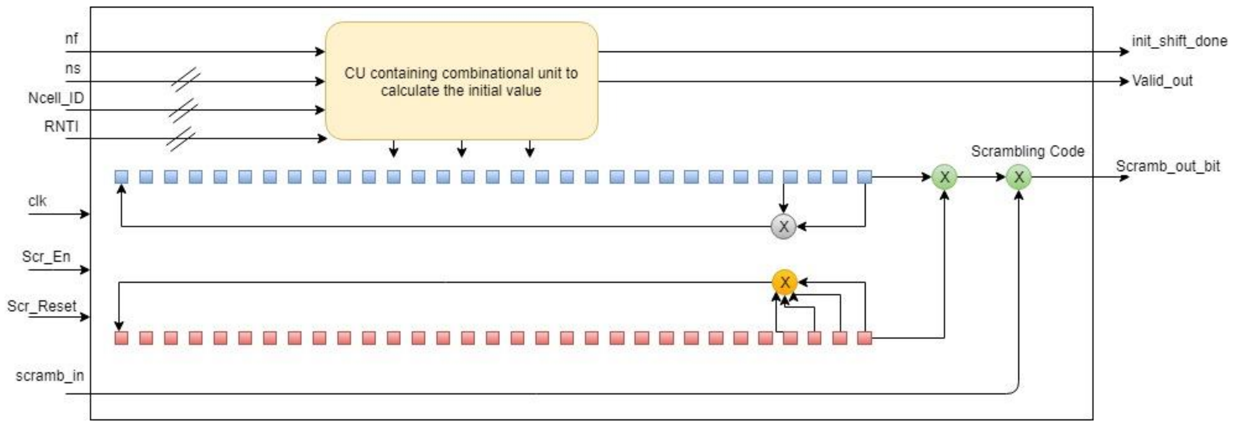


Figure 4.2.1.1-1: Scrambler design block diagram

Its interfaces and description of each pin is illustrated in the following table 4.2.1.1-1.

Table 4.2.1.1-1: Scrambler input and output ports

Port name	Direction	Width	Description
clk	Input	1 bit	Operating Clock
reset	Input	1 bit	Reset the whole block
Scr_Enable	Input	1 bit	Enable signal to enable the scrambler block to operate
Enable	Input	1 bit	Enable block to read input streams
scramb_in_bit	Input	1 bit	Input serial bits from channel interleaver
RNTI	Input	16 bit	Radio Network Temporary ID given from the upper Layer
Ncell_ID	Input	9 bit	Cell ID given From the upper Layer
ns	Input	4 bit	First Slot number 4 MSB in transmission given from the upper layer
nf	Input	1 bit	First System Frame number LSB in transmission given from the upper layer

Data_Size	Input	12 bit	Total input stream length
init_shift_done	output	1 bit	Output flag indicates that the 1600 initial shift are done so the input can be received.
scramb_out_bit	output	1 bit	Output bit stream from scrambler block
Valid_sc_out	output	1 bit	Valid output enable to inform next block to read scrambler output

4.2.1.2. Operation

a. After reset, the scrambler goes to an initialization state to assert the initial values of shift registers.
b. When initialization completed, shift state is enabled so that 1600 shifts are performed taking 1600 clock cycles
c. Then the block is disabled waiting for a valid input coming from channel interleaver.
d. The last step is that XOR logical operation is performed on the valid input with the gold sequence generated by LFSR to get the output sequence serially.

4.2.1.3. Challenges and enhancement

- First challenge was calculating the initial value of x_2 as there is more than one multiplication but it was noticed that all multiplications are by number of base 2 so multiplications can just be performed using shift and concatenation only.
- Another challenge is the large number of input ports due to upper layer parameters used in x_2 initial value calculations as an optimization not all the bits taken as input from some parameters for example first system frame number consists of 10 bits but only LSB bit can be used in design as compensation of $n_f \bmod 2$, another example is the first slot number which consists of 5 bits but only 4 MSB can be used as compensation of $\lfloor n_s/2 \rfloor$.

4.2.2. Modulation Mapper

4.2.2.1. Structure and Block interfaces

Modulator design components are simply a Look Up Table to map the input bits to the required symbol and a control unit to handle this mapping operation as shown in figure 4.2.2.1-1.

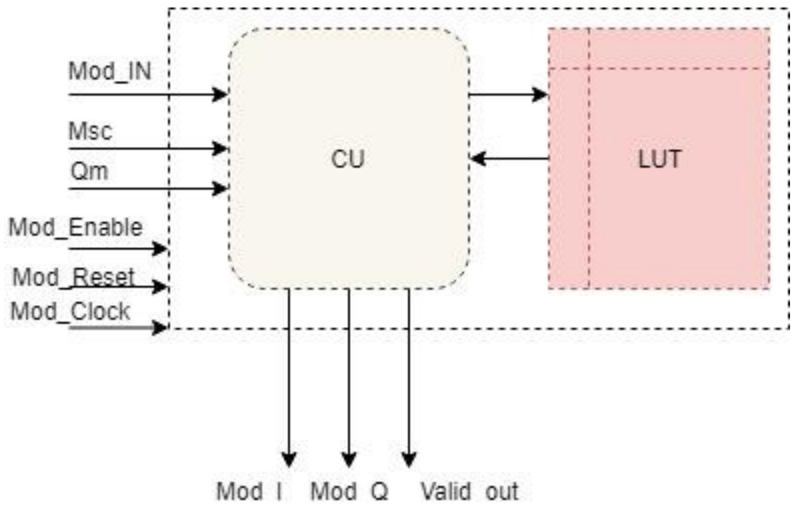


Figure 4.2.2.1-1: Modulation Mapper Design Architecture

This block require some input and output interfaces for this mapping operation which depends mainly on the number of subcarriers. These interfaces are shown in figure 4.2.2.1-2 and illustrated in table 4.2.2.1-1.

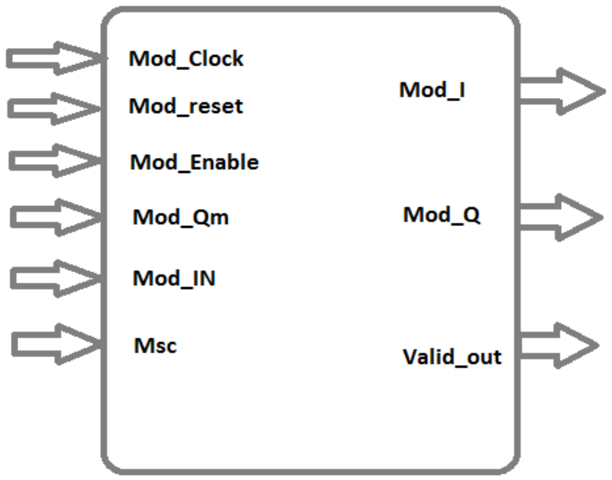


Figure 4.2.2.1-2: Modulation Mapper input and output ports

- **Block interfaces:**

Table 4.2.2.1-1: Modulation Mapper input and output ports

Port name	Direction	Width	Description
Mod_IN	Input	1 bit	Serial input Stream from Scrambler
Mod_clock	Input	1 bit	System clock
Mod_Enable	Input	1 bit	Block Enable
Mod_Reset	Input	1 bit	Block reset
Mod_Qm	Input	1 bit	Upper Layer signal to identify the type of modulation "0" for BPSK and "1" for QPSK
Msc	Input	2 bits	Signal to identify the assigned number of subcarriers 3 , 6 or 12
Mod_I	Output	16 bits	Real Component of the output Symbol
Mod_Q	Output	16 bits	Imaginary Component of the output Symbol
Valid_out	Output	1 bit	Signal high with output symbol and Low in idle time

4.2.2.2. Operation

The ordinary modulator performs only mapping the input bits stream to corresponding symbol, the corresponding symbols is located in a lock up table inside the modulator block as shown in the design Architecture figure 4.2.2.1-1. For a BPSK case, the Modulation Mapper map the input bit to the corresponding symbol each clock cycle. While for the QPSK case, each two bits is mapped to the corresponding symbol so it takes 2 clock cycles to generate one symbol to the next block.

According to one of the important Narrowband LTE IOT requirements which is small power consumption design so our proposed Modulator perform its ordinary function and also help FFT to perform its function to obtain a less power design, due to the linearity in the FFT equation; instead of dividing our variable size FFT by square root of the number of subcarriers that takes values 3,6 or 12 which require a divider which consumes a considerable power; we put in the lock up table the symbols divided by the number of subcarriers and according to upper layer information about used number of subcarriers; the proper symbols is the output of the modulator.

4.2.2.3. Challenges and enhancement

We tried to design the minimum power Modulation Mapper block using less hardware. Two designs are proposed for this block and we chose the minimum power design which is the 2nd design. The Following table shows the difference between the two designs in area and power results.

Table 4.2.2.3-1: Area and Power Synthesis results Comparison between the two proposed designs

Block name	Combinational Area (μm^2)	Non-Combinational Area (μm^2)	Total Area (μm^2)	Total Power (μW)
Modulator1	344.773108	1059	1403.8	0.7556811
Modulator2	406	1051.96	1457.93	0.6291335

4.2.3. FFT

The principle of SC-FDMA as illustrated in Chapter 2 is to generate (in every UE) a single-carrier signal and to shift this signal to the frequency resource that is assigned to the respective UE. The necessary frequency shift is implemented by the N-IDFT. The three principal blocks are:

DFT: Take M QAM symbols y (n) (which can be interpreted as M consecutive symbols of a single-carrier signal) and calculate the frequency-domain representation Y (k) of this signal by an M-DFT.

Subcarrier mapper: "Place" Y (k) in the freq. domain at the freq. resource that is assigned to the UE. This is done by assigning Y (k) to the M IDFT inputs that represent the wanted freq. resource and by setting the remaining N−M IDFT inputs to zero.

IDFT*:* Calculate the time domain signal.

The main advantage of SC-FDMA is the low PAPR of the transmitted signal, PAPR is a big concern for user equipment, as PAPR relates to the power amplifier efficiency as low PAPR allows the power amplifier to operate close to the saturation region resulting in high efficiency that is why SC-FDMA is the preferred technology for user terminals.

4.2.3.1. Challenges

As shown in table 1.7.3-1, the number of allocated subcarriers in NB-IoT with NPUSCH format 1 and subcarrier spacing of 15 KHz, varies between 1, 3, 6 and 12 subcarriers, which arises the need of implementing an FFT block that supports different numbers of inputs. Moreover, none of the number of subcarriers is power of 2, so the simple radix 2 algorithm won’t suit our case. That’s why a mixed radix FFT algorithm was proposed and implemented where both radix 2 and radix 3 algorithms were used.

Figure 4.2.3.1-1 and 4.2.3.1-2 shows radix 2 and radix 3 flow graphs.

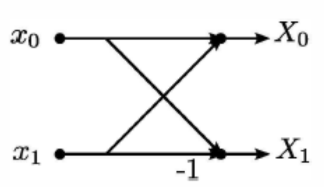


Figure 4.2.3.1-1: Radix-2 flow graph

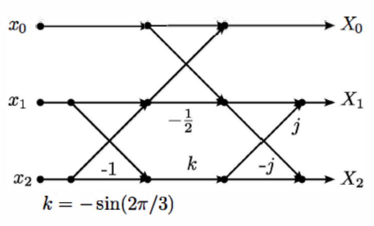


Figure 4.2.3.1-2: Radix-3 flow graph

4.2.3.2. Proposed design

a. Modulation buffer

As discussed in the modulation mapper section, the output of the modulator may be produced every clock cycle as in BPSK or every 2 cycles as in QPSK. That's why a buffer is needed after the modulation mapper and before the FFT to store the results of the modulator and produce data to the FFT every clock cycle in BPSK and QPSK as well.

- **Structure and Block interfaces**

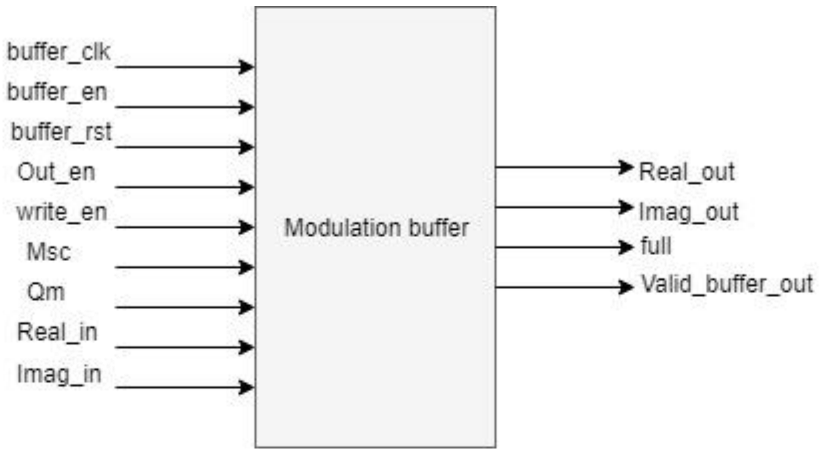


Figure 4.2.3.2-1: Pins description of modulation buffer

Figure 4.2.3.2-1 shows the pins and interfaces of this buffer, and their description are described in table 4.2.3.2-1.

Table 4.2.3.2-1: Modulation Buffer Input and Output Ports' Description

Port name	Direction	Width	Description
buffer_clk	Input	1 bit	Operating Clock
buffer_rst	Input	1 bit	Reset the whole block
out_en	Input	1 bit	Enable reading the data from the buffer
Write_en	Input	1 bit	Enable writing the data in the buffer
Msc	Input	2 bits	Upper layer signal to tell 3,6,or 12 subcarrier mode
Qm	Input	1 bit	Modulation order
Real_in	Input	16 bits	Real part of the input symbol
Imag_in	Input	16 bits	Imaginary part of the input symbol
Valid_buffer_out	output	1 bit	To tell that the output of the buffer is valid
Full	output	1 bit	To tell if the buffer is full or empty
Real_out	output	16 bits	Real part of the output symbol
Imag_out	output	16 bits	Imaginary part of the output symbol

- **Operation**

The buffer operations will be as follows, it will store the data from the modulator till it's full, and when it's full it will be disabled with all of the previous blocks. When the FFT is ready to receive new input, the buffer will be enabled and FFT will read the data from the buffer till its empty again then all of the block re operate till the buffer is full and so on.

b. FFT

As discussed mixed radix algorithm was implemented. We used this algorithm to implement multi point FFT block that could be either 1-FFT, 3-FFT, 6-FFT or 12-FFT. To implement low power FFT design, the memory based hardware architecture was used. Memory-based architecture usually performs the FFT in serial, e.g. [10]–[15], and this results in a low area cost for implementing the memory-based FFT processor, however, memory-based FFT processors need two-port memories for buffering the computational data. Compared to single-port memories, two-port memories have the following drawbacks. First, the area cost of the two-port memory is greater than that of the single-port memory. The two-port memory consumes more power and its test cost is greater.in [16], a new memory-based FFT processor is proposed. In the proposed FFT processor, only four single-port memories are needed for buffering the computational data. Therefore, the area and test costs are much lower than those of the existing memory-based FFT processors.

- **Decimation in frequency**

The DFT of a data sequence x(r) [r=0, 1….N-1] is defined as,

$$x[k] = \sum_{r=0}^{N-1} x[r]\ w_N^{Kr} \quad k = 0,1,\dots N-1 \tag{25}$$

Where $w_N = e^{-j\frac{2\pi}{N}}$, when the even-numbered frequency samples and the odd-numbered frequency samples are computed separately, then the two types of samples can be expressed as.

$$x[2k] = \sum_{r=0}^{\frac{N}{2}-1} (x[r] + x[r+\tfrac{N}{2}])w_{\frac{N}{2}}^{Kr} \quad k = 0,1,\dots N/2-1 \tag{26}$$

$$x[2k+1] = \sum_{r=0}^{\frac{N}{2}-1} (x[r] - x[r+\tfrac{N}{2}])w_{\frac{N}{2}}^{Kr}\ w_N^r \quad k = 0,1,\dots N/2-1 \tag{27}$$

We used decimation in frequency for our FFT processor. Let g[r] = x[r] +x[r +N/2] and h[r] = x[r]-x[r-N/2]. The DFT can be computed by, first, forming the sequences g[r] and h[r] then subsequently computing h[r] w_N^r ; and finally computing the N/2 point DFTs of the sequences g[r] and h[r] to get the even-numbered and odd-numbered output points. However, for the 3,6,12 point FFT , N/2 point DFT couldn't be applied for all stages, so N/2 points will applied till we reach 3 point FFT then radix 3 FFT will be applied as shown in the signal flow graphs (SFGs).

SFG for 3-FFT, 6-FFT and 12-FFT is shown in figures 4.2.3.2-2, 4.2.3.2-3 and 4.2.3.2-4 respectively.

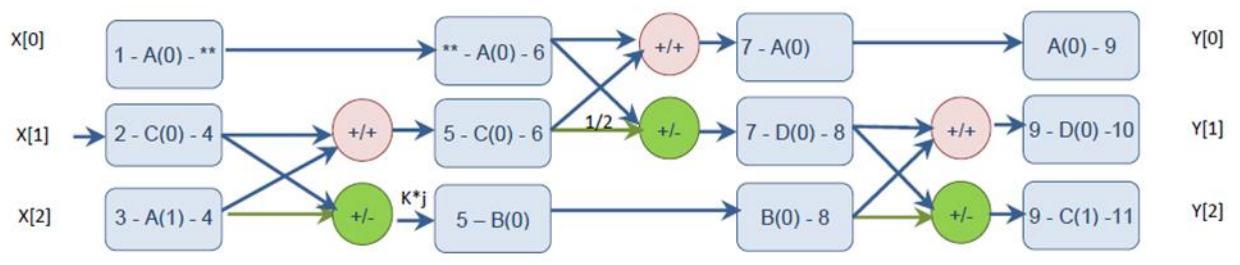


Figure 4.2.3.2-2: SFG of 3-point FFT

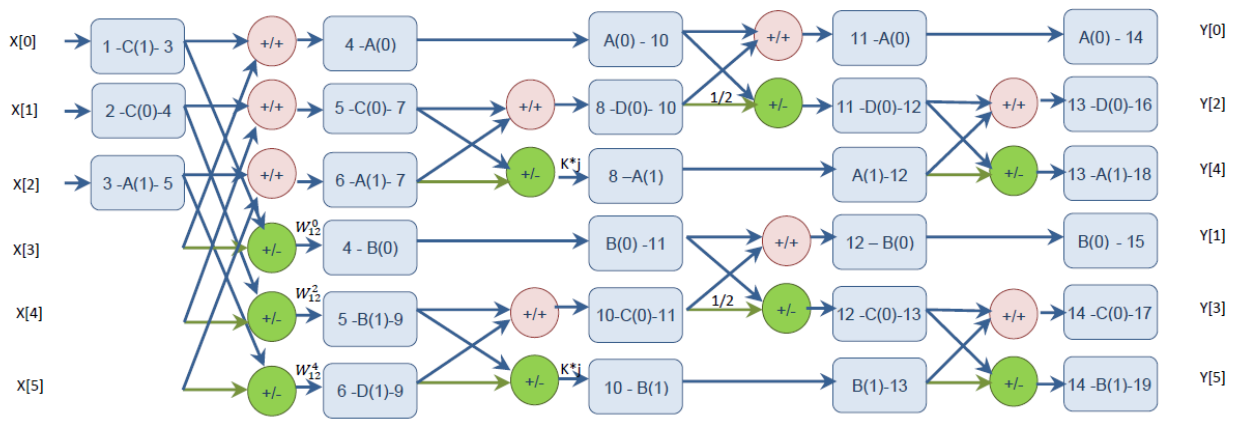


Figure 4.2.3.2-3: SFG of 6-point FFT

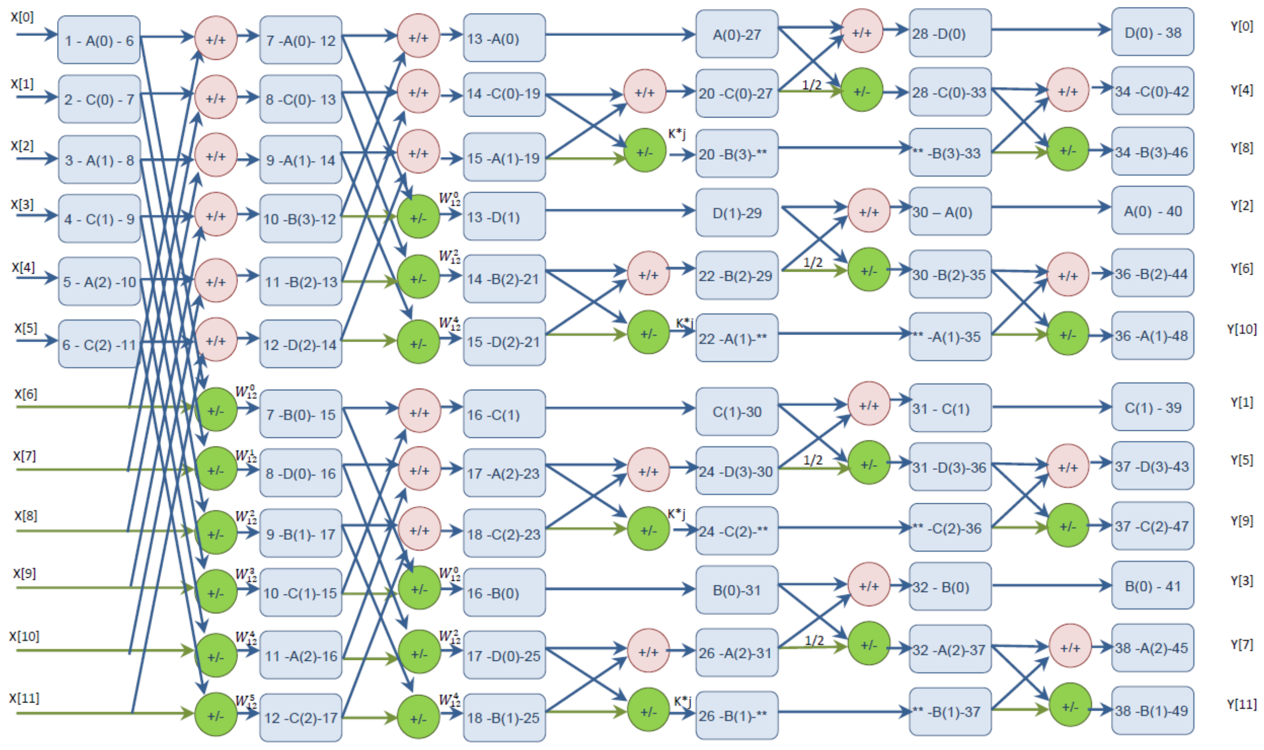
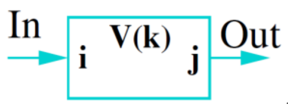


Figure 4.2.3.2-4: SFG of 12-point FFT

→ *Legend:*



Data In is written into the address k of the memory V at the ith clock and the data of the address k of the memory V is read to Out at the jth clock.

In figure 4.2.3.2-2, 3 point FFT was implemented using radix 3 FFT algorithm shown in figure 4. In figure 4.2.3.2-3, 6 point FFT was implemented as multi stage FFT, in the first stage it was split into two 3 point FFT then radix 3 was implemented. In figure 4.2.3.2-4, 12 point FFT was implemented. In the first stage 12 point was split into two 6 point FFT, then in the second stage, each 6 point FFT was split into two 3 point FFT then radix 3 algorithm was implemented.

- **Structure and Block interfaces**

  - **<u>Memory based hardware architecture</u>**

Memory based IFFT is the most suitable choice for low power implementation which is the main goal in case of IoT systems, this reduces the area, power, and test cost. Memory-based architecture usually performs the FFT in serial, i.e one butterfly operation at a time instead of more than one in parallel, and this results in a low area cost for implementing the memory-based FFT processor.

As shown in figure 4.2.3.2-5, Data path of this design consists of four single port RAMs to store and compute intermediate results of the 3,6,12 point FFT, seven multiplexers, two adders, one multiplier, one ROM, and one controller. The four single-port RAMs are used for buffering the computational data. The multiplexers are responsible for switching the data flow between the storage and arithmetic components. The adder and multiplier execute the computation of the two-point FFT. The ROM stores the twiddle factors. The addend and augend of the left adder can be changed by controlling the Ch signal. For example, if Ch=0, then the adder executes A-B. However, if Ch=1, then the adder executes B-A.

The controller, which is shown in figure 4.2.3.2-5, generates the controlling signals for the multiplexers and the four RAMs.

- **Implemented Block diagram**

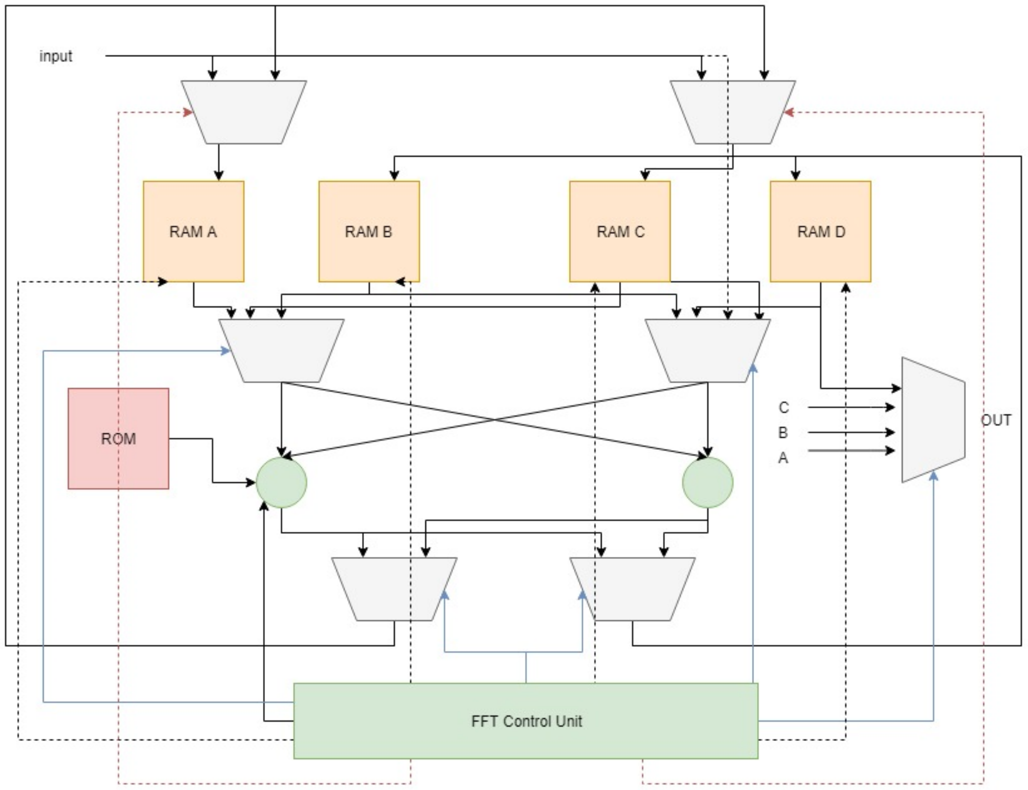


Figure 4.2.3.2-5: Implemented memory based FFT architecture

- **<u>Interfaces and pins description</u>**

The description of all input and output pins shown in figure 4.2.3.2-6 are described in table 4.2.3.2-2.

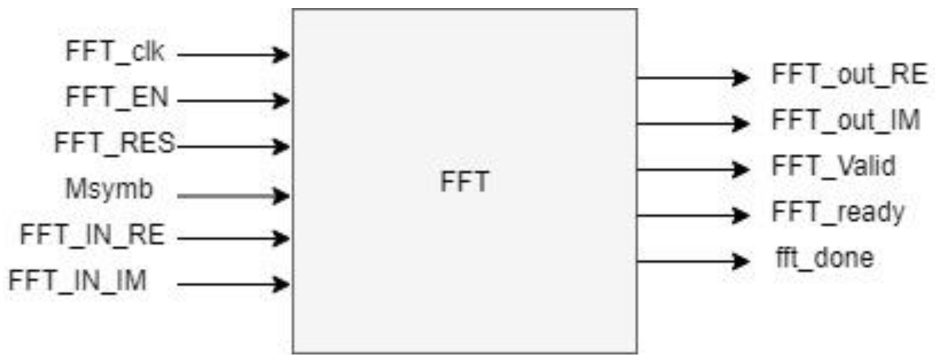


Figure 4.2.3.2-6: FFT Top Level Block diagram interfaces

Table 4.2.3.2-2: FFT Input and Output Ports' Description

Port name	Direction	Width	Description
FFT_clk	Input	1 bit	Operating Clock
FFT_RES	Input	1 bit	Reset the whole block
FFT_EN	Input	1 bit	Enable signal to enable the FFT block to operate
Msymb	Input	2 bits	To choose between 3 and 6 and 12 points FFT
FFT_IN_RE	Input	16 bits	Real part of the input symbol
FFT_IN_IM	Input	16 bits	Imaginary part of the input symbol
FFT_out_RE،	output	16 bits	Real part of the output symbol
FFT_out_Im،	output	16 bits	Imaginary part of the output symbol
FFT_Valid	output	1 bit	To tell that the output of the FFT is valid
FFT_ready،	output	1 bit	To tell that the FFT is ready to receive new input

- **Operation**

Data flow control is performed on this data path based on shown signal flow graph (SFG) described before, input symbols from the modulator are stored serially in memories.

Figure 4.2.3.2-5 shows the block diagram of the memory based FFT architecture. As the hardware is shared, a control unit was built to derive the control signals to the memories and multiplexers to operate properly in the required cycles. Data enters the memories in the first cycles then one butterfly operation from the signal flow graphs is done, in the next cycle another operation is done till finishing all the required operations for computing the output. Results of the intermediate operations are stored in the memories to be used again in the next operations and the flow of cycles assures that no data is over written till finishing its operations.

4.2.3.3. Future Work

- For the FFT implementation, larger number of bits may be taken to represent the fraction parts of the results to increase the output accuracy

- To minimize the complexity of the implementation of the next block (resource element mapper), 3,6,12 point FFT were assumed to take 54 cycles to complete their operations although they take less number of cycles. So the architecture may be improved if a more complex algorithm is implemented such that the 3, 6 and 12 point FFT take their exact number of cycles.

4.2.4. Resource element Mapper

4.2.4.1. Structure and Block interfaces

Figure 4.2.4.1-1 shows the input and output pins to the resource element mapper and their description are found in 4.2.4.1-1.

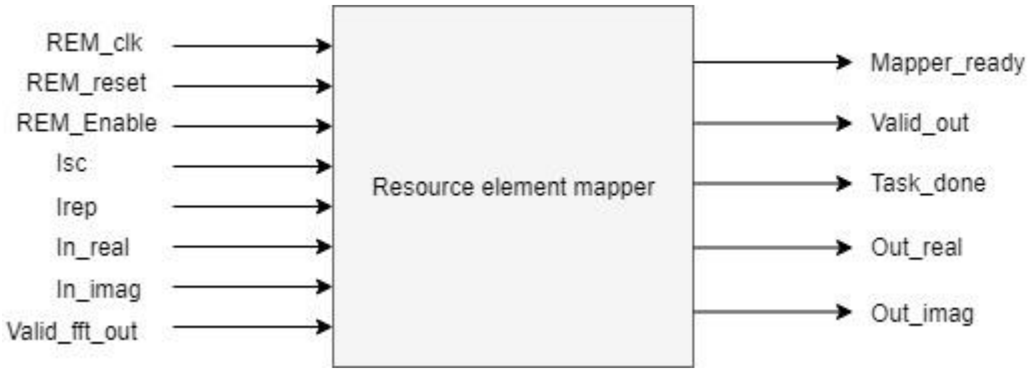


Figure 4.2.4.1-1: Pins description and interfaces for resource element mapper

- **Block interfaces:**

Table 4.2.4.1-1: pins description and interfaces for resource element mapper

Port name	Direction	Width	Description
REM_clk	Input	1 bit	Operating Clock
REM_res	Input	1 bit	Reset the whole block
REM_enable	Input	1 bit	Enable signal to enable the REM block to operate
Isc	Input	5 bits	Upper layer signal to tell the number of subcarriers and their start index
Irep	Input	3 bits	Upper layer signal to tell the number of repetitions
In_real	Input	16 bits	Real part of the input symbol
In_imag	Input	16 bits	Imaginary part of the input symbol
Valid_FFT_out	Input	1 bit	To tell that the output of the previous block is

			valid
Mapper_ready	Output	1 bit	To tell the previous block that the rem is ready receive input
Valid_out	Output	1 bit	Output of REM is valid
Task_done	Output	1 bit	REM has finished its task
Out_real	Output	16 bits	Real part of the output symbol
Out_imag	Output	16 bits	Imaginary part of the output symbol

4.2.4.2. Operation

As mentioned before in Chapter 2, the Function of the Resource Element Mapper is to assign the output symbols from the FFT to the proper subcarriers at the input of the IFFT block. In the case of 12 points FFT, figure 4.2.4.2-1 shows that all the symbols are assigned to the 12 subcarriers allocated for the NB-IoT bandwidth and this is was provided as information from the upper layer as shown in table 1.7.4 2. In the case of 6 point FFT, figure 4.2.4.2-2 the output 6 symbols of the FFT will be assigned to a certain 6 subcarriers of the 12 assigned subcarriers and the rest are padded with zeros and the same will be done with 3 point FFT results as shown in figure 4.2.4.2-3. The location of the symbols within the 12 subcarriers in case of 3 and 6 subcarriers is calculated from table 1.7.4 2.

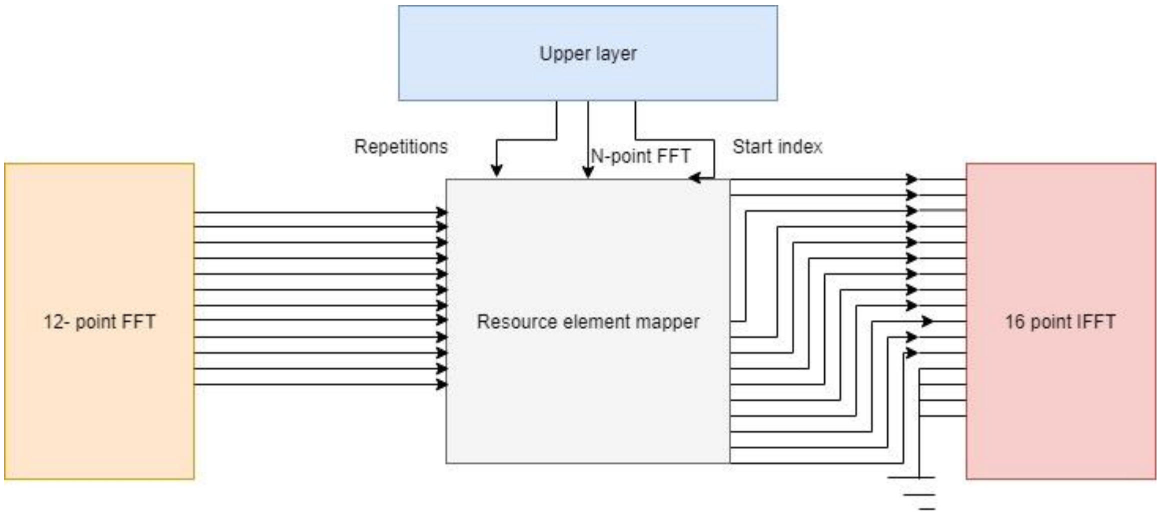


Figure 4.2.4.2-1: Resource allocation in 12-point FFT

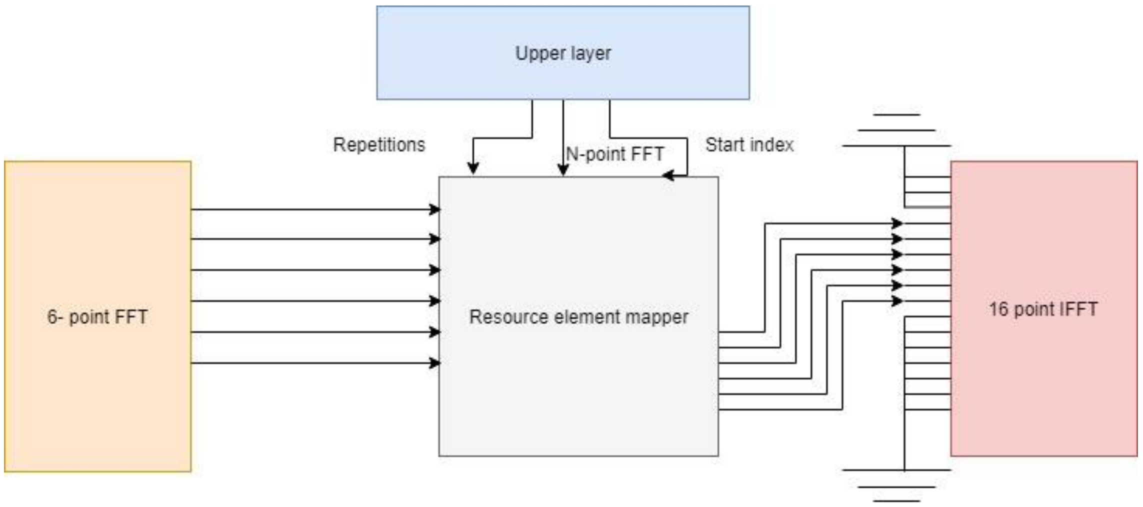


Figure 4.2.4.2-2: Resource allocation in 6-point FFT

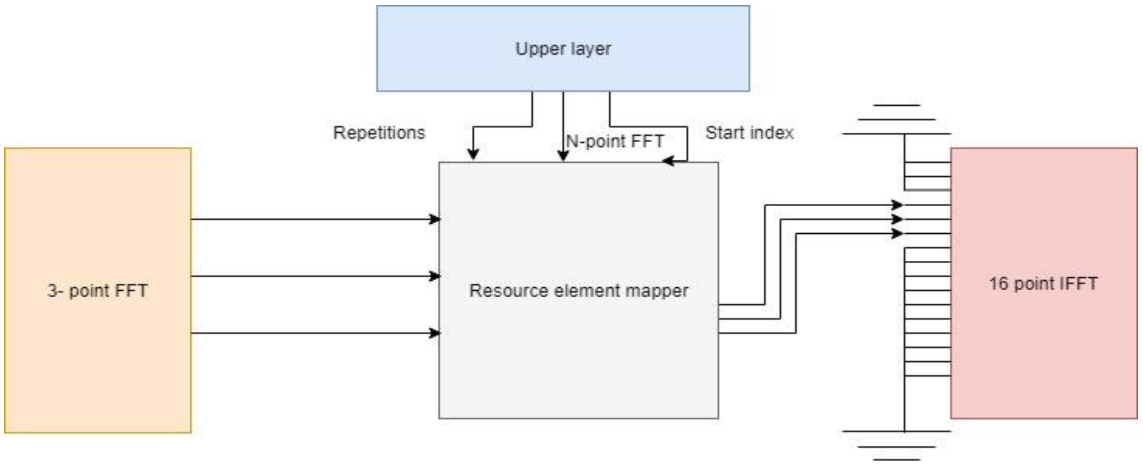


Figure 4.2.4.2-3: Resource allocation in 12-point FFT

4.2.4.3. Challenges and enhancement

- **Repetitions:** As NB-IoT support repetitions as shown in table 3.2.4.4-2, REM must be able to store the symbols for at least 2 slots so that in the case of repetitions it manages to send the old symbols again, however if there are no repetitions this memory will be a waste of resources. That's why the resource element mapper was implemented to operate in 4 modes:

  1. <u>12 FFT without repetitions mode</u>: in this mode the REM acts as a channel to pass the input to the output without any operations on the input data.
  2. <u>3 or 6 point FFT without repetitions:</u> in this mode, REM calculates the position of the symbols then it shifts the data to its exact position and pads the rest of the subcarriers with zeros.
  3. <u>12 point FFT with repetitions:</u> as 1, however a memory will be used to handle repetitions.
  4. <u>3 or 6 point FFT with repetitions:</u> as 2, however a memory will be used to handle repetitions.

  So, now no waste of resources will occur as the memory will be used only in the repetitions case

- **FFT enabling and disabling:** FFT takes 54 cycles to produce new data to the resource element mapper, however if repetitions occur, the REM will be sending and old data stored in its memory, so the data from the FFT will be lost. That's why the REM sends a signal to the control unit to tell if it can receive new input or not, if it can't, the control unit disables the FFT and all the previous blocks, if it can the control unit enables the FFT and all the previous blocks.

4.2.5. IFFT and CP

4.2.5.1. Structure and Block interfaces

Memory based IFFT is the most suitable choice for low power implementation as illustrated in FFT section 4.2.3 which is the main goal in case of IoT systems, this reduces the area, power, and test cost. Memory-based architecture usually performs the IFFT in serial, i.e. one butterfly operation at a time instead of more than one in parallel, and this results in a low area cost for implementing the memory-based FFT processor.

Data path of this design as shown in figure 4.2.5.1-1 consists of 4 RAMSs to store inputs and intermediate results of the 16 point IFFT in addition to ROM to store the twiddle factor values and single butter fly unit as shown in the figure below.

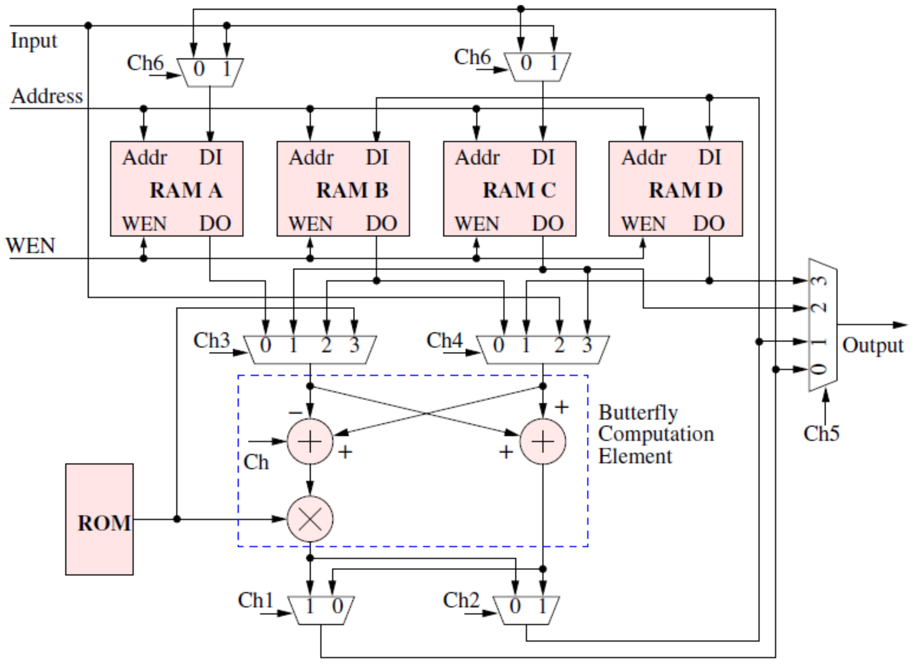


Figure 4.2.5.1-1: 16-point IFFT block Architecture

Data flow control is performed on this data path based on shown signal flow graph (SFG) below, input symbols from Resource Element Mapper (REM) are stored serially in memories in 16 cycles, REM only gives 12 symbols as mapped symbols and last 4 symbols of 16 are always zeros (un allocated bandwidth as NB-IoT has only one resource block 12 subcarrier of 15 KHz spacing) these zeros are encountered instead of output from REM at cycles from 13 to 16.

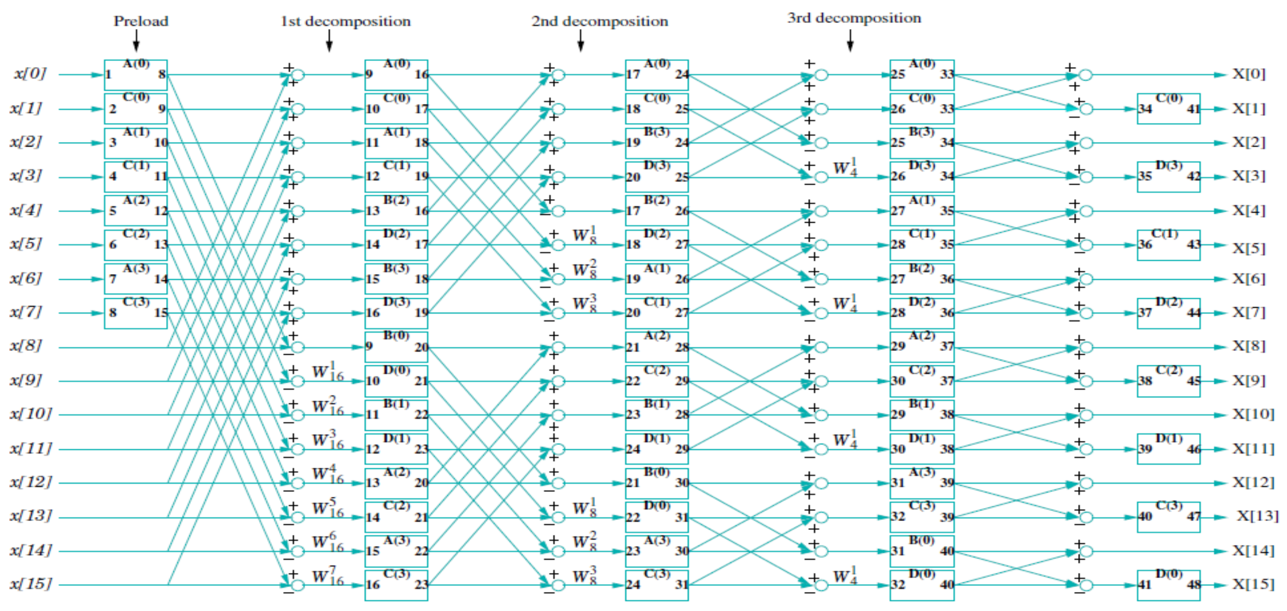


Figure 4.2.5.1-2: 16-point IFFT SFG

The Following Figure 4.2.5.1-3 shows the input and output ports of the IFFT Top level Module using the mentioned Hardware architecture and the Signal flow graph, and they are illustrated in Table 4.2.5.1-1.

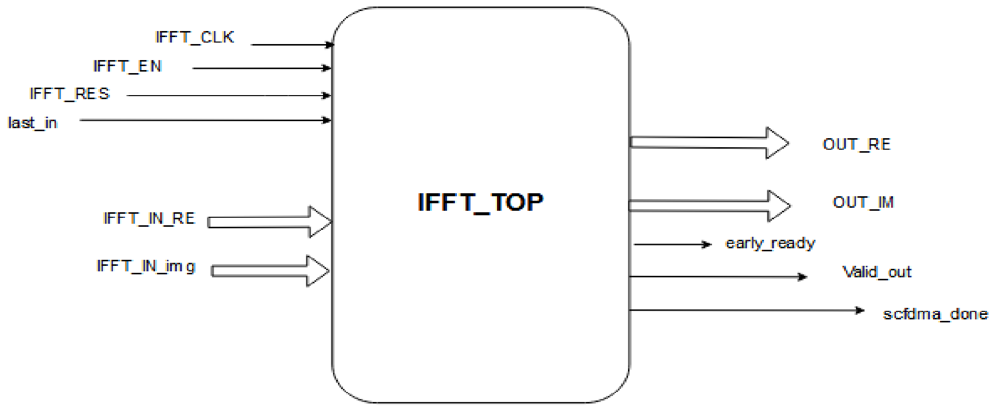


Figure 4.2.5.1-3: IFFT top level block Diagram with pins and interfaces

- **Block Interfaces:**

Table 4.2.5.1-1: IFFT Input and Output Ports' Description

Port name	Direction	Width	Description
IFFT_CLK	Input	1 bit	System clock
IFFT_EN	Input	1 bit	Block enable =1 by the first valid input to the IFFT from Resource Element Mapper and =0 by the end of transport block

			transmission.
IFFT_RES	Input	1 bit	Resets the block counters and outputs (by start of each transport block).
Last_in	Input	1 bit	Input signal from general control unit raised to 1 when the last SC-FDMA symbol is transferred from resource mapper to the IFFT block.
IFFT_IN_RE	Input	16 bits	Real part of input symbol =16 bits as agreed from SQNR simulation(6 bits integer + 10 bits fraction)
IFFT_IN_IM	Input	16 bits	imaginary part of input symbol =16 bits as agreed from SQNR simulation(6 bits integer + 10 bits fraction)
OUT_RE	Output	16 bits	real part of output sample =16 bits as agreed from SQNR simulation(6 bits integer + 10 bits fraction)
OUT_IM	Output	16 bits	imaginary part of output sample =16 bits as agreed from SQNR simulation(6 bits integer + 10 bits fraction)
Early_ready	Output	1 bit	Output signal to general control unit raised to 1 from cycles 54 to 11 at each IFFT iteration to enable the output of resource element mapper to enter the IFFT block from cycles 1 to 12.
Valid_out	Output	1 bit	Output signal raised to 1 by the start of first valid out signal (i.e. after the first iteration of IFFT since the enable setting to 1) and remains 1 till the end of transport block transmission.
Sc_fdma done	Output	1 bit	Output signal to the general control unit raised to 1 when the final SC-FDMA symbol is completely transmitted to let the general control unit disables IFFT block.

4.2.5.2. Operation

IFFT operation is as discussed in the FFT section 4.2.3 using nearly the same hardware and following the SFG shown in figure 4.2.5.1-2 for non-destructive memory storage for the intermediate values from the butterfly unit. Except for the IFFT that it is only 16-point not a mixed radix, so it’s a little bit simpler.

4.2.5.3. Challenges and Ways of Managing them

Constant Output Rate and Cyclic prefix insertion, One challenging issue of the IFFT block at transmitter of SC-FDMA is being at the end the chain that's why it must satisfy the required output rate (i.e new SC-FDMA symbol each SC-FDMA symbol duration) not also produce this symbol in this duration but also with a constant rate without any bubbles. The problem in the used design as shown in the above SFG is that it produces output samples in a burst style (i.e from cycle 33 to 48) only and the rest 32 cycles have no valid output.

One simple and straight forward solution in such cases is to use a buffer with different read and write rates and ensure that read rate from the buffer satisfies the constant output rate required while writing in it could be in a burst style as it is. This is the followed technique in our case with external control unit to control the read and write process and insert the cyclic prefix properly. Figure 4.2.5.2-1 and 4.2.5.2-1 shows the block diagram with its 3 blocks (IFFT, Control unit and buffer) and describes how reading and writing from and into the buffer occurs versus time cycles respectively.

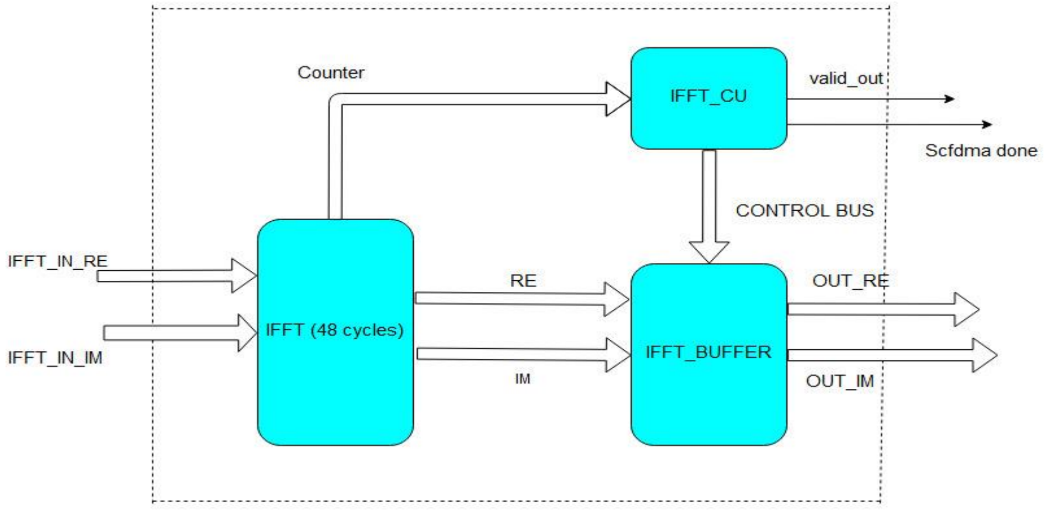


Figure 4.2.5.2-1: Constant Rate Controller, Buffer and IFFT

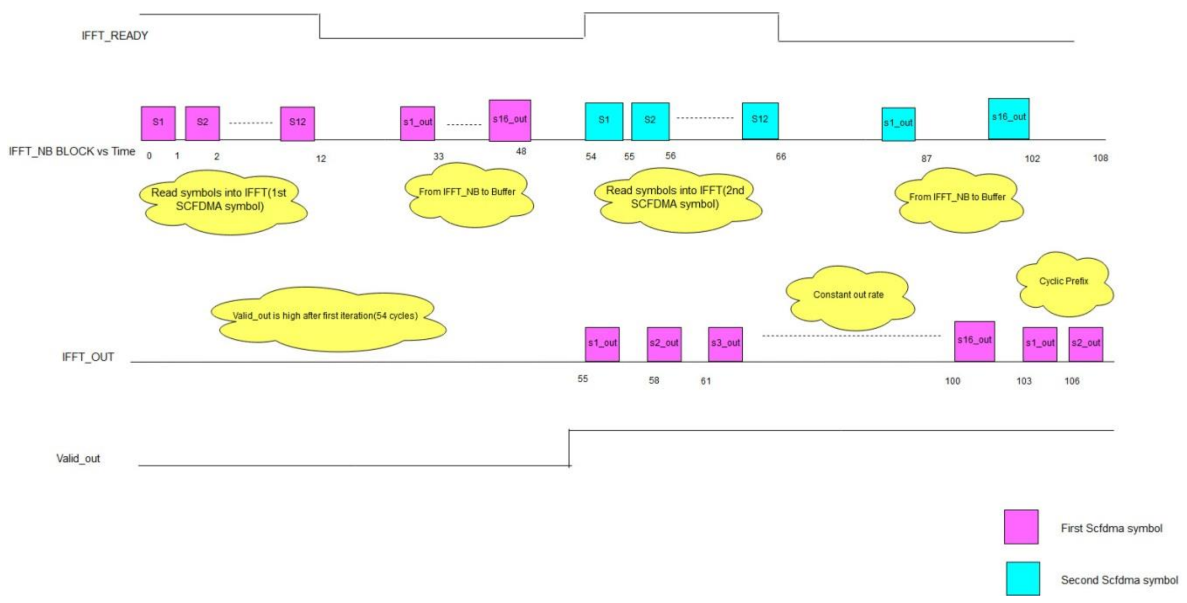


Figure 4.2.5.2-2: IFFT Timing diagram after controlling the output rate

As shown in figure 4.2.5.2-2 writing in the buffer from IFFT block occurs at cycles from 33 to 48 in a burst style then these symbols are read from buffer by a constant rate distributed over the next 54 cycle (this number of cycles is justified in Chapter 7), control unit handles all the destructive overwrite that could happen by sending the appropriate read and write enables and addresses depending on the value of cycles counter.

Chapter 5

Results and Simulation

5. Results and Simulation

In this chapter, Modelsim simulation results is introduced, comparing it with the built MATLAB model for each block to verify our design. Each design is tested individually with different test cases as well as testing the whole integrated chain in RTL Vs the integrated one in MATLAB.

5.1. Channel coding, multiplexing and interleaving

5.1.1. Cyclic Redundancy Check (CRC)

5.1.1.1. Test Case 1

For a TBS = 424 bits test case, The 24 CRC output bits are as shown in sample #1 and sample #2:

→ *Sample #1*

As a comparison for the same test case in both MATLAB model and RTL design, showing an output sample of the CRC which are the first 12 bits. They are the same as seen in the following Figures 5.1.1.1-1 and 5.1.1.1-2.

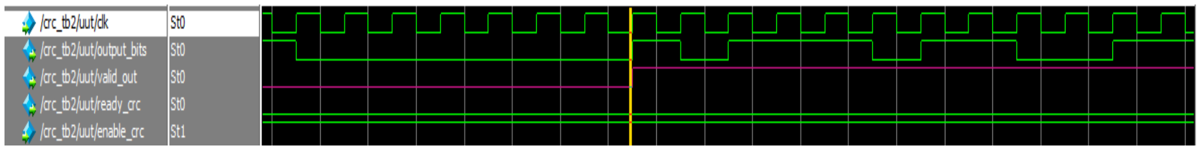


Figure 5.1.1.1-1: CRC RTL Simulation waveform for sample #1 in test case 1

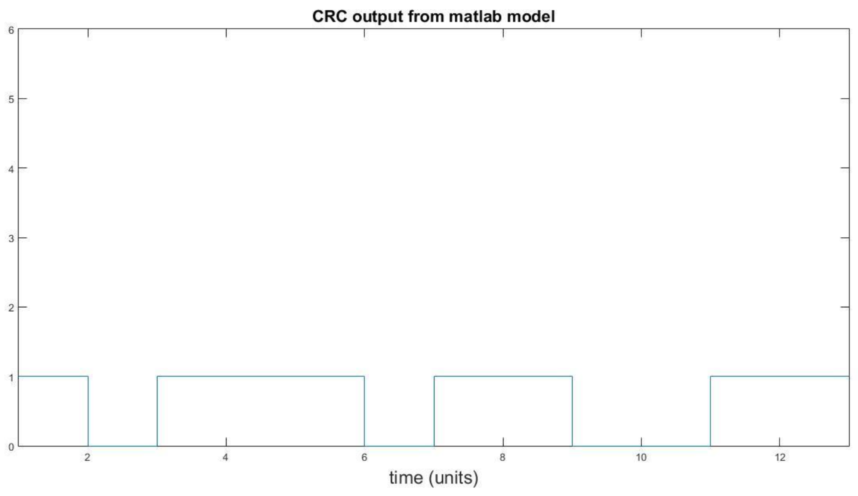


Figure 5.1.1.1-2: CRC MATLAB Model output for sample #1 in test case 1

→ *Sample #2*

Showing another CRC output sample for the same test case which are the last 16 bits, this sample is equal in Modelsim and MATLAB as well as shown in the figures 5.1.1.1-3 and 5.1.1.1-4.

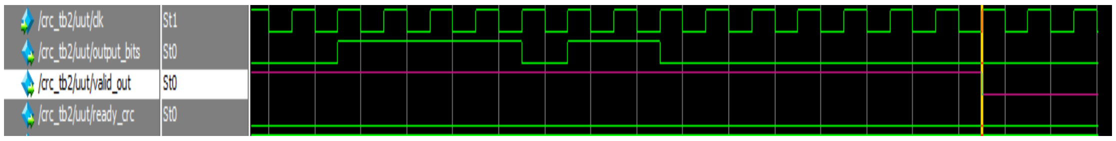

/crc_tb2/uut/clk St1
/crc_tb2/uut/output_bits St0
/crc_tb2/uut/valid_out St0
/crc_tb2/uut/ready_crc St0

Figure 5.1.1.1-3: CRC RTL Simulation waveform for sample #2 in test case 1

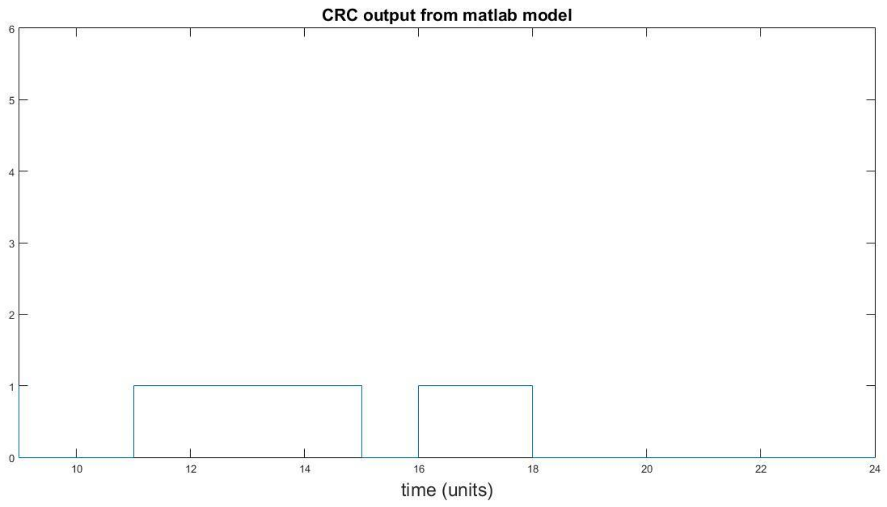


Figure 5.1.1.1-4: CRC MATLAB Model output for sample #2 in test case 1

When the valid out signal becomes '1', the encoder will read the 24 CRC Output bits after reading the Transport.

5.1.2. Turbo Encoder

5.1.2.1. Test Case 1

For a TBS = 16 bits test case, the Turbo Encoder output bits are as shown in sample #1 and sample #2:

→ *Sample #1*

As a comparison for the same test case in both MATLAB model and RTL design, showing an output sample of the Turbo Encoder which are the first 3 bits of the 3 output streams. They are the matched as shown in the following Figures 5.1.2.1-1 and 5.1.2.1-2.

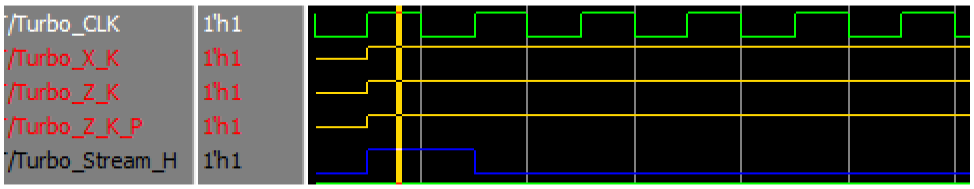
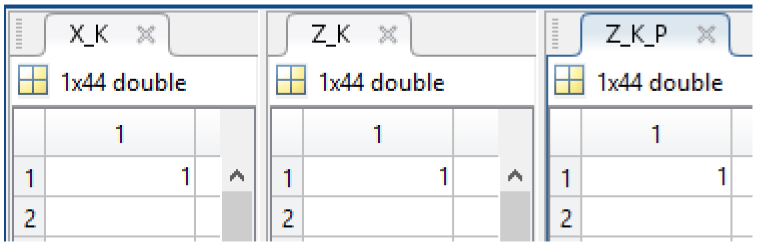


Figure 5.1.2.1-1: Turbo Encoder RTL Simulation waveform for sample #1 in test case 1



X_K	1
1	1
2	

Z_K	1
1	1
2	

Z_K_P	1
1	1
2	

Figure 5.1.2.1-2: Turbo Encoder MATLAB Model output for sample #1 in test case 1

→ *Sample #2*

The last 4 bits in the 3 output streams which are the trellis termination bits of the turbo encoder, figure 5.1.2.1-3 represent the RTL simulation waveform and figure 5.1.2.1-4 represent The MATLAB Model results which both figures are results matched as

X_K stream is terminated by 0 1 0 0

Z_K stream and Z_K_P are terminated by 1 0 0 0

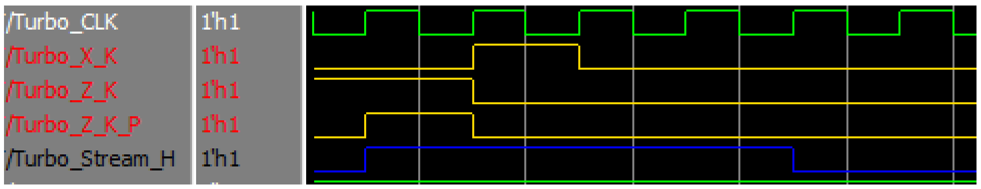
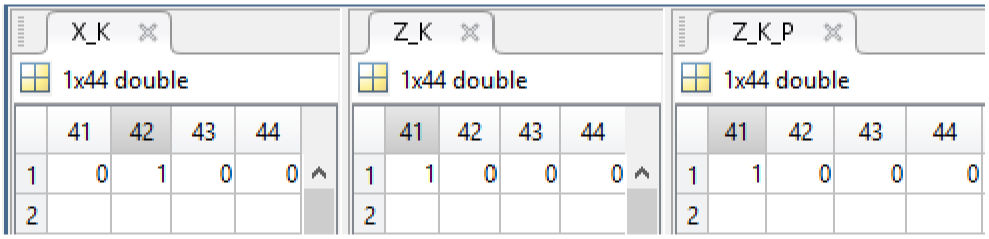


Figure 5.1.2.1-3: Turbo Encoder RTL Simulation waveform for sample #2 in test case 1



X_K

1x44 double

	41	42	43	44
1	0	1	0	0
2				

Z_K

1x44 double

	41	42	43	44
1	1	0	0	0
2				

Z_K_P

1x44 double

	41	42	43	44
1	1	0	0	0
2				

Figure 5.1.2.1-4: Turbo Encodr MATLAB Model output for sample #2 in test case 1

5.1.3. Rate Matching

5.1.3.1. Test Case 1

To verify RM RTL design, using our MATLAB model for the following parameters:

TBS = 32
G = 150
Redundancy version index = 0
Qm = 1 "BPSK Modulation Scheme"

Comparing the MATLAB and RTL results in figure 5.1.3.1-1 and 5.1.3.1-2 for the first 36 output bits for same input shows that the two results are the same.

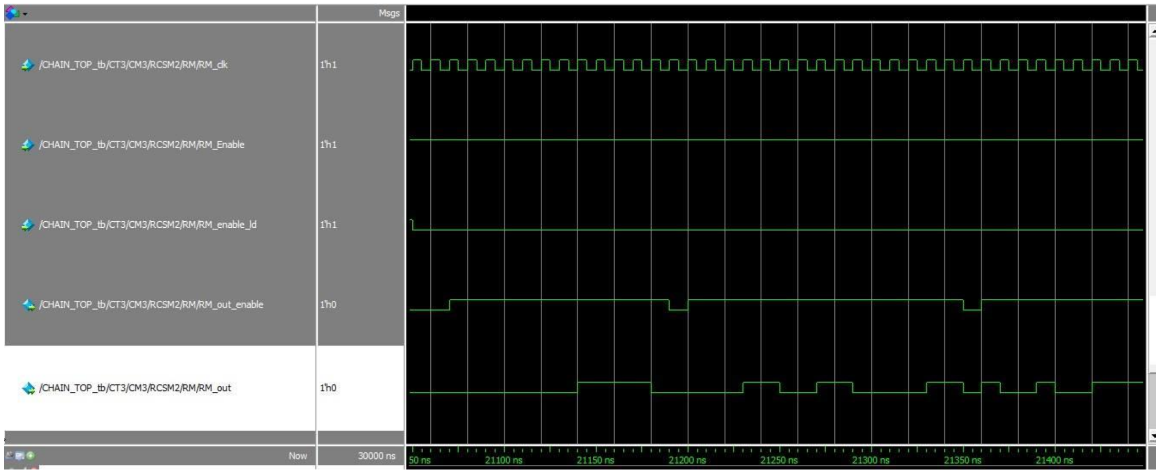

Msgs
/CHAIN_TOP_tb/CT3/CM3/RCSM2/RM/RM_clk 1'h1
/CHAIN_TOP_tb/CT3/CM3/RCSM2/RM/RM_Enable 1'h1
/CHAIN_TOP_tb/CT3/CM3/RCSM2/RM/RM_enable_ld 1'h1
/CHAIN_TOP_tb/CT3/CM3/RCSM2/RM/RM_out_enable 1'h0
/CHAIN_TOP_tb/CT3/CM3/RCSM2/RM/RM_out 1'h0
Now 30000 ns
50 ns
21100 ns
21150 ns
21200 ns
21250 ns
21300 ns
21350 ns
21400 ns

Figure 5.1.3.1-1: Rate Matching RTL Simulation waveform for test case 1

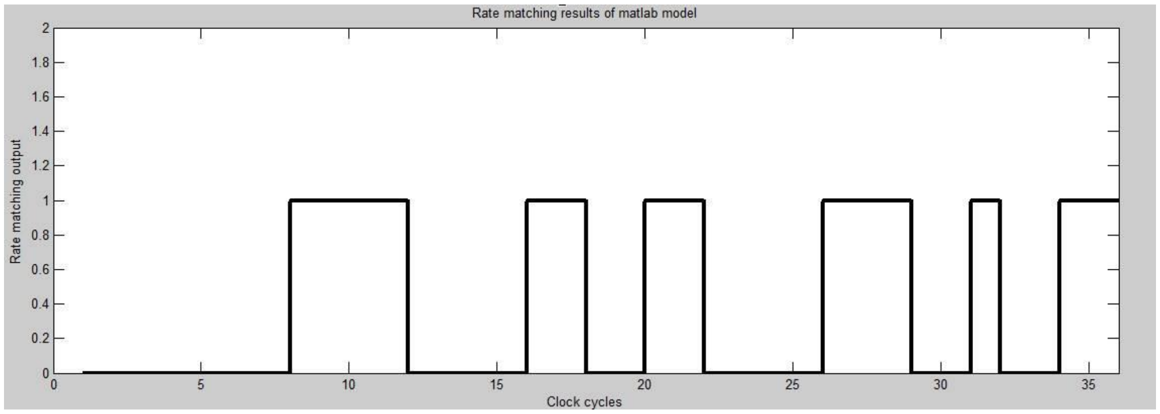


Figure 5.1.3.1-2: Rate Matching MATLAB model output for test case 1

As shown in figure 5.1.3.1-1, valid output bit from the Rate matching is flagged using "RM_out_enable" signal. This signal is activate (HIGH) at the valid output stream and inactive (LOW) at reading the dummy bits from the interleavers which are invalid bits.

5.1.4. Data Multiplexing and Channel interleaver

5.1.4.1. Test Case 1

Using the MATLAB model according to the standard description and our assumptions, we have used it to generate our test cases to verify our RTL design and here is one of our test cases for BPSK case, Nslots=8 which is mapped to 48 columns and E=100 bits:

Showing an output sample of the Data Multiplexing and Channel interleaving block which are the first 40 bits of the output stream, they are the matched as shown in the following Figures 5.1.4.1-1 and 5.1.4.1-2.

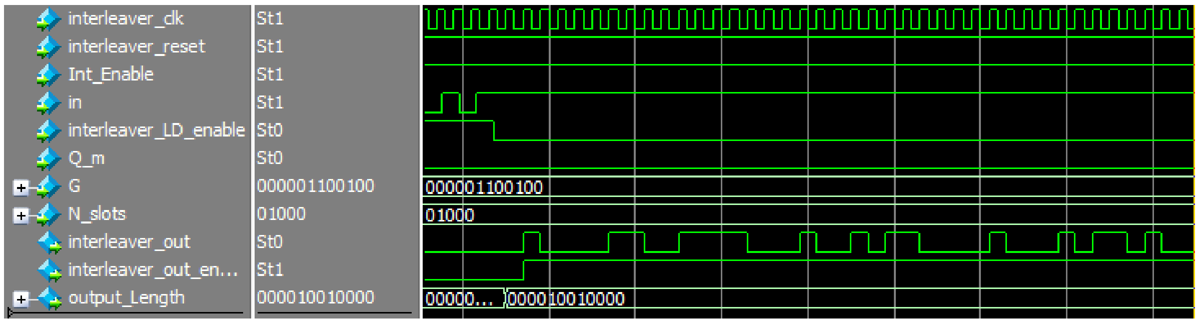


Figure 5.1.4.1-1: Data Multiplexing and Channel Interleaver RTL Simulation waveform for test case 1

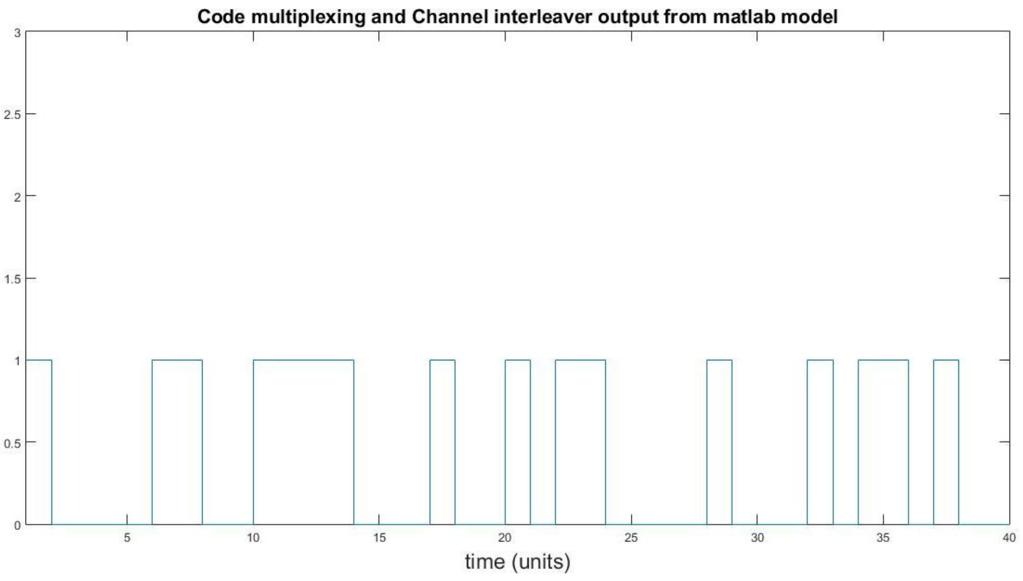


Figure 5.1.4.1-2: Data Multiplexing and Channel Interleaver MATLAB Model output for test case 1

As shown in figure 5.1.4.1-2, the valid output bits from the Channel interleaver is indicated by the interleaver_out_enable signal to be HIGH.

5.2. Physical Channel and modulation

5.2.1. Scrambler

5.2.1.1. Test Case 1

This test case to verify the Scrambler RTL design, using our MATLAB model for the following parameters:

RNTI = 65535
Ncell_ID = 504
nf = 1
ns = 1

Showing an output sample of the Scrambler block which are the first 25 bits of the output stream. They are the matched as shown in the following Figures 5.2.1.1-1 and 5.2.1.1-2.

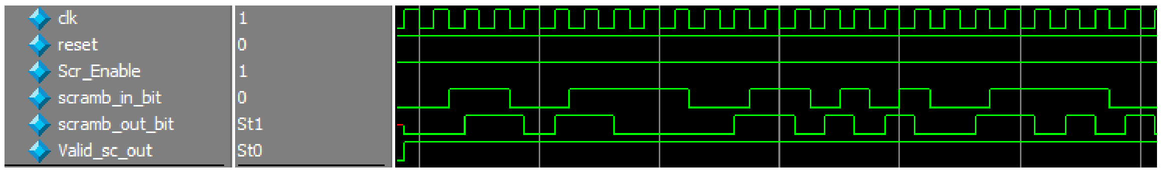


Figure 5.2.1.1-1: Scrambler RTL Simulation waveform for test case 1

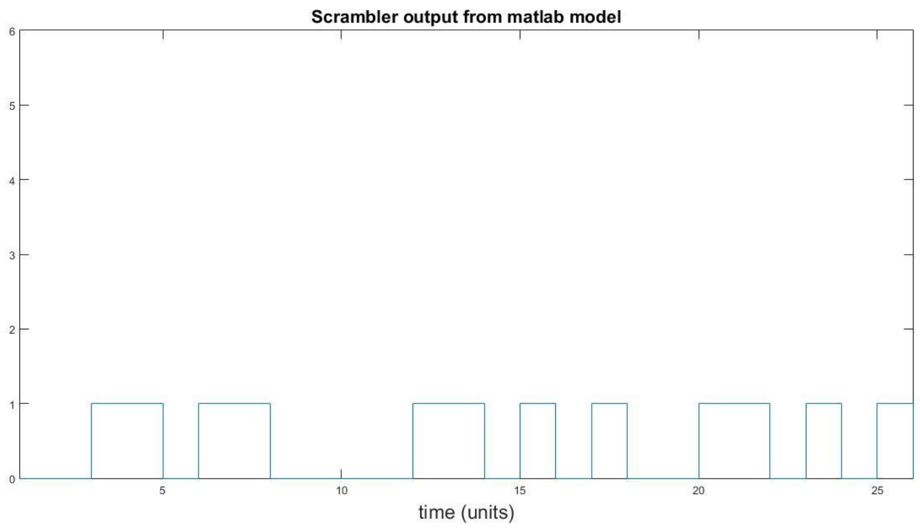


Figure 5.2.1.1-2: Scrambler MATLAB Model output for test case 1

As shown in figures 5.2.1.1-1, the valid output bit from the Scrambler is indicated by the Valid_sc_out signal to be HIGH.

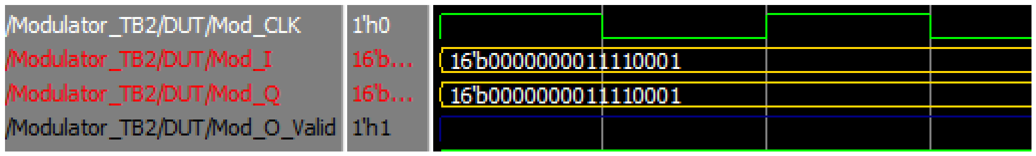
5.2.2. Modulation Mapper

Using the MATLAB model according to the standard description and our assumptions to verify our RTL design, for the Modulation Mapper three test cases are introduced with different inputs and Modulation schemes:

5.2.2.1. Test Case 1

Showing three output symbol of the Modulator block for a BPSK case, they are the matched as shown in the following Figures 5.2.2.1-1 and 5.2.2.1-2.

Modulator input = "000……"
Qm = 1 "BPSK Modulation scheme"
MSC= 3 " 3 subcarriers"



```
/Modulator_TB2/DUT/Mod_CLK       1'h0
/Modulator_TB2/DUT/Mod_I         16'b...   16'b0000000011110001
/Modulator_TB2/DUT/Mod_Q         16'b...   16'b0000000011110001
/Modulator_TB2/DUT/Mod_O_Valid   1'h1
```

Figure 5.2.2.1-1: Modulator RTL Simulation waveform for test case 1

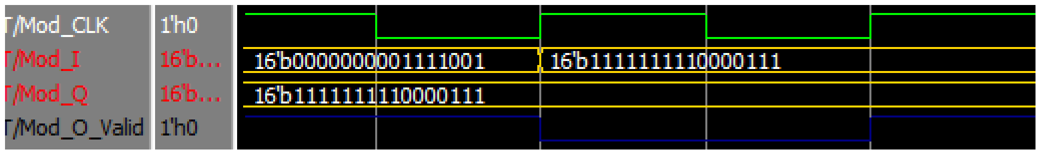
```
Mod_I= 0000000011110001
Mod_Q= 0000000011110001
```

Figure 5.2.2.1-2: Modulator MATLAB Model output for test case 1

5.2.2.2. Test Case 2

Showing one output symbol of the Modulator block for a QPSK case, they are the matched as shown in the following Figures 5.2.2.2-1 and 5.2.2.2-2.

Modulator input = "011111……….."
Qm = 2 "QPSK Modulation scheme"
MSC= 3 " 3 subcarriers"



```
T/Mod_CLK       1'h0
T/Mod_I         16'b...   16'b0000000001111001   16'b1111111110000111
T/Mod_Q         16'b...   16'b1111111110000111
T/Mod_O_Valid   1'h0
```

Figure 5.2.2.2-1: Modulator RTL Simulation waveform for test case 2

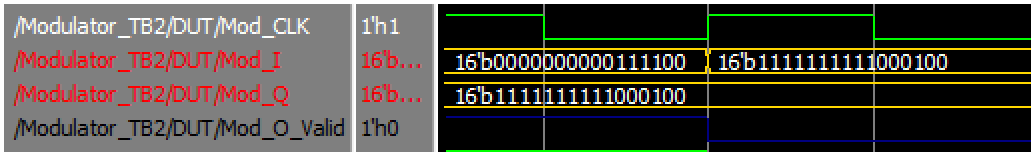
```
Mod_I= 0000000011110001
Mod_Q= 1111111100001111
```

Figure 5.2.2.2-2: Modulator MATLAB Model output for test case 2

5.2.2.3. Test Case 3

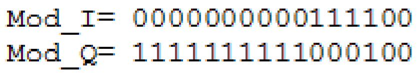
Showing one output symbol of the Modulator block for a QPSK case, they are the matched as shown in the following Figures 5.2.2.3-1 and 5.2.2.3-2.

Modulator input =0110100001…… TBS= 40
Qm = 2 "QPSK Modulation scheme"
MSC= 6 " 6 subcarriers"



```
/Modulator_TB2/DUT/Mod_CLK      1'h1
/Modulator_TB2/DUT/Mod_I        16'b...   16'b0000000000111100   16'b1111111111000100
/Modulator_TB2/DUT/Mod_Q        16'b...   16'b1111111111000100
/Modulator_TB2/DUT/Mod_O_Valid  1'h0
```

Figure 5.2.2.3-1: Modulator RTL Simulation waveform for test case 3



```
Mod_I= 0000000000111100
Mod_Q= 1111111111000100
```

Figure 5.2.2.3-2: Modulator MATLAB Model output for test case 3

As shown in the RTL simulation figures, the valid output from the Modulator is indicated by the Mod_O_Valid signal to be HIGH and it's LOW (inactive) otherwise.

5.2.3. FFT

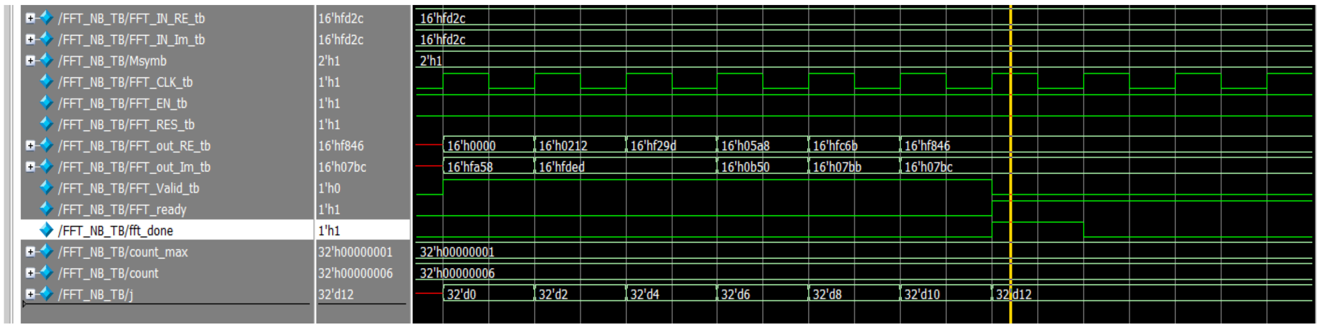
After building the MATLAB model according to our assumptions to verify the RTL Design and using the FFT MATLAB function as well, a test case will be introduced to compare the RTL results with the MATLAB.

5.2.3.1. Test Case 1

The following assumptions were used when implementing the FFT model:

- Number of fraction bits in results = 10
- Number of integer bits in results = 6
- Number of fraction bits in twiddle values = 14
- Number of integer bits in twiddle values = 2

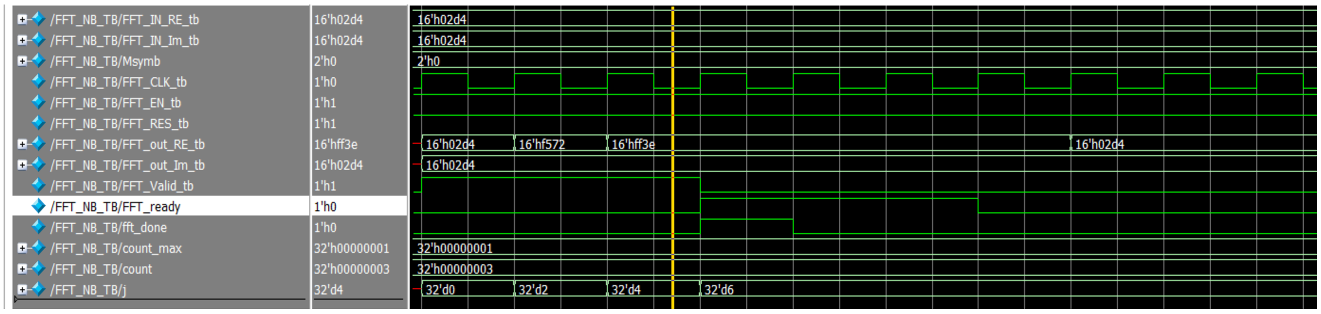
Figure 5.2.3.1-1, 5.2.3.1-2, 5.2.3.1-3 shows the timing diagrams of 3,6,12 point FFT results respectively. As shown the valid out signal is generated with the output. The FFT block produces a ready signal when it's ready for receiving a new input data and a done signal to show that it has finished its task.



/FFT_NB_TB/FFT_IN_RE_tb	16'hfd2c
/FFT_NB_TB/FFT_IN_Im_tb	16'hfd2c
/FFT_NB_TB/Msymb	2'h1
/FFT_NB_TB/FFT_CLK_tb	1'h1
/FFT_NB_TB/FFT_EN_tb	1'h1
/FFT_NB_TB/FFT_RES_tb	1'h1
/FFT_NB_TB/FFT_out_RE_tb	16'hf846
/FFT_NB_TB/FFT_out_Im_tb	16'h07bc
/FFT_NB_TB/FFT_Valid_tb	1'h0
/FFT_NB_TB/FFT_ready	1'h1
/FFT_NB_TB/fft_done	1'h1
/FFT_NB_TB/count_max	32'h00000001
/FFT_NB_TB/count	32'h00000006
/FFT_NB_TB/j	32'd12

16'hfd2c
16'hfd2c
2'h1
16'h0000 16'h0212 16'hf29d 16'h05a8 16'hfc6b 16'hf846
16'hfa58 16'hfded 16'h0b50 16'h07bb 16'h07bc
32'h00000001
32'h00000006
32'd0 32'd2 32'd4 32'd6 32'd8 32'd10 32'd12

Figure 5.2.3.1-1: The RTL waveform for 3-point FFT for test case 1



/FFT_NB_TB/FFT_IN_RE_tb	16'h02d4
/FFT_NB_TB/FFT_IN_Im_tb	16'h02d4
/FFT_NB_TB/Msymb	2'h0
/FFT_NB_TB/FFT_CLK_tb	1'h0
/FFT_NB_TB/FFT_EN_tb	1'h1
/FFT_NB_TB/FFT_RES_tb	1'h1
/FFT_NB_TB/FFT_out_RE_tb	16'hff3e
/FFT_NB_TB/FFT_out_Im_tb	16'h02d4
/FFT_NB_TB/FFT_Valid_tb	1'h1
/FFT_NB_TB/FFT_ready	1'h0
/FFT_NB_TB/fft_done	1'h0
/FFT_NB_TB/count_max	32'h00000001
/FFT_NB_TB/count	32'h00000003
/FFT_NB_TB/j	32'd4

Figure 5.2.3.1-2: The RTL waveform for 6-point FFT for test case 1

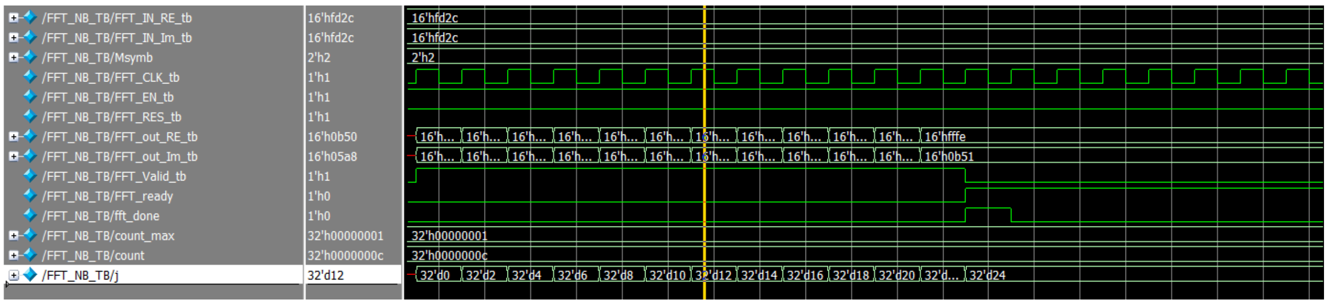

```
/FFT_NB_TB/FFT_IN_RE_tb      16'hfd2c
/FFT_NB_TB/FFT_IN_Im_tb      16'hfd2c
/FFT_NB_TB/Msymb             2'h2
/FFT_NB_TB/FFT_CLK_tb        1'h1
/FFT_NB_TB/FFT_EN_tb         1'h1
/FFT_NB_TB/FFT_RES_tb        1'h1
/FFT_NB_TB/FFT_out_RE_tb     16'h0b50
/FFT_NB_TB/FFT_out_Im_tb     16'h05a8
/FFT_NB_TB/FFT_Valid_tb      1'h1
/FFT_NB_TB/FFT_ready         1'h0
/FFT_NB_TB/fft_done          1'h0
/FFT_NB_TB/count_max         32'h00000001
/FFT_NB_TB/count             32'h0000000c
/FFT_NB_TB/j                 32'd12
```
16'hfd2c
16'hfd2c
2'h2
16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'hfffe
16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h... 16'h0b51
32'h00000001
32'h0000000c
32'd0 32'd2 32'd4 32'd6 32'd8 32'd10 32'd12 32'd14 32'd16 32'd18 32'd20 32'd... 32'd24

Figure 5.2.3.1-3: The RTL waveform for 12-point FFT for test case 1

Figure 5.2.3.1-4, 5.2.3.1-5 and 5.2.3.1-6 shows the output results from simulating the FFT compared to the expected MATLAB results. As shown the results are almost the same and may differ in the least significant bits only due to bits truncation while modeling the FFT and using a limited number of fraction bits.

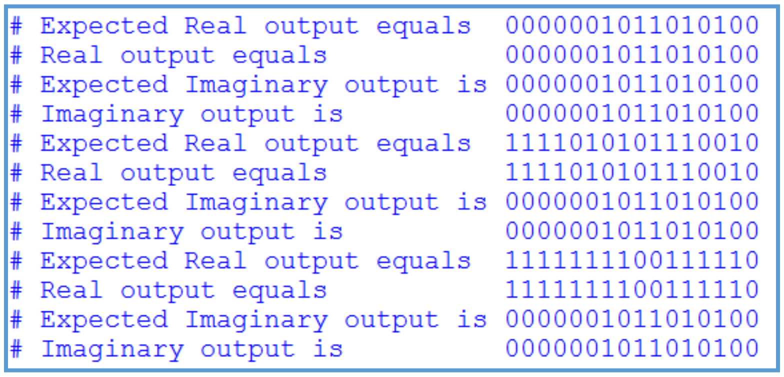
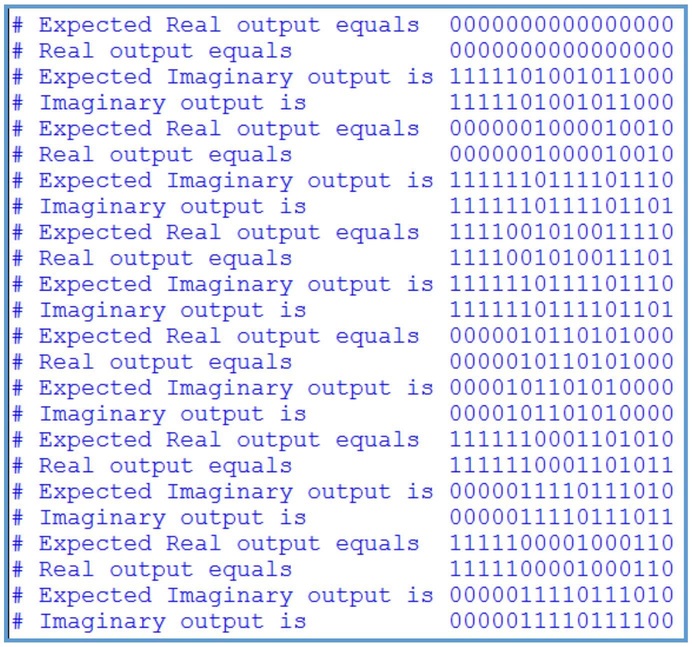
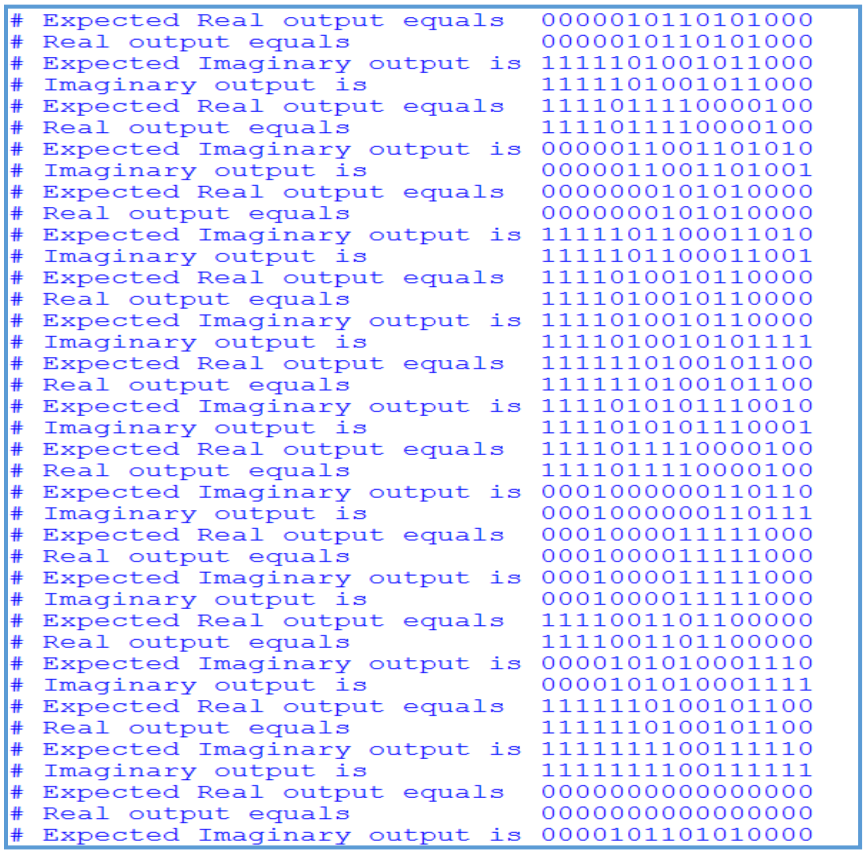

```
# Expected Real output equals   0000001011010100
# Real output equals            0000001011010100
# Expected Imaginary output is  0000001011010100
# Imaginary output is           0000001011010100
# Expected Real output equals   1111010101110010
# Real output equals            1111010101110010
# Expected Imaginary output is  0000001011010100
# Imaginary output is           0000001011010100
# Expected Real output equals   1111111100111110
# Real output equals            1111111100111110
# Expected Imaginary output is  0000001011010100
# Imaginary output is           0000001011010100
```

Figure 5.2.3.1-4: RTL results compared with the MATLAB expexted ones for 3-point FFT for test case 1



```
# Expected Real output equals  0000000000000000
# Real output equals           0000000000000000
# Expected Imaginary output is 1111101001011000
# Imaginary output is          1111101001011000
# Expected Real output equals  0000001000010010
# Real output equals           0000001000010010
# Expected Imaginary output is 1111110111101110
# Imaginary output is          1111110111101101
# Expected Real output equals  1111001010011110
# Real output equals           1111001010011101
# Expected Imaginary output is 1111110111101110
# Imaginary output is          1111110111101101
# Expected Real output equals  0000010110101000
# Real output equals           0000010110101000
# Expected Imaginary output is 0000101101010000
# Imaginary output is          0000101101010000
# Expected Real output equals  1111110001101010
# Real output equals           1111110001101011
# Expected Imaginary output is 0000011110111010
# Imaginary output is          0000011110111011
# Expected Real output equals  1111100001000110
# Real output equals           1111100001000110
# Expected Imaginary output is 0000011110111010
# Imaginary output is          0000011110111100
```

Figure 5.2.3.1-5: RTL results compared with the MATLAB expexted ones for 6-point FFT for test case 1



```
# Expected Real output equals  0000010110101000
# Real output equals           0000010110101000
# Expected Imaginary output is 1111101001011000
# Imaginary output is          1111101001011000
# Expected Real output equals  1111011110000100
# Real output equals           1111011110000100
# Expected Imaginary output is 0000011001101010
# Imaginary output is          0000011001101001
# Expected Real output equals  0000000101010000
# Real output equals           0000000101010000
# Expected Imaginary output is 1111101100011010
# Imaginary output is          1111101100011001
# Expected Real output equals  1111010010110000
# Real output equals           1111010010110000
# Expected Imaginary output is 1111010010110000
# Imaginary output is          1111010010101111
# Expected Real output equals  1111110100101100
# Real output equals           1111110100101100
# Expected Imaginary output is 1111010101110010
# Imaginary output is          1111010101110001
# Expected Real output equals  1111011110000100
# Real output equals           1111011110000100
# Expected Imaginary output is 0001000000110110
# Imaginary output is          0001000000110111
# Expected Real output equals  0001000011111000
# Real output equals           0001000011111000
# Expected Imaginary output is 0001000011111000
# Imaginary output is          0001000011111000
# Expected Real output equals  1111001101100000
# Real output equals           1111001101100000
# Expected Imaginary output is 0000101010001110
# Imaginary output is          0000101010001111
# Expected Real output equals  1111110100101100
# Real output equals           1111110100101100
# Expected Imaginary output is 1111111100111110
# Imaginary output is          1111111100111111
# Expected Real output equals  0000000000000000
# Real output equals           0000000000000000
# Expected Imaginary output is 0000101101010000
```

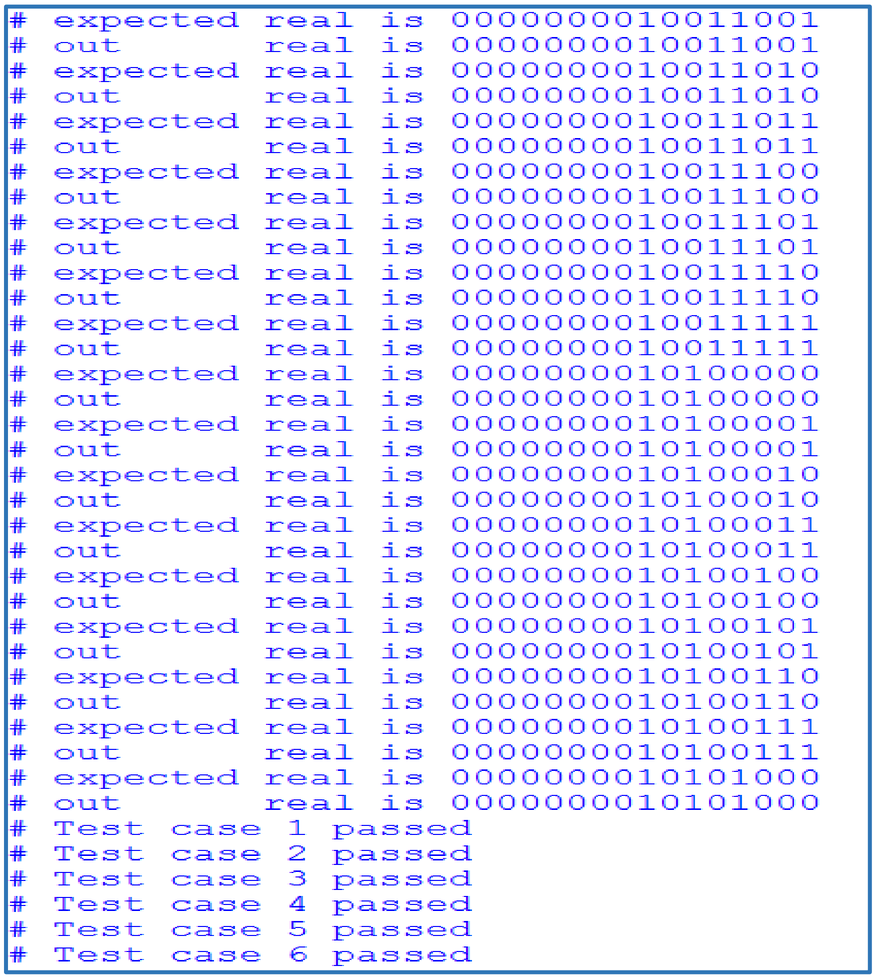
Figure 5.2.3.1-6: RTL results compared with the MATLAB expected ones for 12-point FFT for test case 1

5.2.4. <u>Resource element Mapper</u>

MATLAB results are compared with RTL test bench results for several cases:

a. **12 point FFT without repetitions**
b. **6 point FFT with repetitions and start index of 6**
c. **3 point FFT and start index of 3**
d. **3 point FFT and start index of 6**
e. **12 point FFT with repetitions**
f. **3 point FFT with repetitions and start index of 3**

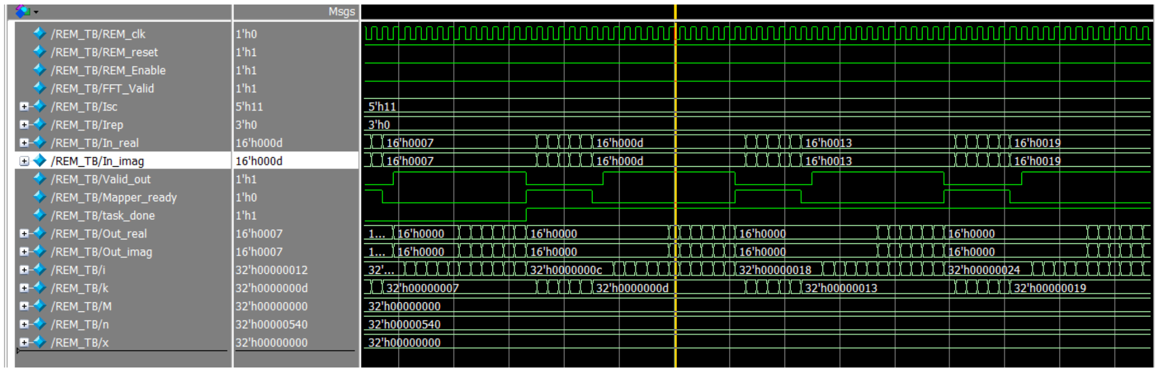
Figure 5.2.4-1 shows that both models' results have the same output.



```
# expected real is 0000000010011001
# out      real is 0000000010011001
# expected real is 0000000010011010
# out      real is 0000000010011010
# expected real is 0000000010011011
# out      real is 0000000010011011
# expected real is 0000000010011100
# out      real is 0000000010011100
# expected real is 0000000010011101
# out      real is 0000000010011101
# expected real is 0000000010011110
# out      real is 0000000010011110
# expected real is 0000000010011111
# out      real is 0000000010011111
# expected real is 0000000010100000
# out      real is 0000000010100000
# expected real is 0000000010100001
# out      real is 0000000010100001
# expected real is 0000000010100010
# out      real is 0000000010100010
# expected real is 0000000010100011
# out      real is 0000000010100011
# expected real is 0000000010100100
# out      real is 0000000010100100
# expected real is 0000000010100101
# out      real is 0000000010100101
# expected real is 0000000010100110
# out      real is 0000000010100110
# expected real is 0000000010100111
# out      real is 0000000010100111
# expected real is 0000000010101000
# out      real is 0000000010101000
# Test case 1 passed
# Test case 2 passed
# Test case 3 passed
# Test case 4 passed
# Test case 5 passed
# Test case 6 passed
```

Figure 5.2.4-1: REM MATLAB expected results compared with the output results

Figure 5.2.4-2 shows the timing diagram of REM simulation. As shown, the mapper ready signal becomes zero whenever a repetition starts to tell the control unit to disable the FFT block. It becomes one again after ending the repetitions. The valid out signal becomes high whenever the REM produces correct output.



	Msgs
/REM_TB/REM_clk	1'h0
/REM_TB/REM_reset	1'h1
/REM_TB/REM_Enable	1'h1
/REM_TB/FFT_Valid	1'h1
/REM_TB/Isc	5'h11
/REM_TB/Irep	3'h0
/REM_TB/In_real	16'h000d
/REM_TB/In_imag	16'h000d
/REM_TB/Valid_out	1'h1
/REM_TB/Mapper_ready	1'h0
/REM_TB/task_done	1'h1
/REM_TB/Out_real	16'h0007
/REM_TB/Out_imag	16'h0007
/REM_TB/i	32'h00000012
/REM_TB/k	32'h0000000d
/REM_TB/M	32'h00000000
/REM_TB/n	32'h00000540
/REM_TB/x	32'h00000000

Figure 5.2.4-2: REM Modelsim simulation Waveform showing it's Control signals

5.2.5. IFFT and CP

5.2.5.1. Test Case 1

Validation for this block is carried out by using test vectors out from the full MATLAB model of the chain before IFFT block, these vectors are encountered to the model of IFFT and the RTL implementation of it and compared using automated test bench to verify the output samples correctness with taking into consideration the Quantization noise in results due to truncation and fixed point representation after performing the SQNR simulation as illustrated at the end of this Chapter, also rate of output is taken into consideration in the test bench where the RTL design gives a new output samples each 3 clock cycles with a total number of samples = 18 (16 + 2 cp) distributed on the 54 cycle of each IFFT iteration.

As shown in figures 5.2.5.1-1 and 5.2.5.1-2, IFFT has its ready signal for new input symbols from cycle 1 to 12 preceded by early_ready port by 1 cycle and valid out =1 starting from the first valid out symbol i.e. after the first IFFT iteration (54 cycles) as discussed before in chapter 4.

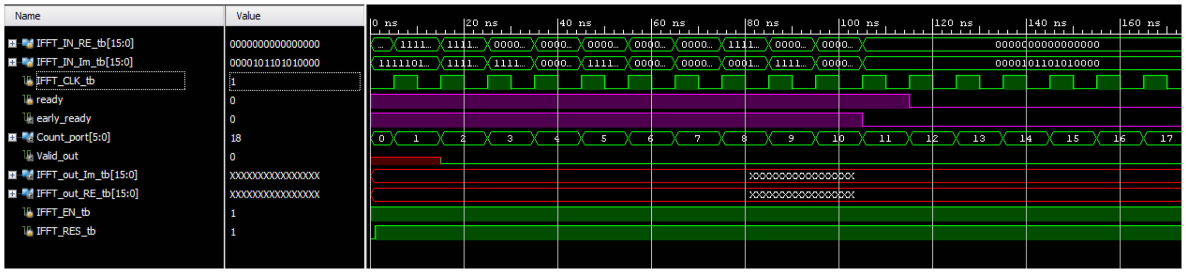


Figure 5.2.5.1-1: The RTL waveform for 16-point IFFT for test case 1 showing the Control signals

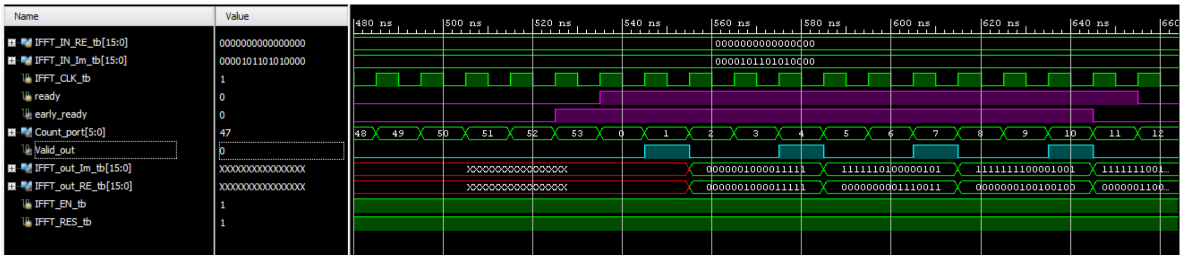


Figure 5.2.5.1-2 The RTL waveform for 16-point IFFT for test case 1 showing the constant Valid output rate

To ensure the correctness of output symbols from RTL the methodology of testing shown in figure 5.2.5.1-3 is followed.

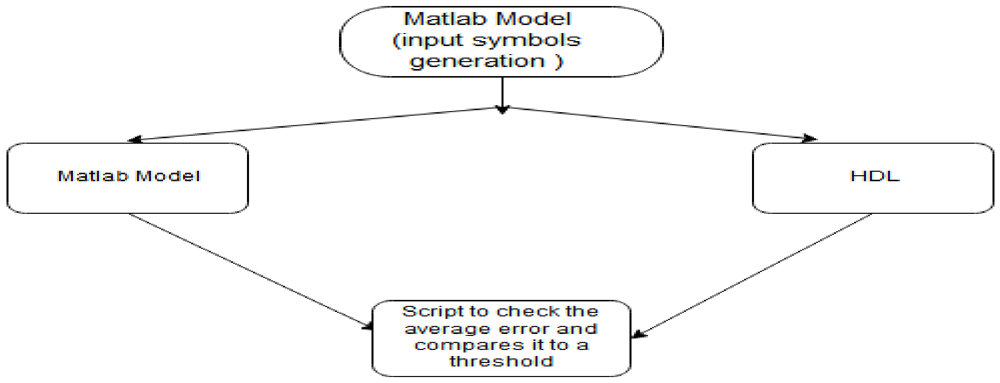

Matlab Model (input symbols generation)
Matlab Model
HDL
Script to check the average error and compares it to a threshold

Figure 5.2.5.1-3: IFFT testing framework process

5.3. SQNR and PAPR

5.3.1. PAPR

The further increasing demand on high data rates in wireless communications systems has arisen in order to support broadband services.3GPP long term evolution (LTE) has adopted orthogonal frequency division multiplexing access (OFDMA) for downlink transmission and single carrier frequency division multiple access (SCFDMA) for uplink ;this is to compensate for a drawback with normal OFDMA, which has a very high peak to average power ratio (PAPR).High PAPR requires expensive and inefficient power amplifiers with high requirements on linearity, which increases the cost of the terminal and drains the battery faster. SCFDMA signal has lower PAPR because of its inherent single carrier structure.

The PAPR in SC-FDMA is required to be as small as possible. The PAPR is defined as the ratio of peak power to average power of the transmitted signal in a given transmission block and is given by

$$10\, log_{10}(\frac{\max(|x(m)|^2)}{\frac{1}{M}\sum_{m=0}^{M-1}|x(m)|^2})$$

5.3.1.1. Sub-carrier mapping techniques

If M is the number of subcarriers allocated to each user then, M point DFT is used for spreading purpose which will further be applied to the subcarriers of inverse DFT and the Way of assigning the subcarriers to each terminal is the main key factor of the PAPR reduction, there are three sub-carriers mapping schemes available for assigning M frequency domain symbols to the sub-carriers in SCFDMA. They are - localized, distributed and interleaved sub-carrier mapping. The distributed and localized subcarrier mapping of assigning the frequency domain M symbols to the N no. of sub-carriers are shown in figure 5.3.1.1-1.

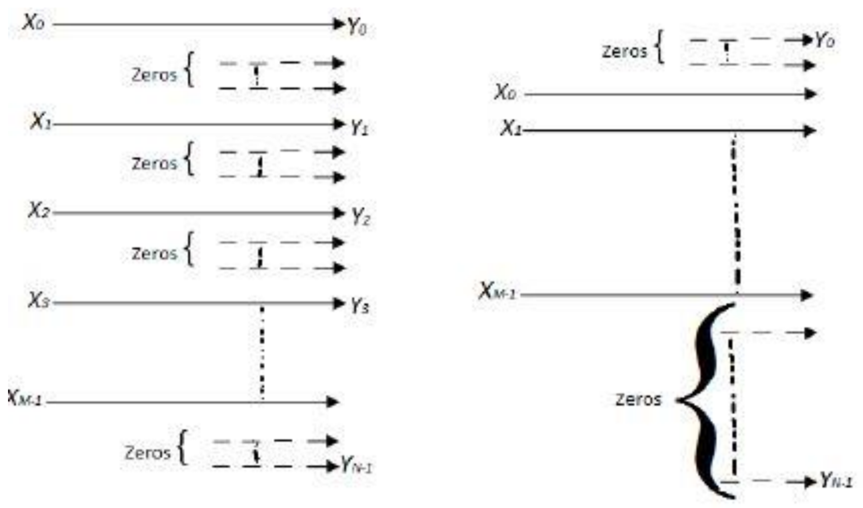


Figure 5.3.1.1-1: DFDMA (Distributed) and LFDMA (Localized)

The localized subcarrier mapping mode & distributed subcarrier mode of SCFDMA is commonly known as the LFDMA and DFDMA respectively. DFT outputs are allocated to M consecutive subcarrier in the total N numbers of subcarriers (where N>M) in LFDMA. On contrary in DFDMA, the M numbers of DFT outputs are distributed into the entire band. In both DFDMA and LFDMA zero amplitude is assigned to (N-M) unoccupied subcarriers. If the DFT outputs are distributed with an equidistance N/M=Q between the occupied subcarriers then the mapping mode is referred as the interleave FDMA (IFDMA), where *S* is named as bandwidth spreading factor [14-15]. For M=4 symbols per block, N=12 subcarriers and Q=N/M=12/4=3 terminals, the SCFDMA symbol transmission in frequency domain is shown in figure 5.3.1.1-2.

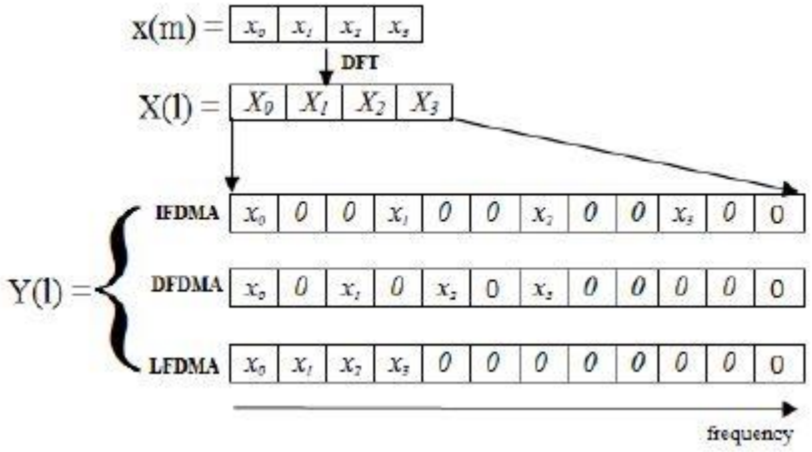

x(m) =
x_0 x_1 x_2 x_3
DFT
X(l) =
X_0 X_1 X_2 X_3
Y(l) =
IFDMA
x_0 0 0 x_1 0 0 x_2 0 0 x_3 0 0
DFDMA
x_0 0 x_1 0 x_2 0 x_3 0 0 0 0 0
LFDMA
x_0 x_1 x_2 x_3 0 0 0 0 0 0 0 0
frequency

Figure 5.3.1.1-2: Example of different subcarriers mapping modes

In order to verify the PAPR performance the mathematical equations are simulated in this section using MATLAB. The simulation is done by taking 542 different IFFT output for different input test vectors, the cumulative distribution function (CDF) of the PAPR is one of the most performance measures for different PAPR reduction techniques. Most researches use complementary CDF (CCDF) instead of CDF. The CCDF of the PAPR denotes the probability that the PAPR of a data block as shown in figure 5.3.1.1-3.

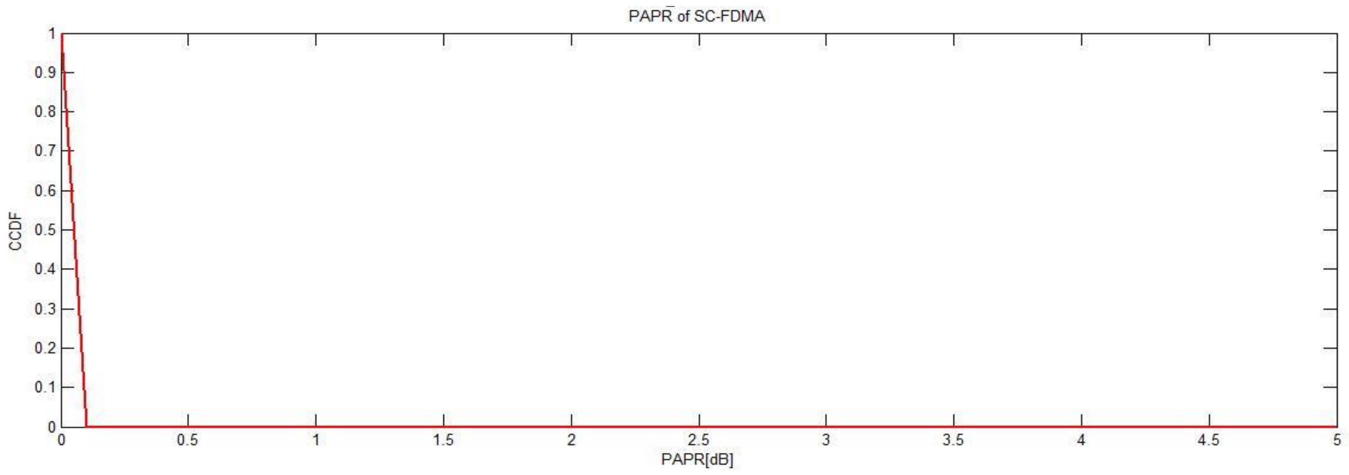


Figure 5.3.1.1-3: PAPR of SCFDMA system using QPSK modulation scheme

5.3.2. SQNR

Similar to the signal to noise ratio(SNR) another measure of performance called signal to quantization noise ratio(**SQNR**) is very important and it has a different source of noise on the values of the signals which is the truncation of word length into fixed number of bits, let's understand why there is a need for truncation mainly, mathematical operations could be performed on digital circuits using floating point representation or fixed point, the first representation method requires complex computational circuits in addition to larger utilization area that's why we need to use fixed point representation.

Fixed point representation means the location of fraction point in any represented number in this system is fixed and no need for extra number to denote its location unlike the floating point representation, to choose the location of fraction point wisely one need to pass through two steps : first, determine the integer part bits through the dynamic range of numbers resulted from floating point simulation and this number of bits (integer length) couldn't be minimized and if one tries to do so he is going to have completely wrong results.

Second step is the SQNR simulation to determine number of fraction bits (fraction length) value of this number could be considered as a soft limit i.e it could be minimized to decrease number of signals for example but accuracy of represented numbers will be decreased as a result. SQNR simulation is performed as shown in figure 5.3.2-1 and fraction length is chosen from graph based on standard specs after taking into consideration a safety factor for the expected decrease in SQNR through the RF chain which is beyond scope of this work.

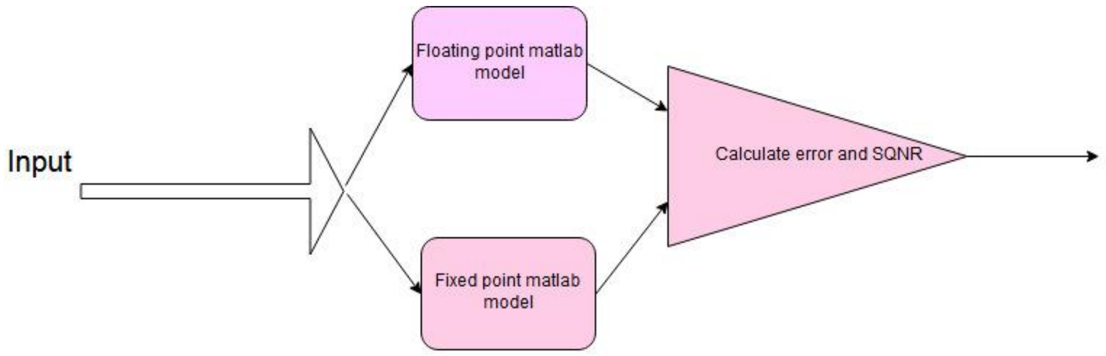


Fig.5.3.2-1 SQNR simulation framework process

Fraction length of 10 is chosen according to the standard specs on output SQNR and error vector magnitude (EVM) values as shown in figure 5.3.2-2. And integer length is chosen to be 6 after determining the dynamic range from floating point simulation.

Standard specs of 3GPP release 14 only puts upper limit on the value of EVM but this limit could be transformed to lower limit of SQNR as shown in this work [17].

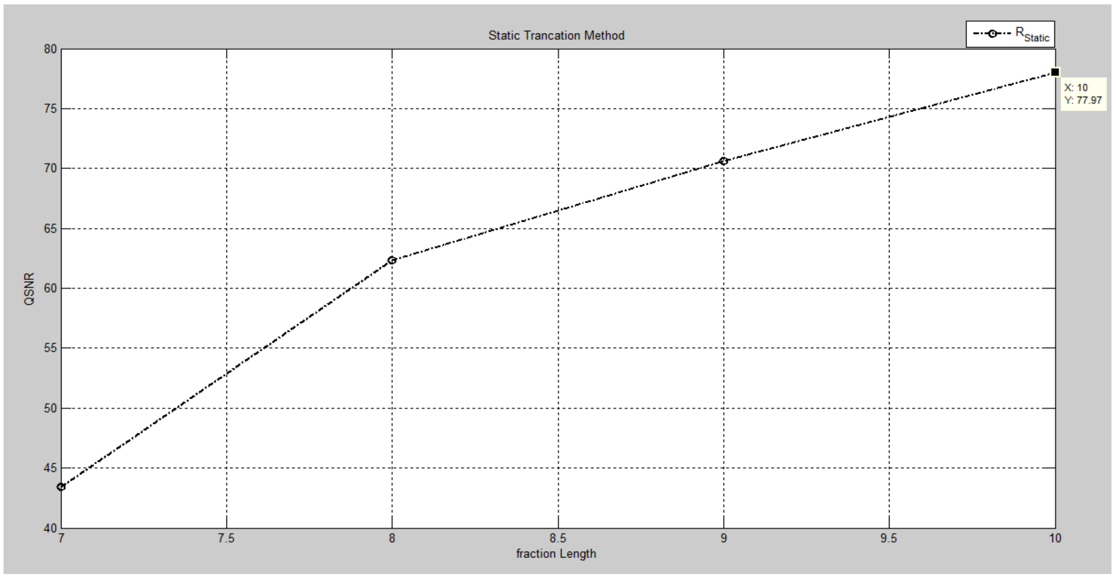


Fig.5..3.2-2 SQNR vs fraction length

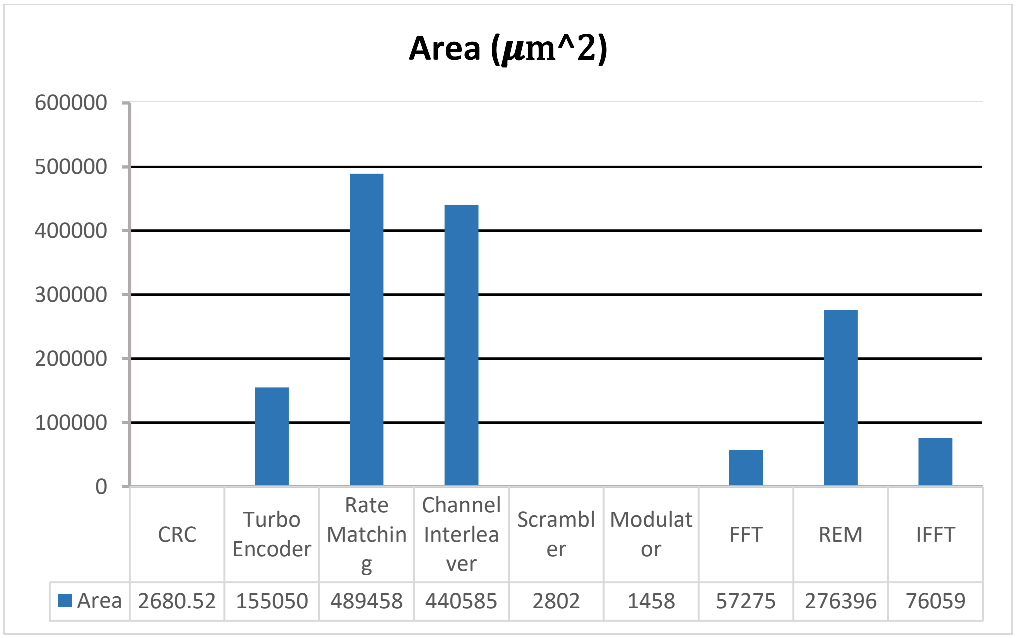
Chapter 6

Synthesis Results

6. Synthesis Results

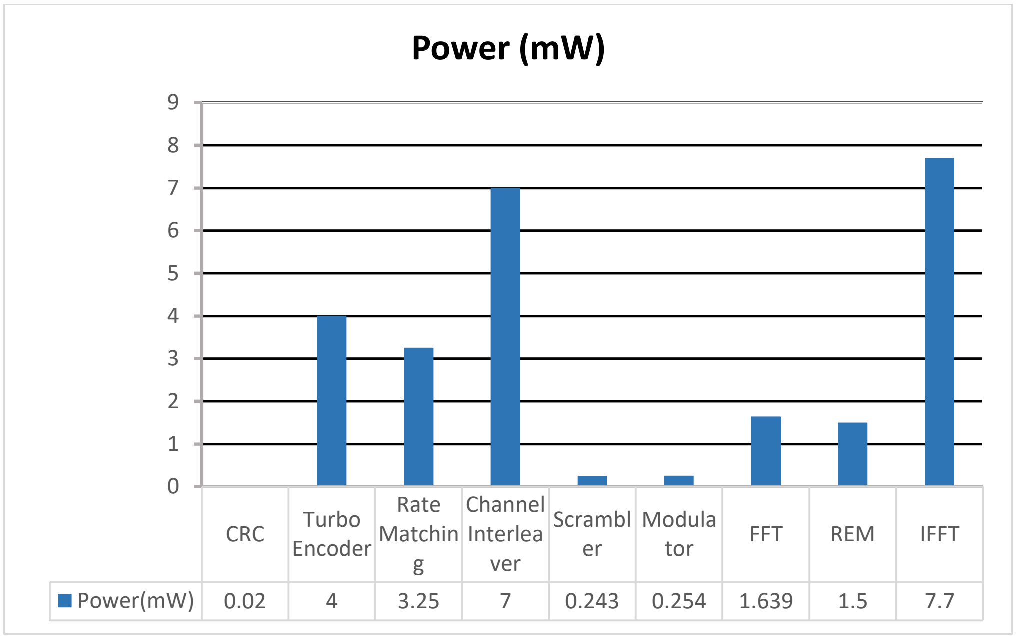
Illustrating in this chapter, the synthesis reports for individual blocks in the chain and the whole integrated chain, design compiler tool with UMC 130nm technology is used to generate power and area for designed chain after adding SAIF (Switching Activity Interchange Format) files to have more accurate dynamic power results.

All blocks and the full chain are synthesized using UMC 130 nm technology; results for area and power after logic synthesis are shown in figure 6-1 and 6-2 respectively followed by a brief discussion and justification for these results.



	CRC	Turbo Encoder	Rate Matching	Channel Interleaver	Scrambler	Modulator	FFT	REM	IFFT
■ Area	2680.52	155050	489458	440585	2802	1458	57275	276396	76059

Figure 6-1: Area results of all blocks in the chain



	CRC	Turbo Encoder	Rate Matching	Channel Interleaver	Scrambler	Modulator	FFT	REM	IFFT
■ Power(mW)	0.02	4	3.25	7	0.243	0.254	1.639	1.5	7.7

Figure 6-2: Power Results of All blocks in the Chain

As expected CRC and Scrambler has the lowest utilization area because they mainly consists of linear feedback shift registers (LFSR). Modulator also has a very low utilization area as it contains mainly one lookup table. One significant result too is for FFT and IFFT which shows relatively too low values for power and utilization area after using memory based architecture which is much more optimized from power and area point of view than the pipelined architecture as shown in figure 3. On the other hand Channel inter-leaver, resource element mapper (REM) and Encoder has the largest utilization area because they have large storage Elements. For the Turbo Encoder, a 1-Dimensional buffer is used and powered one third of the total time of the encoder activity. The Channel Interleaver, has four 2D memories which are gated but they are still large ones. The same applies for the resource element mapper.

Chapter 7

Full integrated Chain controlling and clock rates

7. Full integrated Chain Challenges and determining Clock rate

7.1. <u>Full integrated Chain Challenges</u>

All of the blocks discussed in chapter 4 are integrated to form the top level chain of the NB-IoT transmitter and to check the final design functionality after synthesizing it as a hierarchical design using UMC 130 nm technology.

Challenges of this task aren't in port mapping of the blocks or even writing a generic test bench that covers all modes of operation through encountering appropriate test vectors, but challenges originate from data flow control between blocks given that they have different output rate and difference in latencies of the blocks in the chain. These challenges present two problems; the first is how could the output rate still constant without any bubbles on the output port with taking difference in blocks' output rate into consideration and the second is how to disable the blocks which haven't received valid output yet because of variable latencies in the preceding blocks to optimize the design for low power consumption.

These two problems are expected before integration and techniques to solve them are studied carefully to choose the simplest and lowest in power consumption solutions.

For the first problem it's sufficient to ensure that last block in the chain (IFFT) always have valid and ready input symbols when needed (i.e each IFFT iteration= 54 cycles) and this is satisfied as illustrated in section clock rate determination section 7.2.

Second problem is eliminated using a general control unit that could enable and disable any block in the chain to save power and to power on specific block only when needed. Control Unit instantiation is shown in figure 7.1-1 below and its operation is as follows:

a. <u>CRC Enable</u>

CRC block is enabled by the start of transport block and disabled after processing the transport block and generates the cyclic redundancy bits i.e after (TBS + 24) cycles, where TBS is the transport block size and 24 is the CRC size according to the polynomial used in the standard.

b. Encoder

Encoder is enabled by the start of transport block and disabled after checking encoder_done flag i.e after the encoder finishes processing the input stream.

c. Rate Matching

Rate matching block is enabled by the first valid input to it from encoder i.e. after checking encoder_valid out flag, and disabled after sending the last bit in transport block i.e. after checking the rate match_done flag.

d. Channel interleaver (CI)

CI enable is triggered by rate matching_valid out positive edge, then it's disabled by one of two reasons; first by the flow control buffer after the modulator in case there is no need for new inputs to the FFT block in this case remaining bits of transport block are kept in the repository (CI), second reason is by the general control unit at the end of transport block.

e. Scrambler

Scrambler is enabled by the start of transport block (i.e by the first bit of transport block entering the CRC) to give this block enough time for initialization, and it's disabled one cycle after disabling the CI to enable the last traveling bit form CI to reach the modulator.

f. Modulator

Modulator enable is triggered by Scrambler_valid out positive edge, and disabled with disabling the scrambler –when its valid out is inactive- as scrambler's valid out signal is inactivated afted getting the last bit out by one cycle which lets the Modulator read this bit and prepare the proper symbol from the LUT.

g. FFT and REM

FFT and REM enables are controlled by the general control unit, ready and done signals are sent from FFT and REM to the control unit then the control unit sends enable or disable signals to them based on the values of ready and done together. In addition, the control unit checks if there are valid inputs for these blocks, otherwise it disables them.

h. IFFT

IFFT block is enabled by the first valid out from REM and still enabled till the last sample of the current transport block gets out through its output port at this instance block is disabled.

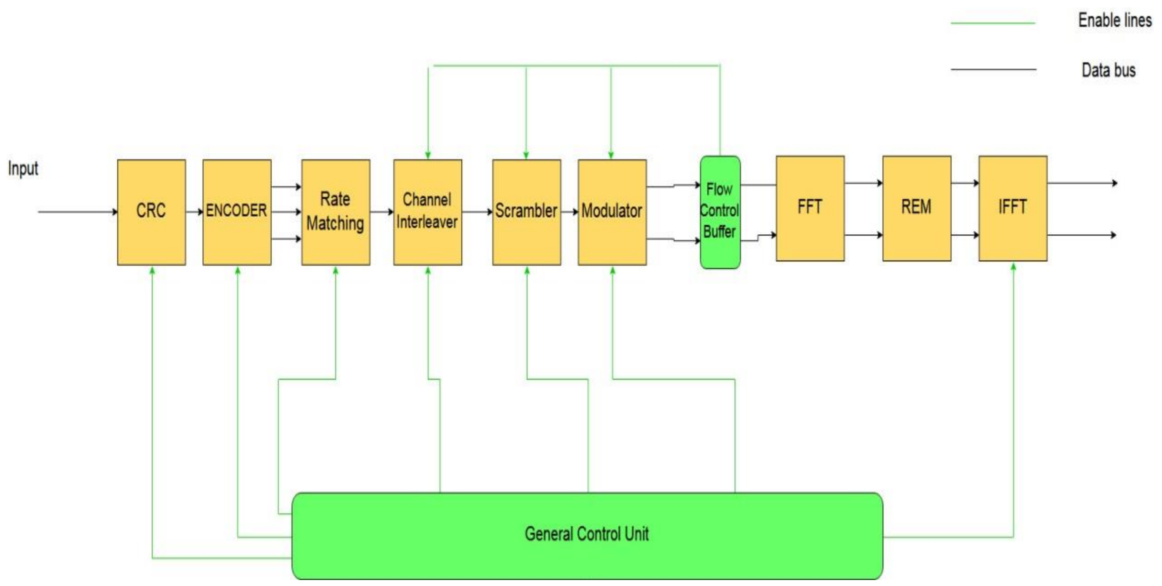


Figure 7.1-1: Detailed Integrated Full Chain

7.2. <u>System clock rate determination</u>

Choosing 54 to be total number of cycles per IFFT iteration to generate one SC-FDMA symbol isn't meaningless but it's for a reason, this is the minimum number of cycles that allows new SC-FDMA to be produced (48 cycles) – recall the memory based proposed architecture - and at the same time it's divisible by 18 – Total number of samples per SC-FDMA duration- to make the block produces output samples with a constant rate during these 54 cycles.

This number affects mainly the clock rate of the whole system as follows:

a. By dividing 54 on 18 we get 3 this 3 means read rate from the buffer = $\frac{Clock\ Rate}{3}$, and this means one clock domain is needed and no need for synchronizers.
b. Then calculation of clock rate could be performed by this simple equation:

   $T_{SC-FDMA}$ = SC-FDMA symbol duration = 54 * T_{clk}

   $T_{SC-FDMA}$ = 71.35 µsec – as defined in standard -.

   Then,

   $T_{clk} = \frac{71.35 * 10^{-6}}{54}$ = 1.32 µsec.

   F_{clk} ~ 756 KHZ.
c. Previous equation contains implicit assumption that there is no bubbles at output of IFFT and the constant output rate is maintained during the period of transport block transmission, this could be guaranteed through:

- Resource Element mapper (REM) provides new valid input for IFFT whenever IFFT ready signal is high.
- FFT block after modification produces new SC-FDMA symbol each 54 cycles to be compatible with the IFFT and eliminate any need for data flow control through this part of chain.
- Data flow control buffer is inserted between FFT and Modulator to ensure availability of valid input for FFT as shown in figure 7.1-1 always this buffer stalls the chain before it (Modulator, Scrambler, Channel Interleaver) when it's full of the needed next valid input for FFT, internal structure of this buffer is as explained in chapter 4 section 4.2.3.

Conclusion

In this work, we presented a full synthesizable design of NB-IoT phy layer transmitter with its full matlab model for testing and other simulations like SQNR and PAPR which are provided in this work too. Results of power and area after synthesizing the full chain using UMC 130 nm technology are also tabulated and justified. New creative solutions and design for blocks are presented clearly. We emphasize on how physical layer is designed differently compared to LTE, and how it is designed to fulfill the performance requirements of IoT such as significant coverage extension using repetitions, low device complexity, long battery lifetime, and supporting a massive number of IoT devices through using smaller bandwidth for each user= 180 KHZ. NB-IoT is also designed to allow easy integration and sharing of radio resources with existing GSM and LTE networks. NB-IoT is a step toward building the fifth generation (5G) radio access technology intended for enabling new use cases like machine type communications. NB-IoT devices, after deployment, are expected to remain in the network for many years, likely beyond network migration toward 5G. It is also important to ensure that NB-IoT continues to evolve toward meeting all 5G requirements for IoT, minimizing any need to introduce a new 5G IoT technology, which may cause market fragmentation and reduce the benefit of economy of scale.

References

[1] http://www.sharetechnote.com/html/Handbook_LTE_NB_FrameStructure_UL.html

[2] 3GPP Technical Specifications 36.211, “Physical channel and modulation.

[3] 3GPP Technical Specifications 36.212, “Physical channel and modulation.

[4] 3GPP Technical Specifications 36.213, “Physical channel and modulation.

[5] A. Abdelbaky and H. Mostafa, "New low area NB-IoT turbo encoder interleaver by sharing resources," 2017 29th International Conference on Microelectronics (ICM), Beirut, 2017, pp. 1-4.

[6] K. Jun and E. E. Swartzlander, "Modified non-restoring division algorithm with improved delay profile and error correction," 2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Pacific Grove, CA, 2012, pp. 1460-1464.

[7] Analysis of Circular Buffer Rate Matching for LTE Turbo Code by Jung-Fu (Thomas) Cheng*, Ajit Nimbalker+, Yufei Blankenship+, Brian Classon+, and T. Keith Blankenship+ * Ericsson Research, RTP, NC, USA + Motorola Labs. Schaumburg, IL, USA

[8] Optimized Rate Matching Architecture for a LTEAdvanced FPGA-based PHY by Karlo G. Lenzi, José A. Bianco F., Felipe A. de Figueiredo, Fabrício L. Figueiredo DRC – Convergent Networks Department CPqD – Research and Development Center Campinas, SP – Brazil

[9] LTE Rate Matching Performance with Code Block Balancing by Josep Colom Ikuno, Stefan Schwarz, Michal ˇSimko Institute of Communications and Radio-Frequency Engineering Vienna University of Technology, Austria Gusshausstrasse 25/389, A-1040 Vienna, Austria

[10] C.-H. Chang, C.-L. Wang, and Y.-T. Chang, “A novel memory-based FFT processor for DMT/OFDM applications,” in IEEE Int. Conference on Acoustics, Speech, and Signal Processing, 1999, pp. 1921–1924.

[11] C.-L. Wang and C.-H. Chang, “A DHT-based FFT-IFFT processor for VDSL tranceivers,” in IEEE Int. Conference on Acoustics, Speech, and Signal Processing, 2001, pp. 1213–1216.

[12] C.-K. Chang, C.-P. Huang, and S.-G. Chen, “An efficient memory-based FFT architecture,” in Proc. IEEE Int’l Symp. on Circuits and Systems (ISCAS), 2003, pp. II–129–II–132.

[13] S.-C. Moon and I.-C. Park, “Area-efficient memory-based architecture for FFT processing,” in Proc. IEEE Int’l Symp. on Circuits and Systems (ISCAS), 2003, pp. 129–132.

[14] S.-Y. Lee, C.-C. Chen, C.-C. Lee, and C.-J. Cheng, “A low-power VLSI architecture for a shared-memory FFT processor with a mixed-radix algorithm and a simple memory control scheme,” in Proc. IEEE Int’l Symp. on Circuits and Systems (ISCAS), 2006, pp. 157–160.

[15] C.-H. Su and J.-M. Wu, “Reconfigurable FFT design for low power OFDM communication systems,” in IEEE Int’l Symp. on Consumer Electronics, 2006, pp. 1–

[16] Design of Cost-Efficient Memory-Based FFT Processors Using Single-Port Memories Yao-Xian Yang, Jin-Fu Li, Hsiang-Ning Liu, and Chin-Long Wey Department of Electrical Engineering National Central University Jhongli, Taiwan, 320

[17] Mahmoud, Hisham A., and Huseyin Arslan. "Error vector magnitude to SNR conversion for nondata-aided receivers." *IEEE Transactions on Wireless Communications* 8.5 (2009).