Zewail City for Science and Technology
University of Science and Technology
Nanotechnology and Nano Electronics Program

"Digital Design and Implementation of Narrowband IOT Physical Uplink Shared Channel Transmitter Chain 3GPP.Rel16 (NPUSCH Tx R16)"

A Graduation Project
Submitted in Partial Fulfillment of
B.Sc. Degree Requirements in
Nanotechnology and Nano Electronics Program

Prepared By

Arwa Ahmed Lamei	201900169
Lobna Tarek Elahraf	201800895
Yasmine Abdelaal	201800256
Yara Ramadan Nofal	201800498

Supervised By

Associate prof. Dr. Hassan Mostafa
Dr. Abdelmohsen Ali

2022/2023

Acknowledgments

We would like to express our sincere gratitude to all those who have contributed to the completion of this thesis.

First and foremost, we want to thank our supervisors: Dr. Abdelmohsen Ali, and Dr. Hassan Mustafa, for their guidance, support, and invaluable feedback throughout the entire process. Their expertise, dedication, and commitment to our success have been truly inspiring.

We are also grateful to the faculty and staff at Zewail City of Science and Technology for providing us with such an excellent academic environment. Their support and encouragement have been instrumental in helping us achieve our goals. Our special thanks are directed to Eng. Youssef Nofal, and our colleagues Tarek Nabil, and Ahmed Hashem for their valuable cooperation and help whenever we need any guidance.

We would like to thank our families, especially our parents and friends for their love, encouragement, and belief in us. Their constant support and understanding have been a source of strength and motivation.

Finally, each one of us would like to express our deepest appreciation to the other members who generously gave their time and effort to help each other and for learning, working and flourishing together, hand in hand, making up the most encouraging environment to work and grow into. Without their collaborations, this work would not have been possible.

Abstract

The recent advances in LPWA (Low Power Wide Area) technology has motivated the emergence of NB-IOT (Narrowband Internet of Things) to be utilized in a wide range of applications, especially low-power and delay-insensitive ones. An uplink of NB-IOT is defined as the link from a user equipment (UE) to a base station (BS). Uplink transmission is a key element for NB-IOT to successfully accomplish the sensitive task of sensor data collection for many applications. The NB-IOT protocol details are clearly investigated through the literature. However, the detailed design and digital implementation that satisfies the strict performance requirements has not been rigorously investigated. In this work, we present the digital design and implementation of the uplink shared channel transmitter chain according to LTE: 3GPP Release.16 standard. First, the standard specifications are thoroughly investigated to address the design requirements for each block. Second, the behavioral simulation for the transmitter chain blocks is implemented using MATLAB as the reference model. Then, the Hardware implementation is performed using Register Transfer Level (RTL). The Hardware Description Language (HDL) implementation output has to prove its correspondence with the reference model output. Finally, the implemented design is tested using FPGA kit, and characterized its performance matrix in terms of Power, Area, and Timing constraints satisfaction.

Key Terms—LTE, NB-IOT, uplink transmission, physical channel, digital design.

Table of Contents

ACKNOWLEDGMENTS **2**

ABSTRACT **3**

TABLE OF CONTENTS **4**

LIST OF TABLES **7**

LIST OF FIGURES **8**

LIST OF ACRONYMS/ABBREVIATIONS **10**

LIST OF SYMBOLS **10**

1 INTRODUCTION AND LITERATURE REVIEW **11**

1.1 General Introduction and overview of the topic 11
1.1.1 LTE NB-IOT Protocol Stack and Architecture *13*
1.1.2 NB-IOT Modes of Operation *18*
1.2 Problem definition 18
1.3 Objectives 20
1.4 Functional Requirements/product specification 22
1.5 Report Organization 22

2 STANDARDS TO BE USED **22**

2.1 Frame structure 22
2.2 Slot structure 24
2.2.1 Resource grid *24*
2.2.2 Resource elements *24*
2.2.3 Resource unit *25*
2.3 SC-FDMA 25
2.4 Transport Block Size (TBS) 26
2.5 Blocks Implementation 27
2.5.1 Cyclic Redundancy Check (CRC) *27*
2.5.2 Turbo Coding *28*
2.5.2.1 Turbo encoder 29
2.5.2.2 Trellis Termination of Turbo encoder 30
2.5.2.3 Internal Inter-leaver of Turbo encoder 31
2.5.3 Rate Matching for turbo coded transport channels *32*
2.5.3.1 Sub-block interleaver 34
2.5.3.2 Bit collection, selection, and transmission 36
2.5.4 Channel Interleaver *37*
2.5.5 Scrambler *38*
2.5.6 Modulator *39*
2.5.7 Fast Fourier Transform (FFT) *40*
2.5.8 Resource Element Mapper (REM) *43*
2.5.8.1 Resource grid 43
2.5.8.2 Resource elements 43
2.5.8.3 Resource Unit 43
2.5.8.4 Resource Allocation 44
2.5.9 Inverse Fast Fourier Transform (IFFT) *46*
2.5.9.1 SC-FDMA baseband signal generation 46
2.5.9.2 IFFT 46

3 MARKET AND LITERATURE REVIEW **48**

3.1 Literature review 48
3.2 Market use cases and deployment 49
3.2.1 NB-IOT devices *49*
3.2.2 Smart parking *49*
3.2.3 Smart city *50*

3.3 TECHNICAL APPROACH 51

4 PROJECT DESIGN 52

4.1 PROJECT PURPOSE AND CONSTRAINTS 52
4.2 PROJECT TECHNICAL SPECIFICATIONS 52
4.3 DESIGN ALTERNATIVES AND JUSTIFICATION 52
4.4 DESCRIPTION OF SELECTED DESIGN 53
4.4.1 CRC 53
4.4.1.1 Design 53
4.4.1.2 Block diagram and architecture 53
4.4.1.3 Block interface 54
4.4.1.4 Operation 54
4.4.2 Turbo Coding 55
4.4.2.1 Design 55
4.4.2.2 Block diagram and architecture 56
4.4.2.3 Block interface 56
4.4.2.4 Operation 57
4.4.3 Rate Matching 58
4.4.3.1 Design 58
4.4.3.2 Block diagram and architecture 59
4.4.3.3 Block interface 59
4.4.3.4 Operation 60
4.4.4 Channel Interleaver 62
4.4.4.1 Design 62
4.4.4.2 Block diagram and architecture 63
4.4.4.3 Block interface 63
4.4.4.4 Operation 64
4.4.5 Scrambler 65
4.4.5.1 Design 65
4.4.5.2 Block diagram and architecture 65
4.4.5.3 Block interface 66
4.4.5.4 Operation 66
4.4.6 Modulator 67
4.4.6.1 Design 67
4.4.6.2 Block diagram and architecture 67
4.4.6.3 Block interface 67
4.4.6.4 Operation 68
4.4.7 FFT 69
4.4.7.1 Design 69
4.4.7.2 Block diagram and architecture 69
4.4.7.3 Block interface 70
4.4.7.4 Operation 71
4.4.8 Resource Element Mapper 72
4.4.8.1 Design 72
4.4.8.2 Block diagram and architecture 73
4.4.8.3 Block interface 73
4.4.8.4 Operation 74
4.4.9 IFFT 75
4.4.9.1 Design 75
4.4.9.2 Block diagram and architecture 76
4.4.9.3 Block interface 77
4.4.9.4 Operation 78

5 PROJECT EXECUTION 78

5.1 SIMULATION RESULTS AND EVALUATION 78
5.1.1 CRC 79
5.1.1.1 MATLAB and Verilog Comparison 79
5.1.1.2 Synthesis and pnr results 80
5.1.2 Turbo Coding 81
5.1.2.1 MATLAB and Verilog Comparison 81
5.1.2.2 Synthesis and pnr results 82

5.1.2.3 Comments....84
5.1.3 Rate Matching....84
5.1.3.1 MATLAB and Verilog Comparison....84
5.1.3.2 Synthesis and pnr results....85
5.1.3.3 Comments....86
5.1.4 Channel Interleaver....87
5.1.4.1 MATLAB and Verilog Comparison....87
5.1.4.2 Synthesis and pnr results....88
5.1.5 Scrambler....89
5.1.5.1 MATLAB and Verilog Comparison....90
5.1.5.2 Synthesis and pnr results....90
5.1.6 Modulator....91
5.1.6.1 MATLAB and Verilog Comparison....91
5.1.6.2 Synthesis and pnr results....93
5.1.7 FFT....94
5.1.7.1 MATLAB and Verilog Comparison....95
5.1.7.2 Synthesis and pnr results....95
5.1.8 Resource Element Mapper....96
5.1.8.1 MATLAB and Verilog Comparison....96
5.1.8.2 Synthesis and pnr results....97
5.1.9 IFFT....99
5.1.9.1 MATLAB and Verilog Comparison....99
5.1.9.2 Synthesis and pnr results....100
5.1.9.3 Comments....101
5.2 FINAL SYNTHESIS AND PNR RESULTS....101
5.2.1 Synthesis summary....101
5.2.2 PnR summary....101
5.3 PROJECT TASKS AND GANTT CHART....102
6 CONCLUSION AND FUTURE WORK....103
6.1 CONCLUSION....103
6.2 FUTURE WORK....104
REFERENCES....106

List of Tables

TABLE 1: NB-IOT PARAMETERS 24
TABLE 2: SUPPORTED COMBINATIONS OF $NSCRU, NslotsUL, and\ NsymbUL$ FOR FRAME STRUCTURE TYPE1 25
TABLE 3: MODULATION ORDER Qm AND TBS INDEX TABLE FOR NPUSCH 26
TABLE 4: TRANSPORT BLOCK SIZE (TBS) FOR NPUSCH 27
TABLE 5: CRC INTERFACE DESCRIPTION AND SYMBOLS 27
TABLE 6: TURBO ENCODER INTERFACE DESCRIPTION AND SYMBOLS 28
TABLE 10: CHANNEL INTERLEAVER INTERFACE DESCRIPTION AND SYMBOLS 38
TABLE 11: SCRAMBLER INTERFACE DESCRIPTION AND SYMBOLS 39
TABLE 12: MODULATOR INTERFACE DESCRIPTION AND SYMBOLS 39
TABLE 13: BPSK MODULATION MAPPING 40
TABLE 14: QPSK MODULATION MAPPING 40
TABLE 15: ALLOCATED SUBCARRIERS FOR $\Delta f = 15\ kHz$ SPACING 45
TABLE 16: NUMBER OF RESOURCE UNITS NRU FOR NPUSCH 45
TABLE 17: NUMBER OF REPETITIONS $NRep$ FOR NPUSCH 45
TABLE 18: SUPPORTED SUBCARRIER COMBINATIONS FOR $\Delta f = 15\ kHz$ SPACING 45
TABLE 19: TECHNICAL SPECIFICATIONS 52
TABLE 20: CRC INTERFACE SIGNALS 54
TABLE 21: TURBO ENCODER INTERFACE SIGNALS 57
TABLE 22: RATE MATCHING INTERFACE SIGNALS 60
TABLE 23: CHANNEL INTERLEAVER INTERFACE SIGNALS 64
TABLE 24: SCRAMBLER INTERFACE SIGNALS 66
TABLE 25: MODULATOR INTERFACE SIGNALS 68
TABLE 26: FFT INTERFACE SIGNALS 71
TABLE 27: REM INTERFACE SIGNALS 74
TABLE 28: IFFT INTERFACE SIGNALS 77
TABLE 29: BINARY REPRESENTATION OF COMPLEX VALUES USED IN MODULATOR 92
TABLE 30: BINARY REPRESENTATION OF COMPLEX VALUES USED IN FFT 94
TABLE 31: SYNTHESIS SUMMARY FOR ALL BLOCKS 101
TABLE 32: PNR SUMMARY FOR SOME BLOCKS 101
TABLE 32: GANTT CHART AND TASKS DISTRIBUTION 102

List of Figures

FIGURE 1: EMERGENCE OF WIRELESS AND CELLULAR NETWORKS [1]. 11
FIGURE 2: NB-IOT APPLICATIONS IN SMART BUILDINGS AND METERS [1]. 13
FIGURE 3: OSI DATA PLANE PROTOCOL STACK [1]. 14
FIGURE 4: NB-IOT DATA-PLANE PROTOCOL STACK [1]. 15
FIGURE 5: NB-IOT CONTROL-PLANE PROTOCOL STACK [1]. 15
FIGURE 6: LTE NB-IOT NETWORK ARCHITECTURE [1]. 16
FIGURE 7: 3GPP LTE NB-IOT PROTOCOL STACK FOR BOTH UE AND ENODEB [1]. 17
FIGURE 8: NB-IOT MODES OF OPERATION [1]. 18
FIGURE 9: UPLINK CHANNEL PROCESSING [1]. 20
FIGURE 10: GENERALIZED DIGITAL DESIGN FLOW STAGES [2]. 21
FIGURE 11: FRAME STRUCTURE TYPE 1 [1] 23
FIGURE 12: UPLINK RESOURCE GRID FOR NB-IOT [1] 24
FIGURE 13: OFDMA TRANSMITTER BLOCKS [4] 26
FIGURE 14: SC-FDMA TRANSMITTER BLOCKS [4] 26
FIGURE 15: STRUCTURE OF THE TURBO ENCODER WITH RATE 1/3 (DOTTED LINES APPLY FOR TRELLIS TERMINATION ONLY) [3]. 29
FIGURE 16: RATE MATCHING FOR TURBO-CODED TRANSPORT CHANNELS [3]. 33
FIGURE 17: RADIX 2 BUTTERFLY 42
FIGURE 18: RADIX 3 BUTTERFLY 42
FIGURE 19: RESOURCE GRID OF $\Delta f = 3.75\ kHz$ SPACING 44
FIGURE 21: INTELLIGENT APPLICATIONS OF NB-IOT [9] 48
FIGURE 22: CRC BLOCK DIAGRAM 53
FIGURE 23: CRC BLOCK INTERFACE 54
FIGURE 24: TURBO ENCODER BLOCK DIAGRAM 56
FIGURE 25: TURBO ENCODER BLOCK INTERFACE 56
FIGURE 26: RATE MATCHING BLOCK DIAGRAM 59
FIGURE 27: RATE MATCHING BLOCK INTERFACE 59
FIGURE 28: RATE MATCHING BLOCK OPERATION 60
FIGURE 29: CHANNEL INTERLEAVER BLOCK DIAGRAM 63
FIGURE 30: CHANNEL INTERLEAVER BLOCK INTERFACE 63
FIGURE 31: SCRAMBLER BLOCK DIAGRAM 65
FIGURE 32: SCRAMBLER BLOCK INTERFACE 66
FIGURE 33: MODULATOR BLOCK DIAGRAM 67
FIGURE 34: MODULATOR BLOCK INTERFACE 68
FIGURE 35: FFT BLOCK DIAGRAM 69
FIGURE 36: RADIX_2 FFT BLOCK INTERFACE 70
FIGURE 37: RADIX_3 FFT BLOCK INTERFACE 70
FIGURE 40: REM BLOCK DIAGRAM 73
FIGURE 42: IFFT BLOCK DIAGRAM 76
FIGURE 42: IFFT BLOCK DIAGRAM 77
FIGURE 43: FIRST 3 STAGES OF 128-POINT IFFT 78
FIGURE 45: RTL RESULTS MATCHED WITH MATLAB FOR CRC 80
FIGURE 46: CRC SETUP TIME RESULT 80
FIGURE 47: CRC AREA 80
FIGURE 48: CRC POWER 81
FIGURE 49: RTL RESULTS MATCHED WITH MATLAB FOR TURBO ENCODER 82
FIGURE 50: TURBO ENCODER SETUP TIME RESULT 82
FIGURE 51: TURBO ENCODER AREA 83
FIGURE 52: TURBO ENCODER POWER 83
FIGURE 53: TURBO ENCODER FINAL CHIP AFTER PNR 83
FIGURE 54: RTL RESULTS MATCHED WITH MATLAB FOR RATE MATCHING 85
FIGURE 55: RATE MATCHING SETUP TIME RESULT 85
FIGURE 56: RATE MATCHING AREA 85
FIGURE 57: POWER 86
FIGURE 59: RTL RESULTS MATCHED WITH MATLAB FOR CHANNEL INTERLEAVER 88

FIGURE 60: CHANNEL INTERLEAVER SETUP TIME RESULT 88
FIGURE 61: CHANNEL INTERLEAVER AREA 88
FIGURE 62: CHANNEL INTERLEAVER POWER 89
FIGURE 65: SCRAMBLER SETUP TIME RESULT 90
FIGURE 66: SCRAMBLER AREA 91
FIGURE 67: SCRAMBLER POWER 91
FIGURE 68: MODULATOR OUTPUT FOR BPSK USING MATLAB 92
FIGURE 69: MODULATOR OUTPUT FOR BPSK WAVEFORM 92
FIGURE 70: MODULATOR OUTPUT FOR QPSK USING MATLAB 92
FIGURE 71: MODULATOR OUTPUT FOR QPSK WAVEFORM 93
FIGURE 72: MODULATOR SETUP TIME RESULT 93
FIGURE 73: MODULATOR AREA 93
FIGURE 74: MODULATOR POWER 93
FIGURE 74: MODULATOR FINAL CHIP AFTER PNR 94
FIGURE 77: FFT SETUP TIME RESULT 95
FIGURE 78: FFT AREA 96
FIGURE 79: FFT POWER 96
FIGURE 82: REM SETUP TIME RESULT 97
FIGURE 83: REM AREA 98
FIGURE 84: REM POWER 98
FIGURE 85: REM FINAL CHIP AFTER PNR 98
FIGURE 87: IFFT SETUP TIME RESULT 100
FIGURE 87: IFFT AREA 100
FIGURE 89: IFFT POWER 100

List of Acronyms/Abbreviations

IOT	Internet of Things
NB-IOT	Narrow Band Internet of Things
3GPP	3rd Generation Partnership Project
LTE	Long Term Evolution
FDD	Frequency Division Duplex
TDD	Time Division Duplex
LAA	License Assisted Access
UE	User Equipment
NPUSCH	Narrowband Physical Uplink Shared Channel
UL-SCH	Uplink Shared Channel
BPSK	Binary Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
TBS	Transport Block Size
CRC	Cyclic Redundancy Check
LFSR	Linear Feedback Shift Register
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
SC-FDMA	Single-Carrier Frequency Division Multiple Access

List of Symbols

Q_m	Modulation Number
M_{SC}^{NPUSCH}	Scheduled Bandwidth for Uplink NPUSCH Transmission, Expressed as a Number of Subcarriers
M_{symb}^{Layer}	Number of Modulation Symbols to Transmit Per Layer for a Physical Channel
N_{symb}^{UL}	Number of SC-FDMA Symbols in an Uplink Slot
N_{slots}^{UL}	Number of Consecutive Slots in an Uplink Resource Unit for NB-IoT
N_{SC}^{UL}	Number of Subcarriers in the Frequency Domain for NB-IoT
n_f	System Frame Number
N_L	Number of Layers
n_{RNTI}	Radio Network Temporary Identifier
n_s	Slot Number Within Radio Frame
N_{ID}^{Ncell}	Narrowband Physical Layer Cell Identity
N_{SC}^{RU}	Number of Consecutive Subcarriers in an UL Resource Unit for NB-IoT
$M_{bit}^{(q)}$	Number of Coded Bits to Transmit on a Physical Channel [for codeword q]
$M_{symb}^{(q)}$	Number of Modulation Symbols to Transmit on a Physical Channel [for codeword q]

1 Introduction and Literature review

1.1 General Introduction and overview of the topic

During the past few decades, wireless communication systems had experienced a great revolution. Wireless technology and networks were evolved from 1G technology to today's 4G systems as shown in figure.1. This evolution started from being voice-centric communication systems such as 1G and 2G networks. Then, several improvements were introduced to support data-centric devices with low to medium data rates (in range of few Mbps), for this purpose 3G wireless networks were introduced showing capability of supporting video, voice, and data services. Finally, 4G activity known as LTE ™ was introduced by 3GPP organization. LTE had revolutionized the wireless communication systems by introducing advanced features compared to its predecessors such as offering high speed, low latency, higher spectrum efficiency, higher cell capacity, and air interface based on Orthogonal Frequency Division Multiple (OFDM) access.

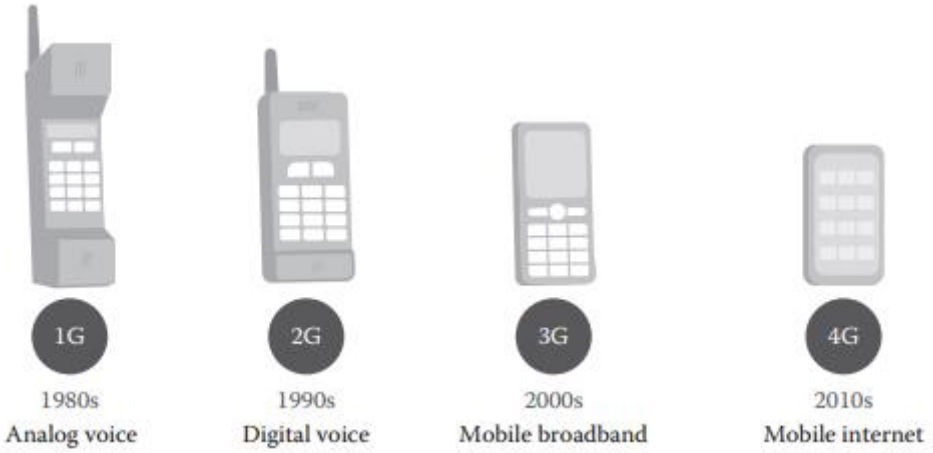

Figure 1: Emergence of wireless and cellular networks [1].

LTE has introduced Machine Type Communication (MTC). It is a technology that enables the communication between devices in addition to the underlying infrastructure for data transport. The communication can take place between an MTC device and a server, or between two MTC devices directly through different networking technologies. MTC significance can be highlighted in a wide range of applications and services in several industrial fields such as manufacturing, energy, process automation, healthcare, and utilities.

Internet of Things (IOT) is a one realization of MTC technology in which all the devices communicate with each other and with network servers or applications. MTC devices number can be very large such that each one have the advantages of low complexity, low power, and low range. They are mostly battery powered without any external power supply source. However, the number of connections between the devices are estimated to be ultra-large with a device density of 1 million devices per square kilometers and an active connection density of 200,000 per square Kilometer.

Starting from Release 13, LTE has introduced one category of the MTC that is known as LTE Narrowband Internet of Things (NB-IOT) that is also known as 3GPP NB-IOT. It delivers different optimization levels for NB-IOT devices such as low power consumption, low data rate, limited bandwidth of 180 KHz, low hardware cost, and extended coverage. NB-IOT devices can be realized as actuators, sensors, wearables such as smart watches, and cameras. One application of NB-IOT is “smart buildings” as shown in Fig.2 where NB-IOT devices that form a large network of connected devices to gather a large amount of information and data and send them remotely to a server for being processed. Additionally, NB-IOT devices can be realized as connected sensors in gas stations that also gather information to be processed by being communicated with base stations (eNodeB) and core networks through cellular infrastructure as shown in Fig.2. These devices are categorized to be non-time critical in terms of data transfer, and they differ from being very simple to extremely complex ones according to the application requirements.

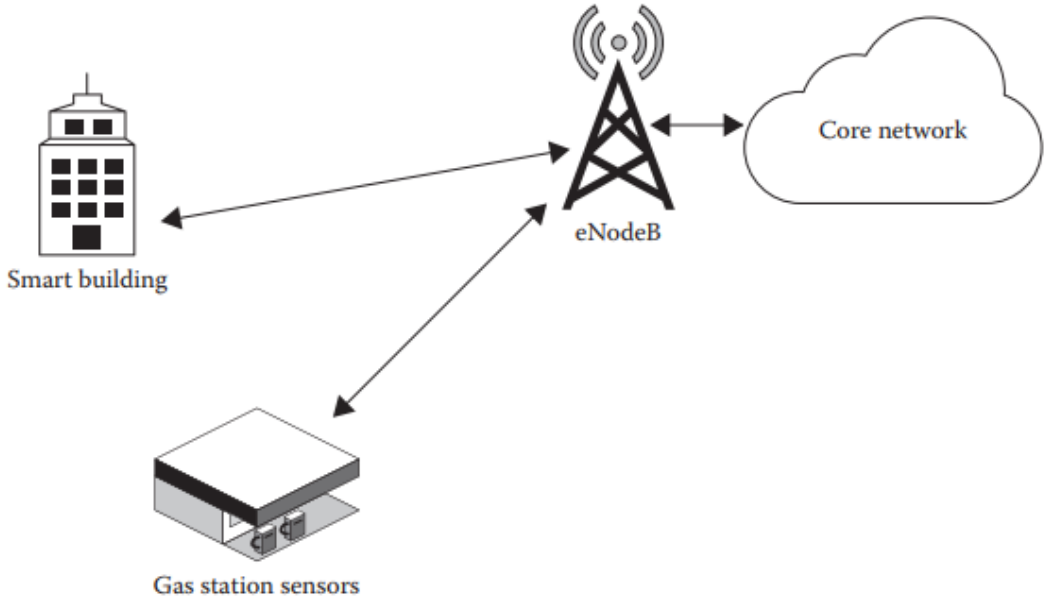

Figure 2: NB-IOT applications in smart buildings and meters [1].

In order to meet the goals of connecting a large number of devices in a wide range of application domains connected through cellular infrastructure to realize the Internet of Things (IOT) with minimal power consumption, low cost, and extended battery lifetime. 3GPP standardized LTE NB-IOT as a stripped version of the full-fledged LTE system extending from release 13 to release 16. NB-IOT is a low power Wide Area Network (WAN) solution that operates in a licensed spectrum band. LTE technology and mobile operators offer a very big robust ecosystem, this motivates 3GPP to standardize and incorporate NB-IOT as part of LTE standards to avoid reestablishment of new cellular infrastructure.

1.1.1 LTE NB-IOT Protocol Stack and Architecture

Network protocol stack is formed through a layered architecture that exists in both transmitting and receiving nodes. For communicating peer nodes at corresponding layers, each layer run a protocol. In order to provide functions or services to the upper layer, this protocol can exchange packets, messages, and Protocol Data Units (PDUs). On the other hand, the protocol exchanges these packets, messages, and PDUs with the lower layer to use its services and functions. The International Standards Organization (ISO) has developed the international standard for computer networks reference model: Open Systems Interconnection (OSI) which designs the structure of the layers as shown

in Fig3. The most bottom two layers (MAC and PHY layers) are called the Access Stratum (AS). They are responsible of handling and processing the physical transmission and reception of the media. The physical media in case of NB-IOT is the wireless channel. The upper five layers are referred to as the Non-Access Stratum (NAS), and they are characterized in terms of their functions and protocols independent of the physical media, thus they are almost the same across different physical media types.

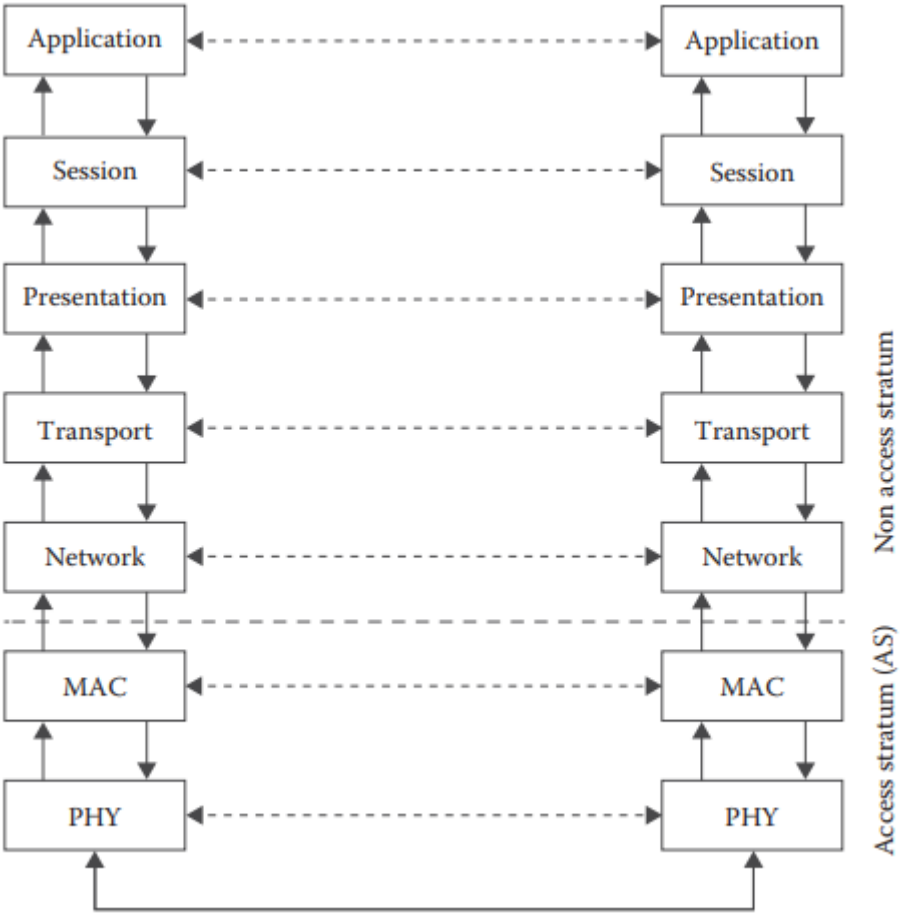

Figure 3: OSI data plane protocol stack [1].

The layering architecture of NB-IOT services, protocol stack, and functions is formed such that they are transmitted and received on a specific media type that is the wireless channel. Hence, NB-IOT does not have all the layers stated in Fig.3. However, only the MAC and PHY layers change while keeping the upper five layers (NAS layers) unchanged. This is because 3GPP protocol stack only defines the air access method and the access stratum and protocols that exist only at the MAC and PHY layers. The layered architecture is further vertically divided into two planes: 1) data plane where

user data flows between the two nodes, and 2) control plane where control information is exchanged. The data plane and control plane for NB-IOT protocol stack are shown in Fig.4 and Fig.5 respectively. In Fig.4, 3GPP defines the access stratum layers which are defined as: Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC), Medium Access Control (MAC), and Physical (PHY) sublayers. Furthermore, in Fig.5, additional control plane sublayers are defined as: Radio Resource Control (RRC), and Non-Access Stratum (NAS) which is considered as a signaling layer.

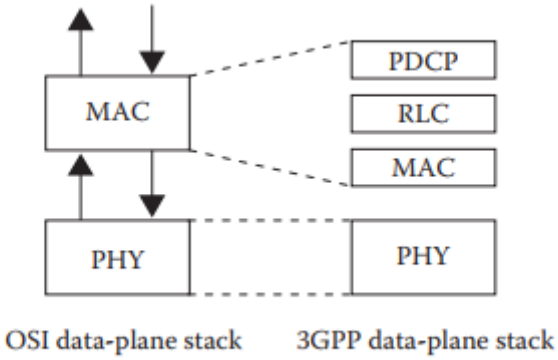

Figure 4: NB-IOT data-plane protocol stack [1].

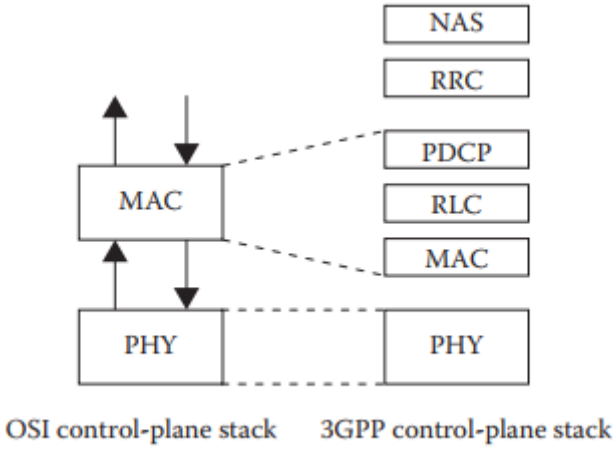

Figure 5: NB-IOT control-plane protocol stack [1].

NB-IOT networking architecture is shown in Fig.6. Such that each eNodeB (base station) is responsible for providing radio coverage to a geographical area, thus all NB-IOT devices in this area can be directly connected to this specific eNodeB. A single or multiple eNodeBs belong to a mobile operator. To enable their services on the mobile operator network, all NB-IOT devices within one service area are equipped with a USIM card. By means of X2 protocol, the eNodeBs are interconnected with each

other in one service area of the mobile operator network. Additionally, eNodeBs are connected to the Evolved Packet Core (EPC) core network by means of S1 protocol. In detail: eNodeB is connected to the Mobility Management Entity (MME) by means of S1-MME protocol which carries control-plane messages and signaling, while eNodeB is connected to the Serving Gateway (S-GW) by means of S1-U protocol which carries the data-plane messages.

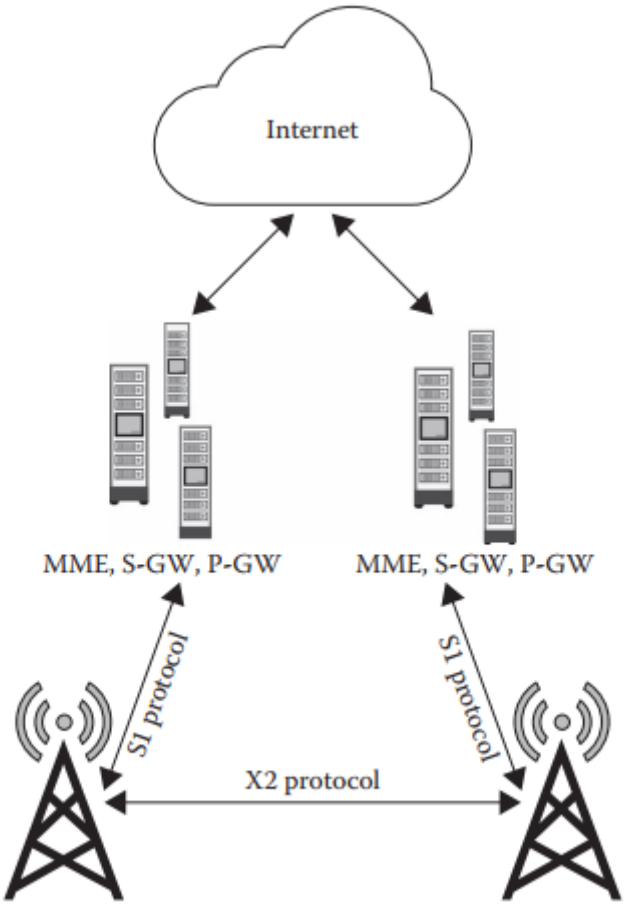

Figure 6: LTE NB-IOT network architecture [1].

The overall 3GPP protocol stack at the three main entities: core network (EPC), eNodeB, and NB-IOT UE (utility equipment), is summarized in Fig.7. Their descriptions are presented in detail as follows:

1) Evolved Packet Core (EPC): The LTE core network has two main interfaces with eNodeB:
   I. S1-MME protocol: it carries all the signaling or control-plane messages, such that control-plane traffic flows from the UE to the eNodeB through S1-MME protocol to the MME. MME is a control-plane component; since it contains the NAS that is considered as an anchor point for signaling or control messages that are exchanged with the UE. The

number of NB-IOT devices within an MME region can extend to hundreds of thousands of devices that cause large number of communications which may overwhelm the MME. For this purpose, there may exist multiple MMEs that can communicate with the same eNodeB and perform load-balancing among themselves. Furthermore, MME performs NAS signaling, and communicates also with S-GW and P-GW, and perform authorization and authentication.

II. II. S1-U protocol: it carries all the user or data-plane messages, such that data-plane messages flow from the UE to eNodeB through S1-U protocol to the Service Gateway (S-GW) that performs packet forwarding and routing to Packet Gateway (P-GW) that allocates IP address to the UE, and perform data rate enforcement in both uplink and downlink, and eventually to the Internet.

Additionally, there exists Home Subscriber Server (HSS) inside the EPC which is used for storing and updating UE subscription information. It also stores UE information where different identity and traffic encryption security keys are generated. In addition, it provide authentication between MME and UE, and protect signaling and data-plane messages exchanged between the UE and eNodeB. It also perform UE identification and addressing, and contains UE profile information such as the subscribed quality of service that includes the maximum allowed bit rate.

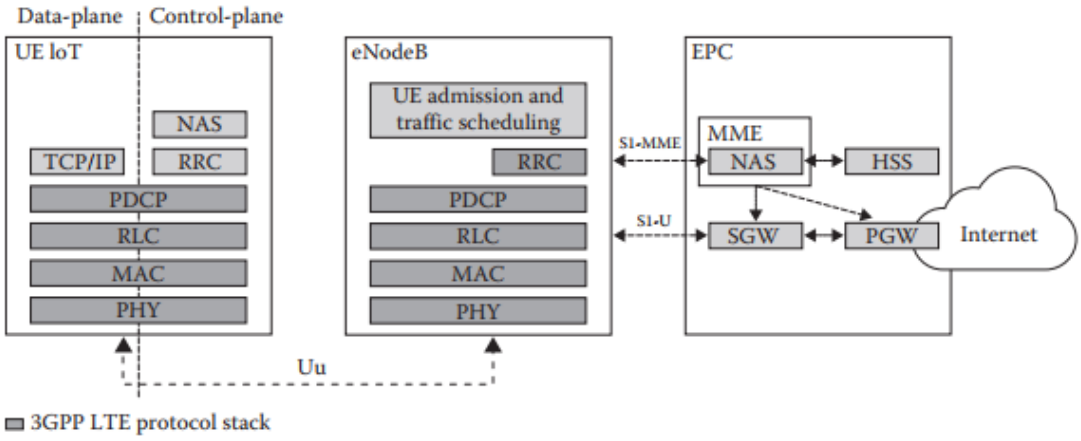

Figure 7: 3GPP LTE NB-IOT protocol stack for both UE and eNodeB [1].

1.1.2 NB-IOT Modes of Operation

The wireless radio interface of the NB-IOT can support three main modes of operation as shown in Fig.8. The modes supported by an NB-IOT device are stated as follows:

1) In-band mode: it utilizes a band of an LTE frequency. Since it utilizes resource blocks within an LTE carrier bandwidth such that one Physical Resource Block (PRB) of LTE occupies 180 KHz of bandwidth. Noting that when the PRB is not used for NB-IOT, eNodeB schedule it to be used for other LTE traffic.
2) Guard-band mode: It utilizes a band of an LTE frequency. Since it utilizes the unused (guard) resource blocks within an LTE carrier's guard-band.
3) Standalone mode: It utilizes a dedicated carrier other than LTE (e.g., GSM). It occupies one GSM channel (200 KHz) [1].

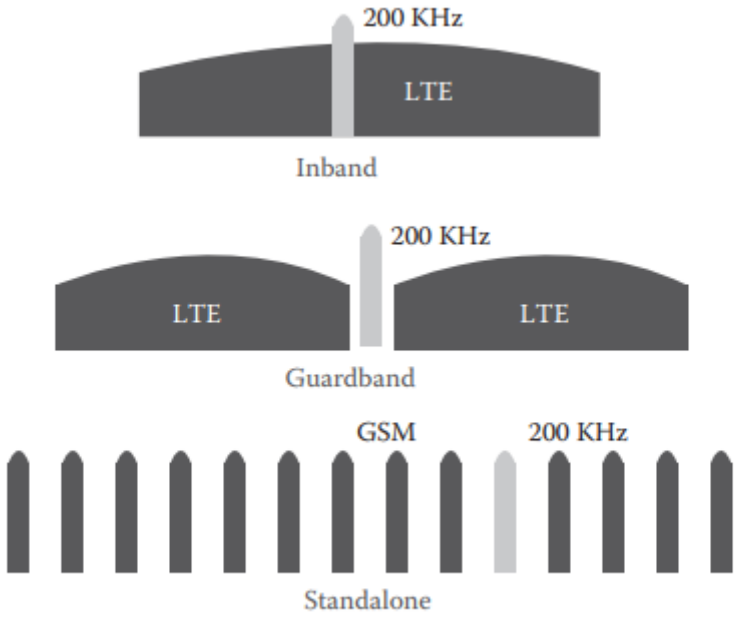

Figure 8: NB-IOT modes of operation [1].

1.2 Problem definition

As discussed in the previous section, this project aims at the design and implementation of the lower most networking layer of the NB-IOT protocol stack which is the Physical (PHY) Sublayer. This layer is responsible for physical channels, transmission, and reception of MAC PDUs (Medium Access Control Protocol Data

Units) as shown in Fig.9. The RRC (Radio Resource Control) provides the configuration parameters to each sublayer, including the PHY sublayer. RRC sends dedicated radio configuration parameters to the PHY sublayer in order to be able to process transmissions and receptions in uplink and downlink, respectively. On the other hand, the PHY configuration parameters are received by RRC from eNodeB during the procedures of RRC connection establishments. At the MAC/PHY interface, transport channels are mapped to physical channels and vice-versa at the transmitter and receiver, respectively.

Specifically, PHY sublayer have Uplink Physical Channel and Downlink Physical Channel to operate in the transmission and reception modes, respectively. The focus on this project is directed towards the Uplink Physical Channel digital design and implementation. The uplink channels have the following physical channels:

- Narrowband Physical Uplink Shared Channel, NPUSCH.
- Narrowband Physical Random Access Channel, NPRACH.
- Narrowband demodulation reference signal.

The focus on this project will be directed towards NPUSCH blocks design and implementation. It is used to transmit uplink transport block such that a maximum of only one transport block is transmitted per carrier. NPUSCH performs the following functionalities as shown in Fig.9, when the MAC sublayer passes a transport block or MAC PDU to PHY layer for uplink transmission:

1. **Cyclic Redundancy Check (CRC) insertion**: 24 bit CRC: it provides error detection capability for transport block transmitted on the uplink.
2. **Channel coding: Turbo coding (coding rate 1/3)**: It is a Parallel Concatenated Convolutional Code (PCCC) with two eight-state constituent encoders and one turbo code internal inter-leaver. The shift registers of the turbo coder are initialized by zeros when starting to encode the input bits.
3. **Rate matching**: It takes the output from the turbo encoder as its input to the three sub-block interleaves, and then to the bit collection, selection and pruning block to output a specified number of rate matched bits according to the number of available resource elements in the resource blocks assigned for transmission. After rate matching, the sequence of

coded bits that correspond to one transport block is referred to as a code-word.

4. **Channel inter-leaver and Scrambler**: bit-level scrambler where the rate matched bits to be transmitted, are scrambled before being modulated.
5. **Modulator**: Each scrambled code-word is modulated using either BPSK or QPSK that corresponds to either 1 bit or 2 bits per complex-value symbol.
6. **Fast Fourier Transform (FFT) and Transform pre-coder**: The number of symbols are divided into a number of sets, each set consists of modulation symbols that corresponds to one SC-FDMA symbol. Since there exists only one single antenna port for the uplink, thus, the modulation symbols are mapped into resource elements directly without any needed precoding.
7. **Resource element mapper**: UE supports only one layer for the uplink. Thus, after modulation, the modulation symbols for the code-word are mapped to one layer.
8. **Inverse Fourier Transform (IFFT) to finally generate SC-FDMA signal to the antenna** [1].

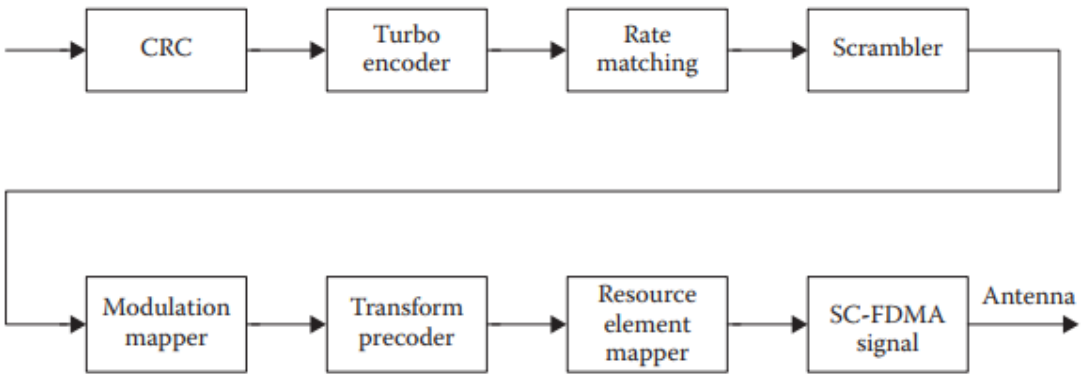

Figure 9: Uplink Channel Processing [1].

1.3 Objectives

The objective is to perform the digital design and implementation for the NPUSCH (Narrowband Physical Uplink Shared Channel) blocks that are illustrated in Fig.9. The design realization will be conducted by applying the ASIC/FPGA design

flow on each block independently, then to perform the design integration at the final stage. The Applied Specific Integrated Circuits/Field Programmable Gate Array (ASIC/FPGA) design flow includes two main design processes: Front-End and Back-End as shown in Fig.10. The project main focus will be on the Front-End design flow that includes HDL Coding, Simulation, and Synthesis. The following stages will represent the project milestones:

1) Specifications: it will be given in reference to the literature models that aims for NB-IOT NPUSCH design.
2) Behavioral Simulation: Using high level language such as MATLAB to act as the golden reference for testing and verifying the RTL design of the NPUSCH blocks in order to ensure that the block design satisfy the functional requirements.
3) RTL design of the NPUSCH blocks.
4) Verification of the designed RTL model in reference to the MATLAB model.
5) Transfer to the Synthesis stage if simulation pass test has positive results
6) Synthesis stage having three inputs: a. Synthesizable RTL code from the previous stage, b. Standard cells according to a specified technology, c. Timing constraints according to the technical specifications. In this stage, RTL design is mapped into standard cells in ASIC design flow or Logic Blocks in FPGA design flow.
7) Transfer to the Back-End flow if pre-layout timing analysis test has positive results [2].

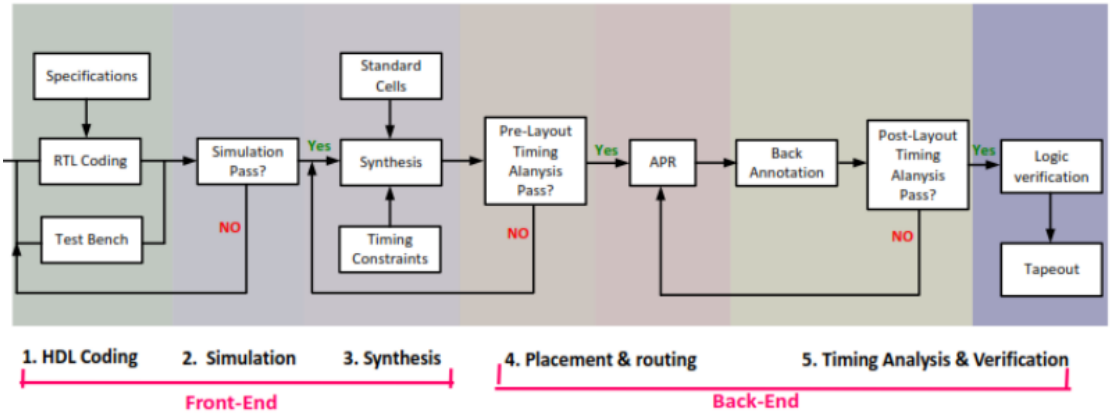

Figure 10: Generalized digital design flow stages [2].

1.4 Functional Requirements/product specification

NB-IOT LTE has a small bandwidth of 180 kHz (when compared with the LTE bandwidth of 1.4-2 MHz) and its main idea depends on its low complexity and low power consumption. According to the 3GPP release 16 in [3], the radio frame of the NB-IOT consists of 10 sub frames, and each sub frame consists of 2 time slots. The NB-IOT supports subcarrier spacing of 3.75 kHz, and 15 kHz. In our design we will be using a subcarrier spacing of 15 kHz as it will decrease the size of both Resource Element Mapper (REM), and Inverse Fast Fourier Transform (IFFT). Hence, the specification of this system is to allocate a bandwidth of 15 kHz for each user. Moreover, only two modulation techniques will be used, Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK).

1.5 Report Organization

Section 1: Background information and literature review about LTE and specifically NB-IoT. Moreover, the objectives and functional requirements for the system are mentioned.

Section 2: General standard specification for the system and the blocks to be implemented.

Section 3: Market and literature review regarding NB-IoT, along with highlighting on its main applications.

Section 4: the design specs, design alternatives, and design flow for each block.

Section 5: MATLAB and RTL implementation and results. Following this, the rest of the possible steps of the ASIC/FPGA flow will be implemented.

2 Standards to be used

All the standards to be used in this project will be according to the 3GPP, Rel 16 V16.2.0 ETSI TS 136 2xx (2020-07) in the NB-IOT section.

2.1 Frame structure

The size of fields in the time domain is expressed as a number of time units $T_s = \frac{1}{15000*2048} = 3.255 * 10^{-8}\ seconds$

The uplink is organized into radio frames with $T_f = 307200 * Ts = 0.01\ second = 10\ ms$.

The supported radio frame structures are:

1. Type 1, applicable to FDD only.
2. Type 2, applicable to TDD only.
3. Type 3, applicable to LAA secondary cell operation only.

In this project we will be using frame structure type 1 which is applicable to half and full duplex FDD only. As shown before, each radio frame is 10 ms long and hence each subframe is 1 ms long. The subframe i in the frame n_f has an absolute subframe $n_{sf}^{abs} = 10n_f + i$. Furthermore, for the subframe using subcarrier spacing of 15 kHz, the subframe i will be define according to 2 slots, $2i$ and $2i + 1$, with each one having a length of 0.5 ms.

for the uplink transmission in FDD case there are 10 sub frames (20 slots or 60 sub slots) available for transmission. Moreover, in the operation of full duplex FDD, the transmission and receiving cannot be done at the same time by the User Equipment (UE). However, this restriction does not apply in the case of full duplex FDD operation.

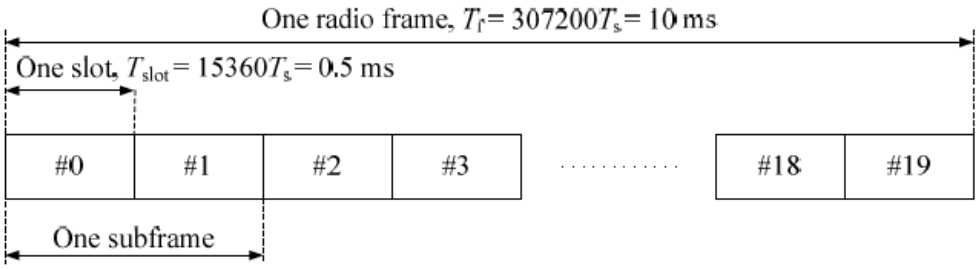

Figure 11: Frame structure type 1 [1]

2.2 Slot structure

2.2.1 Resource grid

The signal transmitted in a slot is described by a resource grid (shown in Fig.12) or multiple ones having subcarriers N_{SC}^{UL}, and SC-FDMA symbols N_{symb}^{UL}. For subcarrier spacing of 15 kHz, the slot number is n_s where $n_s \in \{0,1,2,\dots,19\}$, and for subcarrier spacing of 3.75 kHz the slot number is $n_s \in \{0,1,2,3,4\}$. The values for the uplink bandwidth are given in (table 1) in terms of slot duration T_{slot} and subcarriers N_{SC}^{UL}

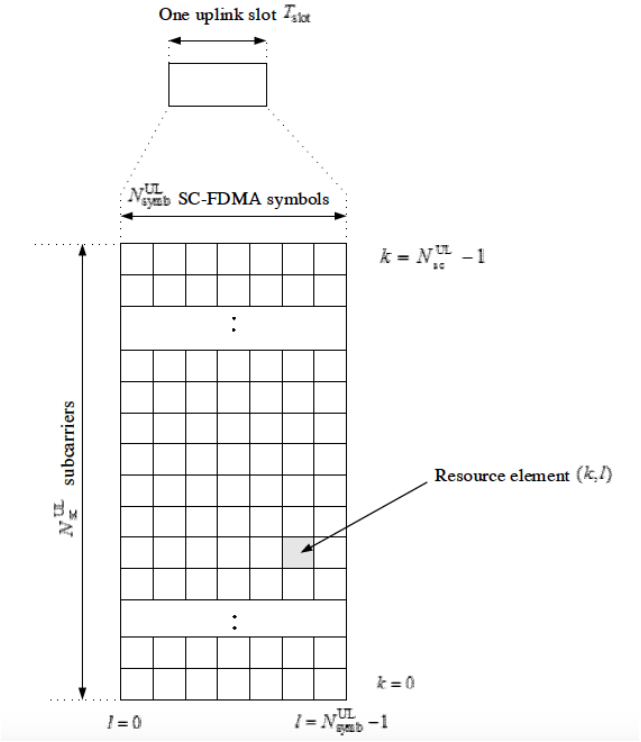

Figure 12: Uplink resource grid for NB-IOT [1]

Table 1: NB-IoT parameters

Subacrrier Spacing	N_{SC}^{UL}	T_{slot}
$\Delta f = 3.75\ kHz$	48	61440 T_s
$\Delta f = 15\ kHz$	12	15360 T_s

2.2.2 Resource elements

The resource grid consists of resource elements defined by (k, l) which, respectively, symbolizes the indices of the frequency and time domains, where $k = 0, \dots, N_{SC}^{UL} - 1$, and $l = 0, \dots, N_{symb}^{UL} - 1$.

2.2.3 Resource unit

The resource unit is utilized in describing the mapping happening between the NPUSCH and the resource elements. The definition of the resource unit is the consecutive subcarriers N_{SC}^{RU} in the frequency domain, and SC-FDMA symbols $N_{symb}^{UL}N_{slots}^{UL}$ in the time domain. N_{symb}^{UL} and N_{SC}^{RU} are shown in (table) for frame structure type 1.

Table 2: supported combinations of $N_{SC}^{RU}, N_{slots}^{UL}, and\ N_{symb}^{UL}$ for frame structure type1

NPUSCH Format	Δf	N_{SC}^{RU}	N_{slots}^{UL}	N_{symb}^{UL}
1	3.75 kHz	1	16	7
	15 kHz	1	16	
		3	8	
		6	4	
		12	2	
2	3.75 kHz	1	4	
	15 kHz	1	4	

2.3 SC-FDMA

The demand for a higher data rate resulted in the implementation of wider transmission bandwidth channels. Upon widening the transmission bandwidth, the channel frequency selectivity becomes difficult and consequently, the inter-symbol interference (ISI) problem becomes more complicated. In order to overcome this issue Orthogonal Frequency Division Multiplexing (OFDM) techniques are used. It used orthogonal subcarriers in order to deliver information. The subcarrier is designed to be smaller than the bandwidth so each one is considered a flat fading channel, and this makes the channel equalization process easier. Thus, OFDM manages to resolve the problem of ISI by splitting the high-rate data stream into a number of lower-rate data that are transmitted in parallel. Unfortunately, OFDM managed to resolve the ISI problem but could not resolve the high peak-to-average power ratio (PAPR) issue.

Single Carrier FDMA (SC-FDMA) is a more adaptable version of the OFDMA where it has the same performance and the same overall complexity and the blocks forming the two systems are nearly equivalent except for the insertion of the DFT block prior to the OFDM blocks. Thus, SC-FDMA may be viewed as DFT-spread OFDMA, where time-domain data symbols are transferred to the frequency domain by DFT before passing through OFDMA modulation. In contrast to OFDMA, which generates a

multicarrier signal, **PAPR is intrinsically low since the entire transmit signal is a single carrier signal [4].**

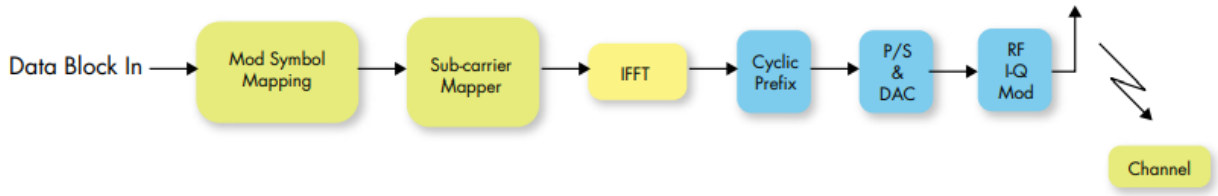

Figure 13: OFDMA transmitter blocks [4]

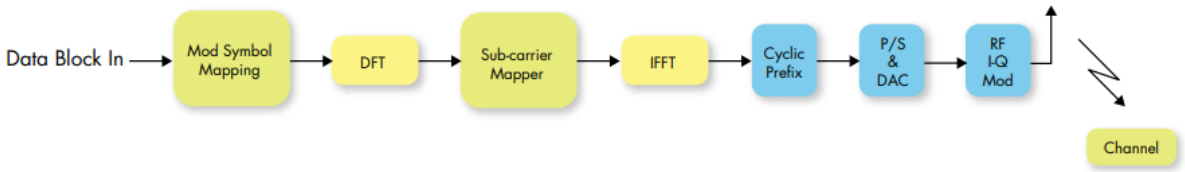

Figure 14: SC-FDMA transmitter blocks [4]

2.4 Transport Block Size (TBS)

The transport block size of the shared channel is configured by the higher layers of the NPUSCH transmission using the following parameters that are read by the UE,

- Modulation and coding scheme field (I_{MCS}), which determines the transport block size index (I_{TBS}) as indicated in Table 3.

Table 3: Modulation order Q_m and TBS index table for NPUSCH

I_{MCS}	Q_m	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

- Resource assignment field (I_{RU}), which determines the transport block size, according to Table 4, based on I_{TBS} determined above.

Table 4: Transport block size (TBS) for NPUSCH

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

2.5 Blocks Implementation

2.5.1 Cyclic Redundancy Check (CRC)

Cyclic redundancy check block represents the first block in the channel coding scheme that is performed as a strategy for error detection and correction, rate matching and interleaving, and transport channel mapping onto the physical layer. Specifically, the CRC task is to generate a sequence of parity bits that are used as an error detection tool that is decoded and checked in the downlink channel, or the receiver, for data validation. CRC code is calculated and added to the transport block as denoted in Table

Table 5: CRC interface description and symbols

CRC interface description	symbol
Input transport block bits, where A is the number of input bits. A takes a value according to the transport block size (TBS) determined as in section 2.4.	$a_0, a_1, a_2, \dots, a_{A-1}$
Parity bits sequence calculated from CRC generation polynomials, where L is the number of parity bits generated. L takes a value of 24, 16 or 8.	$p_0, p_1, p_2, \dots, p_{L-1}$

In NB-IOT, CRC code is generated according to the following CRC generation polynomial,

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

Where the length of CRC $L = 24$.

The encoding is performed by dividing the input sequence by the generation polynomial, where the remainder of the division procedure represents the CRC code to be attached to the transport block. This implies that the data is validated to be correct if the division of the transport block by the same polynomial is found to be zero. Therefore, the output of the CRC block is a sequence of bits denoted by,

$$b_0, b_1, b_2, \dots b_{B-1}; \;\; B = A + L$$

That is composed of two parts as follows,

$$\begin{cases} b_k = a_k & \text{for } k = 0,1,2,\dots,A-1 \\ b_k = p_{k-A} & \text{for } k = A, A+1, A+2, \dots, A+L-1 \end{cases}$$

In NB-IOT, code block segmentation is not required as its maximum block size does not exceed the maximum code block size of $Z = 6144$, according to Table 4.

2.5.2 Turbo Coding

Turbo coding block was designed in order to perform the channel coding such that the inputs and outputs are denoted as shown in Table 6.

Table 6: Turbo encoder interface description and symbols

Turbo encoder interface description	symbol
Bit sequence input for a given code block to channel coding, where *K* is the number of bits to encode (given from CRC output bit sequence)	$c_0, c_1, c_2, \dots, c_{K-1}$
Bit sequence output after encoding, where *D* is the number of encoded bits per output stream noting that *(i)* indexes the output stream (with a range of 0, 1, 2 corresponding to systematic bits, parity bits 1, or parity bits 2, respectively).	$d^{(i)}_0, d^{(i)}_1, d^{(i)}_2, \dots, d^{(i)}_{D-1}$

The channel coding scheme determines the relation between c_k and $d^{(i)}_k$, and between *K* and *D*. The following channel coding schemes can be applied to the transport channels TrCHs:

- Turbo coding.

- Tail biting convolutional coding.

The usage of coding rate and coding schemes is determined according to the type of the TrCH. In UL-SCH the coding scheme used is Turbo coding with a coding rate of **1/3**. The value of *D* is determined according to the Turbo coding scheme with rate 1/3 as in the following equation:

$$D = K + 4$$

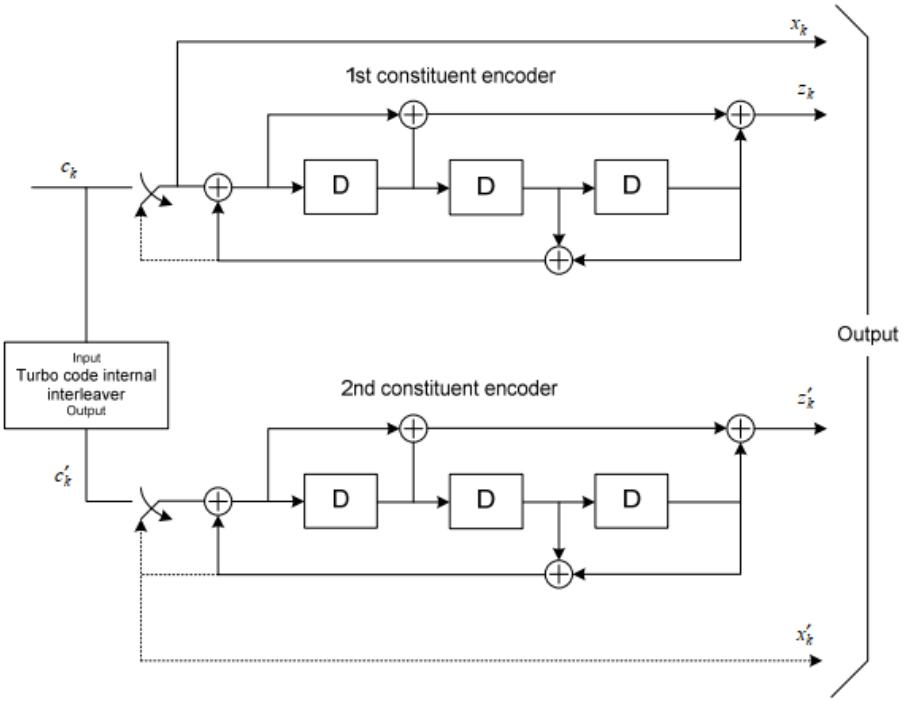

Figure 15: Structure of the turbo encoder with rate 1/3 (dotted lines apply for trellis termination only) [3].

2.5.2.1 Turbo encoder

The scheme of turbo encoder with coding rate 1/3 is shown in Fig.13 such that it consists of:

- Parallel concatenated Convolutional Code (PCCC) with two 8-state constituent encoders.
- One turbo code internal inter-leaver.

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

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

  Such that

$$g_0(D) = 1 + \mathrm{D}^2 + \mathrm{D}^3$$

$$g_1(D) = 1 + \mathrm{D} + \mathrm{D}^3$$

The shift registers of the 8-state constituent encoders are initialized by zeros when starting to encode the input bits. The output from the turbo encoder (before trellis termination) is given by:

Systematic bits: $d^{(0)}_k = x_k$

Parity bits 1: $d^{(1)}_k = z_k$

Parity bits 2: $d^{(2)}_k = z'_k$

Where $k=0, 1, 2\ldots K-1$.

The internal interface (inputs and outputs) of the turbo encoder blocks is described as follows in reference to Fig.13:

1) Internal Inter-leaver:
    - *Input:* Bit sequence input stream to the turbo encoder denoted by $c_0, c_1, c_2 \ldots c_{K-1}$
    - *Output:* Interleaved version of the bit sequence input stream denoted by $c'_0, c'_1, c'_2, \ldots, c'_{K-1}$
2) First Constituent encoder:
    - *Input:* Bit sequence input stream to the turbo encoder denoted by $c_0, c_1, c_2 \ldots c_{K-1}$
    - *Output:* Convoluted version of the bit sequence input stream denoted by $z_0, z_1, z_2, \ldots, z_{K-1}$
3) Second Constituent encoder:
    - *Input:* Interleaved version of the bit sequence input stream denoted by $c'_0, c'_1, c'_2, \ldots, c'_{K-1}$
    - *Output:* Convoluted version of the interleaved bit sequence input stream denoted by $z'_0, z'_1, z'_2, \ldots, z'_K$.

2.5.2.2 Trellis Termination of Turbo encoder

After the encoding of all the information bits, trellis termination is performed by taking the tail bits from the shift register feedback. Such that tail bits are padded after information bits encoding. The termination is made following the two procedures:

- Termination of the first constituent encoder: Use the first three tail bits while disabling the second constituent encoder. This is shown by the upper switch of Fig.13 in the lower position.
- Termination of the second constituent encoder: Use the last three tail bits while disabling the first constituent encoder. This is shown by the lower switch of Fig.13 in the lower position.

The bits that will be transmitted for trellis termination are expressed using the following relations:

$$d_K^{(0)} = x_K, d_{K+1}^{(0)} = z_{K+1}, d_{K+2}^{(0)} = x'_K, d_{K+3}^{(0)} = z'_{K+1}$$
$$d_K^{(1)} = z_K, d_{K+1}^{(1)} = x_{K+2}, d_{K+2}^{(1)} = z'_K, d_{K+3}^{(1)} = x'_{K+2}$$
$$d_K^{(2)} = x_{K+1}, d_{K+1}^{(2)} = z_{K+2}, d_{K+2}^{(2)} = x'_{K+1}, d_{K+3}^{(2)} = z'_{K+2}$$

2.5.2.3 Internal Inter-leaver of Turbo encoder

Having the internal inter-leaver interface as follows:

- *Input:* Bit sequence input stream to the turbo encoder denoted by $c_0, c_1, c_2 \ldots c_{K-1}$
- *Output:* Interleaved version of the bit sequence input stream denoted by $c'_0, c'_1, c'_2, \ldots\ldots, c'_{K-1}$

  Where *K* is the number of input buts.

  The internal interleaver action is controlled by the following relationship between the input and output:

$$c'_i = c_{\mathrm{n}(i)}, i = 0,1 \ldots, (K-1)$$

  Where the relationship between the output index *i* and the input index $\Pi(i)$ satisfies the following quadratic form:

$$\Pi(i) = (f_1 \cdot i + f_2 \cdot i^2) mod K$$

  The parameters f_1 and f_2 depends on the block size *K* according to the output block size from the CRC block. Allowed block size values are summarized in Table 7.

  Noting that there is no need for code segmentation in NB-IOT, since the maximum block size for NB-IOT (K = 2536) is less than the allowed maximum transport block size (K = 6144).

Table 7: Turbo encoder internal interleaver parameters

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

2.5.3 Rate Matching for turbo coded transport channels

The rate matching for turbo coded transport channels is performed per coded block as shown from Fig.14 as follows:

- Firstly, Interleaving the three information bit streams resulted from turbo encoder ($d_k^{(0)}, d_k^{(1)}$ and $d_k^{(2)}$).
- Collection of bits, and generation of a circular buffer.
- Bit selection and pruning.

Inputs and outputs are denoted as shown in Table 8.

Table 8: Rate matching interface description and symbols

Rate matching interface description	symbol
Input information bit stream after encoding, where *D* is the number of encoded bits per input stream noting that *(i)* indexes the output stream (with a range of 0, 1, 2 corresponding to systematic bits, parity bits 1, or parity bits 2, respectively).	$d_k^{(0)}, d_k^{(1)}$ and $d_k^{(2)}$ *Such that* $d_0^{(i)}, d_1^{(i)}, d_2^{(i)}, \dots, d_{D-1}^{(i)}$
Output bit sequence after rate matching, where *E* is the rate matching output sequence length for the coded block.	$e_k, k = 0,1, \dots, E-1.$

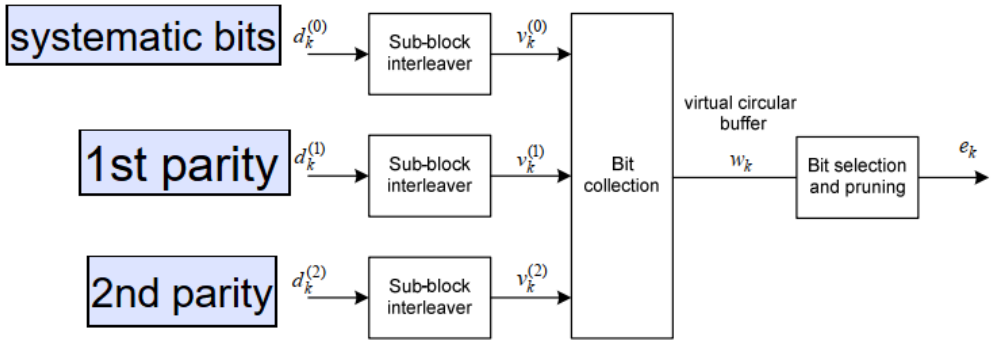

Figure 16: Rate matching for turbo-coded transport channels [3].

The internal interface (inputs and outputs) of the Rate matching blocks is described as follows in reference to Fig.15:

1) Sub-block Interleaver (three parallel blocks):
   - *Input:* Bit sequence input stream to the rate matching block denoted by $d_k^{(0)}, d_k^{(1)}$ and $d_k^{(2)}$; each one is considered as an independent input to one of the three parallel sub-block interleaver blocks.
   - *Output:* Independent interleaved version corresponding to each input bit stream denoted by $v_k^{(0)}, v_k^{(1)}$ and $v_k^{(2)}$; such that $v_k^{(i)}$ is expanded as $v_0^{(i)}, v_1^{(i)}, v_2^{(i)}, \dots, v_{K_\Pi-1}^{(i)}$ where *i* corresponds to each sub-block interleaver index *0,1, or 2*, and K_Π is defined in the sub-block interleaver section 2.3.3.1.
2) Bit collection:
   - *Input:* Three independent interleaved version corresponding to each input bit stream denoted by $v_k^{(0)}, v_k^{(1)}$ and $v_k^{(2)}$
   - *Output:* Collected bit stream denoted by w_k

3) Bit selection and pruning:
    - *Input:* Collected bit stream denoted by w_k
    - *Output:* Rate matched bit stream for transmission denoted by e_k that is generated according to section 2.3.3.2.

2.5.3.1 Sub-block interleaver

Sub-block interleaver three parallel blocks represent the interfacing blocks with the turbo encoder. The input bits to the sub-block inter-leaver are denoted by $d_k^{(0)}, d_k^{(1)}$and $d_k^{(2)}$; such that $d_k^{(i)}$ is expanded as $d_0^{(i)}, d_1^{(i)}, d_2^{(i)}, \dots, d_{D-1}^{(i)}$. However, The output bit sequence from each block interleaver is denoted by $v_k^{(0)}, v_k^{(1)}$ and $v_k^{(2)}$; such that $v_k^{(i)}$ is expanded as $v_0^{(i)}, v_1^{(i)}, v_2^{(i)}, \dots, v_{K_\Pi-1}^{(i)}$ where *i* corresponds to each sub-block interleaver index *0,1, or 2,* and *D* is the number of bits.

The interleaving procedure depends on redistribution of the bit sequence into a rectangular matrix of size $\left(R_{\text{subblock}}^{TC} \times C_{\text{subblock}}^{TC}\right)$. The output bit sequence for each sub-block interleaver is derived as follows:

1) The number of columns inside the matrix is assigned such that $C_{\text{subblock}}^{TC} = 32$, the matrix columns are numbered from left to right as *0,1,2,3, …,* $C_{subblock}^{TC} - 1$
2) The rows of the matrix is determined such that the bit sequence stream input to each sub-block interleaver can fit through a matrix that has 32 columns; thus the number of rws of the matrix is determined by finding the minimum integer R_{subblock}^{TC} that satisfies the following relation:

$$D \leq \left(R_{\text{subblock}}^{TC} \times C_{\text{subblock}}^{TC}\right)$$

Noting that *D* is the length of the input bit sequence stream, and the rows are numbered from top to bottom as *0,1,2,3, … ,* $R_{subblock}^{TC} - 1$

3) If $\left(R_{\text{subblock}}^{TC} \times C_{\text{subblock}}^{TC}\right) > D$, then there have to be N_D number of padded dummy bits that are given by the following relation:

$$N_D = \left(R_{\text{subblock}}^{TC} \times C_{\text{subblock}}^{TC} - D\right)$$

Such that $y_k = <\text{NULL}>$ for $k = 0,1, \dots, N_D - 1$.

Then, $y_{N_D+k} = d_k^{(i)}, k = 0,1,\dots,D-1$, and the bit sequence y_k is written into the $\left(R^{TC}_{\text{subblock}} \times C^{TC}_{\text{subblock}}\right)$ matrix row by row starting with bit y_0 in column 0 of row 0 as shown in the following rectangular matrix:

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

For sub-block interleaver of $d_k^{(0)}, d_k^{(1)}$ $(i = 0,1)$:

4) Inter-column permutation is performed to the generated rectangular matrix based on the pattern $\langle P(j)\rangle_{j\in\{0,1\dots C_{\text{subblock}}-1\}}$, that is shown in table.6, noting that $P(j)$is the original column position of the *j*-th permuted column. The representation of the inter-column permuted $\left(R^{TC}_{\text{subblock}} \times C^{TC}_{\text{subblock}}\right)$ matrix, after permutation of columns is shown as follows:

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

5) The sub-block interleaver output is the bit sequence read out column wise from the inter-column permuted $\left(R^{TC}_{\text{subblock}} \times C^{TC}_{\text{subblock}}\right)$ matrix. Where the bits after sub-block interleaver are denoted by: $v_0^{(i)}, v_1^{(i)}, v_2^{(i)}, \dots, v_{K_\Pi-1}^{(i)}$, where $v_0^{(i)}$corresponds to $y_{P(0)}, v_1^{(i)}$ to $y_{P(0)+C^{TC}_{\text{Subblock}}}$ …. and $K_\Pi = \left(R^{TC}_{\text{subblock}} \times C^{TC}_{\text{subblock}}\right)$.

For sub-block interleaver of $d_k^{(2)}$ $(i = 2)$:

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

$$\pi(k) = \left(P\left(\left(\left\lfloor \frac{k}{R^{TC}_{\text{subblock}}} \right\rfloor\right)\right) + C^{TC}_{\text{subblock}} \times \left(k mod R^{TC}_{\text{subblock}}\right) + 1\right) mod K_\Pi$$

The permutation function *P* is defined in reference to Table.6.

Table.6: Inter-column permutation pattern for sub-block interleaver [3].

Number of columns $C_{subblock}^{TC}$	Inter-column permutation pattern $< P(0), P(1), \ldots, P(C_{subblock}^{TC} - 1) >$
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 >

2.5.3.2 Bit collection, selection, and transmission

The circular buffer performs the collection of the three output bit streams from each one of the three parallel sub-block interleavers such that the output bit sequence of length $K_W = 3K_\Pi$, such that the inputs are assigned to the buffer outputs as shown in the following relations:

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

Noting that for NB-IOT, there exists some special parameters that represent constants to be used in the bit collection, selection, and transmission blocks in rate matching. Those parameters are summarized in Table 9 as shown below:

Table 9: Rate matching block parameters

Rate matching block parameter	Symbol	Value used for NB-IOT design
Number of coded blocks (There exist only single coded block for NB-IOT)	C	1
Code block index	r	1
Modulation order (Type of modulation that will be used to send in the uplink)	Q_m	1: $\pi/2$-BPSK 2: QPSK
The redundancy version index for the HARQ process of this transmission.	Rv_{idx}	0,1,2, or 3
Total number of bits available for transmission of one transport block	G	Input from the top module.
The number of layers a transport block is mapped onto (NB-IOT does not support MIMO transmission).	N_L	1
Soft buffer size for the single coded block (It is defined here for ULSCH).	N_{cb}	K_W

The output sequence from the rate matching has length denoted by *E*, Such that the rate matching output bit sequence is $e_k, k = 0,1, \dots, E-1$, We calculate *E* using the following procedures:

- We define a relation between the modulation order and the number of available block for transmission of one transport block as $G' = G/(N_L \cdot Q_m)$
- Set $E = N_L \cdot Q_m \cdot \lceil G'/C \rceil$

The mapping between the input and the output of the input selection and pruning block is made using the following equations:

$$k_0 = R_{\text{subblock}}^{TC} \cdot \left(2 \cdot \left\lfloor \frac{N_{cb}}{8R_{\text{subblock}}^{TC}} \right\rfloor \cdot rv_{idx} + 2\right)$$

Then the following loop is followed for placing the output elements from the rate matching unit:

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

2.5.4 Channel Interleaver

This block is implemented to minimize the burst errors by rearranging the input such that the noise or error occurring affects only bits in different code words and not the whole code word. The input and outputs of the channel interleaver are shown in Table 10 [3].

Table 10: Channel interleaver interface description and symbols

Channel interleaver interface description	Symbol
The input is the bit sequence resulting from the rate matching, where r is the coded block number, and $\boldsymbol{E_r}$ is the number of rate matched bits for code block number r.	$e_{r0}, e_{r1}, e_{r2}, \dots, e_{r(E_r-1)}$
The output of the channel interleaver is the bit sequence read out column by column form the formed matrix ($\boldsymbol{R_{mux}} * \boldsymbol{C_{mux}}$)	$h_0, h_1, h_2, \dots, h_{R'_{mux}*C_{mux}-1}$

The algorithm for the input rearranging is as follows

- Depending on the modulator used (BPSK or QPSK)
  - If BPSK ($Q_m = 1$), the input stream will be divided into two rows one for the even indexed bits and one for the odd indexed bits.
  - If QPSK ($Q_m = 2$), the input stream stays the same.
- The input is written row by row in the matrix $R_{mux} * C_{mux}$ where the number of rows and columns is determined as follows:
  - $C_{mux} = \left(N_{symb}^{UL} - 1\right) * N_{slots}^{UL}$ where $N_{symb}^{UL}, and\ N_{slots}^{UL}$ are given in table 2 and their values are 7, 16 respectively.
  - $R_{mux} = (H' * Q_m * N_L)/C_{mux}$ and $R'_{mux} = R_{mux}/(Q_m * N_L)$ where $H' = H/(N_L * Q_m)$
  - Hence, $R_{mux} = H/C_{mux}$ where H is the total number of code bits.

- Finally, the matrix is written as follows [3]

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

2.5.5 Scrambler

This block is implemented to convert the input coming from the channel interleaver into a random stream to avoid long sequences of bits having the same values. Each receiver is characterized by a number used in generating a unique scrambling code for the transmitted data and this data cannot be descrambled unless the receiver has the same number. The input and outputs of the scrambler are shown in Table 11 [5].

Table 11: Scrambler interface description and symbols

Scrambler interface description	Symbol
The input is the block of bits where q is the codeword and $\boldsymbol{M_{bit}^{(q)}}$ is the number of transmitted bits on the PUSCH	$b^{(q)}(0), b^{(q)}(1), \dots, b^{(q)}\left(M_{bit}^{(q)} - 1\right)$
The output of the scrambler is the scrambled bits	$\tilde{b}^{(q)}(0), \tilde{b}^{(q)}(1), \dots, \tilde{b}^{(q)}\left(M_{bit}^{(q)} - 1\right)$

The scrambler consists of two Linear Feedback Shift Registers (LFSR) which generates a golden sequence $c(n)$ initialized by two different values. The algorithm for the LFSR is as follows

- The sequences are defined by a gold sequence having a length of 31 bits.
- The output sequence is $c(n)$ where $n = 0,1, \dots, M_{PN} - 1$
- $c(n) = \left(\left(x_1(n + N_c) + x_2(n + N_c)\right) mod 2\right)$ where $N_c = 1600$
- The first m-sequence is $x_1(n) = 1 + D^3 + D^0$ where $x_1$is initialized using $x_1(0) = 1,\ x_1(n) = 0, n = 1,2,3,..,30.$
- The second m-sequence is $x_2(n) = 1 + D^3 + D^2 + D^1 + D^0$ where x_2 is initialized using $c_{init} = \sum_{i=0}^{30} x_2(i) * 2^i = n_{RNTI} * 2^{14} + n_f\ (mod\ 2) * 2^{13} + \frac{n_S}{2} * 2^9 + N_{ID}^{Ncell}$
- Finally, the two sequences are XORed the golden sequence which then gets XORed with the input data.

2.5.6 Modulator

This block is implemented to modulate the scrambled bits coming from the scrambler block onto a carrier. The input and outputs of the Modulator are shown in Table 12 [5].

Table 12: Modulator interface description and symbols

Scrambler interface description	Symbol
The input is the block of scrambled bits where q is the codeword and $\boldsymbol{M_{bit}^{(q)}}$ is the number of transmitted bits on the PUSCH	$\tilde{b}^{(q)}(0), \tilde{b}^{(q)}(1), \dots, \tilde{b}^{(q)}\left(M_{bit}^{(q)} - 1\right)$
The output is a block of complexed valued symbols	$d^{(q)}(0), d^{(q)}(1), \dots, d^{(q)}\left(M_{symb}^{(q)} - 1\right)$

The algorithm used for the modulation will be either BPSK or QPSK

- In case of using BPSK modulation, each bit $b(i)$ is mapped to $x = I + jQ$ according to Table 13.

Table 13: BPSK modulation mapping

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

- In case of using BPSK modulation, each pair of bits $b(i), b(i+1)$ is mapped to $x = I + jQ$ according to Table 14.

Table 14: 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	$-1/\sqrt{2}$	$-1/\sqrt{2}$

2.5.7 Fast Fourier Transform (FFT)

DFT

According to [5], For each layer, $\lambda = 0,1,\dots,v-1$ the block of complex-valued symbols $x^{(\lambda)}(0),\dots,x^{(\lambda)}\left(M_{\text{symb}}^{\text{layer}}-1\right)$is divided into $M_{\text{symb}}^{\text{layer}}/M_{\text{sc}}^{\text{PUSCH}}$ sets, each corresponding to one SC-FDMA symbol. Transform precoding shall be applied according to

$$y^{(\lambda)}(l\cdot M_{\text{sc}}^{\text{PUSCH}}+k)=\frac{1}{\sqrt{M_{\text{sc}}^{\text{PUSCH}}}}\sum_{i=0}^{M_{\text{sc}}^{\text{PUSCH}}-1}x^{(\lambda)}(l\cdot M_{\text{sc}}^{\text{PUSCH}}+i)e^{-j\frac{2\pi ik}{M_{\text{sc}}^{\text{PUSCH}}}}$$
$$k=0,\dots,M_{\text{sc}}^{\text{PUSCH}}-1$$
$$l=0,\dots,M_{\text{symb}}^{\text{layer}}/M_{\text{sc}}^{\text{PUSCH}}-1$$

resulting in a block of complex-valued symbols $y^{(\lambda)}(0),\dots,y^{(\lambda)}\left(M_{\text{symb}}^{\text{layer}}-1\right)$.

The variable $M_{\text{sc}}^{\text{PUSCH}}=M_{\text{RB}}^{\text{PUSCH}}\cdot N_{\text{sc}}^{\text{RB}}$ where $M_{\text{RB}}^{\text{PUSCH}}$ represents the bandwidth of the PUSCH in terms of resource blocks, and shall fulfil

$$M_{\text{RB}}^{\text{PUSCH}}=2^{\alpha_2}\cdot 3^{\alpha_3}\cdot 5^{\alpha_5}\le N_{\text{RB}}^{\text{UL}}$$

where $\alpha_2,\alpha_3,\alpha_5$ is a set of non-negative integers.

According to [5], the previous equations can be interpreted into a mixed radix DFT that can support 1,3,6,and 12 subcarriers and the theory behind its implementation can be generalized from combining the basic prime DFTs. You can find the theory behind radix-2 and radix-3 in the upcoming lines.

Mixed-radix DFT algorithm :

The Discrete Fast Fourier Transform :

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$

where x (n) is a sequence of N input data and is the W_N^{kn} so called twiddle factor. Calculating the DFT directly, using (1) will cost a large number of operations (N^2 operations). Fortunately, due to the symmetries in the calculations, this large number can be reduced and consequently the complexity shall be reduced as well. **Cooley-Tukey** is one of the most used algorithms in DFT implementations. It presents a method to divide the DFT into two smaller DFTs so that N = N1× N2, i.e., the product of the new DFTs is equal to the size of the original DFT. This can be recursively continued down to the prime factors of the size of the original DFT. This resulted in a reduction in the DFT complexity down to (N log N operations).

Radix-2 Algorithm:

(N = 2^m) where m is an integer. The DFT is thus broken down into m DFTs of size 2. This split operation is shown in the following equations:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N/2-1} x(n) W_N^{kn} + \sum_{n=N/2}^{N-1} x(n) W_N^{kn} \\ &= \sum_{n=0}^{N/2-1} x_1(n) W_N^{kn} + \underbrace{W_N^{kN/2}}_{(-1)^k} \sum_{n=0}^{N/2-1} x_2(n) W_N^{kn} \end{aligned}$$

$$\begin{aligned} X(2k) &= \sum_{n=0}^{N/2-1} (x_1(n) + x_2(n)) W_{N/2}^{kn} \\ &= \mathrm{DFT}_{N/2}(x_1(n) + x_2(n)) \\ &= \mathrm{DFT}_{N/2}(B_0) \\ X(2k+1) &= \sum_{n=0}^{N/2-1} \left((x_1(n) - x_2(n)) W_N^n\right) W_{N/2}^{kn} \\ &= \mathrm{DFT}_{N/2}\left((x_1(n) - x_2(n)) W_N^n\right) \\ &= \mathrm{DFT}_{N/2}\left(B_1 W_N^n\right) \end{aligned}$$

The initial DFT is split into two new DFTs, with arguments B_0 and $B_1 W_N^n$ The calculation of B_0 and B_1 can be represented in a flow graph as shown in Fig.17. This graph is often referred to as a butterfly graph due to its shape. As can be seen in the equations, the output of the lower path of the butterfly, B_1 , needs to be multiplied by a twiddle factor, W_N^n. This is the only multiplication needed in the radix 2 FFT.

Stage 1
$B_0 = x_0 + x_1$
$B_1 = x_0 - x_1$

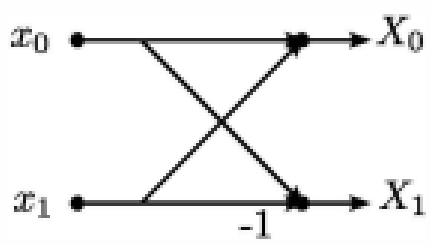

Figure 17: Radix 2 butterfly

Radix-3 Algorithm:

Similar to the radix 2 split, it is possible to split a DFT into radix 3 units, if the original DFT is of a size that can be factored down to one or more threes, i.e., N = 3m. In the radix 3 case the calculation is split into three different DFTs, instead of two as shown in the following equations

$$X(3k) = \mathrm{DFT}_{N/3}\,(B_0)$$
$$X(3k+1) = \mathrm{DFT}_{N/3}\,(B_1 W_N^n)$$
$$X(3k+2) = \mathrm{DFT}_{N/3}\,(B_2 W_N^{2n})$$

Stage 1	Stage 2	Stage 3
$a_0 = x_0$	$b_0 = a_0 + a_1$	$B_0 = d_0$
$a_1 = x_1 + x_2$	$b_1 = a_0 - \frac{1}{2}a_1$	$B_1 = d_1 + jd_2$
$a_2 = x_1 - x_2$	$b_2 = ka_2$	$B_2 = d_1 - jd_2$

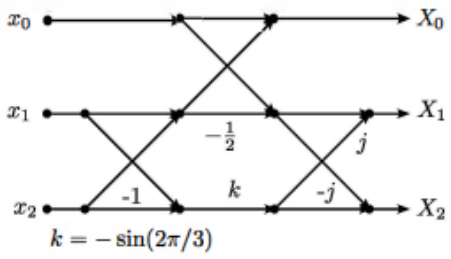

Figure 18: Radix 3 butterfly

$$\Re(W_3^1) = \Re(W_3^2) = -\frac{1}{2} \text{ and}$$
$$\Im(W_3^1) = -\Im(W_3^2) = -\sin\left(\frac{2\pi}{3}\right)$$

where $\Re(.)$ is the real and $\Im(.)$ is the imaginary part of the number. This allows a flow graph as shown in Fig. 18. In this flow graph two internal multiplications, in addition to -1 and j, are shown. However, one of them is a trivial multiplication with $1/2$ and will therefore not add to the hardware complexity. It is crucial to emphasize that radix

3 units can be generated without internal multiplications. This is only advantageous in the case of a DFT with multiple radix 3 units since it demands a non-trivial change in basis for the inputs and outputs. Two of the three outputs must be multiplied using twiddle factors in addition to the internal multiplications [6].

2.5.8 Resource Element Mapper (REM)

Resource element mapper block is the intermediate stage between FFT block and IFFT block, which allocates the outputs of the FFT block in the constructed resource units. The resource element mapping procedure consists of 4 factors as follows,

2.5.8.1 Resource grid

A slot of transmitted information is represented by a resource grid as mentioned in 2.2.1 based on the supported subcarrier spacings in the NB-IoT, according to Table 1. This grid is repeated as a building block to construct the resource unit to be filled with transmitted information denoted as symbols. In this design, a spacing of $\Delta f = 15\ kHz$ is used, hence $N_{symb}^{UL} = 7$ according to table. 2. Thus, each time slot consists of 7 symbols in the time domain that are divided into subcarriers in the frequency domain. The supported number of subcarriers is to have a value of 1, 3, 6, or 12.

2.5.8.2 Resource elements

Resource elements are the complex quantities to be allocated in the frame structure that are obtained as an output from the FFT block. After being allocated, the resource elements take indices (k, l) in the frequency domain and the time domain as described in 2.2.2, where $a_{k,l}$ corresponds to a single complex value. The elements that are not used for transmission are set to zero in their assigned slot.

2.5.8.3 Resource Unit

A sequence of resource units, as described in 2.2.3, is transmitted after the mapping of the resource elements on the NPUSCH according to one of the combinations mentioned in Table 2, in order to formulate the frame structure of the NB-IoT. Based on these information and parameters, the frame structure is graphically interpreted in Fig.19 and Fig.20, according to the type of spacing.

This design is based on the spacing of $\Delta f = 15\ kHz$, therefore the transmission frame holds a structure similar to the provided in Fig.19.

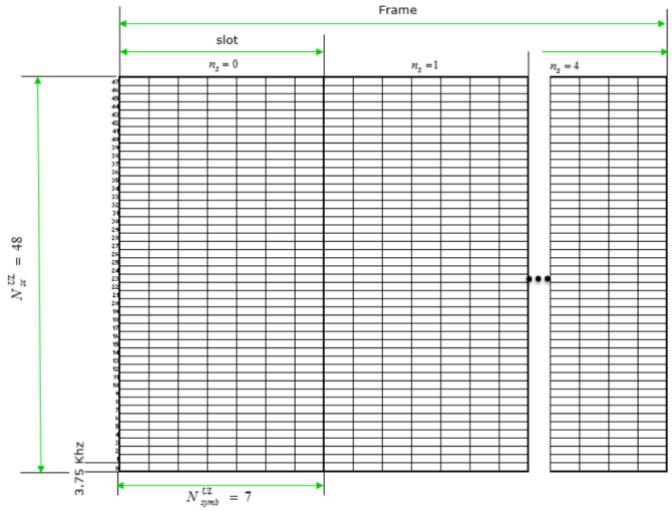

Figure 19: Resource grid of $\Delta f = 3.75\ kHz$ spacing

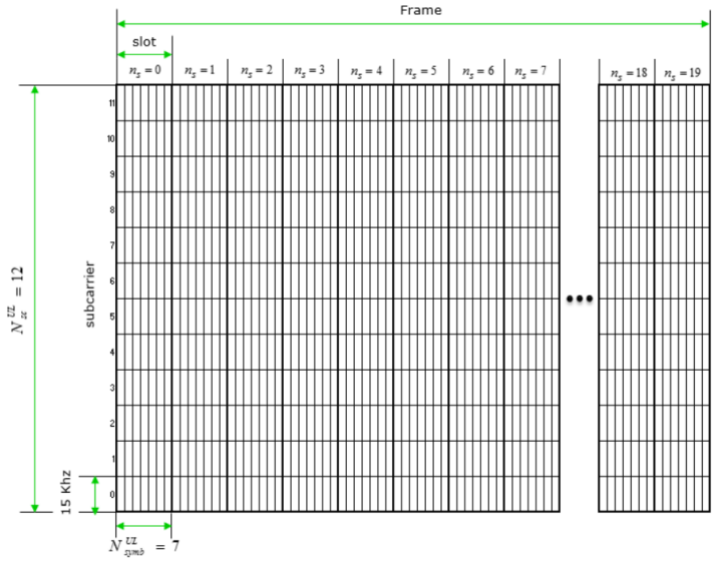

Figure 20: Resource grid of $\Delta f = 15\ kHz$ *spacing*

2.5.8.4 Resource Allocation

Resource allocation is a process in which the complex-valued resource elements are to be placed into the frame of the assigned bandwidth on the shared channel. The steps of this process are determined based on parameters configured by the higher layers for NPUSCH transmission that are indicated by the UE. These parameters are,

- Subcarrier indication field (I_{sc}), which determines the set of contiguously allocated subcarriers n_{sc} between the 12 subcarriers assigned for the NB-IoT according to the $15\ kHz$ spacing. n_{sc} is determined as indicated in Table 15.

Table 15: Allocated subcarriers for $\Delta f = 15\ kHz$ spacing

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

- Resource assignment field (I_{RU}), which specifies the number of resource units N_{RU} according to Table 16.

Table 16: Number of resource units N_{RU} for NPUSCH

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

- Repetition number field (I_{Rep}), which determines the repetition number N_{Rep} according to Table 17.

Table 17: Number of repetitions N_{Rep} for NPUSCH

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

Having the previous parameters determined, the resource allocation process takes place by allocating the resource elements with one of the four following combinations, Table 2, of the 15 kHz spacing as a part of the NB-IoT supported combinations in Table 18.

Table 18: Supported subcarrier combinations for $\Delta f = 15\ kHz$ spacing

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

As indicated above, the number of subcarriers has a value of 1, 3, 6 or 12 which follows the number of points of the FFT producing the allocated resource elements. The position of these allocated subcarriers between the 12 available subcarriers is determined as mentioned above by n_{sc}. The rest of the subcarriers are padded with zeroes when no information is available for transmission. The allocation process follows the indexing (k, l), where $k = 0, \dots, N_{SC}^{UL} - 1$, and $l = 0, \dots, N_{symb}^{UL} - 1$, in increasing order of the index k, then the index l, starting with the first slot in the assigned resource unit by excluding the symbols assigned for the transmission of the reference signal. The mapped subcarriers are then placed between the 128 subcarriers of the physical layer, for the 128-point IFFT to take its input from.

2.5.9 Inverse Fast Fourier Transform (IFFT)

2.5.9.1 SC-FDMA baseband signal generation

The time-continuous signal $s_l^{(p)}(t)$ for antenna port p in SC-FDMA symbol l in an uplink slot is defined by

$$s_l^{(p)}(t) = \sum_{k=-\left\lfloor N_{\mathrm{RB}}^{\mathrm{UR}} N_{\mathrm{sc}}^{\mathrm{RB}}/2 \right\rfloor}^{\left\lceil N_{\mathrm{RB}}^{\mathrm{UR}} N_{\mathrm{sc}}^{\mathrm{RB}}/2 \right\rceil - 1} a_{k^{(-)},l}^{(p)} \cdot e^{j2\pi(k+1/2)\Delta f\left(t - N_{\mathrm{CP},T_{\mathrm{s}}}\right)}$$

for $0 \le t < \left(N_{\mathrm{CP},l} + N\right) \times T_{\mathrm{s}}$ where $k^{(-)} = k + \lfloor N_{RB}^{UI} N_{sc}^{RB}/2 \rfloor, N = 204\varepsilon, \Delta f = 15\text{kHz}$ and $a_{k,l}^{(p)}$ is the content of resource element (k, l) on antenna port p.

2.5.9.2 IFFT

The previous equation describes three consecutive blocks. In order to implement the IFFT only, the following butterfly algorithm is used as a reference for performing decimation in time based on radix-2 then some manipulations are done on this block's result in order to map for the standard equation.

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn}, k = 0,1, \dots, N-1 \\ &= \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn} \\ &= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)} \end{aligned}$$

Where $W_{N=e^{-j2\pi/N}}$

Using the following substitution $W_N^2 = W_{N/2}$:

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m)W_{N/2}^{km} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m)W_{N/2}^{km}$$
$$= F_1(k) + W_N^k F_2(k), k = 0,1, \dots, N-1$$
$$f_1(n) = x(2n)$$
$$f_2(n) = x(2n+1), n = 0,1, \dots, \frac{N}{2} - 1$$

Where $F_1(k)$ and $F_2(k)$ are the N/2 point DFT of sequences $f_1(m)$ and $f_2(m)$ respectively.

Since $F_1(k)$ and $F_2(k)$ are periodic , with period N/2 then $F_1(k + N/2) = F_1(k)$ and $F_2(k + N/2) = F_2(k)$. In addition , the factor $W_N^{K+N/2} = -W_N^K$. Hence,

$$X(k) = F_1(k) + W_N^k F_2(k), k = 0,1, \dots, \frac{N}{2} - 1$$
$$X\left(k + \frac{N}{2}\right) = F_1(k) - W_N^k F_2(k), k = 0,1, \dots, \frac{N}{2} - 1$$

And the sequence can be further split to account for any integer number of stages raised to the power of 2 [7].

According to the standard parameters :

N = 128, Which requires 7 stages from the above algorithm.

3 Market and Literature Review

3.1 Literature review

The 3GPP has suggested the low power wide area (LPWA) technology known as NB-IoT, which is standards-based and intended to enable a variety of new IoT products and services. In comparison to older technologies, NB-IoT significantly increases system capacity, reduces connected devices' power consumption, and increases spectral efficiency. For a variety of use cases, NB-IoT can provide 10 years and longer of device's battery life [8]

The NB-IoT technology can achieve the needs of wide coverage, low data transmission rate along with low power consumption, and huge capacity due to its characteristics, but its challenge is supporting high mobility. Therefore, NB-IoT is better for services that require real-time data transmission, discontinuous movement, low latency sensitivity, or static. The various NB-IoT applications can be classified as , smart buildings, intelligent user services, smart metering, intelligent environment monitoring, and smart cities, as shown in Fig. 21. intelligent user services include smart homes, wearable technology, people tracking, etc. Intelligent environment monitoring includes pollution monitoring, Intelligent agriculture, soil detection, water quality monitoring, etc [9].

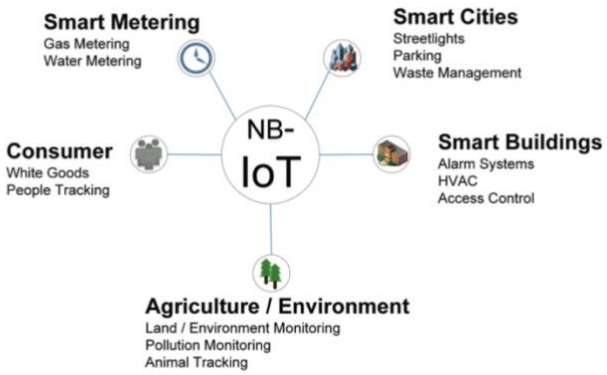

Figure 21: Intelligent applications of NB-IoT [9]

3.2 Market use cases and deployment

3.2.1 NB-IOT devices

Millions of NB-IoT devices are anticipated to be deployed. These devices are gathering a significant amount of structured and unstructured data, which is then transmitted to a centralized spot such as a cloud infrastructure, where it is stored, processed, and then made available to users. such devices are commonly found in actuators and sensors [1].

3.2.2 Smart parking

NB-IoT devices with ultrasonic sensors, having NB-IoT UE chip, can now be used for smart parking for automobiles, trucks, and motorbikes to find parking spots. the availability of parking spots is detected by the UE and transmitted to a centralized server via the eNodeB. All data from cellular and local NB-IoT devices is received by the server, which then is stored in a storage area that is cloud based for later analysis and processing. Co-location of the storage and server is enabled in the cloud. For smart parking, when using the NB-IoT, each parking space has a sensor. The sensor node is a tiny, ultra-low power consumption device made up of an ultrasonic device and a NB-IoT module.

The sensor node is an integral part of the technology, which enables devices to communicate with each other. In a parking lot, these nodes can be installed in each spot and activated every few seconds. Once a change in status occurs, such as when someone parks or leaves their car, the new information will be sent to the Cloud server so that it can be shared with all drivers who subscribe to this service. The node then goes into sleep mode until another event takes place at that spot. By using NB-IoT devices for communication between sensors and cloud servers, full details about any changes in status are delivered quickly and accurately; including time/date stamps for when events occur at specific spots within a parking lot or garage area. This system helps drivers by providing them up-to-date information on available spaces without having to search around themselves, saving both time and energy while also reducing traffic congestion due to people searching for empty spots unnecessarily.

Overall, this system provides many benefits not only by helping locate open spaces but also allowing administrators improved management capabilities over their lots through detailed analytics reports generated from collected data points about usage patterns throughout different times of day/week etc. This type of technology is becoming increasingly popular among cities looking towards more efficient ways manage public transportation infrastructure like roads & highways as well as private businesses seeking better control over customer flow rates inside retail stores etc [1].

3.2.3 Smart city

Numerous NB-IoT applications in the fields of energy plants and management, underground transportation, traffic signals, law enforcement, sewage and water systems, and other applications will be present in modern and smart cities. A smart city promotes a data-driven economy in addition to implementing smart apps. Smart cities have advantages for its citizens as well as for investors, tourists, and the government. These applications depend on NB-IoT to send a significant quantity of structured and unstructured data that may be utilized for analysis, automation, and decision-making.
By using analytics to the data that is sent by NB-IoT sensors located all across the grid, smart electrical grids increase the efficiency of electricity distribution. A cloud-based server is used to configure, manage, and analyze the grid using connected NB-IoT sensors for monitoring the grid. The data can be used by grid operators to forecast and predict demand and capacity. Public rivers, parks, and green areas are monitored by environmental NB-IoT sensors. These sensors send data that is used to pinpoint areas that need cleaning or protection. These environmental sensors can also be used to monitor ambient environmental factors including temperature, rainfall, humidity, and air quality at various points throughout the city [1].

3.3 Technical approach

The technical approach of this project will be based on executing the possible stages of the ASIC/FPGA flow. The flow includes:

1) Functional Specifications: it will be given in reference to the literature models that aims for NB-IOT NPUSCH design.
2) High Level Code: creating MATLAB codes for the NPUSCH blocks to act as a reference for the behavioral simulation stage.
3) HDL: creating RTL design of the NPUSCH blocks.
4) Behavioral Simulation: Using the high-level code to act as the golden reference for testing and verifying the RTL design of the NPUSCH blocks in order to ensure that the block design satisfy the functional requirements.
5) Synthesis: in this stage, RTL design will be mapped into standard cells in ASIC design flow or Logic Blocks in FPGA design flow.
6) Floor planning (only in ASIC flow): in this stage, the main design's objects' size and placement will be decided.
7) Place and Route: the standard cells are placed inside the core boundary and after that the routing takes place.

4 Project Design

4.1 Project purpose and constraints

The objective is to perform the digital design and implementation for the NPUSCH blocks. The project main focus will be on the Front-End design flow that includes HDL Coding, Simulation, and Synthesis. NB-IOT LTE has a small bandwidth of 180 kHz, and its main idea depends on its low complexity and low power consumption. The radio frame of the NB-IOT consists of 10 sub frames, and each sub frame consists of 2 time slots. The NB-IOT supports subcarrier spacing of 3.75 kHz, and 15 kHz. In our design we will be using a subcarrier spacing of 15 kHz.

4.2 Project technical specifications

Table 19: Technical specifications

Specifications	
Uplink Peak Rate (Mbps)	~105/159 kbps
UE Transmit Power (dBm)	20
Max Uplink TBS	2536 bits
Latency	1.6-10 seconds

4.3 Design alternatives and justification

Alternative 1: NB-IOT via LEO satellites

In order to provide the NB-IOT connectivity to the on-ground users equipments (UEs), this alternative introduces the use of Low-Earth Orbit (LEO) satellites. This is proposed to be done by replacing the conventional resource allocation algorithms which were designed for terrestrial infrastructures NB-IOT systems. The conventional approach is characterized by having a very slow variation with time for the system as a whole, furthermore, the devices are under the coverage of a specific base station (BS). The existing design strategies cannot be applied or integrated to the LEO satellite-based NB-IOT systems. The reasons include: First, the change over time of the corresponding channel parameters for each user with the movement of the LEO satellite, thus, delaying the user scheduling would result in an outdated resource allocation. Second, the LEO communications side effects such as the differential Doppler shift dependence on the relative distance among users. Thus, users who overcome a certain distance will be scheduled at the same radio frame leading to violation in the differential

Doppler shift limit supported by the NB-IOT standard. Third, increase in the propagation delay over a LEO satellite to be 4 to 16 times higher compared to the terrestrial system. Thus, imposing the need for minimization of messages exchange between the users and the base station. However, novel design approaches were investigated to propose an uplink resource allocation strategy that incorporates the advantages of using NB-IOT via LEO satellites with considerations to the distinct channel conditions, data demands of several users on earth, and satellite coverage times [10].

4.4 Description of selected design

4.4.1 CRC

4.4.1.1 Design

The design of the CRC block is implemented based on an LFSR, linear feedback shift register, in which the feedback is introduced to the registers by XORing the output signal with the registers placed according to the polynomial mentioned in section 2.5.1.

4.4.1.2 Block diagram and architecture

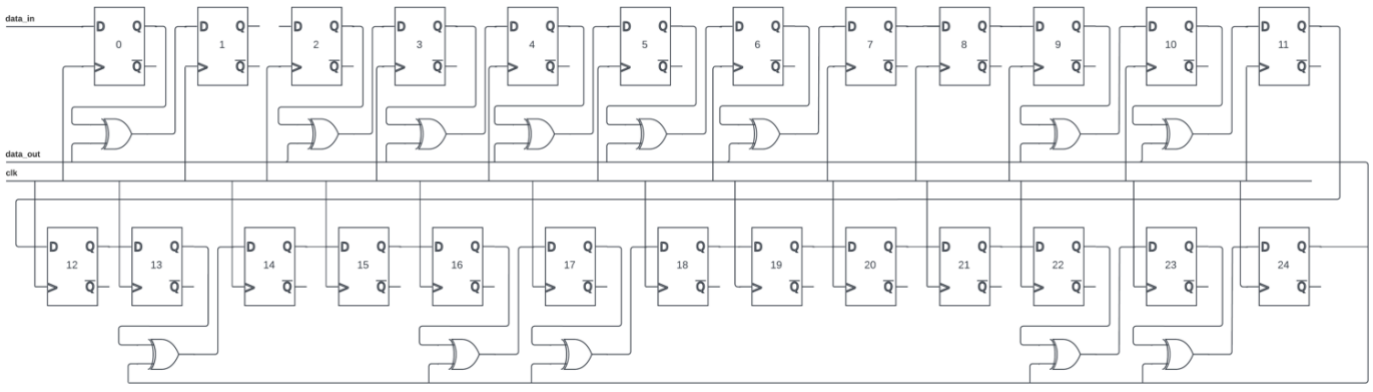

Figure 22: CRC block diagram

4.4.1.3 Block interface

The following figure shows the interface of the CRC block as implemented in the RTL,

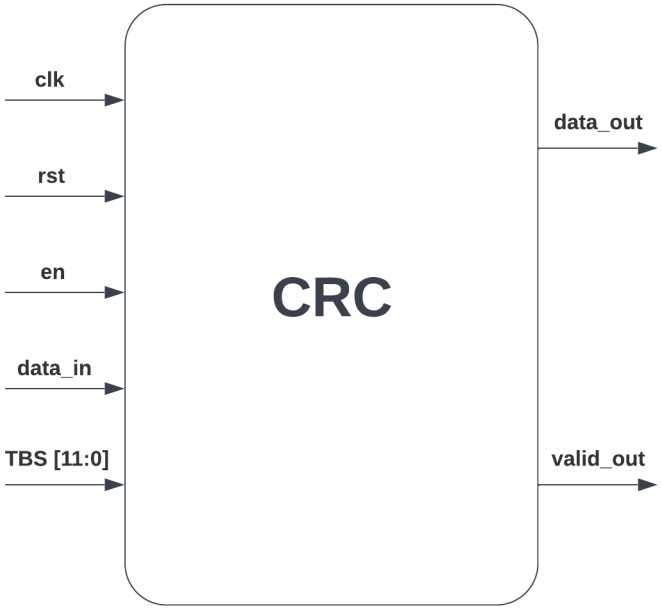

Figure 23: CRC block interface

Table 20: CRC interface signals

Signal	Width	Port type	Description
clk	1 bit	Input	System clock signal
rst	1 bit	Input	Reset signal
en	1 bit	Input	CRC enable signal.
data_in	1 bit	Input	Input bit stream
TBS	12 bits	Input	Transport block size as received from the upper layer
data_out	1 bit	Output	Output bit stream followed with 24 CRC code bits
valid_out	1 bit	Output	Flag to indicate that the output is valid to propagate to the following blocks (output is available)

4.4.1.4 Operation

The flow of the block starts by introducing the input bit stream to the shift register that is initialized by zeros. The shifting is performed along with XORing the states of the registers with the feedback signal form the output according to the polynomial. The output stream starts with the flow of the input followed by 24 bits that represent the generated CRC code. A flag (valid_out) is raised when the output is ready to propagate to the following blocks. A single bit takes 25 clock cycles to get out from

the LFSR, while the block takes (TBS+50) clock cycles to finish the operation and all the CRC code bits are out from the LFSR.

4.4.2 Turbo Coding

4.4.2.1 Design

The design of the Turbo Encoder is based on:

1) LUT: look up table that is used to find the f1, f2 parameters according to the specified Transport block size (TBS).
2) Pi: It is used to calculate the interleaved index PI(i) according to the calculated f1,f2 parameters from LUT.
3) Buffer: It is used in order to map the input stream of the calculated index to be the output interleaved version of the internal inter-leaver.
4) Upper Constituent Encoder: It is used to encode the normal output stream (generated from CRC), and to perform some processing some processing in order to extract the systematic bit output stream from the turbo encoder (x_k), in addition to the parity 1 bit stream (z_k). Furthermore, here the calculation of the termination bits (used to flush the upper encoder registers) for the upper encoder bit stream takes place.
5) Lower Constituent Encoder: It is used to encode the interleaved output stream (generated from sub_block_interleaver), and to perform some processing in order to extract the termination used bit stream from the turbo encoder (x_k_bar), in addition to the parity 2 bit stream (z_k_bar). Furthermore, here the calculation of the termination bits (used to flush the lower encoder registers) for the lower encoder bit stream takes place.
6) Mux: It exists in the top modeule, and it is composed of several internal muxex for computation of the termination bit stream.

4.4.2.2 Block diagram and architecture

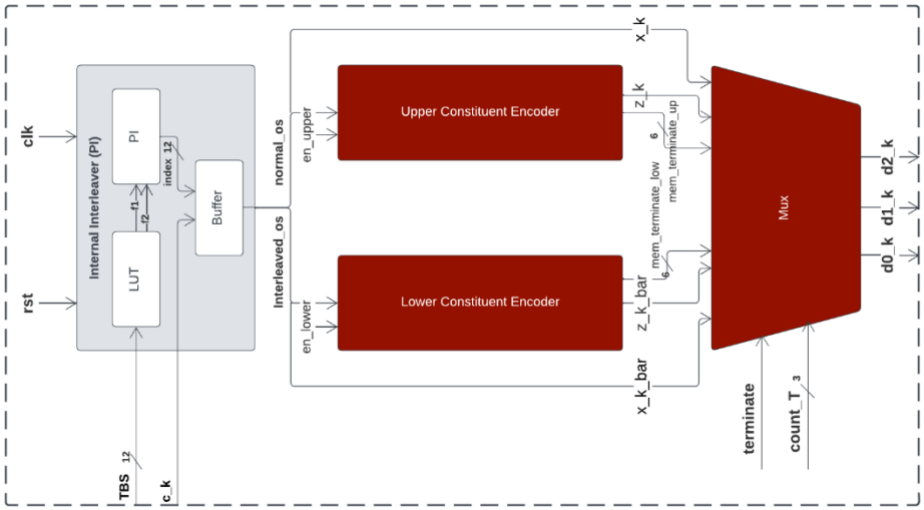

Figure 24: Turbo Encoder block diagram

4.4.2.3 Block interface

The following figure shows the interface of the Turbo Encoder block as implemented in the RTL,

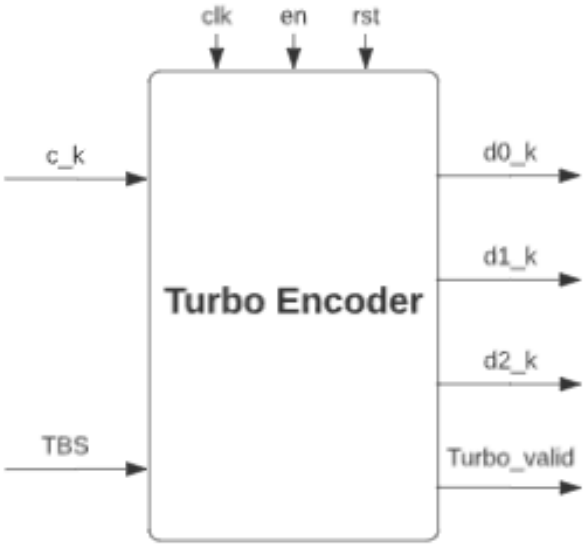

Figure 25: Turbo Encoder block interface

Table 21: Turbo Encoder interface signals

Signal	width	Port type	Description
clk	1 bit	input	System clock signal
rst	1 bit	input	Turbo Encoder reset signal
en	1 bit	input	Turbo Encoder enable signal.
TBS	12 bit	input	Transport block size
c_k	1 bit	input	Input stream (bit wise) to the Turbo encoder from CRC (length TBS+24)
d0_k	1 bit	Output	Systematic bit stream output from the Turbo encoder (bit wise)
d1_k	1 bit	Output	Parity 1 bit stream output from the Turbo encoder (bit wise)
d2_k	1 bit	Output	Parity 2 bit stream output from the Turbo encoder (bit wise)
Turbo_valid	1 bit	Output	Validation signal for Turbo_encoder output

4.4.2.4 Operation

The working scheme was made according to the following steps:

1) The Look Up Table (LUT) module reads the TBS from the system top level and accordingly it extracts the f1, f2 parameters.
2) The Pi module calculates the interleaved indices that will be stored in the buffer according to an optimized equation.
3) Then the internal buffer is used to extract two streams of bits, the first one is the normal output stream (normal_os) that was directly mapped from the input stream (c_k), and the second one is the interleaved output stream (interleaved_os) that was mapped according to the calculated interleaved indices.
4) Both streams are entered in parallel to the upper and lower constituent encoders for synchronization.
5) The streams are placed in three shift registers and some processing (XOR operations) is made for encoding them to be extracted as follows before entering the trellis termination mode:
    - x_k --> extarcted from the internal_interleaver buffer as the normal output stream (normal_os) directly from the input bit stream after CRC.

- x_k_bar --> extracted after passing through the turbo upper constituent encoder shift registers that has the normal bit stream (normal_os) as its input.
- z_k --> extarcted from the internal_interleaver buffer as the interleaved output stream (interleaved_os) directly from the interleaved bit stream after the internal inter-leaver.
- z_k_bar --> extracted after passing through the turbo lower constituent encoder shift registers that has the interleaved bit stream (interleaved_os) as its input.

6) The extracted four streams (in parallel) are then padded to a specified combination of the termination bits that are used to flush on the constituent encoders registers.
7) The padding sequence is made in order to formulate the turbo_encoder output as follows:
   - Systematic bit stream: d0_k --> x_k + trellis termination bits (4).
   - Parity1 bit stream: d1_k --> z_k + trellis termination bits (4).
   - Parity2 bit stream: d2_k --> z_k_bar+ trellis termination bits (4).

4.4.3 Rate Matching

4.4.3.1 Design

The design of the Rate Matching block is implemented based on using three memories for the subblock interleaver and a circular buffer. The specification for each component is done according to section 2.5.3.

4.4.3.2 Block diagram and architecture

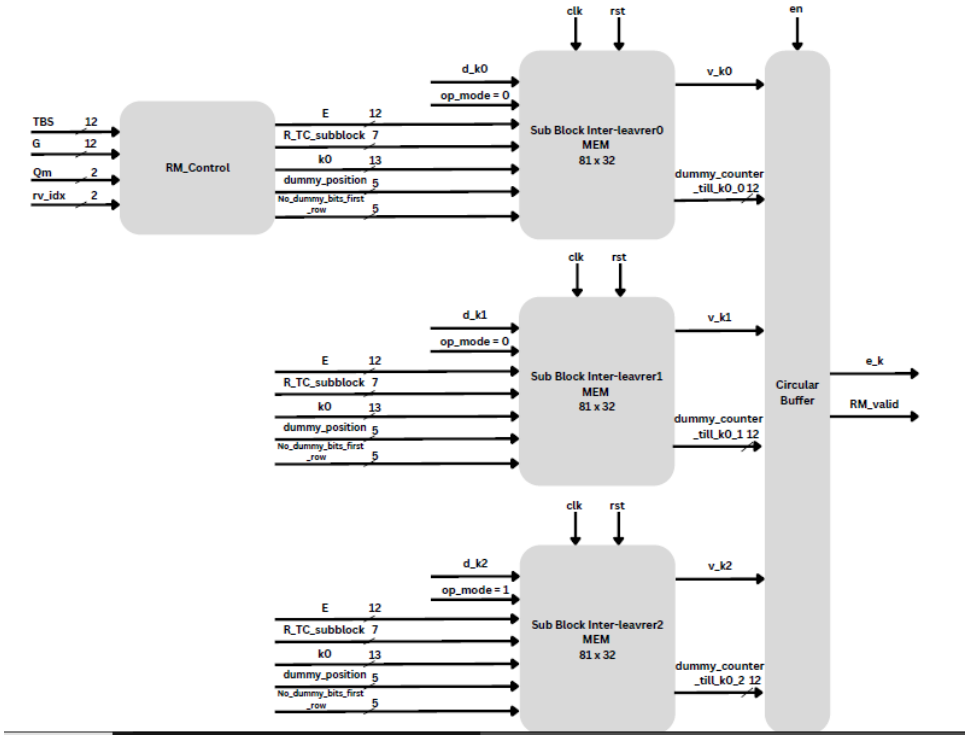

Figure 26: Rate Matching block diagram

4.4.3.3 Block interface

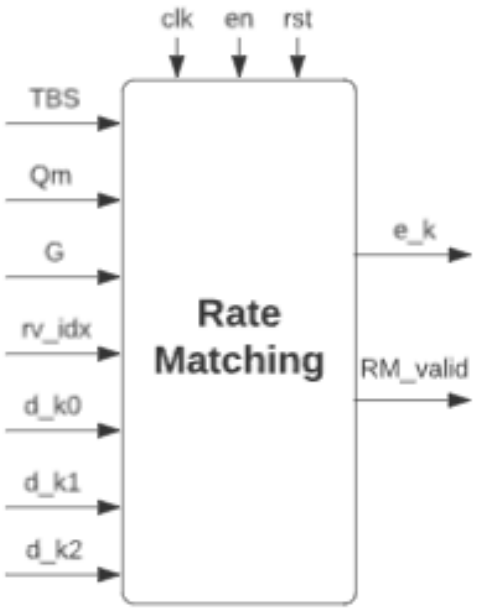

Figure 27: Rate Matching block interface

Table 22: Rate Matching interface signals

Signal	width	Port type	Description
clk	1 bit	Input	System clock signal
rst	1 bit	Input	Rate Matching reset signal
en	1 bit	Input	Rate Matching enable signal.
TBS	12 bit	Input	Transport block size
Qm	2 bit	Input	Modulation order: (type of modulation that will be used to send uplink): Qm = 1(BPSK) or 2(QPSK) Noting that in shared channel communication we use either BPSK or QPSK Modulation
G	12 bit	Input	The total number of bits available for the transmission of one transport block
rv_{idx}	2 bit	Input	The redundancy version index for the HARQ process (There exist four redundancy versions for each HARQ process 0,1,2,3)
d_{k0}	1 bit	Output	Input Stream 0: Systematic bits stream output from the Turbo Encoder
d_{k1}	1 bit	Output	Input Stream 1: Parity 1 bits stream output from the Turbo Encoder
d_{k2}	1 bit	Output	Input Stream 2: Parity 2 bits stream output from the Turbo Encoder
e_k	1 bit	Output	Output stream from the Rate Matching unit
RM_valid	1 bit	Output	Output validation signal (ON: output valid, OFF: output is invalid)

4.4.3.4 Operation

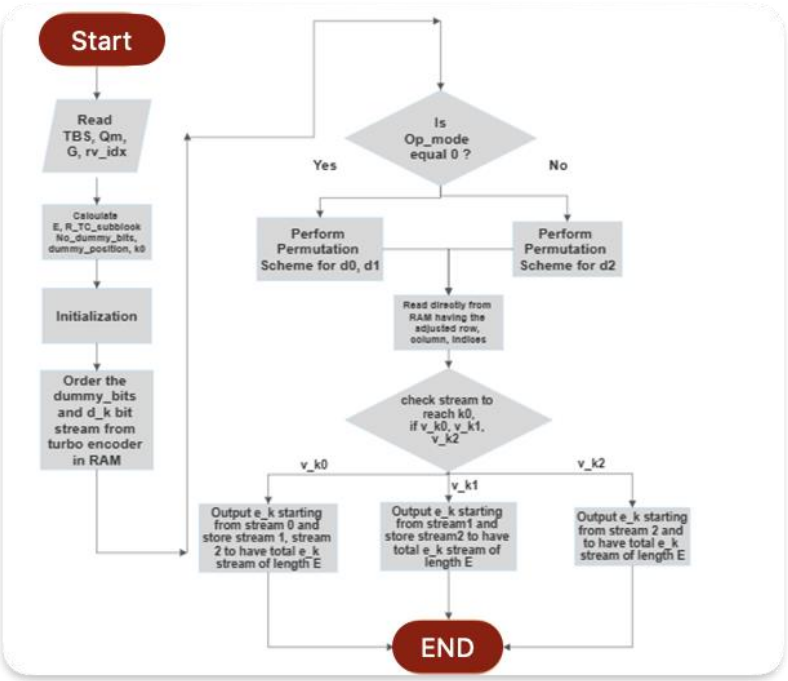

Figure 28: Rate Matching block operation

The working scheme was made according to the following steps:

1) RM_Control starts its action first by reading the following from the top module:

- Transport Block Size: TBS.
- Modulation order: Qm.
- The total number of bits available for transmission of one transport block: G.
- The redundancy version index: rv_idx.

Then accordingly it calculates

- Required length of the output from the Rate Matching: E.
- Required number of rows to order the output stream: R_TC_subblock in a matrix of 32 column.
- No of dummy bits that will be placed in the first row: no_dummy_bits_ first row.
- Index after which we start oredering the output stream from the turbo encoder: dummy_position.
- Starting point of the circular buffer: k0.

2) This is followed by an initialization state during the reset condition to all the required signals to allow proper permutation in the sub_block inter-leavers considering the truncated dummy bits conditions, in addition to storing the permutation table (table 5.1.4.1).
3) The third step is to start encoding the dummy bits according to the count and indices calculated from the control unit, in addition to ordering the turbo coded bits in a RAM of R_TC_subblock rows and 32 columns.
4) The fourth step is dependent on the type of input from the turbo encoder such that:

- For Systematic bit stream, and Parity 1 bit stream:

The corresponding sub_block interlever directly map the input bit stream to the output bit stream (denoted by v_k0/v_k1) considering truncating the dummy bits and accessing the permuted elements by indices control.

- For Parity 2 bit stream:

It calculates the respective indices mentioned by the equation stated in section 2.5.3.1 and directly map the calculated indices –considering truncating the dummy elements- to their corresponding input bit stream to be considered as the output from the sub_block_interleaver 2.

5) The output streams from the threes sub_block inter_leavers are extracted as three parallel streams which have outputs bit streams that are free of dummy_bits in addition to be inter-leaved according to its respective interleaving schemes, the top module directly access the stream at which starting point of the circular buffer is located then:

- If it was at v_k0 (the output stream from sub_block_interleaver 0), then the circular buffer stores the corresponding bit streams of v_k1, v_k2 at their respective location within the circular buffer. To be later used as the successive bits in case v_k0 stream was completely extracted and the required number length of the output (E) was not yet reached.
- If it was at v_k1 (the output stream from sub_block_interleaver 1), then the circular buffer stores the corresponding bit streams of v_k2 at their respective location within the circular buffer. To be later used as the successive bits in case v_k1 stream was completely extracted and the required number length of the output (E) was not yet reached.
- If it was at v_k2 (the output stream from sub_block_interleaver 2), then the output bit stream will start after k0 elemnts (considering the truncated dummies count till k0) till the length of the output (E) is reached.

4.4.4 Channel Interleaver

4.4.4.1 Design

The proposed design of the channel interleaver divides the block into 5 subblocks which are a control unit, two serial-to parallel shift registers, a parallel-to serial shift register and a register file. According to the control unit functionality as mentioned in section 2.5.4, multiplication and division operations are required to determine the number of rows and columns, into which the input bits are placed to be interleaved. To minimize the power, number of clock cycles, and the area of this subblock, the design proposes the utilization of shift registers to perform the division and multiplication instead of a multiplier and a divider. Moreover, a register file is used to hold the input bits instead of RAM to enhance the performance and the flexibility of the block, hence easing the retrieving of the bits out of it by columns as required in the channel interleaver functionality. Additionally, a load signal is added to the serial-to

parallel and parallel-to serial registers to be able to stall the input and perform the shifting only when needed. In order to control the flow of the input bit stream into and out from the channel interleaver block, different outputs are used to track the number of bits, number of rows, and number of columns. A flag signal (valid_out) is raised when the output bit stream is ready to go.

4.4.4.2 Block diagram and architecture

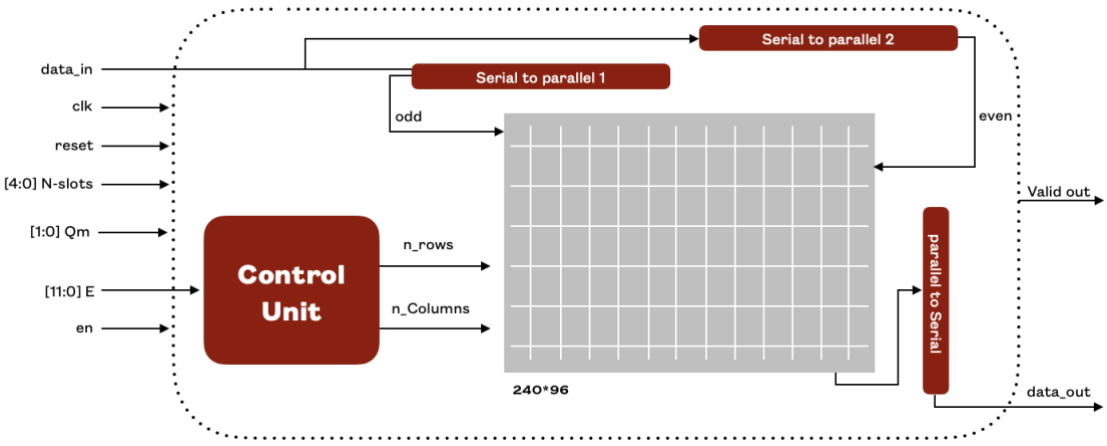

Figure 29: Channel Interleaver block diagram

4.4.4.3 Block interface

The following figure shows the interface of the Scrambler block as implemented in the RTL,

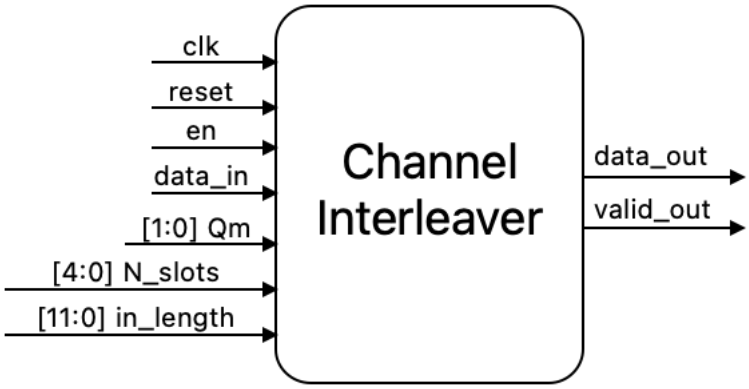

Figure 30: Channel Interleaver block interface

Table 23: Channel Interleaver interface signals

Signal	width	Port type	Description
clk	1 bit	Input	System clock signal
reset	1 bit	Input	Channel Interleaver reset signal
en	1 bit	Input	Channel Interleaver enable signal.
Q_m	2 bits	Input	Modulation order: Qm = 1(BPSK) or 2(QPSK)
data_in	1 bit	Input	Serial input from Rate Matching
in_length	16 bits	Input	The input length
N_slots	5 bits	Input	Number of slots (upper layer parameter)
data_out	1 bit	Output	Serial output going to the Scrambler
Valid_out	1 bit	Output	A valid output is ready

4.4.4.4 Operation

The operation of the channel interleaver starts with introducing the input bit stream besides the upper layer parameters needed to calculate the number of rows and columns according to the equations mentioned in section 2.5.4. The calculation is done in one clock cycles, then the interleaving is performed through the following steps,

a. Based on the modulation order,
    i. Q_m = 1, the input stream enters the first serial-to parallel register until a number of bits that is equal to the number of columns calculated by the control unit are added.
    ii. Q_m = 2, the input stream enters the two serial-to parallel registers alternatingly, where the even bits are added to the first serial-to parallel register and the odd bits are added to the other one.
b. When the number of bits inside the assigned serial-to parallel registers is equal to the number of columns calculated by the control unit, the parallel output is loaded onto a row in the register file.
c. The previous steps are repeated until the number of input bits is reached by the counter and the register file is filled with rows and columns that are equal to the numbers calculated by the control unit.
d. The columns of the register file are then loaded to the parallel-to serial register to generate the interleaved output stream.
e. The load of the parallel-to serial register is monitored, so that it reloads every number of rows clock cycles which assures that the previously loaded bits are out.

f. Steps d and e are repeated until the register file is empty and all the columns are read and retrieved by the parallel-to serial register.

4.4.5 Scrambler

4.4.5.1 Design

The design of the Scrambler block is implemented based on an LFSR, linear feedback shift register, in which the feedback is introduced to the registers by XORing the output signal with the input according to section 2.5.5.

4.4.5.2 Block diagram and architecture

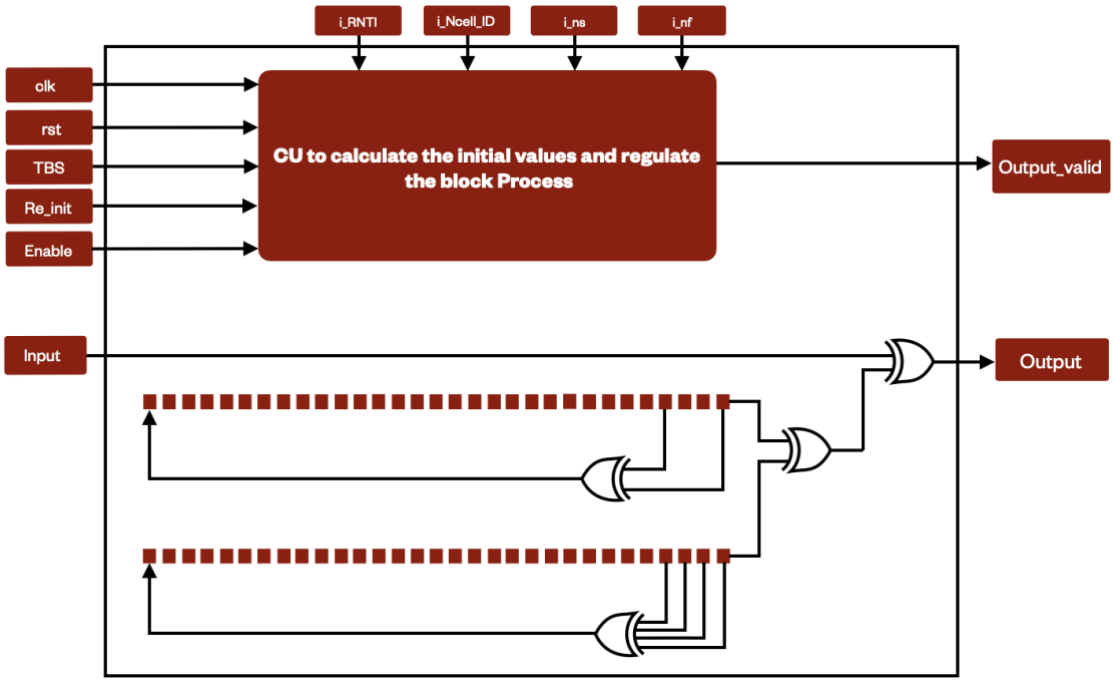

Figure 31: Scrambler block diagram

Functions of each unit:

1) Control unit: the control unit is designed to control the process of the scrambler and its combinational logic that is used to calculate the initialization value for each LFSR.

2) LFSR: two registers having a length of 31 bit that are used to generate the golden sequence for the scrambler's process.

4.4.5.3 Block interface

The following figure shows the interface of the Scrambler block as implemented in the RTL,

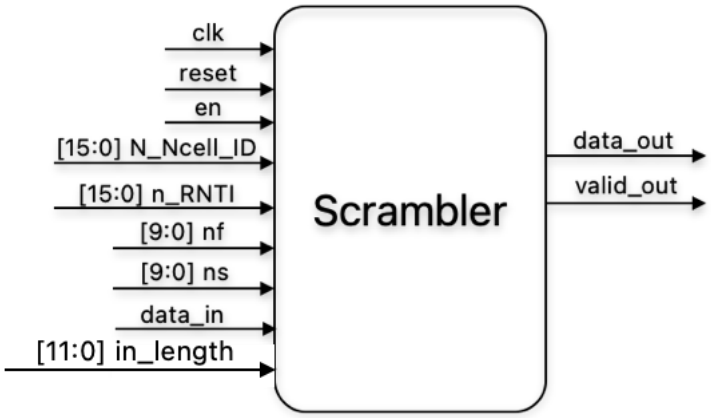

Figure 32: Scrambler block interface

Table 24: Scrambler interface signals

Signal	width	Port type	Description
clk	1 bit	Input	System clock signal
reset	1 bit	Input	Scrambler reset signal
en	1 bit	Input	Scrambler enable signal.
n_{RNTI}	16 bits	Input	Radio Network Temporary Identifier (upper layer parameter)
n_f	10 bits	Input	System Frame Number (upper layer parameter)
$n_s \in (0, 19)$	10 bits	Input	Slot Number Within Radio Frame (upper layer parameter)
data_in	1 bit	Input	Serial input from Channel Interleaver
In_length	12 bits	Input	The input length
$N_{ID}^{Ncell} \in (0, 503)$	16 bits	Input	Narrowband Physical Layer Cell Identity (upper layer parameter)
data_out	1 bit	Output	Serial output going to the Modulator
valid_out	1 bit	Output	A valid output is ready

4.4.5.4 Operation

The operation of the scramble is shown in the following steps:

1) Calculate the initialization for both LFSRs according to section 2.5.5 after receiving the upper layer parameters needed and the output of the channel interleaver.
2) Perform 1600 shift cycles in order to increase the randomization of the sequence and generate the golden sequence.

3) After this, the last bit of each LFSR's golden sequence is taken to the scrambler module and are XORed together.
4) Finally, this value is XORed with input to produce the output of the scrambler along with a $valid_{out}$ signal.

4.4.6 Modulator

4.4.6.1 Design

The design of the Modulator block is implemented based on using LUTs for each modulation scheme, and then using a mux to decide which of them will be used according to the modulation number Q_m. The modulation for each scheme will be done according to section 2.5.6.

4.4.6.2 Block diagram and architecture

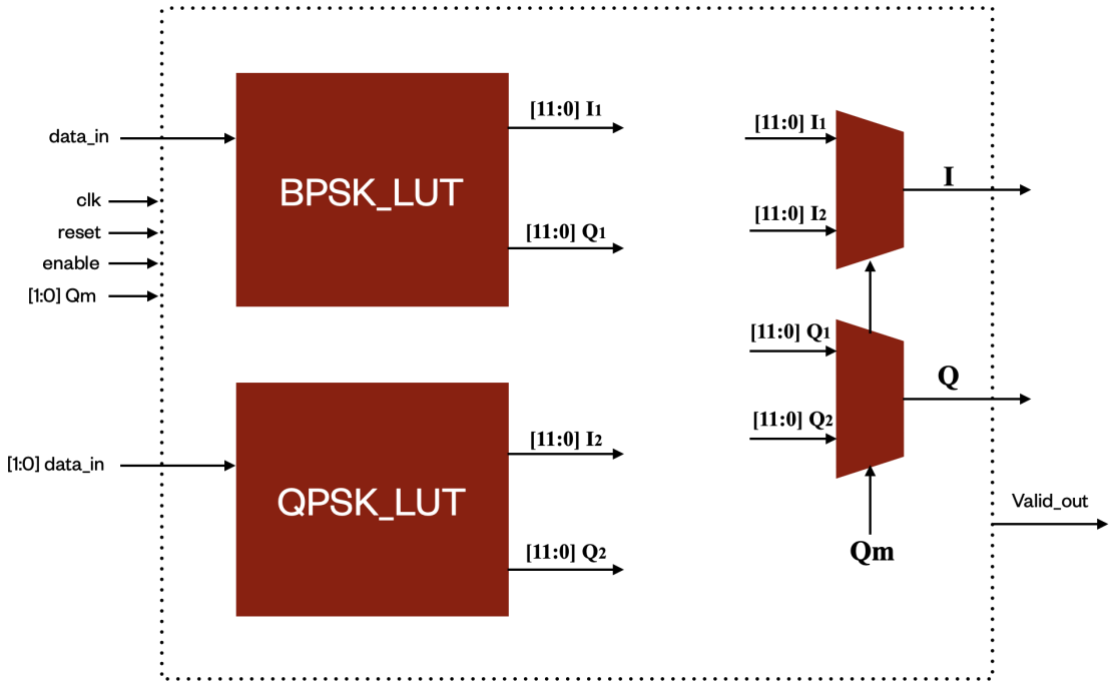

Figure 33: Modulator block diagram

4.4.6.3 Block interface

The following figure shows the interface of the Modulator block as implemented in the RTL,

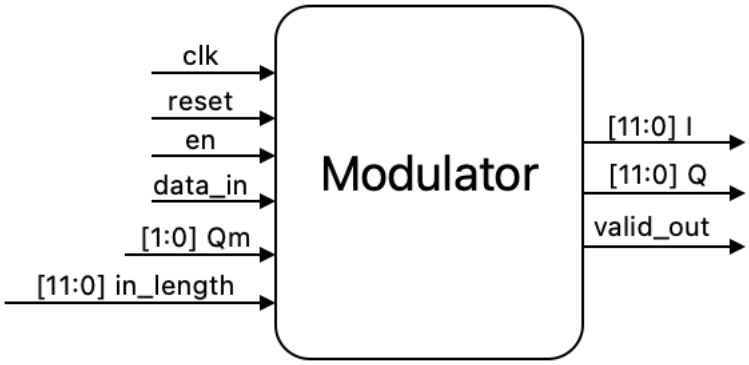

Figure 34: Modulator block interface

Table 25: Modulator interface signals

Signal	width	Port type	Description
clk	1 bit	Input	System clock signal
reset	1 bit	Input	Modulator reset signal
en	1 bit	Input	Modulator enable signal.
Q_m	2 bits	Input	Modulation number: Qm = 1(BPSK) or 2(QPSK)
data_in	1 bit/ 2bits	Input	Serial input from Channel Interleaver
in_length	16 bits	Input	The input length
I	12 bits	Output	Real part of the output going to FFT
Q	12 bits	Output	Imaginary part of the output going to FFT
Valid_out	1 bit	Output	A valid output is ready

4.4.6.4 Operation

The modulator is operated using two MUXs that are controlled by the value of Q_m. After deciding which modulation scheme to use, the corresponding LUT will be used to modulate each bit onto a carrier. The input width also depends on the modulation scheme as when using BPSK the input width is 1 bit, but when using QPSK the input width is 2 bits. The final output after the modulation will be divided into two parts, one for the real part of the output and the other for the imaginary part. Each output has a width of 12 bits to accommodate with the requirement of the next block which is the FFT.

4.4.7 FFT

4.4.7.1 Design

FFT is a widely used block in digital systems and has numerous implementations. The 12-point FFT is based on two main building blocks: the radix-2 and radix-3. The functionality of the DFT in this project requires it to cover 1-point FFT, 3-point FFT,6-point FFT, and 12-point FFT. Each is according to the number of subcarriers that are given as input to the block. In order to reduce the area, given that this block does not decide the frequency of the whole system, **a pipe-lined** architecture was implemented where an FSM controlled by the NSC (Number of Sub-Carriers) decides which type of FFT is required, and **one block for radix-2 and another one for radix-3**. then if it is 3 then it uses the radix-3 directly. If it is 6 then a pipe-lined radix-6 is used where the **resources** used for it are only one radix-3 and one radix-2 where the radix-3 operates two times to cover all the 6 inputs in the first stage then the outputs of this stage are assigned to the intermediate registers after being multiplied with the corresponding twiddling factors. These outputs are finally inputted to radix-2 where each two of them are calculated and the outputs are assigned to the corresponding output port then the radix-2 is used again for two times to get the outputs of the rest 4 intermediates. Similarly for the 12 NSC , it is designed to operate in 3 stages where the first stage requires the operation of the radix-3 for 4 times and in both the second and the third stages , radix-2 is used 6 times. The twiddling factor multiplications are performed using shift registers which greatly aided to save area and power.

4.4.7.2 Block diagram and architecture

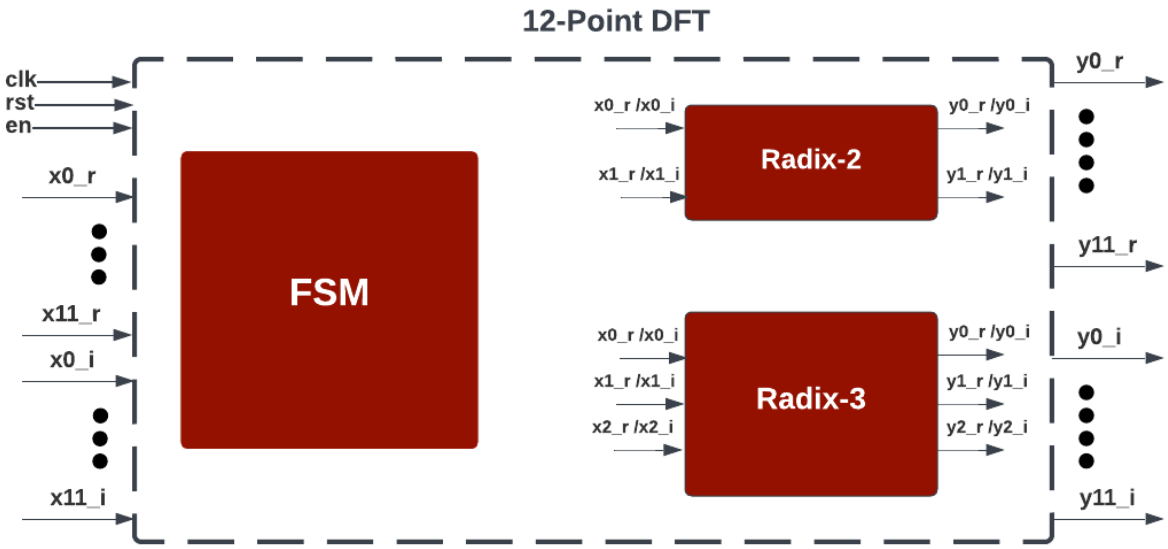

Figure 35: FFT block diagram

4.4.7.3 Block interface

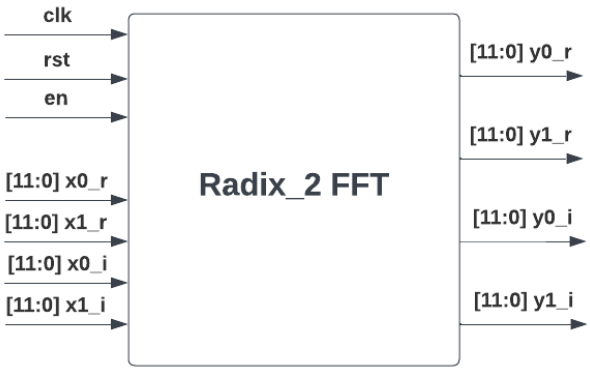

Figure 36: Radix_2 FFT block interface

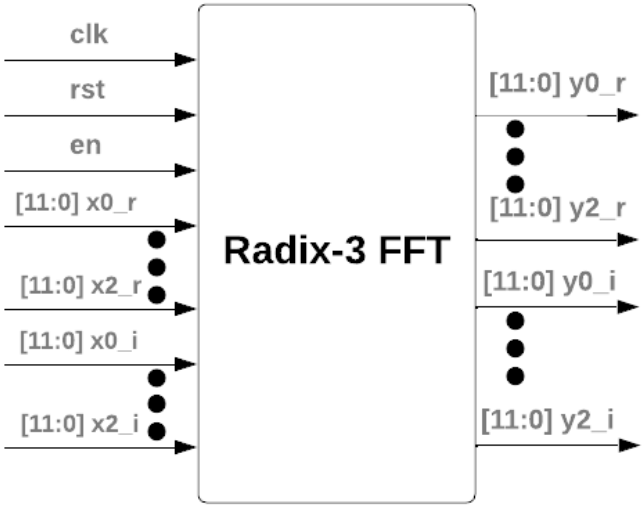

Figure 37: Radix_3 FFT block interface

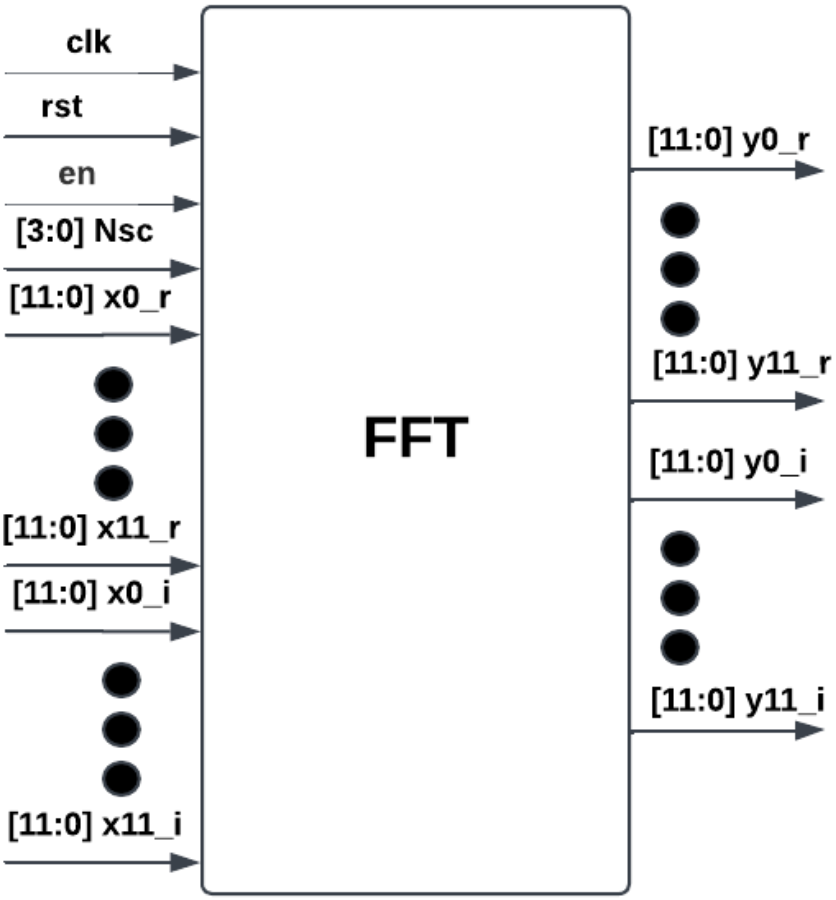

Figure 38: FFT block interface

Table 26: FFT interface signals

Signal	Width	Port type	Description
clk	1 bit	Input	System clock signal
rst	1 bit	Input	Reset signal
en	1 bit	Input	FFT enable signal.
NSC	4 bits	Input	Number of subcarriers
x0_r **:** **x11_r**	12 signed bits	Input	The real part of the 12 input signals Consists of 4 integer bits and 8 fraction bits.
x0_i **:** **x11_i**	12 signed bits	Input	The imaginary part of the 12 input signals Consists of 4 integer bits and 8 fraction bits.
y0_r **:** **y11_r**	12 signed bits	Output	The real part of the 12 output signals Consists of 4 integer bits and 8 fraction bits.
y0_i **:** **y11_i**	12 signed bits	Output	The imaginary part of the 12 output signals Consists of 4 integer bits and 8 fraction bits.

4.4.7.4 Operation

The operation of the block depends mainly on the number of subcarriers (NSC) where:

- NSC=1:

  In this case, the output is the same as the input.

- NSC=3:

  In this case, the inputs are directed to the radix-3 directly.

- NSC=6:

  In this case, a pipelined strategy is used to arrange the operation between the available resources which are the radix-2 and the radix-3 in the design.

  The diagram shown below shows the Radix-6 inherently implemented where the first stage of it requires the operation of radix-3 two times then the outputs are assigned to intermediate signals then in the second stage, the radix-2 operates 3 times using the intermediate signals as inputs to it.

- NSC=12:

  In this case, a pipelined strategy is followed where this case is composed of 3 stages that are shown in detail in the diagram below. The first stage requires the operation of radix-3 for 4 times to account for all the 12 inputs then the outputs of this stage are multiplied by their corresponding twiddling factors (in the order

shown below) and then saved in the intermediate signals. In both the second and third stages, radix-2 is used 6 times per stage.

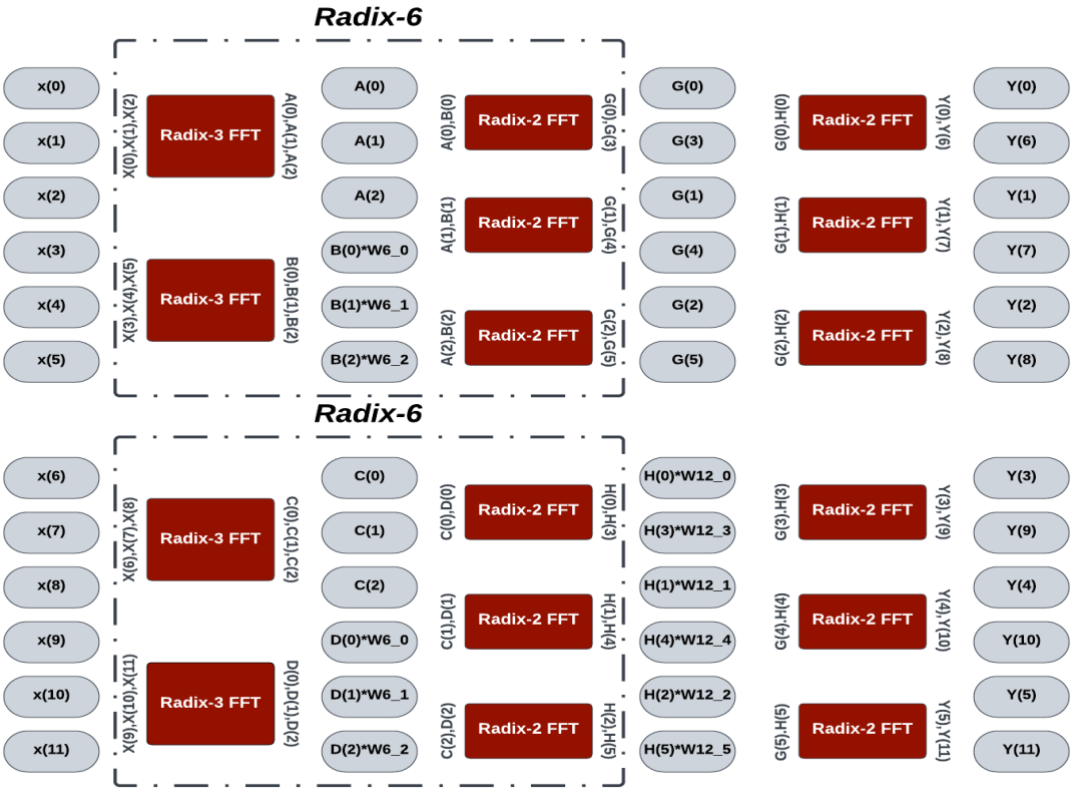

Figure 39: Operation of 12-point FFT including 6-point FFT

4.4.8 Resource Element Mapper

4.4.8.1 Design

The design of the REM block is implemented based on the creation of a resource grid using the output parameters from the control unit which are calculated according to section 2.5.8.

4.4.8.2 Block diagram and architecture

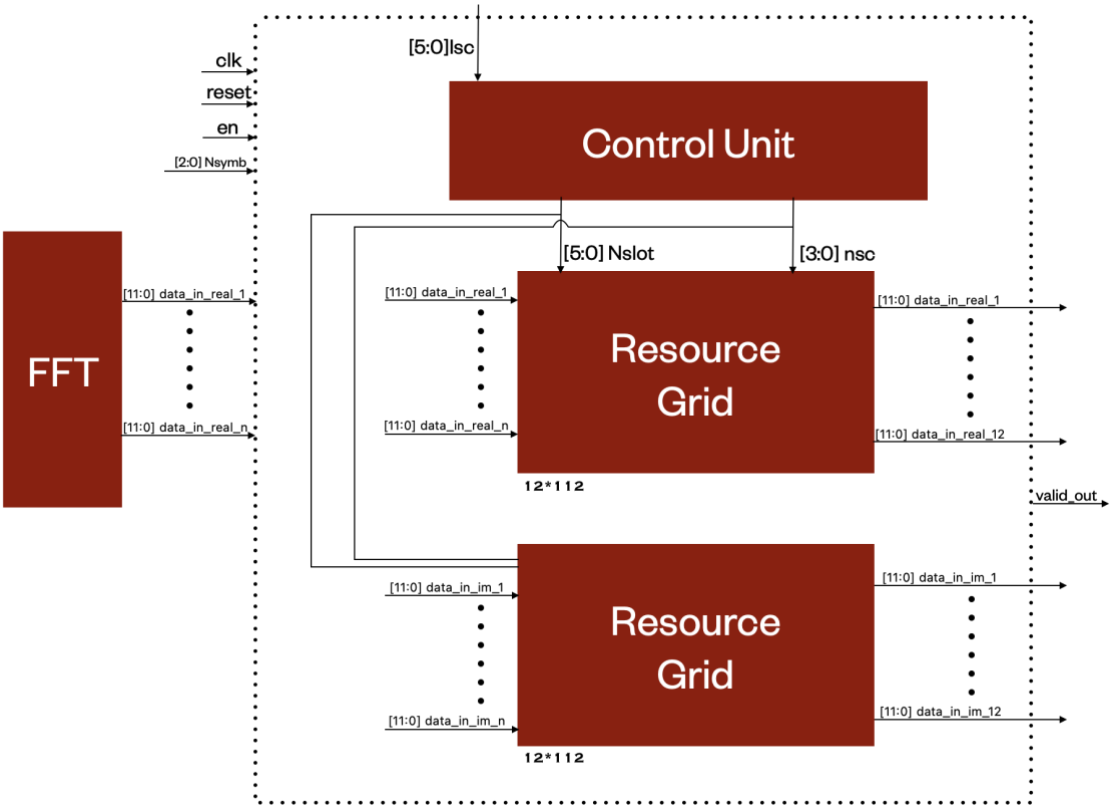

Figure 40: REM block diagram

4.4.8.3 Block interface

The following figure shows the interface of the REM block as implemented in the RTL,

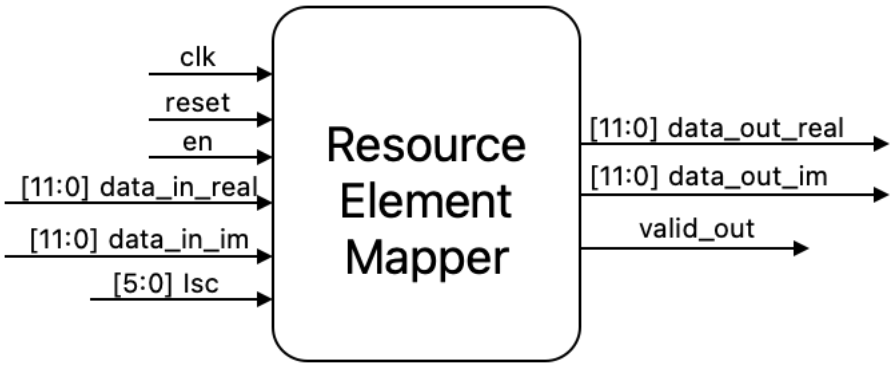

Figure 41: REM block interface

Table 27: REM interface signals

Signal	width	Port type	Description
clk	1 bit	Input	System clock signal
reset	1 bit	Input	REM reset signal
en	1 bit	Input	REM enable signal.
data_in_real	12 bits	Input	Real part from the input data coming from FFT
data_in_im	12 bits	Input	Imaginary part from the input data coming from FFT
N_{symb}	3 bits	Input	Number of SC-FDMA symbols
I_{sc}	6 bits	Input	Subcarrier indication field (upper layer parameter)
data_out_real	12 bits	Output	Real part of the output data going to IFFT
data_out_im	12 bits	Output	Imaginary part of the output data going to IFFT
valid_out	1 bit	Output	A valid output is ready

4.4.8.4 Operation

The REM block contains three modules:

1) The first module is the first memory which receives the real part of the input data coming from the FFT. The size of the memory is set to the maximum possible input which is 12*112. The maximum number of rows for the resource grid is 12 which is the maximum number of subcarriers. The maximum number of columns is calculated using $N_{symb} * N_{slots}$
2) The second module is the second memory which receives the imaginary part of the input data coming from the FFT. The size of the memory is set to the maximum possible input which is 12*112.
3) The third module is the control units which maps the output of the FFT to the assigned subcarriers. The output of this module is N_{slots} and n_{sc} value which specifies the row number where the value is stored.

After filling the memory with the input values, the remaining indices of the memory will be filled with zeros. Also, there is a certain column that is always reserved for the DMRS value which is an upper layer parameter.

4.4.9 IFFT

4.4.9.1 Design

The IFFT block has a variety of design methodologies to be implemented with. Its structure is based on the repetition of radix-2 blocks to construct the 7 stages of the 128-point IFFT. The design proposed below has a pipelined strategy that uses 16 radix-2 blocks to perform the 128-point IFFT functionality. This is implemented using a finite state machine that redirects the inputs and outputs of the 16 radix-2 blocks to cover the required computations. This design reduces the area of the IFFT block by shrinking the number of radix-2 blocks from 448, if all the used blocks are implanted and used once, to 16 radix-2 blocks. Another significant advantage of the proposed design is that the multiplications of the twiddle factors are fully performed using shift registers with pre-determined shift amounts and no multipliers are used. This minimizes the power consumption of the block, in addition to the area and the consumed clock cycles. The total number of clock cycles that are consumed by the 128-point IFFT is 28 clock cycles. The intermediate signals are limited to 128 real and imaginary signals to avoid the redundant use of signals, hence reduce the area, power and routing complexity.

4.4.9.2 Block diagram and architecture

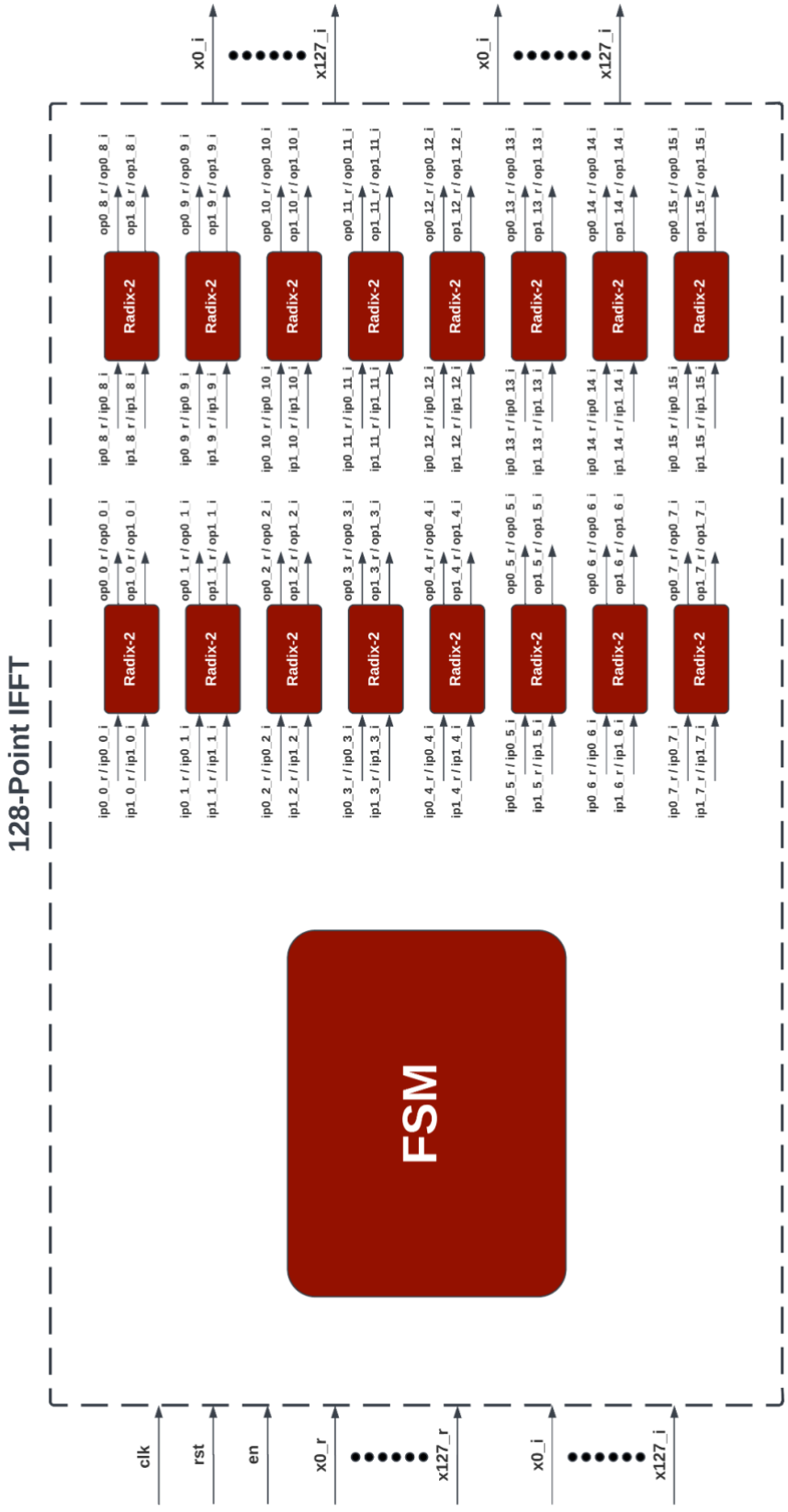

Figure 42: IFFT block diagram

4.4.9.3 Block interface

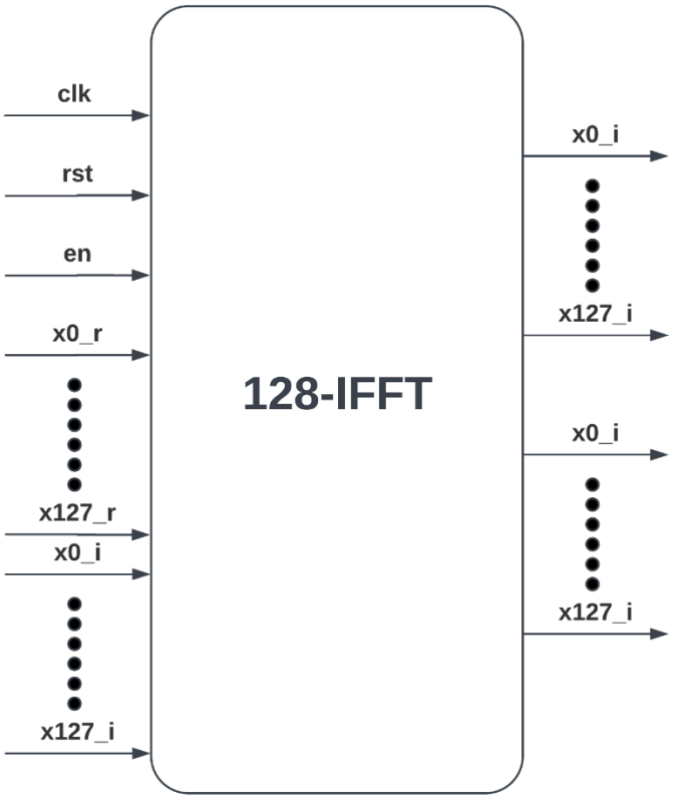

Figure 42: IFFT block diagram

Table 28: IFFT interface signals

Signal	Width	Port type	Description
clk	1 bit	Input	System clock signal
rst	1 bit	Input	Reset signal
en	1 bit	Input	IFFT enable signal.
x0_r ⋮ **x127_r**	14 signed bits	Input	Real part of the 128 input signals resulting from resource element mapper. Consists of 4 integer bits and 10 fraction bits.
x0_i ⋮ **x127_i**	14 signed bits	Input	Imaginary part of the 128 input signals resulting from resource element mapper. Consists of 4 integer bits and 10 fraction bits.
y0_r ⋮ **y127_r**	14 signed bits	Output	Real part of the 128 input signals resulting from resource element mapper. Consists of 4 integer bits and 10 fraction bits.
y0_i ⋮ **y127_i**	14 signed bits	Output	Imaginary part of the 128 input signals resulting from resource element mapper. Consists of 4 integer bits and 10 fraction bits.

4.4.9.4 Operation

The operation of this block is performed by a finite state machine that consists of 28 states. The states determine the 16 real inputs and 16 imaginary inputs to assign the inputs of the 16 radix-2 blocks. In addition, the states include the multiplication of the twiddle factors, using shifters, by the 16 real and imaginary outputs of the 16 radix-2 blocks before being assigned to the next intermediate signals. The twiddle factors are pre-calculated to ease the use of the low power shifting multiplication. The following figure shows the first three stages of the 7 stages of the 128-point IFFT. It shows the main strategy where the indices that are assigned together to the radix-2 blocks are divided by 2 every cycle, until the 7^{th} stage, in which every two consecutive indices are assigned to the same radix-2 block.

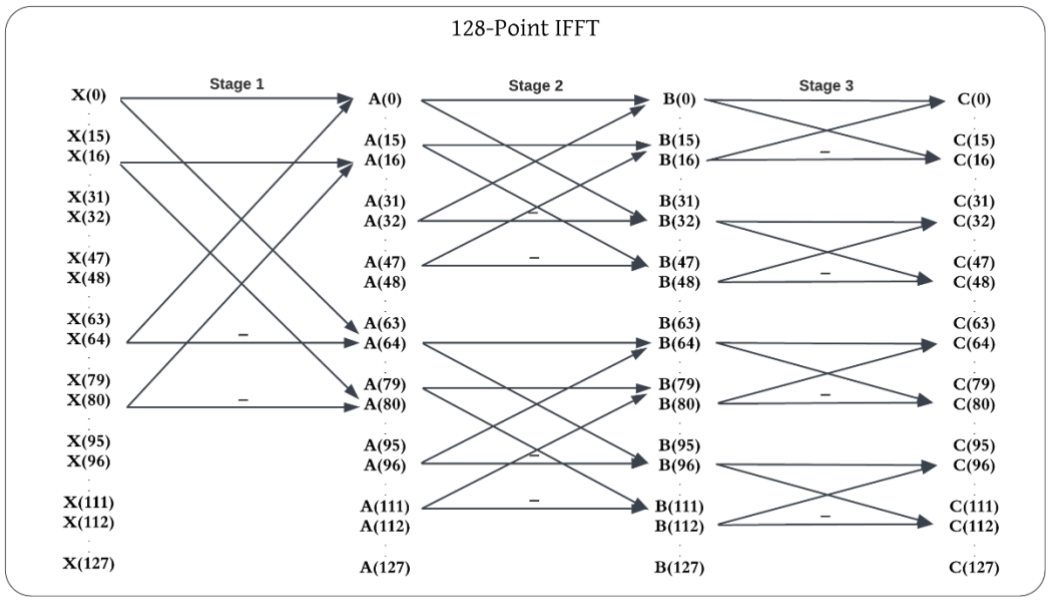

Figure 43: First 3 stages of 128-point IFFT

5 Project Execution

5.1 Simulation results and evaluation

The following subsections present the verification of the RTL blocks by conducting a comparison between the results and the reference model implemented using MATLAB. Then, the synthesis of the blocks is performed using Synopsys Design Compiler, the PnR is performed using IC Compiler, and the area, power and delay reports are reviewed to evaluate the efficiency of the proposed design using 45 nm

technology compared to previous implementations using 130 nm technology as in [11] and 45 nm technology as in [12]. The synthesis and PnR are done with a clock frequency of 765 kHz that corresponds to a clock period of 1.32 μs as in [11].

5.1.1 CRC

To verify the functionality of the CRC block, a testbench is used to give an initial insight about the correctness of the operation which results in the following waveform, where the output matches the output of the reference model implemented using MATLAB. It shows the output that consists of the input stream followed by the 24 bits of the generated CRC code.

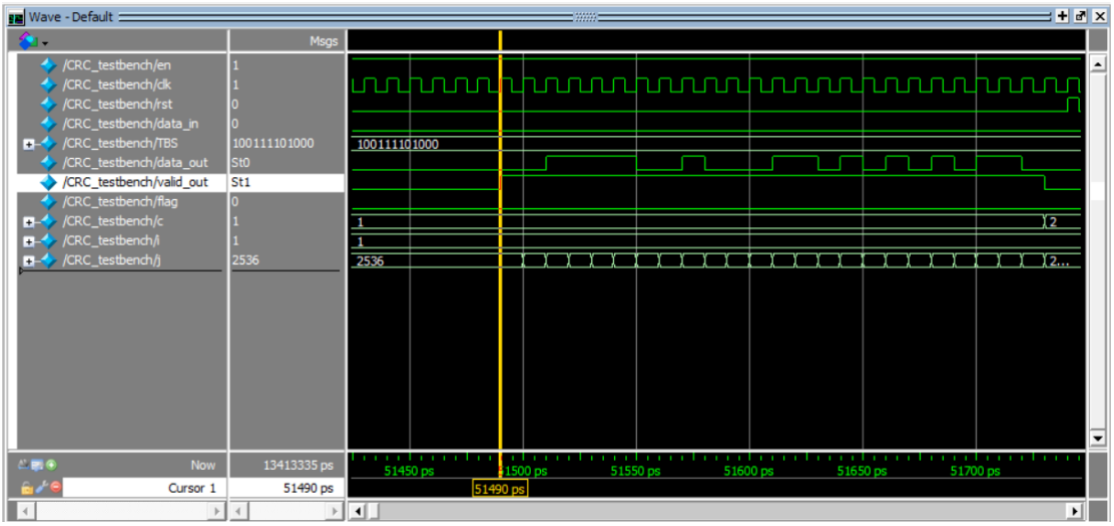

Figure 44: CRC waveform

5.1.1.1 MATLAB and Verilog Comparison

To increase the coverage of the applied test cases, 10 input test vectors are generated by MATLAB with a length of the maximum TBS, 2536, and applied to the input of the CRC block designed with the RTL to verify its functionality. The following figure shows that the proposed design matches the reference model successfully.

```
# Time:                258470  Correct CRC output is 0 and Ref output is 0
# Time:                258480  Correct CRC output is 1 and Ref output is 1
# Time:                258490  Correct CRC output is 0 and Ref output is 0
# Time:                258500  Correct CRC output is 0 and Ref output is 0
# Time:                258510  Correct CRC output is 0 and Ref output is 0
# Time:                258520  Correct CRC output is 1 and Ref output is 1
# Time:                258530  Correct CRC output is 0 and Ref output is 0
# Time:                258540  Correct CRC output is 1 and Ref output is 1
# Time:                258550  Correct CRC output is 1 and Ref output is 1
# Time:                258560  Correct CRC output is 1 and Ref output is 1
# Time:                258570  Correct CRC output is 1 and Ref output is 1
# Time:                258580  Correct CRC output is 1 and Ref output is 1
# Time:                258590  Correct CRC output is 1 and Ref output is 1
# Time:                258600  Correct CRC output is 1 and Ref output is 1
# Time:                258610  Correct CRC output is 1 and Ref output is 1
# Time:                258620  Correct CRC output is 1 and Ref output is 1
# Time:                258630  Correct CRC output is 1 and Ref output is 1
# Time:                258640  Correct CRC output is 0 and Ref output is 0
# Time:                258650  Correct CRC output is 0 and Ref output is 0
# Time:                258660  Correct CRC output is 1 and Ref output is 1
# Time:                258670  Correct CRC output is 0 and Ref output is 0
# Time:                258680  Correct CRC output is 0 and Ref output is 0
# Time:                258690  Correct CRC output is 0 and Ref output is 0
# Time:                258690  END OF TEST VECTOR NO. 10
# Time:                258690  SUCCESSFUL 10 TEST VECTORS OUT OF 10
```

Figure 45: RTL results matched with MATLAB for CRC

5.1.1.2 Synthesis and pnr results

5.1.1.2.1 Setup Time

```
---------------------------------------------------------------------------
data required time                                                1319.58
data arrival time                                                   -0.93
---------------------------------------------------------------------------
slack (MET)                                                       1318.65
```

Figure 46: CRC setup time result

5.1.1.2.2 Area

```
Combinational area:       172.900001
Buf/Inv area:              11.438000
Noncombinational area:    206.947993
Net Interconnect area:     undefined  (Wire load has zero net area)

Total cell area:          379.847993
Total area:                undefined
1
```

Figure 47: CRC Area

The area report shows that the area of the synthesized CRC block is 379.85 μm^2 which is smaller than the reported area value of 2680.52 μm^2 in [11] and the reported area value of 452.73 μm^2 in [12]

5.1.1.2.3 Power

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	1.4290e-02	3.1200e-02	23.7432	6.9233e-02	(3.47%)	
register	0.1589	2.4806e-03	720.6918	0.8820	(44.17%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	9.0935e-03	1.1301e-02	1.0251e+03	1.0455	(52.36%)	
Total	0.1822 uW	4.4982e-02 uW	1.7695e+03 nW	1.9967 uW		

1

Figure 48: CRC power

The power report resulting from Design Compiler shows that the power of the synthesized CRC block is 1.9967 μW which is smaller than the reported power value of 20 μW in [11] and the reported power value f 2.1628 μW in [12]

5.1.2 Turbo Coding

5.1.2.1 MATLAB and Verilog Comparison

Test case:

For testing, an input stream to MATLAB and MODELSIM was used taking TBS=16, thus K= 40. Input length = 40, as follows:

```
c = [1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0
1 0 1 1];
```

The same input stream was encoded for both modules of the Turbo_encoder in MATLAB and MODELSIM, and the output streams are mapped as follows:

- MATLAB: d0,d1,d2 as the output streams from the turbo encoder, representing Systematic bit stream, Parity 1 bit stream, and Parity 2 bit stream, respectively.
- MODELSIM: d0_v, d1_v, d2_v as the output streams from the turbo encoder, representing Systematic bit stream, Parity 1 bit stream, and Parity 2 bit stream, respectively.

The comparison was made at the MATLAB by comparing the generated output file from RTL model, and the MATLAB model. The two models show a matched output indicating that the RTL design is properly verified by the corresponding behavioral reference model. This is indicated by printing a “MATCHED!!” flag as shown below.

For d0_k, and d1_k:

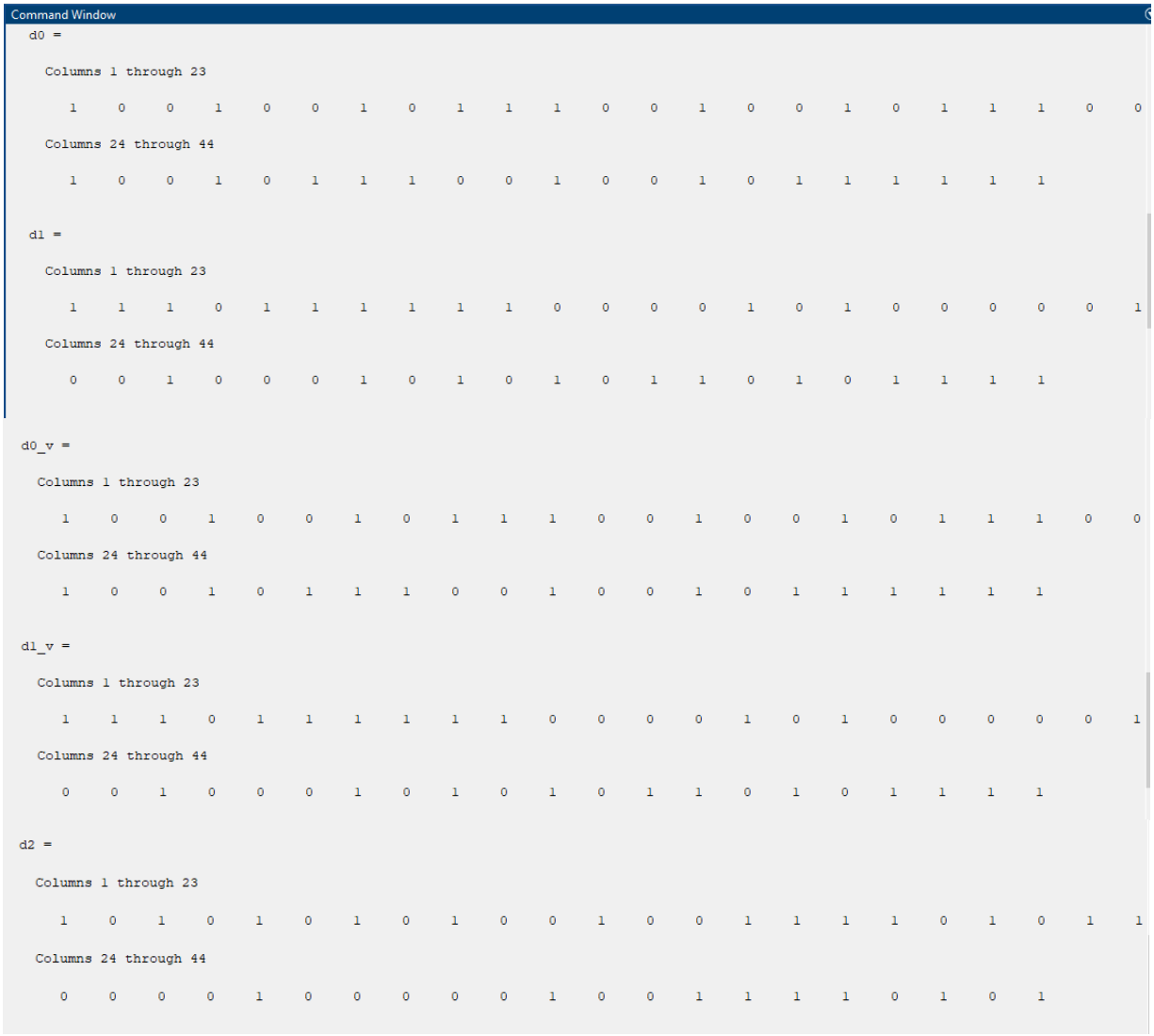

Figure 49: RTL results matched with MATLAB for Turbo Encoder

5.1.2.2 Synthesis and pnr results

5.1.2.2.1 Time

data required time	1319.61
data arrival time	-2.86
slack (MET)	1316.76

Figure 50: Turbo Encoder setup time result

5.1.2.2.2 Area

```
Combinational area:       4596.213989
Buf/Inv area:              150.822000
Noncombinational area:    9184.181695
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          13780.395684
Total area:                 undefined
1
```

Figure 51: Turbo Encoder area

The area report shows that the area of the synthesized Turbo Encoder block is 13,780.4 μm^2 which is smaller than the reported area value of 155,050 μm^2 in [11] and the reported area value of 26,257.13 μm^2 in [12]

5.1.2.2.3 Power

```
                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network      0.4806        4.1930e-02          1.7658e+03            2.2883  (   3.95%)
register           0.3561        1.8653e-02          3.1276e+04           31.6511  (  54.63%)
sequential         0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational   7.6477e-02          0.1277           2.3788e+04           23.9926  (  41.42%)
--------------------------------------------------------------------------------------------------
Total              0.9132 uW        0.1882 uW        5.6831e+04 nW        57.9320 uW
1
```

Figure 52: Turbo Encoder power

The power report shows that the power of the synthesized Turbo Encoder block is 57.93 μW which is smaller than the reported power value of 4 mW in [11] and the reported power value of 125.71 μW in [12]

5.1.2.2.4 Final chip

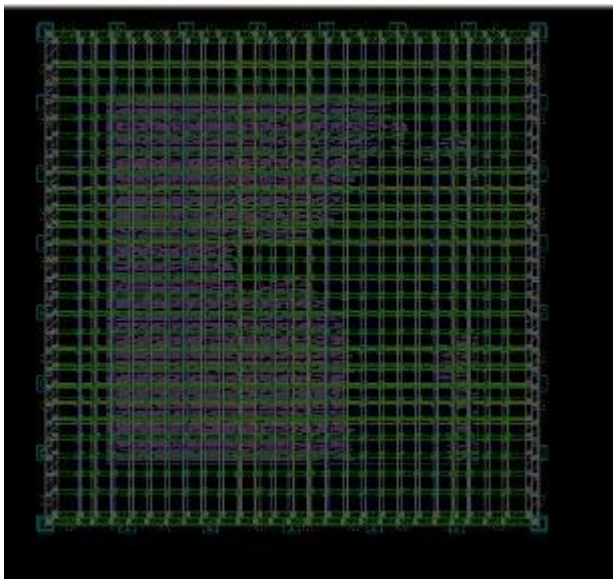

Figure 53: Turbo Encoder final chip after pnr

5.1.2.3 Comments

Optimization:

- The calculation of the interleaving function was greatly optimized by being considered as a recurrency equation at which the calculation is made according to this simplified version of the equation:

$$Pi(i+1) = pi(i) + \Delta pi(i)$$

This equation is dependent only on arithmetic operations avoiding the use of multipliers which greatly reduces the power consumption and area usage of the block.

- The parallel mechanism greatly enhances the system speed, such that the proper output is directly extracted after only two clk cycles from the input encoding with no need of wasting additional clk cycles for further calculations since all the needed calculations are performed in parallel with the input encoding.

5.1.3 Rate Matching

5.1.3.1 MATLAB and Verilog Comparison

Test case:

For testing, an input stream to MATLAB and MODELSIM was used taking TBS=16, thus K= 40, Qm = 1 (BPSK modulation), rv_idx = 2, G (expected RM output length: length (e_k)). Input length = 44 for the three assumed output streams from the turbo encoder (considering the padded trellis termination bits for each stream), as follows:

```
d0 = [1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0
0 1 0 1 1 1 1 0 1];
d1 = [1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0
0 1 0 1 1 1 1 0 1];
d2 = [1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 0
0 1 0 1 1 1 1 0 1];
```

The same input stream was encoded for both modules of the RM in MATLAB and MODELSIM, and the output streams are mapped as follows:

- MATLAB: e_k as the output stream from the Rate matching, representing the bit stream available for transmission of one transport block.

- MODELSIM: e_v as the output streams from the Rate matching, representing the bit stream available for transmission of one transport block.

The comparison was made at the MATLAB by comparing the generated output file from RTL model, and the MATLAB model. The two models show a matched output indicating that the RTL design is properly verified by the corresponding behavioral reference model. This is indicated by printing a “MATCHED!!” flag, further more the output length is 24 bits as it was designed from G system level parameter (that indicates the number of bits available for transmission of one transport block) as shown below.
*It is worth noting that in future work this model must be further tested for several G values.

Figure 54: RTL results matched with MATLAB for Rate Matching

5.1.3.2 Synthesis and pnr results

Initial estimation for the Synthesis results

5.1.3.2.1 Time

Figure 55: Rate Matching setup time result

5.1.3.2.2 Area

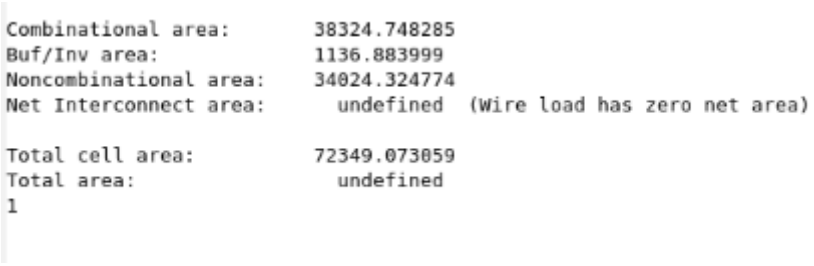

Figure 56: Rate Matching area

The area report shows that the area of the synthesized Rate Matching block is 72,349.1 μm^2 which is smaller than the reported area value of 489,458 μm^2 in [11] and the reported area value of 99,299 μm^2 in [12]

5.1.3.2.3 Power

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.1476	4.6619	348.5646	5.1581	(1.40%)	
register	19.6846	7.7441e-02	1.1999e+05	139.7482	(38.06%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	0.8488	1.6145	2.1979e+05	222.2580	(60.53%)	
Total	20.6811 uW	6.3538 uW	3.4013e+05 nW	367.1643 uW		

1

Figure 57: power

The power report shows that the power of the synthesized Rate Matching block is 367.164 μW which is smaller than the reported power value of 3.25 mW in [11] and the reported power value of 388.54 μW in [12]

5.1.3.3 Comments

Optimization:

1) In step 4 (for systematic bit stream and parity 1 bit streams) in the design mentioned in section 4.4.3.4:

   The output stream from the turbo_encoder is directly mapped to the already stored input stream and dummies in RAM, considering truncating the dummy bits and accessing the interleaved bits after permutation by indices control. This allows having only one memory for each sub_block interleaver which allows reducing the power and area used.

2) In step 4 (for parity 2 bit stream) in the design mentioned in section 4.4.3.4:

   The permutation equation pi(k) was traced to depend only on two registers values: (inner MOD result, and Floor_result), even more their values have certain pattern that was traced to be implemented with the minimal number of needed calculations using shift registers and finite state machines avoiding any multiplication operations or Modulus that will need an additional multipliers

and other units which will add to the required area, increase the used power, in addition to reducing the block speed.

3) In step 5: The size of the circular buffer was reduced by one third of its original value; due to directly accessing the starting point k0 at the respective output stream.

5.1.4 Channel Interleaver

To verify the functionality of the Channel Interleaver, a testbench is used to give an initial insight about the correctness of the operation which results in the following waveform, where the output matches the output of the reference model implemented by MATLAB. It shows the output that represent the input bits after being shifted in the serial-to parallel register and read from the register file by columns after being placed by rows.

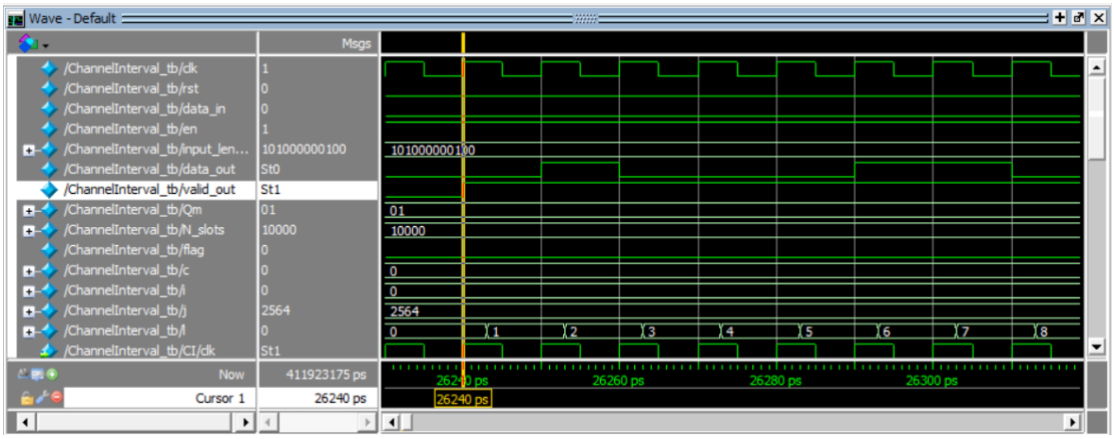

Figure 58: Channel Interleaver waveform

5.1.4.1 MATLAB and Verilog Comparison

To increase the coverage of the applied test cases, 30 input test vectors are generated by MATLAB with a length of 2564 as an example of a typical input to the Channel Interleaver block according to the TBS values, the added bits by the CRC and the Turbo Encoder, and the number of rate matching output bits. These test vectors are applied to the input of the Channel Interleaver block designed with the RTL to verify its functionality. The following figure shows that the proposed design perfectly matches the reference model.

```
# Time:              1564513  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564523  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564533  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564543  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564553  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564563  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564573  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564583  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564593  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564603  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564613  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564623  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564633  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564643  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564653  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564663  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564673  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564683  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564693  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564703  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564713  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564723  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564733  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564743  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564753  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564763  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564773  Correct Channel Interleaver output is 1 and Ref output is 1
# Time:              1564783  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564793  Correct Channel Interleaver output is 0 and Ref output is 0
# Time:              1564793  END OF TEST VECTOR NO. 30
# Time:              1564793  SUCCESSFUL 30 TEST VECTORS OUT OF 30
```

Figure 59: RTL results matched with MATLAB for Channel Interleaver

5.1.4.2 Synthesis and pnr results

5.1.4.2.1 Time

```
---------------------------------------------------------------------------
data required time                                              1319.58
data arrival time                                                -2.21
---------------------------------------------------------------------------
slack (MET)                                                     1317.38
```

Figure 60: Channel Interleaver setup time result

5.1.4.2.2 Area

```
Combinational area:       3968.454089
Buf/Inv area:              138.053998
Noncombinational area:    16046.183426
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          20014.637515
Total area:                 undefined
1
```

Figure 61: Channel Interleaver area

The area report shows that the area of the synthesized Channel Interleaver block is 20,014.638 μm^2, which is smaller than the reported area value of 440,585 μm^2 in [11] and the reported area value of 953.61 μm^2 in [12].

5.1.4.2.3 Power

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.1708	0.2178	527.5815	0.9163	(1.19%)	
register	0.9227	3.9366e-03	5.8342e+04	59.2690	(76.77%)	
sequential	9.6418e-04	8.6484e-04	46.9351	4.8764e-02	(0.06%)	
combinational	5.9692e-02	0.1106	1.6803e+04	16.9731	(21.98%)	
Total	1.1542 uW	0.3332 uW	7.5720e+04 nW	77.2071 uW		

1

Figure 62: Channel Interleaver power

The power report shows that the power of the synthesized Channel Interleaver block is 77.21 μW which is smaller than the reported power value of 7 mW in [11].

5.1.5 Scrambler

To verify the functionality of the Scrambler block, a testbench is used to give an initial insight about the correctness of the operation which results in the following waveform, where the output matches the output of the reference model implemented by MATLAB. It shows the output after the first 1600 cycles of the LFSR of the scrambler and the input shifting and XORing clock cycles.

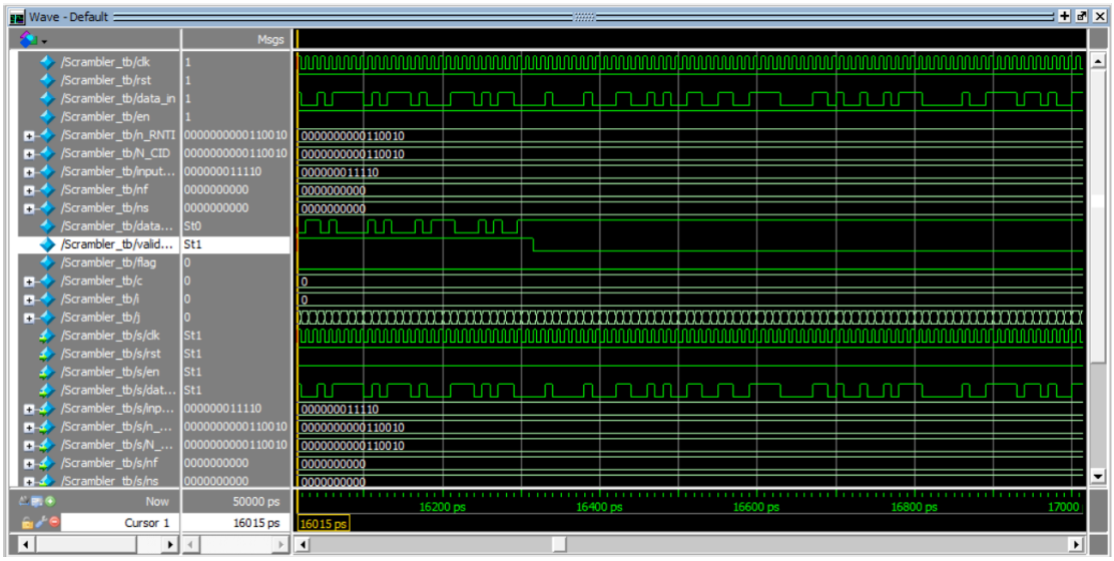

Figure 63: Scrambler waveform

5.1.5.1 MATLAB and Verilog Comparison

To increase the coverage of the applied test cases, 30 input test vectors are generated by MATLAB with a length if 2564 as a typical allowed input length to the Scrambler. The 30 test vectors are then applied to the input of the Scrambler block designed with RTL to verify its functionality. The resulting output is compared with the reference model after the initial 1600 clock cycles at the beginning of the scrambler operation. The following figure shows that the proposed design matches the reference model successfully.

```
# Time:              776160  Correct Scrambler output is 0 and Ref output is 0
# Time:              776170  Correct Scrambler output is 1 and Ref output is 1
# Time:              776180  Correct Scrambler output is 0 and Ref output is 0
# Time:              776190  Correct Scrambler output is 0 and Ref output is 0
# Time:              776200  Correct Scrambler output is 0 and Ref output is 0
# Time:              776210  Correct Scrambler output is 1 and Ref output is 1
# Time:              776220  Correct Scrambler output is 0 and Ref output is 0
# Time:              776230  Correct Scrambler output is 0 and Ref output is 0
# Time:              776240  Correct Scrambler output is 1 and Ref output is 1
# Time:              776250  Correct Scrambler output is 1 and Ref output is 1
# Time:              776260  Correct Scrambler output is 1 and Ref output is 1
# Time:              776270  Correct Scrambler output is 0 and Ref output is 0
# Time:              776280  Correct Scrambler output is 0 and Ref output is 0
# Time:              776290  Correct Scrambler output is 1 and Ref output is 1
# Time:              776300  Correct Scrambler output is 1 and Ref output is 1
# Time:              776310  Correct Scrambler output is 0 and Ref output is 0
# Time:              776320  Correct Scrambler output is 0 and Ref output is 0
# Time:              776330  Correct Scrambler output is 1 and Ref output is 1
# Time:              776340  Correct Scrambler output is 0 and Ref output is 0
# Time:              776350  Correct Scrambler output is 0 and Ref output is 0
# Time:              776360  Correct Scrambler output is 1 and Ref output is 1
# Time:              776370  Correct Scrambler output is 0 and Ref output is 0
# Time:              776380  Correct Scrambler output is 1 and Ref output is 1
# Time:              776390  Correct Scrambler output is 0 and Ref output is 0
# Time:              776400  Correct Scrambler output is 1 and Ref output is 1
# Time:              776410  Correct Scrambler output is 1 and Ref output is 1
# Time:              776420  Correct Scrambler output is 0 and Ref output is 0
# Time:              776430  Correct Scrambler output is 1 and Ref output is 1
# Time:              776440  Correct Scrambler output is 1 and Ref output is 1
# Time:              776450  Correct Scrambler output is 0 and Ref output is 0
# Time:              785790  END OF TEST VECTOR NO. 30
# Time:              785790  SUCCESSFUL 30 TEST VECTORS OUT OF 30
```

Figure 64: RTL results matched with MATLAB for Scrambler

5.1.5.2 Synthesis and pnr results

5.1.5.2.1 Time

```
-------------------------------------------------------------
data required time                                   1319.61
data arrival time                                      -0.65
-------------------------------------------------------------
slack (MET)                                          1318.96
```

Figure 65: Scrambler setup time result

5.1.5.2.2 Area

```
Combinational area:         63.840000
Buf/Inv area:                5.852000
Noncombinational area:      71.819998
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:           135.659998
Total area:                 undefined
1
```

Figure 66: Scrambler area

The area report shows that the area of the synthesized Scrambler block is 135.66 μm^2 which is smaller than the reported area value of 2802 μm^2 in [11] and the reported area value of 327.712 μm^2 in [12]

5.1.5.2.3 Power

```
                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network   7.4546e-03       1.3455e-02             11.8261         3.2735e-02  (   4.23%)
register        5.3570e-02       1.6963e-03            246.4951             0.3018  (  38.96%)
sequential         0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational   2.9789e-03       3.3976e-03            433.7412             0.4401  (  56.82%)
--------------------------------------------------------------------------------------------------
Total           6.4004e-02 uW    1.8548e-02 uW       692.0625 nW          0.7746 uW
1
```

Figure 67: Scrambler power

The power report shows that the power of the synthesized Scrambler block is 0.7746 μW which is smaller than the reported power value of 254 μW in [11] and the reported power value of 1.2976 μW in [12]

5.1.6 Modulator

The modulator is verified by introducing data_in input stream and comparing the outputs I and Q with MATLAB outputs. The test was performed for BPSK modulation, $Q_m = 1$, and QPSK modulation, $Q_m = 2$.

5.1.6.1 MATLAB and Verilog Comparison

The following figures show that the output of the modulator matches the output of MATLAB successfully in the case of BPSK and QPSK modulation types. Note the binary representation of the modulation values,

‘

Table 29: Binary representation of complex values used in Modulator

Decimal value	Binary value
$\frac{1}{\sqrt{2}}$	0000_10110101
$-\frac{1}{\sqrt{2}}$	1111_01001011

5.1.6.1.1 BPSK

1x16 complex fi

	1	2	3	4	5	6	7	8
1	0.7070 + 0.7070i	0.7070 + 0.7070i	-0.7070 - 0.7070i	0.7070 + 0.7070i	0.7070 + 0.7070i	-0.7070 - 0.7070i	-0.7070 - 0.7070i	-0.7070 - 0.7070i
2								

	9	10	11	12	13	14	15	16
1	0.7070 + 0.7070i	0.7070 + 0.7070i	-0.7070 - 0.7070i	0.7070 + 0.7070i	0.7070 + 0.7070i	-0.7070 - 0.7070i	-0.7070 - 0.7070i	-0.7070 - 0.7070i
2								

Figure 68: Modulator output for BPSK using MATLAB

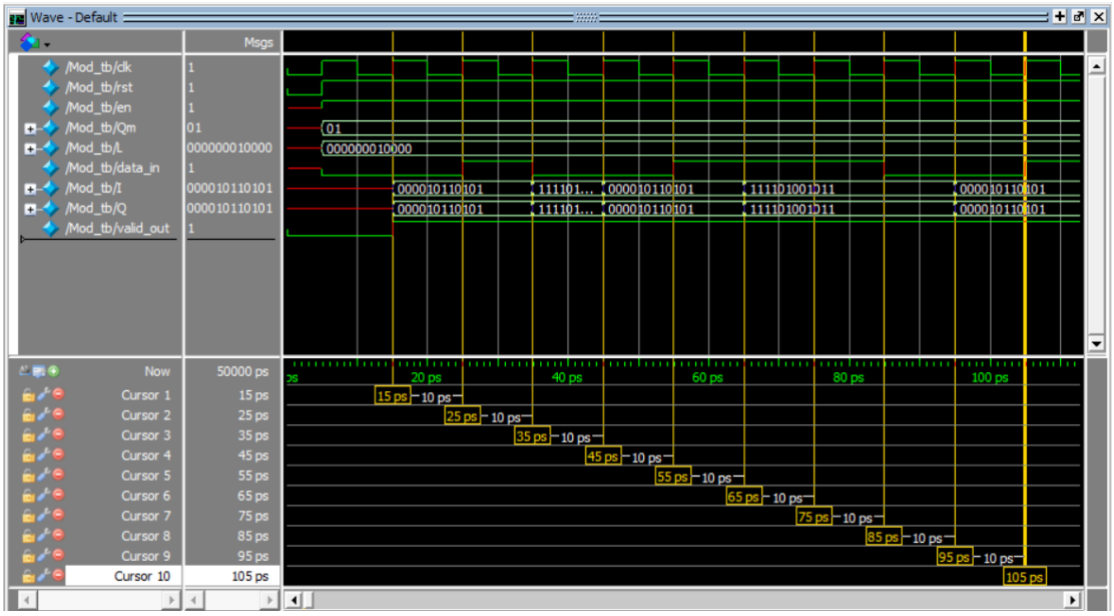

Figure 69: Modulator output for BPSK waveform

5.1.6.1.2 QPSK

1x8 complex fi

	1	2	3	4	5	6	7	8
1	0.7070 + 0.7070i	-0.7070 + 0.7070i	0.7070 - 0.7070i	-0.7070 - 0.7070i	0.7070 + 0.7070i	-0.7070 + 0.7070i	0.7070 - 0.7070i	-0.7070 - 0.7070i
2								

Figure 70: Modulator output for QPSK using MATLAB

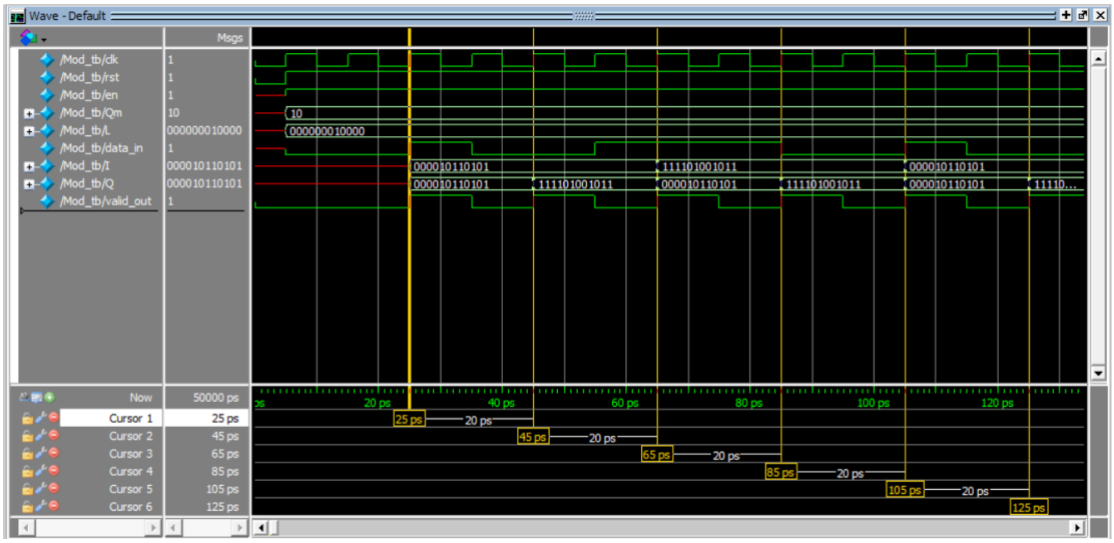

Figure 71: Modulator output for QPSK waveform

5.1.6.2 Synthesis and pnr results

5.1.6.2.1 Time

```
------------------------------------------------------------
data required time                                  1319.61
data arrival time                                    -0.65
------------------------------------------------------------
slack (MET)                                         1318.96
```

Figure 72: Modulator setup time result

5.1.6.2.2 Area

```
Combinational area:       114.646001
Buf/Inv area:              12.768000
Noncombinational area:    131.670002
Net Interconnect area:     undefined  (Wire load has zero net area)

Total cell area:          246.316003
Total area:                undefined
1
```

Figure 73: Modulator area

The area report shows that the area of the synthesized Modulator block is 246.316 μm^2 which is smaller than the reported area value of 1458 μm^2 in [11] and the reported area value of 631.484 μm^2 in [12].

5.1.6.2.3 Power

```
                 Internal         Switching          Leakage            Total
Power Group      Power            Power              Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000             0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000             0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000             0.0000             0.0000  (   0.00%)
clock_network   1.0193e-02       6.9259e-03           24.3134         4.1432e-02  (   3.21%)
register        6.0501e-02       2.0685e-03          420.9121             0.4835  (  37.46%)
sequential         0.0000           0.0000             0.0000             0.0000  (   0.00%)
combinational   6.5762e-03       8.8785e-03          750.3643             0.7658  (  59.33%)
--------------------------------------------------------------------------------------------------
Total           7.7270e-02 uW    1.7873e-02 uW    1.1956e+03 nW         1.2907 uW
1
```

Figure 74: Modulator power

The power report shows that the power of the synthesized Modulator block is 1.2907 μW which is smaller than the reported power value of 254 μW in [11] and the reported power value of 2.6761 μW in [12].

5.1.6.2.4 Final chip

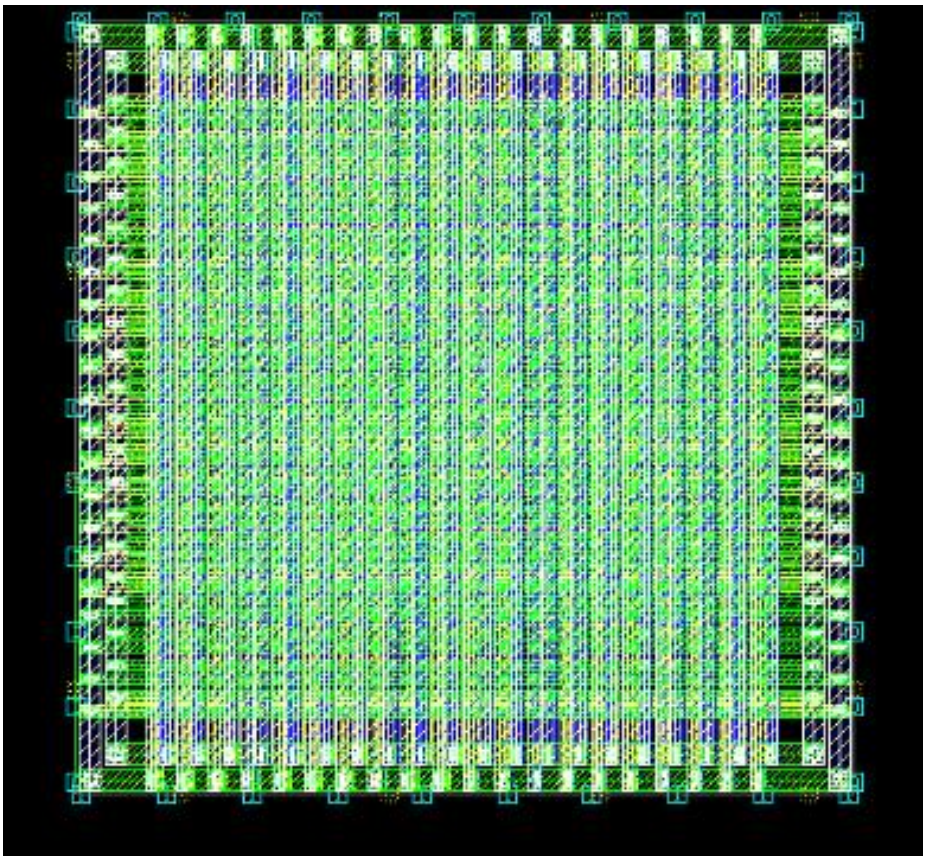

Figure 74: Modulator final chip after pnr

5.1.7 FFT

To verify the functionality of the FFT block, a testbench is used with test cases that consist of the typical outputs from the modulator including the following values.

Table 30: Binary representation of complex values used in FFT

$\frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$	0000_10110101+i 0000_10110101
$-\frac{1}{\sqrt{2}} + i\frac{1}{\sqrt{2}}$	1111_01001011+ i 0000_10110101
$\frac{1}{\sqrt{2}} - i\frac{1}{\sqrt{2}}$	0000_10110101+i 1111_01001011
$-\frac{1}{\sqrt{2}} - i\frac{1}{\sqrt{2}}$	1111_01001011+i 1111_01001011

The results show a good matching between the RTL results and MATLAB results but exhibit an increased error in the small resulting values that are close to zero. This behavior is to be enhanced in the future work by increasing the number of bits utilized for the fraction part as discussed in section 6.1.

5.1.7.1 MATLAB and Verilog Comparison

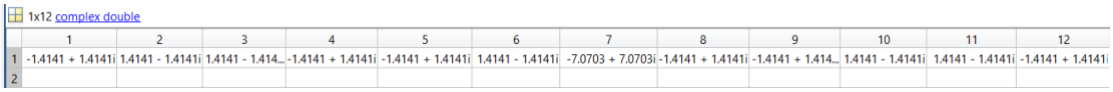

Figure 75: FFT output using MATLAB

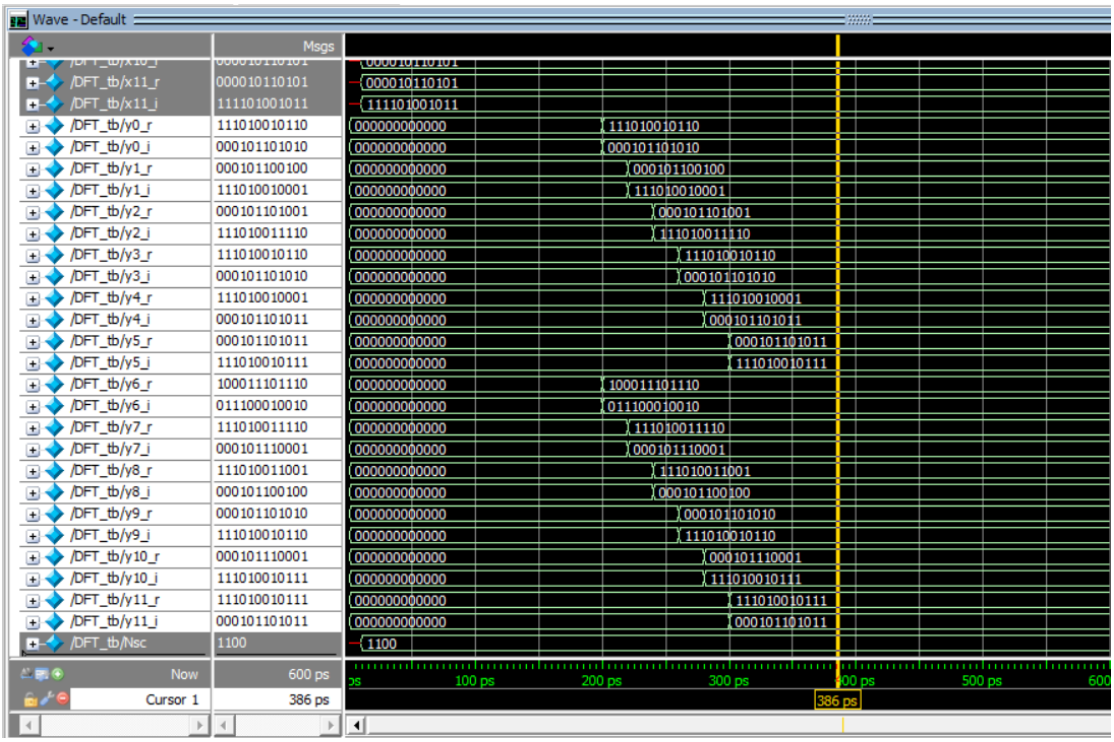

Figure 76: FFT output waveform

5.1.7.2 Synthesis and pnr results

5.1.7.2.1 Time

```
------------------------------------------------------------
data required time                                  1319.62
data arrival time                                    -2.33
------------------------------------------------------------
slack (MET)                                         1317.29
```

Figure 77: FFT setup time result

5.1.7.2.2 Area

```
Combinational area:       5829.124039
Buf/Inv area:              300.580002
Noncombinational area:    4939.619826
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          10768.743864
Total area:                 undefined
1
```

Figure 78: FFT area

The area report shows that the area of the synthesized FFT block is 10,768.74 μm^2 which is smaller than the reported area value of 57,275 μm^2 in [11] and the reported area value of 23,640 μm^2 in [12]

5.1.7.2.3 Power

```
                          ---------
Total Dynamic Power    =   3.2934 uW  (100%)

Cell Leakage Power     =  45.9526 uW

Leakage power with reduced spread = 0


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network      0.1971           0.5504            371.2800             1.1188  (   2.27%)
register           2.4236        1.1183e-02         1.7659e+04            20.0939  (  40.80%)
sequential         0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational  4.7465e-02        6.3715e-02         2.7922e+04            28.0333  (  56.93%)
--------------------------------------------------------------------------------------------------
Total              2.6682 uW        0.6253 uW      4.5953e+04 nW         49.2461 uW
1
```

Figure 79: FFT power

The power report shows that the power of the synthesized FFT block is 49.25 μW which is smaller than the reported power value of 1.639 mW in [11] and the reported power value of 87.847 μW in [12]

5.1.8 Resource Element Mapper

5.1.8.1 MATLAB and Verilog Comparison

The following figures show that the output of the Resource Element Mapper matches the output of MATLAB successfully. According to the tet case used where the values of I_{sc} is 15 hence the position of the allocated subcarriers will be the $10^{th}, 11^{th}, nd\ 12^{th}$ row. It is also noted that the third column is don't contain data as

the value of the DMRS is set to 3. However, more test cases must be tested to make sure that the block functions correctly for all testing possibilities.

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0
10	1.1001e+11	1.0001e+11	0	1.0000e+11	0	0	0
11	1.0001e+11	1.0001e+11	0	1.0000e+11	0	0	0
12	1.0001e+11	1.0001e+11	0	0	0	0	0

Figure 80: REM output using MATLAB

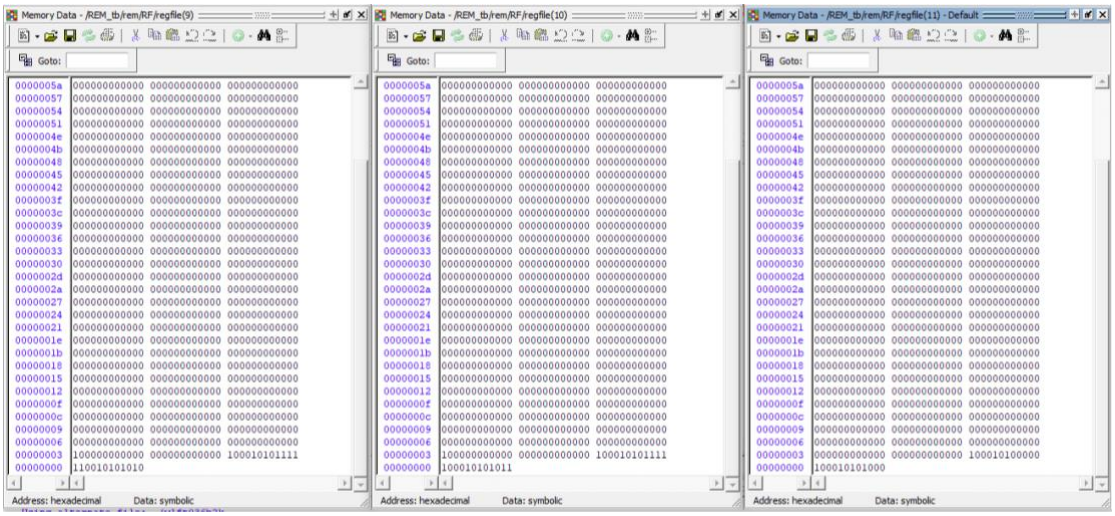

Figure 81: REM memory output

5.1.8.2 Synthesis and pnr results

Initial estimation of synthesis results

5.1.8.2.1 Time

```
-------------------------------------------------------------
data required time                                  1319.62
data arrival time                                    -1.14
-------------------------------------------------------------
slack (MET)                                         1318.48
```

Figure 82: REM setup time result

5.1.8.2.2 Area

```
Combinational area:        805.980004
Buf/Inv area:               40.166000
Noncombinational area:     940.575972
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          1746.555976
Total area:                 undefined
1
```

Figure 83: REM area

5.1.8.2.3 Power

```
                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000            0.0000  (   0.00%)
clock_network  3.6136e-02           0.1671             72.0557            0.2753  (   3.22%)
register           0.4607       1.7640e-02        3.1513e+03            3.6296  (  42.51%)
sequential     2.8294e-03       1.1573e-03          148.1066            0.1441  (   1.69%)
combinational  3.0053e-02       6.3789e-02        4.3962e+03            4.4901  (  52.58%)
--------------------------------------------------------------------------------------------------
Total              0.5297 uW        0.2497 uW     7.7597e+03 nW         8.5391 uW
1
```

Figure 84: REM power

5.1.8.2.4 Final chip

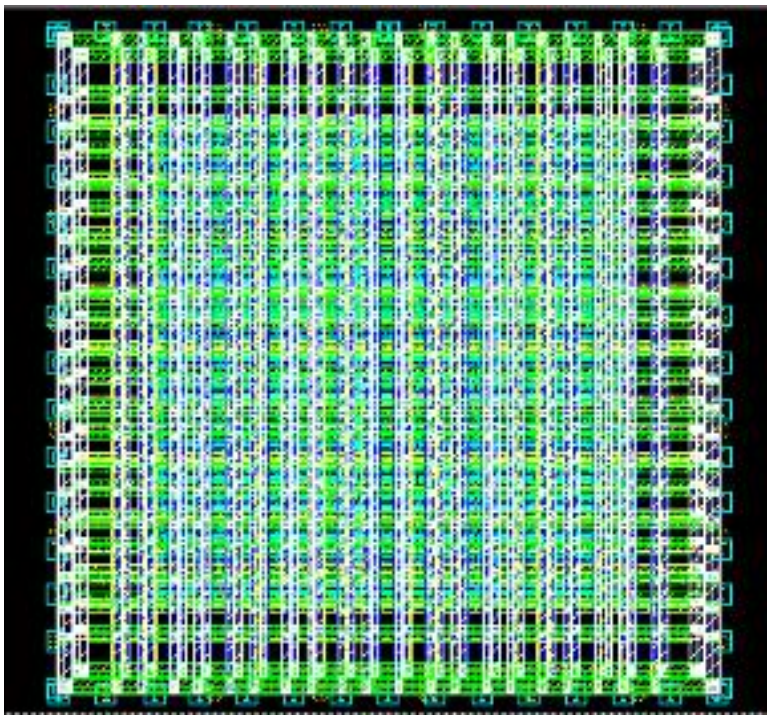

Figure 85: REM final chip after pnr

5.1.9 IFFT

To verify the functionality of the IFFT block, a testbench is used with test cases that use the outputs from the FFT block designed before. The results show a moderate matching between the RTL and the MATLAB reference model with some error. This error can be referred to the accumulation of a division by 2 in one step at the last stage instead of performing it gradually along the stages, which results in a loss of information after truncating the shifted bits. This is to be modified in the future work by distributing the division, by shifting, operation throughout the stages.

5.1.9.1 MATLAB and Verilog Comparison

The following figures present the first 12 outputs out of the 128 output of the 128-point IFFT.

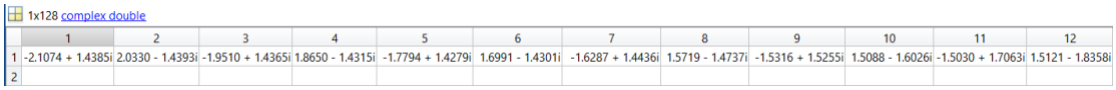

Figure 85: IFFT output using MATLAB

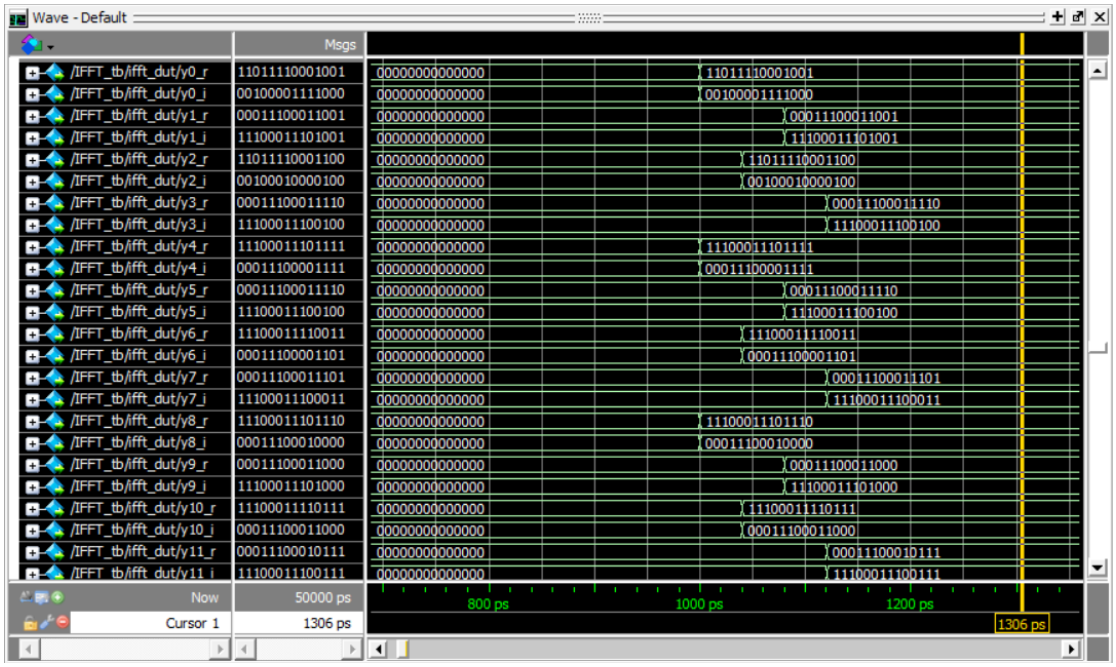

Figure 86: IFFT output waveform

5.1.9.2 Synthesis and pnr results

5.1.9.2.1 Time

```
--------------------------------------------------------------------------
data required time                                              1319.61
data arrival time                                                 -3.34
--------------------------------------------------------------------------
slack (MET)                                                     1316.27
```

Figure 87: IFFT setup time result

5.1.9.2.2 Area

```
Number of ports:                         7171
Number of nets:                         52040
Number of cells:                        45426
Number of combinational cells:          36496
Number of sequential cells:              8914
Number of macros:                           0
Number of buf/inv:                       4673
Number of references:                      71

Combinational area:       44179.940651
Buf/Inv area:             2673.566016
Noncombinational area:    40373.478538
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          84553.419190
Total area:                 undefined
1
```

Figure 87: IFFT area

The area report shows that the area of the synthesized 128-Point IFFT block is 84,553.42 μm^2 which is smaller than the reported area value of 374,142.3 μm^2 in [12]

5.1.9.2.3 Power

```
                           ---------
Total Dynamic Power    =  28.2799 uW  (100%)

Cell Leakage Power     = 413.6490 uW

Leakage power with reduced spread = 0


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network  5.7824e-02           3.6021          194.5135            3.8545  (   0.87%)
register          15.9714           1.9012        1.4487e+05          162.7386  (  36.82%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational      2.8220           3.9252        2.6859e+05          275.3369  (  62.30%)
--------------------------------------------------------------------------------------------------
Total             18.8513 uW        9.4286 uW     4.1365e+05 nW       441.9301 uW
1
```

Figure 89: IFFT power

The power report shows that the power of the synthesized 128-Point IFFT block is 441.93 μW which is smaller than the reported power value of 1.41 mW in [12].

5.1.9.3 Comments

Synthesis power and area results are compared with the previous work in [12] and not with [11] as the IFFT implemented in [11] is 16-Point IFFT, so the valuea are not compatible.

5.2 Final synthesis and pnr results

5.2.1 Synthesis summary

Table 31: Synthesis summary for all blocks

Block	Area (μm^2)	Power (μW)	Setup Slack (ns)
CRC	379.847993	1.9967	1318.65
Turbo Encoder	13780.395684	57.9320	1316.76ns
Rate Matching	72349.073059	367.1643	1316.14
Channel Interleaver	20014.637515	77.2071	1317.38
Scrambler	135.659998	0.7746	1318.96
Modulator	246.316003	1.2907	1318.96
FFT	10768.74386	49.2461	1317.29
REM	1746.555976	8.5391	+1318.48
IFFT	84553.41919	441.9301	1316.27

5.2.2 PnR summary

Table 32: pnr summary for some blocks

Block	Area (μm^2)	Power (μW)	Setup (ns)
Modulator	2516.625974	256.2272	1318.86
REM	10755.975863	1.4463e+03	1318.86

5.3 Project tasks and Gantt chart

Table 32: Gantt chart and tasks distribution

Functional Specifications from the Standard and Literature Review				
General literature review of the NB-IOT protocol	All team members	100%	1/10/2022	20/10/2022
Channel Inter-leaver, Scrambler, Modulator	Yara Nofal	100%	1/11/2022	20/11/2022
CRC, Resource Element Mapper	Arwa Ahmed	100%	1/11/2022	20/11/2022
FFT, IFFT	Lobna Elahraf	100%	1/11/2022	20/11/2022
Turbo_Encoder, Rate Matching	Yasmine Abdelaal	100%	1/11/2022	20/11/2022
High Level Modeling using Matlab				
Channel Inter-leaver, Scrambler, Modulator	Yara Nofal	100%	25/11/2022	30/12/2022
CRC, Resource Element Mapper	Arwa Ahmed	100%	25/11/2022	30/12/2022
FFT, IFFT	Lobna Elahraf	100%	25/11/2022	30/12/2022
Turbo_Encoder, Rate Matching	Yasmine Abdelaal	100%	25/11/2022	30/12/2022
Final Projects, and Final Exams, Winter Break			08/01/2023	26/02/2023
RTL Design and Behavioral Simulation using ModelSim				
CRC, Channel Inter-leaver, Modulator	Arwa Ahmed, and Yara Nofal	100%	28 /02/2023	20/05/2023
FFT, IFFT	Arwa Ahmed, and Lobna Elahraf	100%	28 /02/2023	20/05/2023
Turbo_Encoder	Yasmine Abdelaal, and Lobna Elahraf	100%	28 /02/2023	20/05/2023
Rate Matching	Yasmine Abdelaal	100%	28 /02/2023	20/05/2023
Scrambler	Yara Nofal	100%	28 /02/2023	20/05/2023
Resource Element Mapper	Yara Nofal	85%	28 /02/2023	20/05/2023
RTL Blocks Verification and Reference Model Comparison				
CRC, Channel Inter-leaver, Scrambler	Arwa Ahmed	100%	20/5/2023	12/6/2023
ASIC Flow (synthesis)				
CRC, Channel Interleaver, Modulator, Turbo_Encoder, Scrambler	Yasmine Abdelaal	100%	20 /05/2023	01/06/2023
Rate Matching	Yasmine Abdelaal	90%	20 /05/2023	01/06/2023
FFT, IFFT	Lobna Elahraf	100%	20 /05/2023	01/06/2023
ASIC Flow (PNR)				
Modulator, Turbo_Encoder, Resource Element Mapper	Yasmine Abdelaal	100%	20 /05/2023	01/06/2023

6 Conclusion and future work

6.1 Conclusion

The NB-IoT (Narrowband Internet of Things) is an LPWAN (low-power, wide-area network) technology created for Internet of Things (IoT) applications. A crucial part of an NB-IoT system is the NB-IoT transmitter which is in charge of sending data from IoT devices to the network. In this project, the Transmitter is tackled from different perspectives where a detailed MATLAB code that simulates the architecture was written for every module. After checking that the written MATLAB code verifies the NB-LTE specifications provided in the referenced standard, an RTL code is written, and the implementation of each module was tested using randomly generated test vectors. The results of the RTL were compared to those of MATLAB and they were matching in all the implemented blocks (considering the pre-calculated errors of the blocks that perform mathematical operations that require fixed-point representation). The next stage, according to the ASIC flow, is to take these synthesizable RTL codes along with the library files and input them into the synthesis tool. The generated netlist of the synthesizer was provided to the PnR tool (performed for some of the synthesized blocks). In this project, Synopsis package with a technology size of 45nm was used for the Synthesis and PnR of the transmitter blocks. The synthesis results were compared to the previous work results and it is found that our design managed to obtain better results, especially in area and power for most of the blocks through the optimizations performed earlier in the RTL implementations. The lower power consumption means that the chip can work for longer periods of time which consequently increases the battery life. These improvements can make NB-IOT chips prone to poor network connections making them more reliable. Moreover, a smaller chip area in addition to low power consumption means lower chip cost, and if a chip is affordable, it will be easily accessible through a wide range of IOT applications. These improvements help brighten the future of NB-IOT applications which in turn facilitates the everyday life of people and takes the world a further step to the future.

6.2 Future work

- **Channel Interleaver**

  This module requires further optimizations in the RTL in order to improve its speed and consequently its power consumption. This can be achieved by finding a prediction method for the to-be-used indices instead of the actual placement of the elements and then retrieving them back.

- **Scrambler**

  It takes 1600 cycles to generate the unique Golden Sequence which introduces high latency in the design. It is recommended to work on reducing this number of cycles by finding a prediction method that minimizes the elapsed time in this module.

- **Input Buffer preceding the DFT**

  The Modulator outputs vary according to the modulation type whether it is BBSK or QBSK and the DFT takes its 12 inputs simultaneously. That's why an input buffer should be inserted between the DFT and the Modulator in order to synchronize their operation and make the propagation of bits smooth throughout the whole block. Moreover, this is essential for the integration of the constituent blocks of the NB-IOT transmitter chip.

- **DFT**

  The Accuracy of the outputs of this module can be highly improved by increasing the number of the fraction bits that are accounted for in the widths of the input signals. This will consequently increase the SNR and the accompanied error.

- **IFFT**

  In this module, there was a (division by 2) operation in each stage but in our design, we grouped all these divisions to be performed at the end of the last stage (shift >>>7) which unfortunately made some losses in the output signals due to the truncation resulted due to the limitation on the signal width (fixed-point restriction). This shall be resolved if the shifting is distributed among the stages by adding (shift <<<1) to the outputs of each radix-2.

- **Cyclic Prefix**

  This module acts as a guard band that is made between the LTE symbols and it is essential to reduce the intra-symbol interference and keep the OFDM signals from any interferences. Thus, we highly recommend implementing an optimized design for it to be added to our design just after the IFFT.

References

[1] Fattah, H. (2018). 5G LTE Narrowband Internet of Things (NB-IOT) (1st ed.).

CRC Press. https://doi.org/10.1201/9780429455056

[2] Mostafa, H. (2022). Lecture.1 notes, NANENG 501: Advanced ASIC Digital Design.

[3] TSGR. (2020). *TS 136 212 - V16.2.0 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 16.2.0 Release 16)*. https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[4] "Ixia network |security |application performance." [Online]. Available: https://support.ixiacom.com/sites/default/files/resources/whitepaper/sc-fdma-indd.pdf.

[5] TSGR. (2020). *TS 136 211 - V16.2.0 - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.211 version 16.2.0 Release 16)*. https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

[6] J. Lofgren and P. Nilsson, "On hardware implementation of radix 3 and radix 5 FFT kernels for LTE Systems," 2011 NORCHIP, 2011.

[7]J. G. Proakis and D. G. Manolakis, "CH8:Efficient Computation of the DFT:FFT algoriths," in Digital Signal Processing: Principles, algorithms, and applications, Upper Saddle River, NJ: Prentice Hall, 1996.

[8] "Development guide for industrial using NB-IOT - gsma.com." [Online]. Available: https://www.gsma.com/iot/wp-content/uploads/2019/08/201902_GSMA_IoT-Development_Guide_NB-IoT_for_Industrial.pdf.

[9] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band internet of things," *IEEE Access*, vol. 5, pp. 20557–20577, 2017. https://doi.org/10.1109/ACCESS.2017.2751586

[10] O. Kodheli, N. Maturo, S. Chatzinotas, S. Andrenacci and F. Zimmer, "NB-IoT via LEO Satellites: An Efficient Resource Allocation Strategy for Uplink Data Transmission," in IEEE Internet of Things Journal, vol. 9, no. 7, pp. 5094-5107, 1 April1, 2022, https://doi: 10.1109/JIOT.2021.3109456.

[11] B. H. Mohamed *et al.*, "Design of the baseband physical layer of narrowband IOT LTE uplink digital transmitter," *Journal of Circuits, Systems and Computers*, vol. 29, no. 07, p. 2050111, 2019. doi:10.1142/s021812662050111x

[12] A. Hashem, A. Hossam, M. Hefnawy, M. Roshdy, T. Nabil, "digital design of NB-IoT Rel 16 Physical Layer Uplink Transmitter," B.S. Thesis, Nanotechnology, ZC-UST, 2022.