DESIGN AND IMPLEMENTATION OF 5G-NR PHYSICAL DOWNLINK SHARED CHANNEL RECEIVER

A THESIS

SUBMITTED TO

THE DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS

FACULTY OF ENGINEERING, CAIRO UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN ELECTRONICS AND ELECTRICAL COMMUNICATIONS ENGINEERING

AYA ALLAH MOHAMED MOHYE

KHALED REDA ATRESS

KHALED ABDEL FATAAH

SALMA ALAA AHMED MOUNIER

MOHAMMED ADEL GHAREEB

MOHAMMED WALEED HAFEEZ

UNDER SUPERVISION OF

DR. HASSAN MOSTAFA

ENG. ABDELRAHMAN HAMED

TECHNICALLY SPONSORED BY

ONE LAB EGYPT INC.

ICpedia INC.

JULY 2021

Table of Contents

List of Figures

List of Tables

List of Symbols and Abbreviations:

3gpp: third generation partnership project.

NR: new radio.

Mm-wave: millimeter wave.

LDPC: low density parity check.

PDSCH: physical downlink shared channel.

Physical uplink shared channel: PUSCH.

BPSK: binary phase shift keying

QPSK: quadrature phase shift keying

QAM: quadrature amplitude modulation.

Q_m: the modulation order.

SA: system aspects.

RAN: radio access network.

CT: core network terminals.

FPGA: field programming gate array.

1G: first generation.

2G: second generation.

3G: third generation.

4G: fourth generation.

5G: fifth generation.

LTE:	long term evaluation.
EMBB:	enhanced mobile broadband.
MMTC:	massive machine communication.
URLLC:	ultra-reliable low latency communication.
UE:	user equipment.
BWPs:	bandwidth parts.
CP:	cyclic prefix.
OFDM:	orthogonal frequency division multiplexing.
MIMO:	multi-input multi-output.
FR:	frequency range.
FDD:	frequency division duplex.
TDD:	time division duplex.
UL:	uplink.
DL:	downlink.
SUL:	supplementary uplink.
SDL:	supplementary downlink.
AWGN:	Additive Gaussian noise.
LLR:	log-likelihood ratio.
RE:	resource element.
RS:	reference signals.

DM-RS: demodulation reference signals.

PT-RS: phase tracking reference signals.

CPM: common phase error.

ICI: inter-carrier interference.

DCI: downlink control information.

RBs: resource blocks.

NID: identity number.

RNTI: radio network temporary identifier.

RV: redundancy version.

RAM: random access memory.

Ff: flip flop.

LUT: lookup table.

LFSR: linear feedback shift register.

CRC: cyclic redundancy check.

DFT: discrete Fourier transform.

FFT: fast Fourier transform.

CP: cyclic prefix.

NCP: normal cyclic prefix.

ECP: extended cyclic prefix.

MCM: multichannel modulation.

ICI: inter-channel-interference.

ISI: inter-symbol-interference.

HARQ: hybrid automatic repeat request.

SNR: signal to noise ratio.

SISO: Soft input soft output.

BG: Base graph.

SPC: Single parity check.

Ext: extrinsic

Int: intrinsic.

Z_c : expansion factor.

Abstract:

The fifth generation of wireless network technology (5G), which is known as new radio (NR) is a new cellular generation has many superior advantages and features rather than the previous generation which attract all the users and customers to this new technology. It is not only a mobile telecommunication-based technology but is used in many fields due the most advanced features served by 5G technology.

By 2021 there will be up to 438 million 5G connections ,The attentions of communication world nowadays is focusing towards the 5g due to the benefits that it provides better than any previous communication technology.

Why the 5G should be used is more important than how, 5G speeds are up to 10 times faster than 4G.,5G latency is reduced to 1ms, instead of 10ms in 4G. For the 4G are often at war during the peak hours; 5G allows for a more reliable number of users can connect at the same time. The 5G connection will be stable enough for the machine to feed the animals in the 5G network.
In LTE, there's only one numerology subcarrier spacing which is 15 KHz, but in NR there are different kinds of subcarrier spacing are supported: from 15 KHz to 240 KHz.

5G technology world fulfils miscellaneous requirements in many applications, by improving latency, spectrum efficiency and data rates. The hardware implementation must be able to accommodate the broad requirements of 5G. This is making the FPGA technology which can run a real time operating systems [RTOS] and powerful serial links a proper implementation for the 5G transceivers systems. This technique is optimal since it uses directly HDL description of numerical methods of system models. A number of transmission tests for different scenarios can be done. The real-time results obtained validate the proposed hardware architecture. Furthermore, it demonstrates the efficiency of the proposed solution consisting on the association of wireless protocols.

This project focuses on implementing a 5G receiver physical downlink shared channel [PDSCH] on Xilinx-virtex 7 FPGA toolkit with frequency equal to 200 MHz, the approach for designing the PDSCH uses low density-parity check [LDPC] which is a highly error correcting code. PDSCH also contains rate dematching, deinterleaver and demodulation up to 256-QAM is carried out. The approach is focused to increase data rates and lower error rates. Further the testing of the modules are done in order to justify the designed approach and its functioning. This is accomplished using MATLAB and modelsim, MATLAB was used to compare between proposed design and built-in matlab functions which showed close results and very low bit error rate [BER]. To simulate all the modules of PDSCH channel, Modelsim is used. For synthesis and implementation Xilinx 14.7 tool is used.

Acknowledgments

This dissertation would not have been possible without the support of many people. First and foremost, we would like to thank our advisors and role models Dr. Hassan Mostafa, Eng. Abdelrahman Hamed, Eng. Mohamed Adawi.

Dr. Hassan Mostafa, we would like to thank you both for your patience, guidance, support the great supervision you have offered us through the whole project, and equipment you helped us access and use easily, you provided perfect technical guidance, through the whole project, that helped us reach our desired objective, address various complex problems we faced, perform several advanced techniques. We would specially thank you for the tremendous amount of support and guidance you provided us.

Moreover, we want to thank Eng. Abdelrahman Hamed who has helped us with his relevant experience on several problems we have faced through our project.

I. Introduction:

5G networks are part of a major digital transformation trend that is impacting consumer and commercial spaces. Companies are racing to have the fastest or largest 5G networks and countries are competing to be the first to deploy fully functional, nationwide 5G. That is because the benefits of the new technology are expected to fuel trans-motive new technologies, not just for consumers but also for businesses, infrastructure and defense applications new devices and applications are emerging on the market that take advantage of the dramatically reduced latency and much higher throughput that 5G offers. It is an exciting time. Like with previous generations, consumers are leading the 5G adoption, followed by commercial, industrial, government and medical markets.

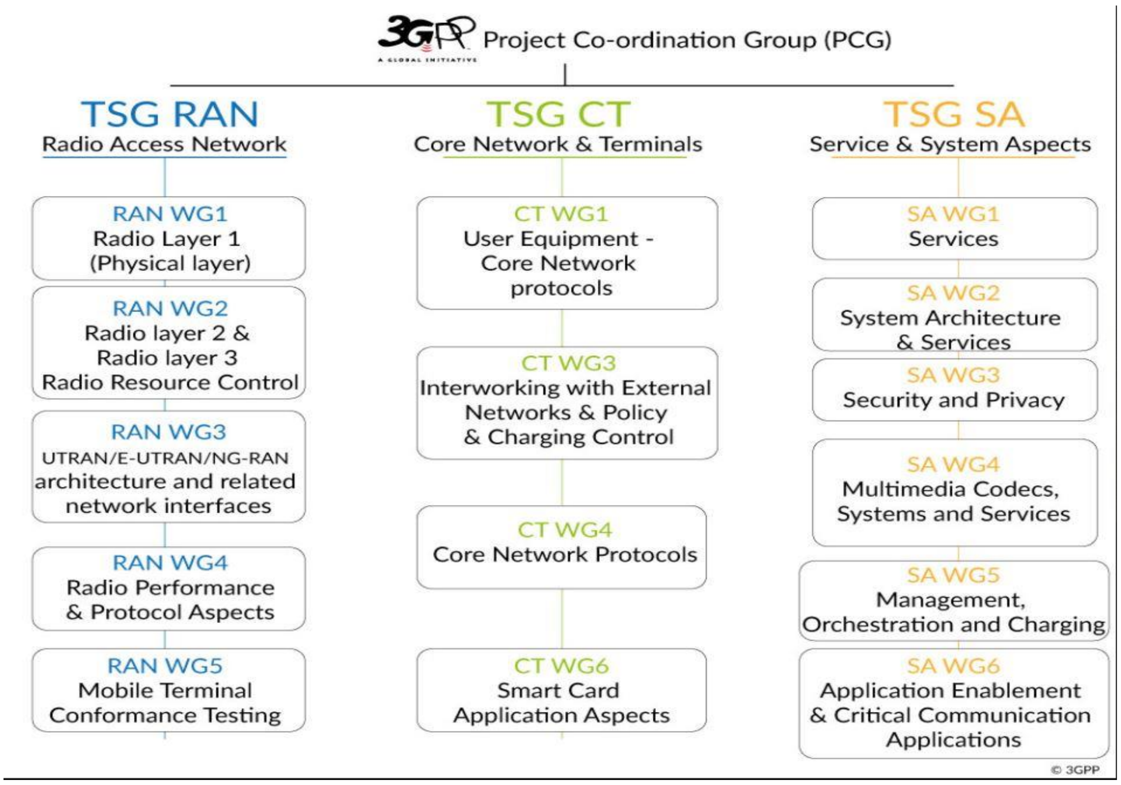
II. 3GPP Organization:

The third-generation partnership project or (3GPP) as known is a telecommunication standards development organization. It is well-known as a partner organization that provides its members with a suitable environment to produce telecommunications standardization and reports on new technology.

The 3GPP project consists of a mobile radio technology, the core network, and maintenance opportunities, and give a complete description of the cellphone.

The main three technical specification groups in 3GPP are:

1- Core network and terminals. (CT)
2- Radio access network. (RAN)
3- Service and systems aspects. (SA)



3GPP A GLOBAL INITIATIVE
Project Co-ordination Group (PCG)
TSG RAN
Radio Access Network
TSG CT
Core Network & Terminals
TSG SA
Service & System Aspects
RAN WG1 Radio Layer 1 (Physical layer)
RAN WG2 Radio layer 2 & Radio layer 3 Radio Resource Control
RAN WG3 UTRAN/E-UTRAN/NG-RAN architecture and related network interfaces
RAN WG4 Radio Performance & Protocol Aspects
RAN WG5 Mobile Terminal Conformance Testing
CT WG1 User Equipment - Core Network protocols
CT WG3 Interworking with External Networks & Policy & Charging Control
CT WG4 Core Network Protocols
CT WG6 Smart Card Application Aspects
SA WG1 Services
SA WG2 System Architecture & Services
SA WG3 Security and Privacy
SA WG4 Multimedia Codecs, Systems and Services
SA WG5 Management, Orchestration and Charging
SA WG6 Application Enablement & Critical Communication Applications
© 3GPP

Figure 1: 3GPP co-ordination projects

III. Generation of wireless communications:

First Generation:

1G is the first cellular telecommunication the concept of cellular and reuse frequencies was developed by AT&T bell labs in the late of 1970’s. 1G was introduced by NIPPON TELEGRAPH AND TELEPHONE in Tokyo in 1979. The First generation is only supporting a voice communication with analog based modulation. 1G suffer from limited coverage, poor quality of sound, low noise immunity and inefficient of utilization of bandwidth.

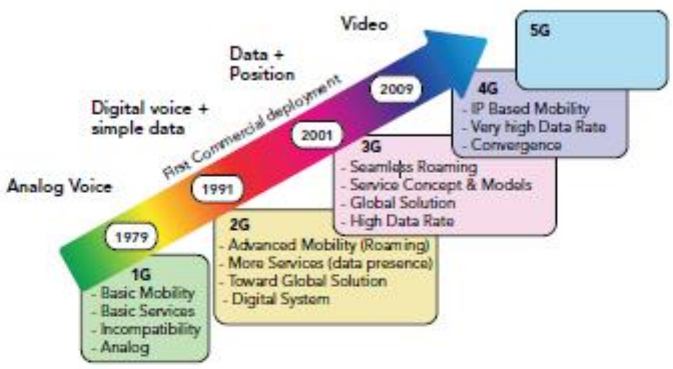


Figure 2:Evolution of standards in mobile technology

Second Generation:

2G is launched under GSM standers in Finland 1991. The most specialized feature in 2G over 1G is type of modulation. 2G support digital modulation technique which make signal is more immune to noise and more efficient utilization of spectrum. 2G is not only used for voice communication but also used in data transmissions but with very limited services. Reuse frequency in 2G is 1 which facilitated planning and better capacity than 1G.

Third Generation:

3G was launched in 2001. The aim of 3G is standardization network protocols which enable users to access data from any location and declared roaming service concept. 3G has higher data transfer than 2G. 3G use CDMA which decrease power needed to transmit data; in addition, it makes user's data undetectable by all users except the intended one that will use certain code to de-encrypt it. That's code synchronization between transmitter and receiver is needed.

Fourth Generation:

4G for the first time implemented in 2009, the Long-Term Evolution (LTE) and 4G as standard. After that, it was introduced worldwide, and is made from high-quality streaming video is a reality for millions of consumers. 4G provides high-speed mobile access to the network up to 1 Gigabit per second (for regular users), which will make it easier for services, such as games, HD video and HQ video.

The catch was that when switching from 2G to 3G, it is as simple as switching between the two SIM cards, mobile devices were specifically designed for the 4G service. This has helped to device makers, and a substantial increase in their income due to the launch of a new 4G phone and it is one of the factors behind the company's rise to be the world's first company to billions of dollars.

Fifth Generation:

5G, it is often referred to as the ' NR (New Radio-is a sign that, in contrast to LTE, it is not the reverse change in behavior), which is a set of technologies from the physical layer to the Network Core, which mainly three distinct types of services:

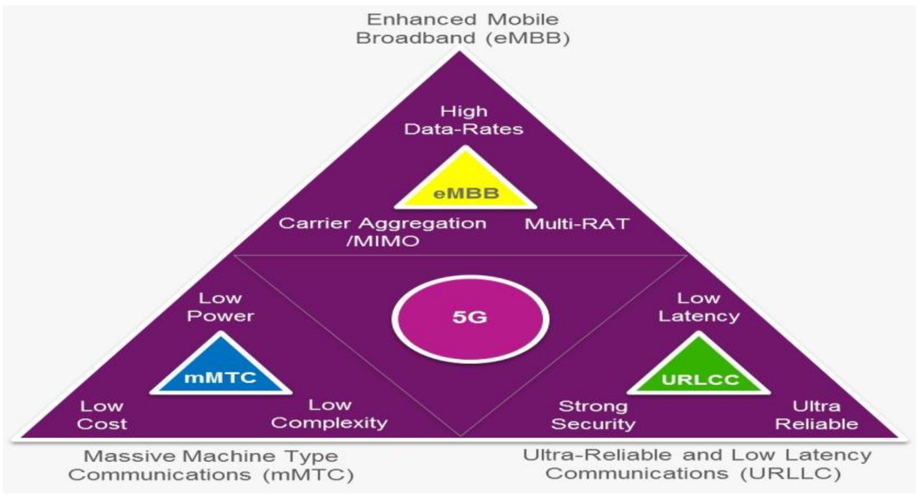

Enhanced Mobile Broadband (eMBB)
High Data-Rates
eMBB
Carrier Aggregation /MIMO
Multi-RAT
Low Power
5G
Low Latency
mMTC
URLLC
Low Cost
Low Complexity
Strong Security
Ultra Reliable
Massive Machine Type Communications (mMTC)
Ultra-Reliable and Low Latency Communications (URLLC)

Figure 3: Main services supported by 5G

1- EMBB (Enhanced Mobile Broadband)

This category considers the natural extension of the traditional mobile communication scenarios. This can be accomplished by the provision of higher data-transfer rates over a wide range, and to enhance the bandwidth of the support, the maximum data transfer rate of more than 10 Gbit / s for the end-users who will move in with a crowd.

2- MMTC (Massive Machine Communication)

The main purpose of the MMTC is to connect with many different devices, it is not necessary to flight delays. MMTC is a type of machine-to-machine communications, which provides internet access to the monitoring, detection, and measurement of the device.

It is expected that the number of MMTC devices is greater than the number of devices that are of human movement, allowing an increase in the Iop of the causes of the proliferation of wireless devices, which the MTS movement.

3- URLLC (Ultra-Reliable Low Latency Communication)

The key to the reliability requirements, and latency characteristics of the use case in the category of low latency of 1 ms and a bottom, and a few more milliseconds and it is a very high degree of reliability, the top 99.9999% for use in mission-critical services such as the smart grid, smart transportation, and remote medical surgery.

5G runs on radio frequencies ranging from below 1 GHz all the way up to very high frequencies, called “millimeter wave”. The combination of these frequencies provides nationwide coverage, massive capacity, and multi-Gigabit peak rates, along with ultra-low latency. The lower the frequency, the farther the signal can travel. The higher the frequency, the more data it can carry. Here are the three frequency bands that are the foundation for 5G networks (high-band, mid-band, low-band)

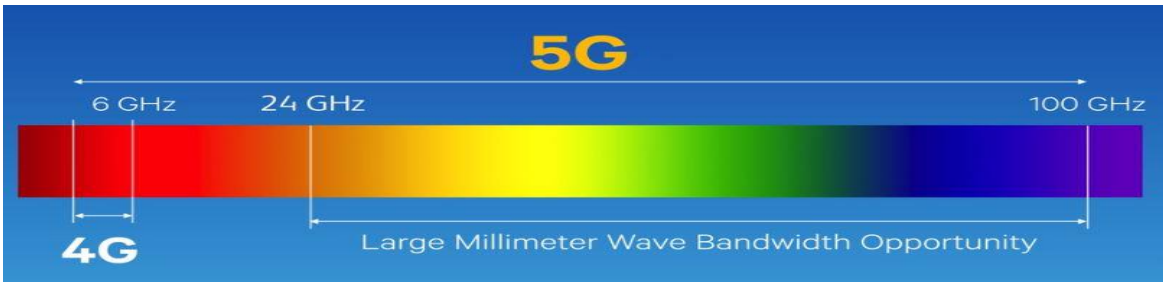


Figure 4: mm-wave frequency band

IV. Features of 5G:

- Ultra-thin design to enhance the performance of the network and aids in the reduction of the interference.
- Low latency, improving performance.

One of the goals of wireless generation to reduce latency. Latency is a measure of how long a signal from a source to a receiver, and then back, because this is an important way to make use of URLLC, so it is even faster than the speed of the human visual processing, which allows you to control devices from a distance, almost in real-time. It features an ultra-low latency of 1 ms, which means that the immediate response of the devices on the network.

- High data rate:

  One of the main benefits of 5G networks is a big speed of the Internet. No, we are not talking about a 2-fold increase in the speed, but about a 100-fold increase in data transfer rate is up in the air, as it is expected to reach speeds of up to 10 GB / sec.

- High capacity:

  The new 5G network will provide support to the more than 100-times higher machine throughput. This means that an increasing number of people or devices can connect to the network without any compromises to the performance of the other users are connected.

- Long battery lifetime:

  One of the most severe effects of a high-speed network, that is, at the end of the loading on the device consumes too much power. Therefore, lead to a decline in the life expectancy of the battery, but 5G is a blend of a variety of technologies, at the same time. The system, however, is smart and knows when to use a particular technology, for maximum space efficiency. In this way, the network can provide you with the best possible connection with devices that do not have to look for, as it saves me a lot of time on different devices, such as mobile phone.

- Reliability:

  Reliability refers to the ability to provide a particular service to a very high level of availability.

- Connecting massive number of services:

  5G is a key factor in the creation of a single world. 5G and Massive IoT, is a term used to describe the large number of sensors and devices that will be connected to the world, that can communicate with each other, they will be incorporated into the infrastructure of the 5G network.

5G is expected that up to 1 million connected devices in a 38-square-mile, so that the network will be able to interact with a wide variety of devices used for IoT applications.

- High coverage:

  One of the most important advantages of 5G technology, it is better to cover for, 5G, users will, where it signals to reach areas that were previously difficult to reach. In this way, the user will get the highest quality, that is they want to see on their devices.
- Flexible spectrum with a high frequency.

Features of 5G over LTE:

- 5G speeds are up to 10 times faster than 4G.
- 5G latency will be reduced to 1ms, instead of 10ms in 4G.
- Get the Cell to the edge of the 4G concept to be removed in 5G, and the user experience flat tires.
- For the 4G are often at war during the peak hours; 5G allows for a more reliable number of users can connect at the same time.
- The 5G connection will be stable enough for the machine to feed the animals in the 5G network
- In LTE, there's only one numerology subcarrier spacing which is 15 KHz, but in NR there are different kinds of subcarrier spacing are supported: from 15 KHz to 240 KHz.

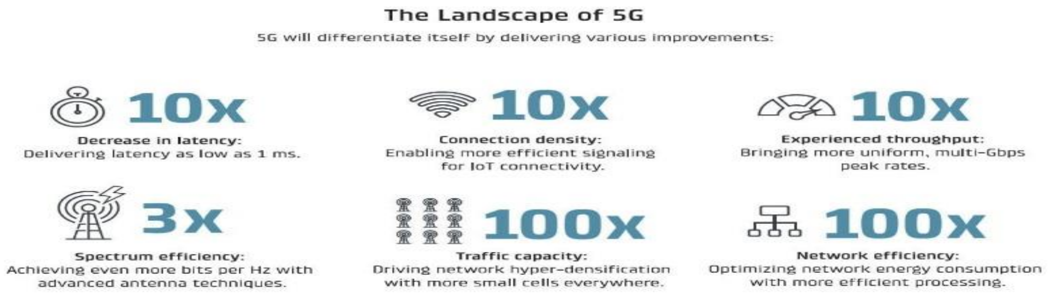

The Landscape of 5G
5G will differentiate itself by delivering various improvements:
10x
Decrease in latency:
Delivering latency as low as 1 ms.
10x
Connection density:
Enabling more efficient signaling for IoT connectivity.
10x
Experienced throughput:
Bringing more uniform, multi-Gbps peak rates.
3x
Spectrum efficiency:
Achieving even more bits per Hz with advanced antenna techniques.
100x
Traffic capacity:
Driving network hyper-densification with more small cells everywhere.
100x
Network efficiency:
Optimizing network energy consumption with more efficient processing.

Figure 5: 5G feature over LTE

V. NR overview:

Bandwidth Part Concept:

A new concept in 5G New Radio (NR) is a function which is known as the bandwidth parts (BWPs). BWPS more flexibility in how the funds is to be allocated to this device. The parts of the tire that is related to the provision of flexibility, so that the different types of channels that can be transmitted in each bandwidth. Most of the base stations may be able to take advantage of the wide bandwidth available in 5G. However, as of the features of the native hardware (UE), will be different, and some of the UE will make it harder to use this large bandwidth. The parts of the tape allow you to multiplex several signals, and the signal-to for the more efficient use and adjustment of the EC-reach and power.

The LTE service providers in a narrow frequency range of 20 MHz, which can be joined together to create a channel capacity-up to 100 MHz), LTE-Advance (up to 640 Mhz, LTE-Advanced Pro. By way of comparison, the maximum bandwidth of the 5G reference for network-up to **100 Mhz** in the **frequency range 1** (FR1: 450 Mhz-6 Ghz) or **400 Mhz** and in the **frequency range 2** (FR2: 24.25 Ghz-52.6 Ghz), which can be combined with a maximum bandwidth of 800 MHz. With the help of several components, and the carrier can be broken up and used for a variety of purposes.

Each 5G NR BWP has its own numerology, meaning that each BWP can be configured differently with its own signal characteristic, enabling more efficient use of the spectrum and more efficient use of power. This feature is good for integrating signals with different requirements. One BWP may have reduced energy requirements, while another may support different functions or services, and yet another may provide coexistence with other systems. Bandwidth parts will support legacy 4G devices with new 5G devices on the same carrier.

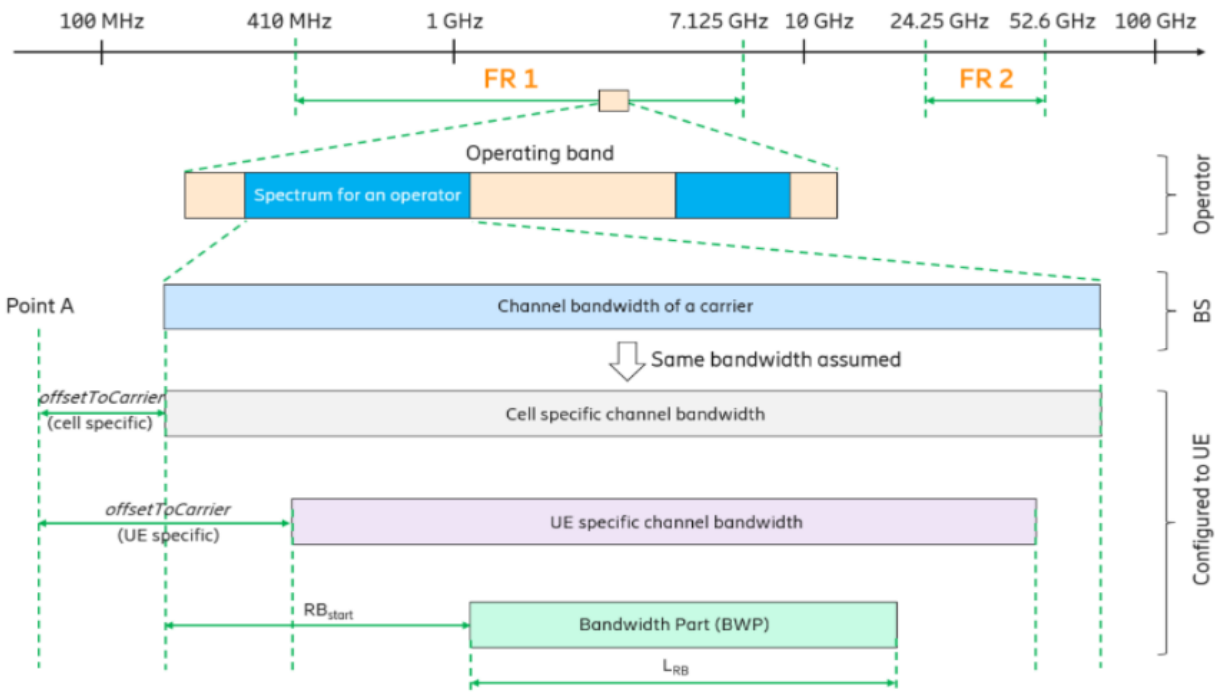


Figure 6: Bandwidth part and its relation to cell bandwidth

The UE can be configured with four of the BWPs in the downlink and the uplink. This is in addition to the original BWPs is configured by using the SIB1. Such as, UL, and there is also a supplementary uplink (SUL). The EU's four, I thought of myself as a SULs.

Even with multiple BWPS is installed, there can be only one of them is active at any given time; that is, the UE transmits and receives within the active BWP-and nothing else. DL PDSCH/PDCCH/the CSI-RS is received on the active DL BWP, the UE can use the measurement gaps to perform measurements other than the active BWP. UL PUSH-DOWN is being sent to the UE, with only the active UL BWP.

BWP switching, it means to turn off the currently BWP, and the activation of the other installed. In TDD, DL and the UL BWPS differ from each other only in that the transmission bandwidth and numerology; and to enable them to be together.

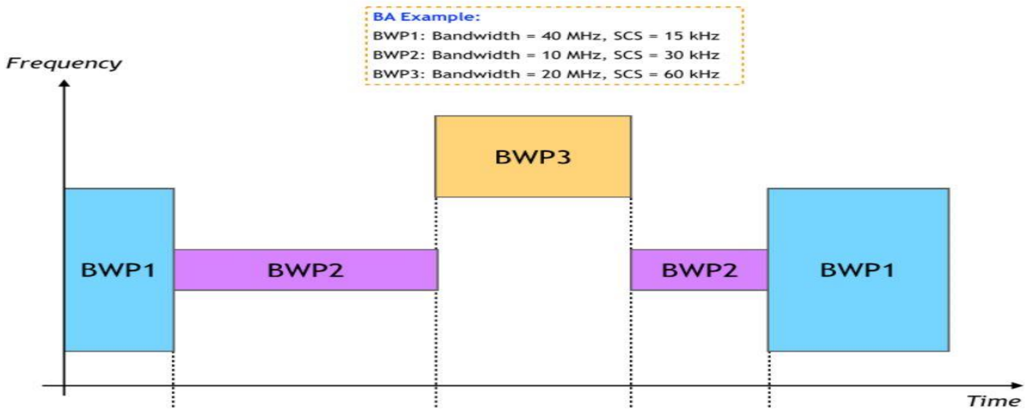

BA Example:
BWP1: Bandwidth = 40 MHz, SCS = 15 kHz
BWP2: Bandwidth = 10 MHz, SCS = 30 kHz
BWP3: Bandwidth = 20 MHz, SCS = 60 kHz
Frequency
BWP1
BWP2
BWP3
BWP2
BWP1
Time

Figure 7: A UE is configured with multiple BWPs but only one is active at a time.

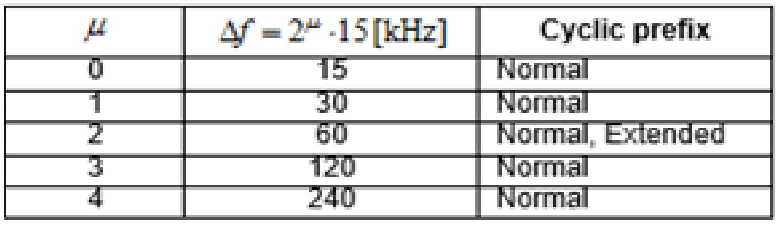
VI. Physical Layer:

The physical layer is the basis for each of the wireless communication technology and to be able to meet the physical layer of the system requirements, the 5G NR supports a wide range of frequencies from 1 Ghz to 100 Ghz and a wide range of services and deployment scenarios. NR is the first mobile radio technology, the mmWave frequency range, the focus is on the bandwidth of the channel in the Ghz frequency band and allows you to create massive multi-antenna systems. In this section we focused on frame structure, wave form showing subframes, slots and symbols structure according to 3GPP standers and release 15.

Transmission Numerology in 5G:

The NR numerology ranges from 15 kHz to 240 kHz, which is, in turn, will lead to a change in the length of cyclic prefix. 5G support CP-OFDM like LTE but with variety in subcarrier spacing rather than fixed 15 KHZ like in LTE. 5G support multiple numerologies.

Table 1 Supported Numerologies in 5G technology



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

Where μ is numerology and subcarrier spacing is defined by **$\Delta f = 2^{\mu} * 15$ KHZ.**

Frame Structure:

Frame of data in 5G is divide to 10 sub frames each subframe consists of number of slots and each slot contains multiple of OFDM symbols around 14 or 12 symbols according to CP type. Duration of each subframe =10 msec. the number of slots per sub frame is equal to 2^{μ} while the $N^{slots}_{symbols}$ is 10 therefore number of constructive OFDM symbols in subframe = $N^{slots}_{symbols}$ * $N^{subframe}_{slots}$.

the frame is divided into 2 half frames first half is containing (0:4) sub frames and the second one contains (5 :9).

There is only one set of frames for uplink and other one for downlink on a single carrier.

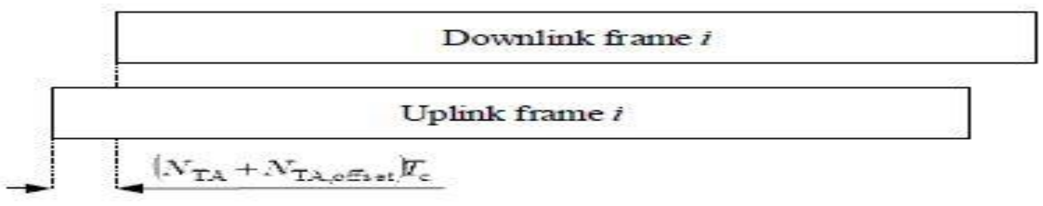


Figure 8: relation between uplink and downlink frames

Uplink frame number i for transmission from the UE shall start $T_{\mathrm{TA}} = (N_{TA} + N_{TA,offset})\, T_c$ before the start of the corresponding downlink frame at the UE where $N_{TA,offset}$ depends on the frequency band.

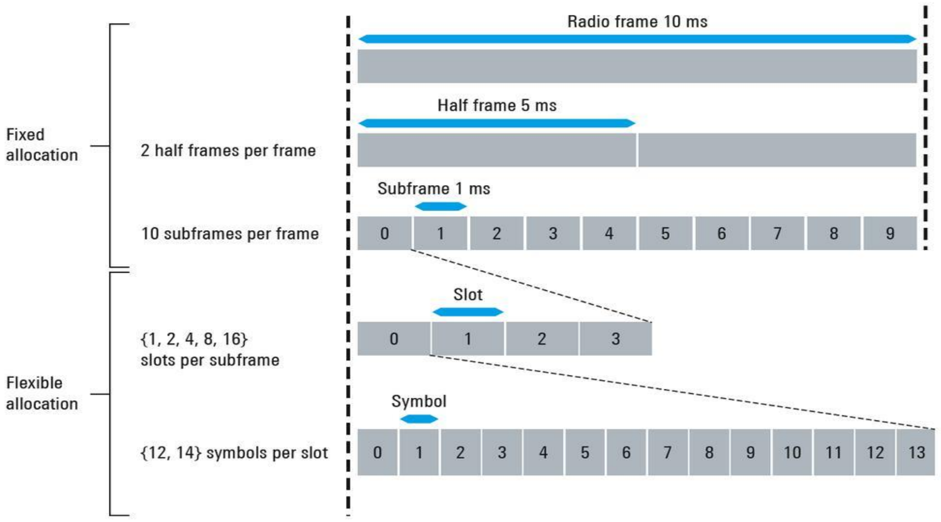
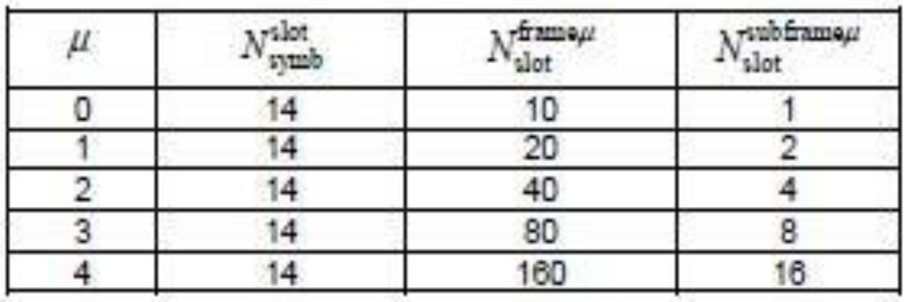


Figure 9: frame structure of data

Table 2 Number of OFDM symbols per slot, slots per frame slots per subframe



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

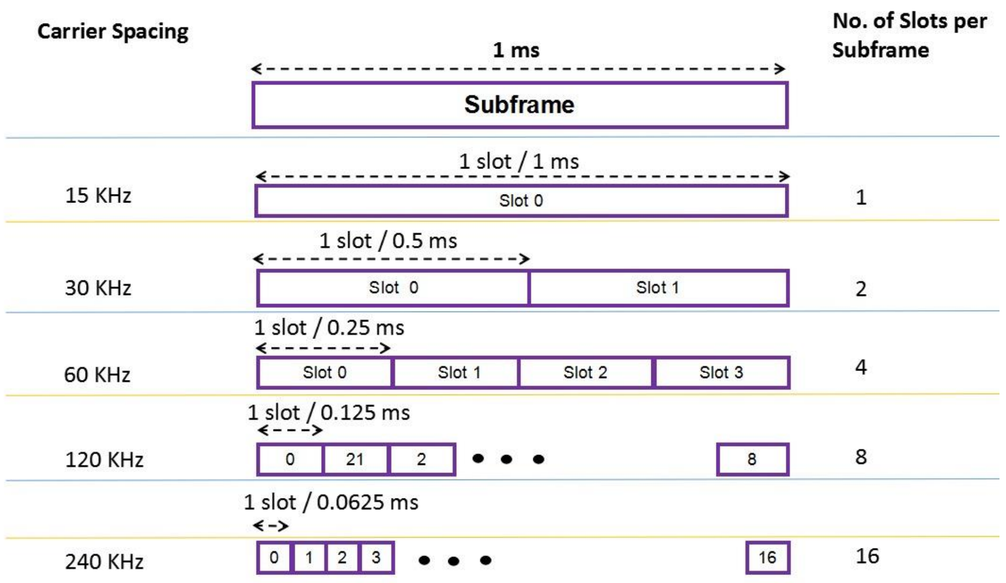
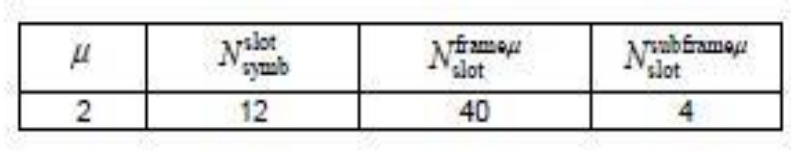


Figure 10: slots number according to numerology/carrier spacing

Table 3 Number of OFDM symbols per slot, slots per frame, and slots per sub frame



μ	N_{symb}^{slot}	$N_{slot}^{frame,\mu}$	$N_{slot}^{subframe,\mu}$
2	12	40	4

Duplex Mode:

Half Duplex Mode: it is bidirectional communication but **not simultaneous** (one at a time) each user must push to talk. The device is incapable of concurrent reception and transmission.

Full Duplex Mode: it is known as **simultaneous** two way of communication. The device here can concurrent the reception and transmission.

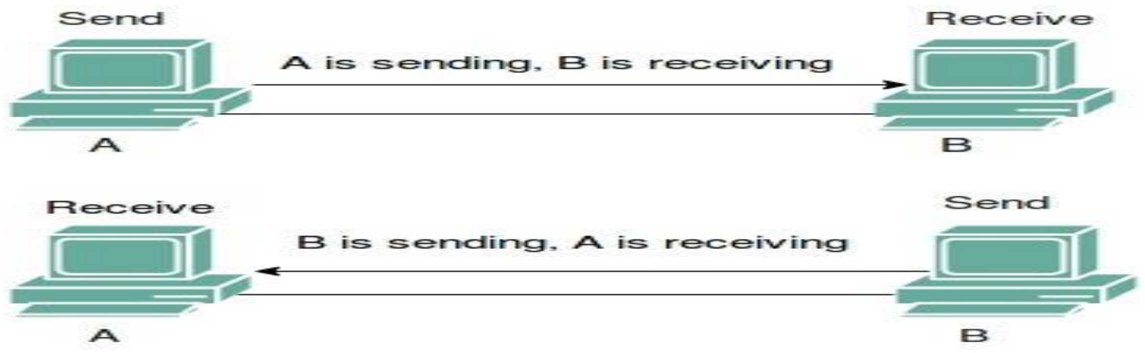


Figure 11: duplex mode

Frequency Division Duplex (FDD): UL and DL are separated in frequency but share the time slot in a frame.

Time Division Duplexing (TDD): UL and DL are separated in time slot but share the same carrier frequency. A guard interval may be provided where no transmission occurs helping in switching between UL and DL transmissions.

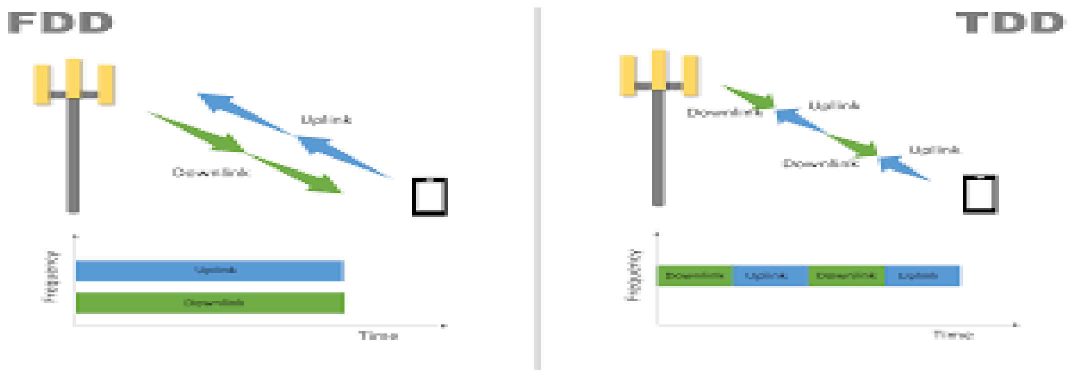


Figure 12: FDD Vs TDD

The duplex mode chosen for data transmission is depending on spectrum allocation. If we are operating in low frequency bands then allocation will be paired that is frequency division duplex and for high frequency bands unpaired spectrum allocation is used, named as time division duplex.

Full duplex technology has become an attractive solution for 5th generation programs to meet the ever-increasing demand for mobile traffic.

Complete duplex allows the node to transmit and receive simultaneously on the same frequency band, thus, in theory, doubling the system flow over normal duplex systems. An important limitation in building a complete duplex node is complete interference, i.e., interference caused by signal transmitted to the desired signal obtained from that same node. This system has been defeated due to recent advances in disruption technology. However, there are other limitations to achieving the full benefits of the theory: residual disturbances, traffic issues, and inter-cell and inter-cell disruption.

In 5G technology TDD has been chosen over FDD for many reasons:

• TDD works best as it requires a single frequency channel DL and UL.

• The features of physical layer such as precoding and beamforming are based on channel state information (CSI). Which makes TDD possible to use uplink channel measurement data to determine downlink orientation and orientation indicators.

• Duplexer is not needed.

• For TDD, it is possible to allocate a different number of UL and DL bands according to system requirements. Therefore, a large portion of the band is assigned to DL in DL data networks such as internet browsing and streaming. And the bulk of the band is assigned to UL-on-UL data networks such as camera hire.

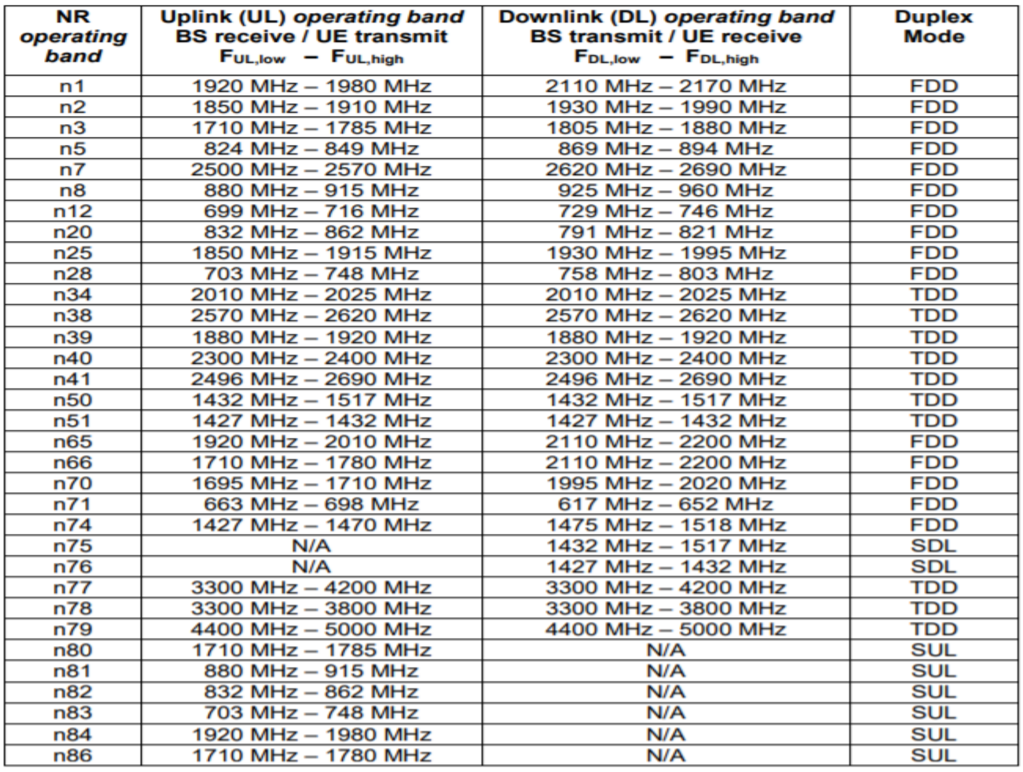
This applies to our use case (eMBB), while FDD is more suitable for other use cases such as URLLC as it offers lower delays due to simultaneous transfer and acceptance.

In addition to FDD and TDD groups, other groups are assigned to provide additional uplink and downlink capacity. SDL-marked bands are made for the bottom and SULs for additional uplinks.

In an additional uplink (SUL), when the channel status is correct, the Network tells UE to use the actual UL frequency and when the channel status becomes worse than a certain process, the Network directs the UE to use the additional UL frequency when the frequency The extra UL is less likely to ensure the availability of larger cells.

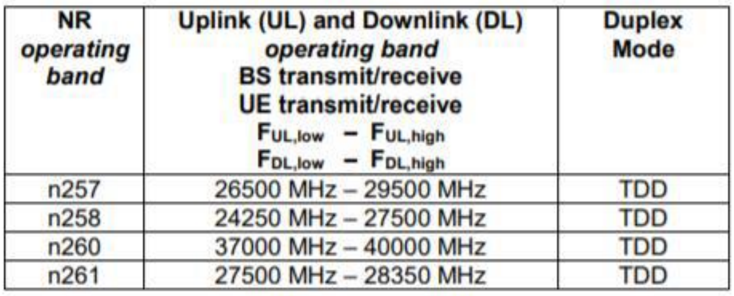
NR operating bands defined by:

Table 4: Operating bands in FR1



NR *operating band*	Uplink (UL) *operating band* BS receive / UE transmit $F_{UL,low}$ – $F_{UL,high}$	Downlink (DL) *operating band* BS transmit / UE receive $F_{DL,low}$ – $F_{DL,high}$	Duplex Mode
n1	1920 MHz – 1980 MHz	2110 MHz – 2170 MHz	FDD
n2	1850 MHz – 1910 MHz	1930 MHz – 1990 MHz	FDD
n3	1710 MHz – 1785 MHz	1805 MHz – 1880 MHz	FDD
n5	824 MHz – 849 MHz	869 MHz – 894 MHz	FDD
n7	2500 MHz – 2570 MHz	2620 MHz – 2690 MHz	FDD
n8	880 MHz – 915 MHz	925 MHz – 960 MHz	FDD
n12	699 MHz – 716 MHz	729 MHz – 746 MHz	FDD
n20	832 MHz – 862 MHz	791 MHz – 821 MHz	FDD
n25	1850 MHz – 1915 MHz	1930 MHz – 1995 MHz	FDD
n28	703 MHz – 748 MHz	758 MHz – 803 MHz	FDD
n34	2010 MHz – 2025 MHz	2010 MHz – 2025 MHz	TDD
n38	2570 MHz – 2620 MHz	2570 MHz – 2620 MHz	TDD
n39	1880 MHz – 1920 MHz	1880 MHz – 1920 MHz	TDD
n40	2300 MHz – 2400 MHz	2300 MHz – 2400 MHz	TDD
n41	2496 MHz – 2690 MHz	2496 MHz – 2690 MHz	TDD
n50	1432 MHz – 1517 MHz	1432 MHz – 1517 MHz	TDD
n51	1427 MHz – 1432 MHz	1427 MHz – 1432 MHz	TDD
n65	1920 MHz – 2010 MHz	2110 MHz – 2200 MHz	FDD
n66	1710 MHz – 1780 MHz	2110 MHz – 2200 MHz	FDD
n70	1695 MHz – 1710 MHz	1995 MHz – 2020 MHz	FDD
n71	663 MHz – 698 MHz	617 MHz – 652 MHz	FDD
n74	1427 MHz – 1470 MHz	1475 MHz – 1518 MHz	FDD
n75	N/A	1432 MHz – 1517 MHz	SDL
n76	N/A	1427 MHz – 1432 MHz	SDL
n77	3300 MHz – 4200 MHz	3300 MHz – 4200 MHz	TDD
n78	3300 MHz – 3800 MHz	3300 MHz – 3800 MHz	TDD
n79	4400 MHz – 5000 MHz	4400 MHz – 5000 MHz	TDD
n80	1710 MHz – 1785 MHz	N/A	SUL
n81	880 MHz – 915 MHz	N/A	SUL
n82	832 MHz – 862 MHz	N/A	SUL
n83	703 MHz – 748 MHz	N/A	SUL
n84	1920 MHz – 1980 MHz	N/A	SUL
n86	1710 MHz – 1780 MHz	N/A	SUL

Table 5: Operating bands in FR2



NR *operating band*	Uplink (UL) and Downlink (DL) *operating band* BS transmit/receive UE transmit/receive $F_{UL,low}$ – $F_{UL,high}$ $F_{DL,low}$ – $F_{DL,high}$	Duplex Mode
n257	26500 MHz – 29500 MHz	TDD
n258	24250 MHz – 27500 MHz	TDD
n260	37000 MHz – 40000 MHz	TDD
n261	27500 MHz – 28350 MHz	TDD

VII. Implementation of PDSCH RECEIVER Chain:

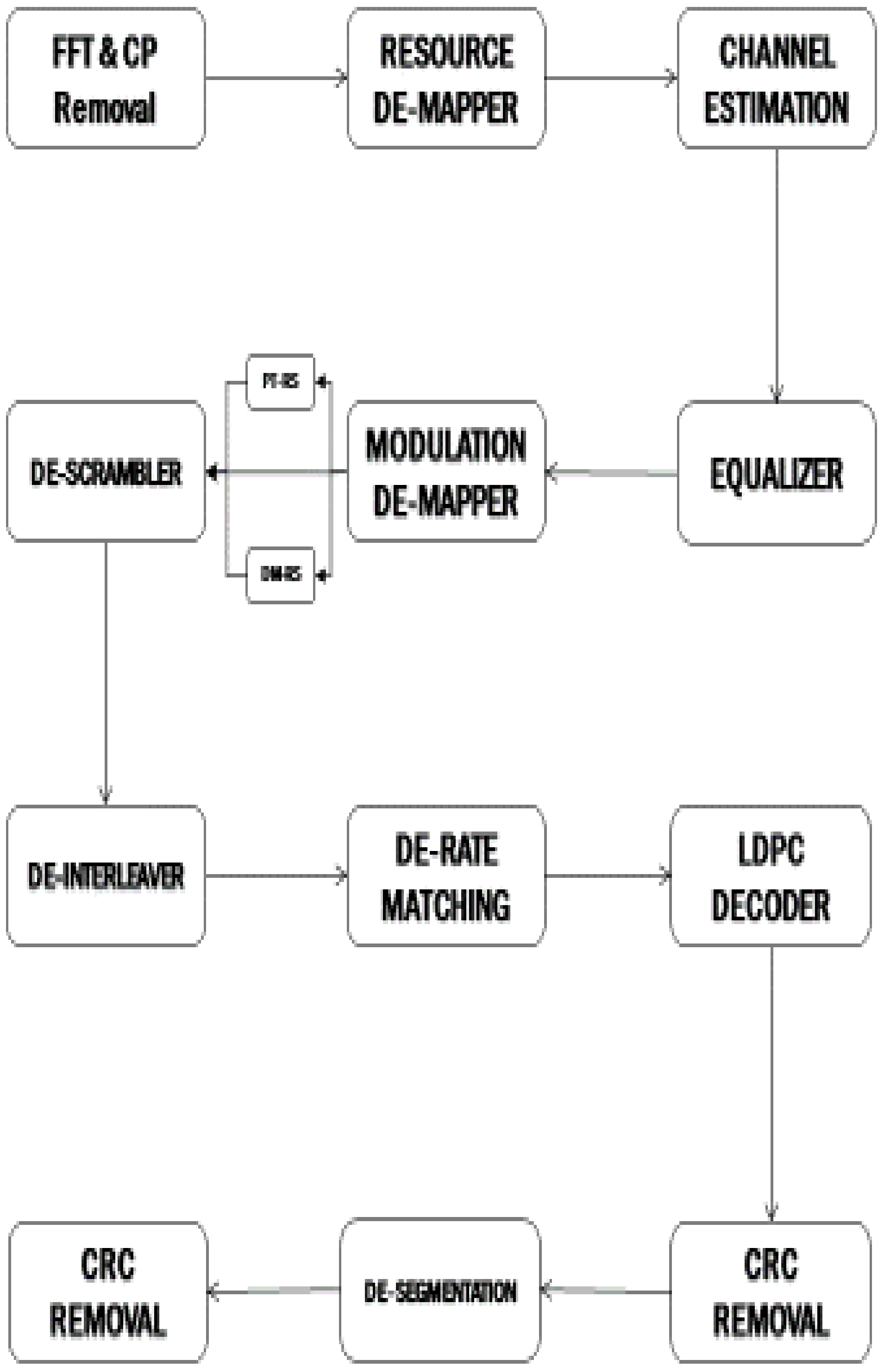


Figure 13: Block chain of PDSCH receiver

Fixed point representation

Nowadays most computers use two kind of numbers representation, floating point and fixed point, since all numbers are encoded using binary digits (0, 1). Floating-point variables are declared as float (32 bits) or double (64 bits), while integer fixed-point variables are declared as short integer.

Fixed point numbers have a decimal in a fixed position and floating-point numbers have a sign. Both types of numbers are set up in sections, and there's a placeholder for every portion of a number. Fixed point numbers have a certain number of reserved digits that are on the left side of the decimal for the integer portion of the number. The numbers to the right of the decimal point are reserved for the fractional part of the number.

We have chosen 16 bits representation as we have found it gives less error performance and more precise results as in fixed point representation memory and bus widths are smaller, leading to definitively lower cost and power consumption. Moreover, floating-point operators are more complex, having to deal with the exponent and the mantissa, and hence, their area and latency are significantly greater than those of fixed-point operators. In spite of that, fixed-point arithmetic introduces an unalterable quantization error which modifies the application functionalities and degrades the desired performance. Thus, the design flow requires a floating-to-fixed-point conversion stage which optimizes the implementation cost under execution time and accuracy constraints.

We have chosen 1 bit integer because we use 64-QAM modulation scheme, which consists of eight constellation points for each set, number of sets=3, the highest absolute value of the constellation point $= 4\sqrt{E_s}\ and\ E_s = 6\ E_{b0}\ , where\ E_{bit}$ which approximately equal 1.8, so we only need one bit for integer representation.

$$E_s : Energy\ of\ symbol$$

1 bit sign	1 bit integer	14 bits fractional part

OFDM & CP Removal:

System logic:

In this block receive the data from transmitter and remove CP from each OFDM symbol. We remove the CP part in the same way the CP was inserted after each OFDM symbol. After the CP removal, the fast Fourier transform of the data vector per OFDM symbol is carried out. The FFT length is 1024 here. After doing this again the data comes in the frequency domain. The need to take the data in the frequency domain is because of the simplicity to implement simpler receiver structures.

The problem of frequency selective channels is a significant one in wireless communications. An equalizer can be used to invert the channel and cancel the effect of channel selectivity, but it is too complex to implement if the channel is too selective. A better approach to combat selective channels Is orthogonal frequency division multiplexing (OFDM) in which the channel is divided into many small subchannels, converting the selective channel into small flat subchannels. At the transmitter, the subcarriers are modulated in frequency domain, then IFFT is used to transform this into time domain samples. At the receiver, the time domain samples are transformed back to frequency domain using FFT for further processing.

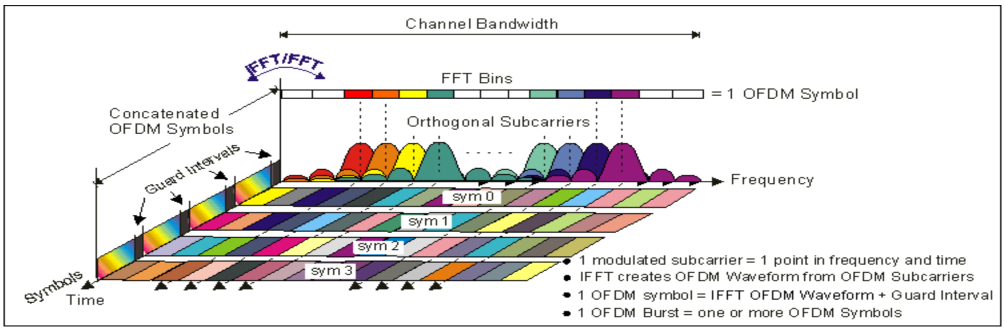
Fourier transforms break a signal down into its frequency components. This process involves writing a signal as a summation of sines and cosines. The fast Fourier transform (FFT) reduces the number of calculations of the DFT by dividing the initial function into repeated sub functions and continues this process until the sub function is no longer divisible.

Fixed-point FFTs are used because input data frequently comes directly from an analog-to-digital converter (A/D). A/Ds typically provide a stream of fixed-point data, so using a fixed-point FFT eliminates the need to convert the data into another format. Not only has that but Integer math tended to execute faster than floating-point calculations. However, some problems are intrinsic to using fixed-point mathematics. Signed short integers cover a much smaller range of values than floating-point values. Therefore, overflow and underflow during mathematical operations are a concern. Precautions to prevent overflow and underflow can add extra instructions, decreasing performance, and/or may scale the input data limiting the accuracy and precision of the algorithm. To avoid decreasing performance we work on 8-bit representation.

OFDM:

Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier module system that expands the concept of single subcarrier fluctuations using multiple subcarriers in a single channel. Instead of transmitting high-speed data streams with a single slide, OFDM uses many transmissions under orthogonal subcarriers transmitted uniformly. Each carrier is built with a standard digital voice fluctuating scheme (such as QPSK, 16QAM… etc.) with a low level of signal. However the combination of many subcarriers enables data rates like conventional single-carrier modulation schemes within equivalent bandwidths.

In the context of the frequency, most of the nearby tones or individual subcarriers are organized independently with complex details. Inverse FFT conversion is performed on frequency-domain subcarriers to produce the OFDM signal in the time zone. Then in the time zone, intervals are placed between all signals to prevent signal interference between the receiver caused by multiple road delays on the radio station. For the receiver FFT is performed on OFDM signals to get bits of real data.



Channel Bandwidth
FFT/IFFT
FFT Bins
= 1 OFDM Symbol
Concatenated OFDM Symbols
Orthogonal Subcarriers
Guard Intervals
Frequency
sym 0
sym 1
sym 2
sym 3
Symbols
Time
1 modulated subcarrier = 1 point in frequency and time
IFFT creates OFDM Waveform from OFDM Subcarriers
1 OFDM symbol = IFFT OFDM Waveform + Guard Interval
1 OFDM Burst = one or more OFDM Symbols

Figure 14 Frequency and time representative of a signal

OFDM signal can be defined as a group of closely related FDM sub-vehicles. At the frequency range, each passenger leads to the performance of a side-by-side since that produces a scattering of sight between subcarriers. This causes subcarrier disturbance without orthogonally separated waves. In orthogonal waves, the peaks of each of the underlying objects are all aligned with the layers of other underlying objects. This combination of spectral power does not affect the system's ability to detect the original signal. The receiver repeats the incoming signal with a known set of sinusoids to retrieve the actual set of sent bits. The use of orthogonal subcarriers allows for more than one bandwidth which leads to an increase in spectral efficiency. For the full signal of OFDM, Orthogonality prevents interference between dispersal carriers. In FDM programs, any overlap of signal signals nearby will result in interference. In OFDM systems, the lower carriers will only interfere with each other if there is a loss of orthogonality. For example, a frequent error would cause subcarrier waves to shift so that spectral nulls could no longer be aligned leading to inter-subcarrier-interference.

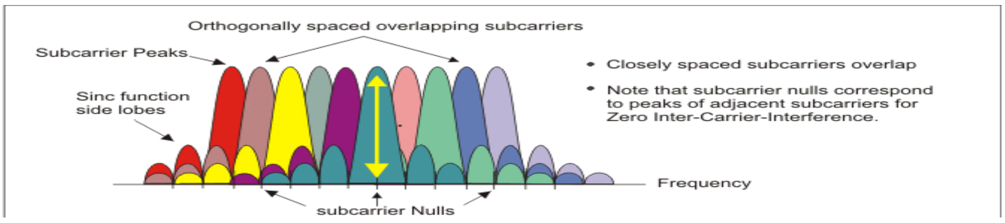

Orthogonally spaced overlapping subcarriers
Subcarrier Peaks
Sinc function side lobes
• Closely spaced subcarriers overlap
• Note that subcarrier nulls correspond to peaks of adjacent subcarriers for Zero Inter-Carrier-Interference.
Frequency
subcarrier Nulls

Figure 15: OFDM signal frequency spectrum

OFDM PROS:

- Eliminate ISI
- Improve SNR
- Achieve diversity gain
- Low senstive to time synchronization
- Easily adapt server channel condition without time quantization
- High data rate
- Enable parallel transmission
- Bandwidth efficient

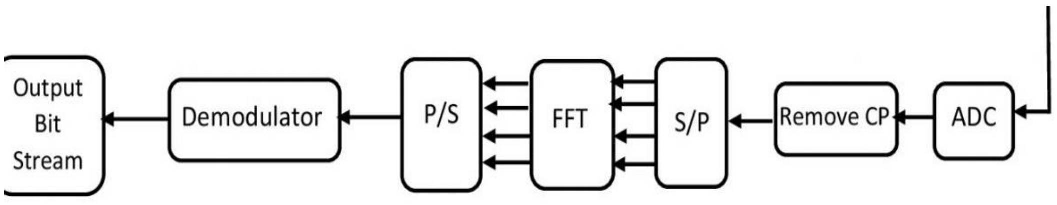


Figure 16: OFDM block diagram

Cyclic prefix:

CP can be included in the monitoring period to reduce ICI. Repeat the sample point following each OFDM symbol before the OFDM symbol. This ensures that the number of waveform times incorporated into the latency copy of the OFDM signal is the number in the FFT period, which ensures the carrier's network carrying. Copying the end of upload and transfer as a cyclic start ensures that there is a ‘circular’ connection between the sent signal and the channel response. This allows the recipient to use a simple repetition to capture power for all delayed items. If the ‘circular’ diagnosis were not completed the recipient would receive an ICI when he or she eliminated the frequency of the normal domain.

Key Factors to Determining CP Length:

Multi path Delay: The multiple and CP length is directly proportional. The larger the multipath delay, requires longer Cyclic Prefix

Length of OFDM Symbol: Given the same OFDM symbol length, a longer CP can be a large system overhead, so to control over overhead the length of CP shall be selected as appropriate.

Design of CP in 5G NR:

The basic CP wish for NR is the same as for LTE with the same placement as for LTE. The CP design ensures that it corresponds to the signals between the various SCS values and the reference number (15 kHz). For example, µ = 15 khz one slot has about 7 symbols lasting 0.5 m2 seconds including CP symbols each and μ = 30 khz one slot has about 14 symbols including CP of each symbol inside of 0.5 milli sec. So here the CP length is adjusted based on the subcarrier (fsc) space.

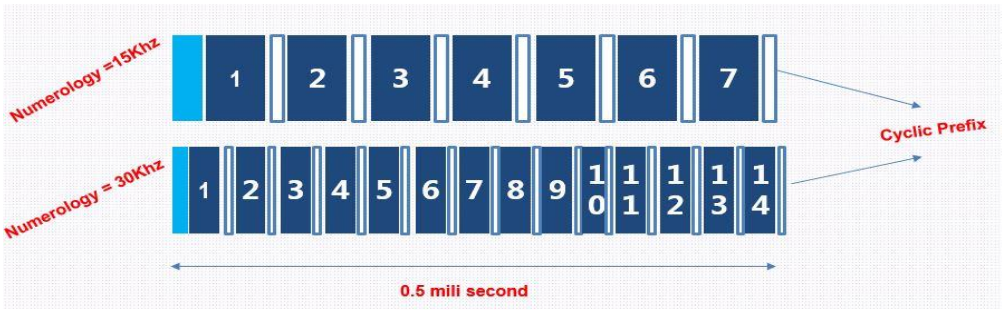


Figure 17: OFDM symbol duration

Properties of CP in 5G:

- 3GPP has specified two types of CPs, Normal Cyclic Prefix (NCP) and Extended Cyclic Prefix (ECP).
- The NCP is specified for all subcarrier spacings
- ECP is currently only specified for the 60 kHz subcarricr spacing.
- If normal CP (NCP) is used, the CP of the first symbol present every 0.5 ms is longer than that of other symbols.
- Cyclic prefix durations decrease as the subcarrier spacing increase. CP Length for Different Subcarriers
- The CP length for different sub-carrier can be calculated

$$N_{CP,l}^{\mu} = \begin{cases} 512\kappa.\, 2^{-\mu} & \text{extended cyclic prefix} \\ 144\kappa.\, 2^{-\mu} + 16\kappa & \text{normal cyclic prefix}, l = 0 \text{ or } l = 7.2^{\mu} \\ 144\kappa.\, 2^{-\mu} & \text{normal cyclic prefix}, l \neq 0 \text{ and } l \neq 7.2^{\mu} \end{cases}$$

and CP time duration can be using following formula

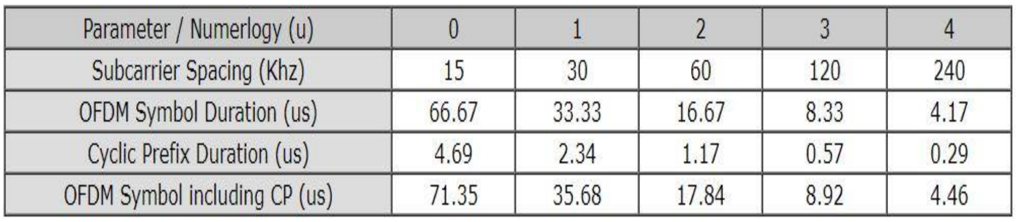
$$T_{cp} = N_{cp}.\, T_c$$

u is numerology, l is the symbol index here and K is a constant to relate NR basic time unit and LTE basic time unit and can be represented by following equation.

$$K = \frac{T_s}{T_c} = 64$$

T_s is LTE basic time unit and **T_c** is NR basic time unit.

Table 6: Different OFDM symbol durations

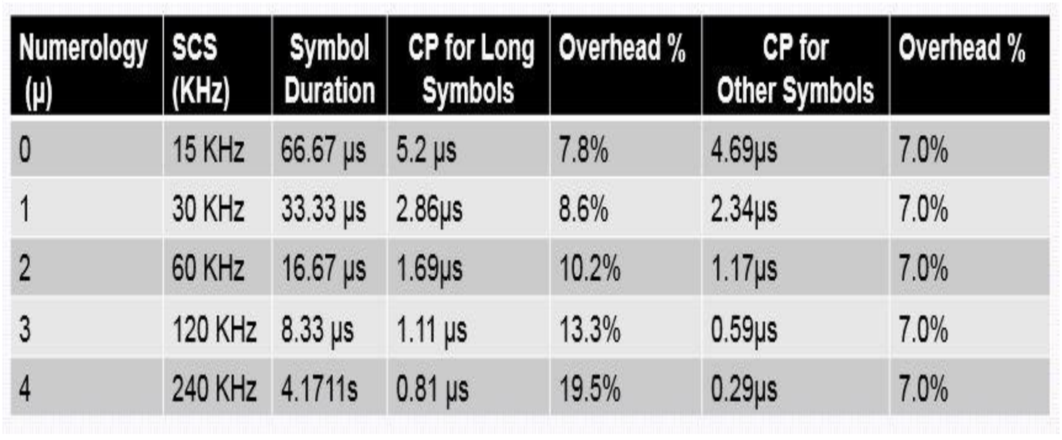


Parameter / Numerlogy (u)	0	1	2	3	4
Subcarrier Spacing (Khz)	15	30	60	120	240
OFDM Symbol Duration (us)	66.67	33.33	16.67	8.33	4.17
Cyclic Prefix Duration (us)	4.69	2.34	1.17	0.57	0.29
OFDM Symbol including CP (us)	71.35	35.68	17.84	8.92	4.46

Calculating CP Overhead

The CP head is the average percentage of CP time and Symbol duration, for example 15KHz the NR signal duration is 66.67 µs and the CP duration is 5.2ss. After that can be calculated 5.2 / 66.67 = 7.8%. Here the longer the signal will be higher as the CP whereas the other signal will be higher. Below the table is provided a top-level summary of General CP with various management spaces.

Table 7: CP over headers



Numerology (µ)	SCS (KHz)	Symbol Duration	CP for Long Symbols	Overhead %	CP for Other Symbols	Overhead %
0	15 KHz	66.67 µs	5.2 µs	7.8%	4.69µs	7.0%
1	30 KHz	33.33 µs	2.86µs	8.6%	2.34µs	7.0%
2	60 KHz	16.67 µs	1.69µs	10.2%	1.17µs	7.0%
3	120 KHz	8.33 µs	1.11 µs	13.3%	0.59µs	7.0%
4	240 KHz	4.1711s	0.81 µs	19.5%	0.29µs	7.0%

Fast Fourier Transform

Introduction

The main reason behind using 5G is to increase accessibility and get rid of limitations of former generations, 5G needs to provide the services a fast and high speed in order to increase efficiency of these systems. That’s why Fast Fourier Transform is used, it’s considered one of the most important blocks in digital communications systems.

FFT is used to transform received time domain signals to frequency domain for the receiver to be able to perform demodulation and decoding on it what happens is the signal is sampled over a period of time and divided into its frequency components. These components are single sinusoidal oscillations at different frequencies each with their own amplitude and phase.

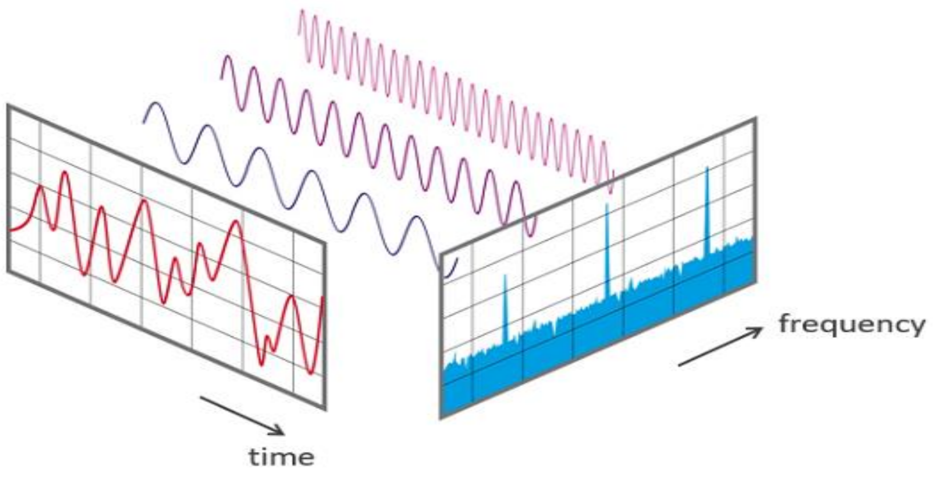


Figure 18 Representation of a signal in the time and frequency domain

Fast Fourier Transform is an efficient algorithm developed by Cooley & Tukey in 1965, used to compute the DFT with reduced computations. Due to the efficiency of FFT, it is used for spectrum analysis, convolutions, correlations, and linear filtering.

FFT reduces the problem of calculating an N – point DFT to calculating many smaller – sized DFTs. It is defined by the following parameters with a number of equally spaced samples N:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}}$$

While its Twiddle factor is defined by:

$$W_N = \frac{e^{-j2\pi}}{N}$$

This Twiddle factor represents a rotating vector that contains trigonometric coefficients that are multiplied by the data which differ according to the number of sample points of DFT, N.

The properties of the twiddle factor W_N used in this algorithm are:

1- $\mathrm{W_N^{k+\frac{N}{2}} = e^{-j2\pi kN}\ e^{-j\pi} = -W_N^k}$ (Symmetry Property)

2- $\mathrm{W_N^{k+N} = e^{-j2\pi kN}\ e^{-j2\pi} = W_N^k}$ (Periodicity property)

3- $\mathrm{W_N^m = e^{-\frac{j2\pi m}{N}} = e^{-\frac{j2\pi m}{N}} = W_{\frac{N}{m}}}$

The Decimation – In – Time (DIT) and Decimation – In – Frequency (DIF) FFT algorithms use the "divide – and – conquer" approach. This is possible if the length of the sequence N is chosen as N = $\mathrm{r^m}$. Here, r is called the radix of the FFT algorithm. The most practically implemented choice for r = 2 leads to radix – 2 FFT algorithms. So, with N = 2^{m}, the efficient computation is achieved by breaking the N – point DFT into two $\frac{N}{2}$ - point DFTs, then breaking each $\frac{N}{2}$ - point DFT into two $\frac{N}{4}$ - point DFTs and continuing this process until 2 – point DFTs are obtained. For N=8, the Decimation – In – Time algorithm decomposition would be as shown in figure 20.

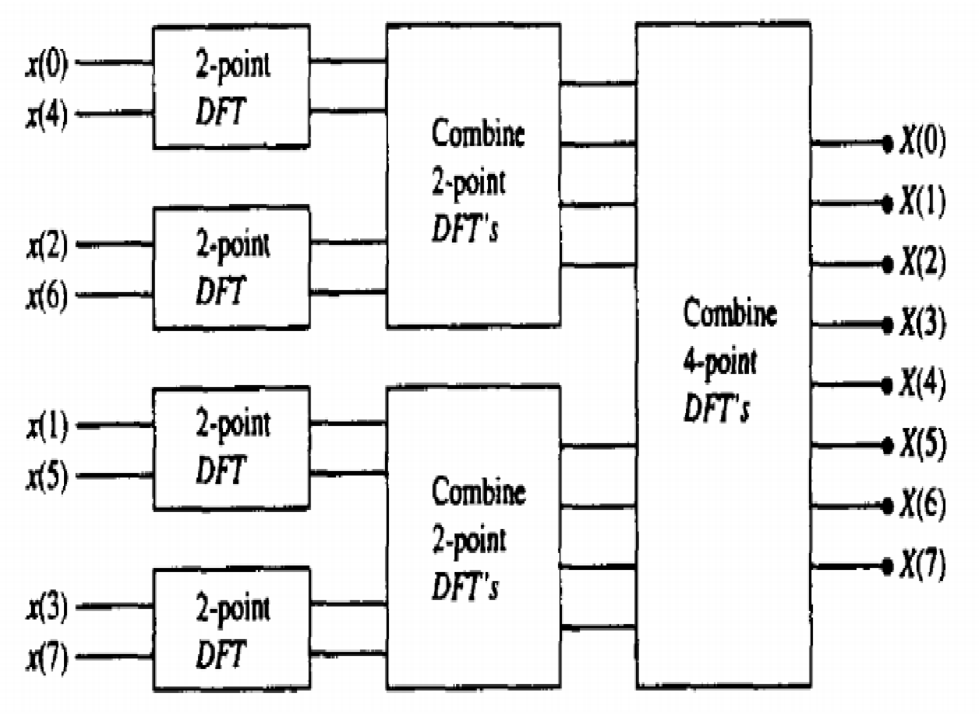


Figure 19: Decimation in time FFT algorithm

In this algorithm, the time – domain sequence x[n] is decimated into two $\frac{N}{2}$ – point sequences, one composed of even – indexed values of x[n], and other composed of odd – indexed values of x[n].i.e.,

$g[n] \; = \; x[2n] \ldots \ldots \ldots .eqn. 1$

$h[n] \; = \; x[2n \; + \; 1] \ldots \ldots \ldots .eqn. 2$

The N – point DFT of x[n] is given by

$$X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk}, k = 0,1, \ldots., N - 1$$

This can be rewritten as

$$X(k) = \sum_{n=0,\text{even}}^{N-1} x[n]W_N^{nk} + \sum_{n=0,\text{odd}}^{N-1} x[n]W_N^{nk}$$

$$= \sum_{n=0,}^{\frac{N}{2}-1} x[2n]W_N^{2nk} + \sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_N^{(2n+1)k}$$

$$= \sum_{n=0,}^{\frac{N}{2}-1} x[2n]W_N^{2nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_N^{2nk}$$

Using the third property of the twiddle factor, W_N, the above equation can be rewritten as

$$= \sum_{n=0,}^{\frac{N}{2}-1} x[2n]W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x[2n+1]W_{\frac{N}{2}}^{nk}$$

Using equations 1 & 2 in the above equation, we obtain

$$X(k) = \sum_{n=0,}^{\frac{N}{2}-1} g[n]W_{\frac{N}{2}}^{nk} + W_N^k \sum_{n=0}^{\frac{N}{2}-1} h[n]W_{\frac{N}{2}}^{nk}$$

Or $X(k) = G(k) + W_N^k H(k)) \ \ldots \ \ldots \ \text{eqn. } 3$

Where G(k) and H(k) are the N/2 – point DFTs of g[n] and h[n] respectively. So, G(k) and H(k) are periodic with period N/2 .i.e.

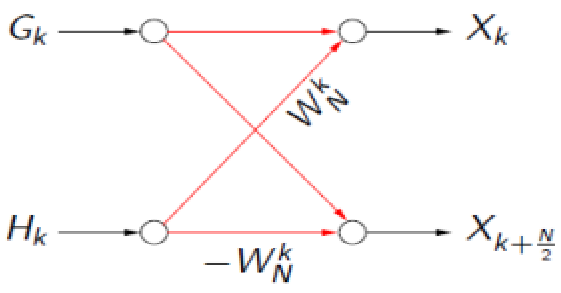
$$G\left(k + \frac{N}{2}\right) = G(k) \ldots \ldots \text{eqn. } 4$$

$$H\left(k + \frac{N}{2}\right) = H(k) \ldots \ldots \text{eqn. } 5$$

And using the symmetry property of the twiddle factor, W_N, and equations 4 & 5

$$X\left(k + \frac{N}{2}\right) = G(k) - W_N^k H(k) \ldots \text{eqn. } 6$$

Equations 3 and 6 result in the following butterfly diagram



For example, for N = 8, the DFT points in terms of G and H are

$$X(0) = G(0) + W_N^0 H(0)$$

$$X(1) = G(1) + W_N^1 H(1)$$

$$X(2) = G(2) + W_N^2 H(2)$$

$$X(3) = G(3) + W_N^3 H(3)$$

For the remaining 4 points X (4) to X (7), we use equations 4, 5, 6 and 7 to get

$$X(4) = G(0) - W_N^0 H(0)$$

$$X(5) = G(1) - W_N^1 H(1)$$

$$X(6) = G(2) - W_N^2 H(2)$$

$$X(7) = G(3) - W_N^3 H(3)$$

The butterfly diagram for the above set of equations is

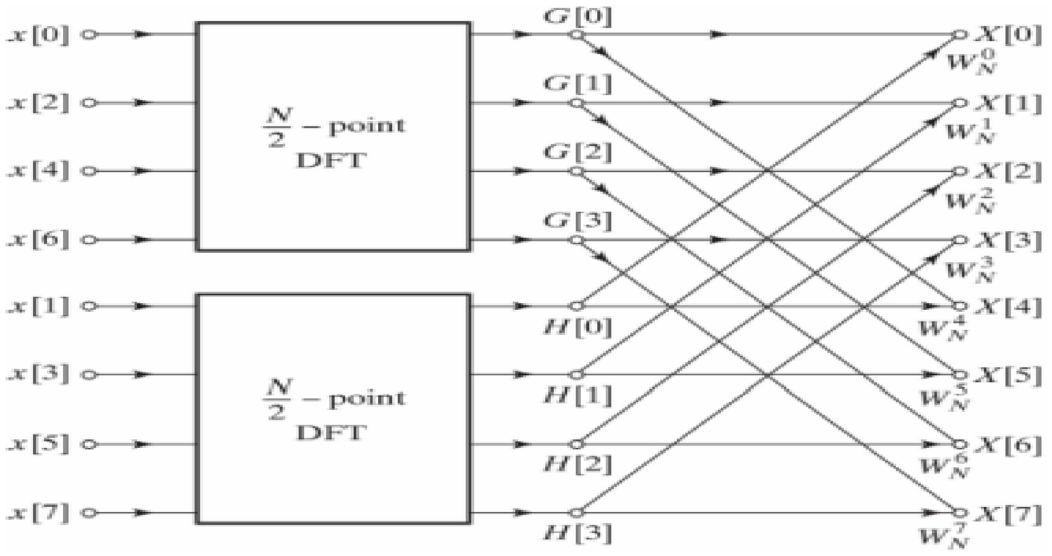


Figure 20 Example of butterfly 8-point FFT

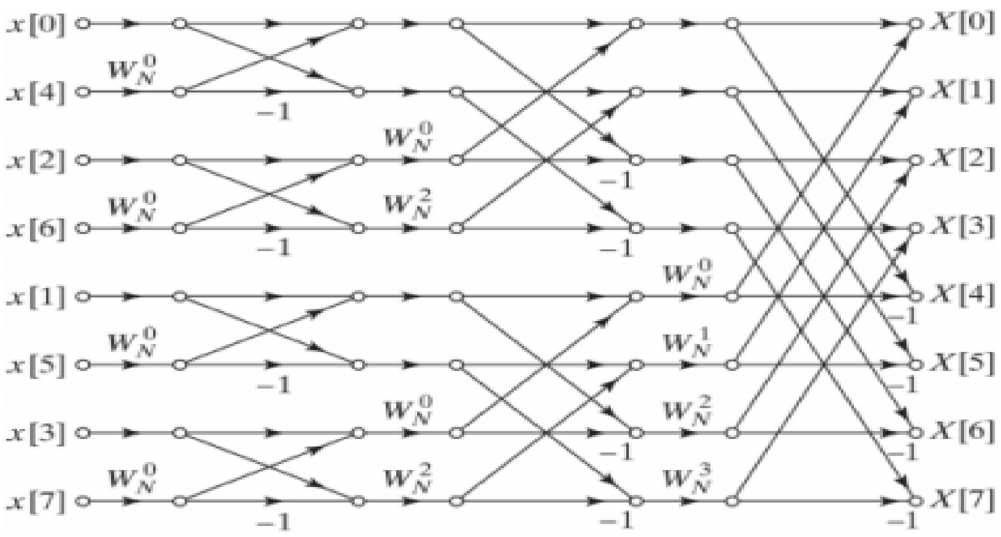
The above process is repeated for calculating the N/2 point DFTs of g[n] and h[n], and this is continued till we get two-point DFTs. Once we reach a two – point sequence, say p[n]={p [0], p[1]}, its 2 – point DFT would be

$$P(k) = \sum_{n=0}^{1} p[n]W_2^{nk}, k = 0,1$$

$$P(0) = p[0] + p[1]$$

$$P(1) = p[0] + W_2 p[1] = p[0] - p[1]$$

The overall butterfly diagram for DIT FFT algorithm for N =8 is



Due to repeated decimations, the input sequence is scrambled, and the order of the final input sequence is obtained as follows

Table 8: Bit reverse for DIT class of FFT

Input sequence	Index	Binary form of index	Bit reversed form of index	Decimal representation	Final input sequence order
X[0]	0	000	000	0	X[0]
X[1]	1	001	100	4	X[4]
X[2]	2	010	010	2	X[2]
X[3]	3	011	110	6	X[6]
X[4]	4	100	001	1	X[1]
X[5]	5	101	101	5	X[5]
X[6]	6	110	011	3	X[3]
X[7]	7	111	111	7	X[7]

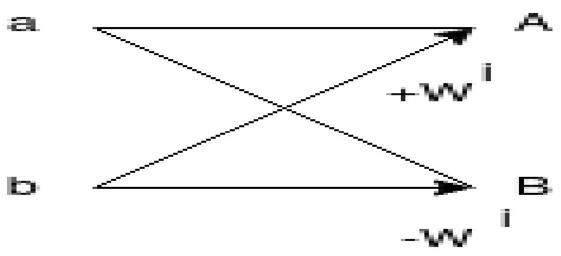
Efficiency of FFT Algorithm

A direct computation of DFT requires many multiplications and additions. An N – point DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x[n]W_N^{nk}, k = 0,1, \ldots., N-1$$

In the above equation, each DFT point computation involves N complex multiplications and (N – 1) complex additions. So, the N – point DFT calculations involve N^2 complex multiplications and N (N – 1) complex additions. This means approximately 106 complex multiplications and additions for a 1024 – point sequence.

On the other hand, in an FFT algorithm, a basic butterfly in DIT – FFT algorithm is represented as shown in the below the basic butterfly unit (2-inputs, 2-outputs) and the equations generated from the input values a and b.



$$A = a + b\, W_{N-pt}^{no\ of\ twiddle\ factor}$$

$$B = a - b\, W_{N-pt}^{no\ of\ twiddle\ factor}$$

- Number of stages in a butterfly diagram = $\log_2 N$
- Number of butterflies in each stage = $\frac{N}{2}$
- Number of complex multiplications in each butterfly = 1
- Number of complex additions in each butterfly = 2

From the above, for an N – point FFT algorithm,

- Total number of complex multiplications = $\frac{N}{2} log_2 N$
- Total number of complex additions = $N log_2 N$

For a 1024 – point sequence, this means approximately 5120 complex multiplications and 10240 complex additions. i.e., approximately 100 times less additions and 200 times less multiplications than direct computation of DFT.

As the number of input samples increase, the savings in the number of computations also increase.

Hardware Algorithm:

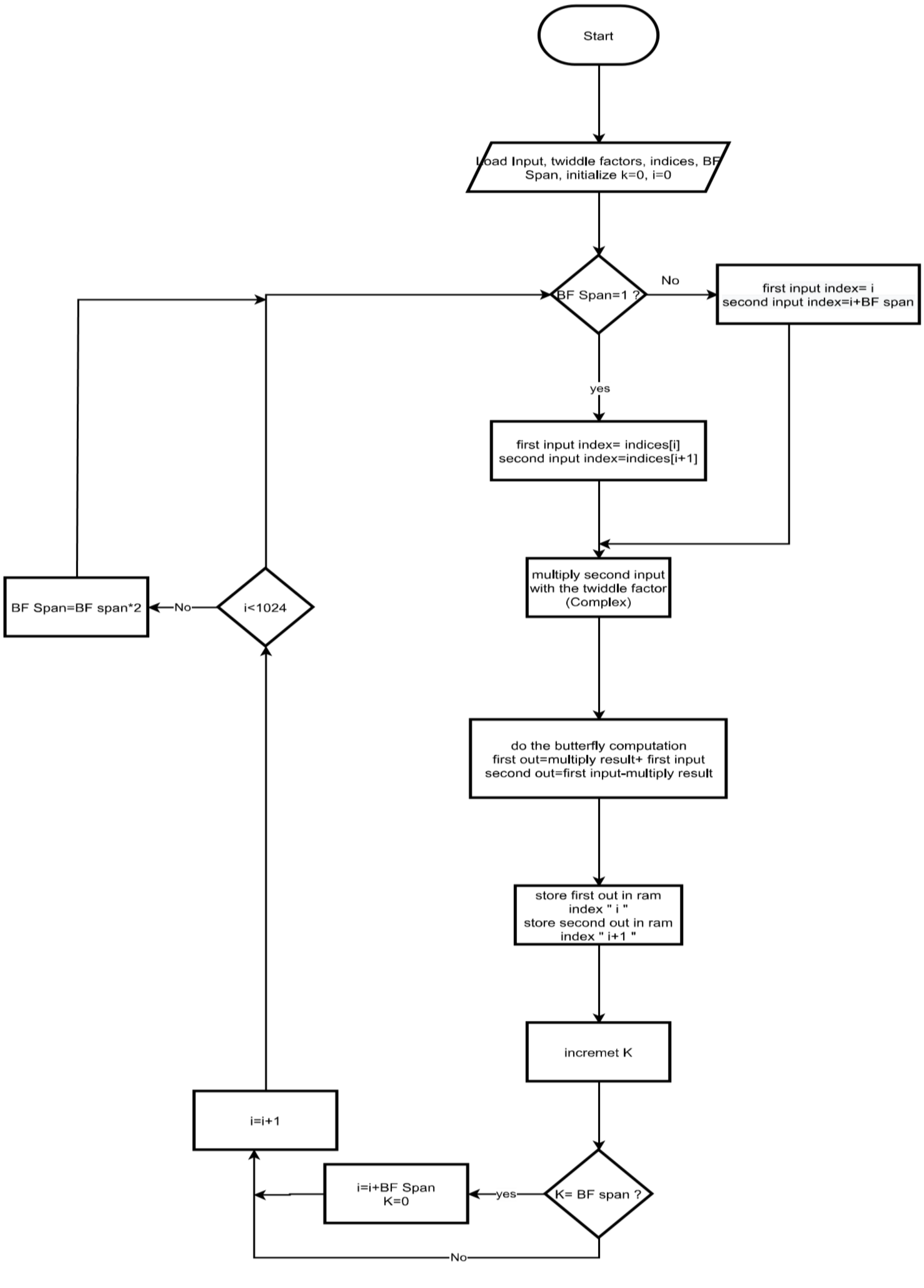

Start
Load Input, twiddle factors, indices, BF Span, initialize k=0, i=0
BF Span=1 ?
No
first input index= i
second input index=i+BF span
yes
first input index= indices[i]
second input index=indices[i+1]
multiply second input with the twiddle factor (Complex)
do the butterfly computation
first out=multiply result+ first input
second out=first input-multiply result
store first out in ram index " i "
store second out in ram index " i+1 "
incremet K
K= BF span ?
yes
i=i+BF Span
K=0
No
i=i+1
i<1024
No
BF Span=BF span*2

Figure 21: Hardware algorithm of FFT

FFT Block Interface:

This thesis contains the implementation of 1024 point FFT using radix-2 Cooley-Tukey represented in decimation-in-time, we use 1024-point because the resource grid used in implementation has 948 subcarriers and 14 symbols per frame, following divide and conquer basic approach the 1024-point is divided into 10 stages and needs $2^{\frac{N-point}{2}}$ twiddle factors per stage

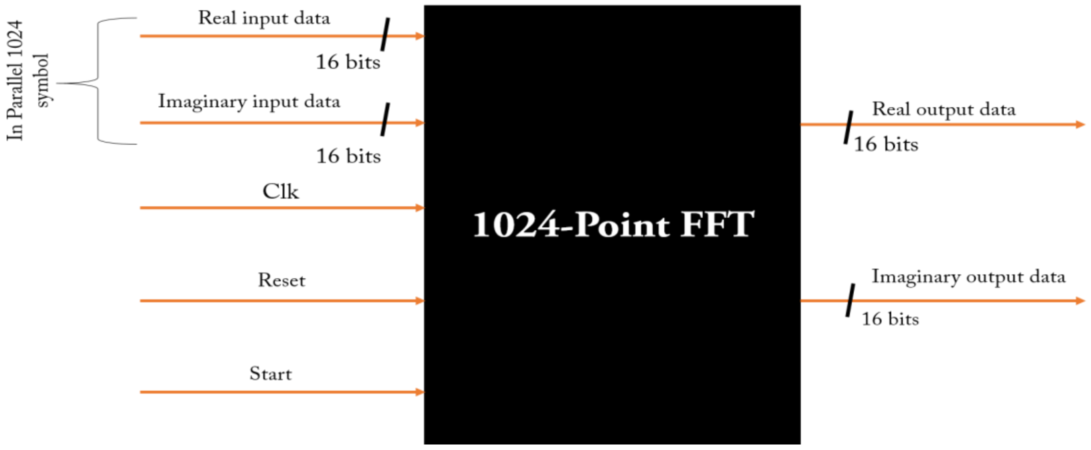


Figure 22: Block Interface of FFT

First, 1024 symbols are received from Analog to digital converter represented in 16 bits, to be loaded into a two multiport RAMs one for real inputs and one for imaginary inputs this RAM has an address width of 10 bits to choose from 1024 symbols and it's multiport because we need two inputs to compute a butterfly.

The twiddle factors are stored in ROM to be used at each stage according to the span which specifies the chosen inputs to be calculated at the butterfly that differ according to the number of sample points at FFT stage, N.

The inputs and twiddle factors enter the complex multiplier which fully operates for fixed point representation and outputs two complex numbers that enter the butterfly base to output the sampled symbols then stored in a RAM.

Block Diagram of FFT:

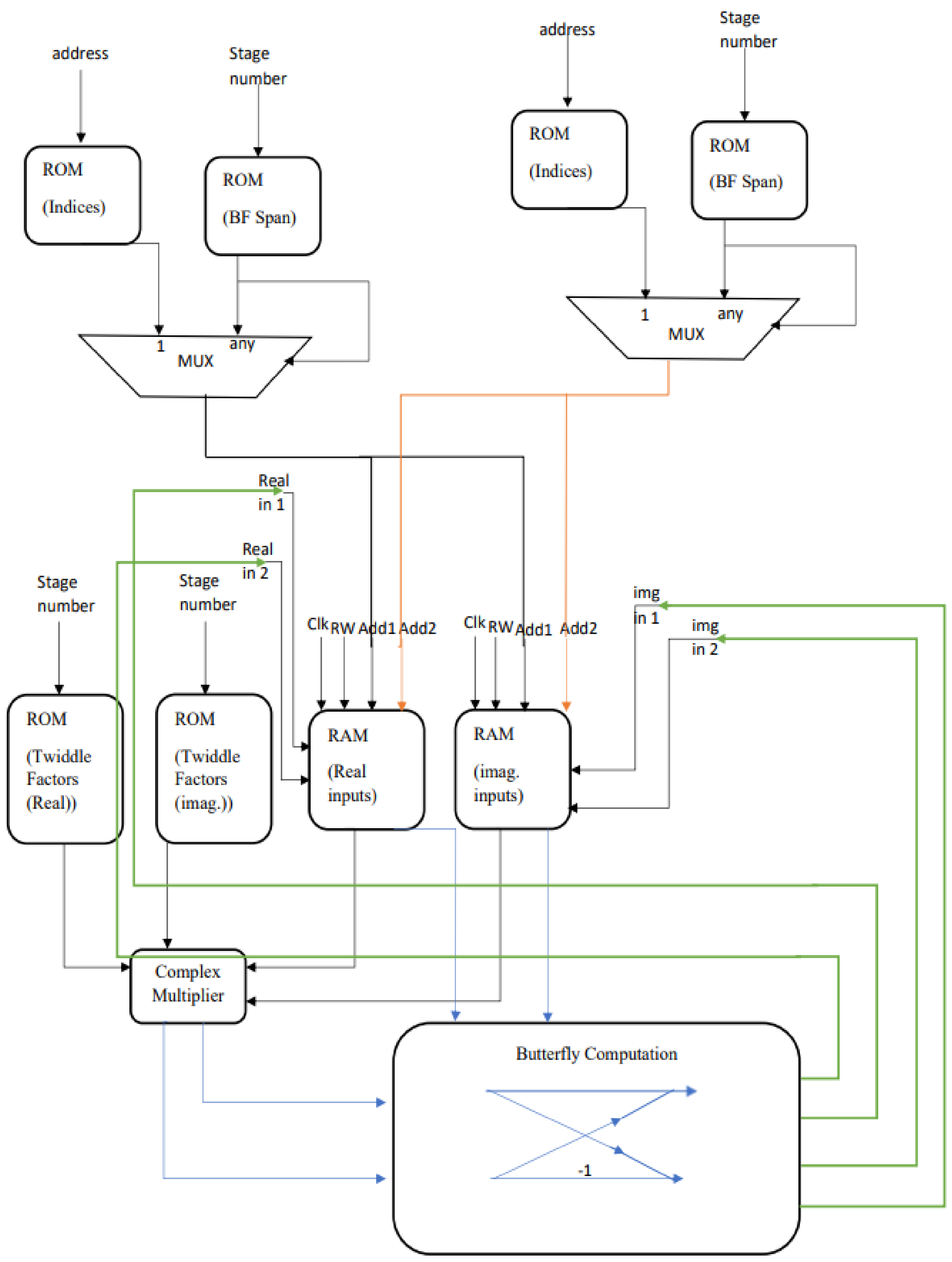
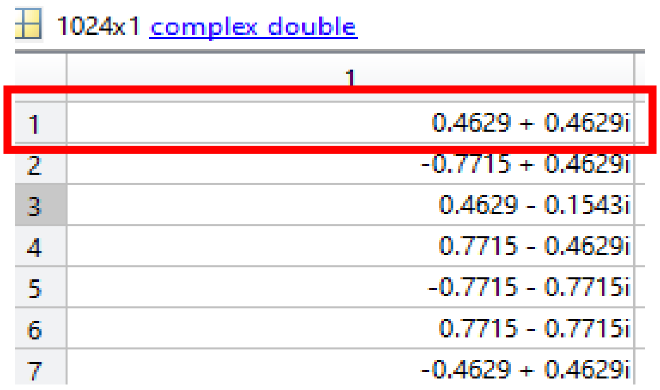


Figure 23: Block diagram of FFT

Note that only one butterfly computation is used in all stages, we preferred to decrease hardware used over parallelism concept because we implement our code on zynq-7000 FPGA which has a limited number of input and output ports and we need them for other blocks that cannot be fully utilized like FFT.

High Level Simulation:



1024x1 complex double

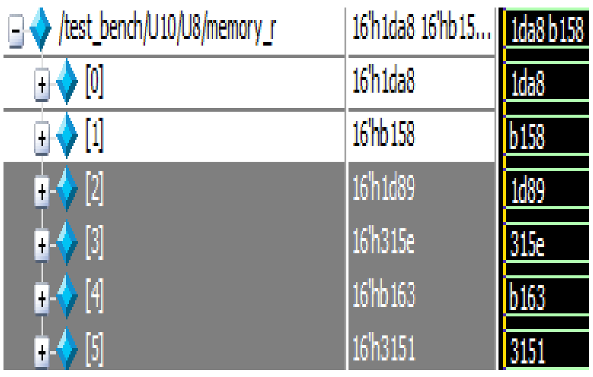
	1
1	0.4629 + 0.4629i
2	-0.7715 + 0.4629i
3	0.4629 - 0.1543i
4	0.7715 - 0.4629i
5	-0.7715 - 0.7715i
6	0.7715 - 0.7715i
7	-0.4629 + 0.4629i

Figure 24: High level results of FFT

We have simulated the MATLAB code we designed and then we compared the results with the built in functions to ensure that the code's functionality is correct.

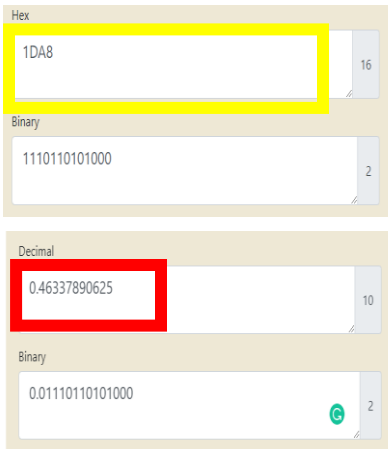
Hardware Simulation Results:

The below figures show the hardware simulation results compared to the matlab results after conversion from fixed to decimal representation for the first output as an example for emphasizing.



```
/test_bench/U10/U8/memory_r   16'h1da8 16'hb15...   1da8 b158
  [0]                         16'h1da8              1da8
  [1]                         16'hb158              b158
  [2]                         16'h1d89              1d89
  [3]                         16'h315e              315e
  [4]                         16'hb163              b163
  [5]                         16'h3151              3151
```

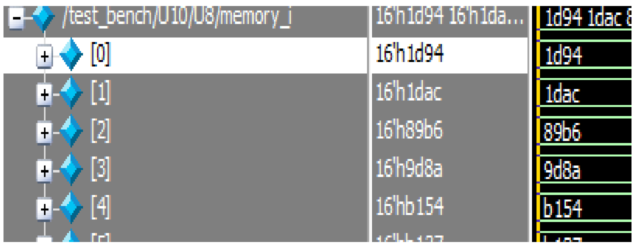
Figure 25: Hardware simulation results of real part of FFT



Hex: 1DA8 (16)
Binary: 1110110101000 (2)

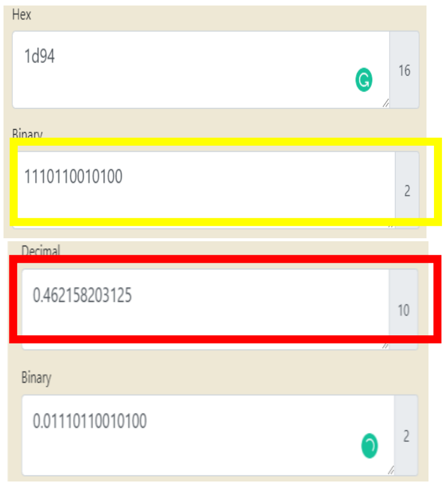
Decimal: 0.46337890625 (10)
Binary: 0.01110110101000 (2)

Figure 26: Fixed to decimal point conversion



/test_bench/U10/U8/memory_i	16'h1d94 16'h1da...	1d94 1dac
[0]	16'h1d94	1d94
[1]	16'h1dac	1dac
[2]	16'h89b6	89b6
[3]	16'h9d8a	9d8a
[4]	16'hb154	b154

Figure 27: Hardware simulation results of imaginary part of FFT



Hex
1d94 16

Binary
1110110010100 2

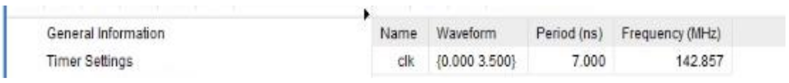
Decimal
0.462158203125 10

Binary
0.01110110010100 2

Figure 28: Fixed to decimal point conversion

Synthesis Results:

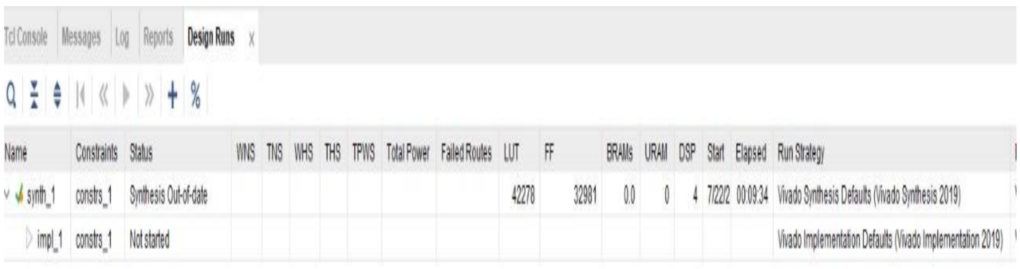
Maximum operating frequency for the FFT= 142.857 MHz which gives zero violations.



General Information
Timer Settings

Name	Waveform	Period (ns)	Frequency (MHz)
clk	{0.000 3.500}	7.000	142.857

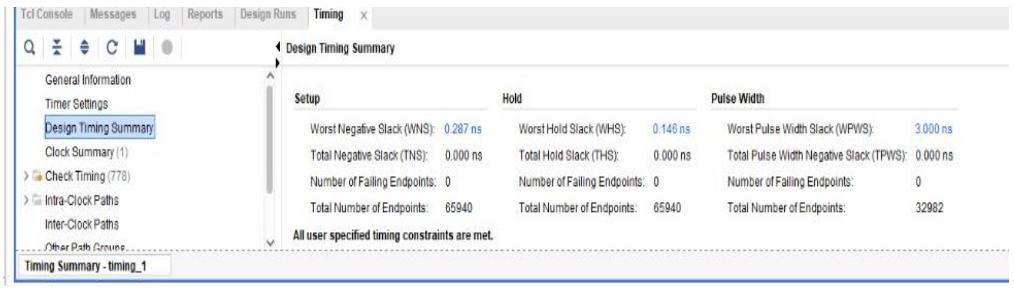
<u>Synthesis validation</u>



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Synthesis Out-of-date								42278	32981	0.0	0	4	7/22/2	00:09:34	Vivado Synthesis Defaults (Vivado Synthesis 2019)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2019)

Figure 29 Design runs of FFT

<u>Timing Report</u>



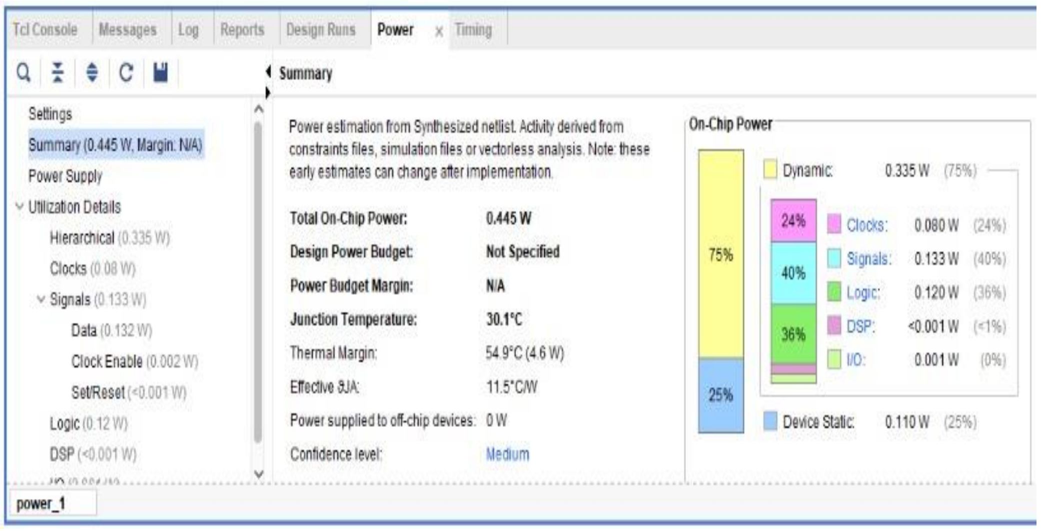
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.287 ns	Worst Hold Slack (WHS):	0.146 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	65940	Total Number of Endpoints:	65940	Total Number of Endpoints:	32982

All user specified timing constraints are met.

Figure 30: Timing report for FFT

<u>Power Report</u>



Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.445 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**30.1°C**
Thermal Margin:	54.9°C (4.6 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Dynamic:	0.335 W	(75%)
Clocks:	0.080 W	(24%)
Signals:	0.133 W	(40%)
Logic:	0.120 W	(36%)
DSP:	<0.001 W	(<1%)
I/O:	0.001 W	(0%)
Device Static:	0.110 W	(25%)

Figure 31: Power report for FFT

Analysis of synthesis results:

It’s obvious that our design is synthesizable having 42278 LUTs, 32981 FFs and only 4 DSPs because we have used only one butterfly, FFT met the time constraints with a large margin given that the clock period used was 7 ns (in the constraint file). Our design consumes chip power= 0.445 watt, the design uses a large memory due to the twiddle factors we load and the outputs that are saved in order to be carried to the next stage.

Resource De-Mapper:

System logic:

NR supports multiple numerologies because of which, there are a multiple resource grids i.e., there is one resource grid for each numerology and carrier. Different numerologies are discussed in the post: 5G NR: Numerologies and Frame Structure.

A resource grid is characterized by one sub frame in time domain and full carrier bandwidth in the frequency domain and There are a set of resource grids per transmission direction (uplink or downlink).

A Resource Element (RE) is the smallest physical resource in NR, and it consists of one subcarrier during one OFDM symbol. It is uniquely identified by (k, l) where k is the index in the frequency domain and l refers to the symbol position in the time domain relative to some reference point.

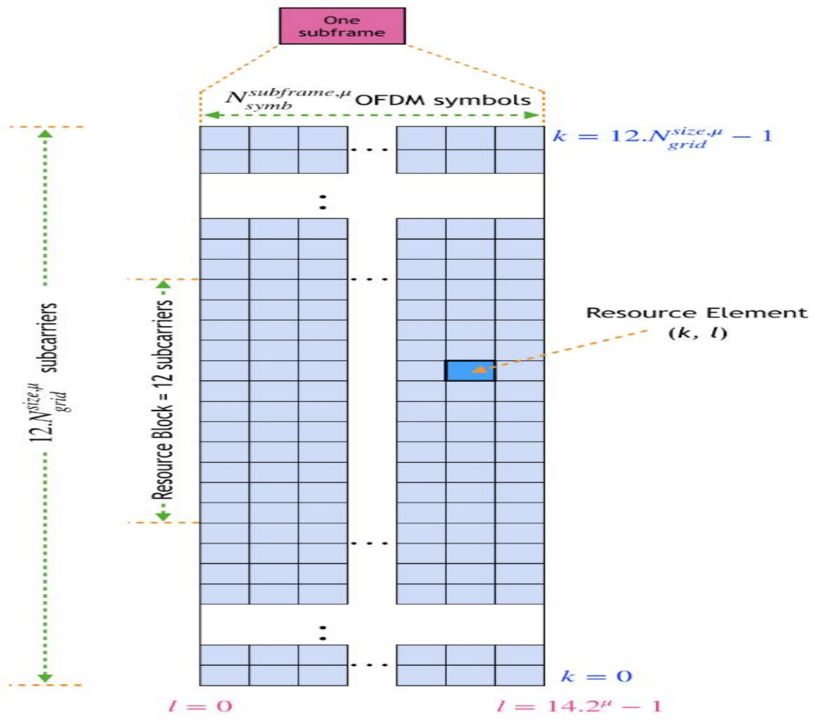

One subframe
$N_{symb}^{subframe,\mu}$ OFDM symbols
$k = 12.N_{grid}^{size,\mu} - 1$
Resource Element (k, l)
$12.N_{grid}^{size,\mu}$ subcarriers
Resource Block = 12 subcarriers
$k = 0$
$l = 0$
$l = 14.2^{\mu} - 1$

Figure 32: REM grid

In 5G NR, PDSCH is the physical downlink channel that carries user data. DM-RS and PT-RS are the reference signals associated with PDSCH. These signals are generated within the PDSCH allocation, as defined in TS 38.211 Sections 7.4.1.1 and 7.4.1.2 [1]. DM-RS is used for channel estimation as part of coherent demodulation of PDSCH. To compensate for the common phase error (CPE), 3GPP 5G NR introduced PT-RS. Phase noise produced in local oscillators introduces a significant degradation at mm-Wave frequencies. It produces CPE and inter-carrier interference (ICI). CPE leads to an identical rotation of a received symbol in each subcarrier. ICI leads to loss of orthogonality between the subcarriers. PT-RS is used mainly to estimate and minimize the effect of CPE on system performance.

PDSCH is the physical channel that carries the user data. The resources allocated for PDSCH are within the bandwidth part (BWP) of the carrier, as defined in TS 38.214 Section 5.1.2 [2]. The resources in the time domain for PDSCH transmission are scheduled by downlink control information (DCI) in the field Time domain resource assignment.

The resources in the frequency domain for PDSCH transmission are scheduled by a DCI in the field Frequency domain resource assignment. This field indicates whether the resource allocation of resource blocks (RBs) is contiguous or noncontiguous, based on the allocation type. The RBs allocated are within the BWP.

DM-RS

DM-RS is used to estimate the radio channel. The signal is present only in the RBs allocated for the PDSCH. The DM-RS structure is designed to support different deployment scenarios and use cases. A front-loaded design supports low-latency transmissions, twelve orthogonal antenna ports for MIMO transmissions, and up to four reference signal transmission instances in a slot to support high-speed scenarios. The front-loaded reference signals indicate that the signal occurs early in the transmission. The DM-RS is present in each RB allocated for PDSCH.

The symbol allocation of PDSCH indicates the OFDM symbol locations used by the PDSCH

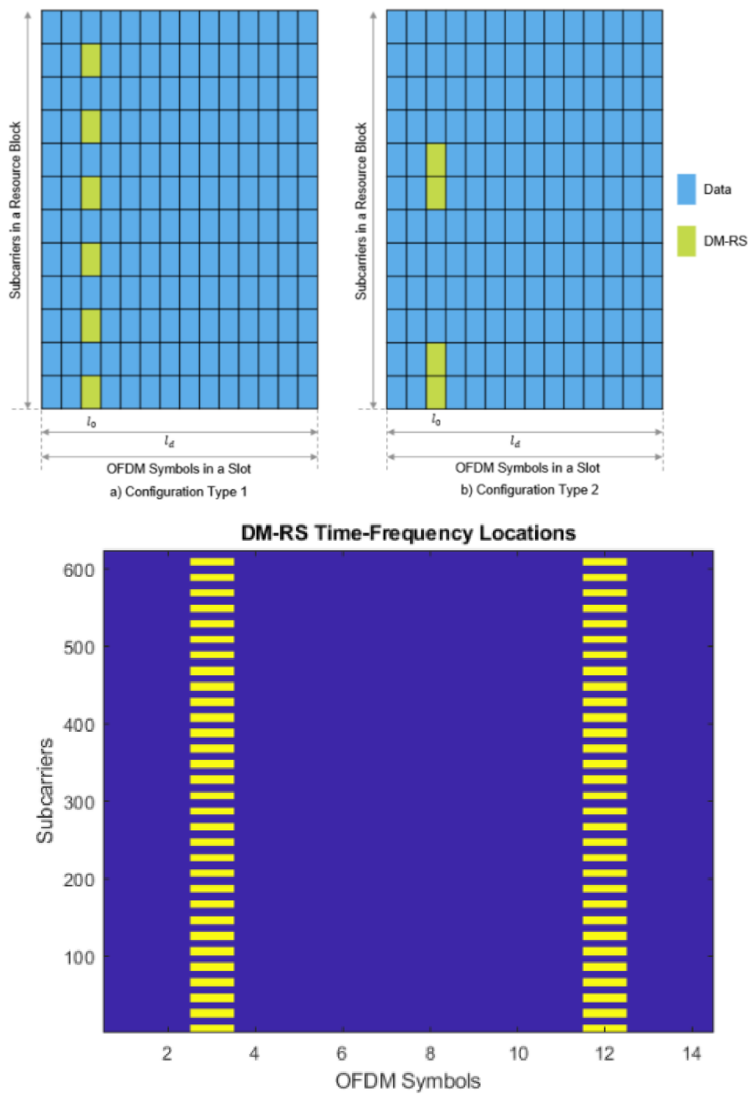


Figure 33: DM-RS time-frequency location

transmission in a slot. DM-RS symbol locations lie within the PDSCH symbol allocation. The positions of DM-RS OFDM symbols depend on the mapping type. The mapping type of PDSCH is either slot-wise (type A) or non-slot-wise (type B). The positions of any additional DM-RS symbols are defined by a set of tables, as specified in TS 38.211 Section 7.4.1.1.2 [1]. To indexing the tables.

PT-RS:

The phase noise of a transmitter increases as the frequency of operation increases. The PT-RS plays a crucial role especially at mm-Wave frequencies to minimize the effect of the oscillator phase noise on system performance. One of the main problems that phase noise introduces into an OFDM signal appears as a common phase rotation of all the sub-carriers, known as common phase error (CPE).

PT-RS stands for Phase Tracking Reference signal, its main function is to track phase of the Local Oscillator at transmitter and receiver.

PT-RS enables suppression of phase noise and common phase error especially at higher mm-wave frequencies.

It is present both in uplink (in NR-PUSCH) and downlink (in NR-PDSCH) channels. Due to phase noise properties, PTRS has low density in frequency domain and high density in time domain.

PTRS is associated with one DMRS port during transmission, moreover it is confined to scheduled BW and duration used for NR-PDSCH/NR-PUSCH

The NR system typically maps the PT-RS information to a few subcarriers per symbol because the phase rotation affects all sub-carriers within an OFDM symbol equally but shows low correlation from symbol to symbol

PT-RS occupies only one subcarrier in an RB for one OFDM symbol.

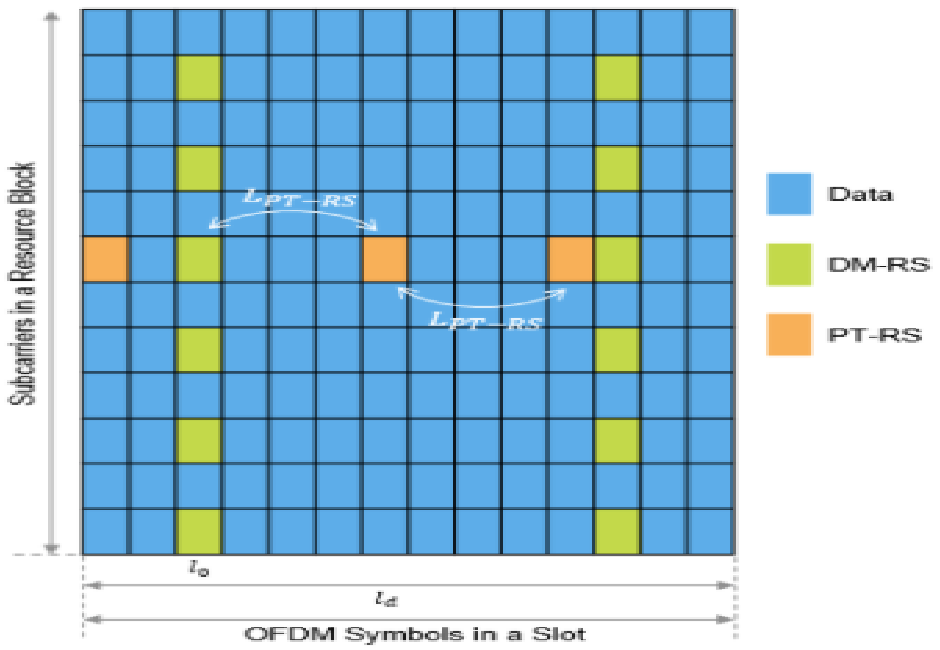
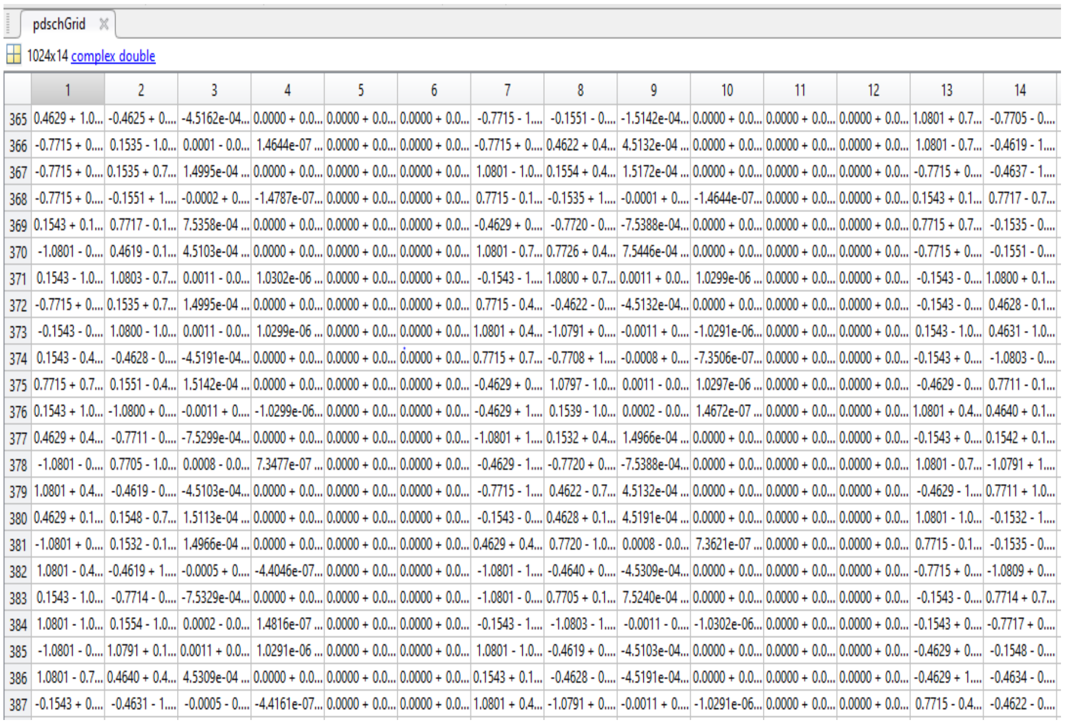


Figure 34: location of RS on the grid

Simulation results:



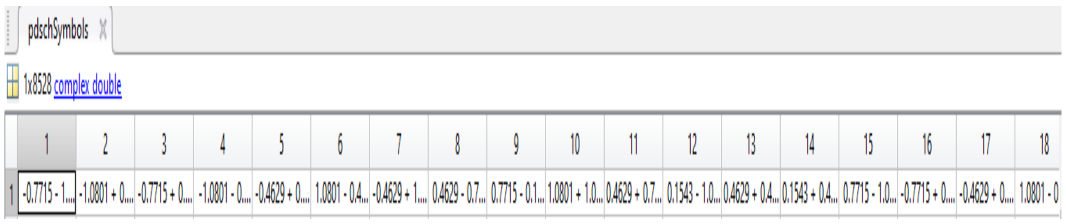
pdschGrid

1024x14 complex double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
365	0.4629 + 1.0...	-0.4625 + 0....	-4.5162e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.7715 - 1....	-0.1551 - 0....	-1.5142e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 + 0.7...	-0.7705 - 0....
366	-0.7715 + 0....	0.1535 - 1.0...	0.0001 - 0.0...	1.4644e-07 ...	0.0000 + 0.0...	0.0000 + 0.0...	-0.7715 + 0....	0.4622 + 0.4...	4.5132e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 - 0.7...	-0.4619 - 1....
367	-0.7715 + 0....	0.1535 + 0.7...	1.4995e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 - 1.0...	0.1554 + 0.4...	1.5172e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.7715 + 0....	-0.4637 - 1....
368	-0.7715 + 0....	-0.1551 + 1....	-0.0002 + 0....	-1.4787e-07...	0.0000 + 0.0...	0.0000 + 0.0...	0.7715 - 0.1...	-0.1535 + 1....	-0.0001 + 0....	-1.4644e-07...	0.0000 + 0.0...	0.0000 + 0.0...	0.1543 + 0.1...	0.7717 - 0.7...
369	0.1543 + 0.1...	0.7717 - 0.1...	7.5358e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 + 0....	-0.7720 - 0....	-7.5388e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	0.7715 + 0.7...	-0.1535 - 0....
370	-1.0801 - 0....	0.4619 - 0.1...	4.5103e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 - 0.7...	0.7726 + 0.4...	7.5446e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.7715 + 0....	-0.1551 - 0....
371	0.1543 - 1.0...	1.0803 - 0.7...	0.0011 - 0.0...	1.0302e-06 ...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 - 1....	1.0800 + 0.7...	0.0011 + 0.0...	1.0299e-06 ...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 - 0....	1.0800 + 0.1...
372	-0.7715 + 0....	0.1535 + 0.7...	1.4995e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	0.7715 - 0.4...	-0.4622 - 0....	-4.5132e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 - 0....	0.4628 - 0.1...
373	-0.1543 - 0....	1.0800 - 1.0...	0.0011 - 0.0...	1.0299e-06 ...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 + 0.4...	-1.0791 + 0....	-0.0011 + 0....	-1.0291e-06...	0.0000 + 0.0...	0.0000 + 0.0...	0.1543 - 1.0...	0.4631 - 1.0...
374	0.1543 - 0.4...	-0.4628 - 0....	-4.5191e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	0.7715 + 0.7...	-0.7708 + 1....	-0.0008 + 0....	-7.3506e-07...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 + 0....	-1.0803 - 0....
375	0.7715 + 0.7...	0.1551 - 0.4...	1.5142e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 + 0....	1.0797 - 1.0...	0.0011 - 0.0...	1.0297e-06 ...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 - 0....	0.7711 - 0.1...
376	0.1543 + 1.0...	-1.0800 + 0....	-0.0011 + 0....	-1.0299e-06...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 + 1....	0.1539 - 1.0...	0.0002 - 0.0...	1.4672e-07 ...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 + 0.4...	0.4640 + 0.1...
377	0.4629 + 0.4...	-0.7711 - 0....	-7.5299e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-1.0801 + 1....	0.1532 + 0.4...	1.4966e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 + 0....	0.1542 + 0.1...
378	-1.0801 - 0....	0.7705 - 1.0...	0.0008 - 0.0...	7.3477e-07 ...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 - 1....	-0.7720 + 0....	-7.5388e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 - 0.7...	-1.0791 + 1....
379	1.0801 + 0.4...	-0.4619 - 0....	-4.5103e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.7715 - 1....	0.4622 - 0.7...	4.5132e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 - 1....	0.7711 + 1.0...
380	0.4629 + 0.1...	0.1548 - 0.7...	1.5113e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 - 0....	0.4628 + 0.1...	4.5191e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 - 1.0...	-0.1532 - 1....
381	-1.0801 + 0....	0.1532 - 0.1...	1.4966e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	0.4629 + 0.4...	0.7720 - 1.0...	0.0008 - 0.0...	7.3621e-07 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.7715 - 0.1...	-0.1535 - 0....
382	1.0801 - 0.4...	-0.4619 + 1....	-0.0005 + 0....	-4.4046e-07...	0.0000 + 0.0...	0.0000 + 0.0...	-1.0801 - 1....	-0.4640 + 0....	-4.5309e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.7715 + 0....	-1.0809 + 0....
383	0.1543 - 1.0...	-0.7714 - 0....	-7.5329e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-1.0801 - 0....	0.7705 + 0.1...	7.5240e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 - 0....	0.7714 + 0.7...
384	1.0801 - 1.0...	0.1554 - 1.0...	0.0002 - 0.0...	1.4816e-07 ...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 - 1....	-1.0803 - 1....	-0.0011 - 0....	-1.0302e-06...	0.0000 + 0.0...	0.0000 + 0.0...	-0.1543 + 0....	-0.7717 + 0....
385	-1.0801 - 0....	1.0791 + 0.1...	0.0011 + 0.0...	1.0291e-06 ...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 - 1.0...	-0.4619 + 0....	-4.5103e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 + 0....	-0.1548 - 0....
386	1.0801 - 0.7...	0.4640 + 0.4...	4.5309e-04 ...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	0.1543 + 0.1...	-0.4628 - 0....	-4.5191e-04...	0.0000 + 0.0...	0.0000 + 0.0...	0.0000 + 0.0...	-0.4629 + 1....	-0.4634 - 0....
387	-0.1543 + 0....	-0.4631 - 1....	-0.0005 - 0....	-4.4161e-07...	0.0000 + 0.0...	0.0000 + 0.0...	1.0801 + 0.4...	-1.0791 + 0....	-0.0011 + 0....	-1.0291e-06...	0.0000 + 0.0...	0.0000 + 0.0...	0.7715 - 0.4...	-0.4622 - 0....

Figure 35 Resource grid for a frame

This is the input to the resource de-mapper which represents a complete frame in time with symbols from different layers assigned to 948 subcarriers and the DREM extracts the PDSCH data from the resource grid depending on the indices generated and given to it as an input.


pdschSymbols

1x8528 complex double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-0.7715 - 1....	-1.0801 + 0....	-0.7715 + 0....	-1.0801 - 0....	-0.4629 + 0....	1.0801 - 0.4...	-0.4629 + 1....	0.4629 - 0.7...	0.7715 - 0.1...	1.0801 + 1.0...	0.4629 + 0.7...	0.1543 - 1.0...	0.4629 + 0.4...	0.1543 + 0.4...	0.7715 - 1.0...	-0.7715 + 0....	-0.4629 + 0....	1.0801 - 0

Figure 36 Extracting channel from the grid

This is the output of the resource de-mapper which is the counter part of the resource mapping. In this block data from different layers come as input and then they are de-mapped and vectorized.

Modulation De-Mapper:

Demodulation process is a way of taking back data from the carrier and return data to base band to proceed to the whole chain to detect and decode data. 5G technology support different type of modulation technique BPSK, QPSK, 16-QAM, 64-QAM, 256 QAM. QAM is very preferable in extreme-speed communication systems because it enhance the frequency spectrum utilization. In this thesis we work on 64 QAM scheme.

System Logic:

Signal goes through AWGN channel, fading channel effect so, A soft demodulation algorithm is always used. For higher order modulation scheme **log-likelihood ratio** (LLR) algorithm is used. LLR algorithm is based on gray map of 64-QAM. Each cancellation point represent symbol of 6 bits which represented by b_{I1}, b_{I2}, b_{I3}, b_{Q1}, b_{Q2}, b_{Q3} and they ϵ to {0, 1}.

Phase component of 64QAM symbol is determined by b_{I1}, b_{I2}, b_{I3} while orthogonal component is determined by b_{Q1}, b_{Q2}, b_{Q3}. b_{Ii} and b_{Qi} has same decision boundary where i = {1,2,3} because of symmetry between symbol constellation I and Q Therefore, we can only analyze bI,1, bI,2, bI,3 when detect mapping bit logarithmic likelihood ratio calculation.

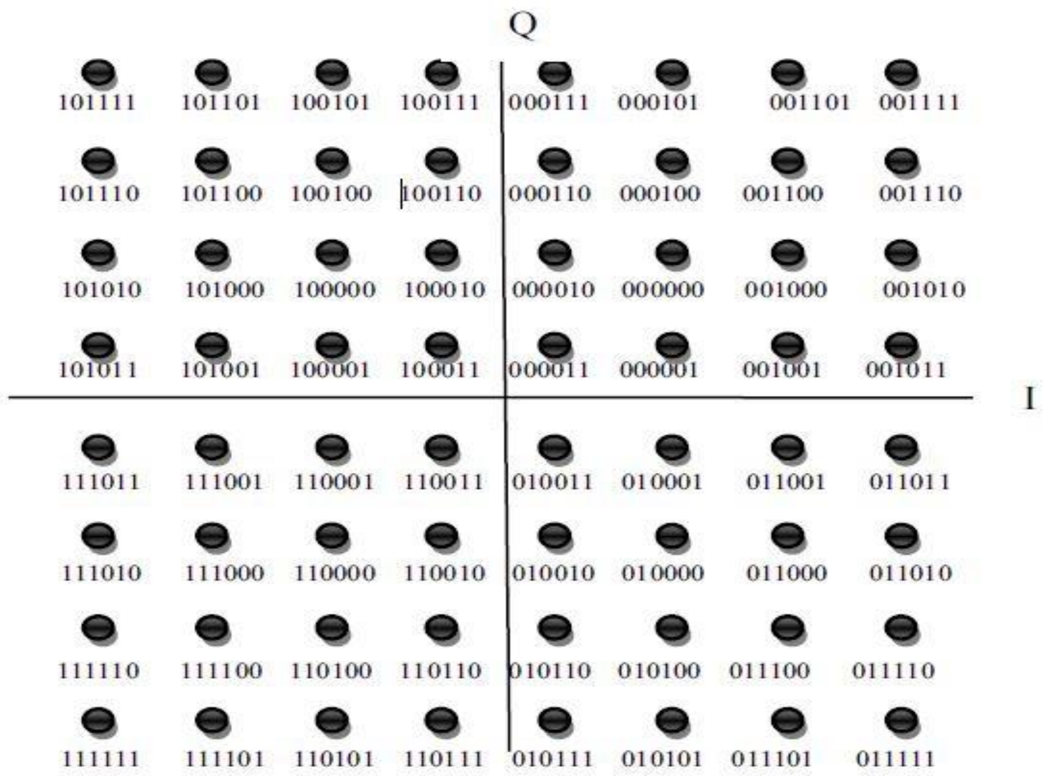


Figure 37: gray map of 64-QAM

The transmitted signal of 64-QAM denoted by x (i) where i = {1, 2…64} then the output of AWGN channel can be denoted by

$$Y(i) = H * X(i) + N$$

H is Gain factor in AWGN channel, it could consider as Constant, but in MIMO system, it is Variable. In this thesis we work on single input single output so we can consider it constant, only the variance of N could be changed.

$$Y(i) = X(i) + N$$

Where:

Y(i) is received signal.

$X(i)= X_I(i)+ j X_Q(i)$.

$N(i) = N_I(i) + j N_Q(i)$,

Where N is a White Gaussian noise with mean 0 and variance σ^2.

LLR is based on maximum probability, mapping equation of bit vector is defined as:

$$LLR\ (b_{I,m}) = \ln \frac{P\left(b_{I,m}\ = \frac{1}{r(i)}\right)}{P\left(b_{I,m} = \frac{0}{r(i)}\right)}$$

$$= \ln\ \frac{P\left(\left(b_{I,m} = 1\right) r(i)\right) P\left(r(i)\right)}{P\left(\left(b_{I,m} = 0\right) r(i)\right) P\left(r(i)\right)}$$

$$= \ln \frac{P\left(\left(b_{I,m} = 1\right) r(i)\right)}{P\left(\left(b_{I,m} = 0\right) r(i)\right)}$$

$$= \ln \frac{P\left(r(i)/b_{I,m} = 1\right) P\left(b_{I,m} = 1\right)}{P\left(r(i)/b_{I,m} = 0\right) P\left(b_{I,m} = 0\right)}$$

$$= \ln \frac{P\left(r(i)/b_{I,m} = 1\right)}{P\left(r(i)/b_{I,m} = 0\right)}$$

Assume $b_{I,m}$= 0 and 1 have the equal probability, according to the sent signal $b_{I,m}$values. The constellation map is divided into two sets, $b_{I,m}$= 1 the collection of S_1, $b_{I,m}$= 0 the collection of S_0 , then the LLR is

$$= \ln \frac{P \sum_{S_1} e^{\frac{[y_{I(i)} - x_{I(i)}]^2 + [y_{Q(i)} - x_{Q(i)}]^2}{2\sigma^2}}}{P \sum_{S_0} e^{\frac{[y_{I(i)} - x_{I(i)}]^2 + [y_{Q(i)} - x_{Q(i)}]^2}{2\sigma^2}}}$$

The Euclidean distance of LLR:

Our aim is to find nearest point on the line I parallel to x-axis in the above graph

$$LLR(b_{I,m}) = \frac{1}{2\sigma^2}\{\min_{S_1}[y_{I(i)} - x_{I(i)}]^2 + [y_{Q(i)} - x_{Q(i)}]^2 - \min_{S_0}[y_{I(i)} - x_{I(i)}]^2 + [y_{Q(i)} - x_{Q(i)}]^2\}$$

$$LLR(b_{I,m}) = \frac{1}{2\sigma^2}\{\min_{S_1}[x_{I(i)}^2 - 2y_{I(i)}x_{I(i)}]^2 - \min_{S_0}[x_{I(i)}^2 - 2y_{I(i)}x_{I(i)}]^2\}$$

Thresholds in LLR algorithm:

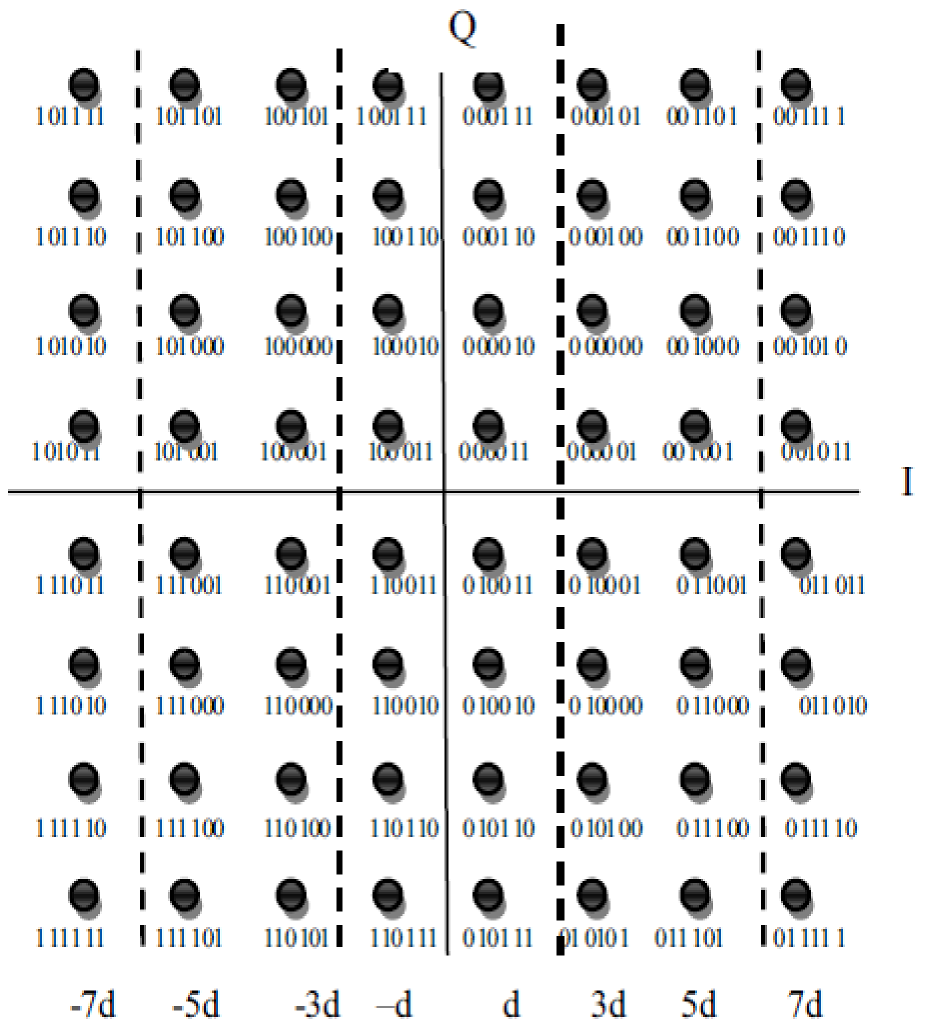


Figure 38: LLR threshold

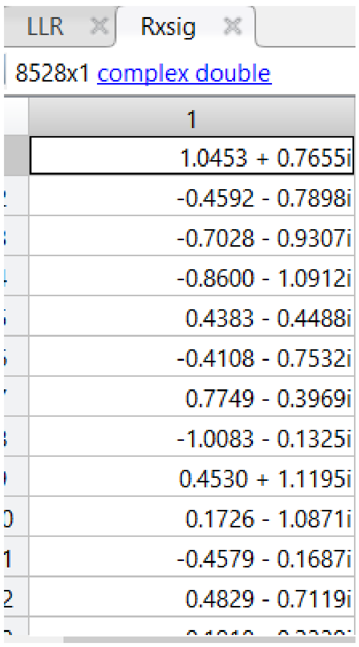
Instead of making a complex calculation to detect or map our signal to 64 constellation point another relatively simple technique is used for LLR calculations according to thresholds line each threshold line detect a single bit in our symbol so complexity of implantation is simpler, and cost is smaller as there is no a division operation here.

$$LLR\ (bI0) = xI(i)$$

$$LLR\ (bI1) = |\,xI(i)\,| - 4d$$

$$LLR\ (bI2) = |\,xI(i) - 4d\,| - 2d$$

High level simulation results

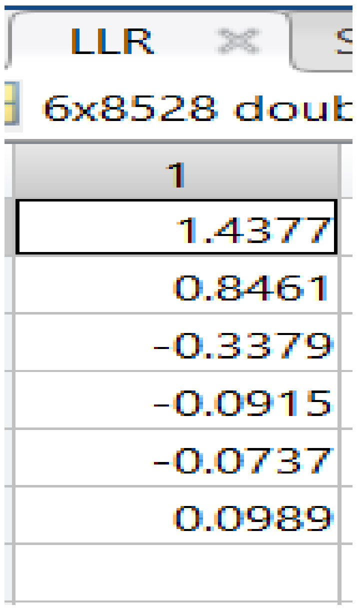


LLR | Rxsig

8528x1 complex double

1
1.0453 + 0.7655i
-0.4592 - 0.7898i
-0.7028 - 0.9307i
-0.8600 - 1.0912i
0.4383 - 0.4488i
-0.4108 - 0.7532i
0.7749 - 0.3969i
-1.0083 - 0.1325i
0.4530 + 1.1195i
0.1726 - 1.0871i
-0.4579 - 0.1687i
0.4829 - 0.7119i

Figure 40: input symbols



LLR

6x8528 doub

1
1.4377
0.8461
-0.3379
-0.0915
-0.0737
0.0989

Figure 39: 6 bits output of first symbol

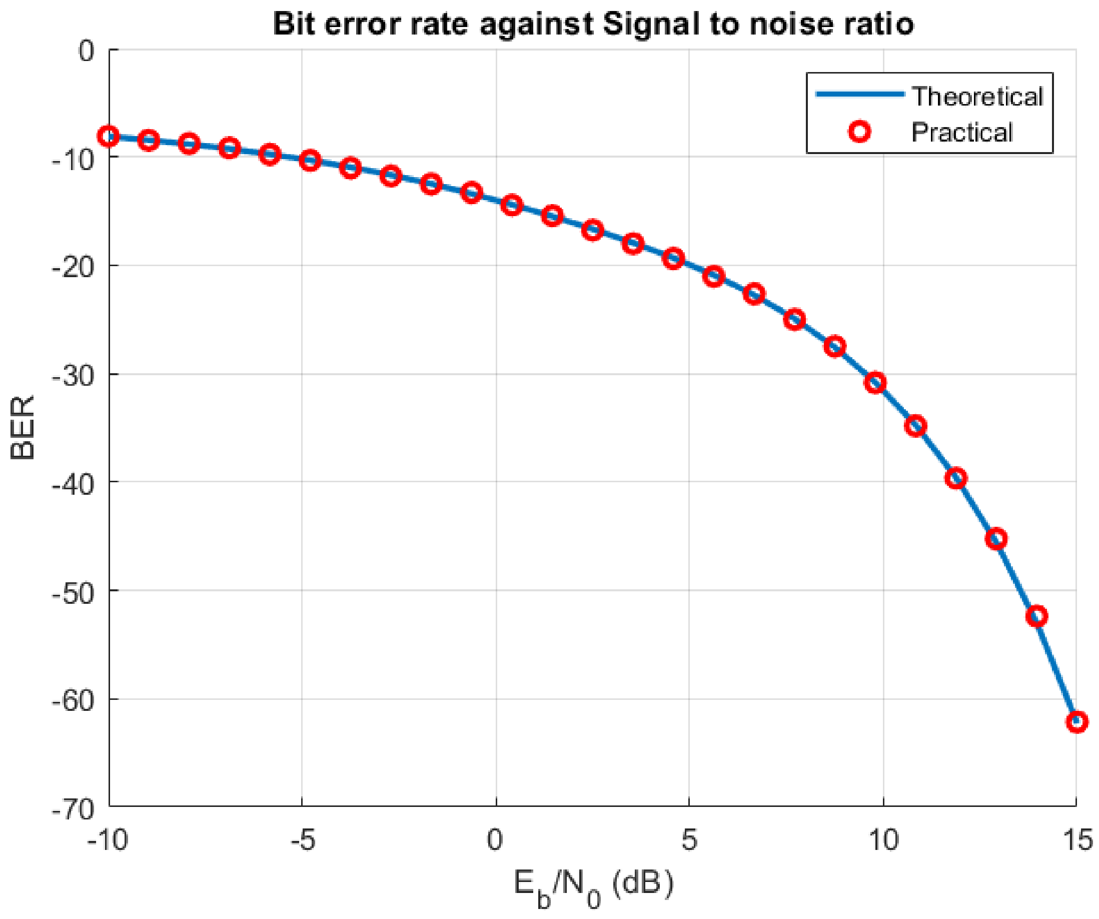


Figure 41: BER results of theoretical results Vs practical results

Hardware Implementation:

The main concept of the Demapper is to map the input symbol to one of the constellations point on the 64 QAM graph as we said above. we use log like hood ratio algorithm in deciding the suitable symbol from the constellation graph by calculating the distances between the input symbol and symbols on the constellation graph, according to the distances calculated we choose the minimum distance which will decide the output symbol by doing subtraction between minimum distances of the two thresholds. the symbol of the 64 QAM modulation has 6 bits so we do log like hood function for every bit of the six and its corresponding thresholds, using the log like hood function 6 times will give us the nearest symbol represented in 6 bits.

Block diagram of the log like hood ratio passes by 3 stages:

Stage 1: Our inputs are the input symbol and two thresholds used to decide needed bit, each threshold contains 32 symbols, calculate the Euclidean Distance between input symbol and the symbols of the 1st &2nd threshold, producing 32 distances for each threshold

Stage 2: It decide the minimum distance for each threshold so it will produce two distances that are minimum distances of the input symbol and symbols of the 1st & 2nd threshold, we decrease the complexity of comparing between 32 distances by doing comparison on 5 stages like the following algorithm but for 32 distances so we could use 16 number of reusable comparators instead of 32 comparators.

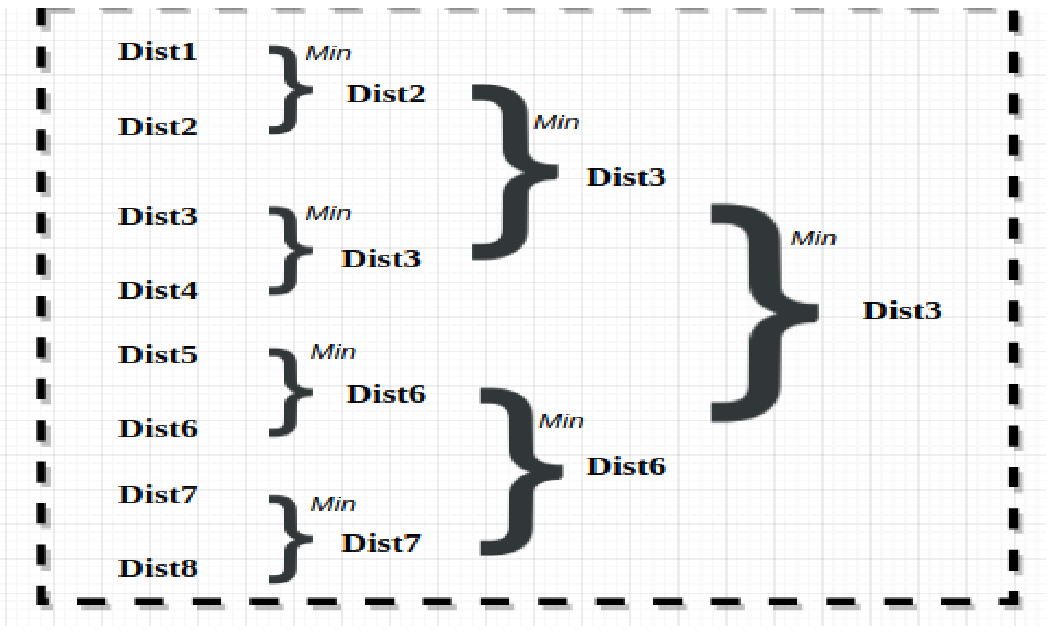


Figure 42: The algorithm used to calculate minimum distance

Stage 3: It do subtraction between the two distances which will output a certain bit then we do formatting to get the original representation which is 16-bit Fixed point.

Block diagram:

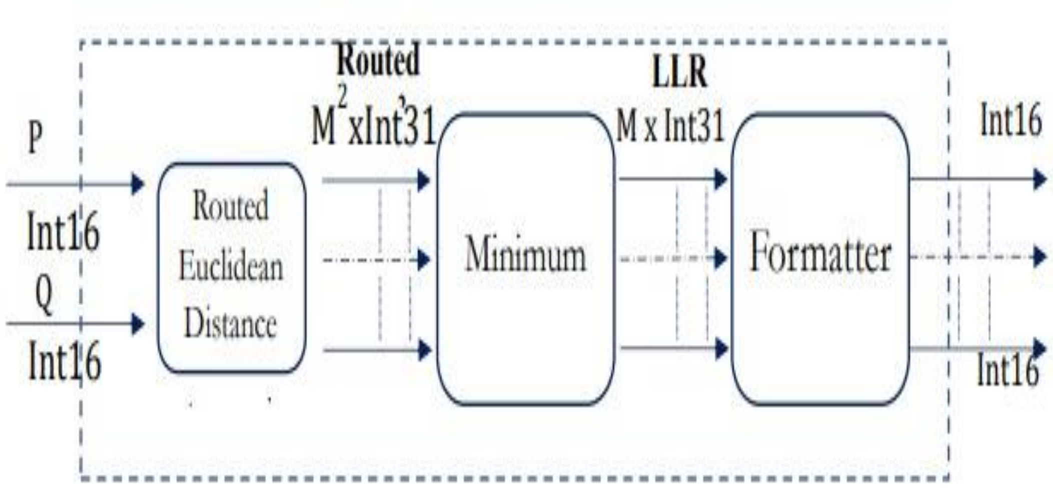


Figure 43: Block Diagram of Demapper

Block Interface:

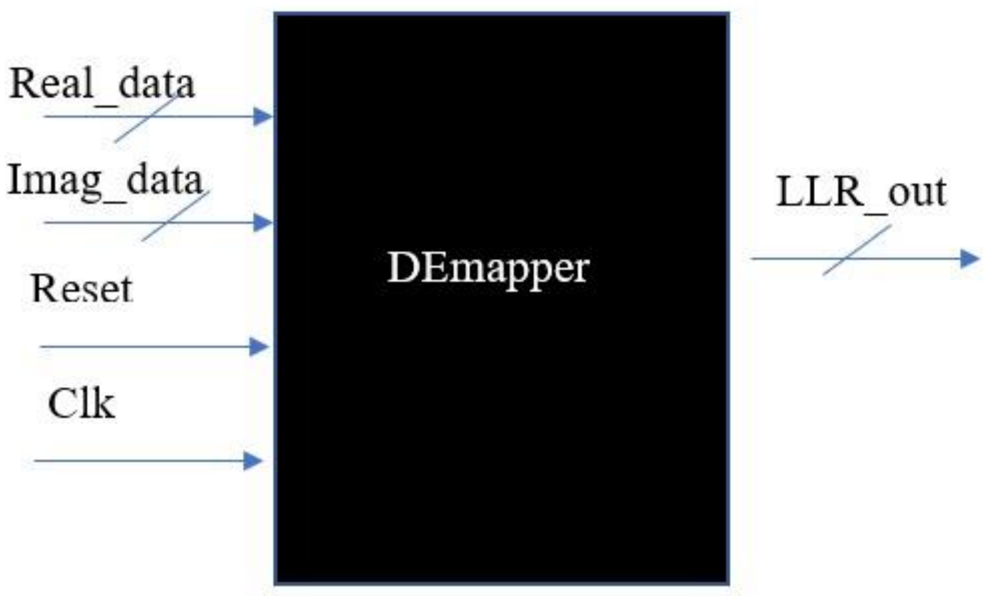
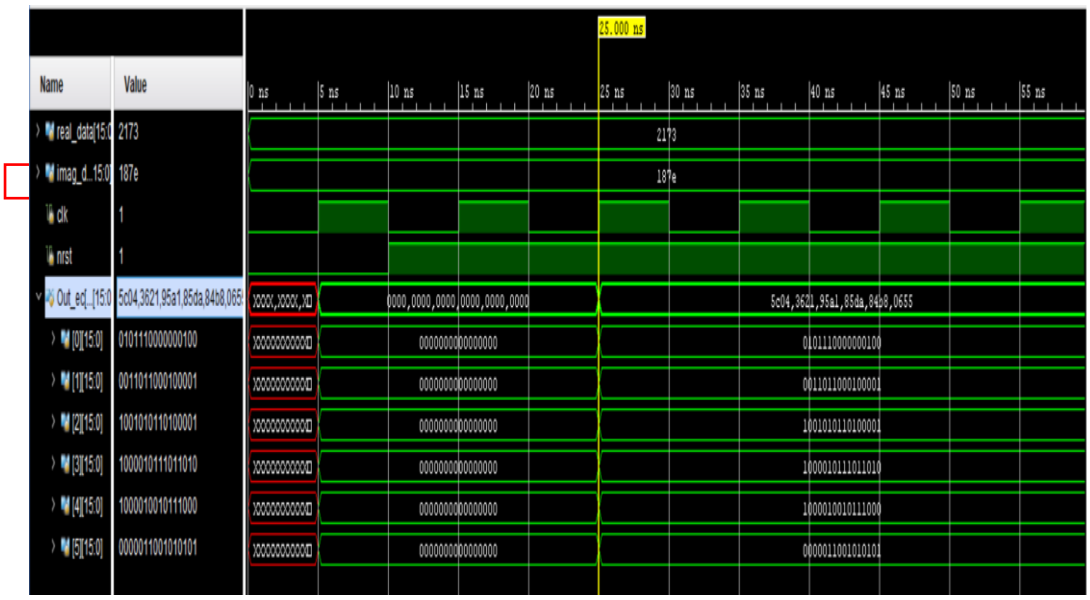


Figure 44: Demapper Block Interface

Inputs & Outputs Ports:

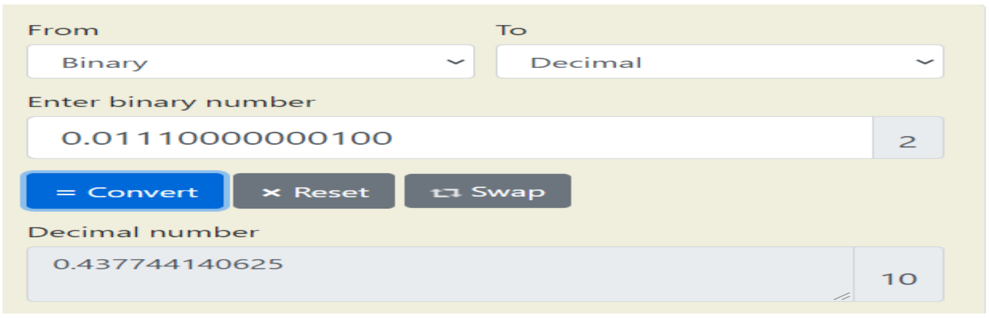
Port Name	Direction	Width	Description
Real_data	input	16 bits	The real part of the input symbol
Imag_data	input	16 bits	The imaginary part of the input symbol
Clk	input	1 bit	Operating clock to write into the block.
Reset	input	1 bit	Resetting the block.
LLR_out	output	6 element each is 16 bits	The output data where each element represent one bit for each symbol where each bit is represented by 16 bits "fixed point"

Simulation results:



Name	Value
real_data[15:0	2173
imag_d...15:0]	187e
clk	1
nrst	1
Out_eq[...[15:0	5c04,3621,95a1,85da,84b8,065
[0][15:0]	0101110000000100
[1][15:0]	0011011000100001
[2][15:0]	1001010110100001
[3][15:0]	1000010111011010
[4][15:0]	1000010010111000
[5][15:0]	0000011001010101

Figure 45: Hardware simulation result of demapper



From: Binary

To: Decimal

Enter binary number: 0.01110000000100 (2)

Decimal number: 0.4377441406250 (10)

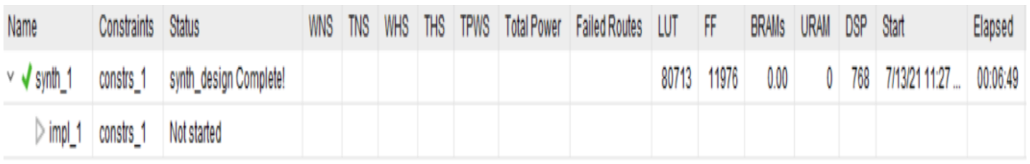
Figure 46: floating to fixed point representation

We have written a test bench for first symbol entering the MATLAB and we have got the same result that we have obtained in the MATLAB, the output appears after 3 clock cycles as every stage in the block diagram takes one cycle.

For example, 1st bit of the first symbol as it's highlighted in the previous graph, we convert the floating part from binary to decimal which will give us 0.43774, so 0101110000000100=1.43774 as MSB=0 which means it's positive value & 15th bit =1 which represent the integer part which is same first bit in MATLAB.

Synthesis analysis:

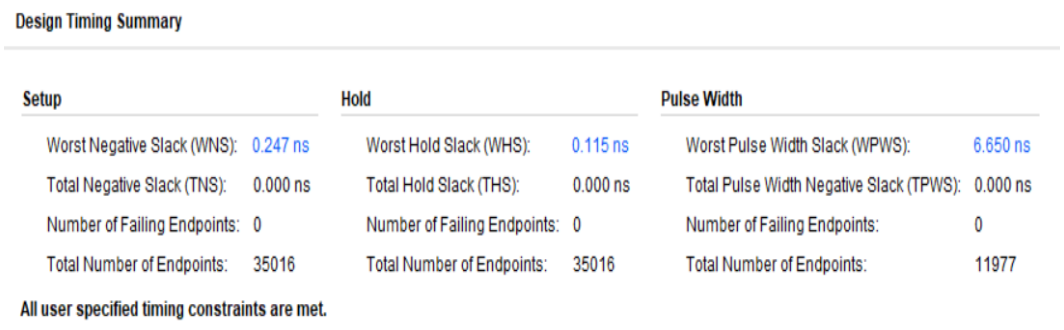
<u>Synthesis validation</u>:



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed
✓ synth_1	constrs_1	synth_design Complete!								80713	11976	0.00	0	768	7/13/21 11:27 ...	00:06:49
impl_1	constrs_1	Not started														

Figure 47: synthesis results

<u>Timing Report</u>:



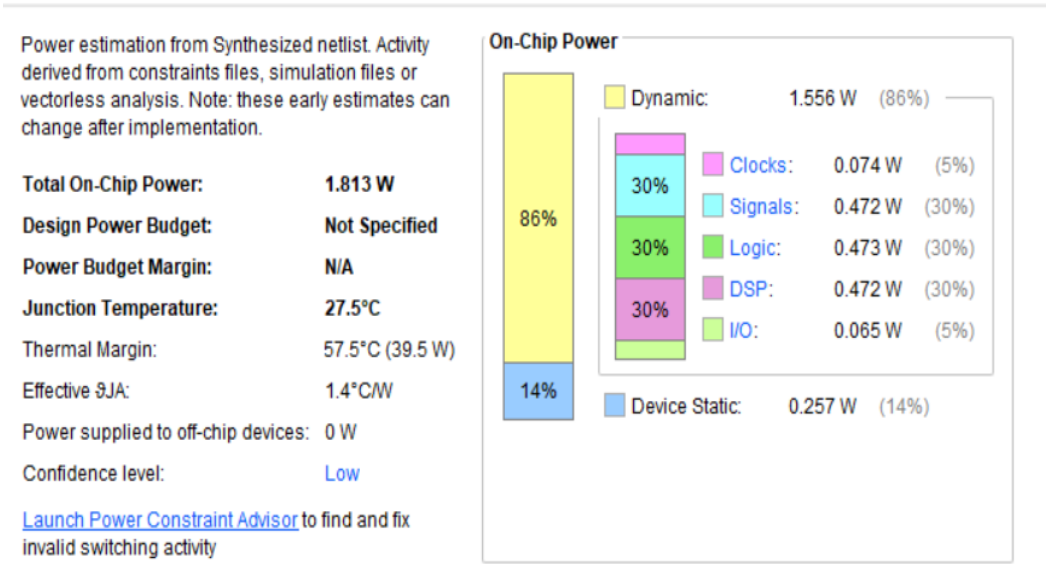
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.247 ns	Worst Hold Slack (WHS):	0.115 ns	Worst Pulse Width Slack (WPWS):	6.650 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	35016	Total Number of Endpoints:	35016	Total Number of Endpoints:	11977

All user specified timing constraints are met.

Figure 48: timing report of Demapper

<u>Power Report</u>:



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**1.813 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**27.5°C**
Thermal Margin:	57.5°C (39.5 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:		1.556 W	(86%)
	Clocks:	0.074 W	(5%)
	Signals:	0.472 W	(30%)
	Logic:	0.473 W	(30%)
	DSP:	0.472 W	(30%)
	I/O:	0.065 W	(5%)
Device Static:		0.257 W	(14%)

Figure 49: power report of demapper

Analysis of Synthesis Results:

It’s obvious that our design is synthesizable having 80713 LUT, 11976 FF and 768 DSP de-mapper design could work at frequency more than 71.428 MHZ as it met the time constraints with a large margin given that the clock period used was 14 ns (in the constraint file). Our design consumes chip power= 1.813 watt The design has large number of components as we use 64 QAM modulation so we calculate 32 distances for each threshold then we do compare getting the minimum one and this sequence is repeated 6 times to get 6 bits.

De-Scrambler:

System Logic:

There are two main reasons why scrambling is used at TX in 5G system:

- To enable accurate timing recovery on receiver equipment without resorting to redundant line coding. It facilitates the work of a timing recovery circuit (see also Clock recovery), an automatic gain control and other adaptive circuits of the receiver (eliminating long sequences consisting of '0' or '1' only).
- For energy dispersal on the carrier, reducing inter-carrier signal interference. It eliminates the dependence of a signal's power spectrum upon the actual transmitted data, making it more dispersed to meet maximum power spectral density requirements (because if the power is concentrated in a narrow frequency band, it can interfere with adjacent channels due to the intermodulation (also known as cross-modulation) caused by non-linearities of the receiving tract).

The scrambler is done by xoring a gold sequence “pseudo code” to the input data using linear feedback shift register (LFSR) to avoid the long sequence of one or zero bits.

Standards specification:

For the single codeword q=0 , the block of bits $b^{\sim(q)}(0)$,……, $b^{\sim(q)}(M_{bit}-1)$,where M_{bit} is the number of bit in the code block (q) where $b^{\sim}$ are the bits need to be descrambled so we will XOR it with the gold sequence to have original bits before being scrambled at Tx $b^{(q)}(0)$,……, $b^{(q)}(M_{bit}-1)$.

$$b\,(q)(i) = (b{\sim}(q)\,(i) + c(q)\,(i))\,mod2$$

The gold sequence used at TX will be the same of that used at RX where the scrambling sequence $c^{(q)}$ (i) is a length-31 Gold sequence, it is xoring of two linear feedback shift registers x1& x2 they have the following equation due to standards

$$c(n) = (x1(n + Nc) + x2(n + Nc))\, mod2$$

$$x1(n + 31) = (x1(n + 3) + x1(n))\, mod2$$

$$x2(n + 31) = (x2(n + 3) + x2(n + 2) + x2(n + 1) + x2(n))\, mod2$$

Due to standards we need to wait $\mathbf{N_c}$ cycles before doing XOR between X1 & X2, X1 is initialized as follow X1(0) =1, X1(1 to 30) =0 while X2 is initialized as follows X2= C_{init}

c_{init} for the PDSCH transmitter is given by $c_{init} = n_{RNTI} \cdot 215 + q \cdot 214 + n_{ID}$, where n_{ID} $\in\{0,1,...,1023\}$ is a higher-layer parameter data Scrambling Identity PDSCH and where n_{RNTI} $\in\{0,1,...,65535\}$ is the radio network temporary identifier which is used to differentiate users from each other and each DCI message is scrambled by a specific RNTI value.

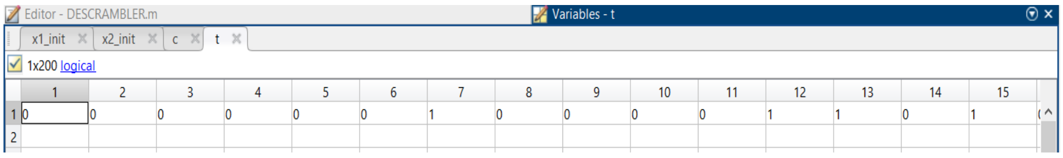
which will be equivalent to the following according to same parameters used in TX:

X2(0 to 9) = NID, X2(10 to 14) =0 , X2(15 to 30)=RNTI.

High level simulation:

We use MATLAB as a high-level language where we compare our written code with the MATLAB built-in function both will give us the same result as shown below:

We work on NID, RNTI=0 &N_c =1600 as that was used in the transmitter



Editor - DESCRAMBLER.m

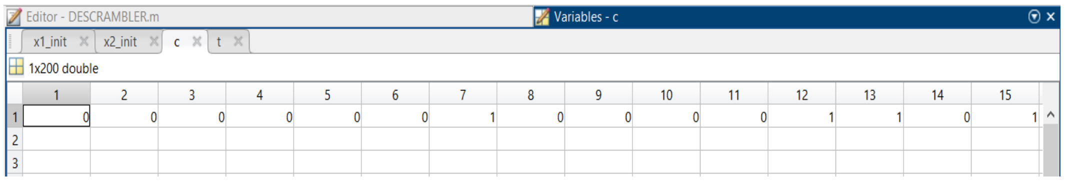
Variables - t

x1_init | x2_init | c | t

1x200 logical

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1
2															

Figure 50:built-in MATLAB function results



Editor - DESCRAMBLER.m

Variables - c

x1_init | x2_init | c | t

1x200 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1
2															
3															

Figure 51: MATLAB code result

Both results are the same so we will go to the next stage in the flow where we will convert the high-level language code into hardware description language using Verilog.

Hardware implementation:

Block interface:

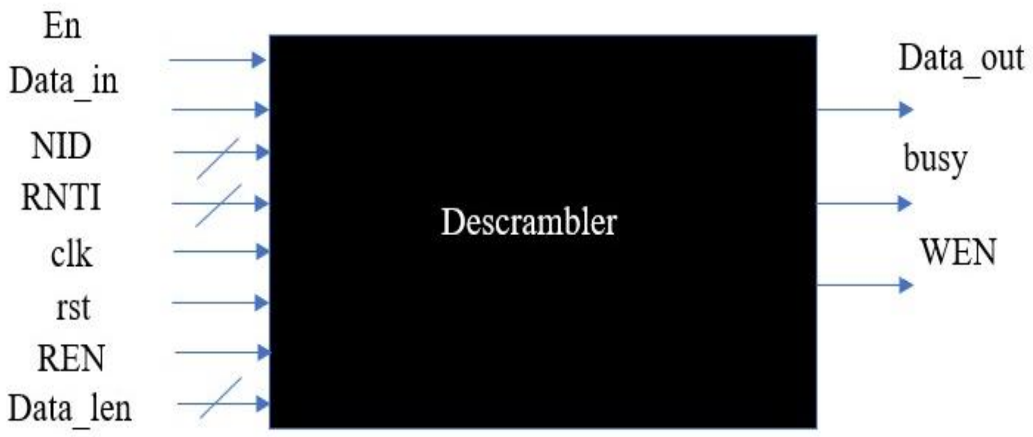


Figure 52: Descrambler Block Interface

Inputs & Outputs Ports:

Port Name	Direction	Width	Description
Enable	input	1 bit	Enable signal to enable the block.
Input Data	input	1 bit	The input data which is output from the Rate Matching block.
RNTI	input	16 bits	The radio network temporary identifier
NID	input	10 bits	A higher-layer parameter data Scrambling Identity PDSCH
Data Length	input	17 bits	Length of input data to the block.
Clock	input	1 bit	Operating clock to write into the block.
Reset	input	1 bit	Resetting the block.
Valid in	input	1 bit	Valid signal to enable writing the input data.
Valid out	output	1 bit	Valid signal which indicates the validation of output.
Busy	output	1 bit	Flag signal to indicate that the block cannot take input now
Data out	output	1 bit	The output data stream.

Block Diagram:

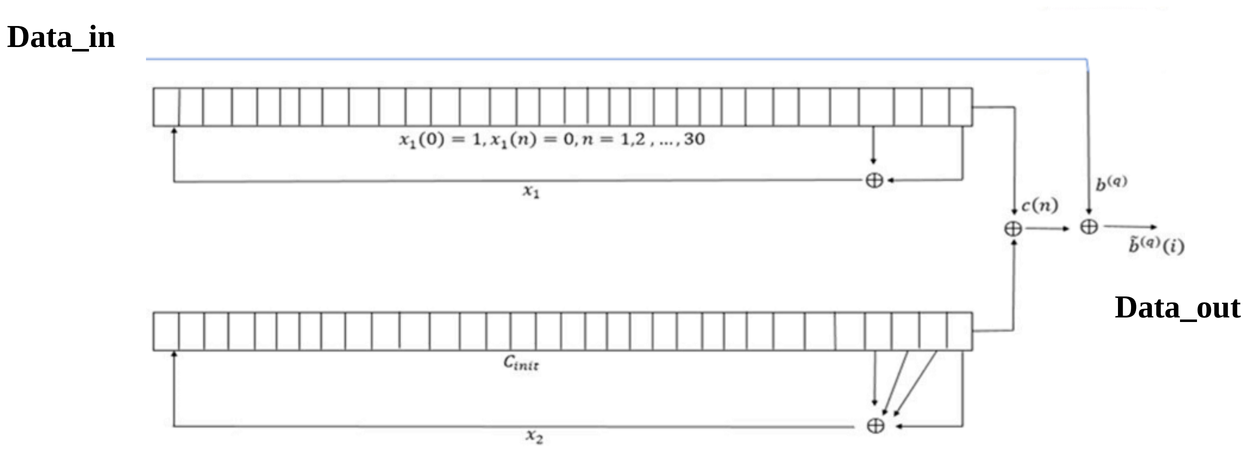


Figure 53: block diagram of DE-scrambler

Operation:

Firstly, we initialize both linear feedback shift registers X1 & X2 due to given in the standards and the given input RNTI & NID, then we will wait N_c cycles equal 1600 cycle then the done count signal will be one and we will start XORING between X1 & X2 to get the needed gold sequence which will be used in scrambling the input data.

Hardware simulation results:

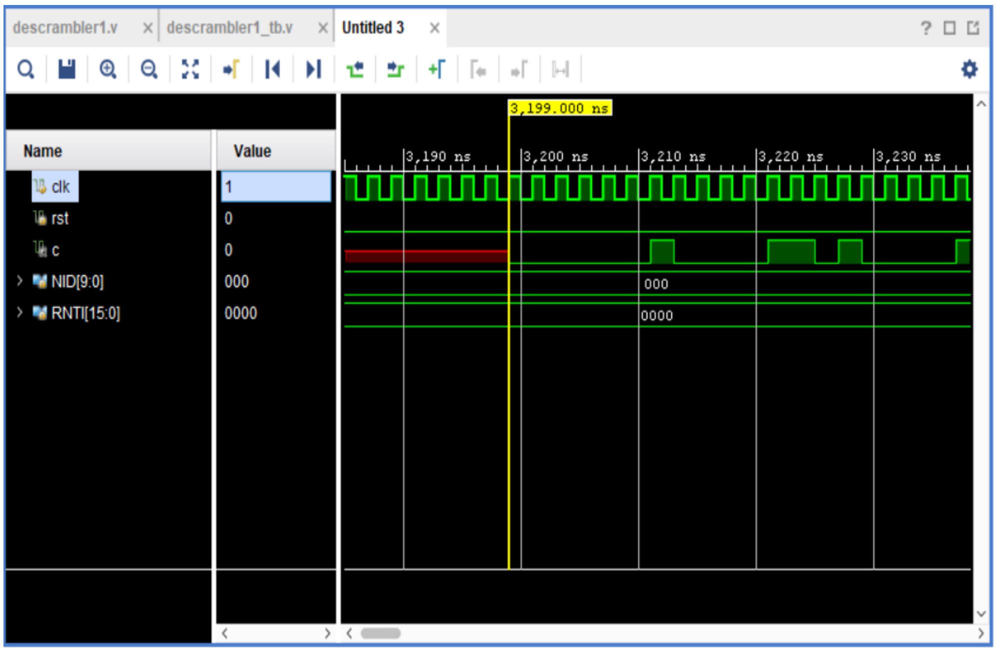
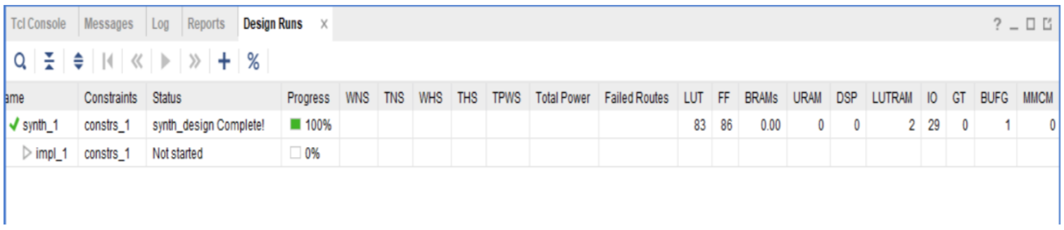


Figure 54: Hardware behavioral simulation results

- Here we use testbench to have period of clock cycle=2ns, output of the gold sequence starts at 3199 ns which means after 1599 clock cycles, so it met the standards
- It’s also clear that we have 6 zeros at the first 6clks then we have 1 for the 7th clk cycle then we have 4 zeros for the next 4 cycles then we have two ones … so the output will be as follows: C= {0,0,0,0,0,0,1,0,0,0,0,1,1,0,1, 0..}
- So, the gold sequence output using Verilog is same as that was calculated using high level language (MATLAB), so it is verified.

Synthesis results:

Synthesis validation:



Tcl Console | Messages | Log | Reports | Design Runs

ame	Constraints	Status	Progress	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	LUTRAM	IO	GT	BUFG	MMCM
✓ synth_1	constrs_1	synth_design Complete!	🟩 100%								83	86	0.00	0	0	2	29	0	1	0
▷ impl_1	constrs_1	Not started	☐ 0%																	

Figure 55: synthesis validation of DE-scrambler

Elaborate results:

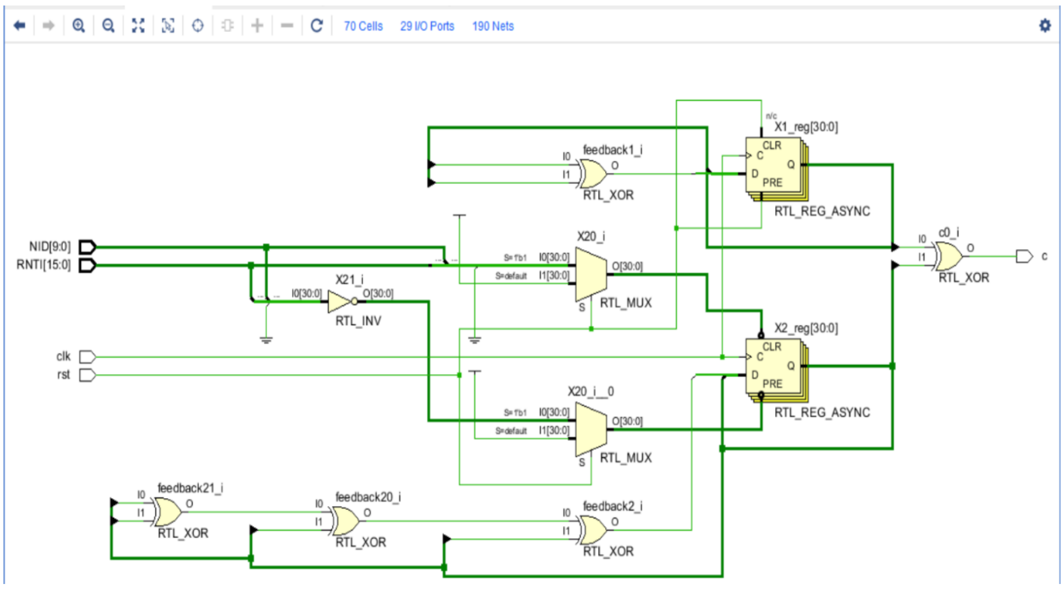
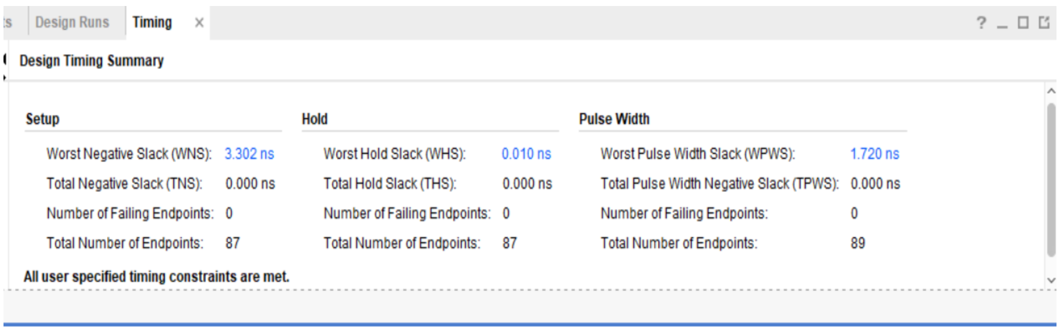


Figure 56: Elaborated design

Timing report:



s | Design Runs | Timing ×

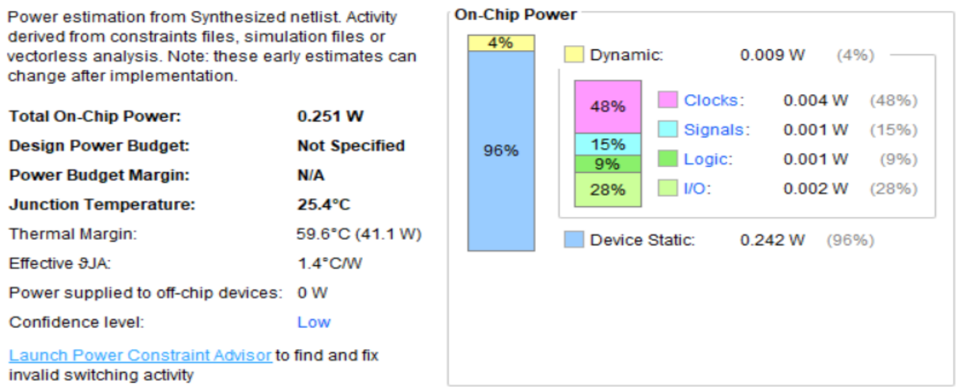
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.302 ns	Worst Hold Slack (WHS):	0.010 ns	Worst Pulse Width Slack (WPWS):	1.720 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	87	Total Number of Endpoints:	87	Total Number of Endpoints:	89

All user specified timing constraints are met.

Figure 57: timing analysis at 5ns period

Power report:



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.251 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.4°C**
Thermal Margin:	59.6°C (41.1 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.009 W	(4%)
Clocks:	0.004 W	(48%)
Signals:	0.001 W	(15%)
Logic:	0.001 W	(9%)
I/O:	0.002 W	(28%)
Device Static:	0.242 W	(96%)

Figure 58: Power on chip for descrambler block

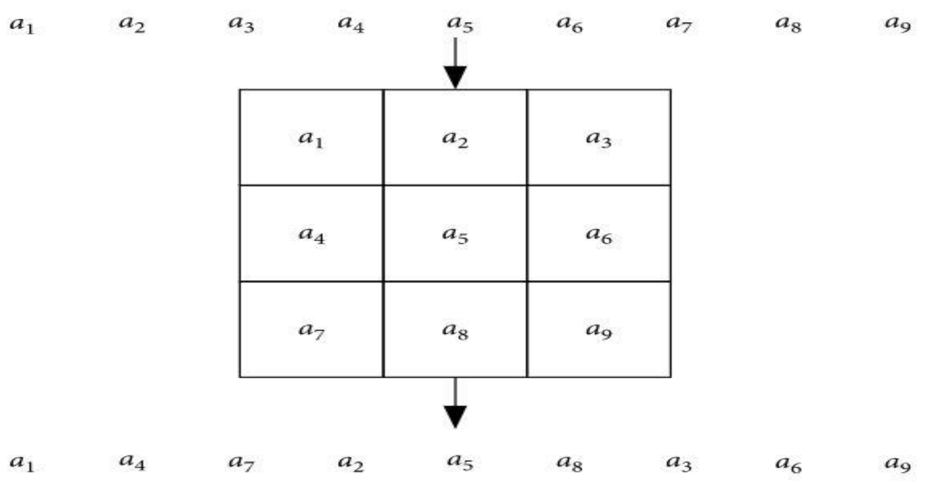
Analysis of synthesis results:

It is obvious that our design is synthesizable having 83 LUT, 86 FF, 29 IO & 2 LUTRAM. Our design could work at frequency more than 200MHZ as it met the time constraints with a large margin given that the clock period used was 5ns (in the constraint file). Power consumption by descrambler = 0.251 watt.

De-Interleaver:

System logic:

The interleaving process is a tool used to develop existing error-correction codes to be used for debugging. Interleaver is a tool that can change the structure of information distribution without changing the content of the information. It has been detected to make explosive errors generated in the channel transfer process reduced. The LDPC code integration system adopted in the 5G standard works less with block interleaver. Interleaving method to read the sequence of insertions into a matrix in rows and then read in columns, as shown in Figure [1]. The process of deinterleaving is the opposite operation, that is, read the combined sequence in the matrix in columns and then read it in rows. The matrix is determined by the length and interleaving depth of the input sequence. The number of rows in the matrix is the interleaving depth, and the number of columns is the length of the input sequence divided by the interleaving depth. The interleaving depth is related to the modulation order Their different modulation schemes defined in the 5G NR standard, like, BPSK, QPSK, 16QAM, 64QAM, and 256QAM. For example, if a 64QAM modulation scheme is used and the input sequence length is 6000 characters, then the matrix size is 6 × 1000. After adding the interleaving function, the code performance has a corresponding improvement. One of main motivations of the interleaver is to allocate information bits to better bit channel locations.



a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9
a_1 a_4 a_7 a_2 a_5 a_8 a_3 a_6 a_9

Figure 59: Interleaver Process

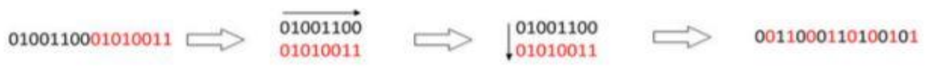


Figure 60: Data flow in interleaver block

Standards Specifications:

The input bit sequence e_0 e_1 $e_{2\ldots}e_{E-1}$ is DE-Interleaved into bit sequence f_0 f_1 f_2 .. f_{E-1} according to the modulation order, as we said above interleaver re-arrange bit streams again by inserting them in 2D matrix row by row, the number of rows is depending on the modulation scheme order. In our project we worked on 64QAM modulation scheme. E is representing the length of data from each segment. we have a limitation on the value of *E which cannot **exceed*** **8192** according to standers of 5G.

$e_0 e_1 e_2 e_3 \ldots e_{E-1}$
De-interleaver
$f_0 f_1 f_2 .. f_{E-1}$

The core of the whole process is to write the sequence of input x (n) in rows and read the sequence of savings as f (n) in the columns. Therefore, the fulfillment of the interleaver finds the correlation between f (n) and x (n), i.e. the interleaving address. Since the intervention process can be the same as that in the matrix, the parameter i in the linking process can be equal to the line parameter, j can be equal to the column parameter, and the row and column correspond to the line and column in the matrix, respectively, where the range i is [0, Qm - 1], and width be- j Bungu- [0, E / Qm - 1]. After that, set the interleaving effect as Ji, j, which is $f_{i+j.Q_m} = e_{J_{i,j}}$,

When the outer loop and its value increase from 0 to E / Qm - 1 and the inner loop with its value increases from 0 to Qm. Therefore, we can find the value of i + j × Qm is 0, 1, 2, 3, ..., E - 1. That is, with the increase of i and j, the value of Ji, j the position of objects in the sequence of output corresponding to the sequence before quitting. For example, the calculated value of Ji, j is written as [1-4] respectively. If the input sequence is e, the output sequence is [e (4), e (3), e (2), e (1)]. In the first process

However, because the formula is not suitable for subsequent hardware use, this paper uses a new

$$J_{i,j} = \frac{i \cdot E}{Q_m + j}.$$

method to achieve the result. Let's start by assuming that the value of the input sequence is 0, 1, 2, 3,…, 19. In other words, the value of an object in the input sequence is equal to its position in the input sequence, i.e., Ji, j. It is easy to see later. Qm = 4 refers to the 16QAM variable, and the rectangle behind the variable process data is shown in Figure 6. When the value of j is 0, the data in the first column are read. Ji, j corresponds to the following data, and remains 5 times higher than previous data. For example, the first thing in the first row corresponding to Ji, j is 0, the next thing corresponding to Ji, j is 5, and the next thing is equal to 10. The rule of the following columns is the same as the first column. Therefore, when the line parameter i is not equal to 0, the value of Ji, j is the value of the last data read by Ji, j (can be set as Ji - 1, j) and E / Qm. Subsequently, when i is equal to 0, Ji, j is the value of column parameter j, so Ji's formula, j can be found as

$$J_{i,j} = \begin{cases} j, & i = 0, \\ J_{i-1,j} + \dfrac{E}{Q_m}, & i \neq 0, \end{cases}$$

Where the value of $\frac{E}{Q_m}$ (e.g., the number of rectangular columns) can be given in the calculation phase.

Block interface:

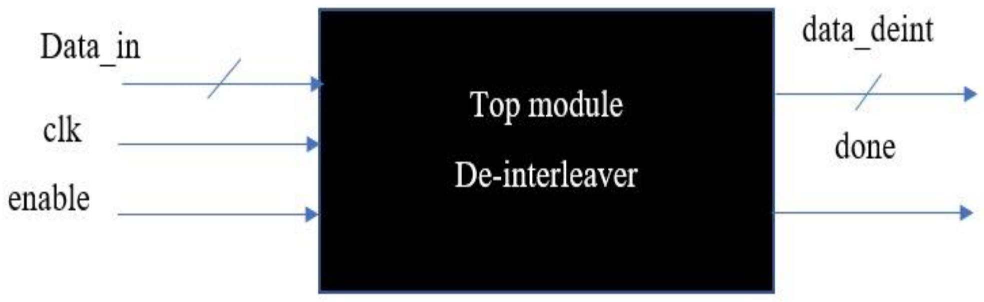


Figure 61: Deinterleaver Block Interface

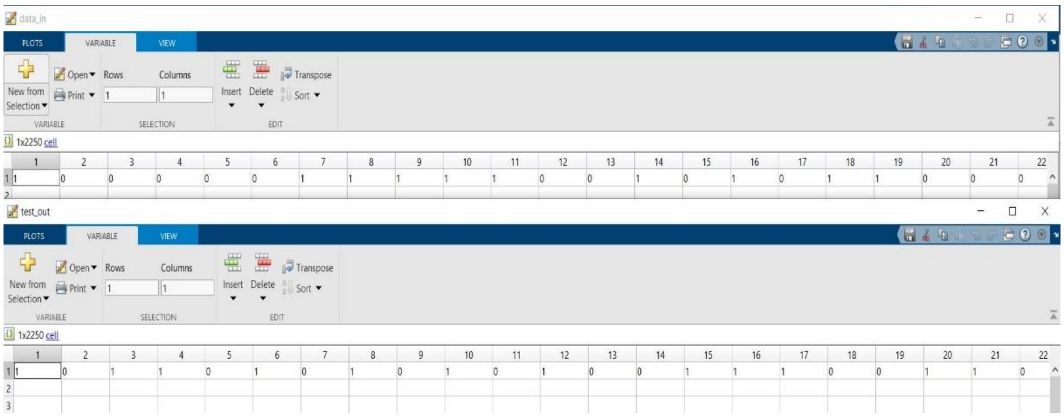
Inputs & Outputs Ports:

Name port	Direction	Width	description
Data_in	Input	[0: segment_len-1]	Input segment that needed to be interleaved
data_deint	output	[0: segment_len-1]	Output deinterleaved data
done	output	1 bit	A marker signal to the next block to indicate that the right data Is available
clk	input	1 bit	Operating clock signal
enable	input	1 bit	Enable signal to enable the block
Segment_len	input	2250	Total segment bits

Segment_len is length of input data segment bus of the De-interleaver.

We work on 64QAM modulation so our De-interleaver size is depending on modulation scheme as we declared in **SYSTEM LOGIC** section e is $6 \times column\_length$ where column_length = $\frac{Segmen\_LEN}{6}$.

High level simulation:



data_in

1x2250 cell

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	1	1	0	0	0

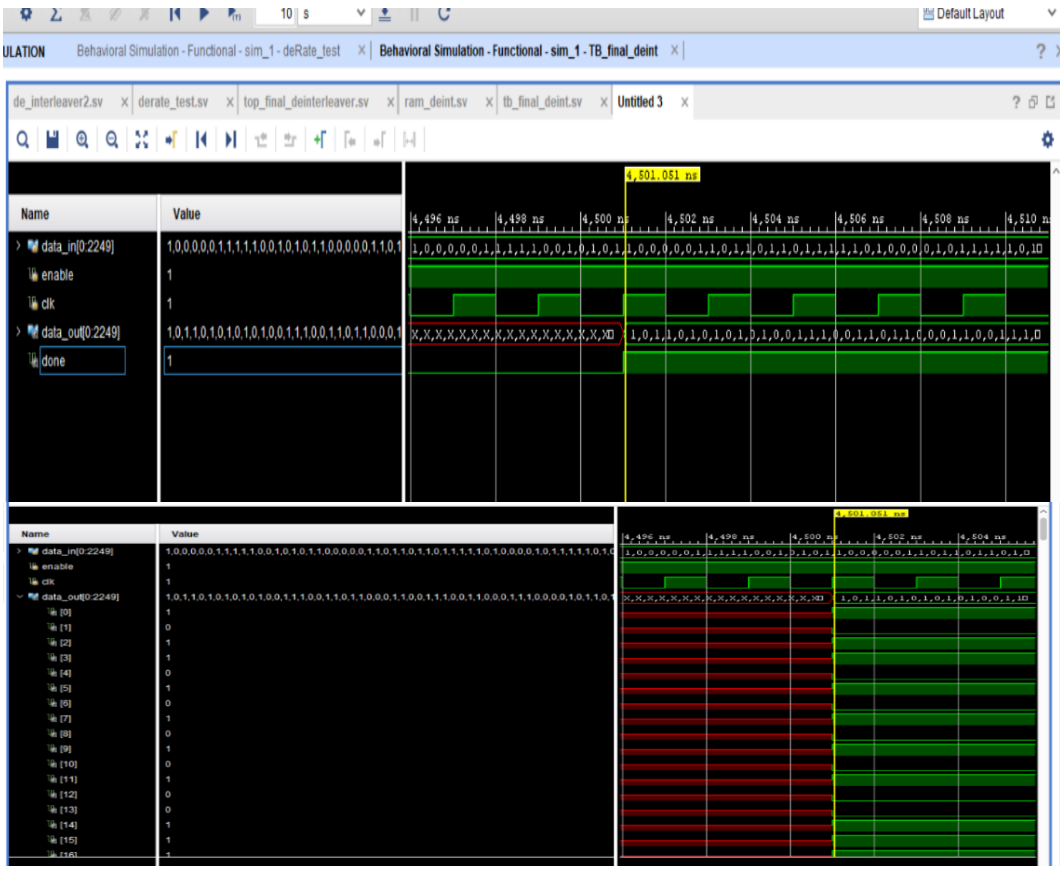
test_out

1x2250 cell

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	1	1	0	1	0	1	0	1	0	1	0	0	1	1	1	0	0	1	1	0

Figure 62: high level simulation of DE-interleaver

Hardware simulation results:



Behavioral Simulation - Functional - sim_1 - deRate_test | Behavioral Simulation - Functional - sim_1 - TB_final_deint

de_interleaver2.sv | derate_test.sv | top_final_deinterleaver.sv | ram_deint.sv | tb_final_deint.sv | Untitled 3

4,501.051 ns

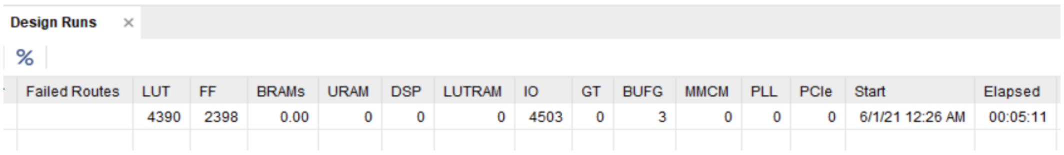
Name	Value
data_in[0:2249]	1,0,0,0,0,0,1,1,1,1,1,0,0,1,0,1,0,1,1,0,0,0,0,0,1,1,0,1
enable	1
clk	1
data_out[0:2249]	1,0,1,1,0,1,0,1,0,1,0,1,0,0,1,1,1,0,0,1,1,0,1,1,0,0,0,1
done	1

Name	Value
[0]	1
[1]	0
[2]	1
[3]	1
[4]	0
[5]	1
[6]	0
[7]	1
[8]	0
[9]	1
[10]	0
[11]	1
[12]	0
[13]	0
[14]	1
[15]	1

Figure 63: de-interleaver simulations

Synthesis Results:

Synthesis validation:

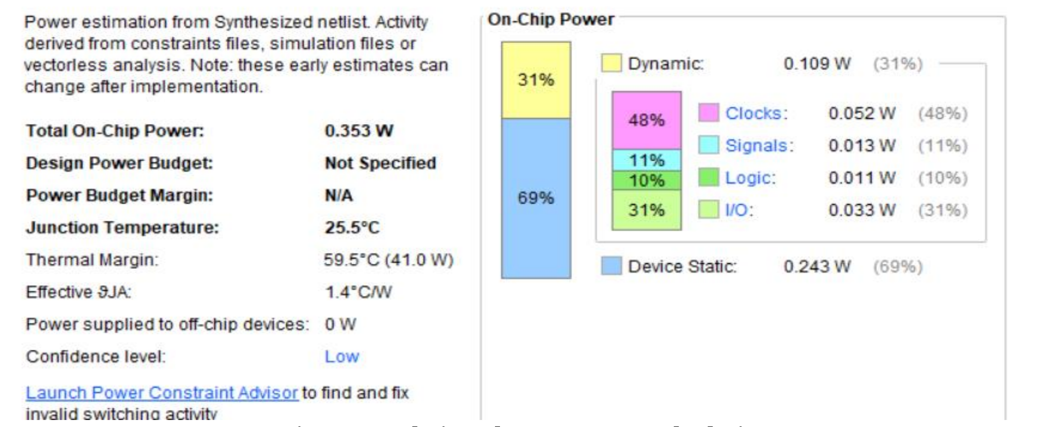


Design Runs

Failed Routes	LUT	FF	BRAMs	URAM	DSP	LUTRAM	IO	GT	BUFG	MMCM	PLL	PCIe	Start	Elapsed
	4390	2398	0.00	0	0	0	4503	0	3	0	0	0	6/1/21 12:26 AM	00:05:11

Figure 64 : de- interleaver synthesis validation

Power report:



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.353 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (41.0 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

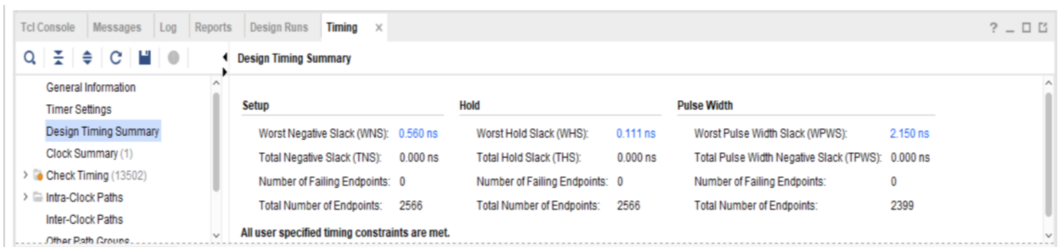
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	0.109 W	(31%)
🟪	Clocks:	0.052 W	(48%)
🟦	Signals:	0.013 W	(11%)
🟩	Logic:	0.011 W	(10%)
🟩	I/O:	0.033 W	(31%)
🟦	Device Static:	0.243 W	(69%)

Figure 65: de-interleaver power calculations

Timing report:



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.560 ns	Worst Hold Slack (WHS):	0.111 ns	Worst Pulse Width Slack (WPWS):	2.150 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	2566	Total Number of Endpoints:	2566	Total Number of Endpoints:	2399

All user specified timing constraints are met.

Figure 66: timing reports for de-interleaver

Analysis of synthesis results:

Our block meets constrains and work on 200 MHZ frequency with acceptable margin. Our de-Interleaver block consume 0.353 watt as represented in power report. The numbers of input-output ports for the block which is later used as intermediate wires. We consume large number of LUT here according to huge data input to De-Interleaver.

De-Rate Matching:

System Logic:

The main purpose of rate matching at transmitter is to match the rate of incoming bits to available resources, the rate can be increased or decreased by this block by repeat the data or pruned part of them, So the de-rate matching process is done for releasing the rate matched state of the received data prior to decoding of the received data repeated or pruned at the receiver of the mobile communication system.

Standards Specifications:

We use code rate equals to 1/5 to increase throughput as we assume that the channel is noisy, the goal of De-rate matching to return pruned data from transmitter to make it easy for low-density parity check block to decode the signal.

De-Rate matching sends hybrid automatic repeat request (HARQ) signal to the rate matching in transmitter to return the data that pruned to match the rate of available resources.

Denoting by E the rate matching output sequence length, the bit selection output bit sequence e_{k}, $_{k=0,1,2,\ldots,E-1}$, is generated as follows:

```
if $E \geq N$     -- repetition
for $k = 0$ to $E$-1
$e_k = y_{\mathrm{mod}(k,N)}$ ;
end for
else
if $K / E \leq 7/16$          -- puncturing
for $k = 0$ to $E$-1
$e_k=y_{k+N-e}$;
end for
else          -- shortening
```

```
for $k$ = 0 to $E$-1
$e_k = y_k$ ;
end for
end if
end if
```

- Puncturing is a method in doing rate matching, in this method the mother code length N is longer than the output code length of the rate matching block E so we will puncture the first N-E interleaved bits where the position of the punctured bits is not systematically, but it's chosen carefully to achieve best SNR
- Shortening is same as puncturing but in this method, we remove last N-E interleaved bits
- Repetition is another method in rate matching, and it is used when the output length needed from the rate matching block is more the mother code length.

So the method used in rate matching will depend on the output length needed from the block, in our design the output length of the block is smaller than the mother code length as the output of the LDPC is 19200 bits while we need 3840 to come out of the rate matching block so we will do puncturing.

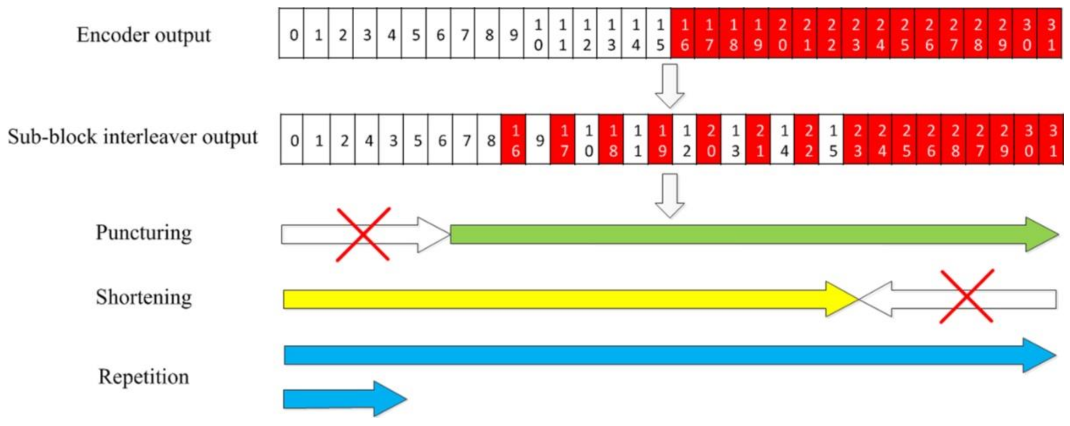


Figure 67:Interleaving and rate matching procedure of 5G polar codes

The output of rate DE-Rate matching denoted by $d_0, d_1, d_2, \ldots\ldots, d_{N-1}$, where N is the total number of output bits, for LDPC base graph number 1 total number of output bits equals $N = 66 * Z_c$ and for LDPC base graph number 2 total number of output bits equals $N = 50 * Z_c$.

According to 38.212 version 15.5.0 release 15:

We used code rate = 1/5 that leads us to use base graph number 2 as it shown in the following figure also, we will choose expansion factor (Z_c) equals 384.

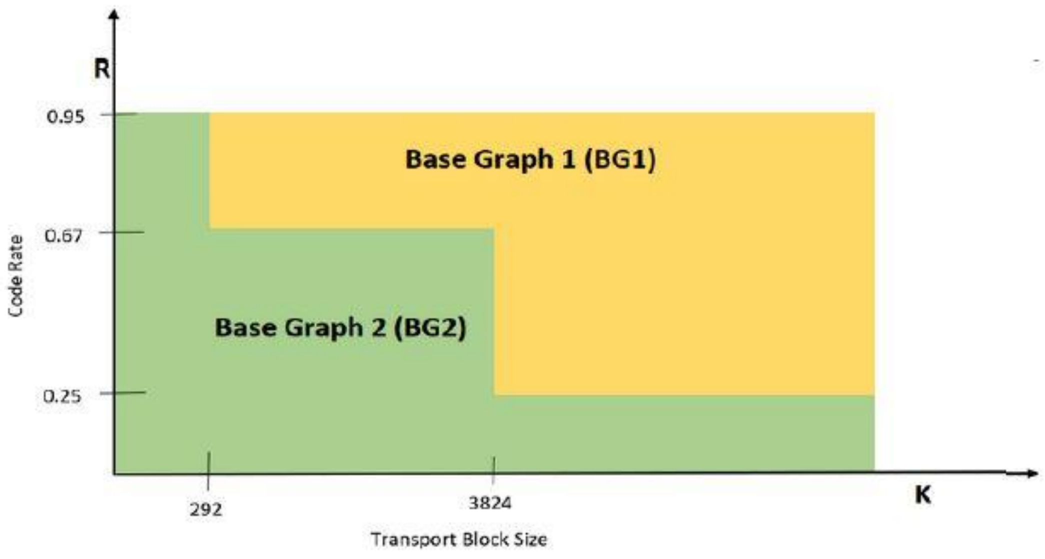
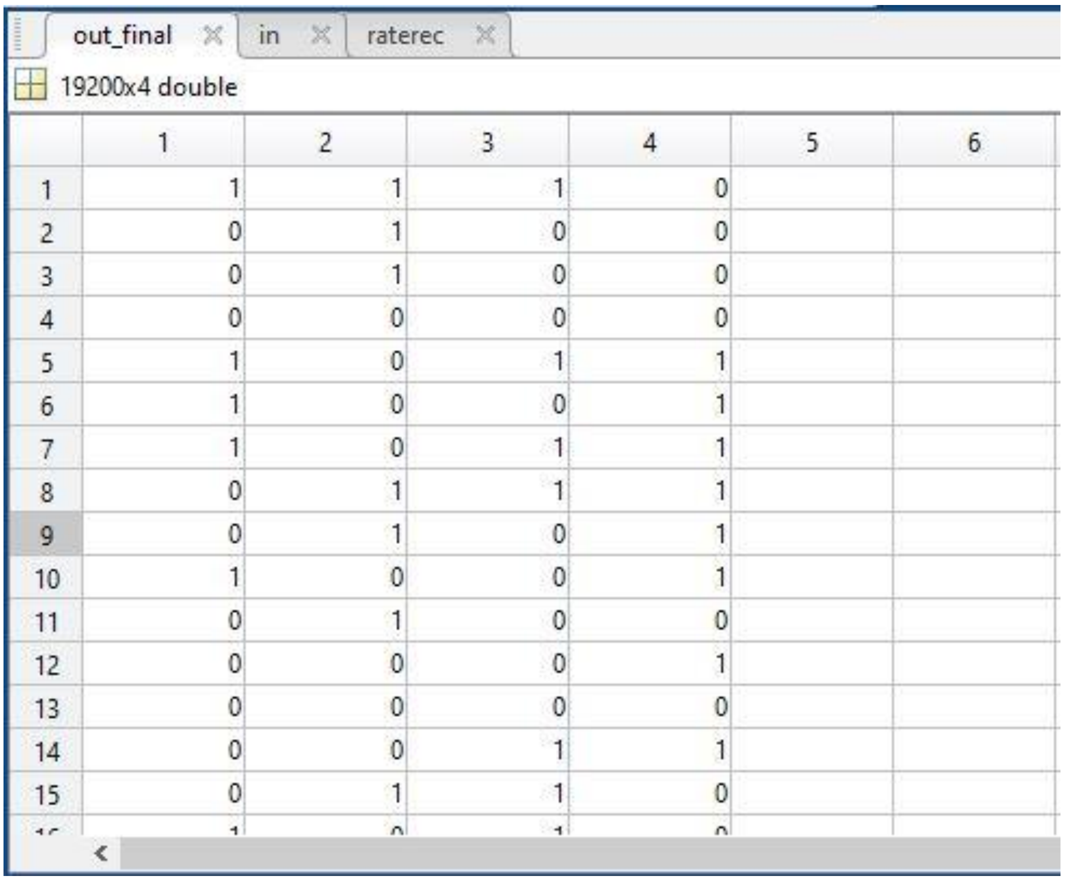


Figure 68: choosing base graph number

Assume the following:

- Maximum modulation order configured for the serving cell, if configured by higher layers, otherwise a maximum modulation order Qm = 6.
- C is the number of code blocks of the transport block.
- Denoting by E_r the rate matching output sequence length
- N_L is the number of transmission layers that the transport block is mapped.
- Q_m is the modulation order.
- G is the total number of coded bits available for transmission of the transport block.

High level simulation:

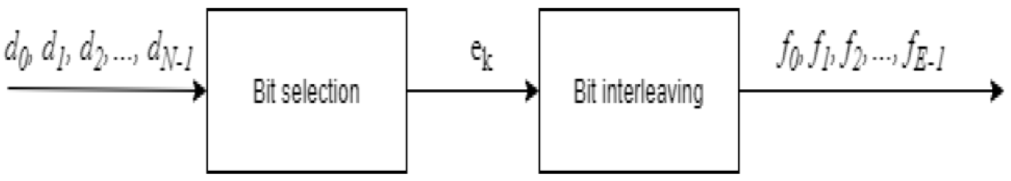


	1	2	3	4	5	6
1	1	1	1	0		
2	0	1	0	0		
3	0	1	0	0		
4	0	0	0	0		
5	1	0	1	1		
6	1	0	0	1		
7	1	0	1	1		
8	0	1	1	1		
9	0	1	0	1		
10	1	0	0	1		
11	0	1	0	0		
12	0	0	0	1		
13	0	0	0	0		
14	0	0	1	1		
15	0	1	1	0		

Figure 69:high level simulation results of rate recover

Hardware implementation:

Block diagram and flow chart:



$d_0, d_1, d_2, \ldots, d_{N-1}$
Bit selection
e_k
Bit interleaving
$f_0, f_1, f_2, \ldots, f_{E-1}$

Figure 70: block diagram of rate matching

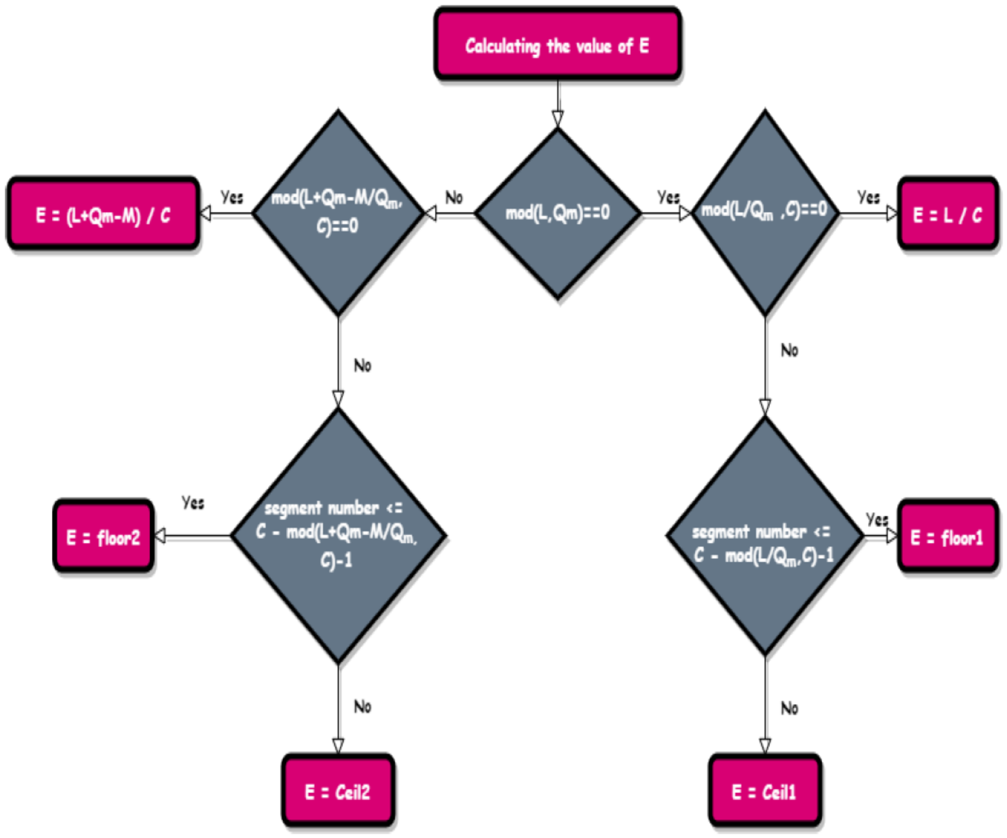


Figure 71: flowchart to calculate E

- E is the length of data from each segment.
- L is the output length.
- Qm is the modulation type.
- M is the modulus of the output length and modulation type.
- C is the number of segments.

If the modulus of output length divide by the modulation type and number of segments is equal to 0, we will take an equal number of bits from each segment equals to the output length divided by the number of segments.

If modulus not equal to 0, the number of bits taken from each segment will change according to the segment number:

- If the segment number is less than or equal the value of this modulus subtracted from the number of segments, then added -1, we will take the floor value from the division of output length divided by the number of segments and modulation type.
- If the segment number is larger than the value of this modulus subtracted from the number of segments then added -,1 we will take the ceil value from the division of output length divided by the number of segments and modulation type.

Operation: Firstly, we calculate the value of E according to the modulation scheme used, number of layers & total bits of the transport block therefore we will have 4 segments as the transport data bits is 15000 bits, Secondly, we do De-interleaving to each segment opposite to the way of interleaving happens at the transmitter then we add filler bits according to the rate we are working on.

Block interface:

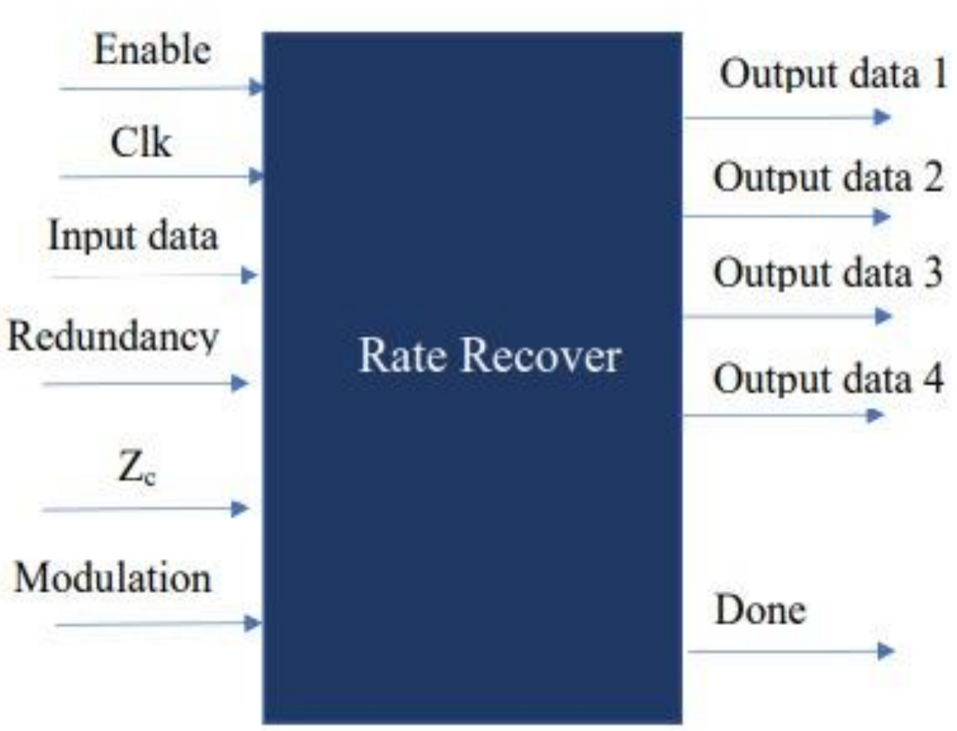
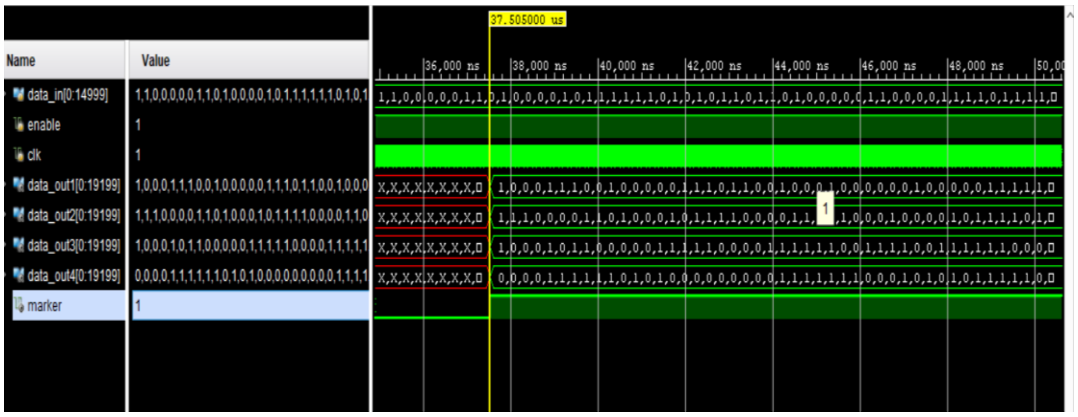


Figure 72: Block interface of Rate recovery block

Input output ports:

Port Name	Direction	Width	Description
Input data	input	15000 bits	Input data to the block
Enable	input	1 bit	Enable signal to the Ram module to assign data in rows
Clk	input	1 bit	Operating clock to write into the block.
Reset	input	1 bit	Resetting the block.
Redundancy version	input	2 bits	Redundancy version to choose the index of starting bit. It has four options (0,1,2,3) and each one indicates a starting bit. 00 → 0 01 → 1 10 → 2 11→ 3 In our design we use redundancy version=0
Modulation order "Q_m"	input	4 bits	Modulation type 0000→QPSK 0100→ 16-QAM 0110→ 64 QAM 1000 →256 QAM
Output data	output	4 segments each is 19200 bits	Output data of rate recover block which will enter LDPC
Done	output	1 bit	Indicate whether output is calculated or not

Hardware simulation results:



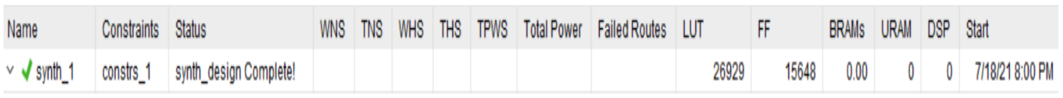
```
Name                 Value
data_in[0:14999]     1,1,0,0,0,0,0,1,1,0,1,0,0,0,0,1,0,1,1,1,1,1,1,0,1,0,1
enable               1
clk                  1
data_out1[0:19199]   1,0,0,0,1,1,1,0,0,1,0,0,0,0,0,1,1,1,0,1,1,0,0,1,0,0,0
data_out2[0:19199]   1,1,1,0,0,0,0,1,1,0,1,0,0,0,1,0,1,1,1,1,0,0,0,0,1,1,0
data_out3[0:19199]   1,0,0,0,1,0,1,1,0,0,0,0,0,1,1,1,1,1,0,0,0,0,1,1,1,1,1
data_out4[0:19199]   0,0,0,0,1,1,1,1,1,1,0,1,0,1,0,0,0,0,0,0,0,0,0,1,1,1,1
marker               1
```

Figure 73: hardware simulation of rate recover

It is obvious that the output of the Verilog simulation is same as output of the MATLAB code where we have 4 output segments each is 19200 bits so each segment could enter LDPC, so Rate Recover could retrieve the original signal

Synthesis results:

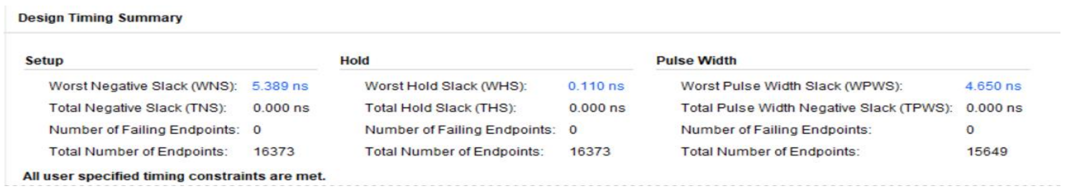
Synthesis validation:



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start
✓ synth_1	constrs_1	synth_design Complete!								26929	15648	0.00	0	0	7/18/21 8:00 PM

Figure 74: Synthesis validation of rate recover

Timing report:



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.389 ns	Worst Hold Slack (WHS):	0.110 ns	Worst Pulse Width Slack (WPWS):	4.650 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	16373	Total Number of Endpoints:	16373	Total Number of Endpoints:	15649

All user specified timing constraints are met.

Figure 75 :timing report of rate recover

Analysis of synthesis result:

Design is synthesizable with 26929 LUT & 15648 FF Also it met the timing constraints with large positive slack at 100 MHZ which enable us to operate this block at higher frequency.

Low Density Parity Check Decoder

Wireless communication applications require a super robust encoding in order to transmit the data from the source to destination through a communication channel with low error rate and high reliability. Low density parity check codes were introduced by Gallager in 1960s, they are linear block codes with parity check matrices containing large number of zero entries.

LDPC codes can outperform turbo codes as they acquire less complex mathematical relations and computations could also support parallelism which is suitable for hardware implementation concepts. The tanner graph is considered as the graphical representation of the code and the base to understand how the decoding works as it represents the parity check matrix while the parity check matrix H of dimension *m*n* defines the LDPC code in matrix form where n is the code word length and m is the parity bits. This class of codes decoded with an iterative soft-input and hard or soft-output decoding algorithm. LDPC codes play an important role in 5G communication systems and have been selected as the coding scheme for the 5G enhanced Mobile Broad Band (eMBB) data channel. Since 5G operates at high frequencies subsequently high data rates, 3rd Generation Partnership Project (3GPP) has agreed to consider two rate-compatible base graphs, BG1 and BG2, for the channel coding to support harmonious rate and scalable data transmission. Low density parity check codes are used to allow correction of transmission errors that had already occurred due to transmission over noisy channel; in addition, they are called capacity approaching codes since they allow transmission of data at maximum rate which is determined by Shannon-Hartley Theorem.

LDPC came as a replacement for Turbo codes which were used in 4G LTE, so we have first to know more about the Turbo codes, and what are the merits of LDPC that made it more preferable.

Turbo Codes:

To sum up, turbo encoder uses 2 recursive systematic convolutional codes, where data is introduced to the first encoder while it is being interleaved and then introduced to another encoder to get another parity bits, so the conventional turbo encoder gives 1/3 code rate which can be increased by puncturing some parity check bits.

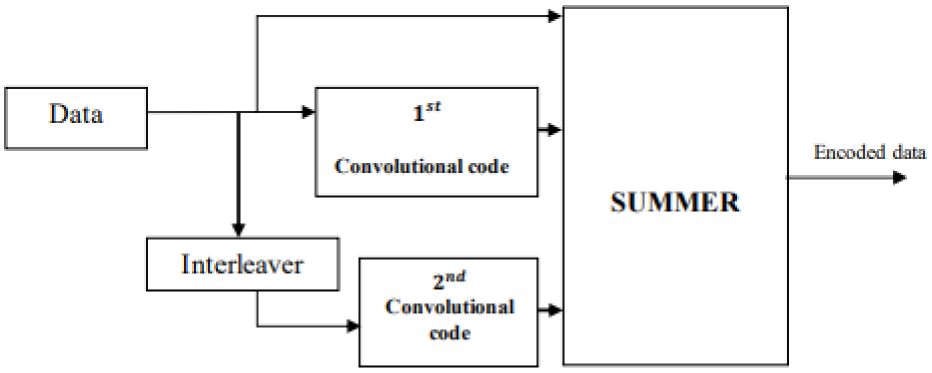


Figure 76: Turbo encoder

In Receiver, the decoder consists of two decoders that everyone works with the corresponding parity bits to its data bits, and there are also an interleaver which is used to retrieve the interleaved data to its original positions.

Turbo decoders depend on Maximum a posteriori Probability (MAP) which calculates the logarithmic ratio between the probability of receiving “1” vs receiving “-1” taking in consideration that data is binary phase shift key (BPSK) encoded.

The log-likelihood ratio is defined as:

$$L(\hat{u}_k) = \ln\left(\frac{P(u_k = +1|y)}{P(u_k = -1|y)}\right)$$

Where $\hat{u}_k$ is the information bit at time k.

It is also important to notice that turbo decoding depends on iterative method not just one iteration, so assuming that decoding process takes 10 iterations, and by testing this encoder at different code rates we found that the number of consumed cycles for 10 iterations:

Code Rate	Number of clock cycles for 10 iterations
1/3	7250K
1/2	7250K
3/4	7250K

Where K is the Frame length.

LDPC Codes:

However, if we applied the same methodology on LDPC; in addition, we increased the number of iterations that improve the Bit Error Rate (BER):

Code Rate	Number of clock cycles for 20 iterations
1/3	1000K
1/2	500K
3/4	326K

So, we will find that at higher code rates LDPC codes complexity decreased as the number of clock cycles needed decreased. However, turbo codes showed constant complexity over all rates, so LDPC is preferred to be used at higher code rates which is much needed in 5G technology.

In the next section, we will explain the LDPC encoder and how it works, in order to be able to understand the decoder.

LDPC Encoder:

LDPC codes are characterized by the overlapping connections between both parity and message bits which make it more immune to channel effects, this structure will be discussed by the next toy example:

Consider an example of 6 bits $(d_1, d_2, d_3, d_4, d_5, d_6)$, if we arranged them in 2x3 block:

$$\begin{vmatrix} d_1 d_2 d_3 \\ d_4 d_5 d_6 \\ - - - \\ p_3 p_4 p_5 \end{vmatrix} \begin{matrix} p_1 \\ p_2 \\ \\ \end{matrix}$$

As shown these 6 bits are protected by 5 parity bits, and the relations between message bits and parity bits are:

$$d_1 + d_2 + d_3 = p_1$$

$$d_4 + d_5 + d_6 = p_2$$

$$d_1 + d_4 = p_3$$

$$d_2 + d_5 = p_4$$

$$d_3 + d_6 = p_5$$

Where, the addition operation here is addition modulo 2 (XOR).

As we see that each relation connects different message bits with parity bit, and these relations can be increased according to the required code rate. Code rate is the ratio between number of data bits and number of total bits in code word (data bits + parity bits), from this definition, our code rate here is 6/11.

Now, the code word that will be sent over the channel is:

$$C = [d_1\ d_2\ d_3\ d_4\ d_5\ d_6\ p_1\ p_2\ p_3\ p_4\ p_5]$$

It is also important to notice that this coding method provides constant number of parity bits for a constant number of data bits. That’s why it is called linear block code.

The coding done can also be described using sparse parity check matrix (H), and it is called sparse because the number of zeros exceeds the number of ones in the matrix:

$$H = \left(\begin{array}{ccccccccccc} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & p_1 & p_2 & p_3 & p_4 & p_5 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)_{5x11} = (C^T|I)$$

Each row in H-matrix enforces one parity check constraint, and the number of rows represents the number of available parity bits. For example, if d_2 bit is lost, we can get it from the first or the fourth row by applying this condition:

$$HC^T = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}_{5x11} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{pmatrix}_{11 \times 1} = 0$$

Now, you can extract from first row that:

$$d_1 + d_2 + d_3 + p_1 = 0$$

So, if d_2 is missing, you can retrieve it using this relation.

This matrix can also be represented graphically using tanner graph:

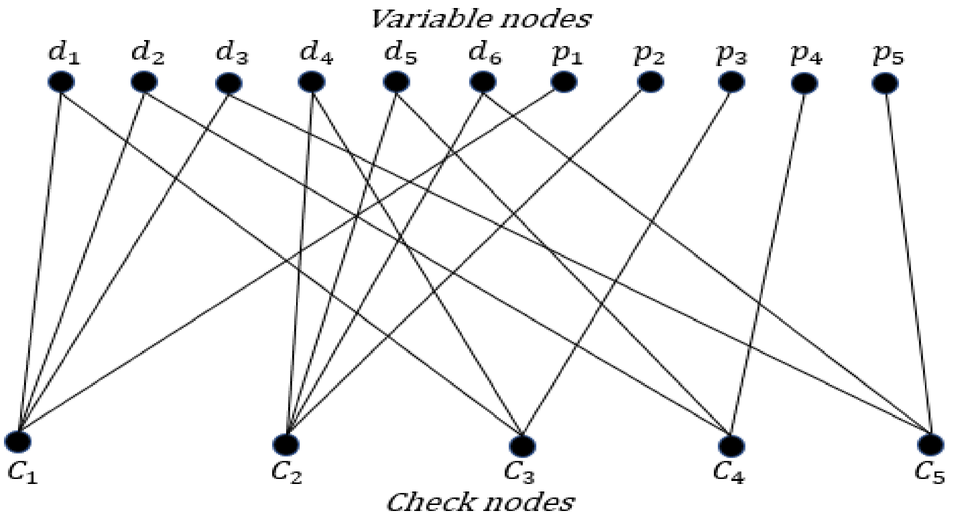


Figure 77 Tanner graph

Where, **variable nodes** are the bits that are transferred within the code word & **check nodes** are nodes that are stating which bits are conserved within the same parity check equation.

It is also important to notice that LDPC codes must ensure that each bit is connected by at least two edges (has two equations), so this example needs modification as parity bits are connected by single edge. For example, parity check matrix can be updated to be:

$$H = \left(\begin{array}{ccccccccccc} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & p_1 & p_2 & p_3 & p_4 & p_5 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right)_{5x11}$$

Now, the updated tanner graph will be including 2 edges for $p_1,\ p_2,\ p_3,\ p_5$, and 3 edges for p_4. So, we can say that number of edges for each bit can be known by counting the number of ones in each column.

Since that interoperability is being aimed nowadays, the **3rd Generation Partnership Project** (**3GPP**) issued standard specifications for the LDPC.

LDPC in standards:

Base Graph (BG):

It is a matrix which includes numbers ranging from 0 to Z_c-1, where Z_c is the expansion factor that is used to expand the base graph in order to get the parity check matrix (H-matrix).

In order not to use different H-matrices, 3GPP standardize 2 Base graphs that can be used by the operators:

- BG1 is matrix of 46 rows and 68 Columns.
- BG2 is matrix of 42 rows and 52 Columns.

The **selection of base graph** depends on the following criteria: (ETSI TS 138 212 V15.5.0 (2019-05), section 6.2.2 LDPC base graph selection)

- if K<=3824 and R<=0.67 then BG2 is selected.
- If K<= 292 then BG2 is selected
- if R<=0.25 then BG2 is selected.
- Else BG1 is selected

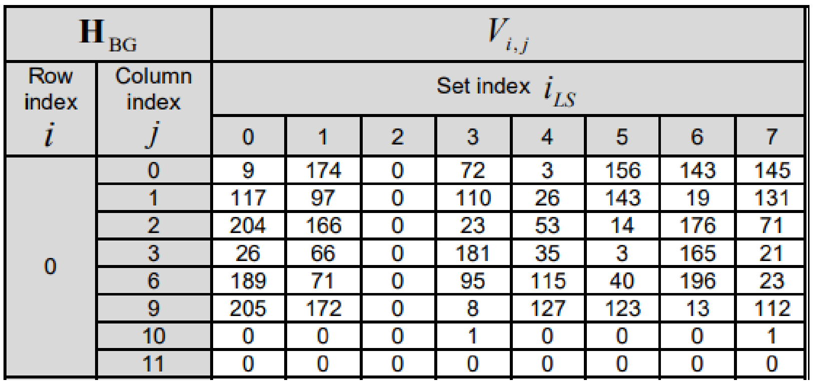
Where K is the payload size.

Since that our project has the following specifications:

- Transport block size=14856 bit.
- Code Rate=1/5. (<0.25)

Therefore, we will use BG2 in our system design.

If we look at BG2 in the standards, we will see that:



$\mathbf{H}_{BG}$		$V_{i,j}$							
Row index i	Column index j	Set index i_{LS}							
		0	1	2	3	4	5	6	7
0	0	9	174	0	72	3	156	143	145
	1	117	97	0	110	26	143	19	131
	2	204	166	0	23	53	14	176	71
	3	26	66	0	181	35	3	165	21
	6	189	71	0	95	115	40	196	23
	9	205	172	0	8	127	123	13	112
	10	0	0	0	1	0	0	0	1
	11	0	0	0	0	0	0	0	0

So, what is set index ($\boldsymbol{i_{LS}}$) and how can we obtain it?

According to our system, transport block data experienced some steps before entering LDPC:

First Step: Calculating total number of bits after adding overheads:

According to standards, CRC24 is added to the transport block before segmentation:

$$transport\ block + CRC24 \ = 14856 + 24 = 14880\ bits$$

For using BG2, the maximum code block size must be ≤ 3840, so this transport block needs segmentation:

$$Number\ of\ code\ blocks = ceil\left(\frac{14880}{3840 - 24}\right) = 4\ blocks$$

Note that:

We subtracted 24 bits from the maximum code block size in order to leave a space for CRC bits that will be applied over each block again.

Now, we need to calculate the total block size after applying CRC24 for each block:

$$Total\ number\ of\ bits = transport\ block + CRC24 + Number\ of\ code\ blocsk \times CRC24$$

$$= 14880 + 4 \times 24 = 14976\ bits$$

Second Step: Calculating expansion Factor:

To choose the i_{ls} in order to define which BG elements you will use, you have to calculate the expansion factor (Z_c), and according to standards this can be calculated by the following procedure:

For BG2: $Maximum\ code\ block\ size = 10Z_c$

So, we have to calculate number of bits in each code block:

$$Bits\ in\ each\ code\ block = \frac{14976}{4} = 3744\ bits$$

For BG2:

- if K > 640 then K_b = 10
- if K is between 560 < K <= 640 then K_b = 9
- if K is between 192 < K <= 560 then K_b =8
- If K is <= 192 then K_b is = 6

Where K_b constant, and K is is number of bits in each code block (Payload size).

Then: $K_b.Z_c \geq K$

$$\therefore Z_c \geq \frac{3744}{10} = 374.4$$

So, we will choose $\boldsymbol{Z_c} = \mathbf{384.}$

<u>Third Step:</u> choosing $\boldsymbol{i_{ls}}$:

Set index (i_{LS})	Set of lifting sizes (Z)
0	{2, 4, 8, 16, 32, 64, 128, 256}
1	{3, 6, 12, 24, 48, 96, 192, 384}
2	{5, 10, 20, 40, 80, 160, 320}
3	{7, 14, 28, 56, 112, 224}
4	{9, 18, 36, 72, 144, 288}
5	{11, 22, 44, 88, 176, 352}
6	{13, 26, 52, 104, 208}
7	{15, 30, 60, 120, 240}

It is obvious that, $Z_c = 384$ exists within $\boldsymbol{i_{ls}} = \mathbf{1}$ expansion factors.

Now, we can get our base graph matrix using $\boldsymbol{i_{ls}}$.

As we said before, $\boldsymbol{Z_c}$ is used to expand Base graph matrix in order to get parity check matrix, but how is it done?

For example, we will take part of our Base graph matrix to describe how it will be expanded.

Assuming $Z_c = 4$, this means that each number in our base graph matrix will be expanded to 4x4 identity matrix shifted circularly to the right by the amount of the number.

From BG2:

$\mathbf{H}_{\text{BG}}$		$V_{i,j}$							
Row index i	Column index j	Set index i_{LS}							
		0	1	2	3	4	5	6	7
0	0	9	174	0	72	3	156	143	145
	1	117	97	0	110	26	143	19	131
	2	204	166	0	23	53	14	176	71
	3	26	66	0	181	35	3	165	21
	6	189	71	0	95	115	40	196	23
	9	205	172	0	8	127	123	13	112
	10	0	0	0	1	0	0	0	1
	11	0	0	0	0	0	0	0	0

Since that $Z_c = 4$, therefore $i_{ls} = 0$, and the shifting elements should be ranging from 0 to $Z_c - 1$, so we will first calculate the shifting elements ($P_{i,j} = mod(V_{i,j}, Z_c)$):

Taking the first row ($i = 0$) and columns with index ($j = 0,1,2,3,6,9,10,11$),

$P_{0,0} = mod(9,4) = 1$

$P_{0,1} = mod(117,4) = 1$

$P_{0,2} = mod(204,4) = 0$

$P_{0,3} = mod(26,4)$ =2

$P_{0,4} = -1$ (Any unavailable index in standards has value of -1).

According to standards:

"-1": All Zeros 4x4 matrix.

"0": Identity 4x4 matrix.

"1": Identity 4x4 matrix with the last column circularly shifted to the right.

"2": Identity 4x4 matrix with the last 2 columns circularly shifted to the right, and so on.

Therefore, the first row will be:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Now, we have understood the main idea of LDPC encoding; as a result, in the next sections, we will discuss the LDPC decoding process.

System Logic

As we have said before, each message bit is connected to multiple edges of parity bits and message bits, so for each edge we need single parity check (SPC) decoder to get the value of this bit based on the other connected bits within the same check node. As a result, we will finish having many different values of ones and zeros which needs repetition decoder to simplify them into one bit which we will take the final decision based on it.

So, LDPC decoder is mainly consists of SPC decoder and Repetition decoder.

Soft Input Soft Output decoder for (n, n-1) single parity check code

The received message contains intrinsic and extrinsic parts, the intrinsic part could be decoded knowing only the information of the channel and determined using log likeli-hood function (LLR), while the extrinsic part is decoded knowing information about the other received message bits.

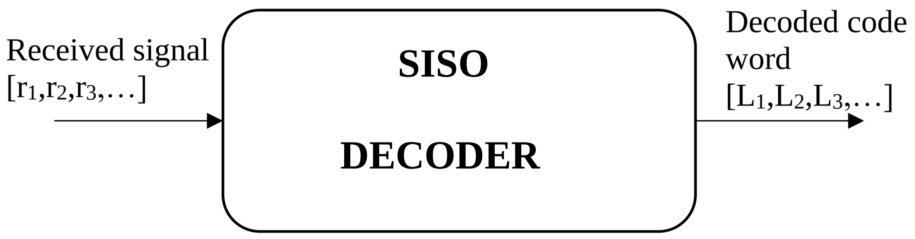


Figure 78 SISO Decoder Block Interface

L: represents decoded word which consists of the intrinsic and extrinsic parts.

The intrinsic part for $L_n \propto r_n$ while the extrinsic part $L_n \propto r_{j\neq n}$.

For example if we have SPC code (3,2), due to parity check condition in the encoding that tells (numbers of ones are always even) we have a relation between coded message bits, $c_1 = c_2 \oplus c_3$ (mod 2) by that we have the extrinsic information of c_1.The decoded message is hard decoded so it's either 0 or 1, given the received message, so we have two conditional probabilities to calculate the extrinsic part of the decoded word. Note that the values above zero is decoded zero.

$$l_{ext,n} = \log \frac{P(C_n = 0|r_n)}{P(C_n = 1|r_n)} = \log \frac{P_n}{1 - P_n}$$

For SPC code (3, 2) given P_2 and P_3 what is P_1? $P_1 = p(C_1 = 0|r_2, r_3)$, how to calculate P_1?

The table shown below is the cases for coded single parity check messages.

C1	**C2**	**C3**
0	0	0
0	1	1
1	0	1
1	1	0

We want to see how we could represent the PC condition of XORs in terms of probabilities.

$$P_1(probability\ of\ C_1 = 0) = P_2P_3 + (1 - P_2)(1 - P_3)$$

$$(1 - P_1)(probability\ of\ C_1 = 1) = P_2P_3 + (1 - P_2)\ (1 - P_3)$$

$$Subtracting\ equation\ nos\text{:}\ P_1 - (1 - P_1) = P_2\big(P_3 - (1 - P_3)\big) +$$

$$(1 - P_2)\big((1 - P_3) - P_3\ \big) = (P_2 - (1 - P_2))(P_3 - (1 - P_3))$$

The extrinsic information of L_1 now needs to be calculated so by dividing equation no by $P_n + (1 - P_n)$

$$\frac{P_1 - (1 - P_1)}{P_1 + (1 - P_1)} = \frac{(P_2 - (1 - P_2))(P_3 - (1 - P_3))}{(P_2 + (1 - P_2))(P_3 + (1 - P_3))}$$

$$\frac{1 - \frac{(1 - P_1)}{P_1}}{1 + \frac{(1 - P_1)}{P_1}} = \frac{(1 - \frac{(1 - P_2)}{P_2})(1 - \frac{(1 - P_3)}{P_3})}{(1 + \frac{(1 - P_2)}{P_2})(1 + \frac{(1 - P_3)}{P_3})}$$

$\because\ Tanh\ (x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}$ and $L_{ext,n} = \log \frac{P_n}{1 - P_n}$

$\therefore equation\ no\ yields\ to\ \frac{1-e^{-l_{ext,1}}}{1+e^{-l_{ext,1}}} = \frac{\left(1-e^{-l_{ext,2}}\right)\left(1-e^{-l_{ext,3}}\right)}{\left(1+e^{-l_{ext,2}}\right)\left(1+e^{-l_{ext,3}}\right)}$

$$\longrightarrow \tanh\left(\frac{l_{ext,1}}{2}\right) = \tanh\left(\frac{l_2}{2}\right).\tanh\left(\frac{l_3}{2}\right)\ ---\ equation\ (A)$$

To sum up in the SPC code when you have $c_1 = c_2 \oplus c_3$ given $L_2\ and\ L_3$ we can get $L_{ext,1}$ by the previous **tanh rule**.

For simplicity of tanh rule we can do the following,$\because$ $\tanh(x)$ is an odd function

where$\begin{cases} x < 0\ (negative), \tanh(x) < 0 \\ x > 0\ (positive), \tanh(x) > 0 \end{cases}$

So we can simplify equation A to two parts sign equation and absolute value equation, the sign equation is very straightforward

$$sign\left(l_{ext,1}\right) = sign(l_2).sign(l_3)$$

$$\tanh\left(\frac{\left|l_{ext,1}\right|}{2}\right) = \tanh\left(\frac{|l_2|}{2}\right).\tanh\left(\frac{|l_3|}{2}\right)$$

Since summation is much easier in implementation than product we take log of equation no to transfer product to summation.

$$\log(\tanh\left(\frac{\left|l_{ext,1}\right|}{2}\right)) = \log(\tanh\left(\frac{|l_2|}{2}\right)) + \log(\tanh\left(\frac{|l_3|}{2}\right))$$

Define $f(x)$ which captures this log tanh function $\longrightarrow f(x) = \left|\log\left(\tanh\left(\frac{|x|}{2}\right)\right)\right|, for\ x > 0$ which has a property that $f^{-1}(x) = f(x)$ now we can express the relation between $l_{ext,1} and\ l_2\ and\ l_3$ using the following definition:

$$f\left(\left|l_{ext,1}\right|\right) = f(|l_2|) + f(|l_3|)$$

$$\boxed{\begin{array}{c} \therefore \left| l_{ext,1}\right| = f\left(f(|l_2|) + f(|l_3|)\right) \\ \\ sign\left(l_{ext,1}\right) = sign(l_2).sign(l_3) \end{array}}$$

To summarize the SISO decoder of SPC code we can define:

$$L_i \;\; (Decoded\, word) = l_i + l_{ext,i}$$

$$S\ (absolute\ value\ function) = \ f(|l_1|) + f(|l_2|) + f(|l_3|) + \cdots .f(|l_n|)$$

$$Parity(product\ of\ signs\ ) = \ sign(l_1).sign(l_2).sign(l_3)\dots\dots sign(l_n)$$

$$\boldsymbol{n}\text{: }\boldsymbol{code\ word\ block\ length.}$$

$$\boxed{\therefore l_{ext,i} = \underbrace{f\left(S - f(|l_i|)\right)}_{absolute\ value} . \underbrace{P.sign(l_i)}_{sign}}$$

This means the extrinsic value of the message could be retrieved using the intrinsic part of the message.

Steps to decode SPC code:

1. Calculate intrinsic values
2. Refer to parity check matrix which gives the relation between coded words.
3. Get probability of the expected value of the decoded word.
4. Use simplified tanh rule to retrieve the decoded word (extrinsic part).

As we know LDPC decoder is separately defined into SPC code which we have discussed and repetition code which will be discussed in the following section.

Repetition decoder:

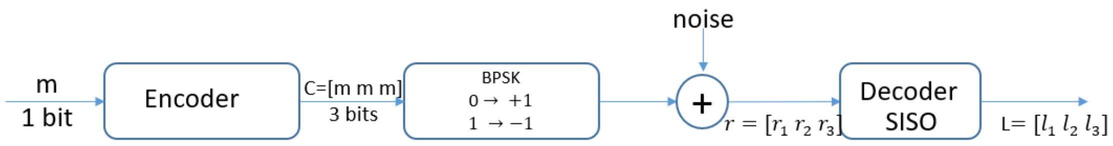


Figure 79 SISO Decoder for repetition decoder over BPSK

Repetition code is the idea of repeating the message several times, by this way in order the receiver can notice the error happened in the received message easily and can recover the original message by looking at its replicas by this the data is robust to the noise caused by channel.

From figure 79 we can notice that m (message) is encoded to c which is 3 repeated bits so how can we retrieve m (intrinsic part)?

m retrieval now depends on a decision based on the 3 (repetition code degree) received bits which is represented by L.

To find strong belief for L we use log likelihood function that retrieves intrinsic value of the message.

$$p(c_i = 0|r_i) = \frac{f(r_i|c_i = 0).p(c_i = 0)}{f(r_i)}$$

$$p(c_i = 1|r_i) = \frac{f(r_i|c_i = 1).p(c_i = 1)}{f(r_i)}$$

Divide equation nos: $\frac{p(c_i = 0|r_i)}{p(c_i = 1|r_i)} = \frac{f(r_i|c_i = 0)}{f(r_i|c_i = 1)}$

This ratio is called likelihood ratio (LLR) which simplifies to $\boldsymbol{e}^{\frac{2r_n}{\sigma^2}}$

$\boldsymbol{\sigma\ is\ the\ stanard\ deviation\ of\ the\ noise\ affecting\ the\ channel}$

<u>Intrinsic LLR or Channel LLR</u>

$$\boxed{\boldsymbol{l_i = \log\left(\frac{p_r(c_i = 0|r_i)}{p_r(c_i = 1|r_i)}\right) = \frac{2}{\sigma^2} \times r_i}}$$

<u>What is the final total belief after observing all received repeated values?</u>

$$\boxed{\boldsymbol{L_i = \log\left(\frac{p_r(c_i = 0|r_1, r_2, r_3, \dots r_n)}{p_r(c_i = 1|r_1, r_2, r_3, \dots . r_n)}\right) = \underbrace{\underbrace{\frac{2}{\sigma^2} \times (r_i}_{Intrinsic\ value} + r_{j\neq i} + \cdots .. + r_n)}_{extrinsic\ value}}}$$

In practical life, calculating the extrinsic part with the previous algorithm will be impossible, since it contains log and tanh which is impossible to implement them as a hardware, so we will try to modify these functions.

Minimum Sum Approximation:

By taking a look at the plot of $f(|x|) = \left|\log\left(\tanh\left(\frac{x}{2}\right)\right)\right|$, we'll find it dominates (have its highest value) at the minimum value of abs(x).

$\therefore\ f(|L_2|) + f(|L_3|) \cong f(\min(|L_2|, |L_3|))$

$$\therefore |L_{ext}, 1\,| = f\big(f(\min(|L_2|, |L_3|))\big)$$

$$\boxed{\therefore |L_{ext}, 1\,| \cong \min(|L_2|, |L_3|)}$$

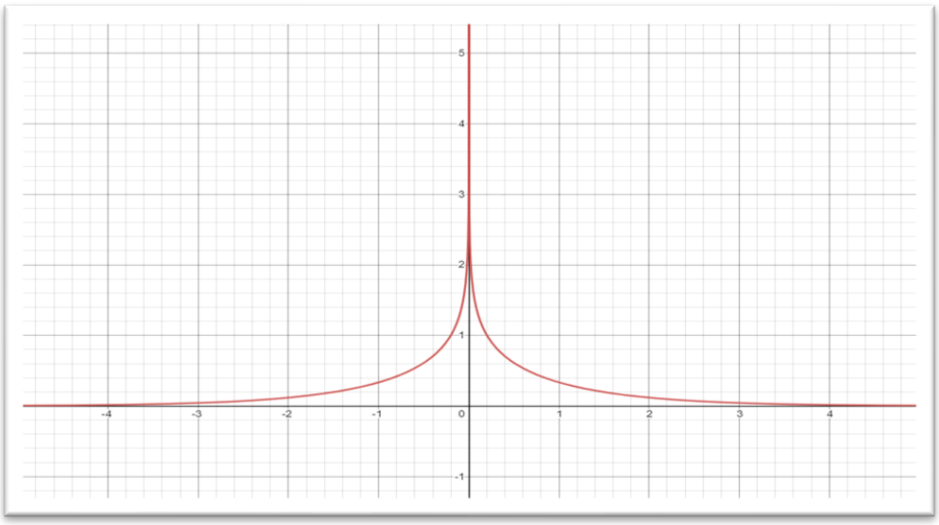


Figure 80 Plot of f(x)

So, the above algorithm of SPC decoder will be simplify to certain steps that we have already used in our High level model.

SPC Decoder for code word of size n:

1. $l_i = \frac{2}{\sigma^2} \times r_i$.
2. determine the sign of $\underline{l}$.
3. Compute S=sign of each l multiplied, $S = sign(l_1) \times sign(l_2) \times .. sign(l_n)$.
4. Sign($l_{ext,i}) = S \times Sign(l_i)$.
5. Find the first minimum in each row of data

$$m_1 = \min\,(|l_1|, |l_2|, \dots \dots., |l_n|)$$

6. Find the position of first minimum in the row

$$pos = \arg\min\,(|l_1|, |l_2|, |l_3|, \dots \dots, |l_n|)$$

7. Find the second minimum in the row

$$m_2 = \min\,(|l_1|, |l_2|, \dots, |l_{pos-1}|, |l_{pos+1}|, \dots., |l_n|)$$

8. $|l_{ext,pos}| = m_2$, $|l_{ext,i}| = m_1 , i \neq pos.$

Finally, we can .say that the LDPC decoder is a combination of SPC decoder and Repetition decoder with minimum sum approximation, so it is call minimum sum decoder.

High level simulation

Minimum Sum Algorithm:

The type of algorithms used to decode LDPC codes are inclusively called message-passing algorithms since their behavior can be explained by the passing of messages along the edges of mentioned Tanner graph. Each Tanner graph node works separately, it only has access to the information contained in the messages on the edges connected to it. The message-passing algorithms also are referred to as iterative decoding algorithms because the messages pass back and forth between the variable (bit) and check nodes iteratively until a result is achieved (or the process halted) which is determined by the designer according to a certain quantization error that we want to be achieved.

As mentioned before, min-sum algorithm simplifies the calculation of check nodes by observing that the term corresponding to the smallest (minimum) variable (bit) node message takes over the product term and so the product can be approximated by a minimum, recall that $|\boldsymbol{l_{ext,i}}| \cong \mathbf{min}(|\boldsymbol{l}_{\mathrm{j\neq i}}| \ldots . |\boldsymbol{l_n}|)$

We will use the following example to illustrate the operation of minimum sum algorithm, we assume binary phase-shift keying (BPSK) transmission over an additive white Gaussian noise (AWGN) channel, let the binary encoded word C=$\{C_i\}$ $(i \in [1, n])$ where n is the code word block length. If additive white noise is an independent and identically distributed random variable that has mean value and standard deviation of values 0 and σ respectively, then the signal received from the channel is $r_i = c_i + n_i$.

Let the received signal be:

$$r = [0.2\ -0.3\ \ 1.2\ -0.5\ \ 0.8\ \ 0.6\ \ -1.1]$$

And the parity check matrix (H) is:

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

Loglikli-hood Ratio (Belief propagation)

<u>Step (1): Initialization:</u>

All messages passing from a variable node to a check node are initialized to $l_i = \frac{2}{\sigma^2} \times r_i$. This value depends on the type of channel under estimation. This is the discussed SISO decoder that deals with beliefs or log likelihood ratio for the received values for the bits, this is called the first estimate from the channel itself.

Since that: $H.r^T = 0$

$$L\ is\ a\ storage\ matrix\ of\ the\ same\ dimension\ of\ H\ (PC\ matrix)$$

L is of size (4,7)

$$\therefore L_{4\times 7} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.2 \\ -0.3 \\ 1.2 \\ -0.5 \\ 0.8 \\ 0.6 \\ -1.1 \end{bmatrix}$$

$$\therefore L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & 0 & 0.8 & 0 & 0 \\ 0 & -0.3 & 1.2 & -0.5 & 0 & 0.6 & 0 \\ 0.2 & -0.3 & 0 & -0.5 & 0 & 0 & -1.1 \\ 0.2 & 0 & 1.2 & 0 & 0.8 & 0.6 & -1.1 \end{bmatrix}$$

Iterative Message Passing Decoder

We use partial info from the code then do some decoding then iterate on the message until hard output decision is taken. Every single parity check gives an estimate using SPC decoder discussed earlier and each row of the PC matrix defines SPC code.

We start by mapping SPC code by looking at tanner graph, where every check node enforces spc (constraint) for code word.

For each bit we have 4 Expectations which are calculated by the SPC decoder. We need repetition decoder to get the final decision for this bit from the 4 beliefs.

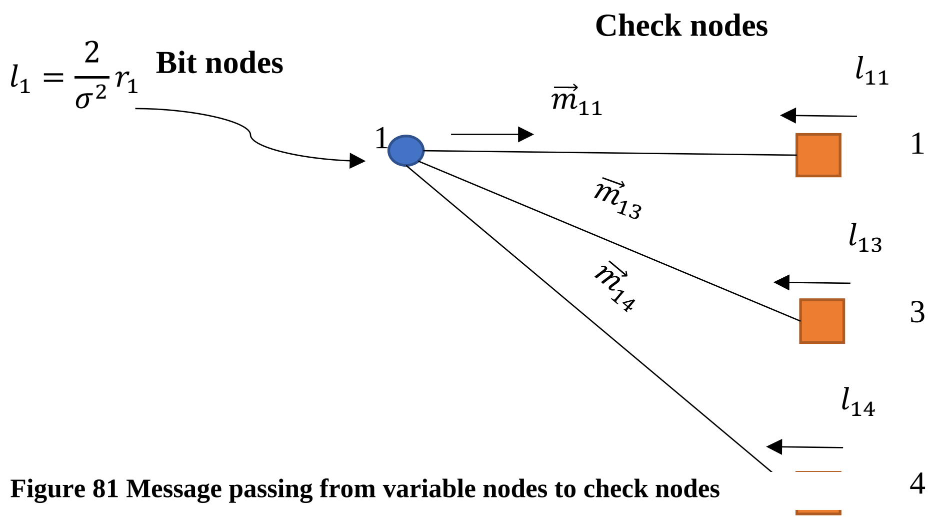


Figure 81 Message passing from variable nodes to check nodes

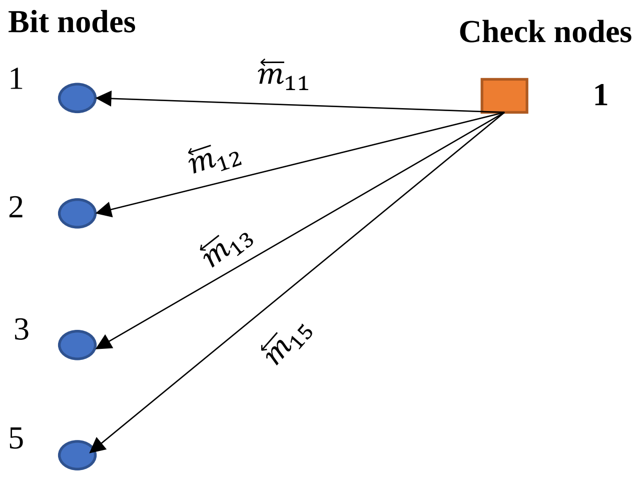


Figure 82 Message passing from check nodes to variable nodes

We get l_i from **step 1** then it gets passed to all the check nodes on the tanner graph to perform min-sum approximation algorithm then passes back the extrinsic value to the neighboring bit nodes, this process is done for all bit nodes in parallel, more details will be discussed in the following steps.

Step (2): Horizontal (Row) Operation:

It represents message passing from variable nodes to check nodes, we compute conditional LLRs for one bit.

For each row we do min-sum **SPC SISO decoding** which involves finding the two minimum values in the row and the sign (S) and by that we will have l_{ext}.

For each row:

1. Find the first minimum in each row of data

$$\mathrm{minimum}_1 = \min\left(|l_1|, |l_2|, \ldots\ldots., |l_n|\right)$$

2. Find the position of first minimum in the row

$$\mathrm{pos}_1 = \arg\min\left(|l_1|, |l_2|, |l_3|, \ldots\ldots, |l_n|\right)$$

3. Find the second minimum in the row

$$\mathrm{minimum}_2 = \min\left(|l_1|, |l_2|, \ldots, |l_{pos-1}|, |l_{pos+1}|, \ldots., |l_n|\right)$$

4. $|l_{ext,pos_1}| = \mathrm{minimum}_2$, $|l_{ext,i}| = \mathrm{minimum}_1$, where $i \neq pos$. (we set each value (except for zeroes we keep them as they're as they don't represent a relation between the message and the PC matrix).

Let P (Parity) =product of signs, then set a new sign for each entry

$$New\ sign_{(entry)} = old\ sign \times P.$$

Apply the former steps to our example:

$$For\ 1^{st} row:\ 1^{st}\ minimum = 0.2\ and\ 2^{nd} minimum = 0.3\ , P_{row1} = -1$$

$$For\ 2^{nd} row:\ 1^{st}\ minimum = 0.3\ and\ 2^{nd} minimum = 0.5\ , P_{row2} = 1$$

$$For\ 3^{rd} row:\ 1^{st}\ minimum = 0.1\ and\ 2^{nd} minimum = 0.3\ , P_{row3} = -1$$

$$For\ 4^{th}\ row:\ 1^{st}\ minimum = 0.2\ and\ 2^{nd} minimum = 0.6\ , P_{row4} = -1$$

First find the absolute two minima in the storage matrix and change the entries of the matrix for each row (note that each row is done in parallel):

$$\therefore L = \begin{bmatrix} 0.3 & -0.2 & 0.2 & 0 & 0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ 0.3 & -0.2 & 0 & -0.2 & 0 & 0 & -0.2 \\ 0.6 & 0 & 0.2 & 0 & 0.2 & 0.2 & -0.2 \end{bmatrix}$$

Second change the sign of all entries according to the calculated P:

$$\therefore L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

Step (3): Vertical (Column) Operation:

It represents message passing from check nodes to variable nodes, we compute consolidate conditional LLRs.

We find the sum for all entries plus the received signal values that came from the channel which represents discussed ***repetition decoding.***

For each column

- ❑ $Sum_j = r_j + sum\ of\ all\ entries\ in\ col_j\ (Total\ belief).$

$$: j = \{1 \ldots n-k\}$$

- ❑ $New\ entry = Sum_j - old\ entry_j\ .(extrinsic).$

$$\mathrm{r} = [\,0.2 \quad -0.3 \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1\,]$$

$$\therefore L = \begin{bmatrix} -0.3 & 0.2 & -0.2 & 0 & -0.2 & 0 & 0 \\ 0 & -0.5 & 0.3 & -0.3 & 0 & 0.3 & 0 \\ -0.3 & 0.2 & 0 & 0.2 & 0 & 0 & 0.2 \\ -0.6 & 0 & -0.2 & 0 & -0.2 & -0.2 & 0.2 \end{bmatrix}$$

First step: $\mathrm{Sum}_{1\mathrm{st}\,\mathrm{i}} = [\,-1 \quad -0.4 \quad 1.1 \quad -0.6 \quad 0.4 \quad 0.7 \quad -0.7\,]$

Second step: new entries (updated belief after 1st iteration):

$$\therefore L = \begin{bmatrix} -0.7 & -0.6 & 1.3 & 0 & 0.6 & 0 & 0 \\ 0 & 0.1 & 0.8 & -0.3 & 0 & 0.4 & 0 \\ -0.7 & -0.6 & 0 & -0.8 & 0 & 0 & -0.9 \\ -0.4 & 0 & 1.3 & 0 & 0.6 & 0.9 & -0.9 \end{bmatrix}$$

Then we can iterate on L as much as we want till we have a strong belief.

Step (4): Hard decision:

$$\widehat{l_\iota} = \begin{cases} 0, if\ sum_i > 0 \\ 1, if\ sum_i < 0 \end{cases}$$

Finally, we had implemented this model on MATLAB and compare our results with the Built-in function (nrLDPCDecode) in 5G tool box to check if our model is working correctly or not, and we got the following results:

BER of LDPC model vs BER of LDPC Built-in Function:

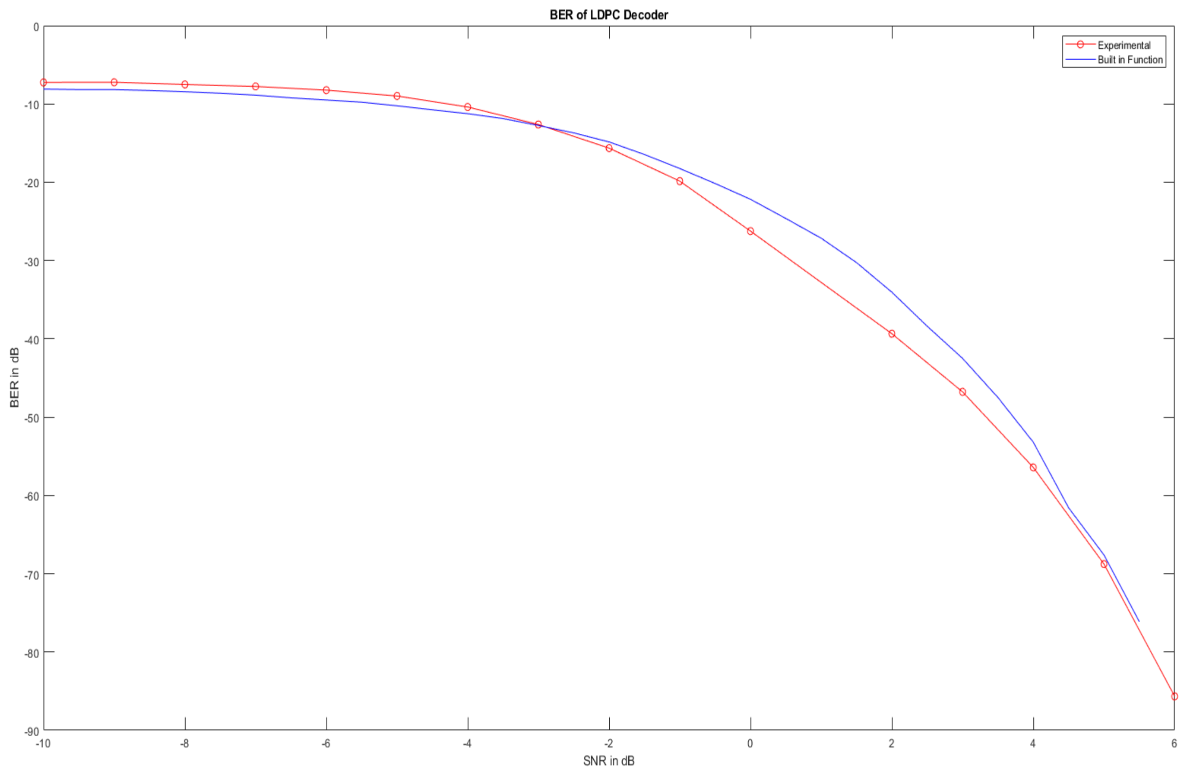


Figure 83 BER of LDPC Experimental vs Built-in function

As we see in the previous figure, our model behaves with the same manner as LDPC built-in function, which validate that our model is working properly.

Hardware Implementation

Fixed point representation

1 sign bit	3-bit integer	12-bit fraction part

Three bits were selected to represent the integer part so we could select a practical value of standard deviation of noise to calculate the intrinsic part of the input message bits, we have found that 3 bits is the most suitable number of bits.

Punctured LDPC Codes

At high code word length, we adopt a puncturing strategy which means getting rid of redundancy bits that could be easily retrieved from other message bits, that happens to maintain a good performance and increase code rate consequently maximize the throughput.

At the receiver, the belief propagation algorithm that operates on the Tanner graph representing the code must be adapted to manage the missing bits.

According to standards, first $2Z_c$bits are punctured in LDPC while transmitting, since they can be easily retrieved from other parity and message bits; in addition, it reduces the number of transmitted bits over channel. For example, if we are transmitting 3000 bits with code rate 1/5, 15,000 bits will be loaded over channel, however if we punctured the first $2Z_c$bits, assuming $Z_c = 384$, we will only transmit 11,160 encoded bits.

In conclusion, you will find that removing these bits will not affect your bit error rate performance; in addition, it reduces the number of transmitted bits over channel which needs less bandwidth.

Criteria of selecting punctured bits

As mentioned above, the proposed LDPC decoder design works on BG2 with expansion factor $Z_c = 384$ and code rate= $\frac{1}{5}$, that means we will puncture 768 bits from our message bits.

Punctured bits ($2 \times Z_c$)	Transmitted code word bits($50 \times Z_c$)

Retrieval of punctured data

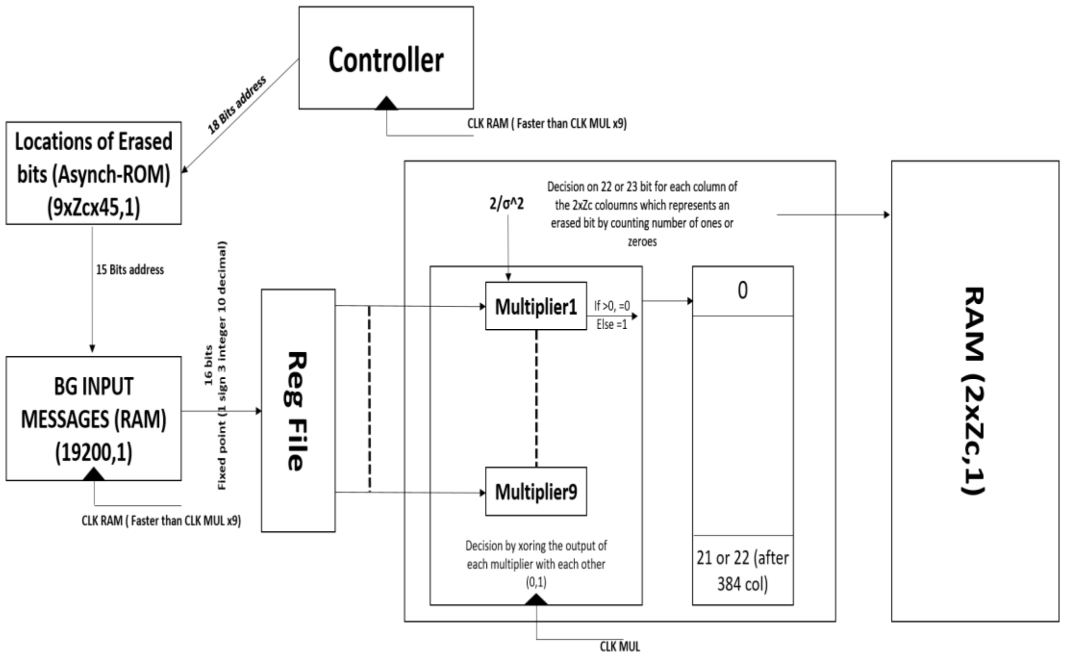

Controller
18 Bits address
CLK RAM (Faster than CLK MUL x9)
Locations of Erased bits (Asynch-ROM) (9xZcx45,1)
15 Bits address
BG INPUT MESSAGES (RAM) (19200,1)
CLK RAM (Faster than CLK MUL x9)
16 bits
Fixed point (1 sign 3 integer 10 decimal)
Reg File
2/σ^2
Decision on 22 or 23 bit for each column of the 2xZc coloumns which represents an erased bit by counting number of ones or zeroes
Multiplier1
If >0, =0
Else =1
Multiplier9
Decision by xoring the output of each multiplier with each other (0,1)
0
21 or 22 (after 384 col)
CLK MUL
RAM (2xZc,1)

Figure 84 Punctured bits retrieval block diagram

As we said before, bits within the same row have direct relation that could be executed using SPC. For example, $r_{11} + r_{13} + r_{19} = 0$, where r_{13} is bit at the first row and third column.

By looking at the previous figure, we can find that:

1. We load the locations of ones (message bit or parity bit has direct relation with the erased bit through SPC equation) in each row of the erased bits from the parity check matrix.
2. Using a controller, we select input message bits that are located within the loaded locations.
3. These message bits are then multiplied by $\frac{2}{\sigma^2}$ in order to calculate their intrinsic values.
4. Hard decision is applied for these message bits:
   - ✓ If $r_i > 0\ then\ r_i = 0$
   - ✓ If $r_i < 0\ then\ r_i = 1$
5. Calculate the XOR equation (SPC condition) to retrieve the erased bit.
6. Since that each erased bit has multiple edges (multiple check nodes), we use Repetition decoder to determine the final bit result based on the dominating estimation (Ones or zeros).

   Note:

   - ✓ We have found that each erased bit forms an equation with maximum 9 message bits within the same row, so we have implemented 9 multipliers to simultaneously calculate intrinsic values for the input message bits with a $frequency\ = \frac{1}{9}$ RAM frequency, so we can employ time sharing concept in our design.
   - ✓ The punctured bits here are in their form of final decision, there's no need for the min-sum algorithm to be done on them.
7. The bits are then saved to a RAM to be concatenated to the output of the LDPC decoder.

DE-Segmentation:

System logic:

At transmitter, we have segmentation which is divides the large input data into small segments so low-density parity check (LDPC) could take it and make a processing on the data, each segment has its own cyclic redundancy check (CRC), At receiver, First, we should check if there was an error and then combine these segments into one large portion of data to be ready for the next block.

Standard specification:

Segmentation divides data of length 14880-bits (pure data bits + CRC bits) into 4 segments each 3840-bits so we have 15360-bits in total (pure data bits + CRC bits + filler bits), according to 3GPP TS 38.212 version 15.5.0 release 15 we should use LDPC with base graph number 2 which has 96 filler bits and use G_{CRC24B}. So, all we need in receiver is to return the data back to the original form in one segment without filler bits and CRC bits.

According to 38.212 version 15.5.0 release 15:

Let K_{cb} the maximum code block size, B is the number of input bits, L is the number of CRC-bits, so if B > K_{cb} we will make segmentation.

For LDPC base graph 2, $K_{cb} = 3840$

Total number of code blocks C is determined by:

```
If $B \leq K_{cb}$
    $L = 0$
    Number of code blocks: $C = 1$
    $B^{'} = B$                    // $B^{'}$: total number of bits
Else
    L=24
```

Number of code blocks: $C = \lceil B/(K_{cb} - L\rceil$

$B^{'} = B + C * L$

End if

After this we add filler bits to complete 3840 bits each code block that will be input to low-density parity check block, also at receiver LDPC block will give de-segmentation block 4 code blocks each code block have 3480-bit.

Operation and flow of design:

Data input coming from previous block should pass to filler bits removal and CRC'24B check block so filler bits and CRC parity bits will be removed. Then these two flowcharts will demonstrate the rest of the code.

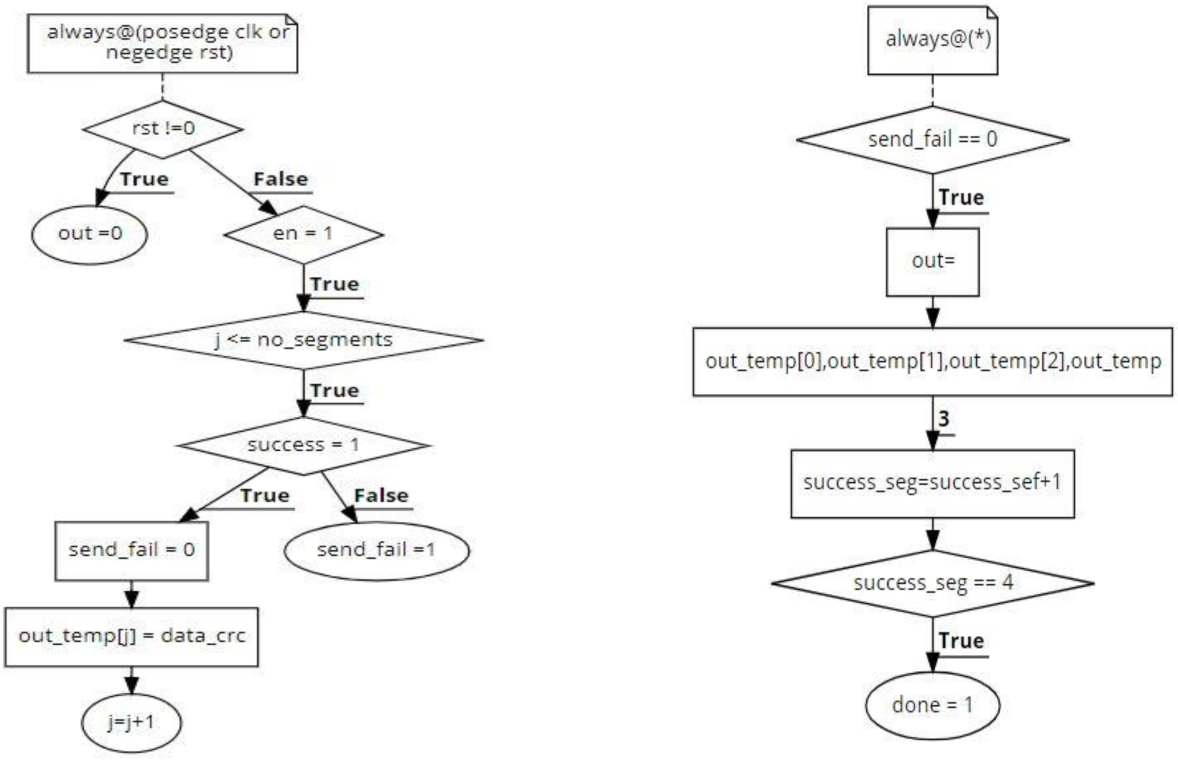
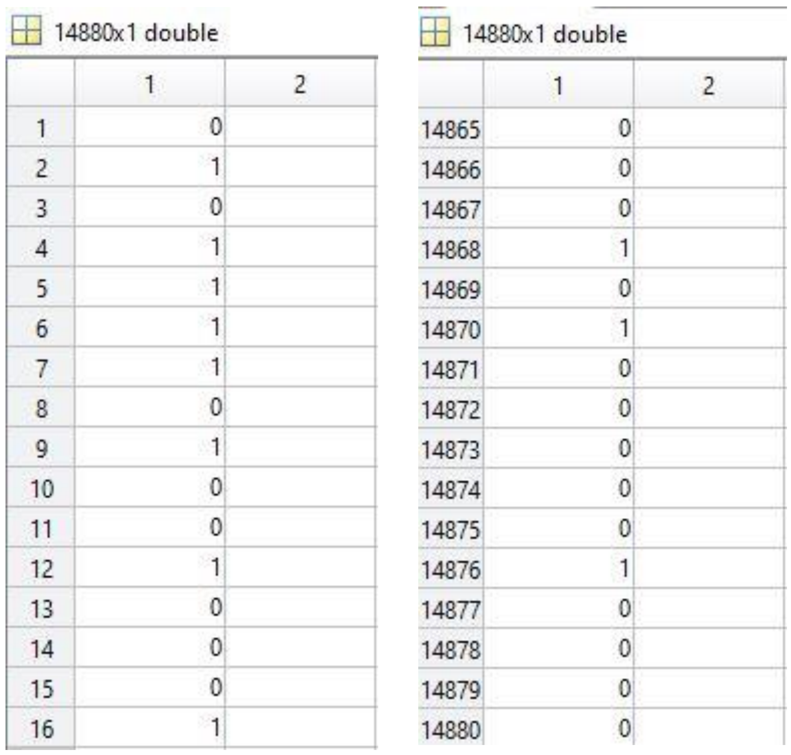


Figure 85: Flowchart of DE-Segmentation

High level simulation:


14880x1 double

	1	2
1	0	
2	1	
3	0	
4	1	
5	1	
6	1	
7	1	
8	0	
9	1	
10	0	
11	0	
12	1	
13	0	
14	0	
15	0	
16	1	

14880x1 double

	1	2
14865	0	
14866	0	
14867	0	
14868	1	
14869	0	
14870	1	
14871	0	
14872	0	
14873	0	
14874	0	
14875	0	
14876	1	
14877	0	
14878	0	
14879	0	
14880	0	

Figure 86: First and last bits of de-segmentation on MATLAB

Hardware implementation of DE-segmentation:

First, we removed the last 96-bit of the data as they are filler bits and make pass it through CRC 24B, if CRC block worked successfully without any error detected we started de-segmentation process.

According to 38.212 version 15.5.0 release 15, we have 4 code blocks each one has 3744-bits without filler bits (data + CRC parity bits) after CRC 24B last 24-bits removed so now we have pure data with 3720-bits, so the output of de-segmentation will be 14880-bits.

Block interface:

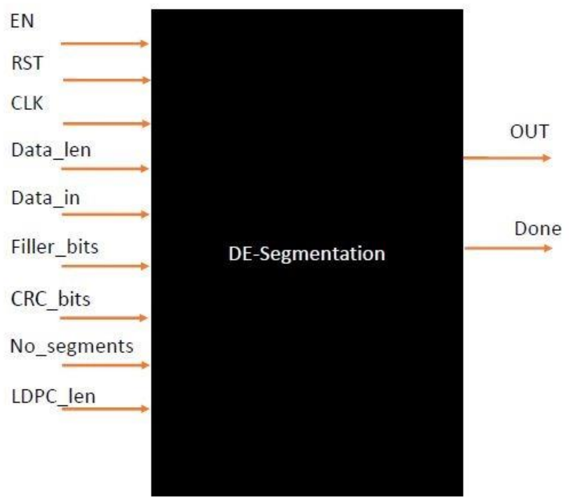


Figure 87:DE-segmentation block interface

Inputs & Outputs Ports:

Port name	Direction	Width	description
EN	input	1-bit	Enables the block
RST	Input	1-bit	Reset the block
clk	input	1-bit	Clock signal of the block
Data_len	input	12-bit	The length of the data input. (3840)
Data_in	input	1-bit 3840-bit	Input data to DE-segmentation
Filler_bits	input	7-bit	Filler bits added to data stream in segmentation. (96)
CRC_bits	Input	5-bit	CRC bits added to data stream. (24)
No_segments	input	2-bit	Number of segments comes from LDPC. (4)
LDPC_len	input	12-bit	The length of output data of LDPC. (3840)
out	output	14880-bit	Output of the DE-Segmentation Block.
done	output	1-bit	Flag to verify that DE-segmentation Block gives off all of the output.

Block diagram:

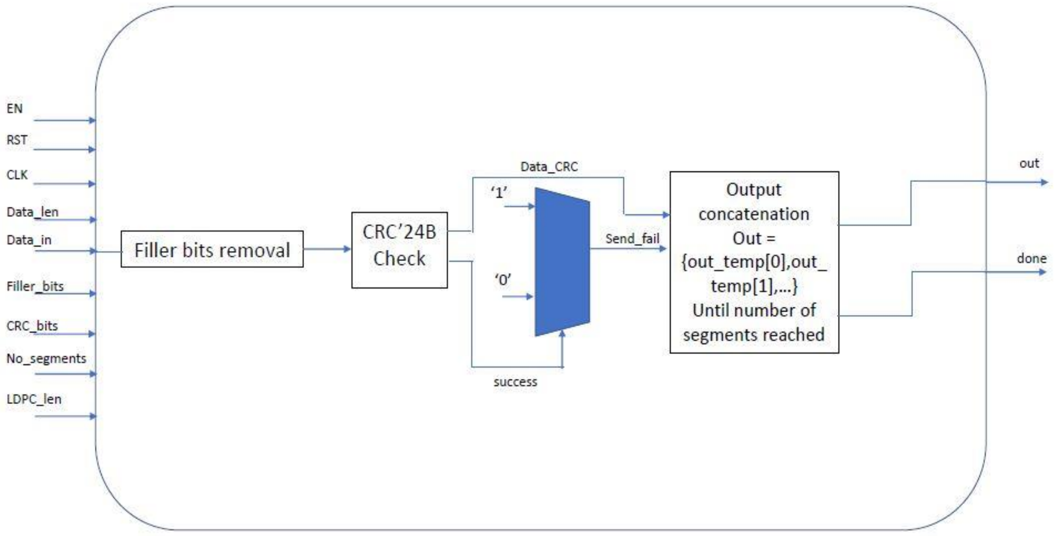
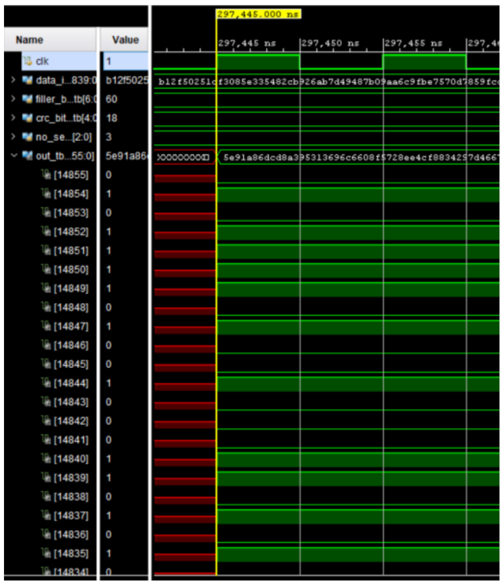
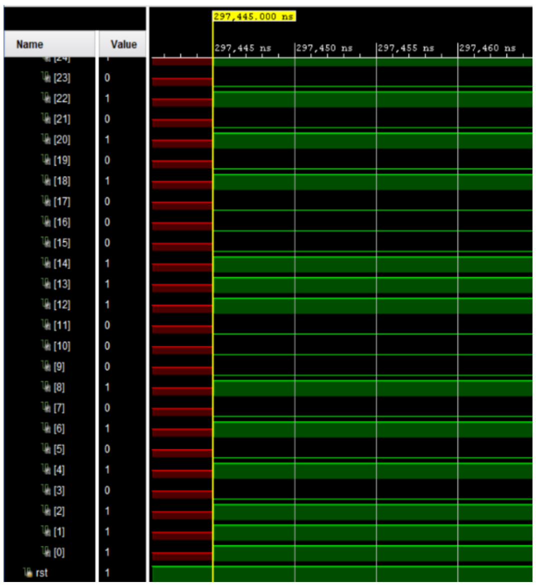


Figure 88: block diagram of de-segmentation

Hardware simulation:


297,445.000 ns
297,445 ns
297,450 ns
297,455 ns
b12f50251cf3085e335482cb926ab7d49487b09aa6c9fbe7570d7859fc
XXXXXXXX
5e91a86dcd8a395313696c6608f5728ee4cf8834257d4667

Name	Value
clk	1
data_i...839:0	b12f5025
filler_b...tb[6:0	60
crc_bit...tb[4:0	18
no_se...[2:0]	3
out_tb...55:0]	5e91a86
[14855]	0
[14854]	1
[14853]	0
[14852]	1
[14851]	1
[14850]	1
[14849]	1
[14848]	0
[14847]	1
[14846]	0
[14845]	0
[14844]	1
[14843]	0
[14842]	0
[14841]	0
[14840]	1
[14839]	1
[14838]	0
[14837]	1
[14836]	0
[14835]	1
[14834]	0

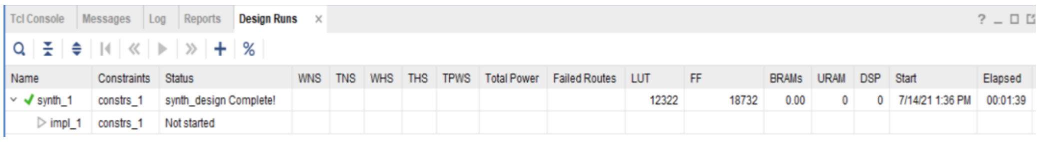


Name	Value
[23]	0
[22]	1
[21]	0
[20]	1
[19]	0
[18]	1
[17]	0
[16]	0
[15]	0
[14]	1
[13]	1
[12]	1
[11]	0
[10]	0
[9]	0
[8]	1
[7]	0
[6]	1
[5]	0
[4]	1
[3]	0
[2]	1
[1]	1
[0]	1
rst	1

Figure 89: first and last bit of de-segmentation on modalism

Synthesis results:

<u>Synthesis validation:</u>



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed
✓ synth_1	constrs_1	synth_design Complete!								12322	18732	0.00	0	0	7/14/21 1:36 PM	00:01:39
impl_1	constrs_1	Not started														

Figure 90:synthesis validation of DE-segmentation

Elaborated design:

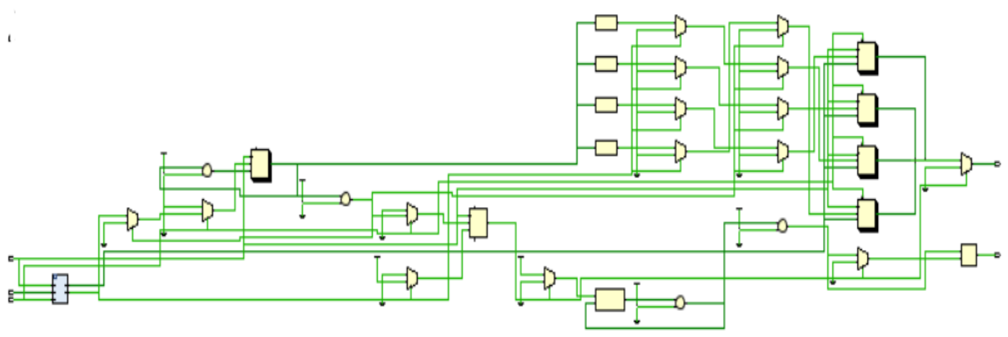
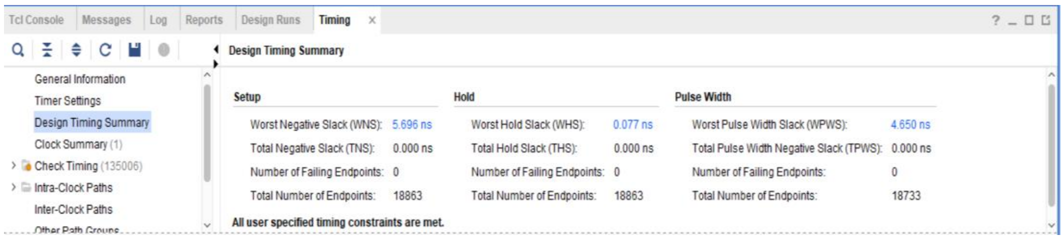


Figure 91: elaborated de-sign of DE-segmentation

Timing report:



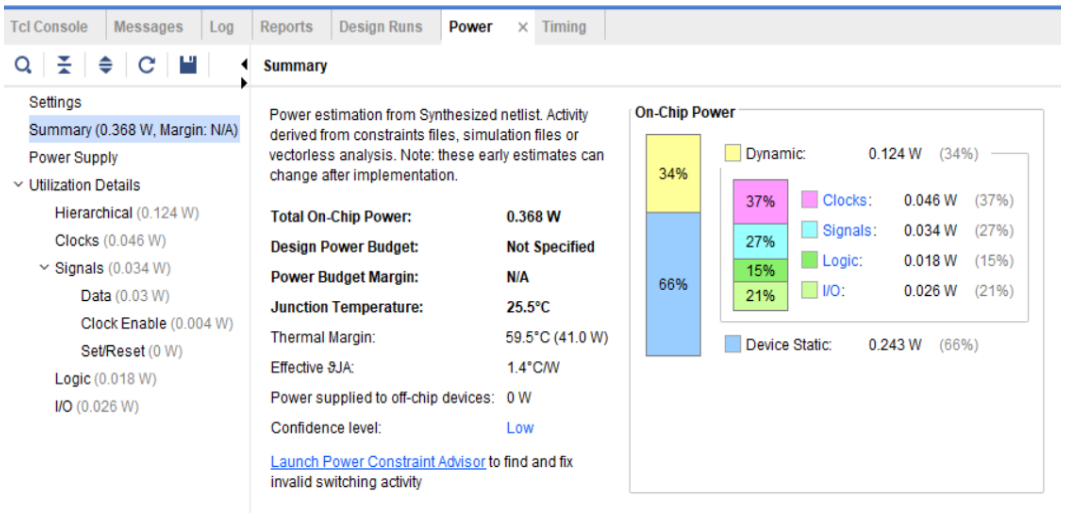
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.696 ns	Worst Hold Slack (WHS):	0.077 ns	Worst Pulse Width Slack (WPWS):	4.650 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	18863	Total Number of Endpoints:	18863	Total Number of Endpoints:	18733

All user specified timing constraints are met.

Figure 92: timing report of DE-segmentation

Power report:



Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.368 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (41.0 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.124 W	(34%)	34%
Clocks:	0.046 W	(37%)	37%
Signals:	0.034 W	(27%)	27%
Logic:	0.018 W	(15%)	15%
I/O:	0.026 W	(21%)	21%
Device Static:	0.243 W	(66%)	66%

Figure 93: power consumption by DE-segmentation

Analysis of synthesis result:

This design is worked at 100 MHz with large positive slack which enable this block to operate at higher frequency. This design consumes 0.368 watts power on chip.

CRC removal:

System logic:

Our society greatly depends on mobile technologies. As wirelessly connected devices take over the control of the electricity in our homes, so we must guarantee the security of the network. Method used to implement the CRC in the 5G system is using the **LFSR** implementation in both encoding and decoding is required to have re-programmable connection.

We use a certain polynomial generator is used at both transmitter and receiver to add parity bits at the end of the data entered then this data will enter the receiver where the CRC part at the end of the data will be divided with the same polynomial generator used at the transmitter if the remainder is zero then data is passed successfully without any errors.

Standards Specifications:

There are many polynomials used for the cyclic redundancy check, the used polynomial will depend on the length of the data entering the system .each polynomial will add a certain number of parity check at the end of the segment, at the receiver there will be CRC removal according to whether data reached Rx correctly or not.

Denote the input bits to the CRC computation by $a0$, $a1$, $a2$, $a3$..., $aA-1$, and the parity bits by $p0$, $p1$, $p2$, $p3$..., $pL-1$, where A is the size of the input sequence and L is the number of parity bits. The parity bits are generated by one of the following cyclic generator polynomials:

G_{CRC24A} (D) = [D24 + D23 + D18 + D17 + D14 + D11 + D10 + D7 + D6 + D5 + D4 + D3 + D + 1] for a CRC length L = 24

G_{CRC24B} (D) = [D24 + D23 + D6 + D5 + D + 1] for a CRC length L = 24

G_{CRC24C} (D) = [D24 + D23 + D21 + D20 + D17 + D15 + D13 + D12 + D8 + D4 + D2 + D + 1] for a CRC length L = 24

G_{CRC16} (D) = [D16 + D12 + D5 + 1] for a CRC length L = 16

G_{CRC11} (D) = [D11 + D10 + D9 + D5 + 1] for a CRC length L = 11

G_{CRC6} (D) = [D6 + D5 + 1] for a CRC length L = 6

The bits after CRC attachment are denoted by $b0$, $b1$, $b2$, $b3$ ..., b_{B-1}, where $B = A + L$. The relation between a_k and bk is:

$b_k = a_k$ for k = 0,1, 2..., A−1

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

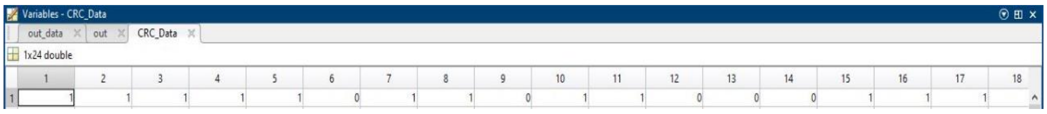
We choose the working polynomial in CRC unit as follow:

G_{CRC24A} for length of A ≥ 3824 bits “which will be used in our system”

G_{CRC16} for otherwise

and G_{CRC24B} in the segmentation unit as we will see next. Other polynomials are not used as we work in PDSCH.

High level simulation results:

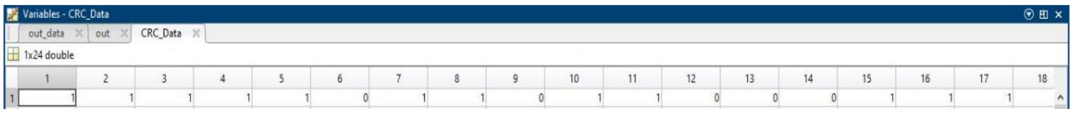

Variables - CRC_Data

out_data | out | CRC_Data

1x24 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	1	1	1	0	1	1	0	1	1	0	0	0	1	1	1	

Figure 94: built in function result


Variables - CRC_Data

out_data | out | CRC_Data

1x24 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	1	1	1	1	0	1	1	0	1	1	0	0	0	1	1	1	

Figure 95: our code results

As shown, both results are the same so we will go to the next stage in the flow where we will convert the high-level design code into HDL using Verilog.

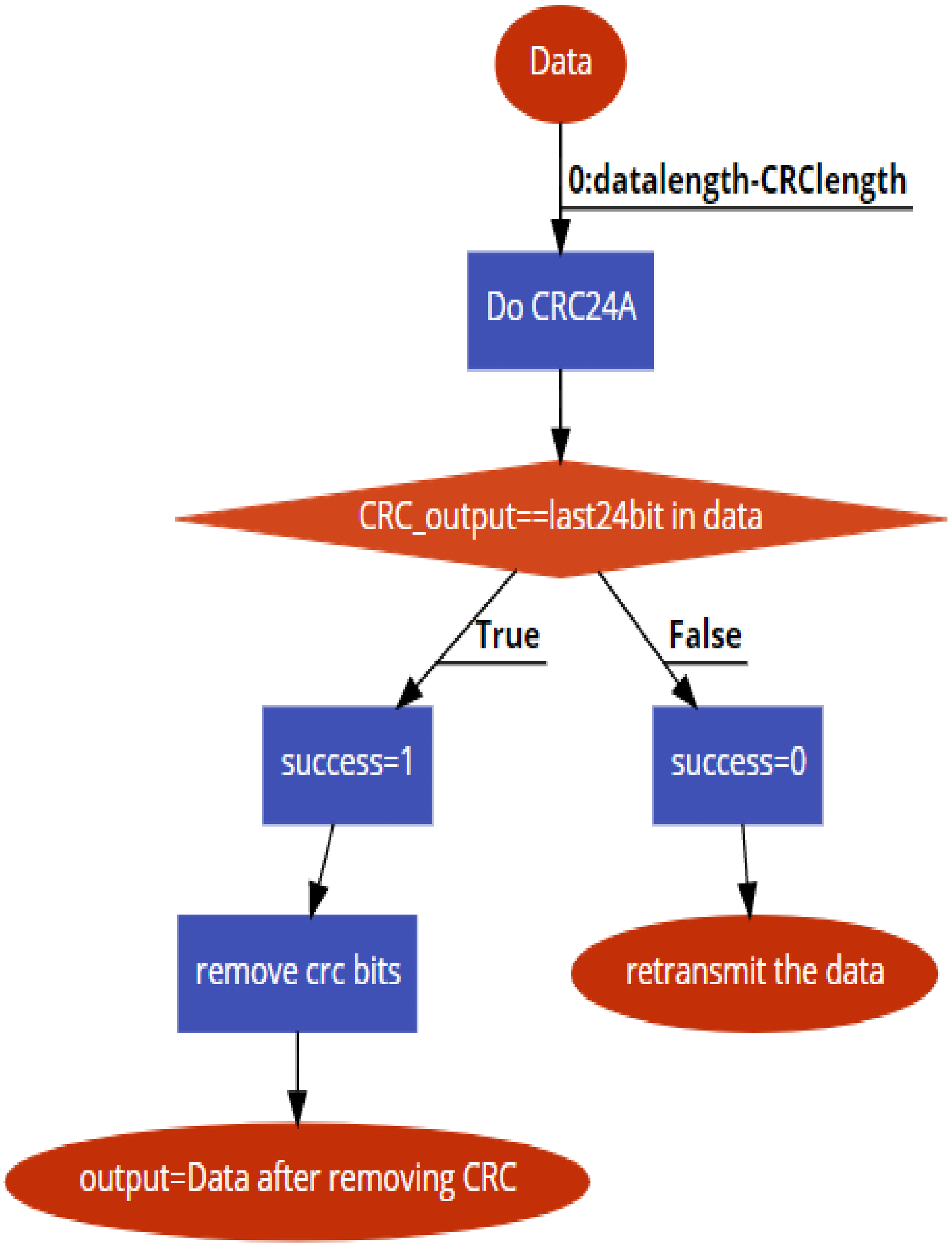


Figure 96:CRC flow chart

Block interface:

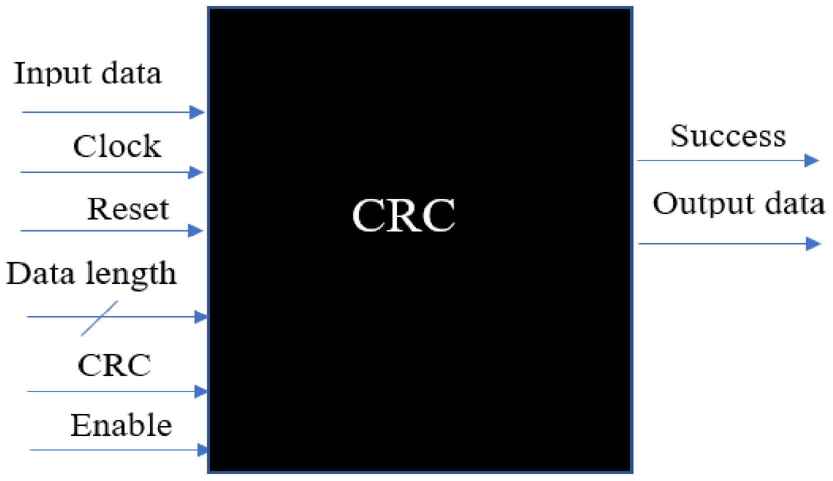


Figure 97:Block Interface of CRC

INPUT AND OUTPUT PORTS:

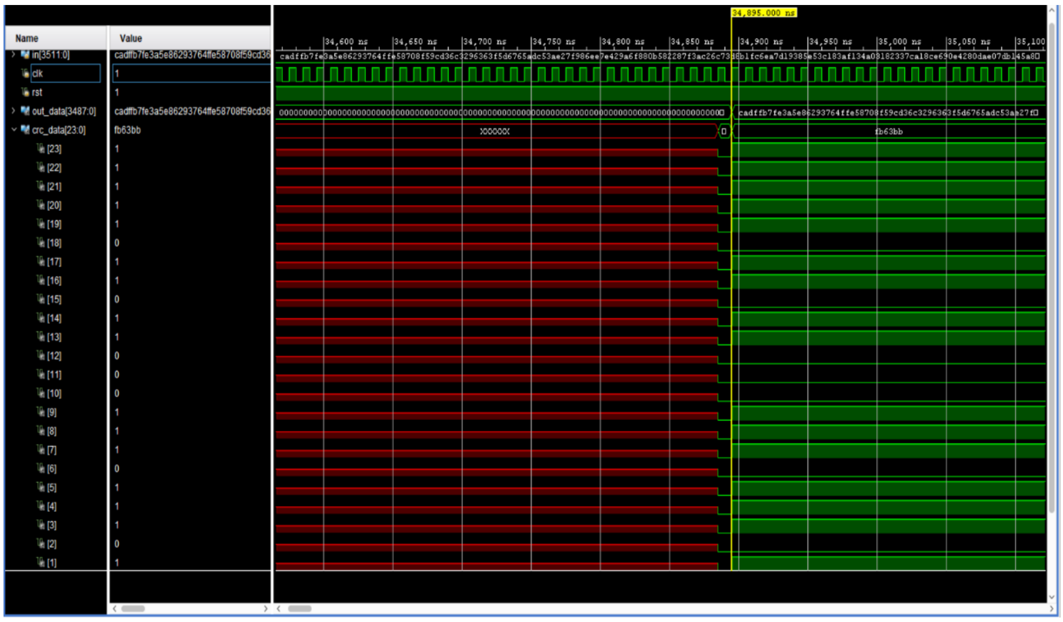
Port Name	Direction	Width	Description
Input Data	input	3512 bits	The input data with CRC added to it.
Clock	input	1 bit	Operating clock to write into the block.
Reset	input	1 bit	Resetting the block.
Enable	input	1 bit	Block is enabled when data is available on port after DE-segmentation
success	output	1 bit	Signals indicate whether data reached Rx is right or not
Data out	output	3488 bits	The output data stream after crc removal.

Operation

Data of each segment comes from the transmitter with the CRC parity bits added at the end of it , firstly we separate the last 24 bits as we are using CRC24B polynomial then we will divide these bits by the polynomial generated at the receiver which is the same generator used at the transmitter, the division is done by implementation of the linear feedback shift register and doing XOR, if the remainder of the division is zero then we will do CRC removal and out the remaining data, if there's an error and the remainder isn't zero then we will retransmit the data

In CRC24A we do the same operation but with using different polynomial as the following: $G_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^{7} + D^{6} + D^{5} + D^{4} + D^{3} + D + 1]$, we use CRC24A at the end of the chain after doing de segmentation & CRC24B

Hardware simulation:



Name	Value
in[3511:0]	cadffb7fe3a5e86293764ffe58708f59cd36
clk	1
rst	1
out_data[3487:0]	cadffb7fe3a5e86293764ffe58708f59cd36
crc_data[23:0]	fb63bb
[23]	1
[22]	1
[21]	1
[20]	1
[19]	1
[18]	0
[17]	1
[16]	1
[15]	0
[14]	1
[13]	1
[12]	0
[11]	0
[10]	0
[9]	1
[8]	1
[7]	1
[6]	0
[5]	1
[4]	1
[3]	1
[2]	0
[1]	1

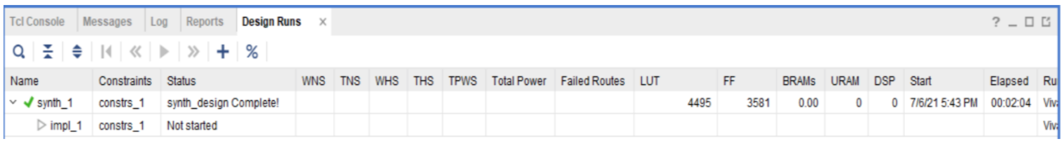
Figure 98: Hardware behavioral simulation of CRC

The CRC block take a time to enter data in the polynomial generator to calculate the CRC polynomial which comes **as shown in the above figure**. It is also clear that the output of the RTL design is same to that has been calculated in MATLAB

The output data is after the CRC removal in case that the CRC generated at the receiver is same to that calculated at the transmitter

Synthesis results:

Synthesis validation:



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Ru
✓ synth_1	constrs_1	synth_design Complete!								4495	3581	0.00	0	0	7/6/21 5:43 PM	00:02:04	Viv
▷ impl_1	constrs_1	Not started															Viv

Figure 99: synthesis results

Elaboration results:

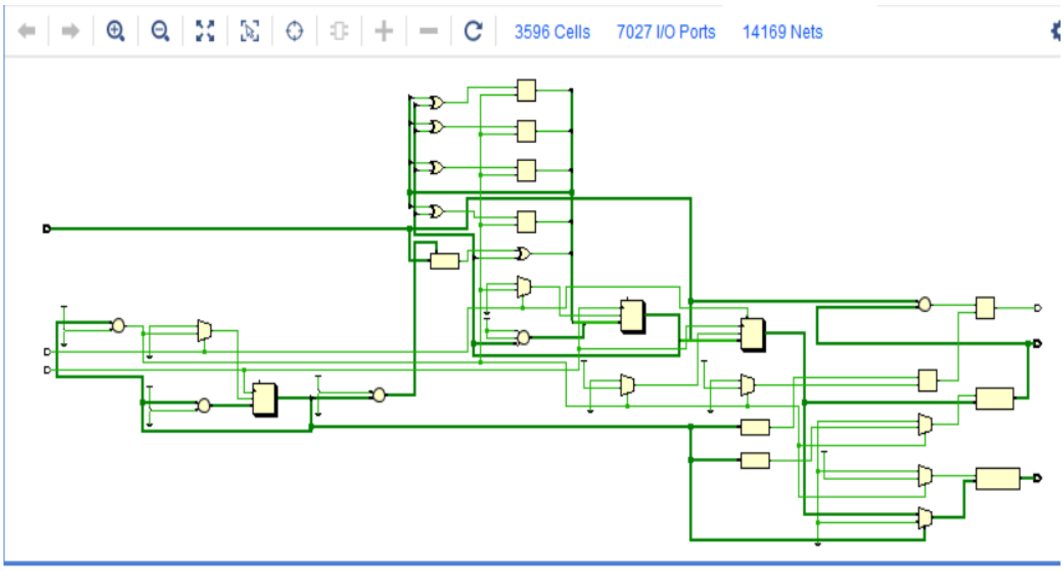
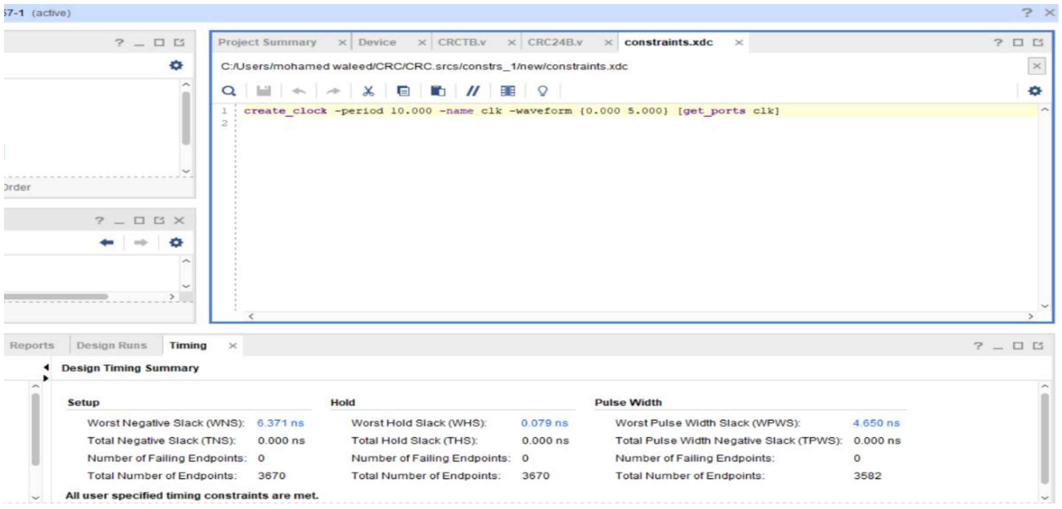


Figure 100: elaboration of the design

Timing report:



Project Summary | Device | CRCTB.v | CRC24B.v | constraints.xdc

C:/Users/mohamed waleed/CRC/CRC.srcs/constrs_1/new/constraints.xdc

```
create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]
```

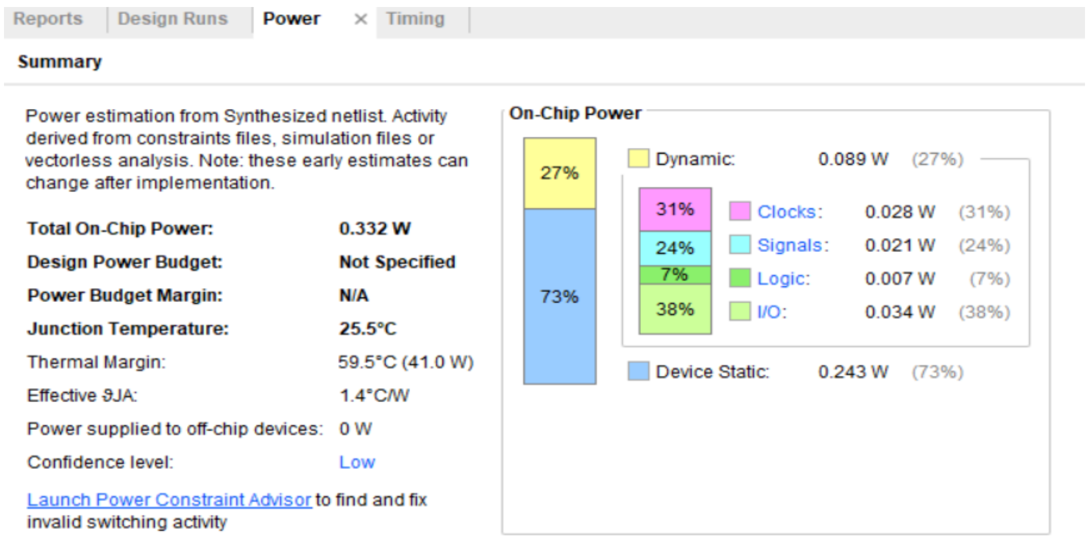
Reports | Design Runs | Timing

Design Timing Summary

```
Setup                                          Hold                                        Pulse Width
Worst Negative Slack (WNS):      6.371 ns      Worst Hold Slack (WHS):          0.079 ns   Worst Pulse Width Slack (WPWS):              4.650 ns
Total Negative Slack (TNS):      0.000 ns      Total Hold Slack (THS):          0.000 ns   Total Pulse Width Negative Slack (TPWS):     0.000 ns
Number of Failing Endpoints:     0             Number of Failing Endpoints:     0          Number of Failing Endpoints:                 0
Total Number of Endpoints:       3670          Total Number of Endpoints:       3670       Total Number of Endpoints:                   3582
All user specified timing constraints are met.
```

Figure 101: Timing report of CRC block

<u>Power report:</u>



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.332 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (41.0 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

Dynamic:	0.089 W	(27%)
Clocks:	0.028 W	(31%)
Signals:	0.021 W	(24%)
Logic:	0.007 W	(7%)
I/O:	0.034 W	(38%)
Device Static:	0.243 W	(73%)

Figure 102: Power consumption in CRC block

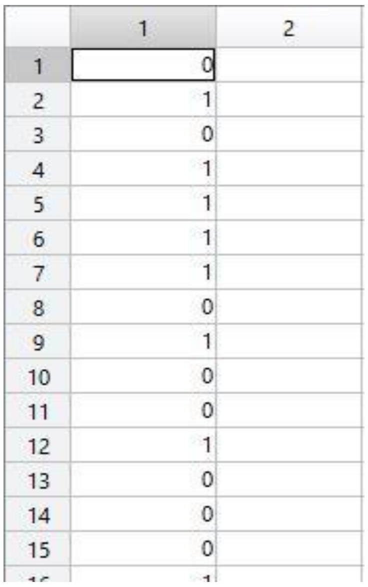
Analysis of synthesis results:

It is obvious that our design is synthesizable having 4495 LUT, 3581 FF this design could work at frequency more than 100MHZ as it met the time constraints with a large margin given that the clock period used was 10ns (in the constraint file). Our design has on chip power= 0.332watt.

VIII. Integration of the whole RX chain

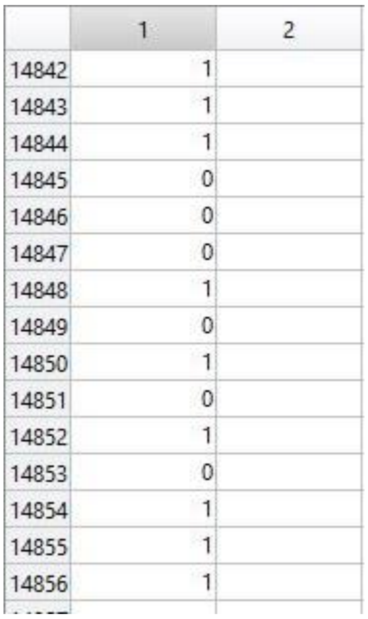
DE segmentation & CRC24A

High level simulation:



	1	2
1	0	
2	1	
3	0	
4	1	
5	1	
6	1	
7	1	
8	0	
9	1	
10	0	
11	0	
12	1	
13	0	
14	0	
15	0	

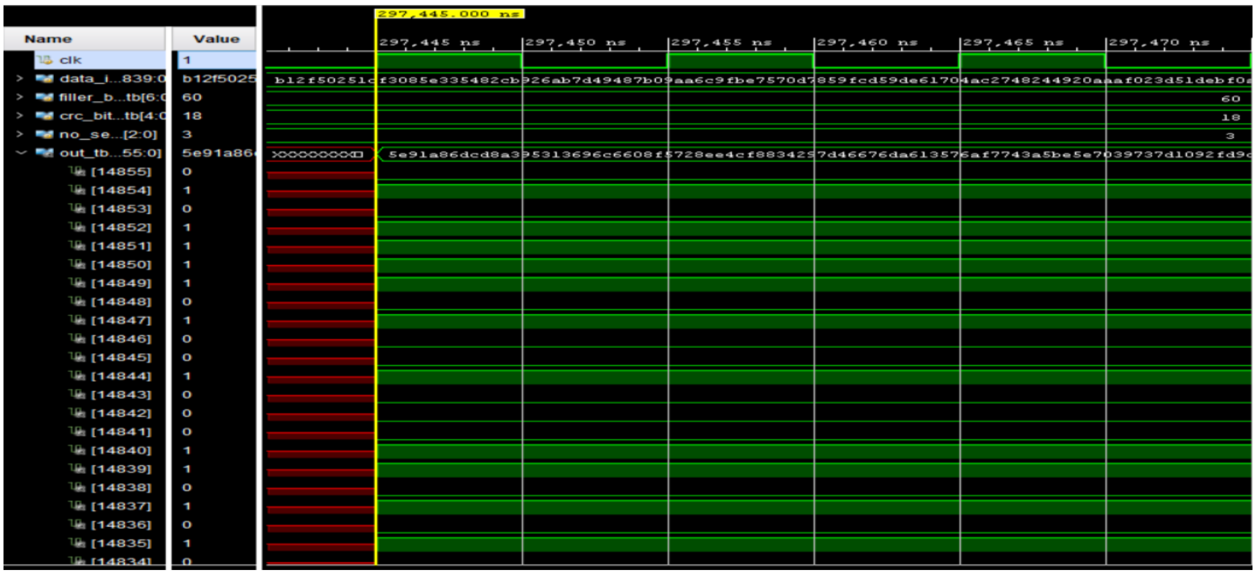
Figure 104:first 24 bits



	1	2
14842	1	
14843	1	
14844	1	
14845	0	
14846	0	
14847	0	
14848	1	
14849	0	
14850	1	
14851	0	
14852	1	
14853	0	
14854	1	
14855	1	
14856	1	

Figure 103:last 24 bits

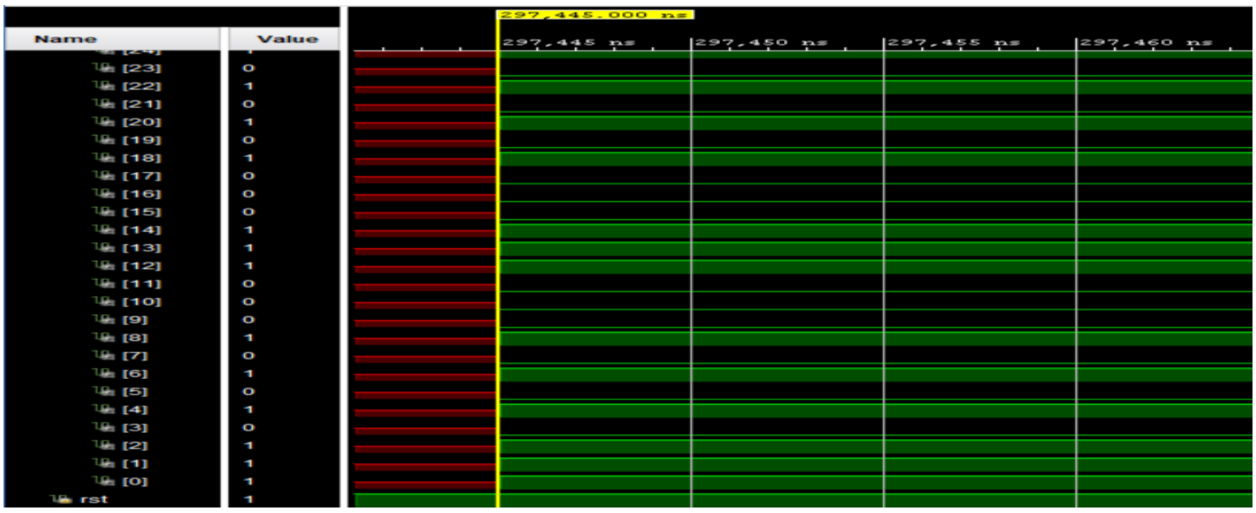
HDL simulation:



Name	Value
clk	1
data_i...839:0	b12f5025
filler_b...tb[6:0	60
crc_bit...tb[4:0	18
no_se...[2:0]	3
out_tb...55:0]	5e91a86
[14855]	0
[14854]	1
[14853]	0
[14852]	1
[14851]	1
[14850]	1
[14849]	1
[14848]	0
[14847]	1
[14846]	0
[14845]	0
[14844]	1
[14843]	0
[14842]	0
[14841]	0
[14840]	1
[14839]	1
[14838]	0
[14837]	1
[14836]	0
[14835]	1
[14834]	0

297,445.000 ns
297,445 ns
297,450 ns
297,455 ns
297,460 ns
297,465 ns
297,470 ns
b12f50251cf3085e335482cb926ab7d49487b09aa6c9fbe7570d7859fcd59de61704ac2748244920aaaf023d51debf0a
60
18
3
XXXXXXXX
5e91a86dcd8a395313696c6608f5728ee4cf8834257d46676da613576af7743a5be5e7039737d1092fd9c

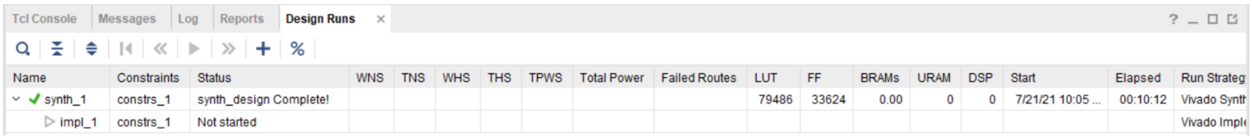
Figure 105:First 24 output bits from integration of de segmentation & CRC24A



Name	Value
[23]	0
[22]	1
[21]	0
[20]	1
[19]	0
[18]	1
[17]	0
[16]	0
[15]	0
[14]	1
[13]	1
[12]	1
[11]	0
[10]	0
[9]	0
[8]	1
[7]	0
[6]	1
[5]	0
[4]	1
[3]	0
[2]	1
[1]	1
[0]	1
rst	1

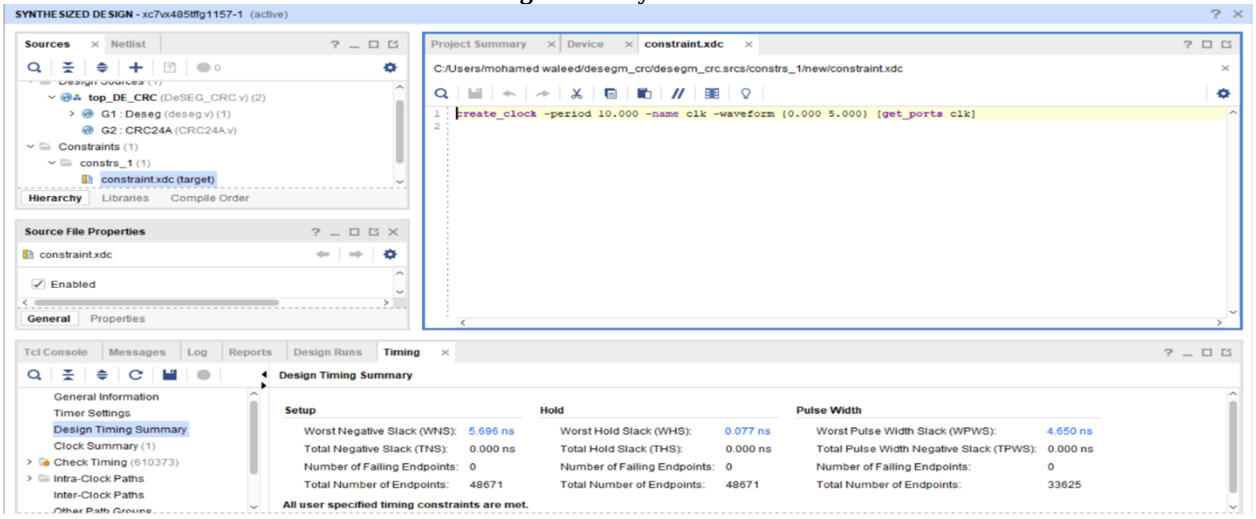
Figure 106:Last 24 output bits from integration of de segmentation & CRC24A

Synthesis results:



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAMs	URAM	DSP	Start	Elapsed	Run Strateg
✓ synth_1	constrs_1	synth_design Complete!								79486	33624	0.00	0	0	7/21/21 10:05 ...	00:10:12	Vivado Synth
impl_1	constrs_1	Not started															Vivado Impl

Figure 107:synthesis results



SYNTHESIZED DESIGN - xc7vx485tffg1157-1 (active)

```
top_DE_CRC (DeSEG_CRC.v) (2)
  G1 : Deseg (deseg.v) (1)
  G2 : CRC24A (CRC24A.v)
Constraints (1)
  constrs_1 (1)
    constraint.xdc (target)
```

C:/Users/mohamed waleed/desegm_crc/desegm_crc.srcs/constrs_1/new/constraint.xdc

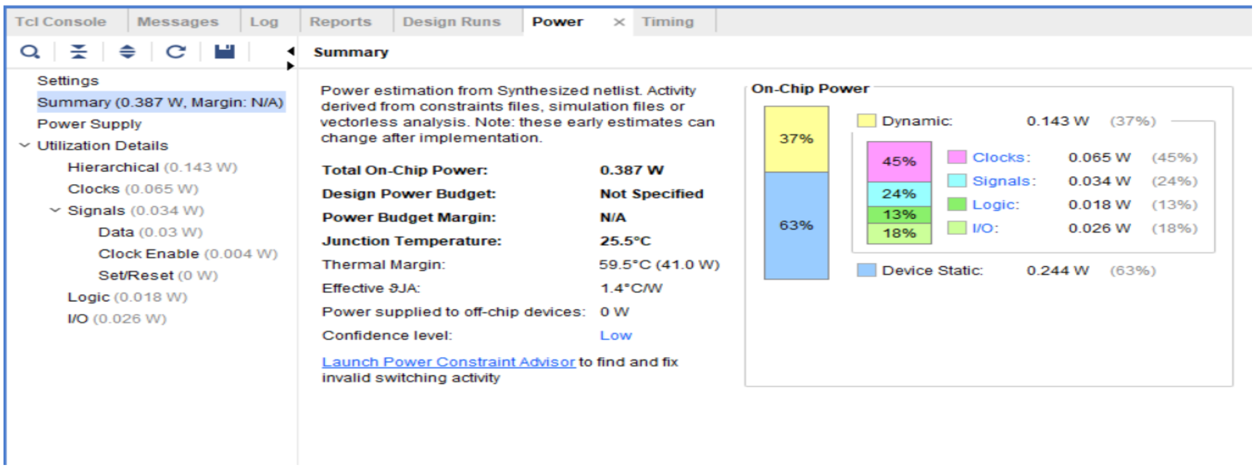
```
create_clock -period 10.000 -name clk -waveform {0.000 5.000} [get_ports clk]
```

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	5.696 ns	Worst Hold Slack (WHS):	0.077 ns	Worst Pulse Width Slack (WPWS):	4.650 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	48671	Total Number of Endpoints:	48671	Total Number of Endpoints:	33625

All user specified timing constraints are met.

Figure 108:timing report


Tcl Console | Messages | Log | Reports | Design Runs | Power | Timing

Settings
Summary (0.387 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (0.143 W)
Clocks (0.065 W)
Signals (0.034 W)
Data (0.03 W)
Clock Enable (0.004 W)
Set/Reset (0 W)
Logic (0.018 W)
I/O (0.026 W)

Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.387 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (41.0 W)
Effective ϑJA:	1.4°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.143 W	(37%)
Clocks:	0.065 W	(45%)
Signals:	0.034 W	(24%)
Logic:	0.018 W	(13%)
I/O:	0.026 W	(18%)
Device Static:	0.244 W	(63%)

Figure 109:power results

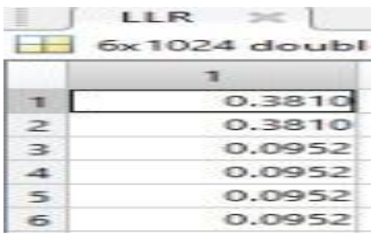
Comments:

- It's clear that the MATLAB simulations are same as the Verilog simulations where the error is checked and the 24 CRC bits are removed successfully
- The integrated design of the two blocks is synthesizable working at 100 MHZ with 5.695 ns positive slack so it could be work up to 200 MHZ
- The total power on chip for design is 0.387 watt

FFT & DE mapper

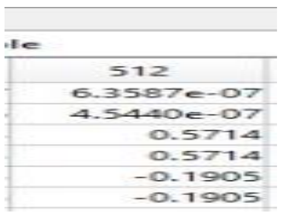
- In the integration of the FFT & DEMAPPER blocks we need to add fixed point matching module as the fixed-point representation input of DEMAPPER block is 1 bit for sign ,2 for integer, 14 for decimal part while the output of FFT is represented by 1 bit for integer so we do fixed-point matching between the two blocks
- FFT block output is two points in serial due to the butterfly algorithm at each clock after FFT calculations are done so we will have 12 bits output from the DEMAPPER block where each point has 6 bits output

High Level simulation:


LLR
6x1024 double

	1
1	0.3810
2	0.3810
3	0.0952
4	0.0952
5	0.0952
6	0.0952

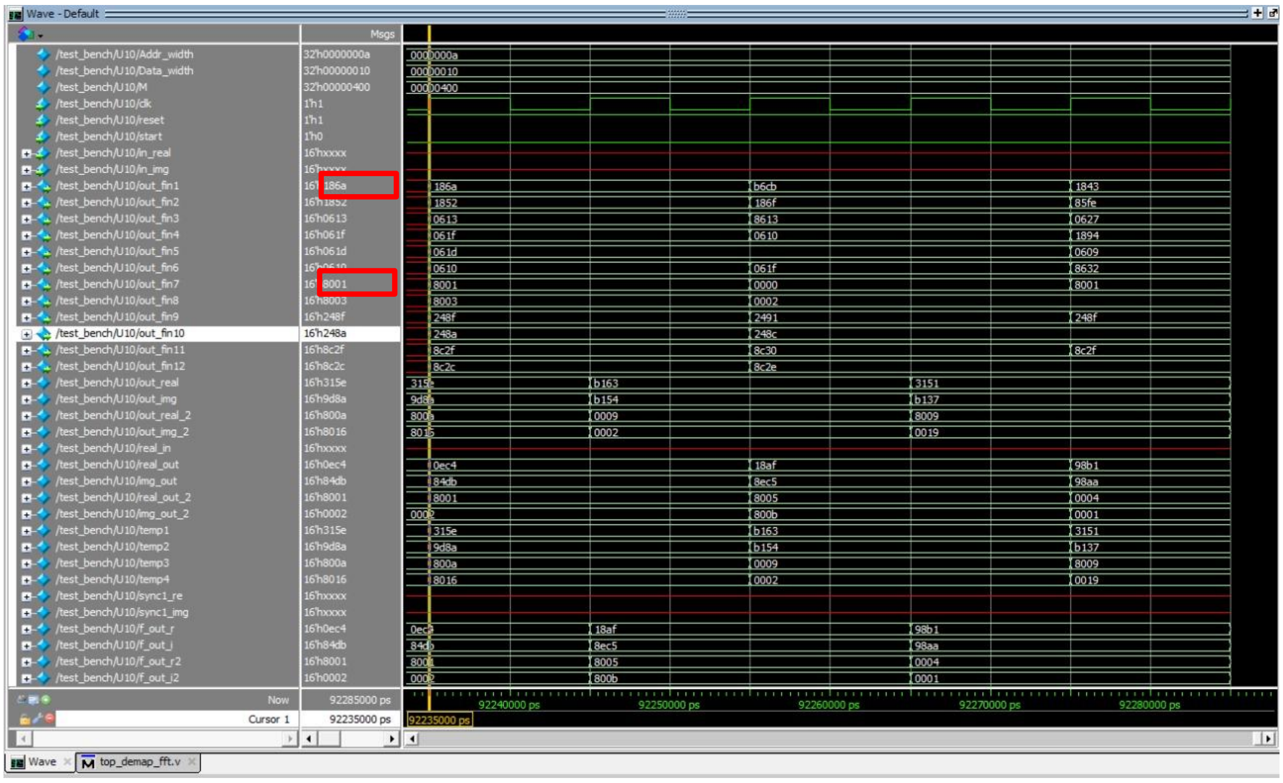
Figure 111:demapper output for 1st point



512
6.3587e-07
4.5440e-07
0.5714
0.5714
-0.1905
-0.1905

Figure 110:Demapper output for 512nd point

HDL simulation:



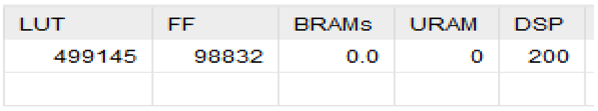
```
Wave - Default
                                        Msgs
/test_bench/U10/Addr_width      32'h0000000a
/test_bench/U10/Data_width      32'h00000010
/test_bench/U10/M               32'h00000400
/test_bench/U10/clk             1'h1
/test_bench/U10/reset           1'h1
/test_bench/U10/start           1'h0
/test_bench/U10/in_real         16'hxxxx
/test_bench/U10/in_img          16'hxxxx
/test_bench/U10/out_fin1        16'h186a
/test_bench/U10/out_fin2        16'h1852
/test_bench/U10/out_fin3        16'h0613
/test_bench/U10/out_fin4        16'h061f
/test_bench/U10/out_fin5        16'h061d
/test_bench/U10/out_fin6        16'h0610
/test_bench/U10/out_fin7        16'h8001
/test_bench/U10/out_fin8        16'h8003
/test_bench/U10/out_fin9        16'h248f
/test_bench/U10/out_fin10       16'h248a
/test_bench/U10/out_fin11       16'h8c2f
/test_bench/U10/out_fin12       16'h8c2c
/test_bench/U10/out_real        16'h315e
/test_bench/U10/out_img         16'h9d8a
/test_bench/U10/out_real_2      16'h800a
/test_bench/U10/out_img_2       16'h8016
/test_bench/U10/real_in         16'hxxxx
/test_bench/U10/real_out        16'h0ec4
/test_bench/U10/img_out         16'h84db
/test_bench/U10/real_out_2      16'h8001
/test_bench/U10/img_out_2       16'h0002
/test_bench/U10/temp1           16'h315e
/test_bench/U10/temp2           16'h9d8a
/test_bench/U10/temp3           16'h800a
/test_bench/U10/temp4           16'h8016
/test_bench/U10/sync1_re        16'hxxxx
/test_bench/U10/sync1_img       16'hxxxx
/test_bench/U10/f_out_r         16'h0ec4
/test_bench/U10/f_out_i         16'h84db
/test_bench/U10/f_out_r2        16'h8001
/test_bench/U10/f_out_i2        16'h0002
                         Now    92285000 ps
                    Cursor 1    92235000 ps
```

Figure 112:Verilog simulation

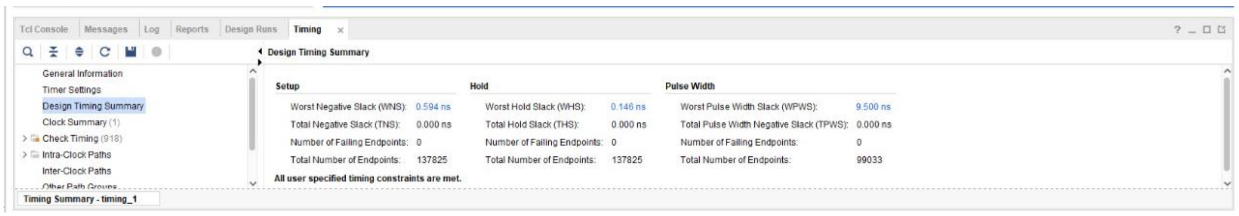
Comment:

- The first red square represents the first bit for the first point output from the FFT which is equal to the MATLAB output where 16’h186a = 0.3810
- The second red square represents the first bit for the 512 point output from the FFT which is equal to the MATLAB output where 16’h8001 = 0.0000006

Synthesis results:



LUT	FF	BRAMs	URAM	DSP
499145	98832	0.0	0	200

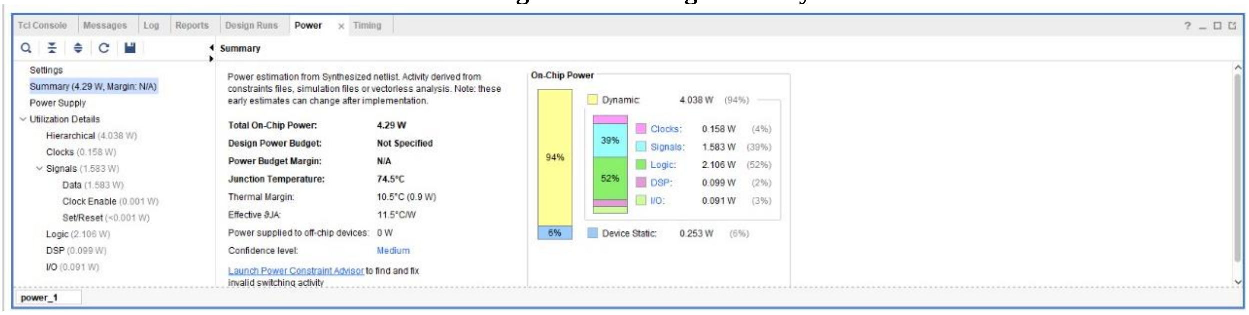


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.594 ns	Worst Hold Slack (WHS):	0.146 ns	Worst Pulse Width Slack (WPWS):	9.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	137825	Total Number of Endpoints:	137825	Total Number of Endpoints:	99033

All user specified timing constraints are met.

Figure 113:timing summary



Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**4.29 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**74.5°C**
Thermal Margin:	10.5°C (0.9 W)
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	4.038 W	(94%)
Clocks:	0.158 W	(4%)
Signals:	1.583 W	(39%)
Logic:	2.106 W	(52%)
DSP:	0.099 W	(2%)
I/O:	0.091 W	(3%)
Device Static:	0.253 W	(6%)

Figure 114:power results

Comment:

- Integrated block is synthesizable without any violations at 50 MHZ with power of 4.29 watt

IX. Conclusion:

No doubts that the 5G is going nowadays through a massive deployment to support not only the telecommunication field as it’s expected to drive global economic growth.

Our role in this project was to implement 5G receiver in the physical layer which is the start of network creation between devices. Through our work, we have studied digital design and implementation of different units, and how to integrate between different units using extra blocks if needed for this integration and we studied different ways to optimize the delay, area and the power. The results achieved in this thesis represents a starting point for a future vision in the 5G system.

X. Future work:

In this thesis we implement the whole chain of PDSCH Rx on high level model, we compared the results with the built-in function and checked the functionality of all blocks. After that, we went through digital design flow by implementing each block on hardware level except LDPC hardware implementation we only implement the retrieval punctured bits block but the rest of LDPC is not completed yet. We integrated the whole chain assuming the output of LDPC is passed to the next block from high level model. so, our future work is to complete the implementation of LDPC. From the synthesis results, the blocks which have a high power will be optimized by optimizing area or optimizing I/O or using power consumption techniques like clock gating. Our last step is to burn our chain on FPGA.

XI. References:

[1] 3GPP, *5G NR Multiplexing and channel coding*, version 15.5.0 release 15, 2019.

[2] Pedro Henrique & Prof. Leonardo Aguayo 2019, *DVBS2X standard FPGA implementation of LLR*, 2019.

[3] Qiang Sun & Wenfei Qi, *Soft-demodulation algorithm for 64QAM and its application in HSPA+*, 2012.

[4] Vahid Meghdadi, *BER calculation*, January 2008.

[5] Sarath Mohan & Sudeep Vasudevan, *Design of Multimode Deinterleaver for different Wireless Communication Standards*, October 2015.

[6] Zhen-dong Zhang, Bin Wu, Yu-mei Zhou, and Xin Zhang, *Low-Complexity Hardware Interleaver/Deinterleaver for IEEE 802.11a/g/n WLAN*, December 2011.

[7] Xiaokang Xiong , Yuhang Dai , Zhuhua Hu , Kejia Huo , Yong Bai , Hui Li , and Dake Liu, *Hardware Sharing for Channel Interleavers in 5G NR Standard*, 2021.

[8] Tsutomu Usui, Fumio Ishizu, and Keishi Murakami, *A Study of Adaptive Modulation Technique in OFDM*, March 2005.

[9] *5G TDD Synchronization Guidelines and Recommendations for the Coexistence of TDD Networks in the SBOR.5 GHz Range*, April 2020.

[10] Pekka Pirinen, *A brief overview of 5G research activities*, November 2014.

[11] Xingqin Lin, Jingya Li, Robert Baldemair, Jung- Fu Thomas Cheng, Stefan Parkvall, Daniel Chen Larsson, Havish Koorapaty, Mattias Frenne, Sorour Falahati, Asbjorn Grovlen & Karl werner, *5G New Radio: Unveiling the Essentials of the Next Generation Wireless Access Technology,* December 2019.

[12] AMTA, *5G and EMF Explained*, August 2019.

[13] Alaa Eldin.S. Hassan, Moawad Dessouky, Atef Abou Elazm and Mona Shokair *Evaluation of Complexity Versus Performance for Turbo Code and LDPC Under Different Code Rates*, 2012.

[14] Prof. Andrew Thangaraj, *LDPC and Polar codes in 5G standard*, November 2018.

[15] Kumar, Gautham & Kuchi Kiran, *Implementation of PDSCH Receiver and CSI-RS for 5G-NR. Master's thesis*, 2019.

[16] R. G. Gallager, *Low-Density Parity-Check Codes*, 1963.