Cairo University

Faculty of Engineering

Electronics and Electrical

Communications Department

DESIGN AND IMPLEMENTATION OF THE PHYSICAL LAYER OF NARROW-BAND IOT LTE DOWNLINK LOW POWER DIGITAL TRANSMITTER

A Thesis submitted in partial
fulfillment of the requirements for
the Degree of Bachelor of Science

Ahmed Ragab Ahmed

Hassan Mohamed Elbaroudy

Khaled Mahmoud Ibrahim

Khaled Mostafa Abdelhakim

Salma Mohamed Sultan

Under the supervision of

Dr. Hassan Mostafa

Si-Vision

July, 2022

Acknowledgements

Words cannot express our gratitude to our professor Dr. Hassan Mostafa for his invaluable patience and feedback. We also could not have undertaken this journey without his support, who generously provided knowledge and expertise.

We are also grateful to our advisors from Si-Vision, Eng. Ramy Raafat, Eng. Khaled Ismail, Eng. Mahmoud Samy, Eng. Mohamed Maher, Eng. Mohamed Talha, and Eng. Mahmoud Tantawi for their time, effort, regular feedback sessions, and moral support.

Finally, we wouldn’t forget to mention our families, especially our parents, brothers, and sisters. Their belief in us has kept our spirits and motivation high during this process.

Abstract

LTE cellular technology is now globally widespread. One of the most interesting applications that this technology offers concerns the "Internet of Things", better known as IoT which gave birth to several new technologies such as the Narrow-Band IoT (NB-IoT) which is a new cellular technology introduced by the 3GPP Release-13. This technology provides a communication standard for wide areas and its main feature is low power consumption.

In this thesis, the digital design and the FPGA implementation of the Narrowband Physical Downlink Shared Channel (NPDSCH) for a low-power OFDM transmitter for NB-IoT applications is proposed. The thesis shall go through all project’s steps starting from theoretical concepts & extracting the specifications from the 3GPP Release-14 standard, going through the detailed design of each block separately using MATLAB modeling then RTL implementation and use the MATLAB functions to verify the RTL design, ending up with system integration & ASIC synthesis. The whole developed system has been implemented on a Vertex-7 FPGA.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

OFDM	Orthogonal Frequency Division Multiplexing
NB-IoT	Narrowband Internet of Things
NPDSCH	Narrowband Physical Downlink Shared Channel
ICI	Inter-Carrier Interference
ISI	Inter-Symbol Interference
3GPP	3rd Generation Partnership Project
IFFT	Inverse Fast Fourier Transform
LTE	Long Term Evolution
PAPR	Peak to Average Power Ratio
SC-FDMA	Single carrier frequency division multiple access
CP	Cyclic Prefix
CRC	Cyclic Redundancy Check
NPSS	Narrowband Primary Synchronization Signal
NRS	Narrowband Reference Signal
FDD	Frequency division duplexing
TDD	Time division duplex
LAA	Licensed-Assisted Access
MSB	Most Significant Bit
ASK	Amplitude Shift Keying
FSK	Frequency Shift Keying
PSK	Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
N_{ID}^{cell}	Narrowband physical layer cell identity
RNTI	Radio network temporary identifier
n_f	System frame number
n_s	Slot number within a radio frame
T_f	Radio frame duration
Δf	Subcarrier spacing
T_s	Slot duration
$N_{RB}^{max,DL}$	The maximum number of resource blocks in uplink.

Chapter One

1. Introduction

1.1. Motivation :

Narrowband Internet of things “NB-IoT” is a low-power wide-area network “LPWAN” radio technology standard developed by 3GPP for cellular devices and services. LTE was designed for high speed communication and is not optimized for applications that need to support a potentially large number of low-rate and low power devices.

These low cost devices, typically used in applications such as sensors, remote maintenance and tracking, health-care, utilities etc, are expected to have very low complexity and low power consumption with a very long battery life. With the advent of IoT devices, the total number of connected devices is expected to reach 28 billion by 2023. Hence, it is desirable to develop LTE based wireless systems suitable for low data-rate and low power IoT applications. In NB-IoT applications, typically small data volume amounts need to be infrequently transmitted.

Realizing the importance of IoT for low-power and low cost applications with extended coverage and very long battery life, 3GPP introduced NB-IoT, as a part of their LTE-Release-14, where NB-IoT is a new mobile network, which is based on the LTE standard and is used exclusively for IoT applications. Compared to mobile networks (2G, 3G and 4G), NB-IoT offers energy-saving capabilities that increase the battery life of simple IoT applications up to 10 years.

NB-IOT technology in particular supports a range of data rates. It depends on the channel quality, or the signal-to-noise ratio and the quantity of resources in certain areas (bandwidth). Also, each device has a specific power budget, which leads to combine the power of several devices. There are many applications on IOT like smart homes, wearables, traffic management, water distribution, smart grid, connected cars, connected health, … etc.

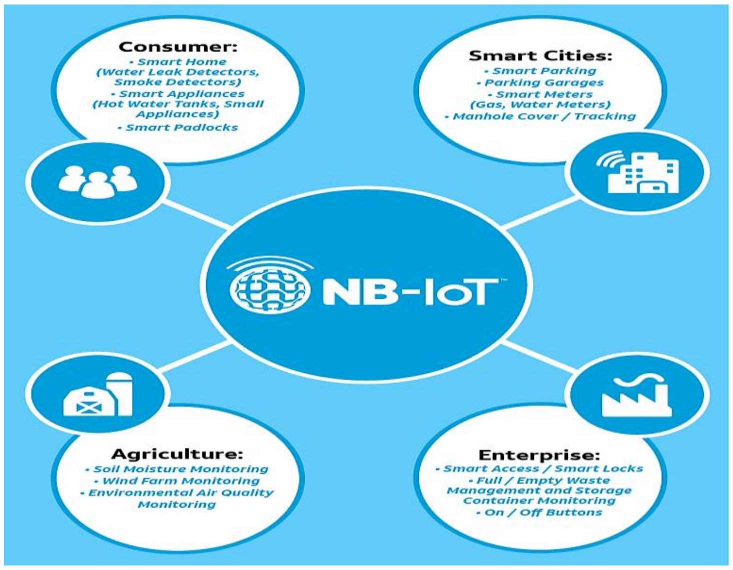

Consumer:
• Smart Home
(Water Leak Detectors, Smoke Detectors)
• Smart Appliances
(Hot Water Tanks, Small Appliances)
• Smart Padlocks
Smart Cities:
• Smart Parking
• Parking Garages
• Smart Meters
(Gas, Water Meters)
• Manhole Cover / Tracking
NB-IoT
Agriculture:
• Soil Moisture Monitoring
• Wind Farm Monitoring
• Environmental Air Quality Monitoring
Enterprise:
• Smart Access / Smart Locks
• Full / Empty Waste Management and Storage Container Monitoring
• On / Off Buttons

Figure 1: NB-IoT Applications

1.2. OFDM overview :

There are several complexities associated with wireless channels :

- Multi-path Fading : Unlike a wired channel which uses a fixed path, the signals in a wireless channel can reach a user using multiple paths. All these signals known as multipath components may have different channel gain and time delay. This combined effect causes what we know as multipath fading.

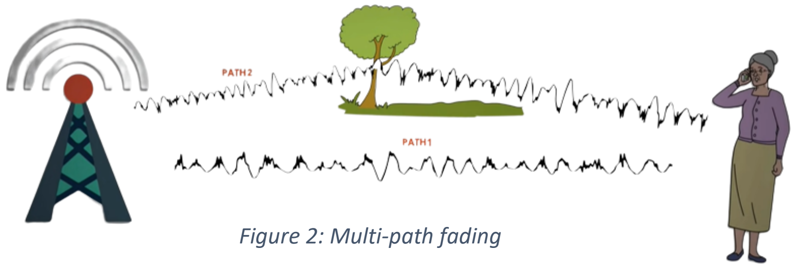


Figure 2: Multi-path fading

- Delay Spread : as a consequence of multipath propagation, the duration of a symbol gets extended. This may interfere with the next symbol. This is called Inter-symbol interference (ISI) or cross-talk. Guard periods are introduced to avoid cross-talk.

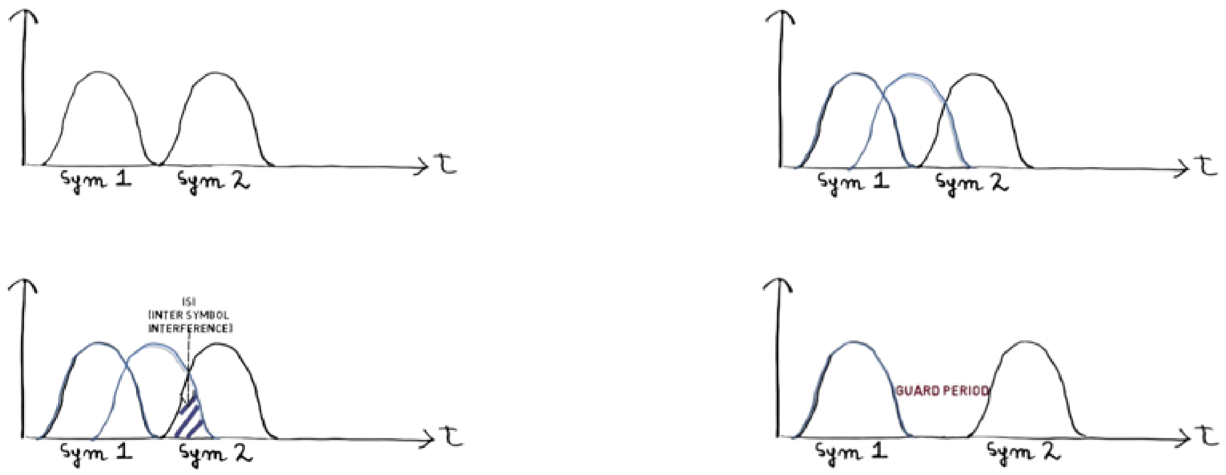


Figure 3: ISI caused by delay spread and guard periods inserting

- Frequency Selective Fading : Signals having bandwidth higher than the coherence bandwidth of the channel faces variable attenuation at different frequencies, this ultimately distorts the signal and giving rise to frequency selective fading. Complex channel equalization techniques are employed to reduce it.

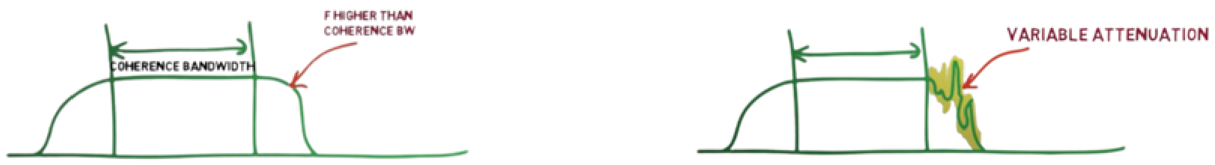


Figure 4: Frequency Selective Fading

- Inter-channel Interference : Often signal bandwidth of adjacent carrier frequencies overlap with each other giving rise to inter-channel interference. Guard bands were introduced to avoid inter-channel interference.

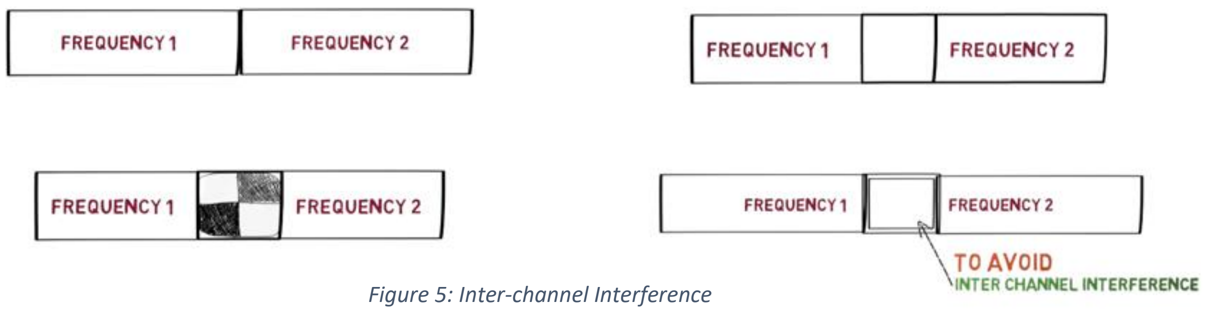


Figure 5: Inter-channel Interference

All these limitations compounded with the scarcity of bandwidth gave rise to multiple access technique OFDMA. Before diving into OFDMA, let us understand multi-carrier wireless transmission system. Suppose a signal to be transmitted over a bandwidth B, and carrier frequency F_c. Then symbol time for this would be $T_s = {}^{1}/_{B}$, for a single carrier wide-band channel of 1 MHz for example, the symbol time will be 1 microsecond. Given that the delay spread of the channel is 2 microseconds, then the combined symbol time would be 3 microseconds, which means delay spread occupies 66 % of the combined symbol time, thus reducing the efficiency of the channel by 1/3.

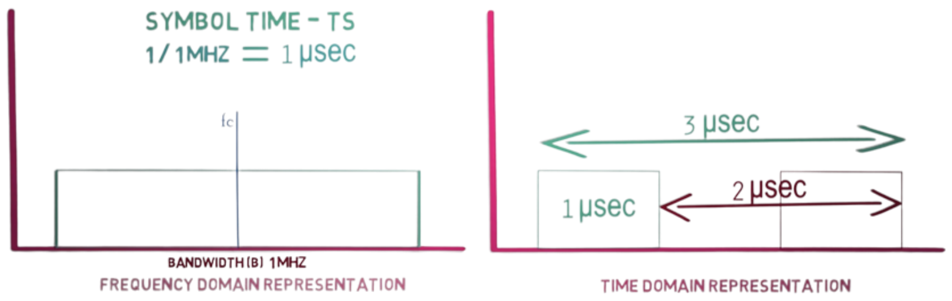


Figure 6: Symbol duration in time domain

As delay spread is difficult to control, the effect of delay spread can be minimized by using multiple sub-carriers of lesser bandwidth, so instead of having a single carrier of 1 MHz, we divide it into 100 sub-carriers of 10 KHz, each having a symbol duration of 100 microseconds. So a delay spread of 2 microseconds will have a negligible effect over the channel efficiency.

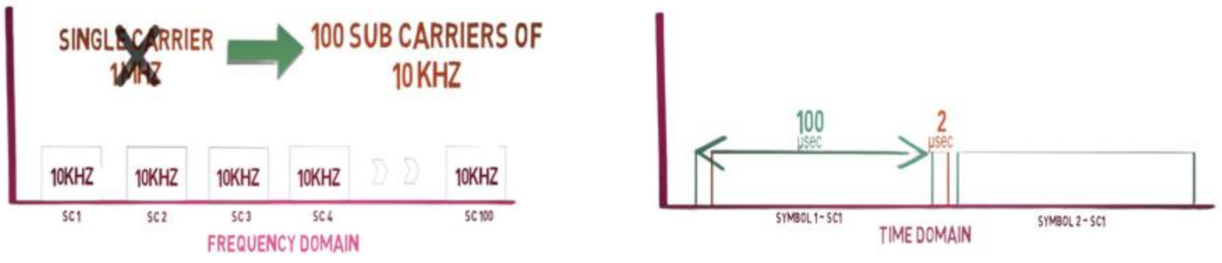


Figure 7: Dividing the carrier into sub-carriers of lesser frequencies

This concept is used in FDMA which uses slowly modulating sub-carriers of higher symbol duration. As these sub-carriers are modulated with data, they gain bandwidth centered around the sub-carriers frequencies. Guard bands are used to separate them in frequency domain.

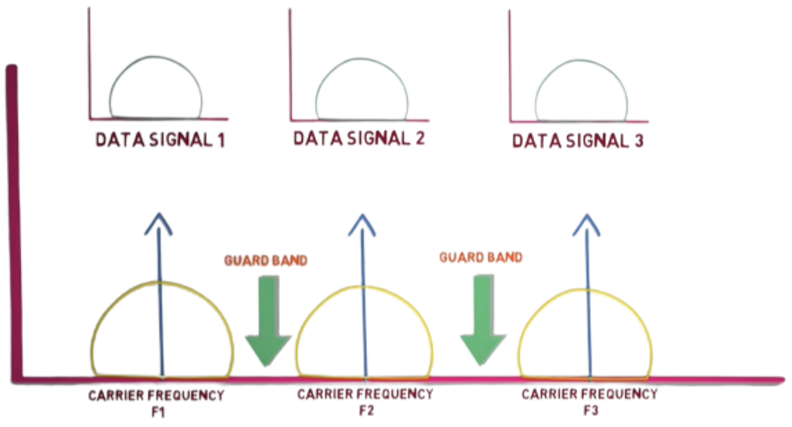


Figure 8: FDMA

We can represent this transmitted signal in the equation form as follows :

$$Transmitted\ signal = \sum_{n=0}^{N} X_n e^{jwtn}$$

Where the summation of individual symbol multiplied with different carrier frequencies and transmitted at radio frequencies. OFDMA or Orthogonal Frequency Division Multiple Access is a special case of FDMA, where users are provided a set of sub-carriers overlapping in frequency domain.

However, these sub-carriers are specially designed to be orthogonal to each other and are placed in a manner that all other sub-carriers have a zero component at the peak of one sub-carrier which allows them to occupy the same bandwidth without any interference. This in turn negates the use of guard bands. As a result, the sub-carriers can be closely packed to increase channel efficiency.

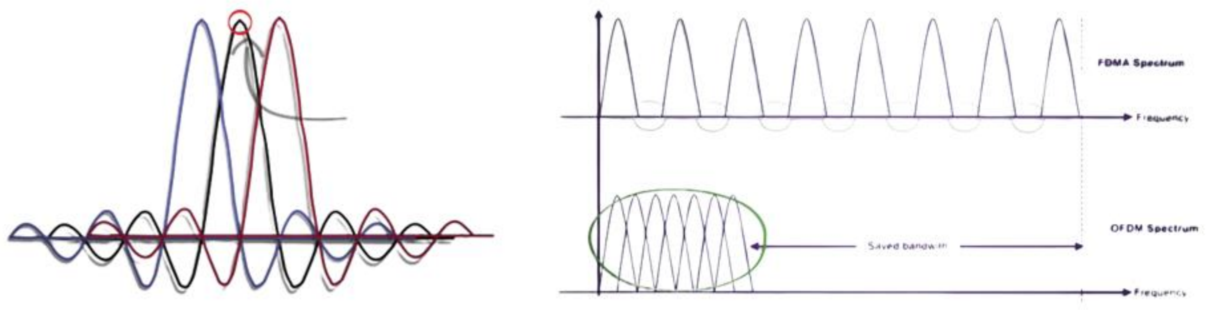


Figure 9: Orthogonal Frequency Multiple Access vs FDMA

- For the LTE uplink, a different concept is used for the access technique. Although still using a form of OFDMA technology, the implementation is called Single Carrier Frequency Division Multiple Access (SC-FDMA). One of the key parameters that affects all mobiles is that of battery life. Even though battery performance is improving all the time, it is still necessary to ensure that the mobiles use as little battery power as possible. With the RF power amplifier that transmits the radio frequency signal via the antenna to the base station being the highest power item within the mobile, it is necessary that it operates in as efficient mode as possible. This can be significantly affected by the form of radio frequency modulation and signal format.

- Signals that have a high peak to average ratio and require linear amplification do not lend themselves to the use of efficient RF power amplifiers. As a result it is necessary to employ a mode of transmission that has as near a constant power level when operating. Unfortunately OFDM has a high peak to average ratio (PAPR) which is proportional to the square of the number of carriers involved.

- In OFDMA, we have one-to-one mapping between symbol and sub-carrier, but SC-FDMA allows a symbol to be transmitted in parts over multiple sub-carriers. For example, in OFDMA, one symbol occupies one subcarrier of 15 kHz, but in SC-FDMA, the same symbol is distributed among multiple sub-carriers of 15 kHz. In short, SC-FDMA behaves like a single carrier system with short symbol duration compared to OFDMA. SC-FDMA reduces PAPR by reducing the number of carriers.

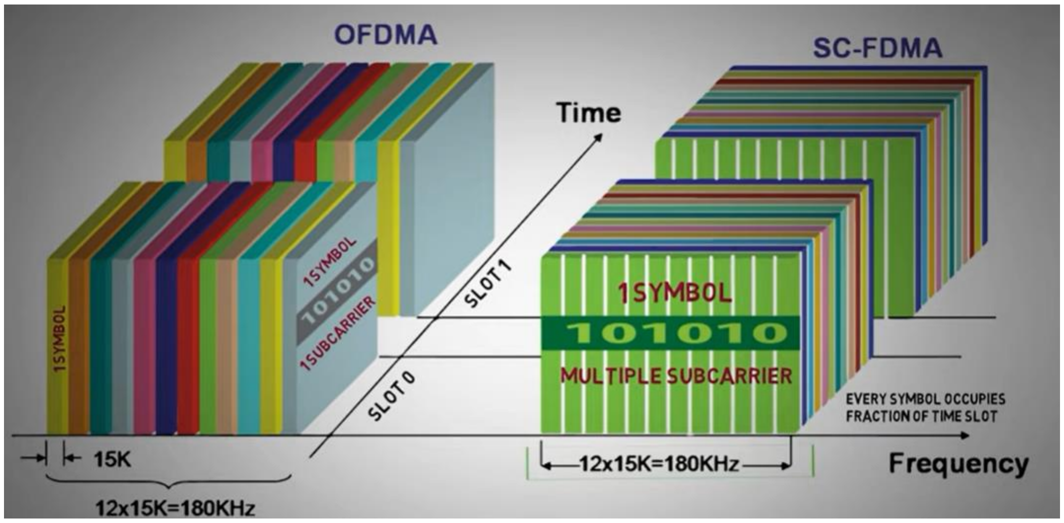


Figure 10: OFDMA for downlink vs SC-FDMA for uplink

1.3. Modes of operation :

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

- **Stand alone :** NB-IoT carrier is deployed independently of any LTE carrier and it can work as a replacement to GSM carriers.
- **Guard band :** Occupying a 180 kHz wide physical resource block in the guard band of existing LTE carrier band and it does not take any capacity from the main LTE traffic carrier.
- **In-band :** Occupying one physical resource block (PRB) within the LTE carrier bandwidth. This mode is the most efficient one as it allows the base station schedule to multiplex LTE and NB-IoT traffic in the same spectrum.

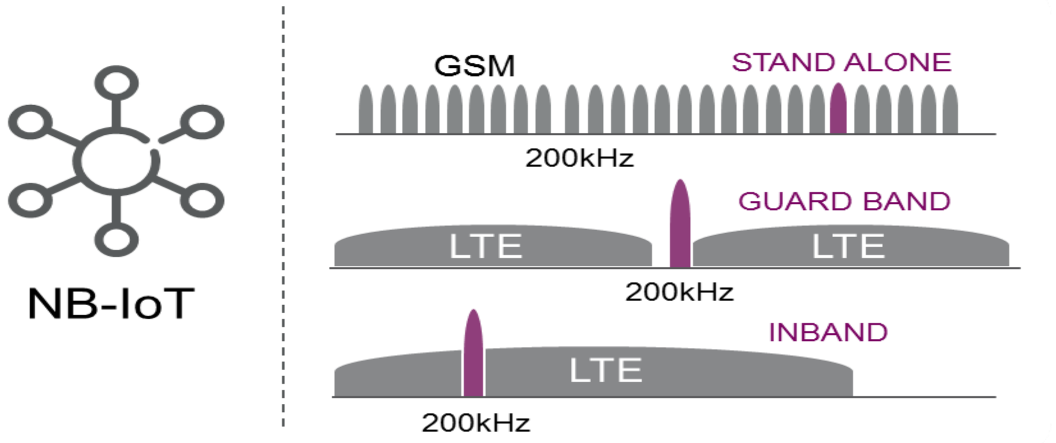


Figure 11: NB-IoT deployment scenarios

1.4. Frame Structure

According to 3GPP there are three radio frame structures designed for LTE:

- **The radio frame structure type 1:**
  Is only applicable to FDD (for both full duplex and half duplex operation) and has a duration of 10ms and consists of 20 slots with a slot duration of 0.5ms. Two adjacent slots form one sub-frame of length 1ms, except when the sub-carrier bandwidth is 1.25 kHz, in which case one slot forms one sub-frame.

- **The radio frame structure type 2:**
  Is only applicable to TDD and consists of two half-frames with a duration of 5ms each and containing each either 10 slots of length 0.5ms, or 8 slots of length 0.5ms and three special fields (DwPTS, GP and UpPTS) which have configurable individual lengths and a total length of 1ms.

- **The radio frame structure type 3:**
  Is only applicable to LAA secondary cell operation. It has a duration of 10ms and consists of 20 slots with a slot duration of 0.5ms. Two adjacent slots form one sub-frame of length 1ms. Any sub-frame may be available for downlink or uplink transmission.

Frequency Division Duplexing (FDD) is a method for establishing a full-duplex communications link that uses two different radio carrier frequencies for transmitting (uplink) and receiving (downlink). Both channels are separated by a defined offset frequency, which is also known as "guard band" which stops the interference between uplink channel and downlink channel

Time Division Duplexing (TDD) is a technique by which the uplink & downlink transmissions are carried over the same frequency by using synchronized time intervals,

- Factors to decide whether to use FDD or TDD :
  - **Latency :** FDD offers lower latency than TDD, since transmit and receive functions operate simultaneously and continuously, whereas in TDD, switching from transmit to receive incurs a delay that causes traditional TDD systems to have greater inherent latency.
  - **Equipment cost :** No diplexer is needed to isolate the transmitter and receiver in TDD, whereas in FDD, a diplexer is needed.
  - **Distance preferability :** As the distance increases, the guard period also increases in TDD because the point-to-point signal propagation time increases. The increased guard period significantly affects the efficiency, whereas FDD doesn't have a problem with small or large distances.
  - **Unbalanced Traffic :** In real-life network, the volume consumed in downlink is much higher than in uplink. In TDD, it is possible to dynamically adjust the capacity by utilizing more timeslots for downlink than uplink, whereas in FDD, the capacity is normally balanced in both directions.
  - **MAC layer complexity :** MAC layer in TDD system is complex compared to FDD system, as it has to deal with accurate time synchronization between the systems.

As far as NB-IoT-LTE is concerned, it is defined to support both the paired & unpaired spectrum for FDD (Radio frame structure Type 1 which is applicable to FDD only).

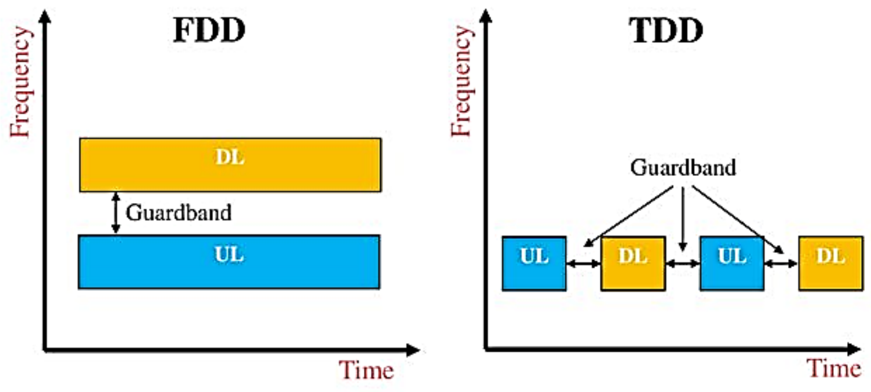


Figure 12: FDD vs TDD

For Narrow Band Downlink Physical Shared Channel, frame type 1 is used.

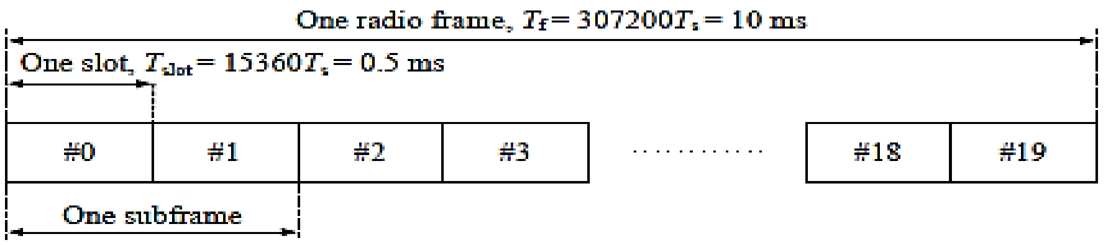

One radio frame, $T_f = 307200T_s = 10$ ms
One slot, $T_{slot} = 15360T_s = 0.5$ ms
#0
#1
#2
#3
#18
#19
One subframe

Figure 13: Radio frame structure type 1

According to the 3GPP

$$T_{frame} = 307200\, T_s = 10\, ms$$

$$T_{slot} = 15360\, T_s = 0.5\, ms$$

$$T_{sub-frame} = 30720\, T_s = 1\, ms$$

1.5. Physical Resource Block Structure for NB-IoT

- Resource block is also known as slot and its transmission time is about 0.5ms.
- Twelve consecutive subcarriers in the frequency domain and seven symbols in the time domain form each Resource Block.
- Two slots are combining together forming subframe and ten subframes are combining together forming a whole frame and it takes 10ms to transmit the whole frame where each subframe needs 1ms to be transmitted.
- The length of slot differs depends on the CP, in normal CP the slot contains 7 OFDM symbols and the time taken by the first symbol is greater than the rest, while in ended CP the slot contains 6 OFDM symbols and the slot's time is divided equally between the symbols.

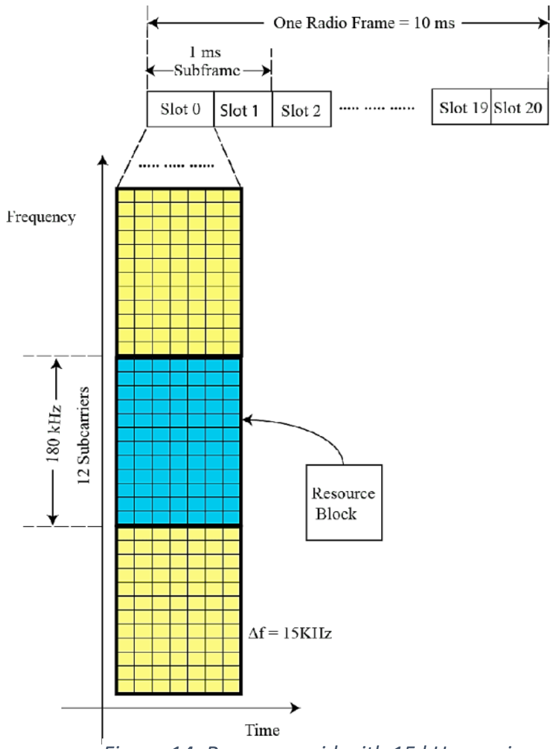


Figure 14: Resource grid with 15 kHz spacing

➢ For NB-IoT, Normal Cyclic Prefix is used and the number of Resource Blocks used is 1 Resource Block for the NB-IoT with 12 frequencies sub-carriers and 14 symbols for each sub-frame with total BW=180 KHz.

1.6. Problem Description

- Realizing the importance of IoT for low-power and low cost applications with extended coverage and very long battery life, 3GPP introduced narrowband IoT (NB-IoT), as a part of their LTE-Release-14. Although NB-IoT is based on LTE, but it’s a new radio-access technology as it is not fully backward compatible with the existing LTE devices. However, it can be easily integrated in the existing LTE network by allocating some of the time and frequency resources to NB-IoT. NB-IoT occupies 180 kHz of spectrum, which is substantially smaller than LTE bandwidths of 1.4–20 MHz
- The target of this project is to implement low power “Downlink Transmitter” for the Narrow Band Long Term Evolution (NB-LTE). The main goal is to design the Physical Layer chain of the LTE Rel.14 that targets the NB-LTE.
- The project has gone through the following phases:
  - Standard and literature reading
  - Block level Modeling using MATLAB
  - RTL Design
  - System Integration
  - ASIC Synthesis using Synopsys Design Compiler
  - Prototyping with FPGA
  - Documentation

1.7. Thesis Organization

Chapter One : We proposed an introduction and brief overview about NB-LTE and IoT applications, then we showed up the complexities associated with wireless transmission, after that we summarized the NB-LTE frame structure & resource grid.

Chapter Two : We’ll show the NPDSCH LTE Transmitter chain main architecture and a brief definition for each block.

Chapter Three : We’ll introduce the standard specifications and assumptions according to the 3GPP Release-14.

Chapter Four : We’ll show the detailed design for each block starting from block diagram then design steps ending up with the operation clarification followed by simulation results for both MATLAB and RTL and ASIC synthesis results on DC compiler of timing, area, and power reports

Chapter Five : The system integration is discussed in this chapter with its simulation & synthesis results.

Chapter Six : FPGA prototyping on Vertex-7 and the conclusion are discussed.

Chapter Two

2. Physical Downlink Transmitter Chain and sub-blocks' function

2.1. Transmitter Chain

The Narrow Band IoT LTE Transmitter chain is composed of the shown blocks, in addition it has register file which supplies the chain with the upper layer parameters, also a controller which feeds and control the whole chain and that will be discussed in chapter five.

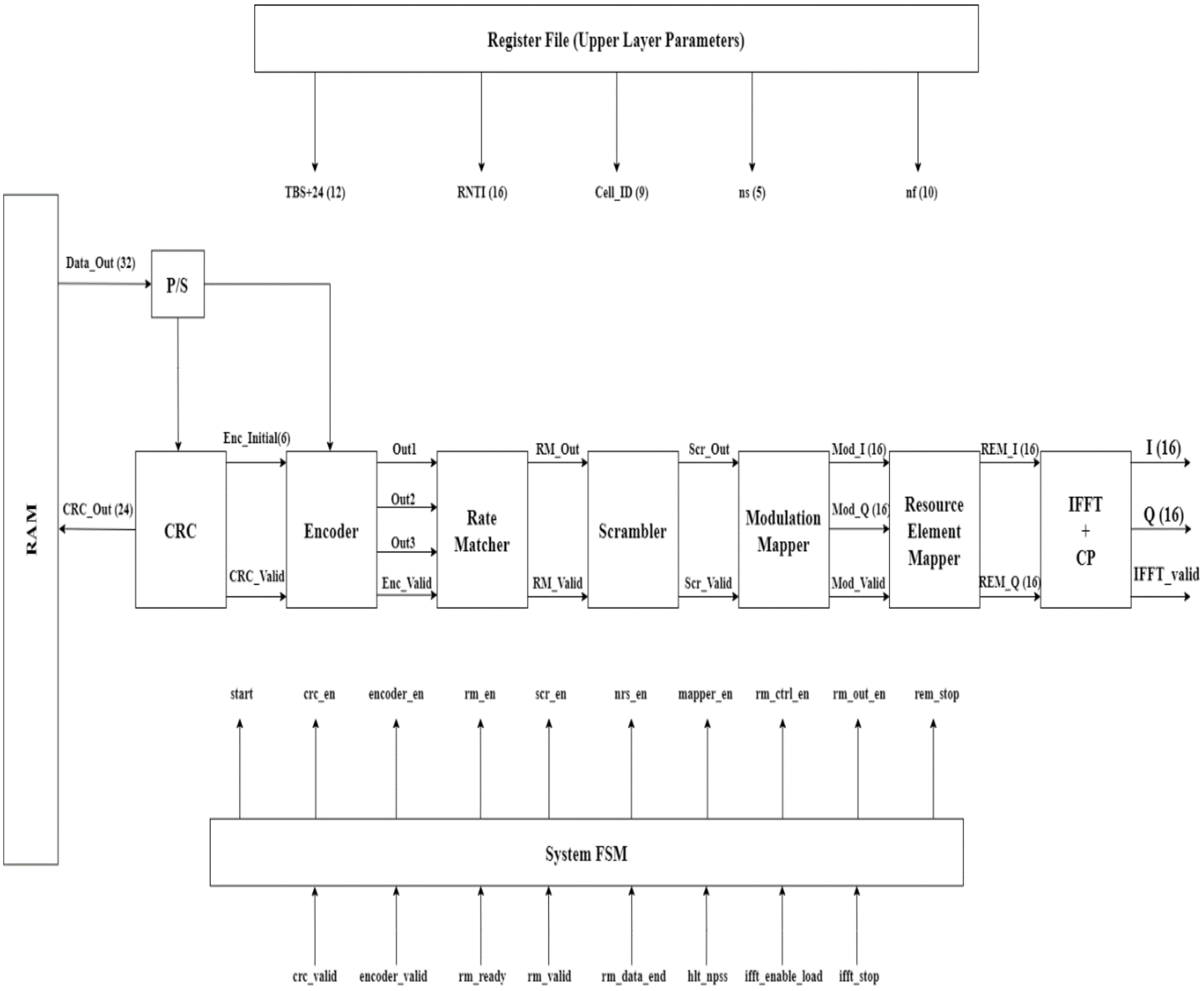


Figure 15: Transmitter Chain Block Diagram

2.2. Sub-blocks' function

2.2.1. Cyclic Redundancy Check (CRC)

We use Cyclic redundancy check as it is easy in binary domain, easy to analyze its mathematics and the hardware block that generate it consumes low power as it is consists of LFSR with constant length so it is useful to detect the famous errors. Error detection at the receiving end is achieved by generation the parity bits from the received information by dividing the data row by the same polynomial equation, if the reminder is zero so there is no error otherwise there is a presence of transmission errors in the received frame.

2.2.2. Encoder

In Communication systems, the most important thing is the BER and the efficiency of reliability of the sending data, as we send digital bits between source and destination through the channel which suffers from noise and interference, so we need to use channel coding techniques to overcome this problem and ensure that the data will arrive well, the encoding technique is to add redundancy bits to the data so there is a decrease in data rate, thus the price paid is to increase the bandwidth to send with the same data rate.

Convolutional Encoding is one of the most powerful techniques for Forward Error correction (FEC), there are many algorithms for decoding but the most powerful one is Viterbi algorithm in Viterbi Decoding.

As encoder starts and terminates in the same state, the conventionally terminated convolutional codes in which the LFSR is initialized with zero so the initial state is all zeros so zero tail bits are appended to the data to drive the encoder back to the all-zero state, In LTE systems, for example, the convolutional encoder has a constraint length of 7 which means that 6 tail bits are needed, so we append six zeros to the data which results in 18 bits of overhead for a code rate of 1/3 as shown in figure 16.

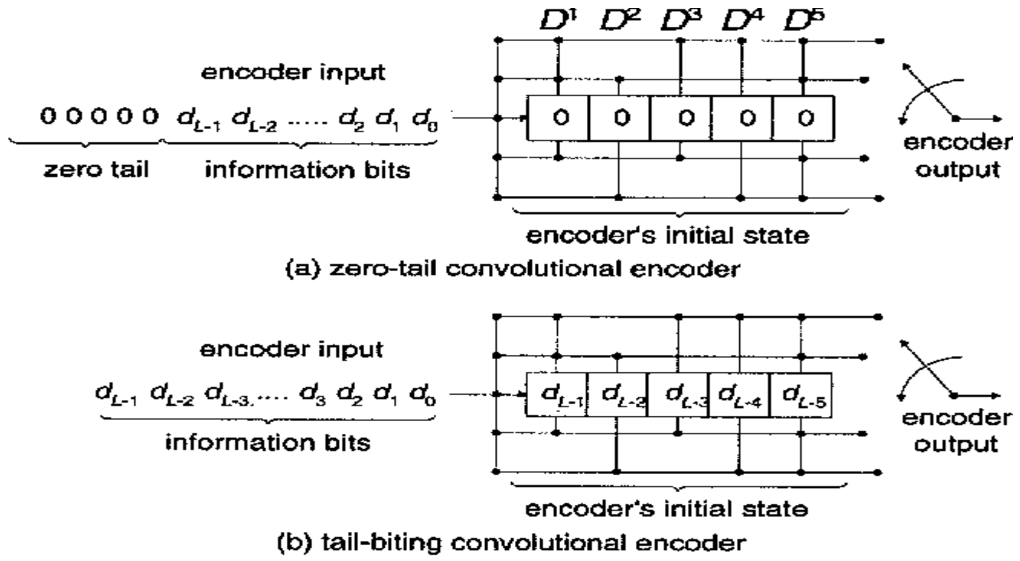

D^1 D^2 D^3 D^4 D^5
encoder input
0 0 0 0 0 d_{L-1} d_{L-2} ..... d_2 d_1 d_0
zero tail
information bits
0 0 0 0 0
encoder output
encoder's initial state
(a) zero-tail convolutional encoder
encoder input
d_{L-1} d_{L-2} d_{L-3}.... d_3 d_2 d_1 d_0
information bits
d_{L-1} d_{L-2} d_{L-3} d_{L-4} d_{L-5}
encoder output
encoder's initial state
(b) tail-biting convolutional encoder

Figure 16: The difference between tail-biting convolutional encoder and convolutional encoder

In tail-biting encoding, the encoder is initialized by the last bits of the input stream, this is different in conventional convolutional coding. The tail bits introduce overhead in the system and causes the terminated convolutional code to introduce some rate loss which results in 18 bits of overhead for a code rate of 1/3. This represents a large overhead especially if the data blocks to be encoded are short (which is the case here since the convolutional encoder is used to encode narrow band IoT). Also, tail-biting provides an equal error protection for all bits compared to the conventional approach where the first few input bits to the encoder are more protected since only a subset of transitions are allowed for the first few bits, so we use Tail biting convolution encoder.

Advantages:

- This type of coding saving carrier wave resources especially in OFDM system.
- Overcomes the problem of the added cost of transmitting extra termination bits experienced by trellis terminations.
- It avoids the rate loss.

Disadvantages:

- Decoding latency is increased due to the fact that initial and final states are required to correctly start tracing back.
- Receiver complexity is increased.

2.2.3. Rate Matcher

Rate matcher is one of the important blocks in our NPDSCH transmitter chain, it exists at the middle of the chain to match the number of bits in transport block to the number of bits that can be transmitted to a given allocation in the resource block.

It also controls the rate of the chain, since the tail bit encoder rate is 1/3 and by using this block, we can increase the rate or decrease it according to DCI and channel quality information.

This block is also used for interleaving which is very important in communication system based on encoding techniques to decrease the bit error rate.

2.2.4. Scrambler

Scrambler is an important block in communication systems which used to randomize the input stream of data and generate an output stream of data with the same length of the input. It is used to eliminate long run of zeros or ones to avoid both idle and reset cases and helps in clock recovery and having no DC component. Also, changing in the scrambling codes helps in security.

2.2.5. Modulation Mapper

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

2.2.6. Resource Element Mapper

- The resource element mapper maps :
  - Incoming QPSK symbols from modulator (data)
  - **N**arrowband **R**eference **S**ignals (NRS)
  - **N**arrowband **P**rimary **S**ynchronization **S**ignals (NPSS)
- Resource element mapper's main function is to take the output (Complex-Valued) symbols of the modulation mapper & NRS & NPSS and assign them to the proper subcarriers allocated for the NB-IoT bandwidth.
- NRS stands for Narrow-band Reference Signals that acts as pilots, which help in channel estimation process to get an estimate for the channel and equalize the effect of this channel. These pilots found inside each Sub-frame except two Sub-frames NPSS and NSSS Sub-frame.
- NRS is transmitted with data symbols at the transmitter and is used for estimating the channel performance in the downlink NB-IoT system at the receiver. NRS symbols are inserted into the specifically assigned subcarriers at every NB-IoT downlink slot. On each content of the resource grid which consists of NRS, an inverse discrete Fourier transform (IDFT) process is executed for converting it into a time-domain reference sequence with the addition of CP.

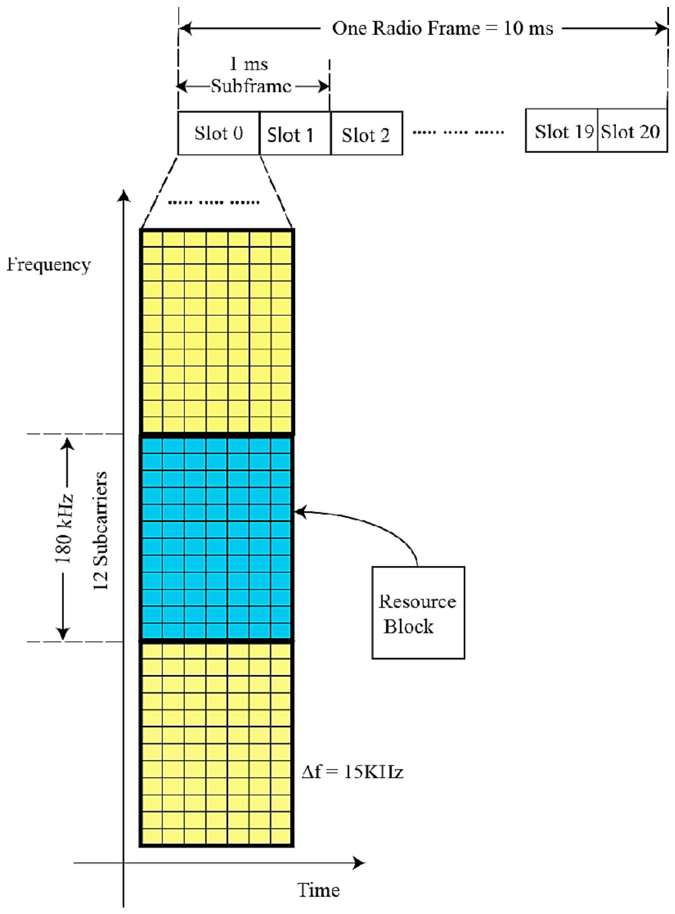


Figure 17: Physical Resource Block

- The NPSS helps UE to synchronize downlink to an NB-IoT cell.
  These are transmitted in certain slots based on an 80-ms repetition interval. NPSS is designed to allow devices to use a unified synchronization algorithm during initial acquisition without knowing the NB-IoT deployment mode (In-band, Guard-band or stand-alone). This is achieved by avoiding collision with REs used by LTE as much as possible.
- According to 3GPP, there are specific equations to get out the values of these pilots and other equations to get the Location of these NRS signals and synchronization signals.

2.2.7. IFFT

OFDM (Orthogonal Frequency Division Multiplexing) is a multicarrier modulation technique that uses 'n' number of orthogonal sub-carriers to become easier to separate out the information contained in each of these sub-carriers if they're orthogonal, at the receiver.

They're called sub-carriers because of the fact that a given bandwidth (B) is split into 'n' number of frequencies and each of these frequencies act as individual carriers.

Multiplexing is made possible by this technique. In fact, this technique converts a frequency selective fading channel into 'n' individual number of narrow band flat fading channels (Frequency Flat Fading channel is one whose signal bandwidth is less than the coherence bandwidth (bandwidth over which the channel's frequency response essentially remains constant) as in figure 18. No distortion occurs in flat fading channel. However, frequency selective fading channel is one whose signal bandwidth is much greater than the coherence bandwidth and hence different frequency components get attenuated/distorted by different amounts, hence aptly called frequency selective fading).

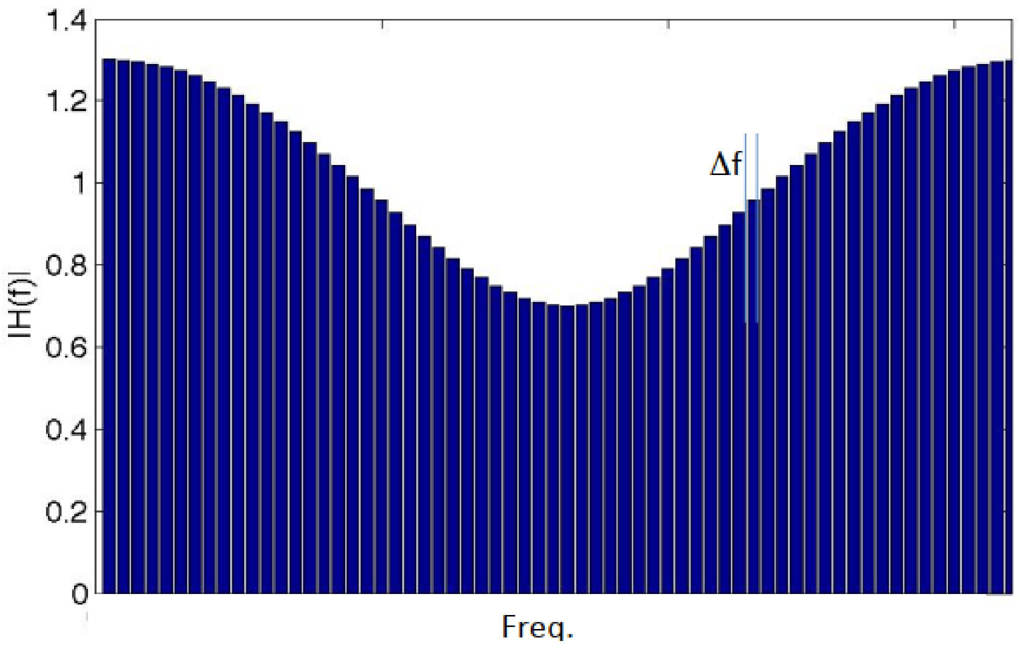
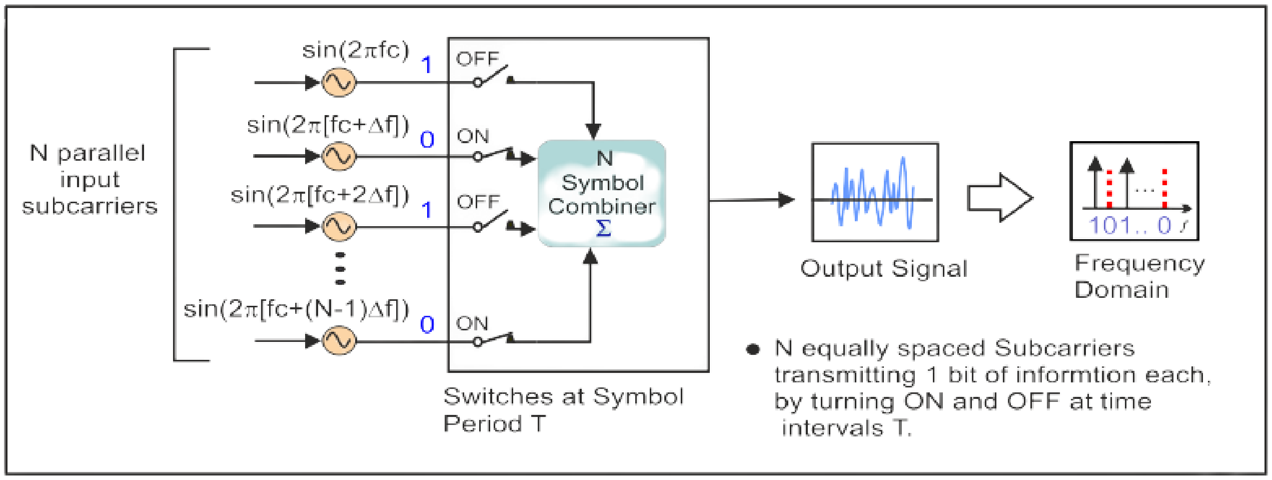


Figure 18: Multi-carrier System

Since 'n' number of sub-carriers are involved, it would necessitate the requirement of 'n' number of modulators (oscillators) as in figure 19. This would in-turn, increase the cost of design and the device size. Hence, alternatives were looked upon.

It was found that if complex exponential signals were used as carriers, the function of 'n' modulators could be replaced completely by a single IFFT operation at the transmitter(By mathematical manipulation).It's easier to design a single IFFT DSP chip than to design 'n' number of modulators. Hence, IFFT is used in OFDM transmitters.

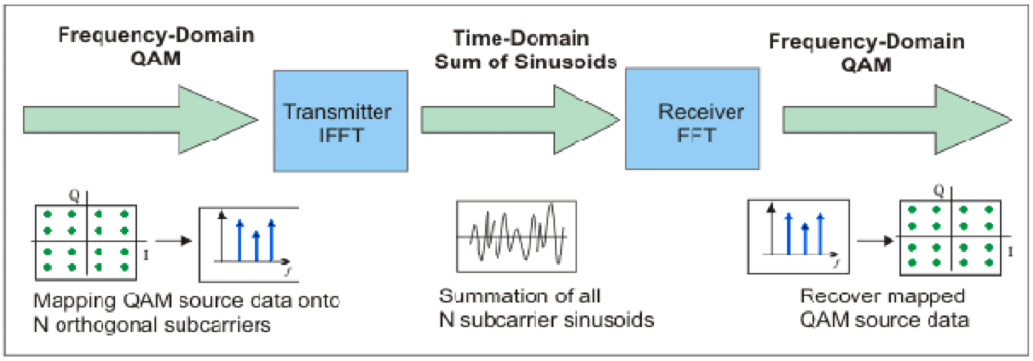


N parallel input subcarriers
sin(2πfc)
1
OFF
sin(2π[fc+Δf])
0
ON
sin(2π[fc+2Δf])
1
OFF
sin(2π[fc+(N-1)Δf])
0
ON
N Symbol Combiner Σ
Switches at Symbol Period T
Output Signal
Frequency Domain
101.. 0
N equally spaced Subcarriers transmitting 1 bit of informtion each, by turning ON and OFF at time intervals T.

Figure 19: Simple Analog OFDM System Implementation

The concepts used in the simple analog OFDM implementation can be extended to the digital domain by using a combination of Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) digital signal processing. These transforms are important from the OFDM perspective because they can be viewed as mapping digitally modulated input data (data symbols) onto orthogonal subcarriers, as shown in figure 21.

In OFDM system, the input bits are grouped and mapped to source data symbols that are a complex number representing the modulation constellation point (e.g. QPSK). These complex source symbols are treated by the transmitter as though they are in the frequency-domain and are the inputs to an IFFT block that transforms the data into the time-domain. The IFFT takes in N source symbols at a time where N is the number of subcarriers in the system. Each of these N input symbols has a symbol period of T seconds. Recall that the output of the IFFT is N orthogonal sinusoids. These orthogonal sinusoids each have a different frequency and the lowest frequency is DC.



Frequency-Domain QAM
Transmitter IFFT
Time-Domain Sum of Sinusoids
Receiver FFT
Frequency-Domain QAM
Mapping QAM source data onto N orthogonal subcarriers
Summation of all N subcarrier sinusoids
Recover mapped QAM source data

Figure 20: Simplified OFDM System Block Diagram

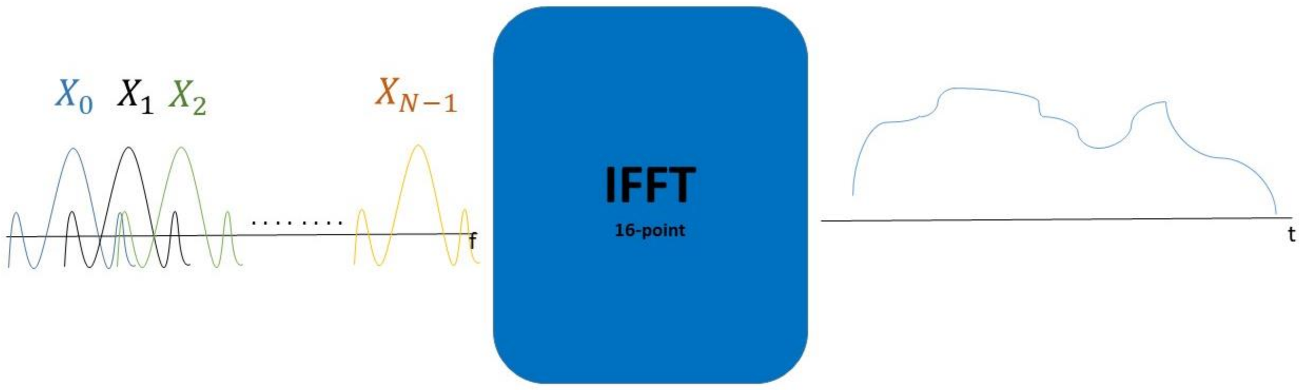


Figure 21: Symbols Mapping by IFFT

Cyclic Prefix (CP)

One of the primary reasons for using OFDM as a modulation format within LTE (and many other wireless systems for that matter) is its resilience to multipath delays and spread. However it is still necessary to implement methods of adding resilience to the system. This helps overcome the inter-symbol interference (ISI) that results from this.

In areas where inter-symbol interference is expected, it can be avoided by inserting a guard period into the timing at the beginning of each data symbol. It is then possible to copy a section from the end of the symbol to the beginning. This is known as the cyclic prefix, CP. The receiver can then sample the waveform at the optimum time and avoid any inter-symbol interference caused by reflections that are delayed by times up to the length of the cyclic prefix, CP.

The length of the cyclic prefix, CP is important. If it is not long enough then it will not counteract the multipath reflection delay spread. If it is too long, then it will reduce the data throughput capacity. For LTE, the standard length of the cyclic prefix has been chosen to be 4.69 µs. This enables the system to accommodate path variations of up to 1.4 km. With the symbol length in LTE set to 66.7 µs.

The symbol length is defined by the fact that for OFDM systems the symbol length is equal to the reciprocal of the carrier spacing so that orthogonality is achieved. With a carrier spacing of 15 kHz, this gives the symbol length of 66.7 µs

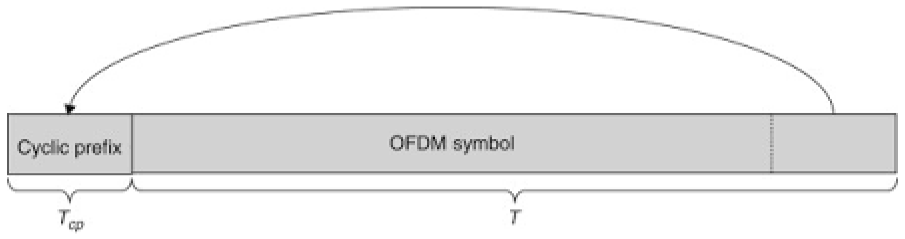


Figure 22: Cyclic Prefix

Chapter Three

3. Standard Specifications and Assumptions

3.1. CRC

According to Release 14, Denote the input bits to the CRC computation by $a_0, a_1, a_2, a_3, ..., a_{A-1}$, and the parity bits by $p_0, p_1, p_2, p_3, ..., p_{L-1}$,then we attached the CRC bits to the data bits $b_0, b_1, b_2, b_3, ..., b_{B-1}$,where A is the size of the input sequence and L is the number of parity bits and $B = A + L$.

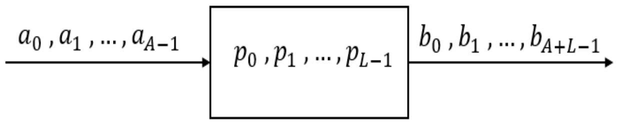


Figure 23: Parity bits added to input bits

The parity bits are generated by one of the following cyclic generator polynomials:

- $\mathbf{g_{CRC24A}}(\boldsymbol{D}) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$, which is added after each code block
- $\mathbf{g_{CRC24B}}(\boldsymbol{D}) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$, which is added only when code block segmentation is applied
- $\mathbf{g_{CRC16}}(\boldsymbol{D}) = [D^{16} + D^{12} + D^5 + 1]$, which is added to the Master Information Block
- $\mathbf{g_{CRC8}}(\boldsymbol{D}) = [D^8 + D^7 + D^4 + D^3 + D + 1]$, which is added to the control signal to transmit Channel Quality Indicator (CQI) information to the uplink channels like (PUCCH and PUSCH) by the Downlink Control Information (DCI) and (MIB).

Since we are adopting NB-IOT physical downlink shared channel, so max TBS = 2536 and the max value before using segmentation is Z = 6144, so we will not use segmentation so we choose $\mathbf{g_{CRC24A}}(\boldsymbol{D})$

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

The design is consisting of 24 LFSR that describes the above generator polynomial.

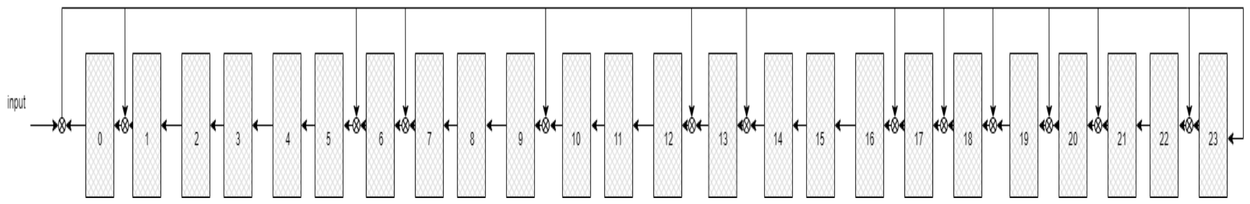
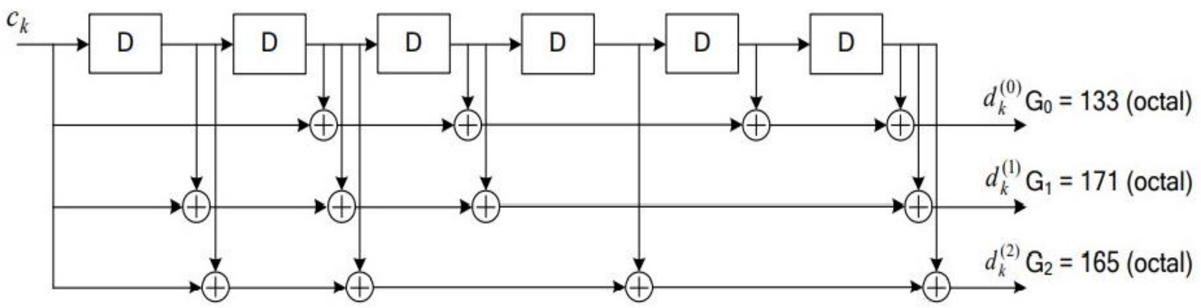


Figure 24: CRC Linear Feedback Shift Registers

3.2. Encoder

The tail-biting convolutional encoding is consisting of an LFSR and the initialization comes from the last 6 bits of data, at each clock cycle one bit enter the block and three bits comes out from the block, so this means that the rate = 1/3, the XOR of the output streams is as shown in Figure 25.



c_k
D
D
D
D
D
D
$d_k^{(0)}$ G_0 = 133 (octal)
$d_k^{(1)}$ G_1 = 171 (octal)
$d_k^{(2)}$ G_2 = 165 (octal)

Figure 25: XORing of encoder's output streams

The encoder output streams $d_k^{(0)}$, $d_k^{(1)}$ and $d_k^{(2)}$ correspond to the first, second and third parity streams, respectively as shown in Figure 25.

The polynomial of the output and its weights:

$G0$ = [1 0 1 1 0 1 1],

$G0$ = 1 + 2 + 8 + 16 = 133($octal$)

$G1$ = [1 1 1 1 0 0 1],

$G1$ = 1 + 8 + 16 + 32 + 64 = 171($octal$)

$G2$ = [1 1 1 0 1 0 1],

$G2$ = 1 + 4 + 16 + 32 + 64 = 165($octal$)

The rate of the encoder is denoted by $R = k/n$. Here k=1 and n=3 so R=1/3.

3.3. Rate Matcher

The rate matching for convolutional coded transport channels and control information consists of interleaving the three-bit streams, $d_k^{(0)}, d_k^{(1)} and\ d_k^{(2)}$ followed by the collection of bits and the generation of a circular buffer as depicted in Figure 26.

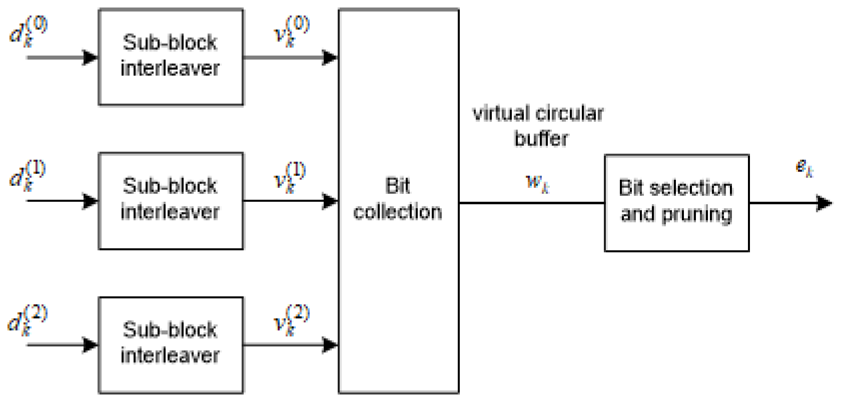


Figure 26: Rate Matcher stages

Rate matcher stages: -

1. **Sub-block interleavers**
   In this stage inputs from tail bit encoder which are $d_k^{(0)}, d_k^{(1)} and\ d_k^{(2)}$ are interleaved according to specific sub-block interleaving scheme to generate outputs which are $v_k^{(0)}, v_k^{(1)} and\ v_k^{(2)}$

The sub-block interleaving scheme: -

The bits input to the block interleaver are denoted by $d_0^{(i)}, d_1^{(i)}, \dots \dots, d_{D-1}^{(i)}$ where D is the number of bits *"D = TBS + 24"*, The output bit sequence from the block interleaver is derived as follows:

1. Assign $C_{subblock}^{cc}$ = 32 to the number of columns of the matrix, the columns of the matrix are numbered from 0,1, 2….., $C_{subblock}^{cc} - 1$ from left to right.
2. The number of rows of the matrix will be $R_{subblock}^{cc}$, such that $D \leq R_{subblock}^{cc} \times C_{subblock}^{cc}$, since the number matrix elements have to be equal to the maximum possible number of bits, therefore $R_{subblock}^{cc} = \frac{2536+24}{32}$ = 80.
   The rows of rectangular matrix are numbered 0, 1, 2,…,$R_{subblock}^{CC} - 1$ from top to bottom.
3. If $\left(R_{subblock}^{CC} \times C_{subblock}^{CC}\right) > D$, then $N_D = \left(R_{subblock}^{CC} \times C_{subblock}^{CC} - D\right)$ dummy bits are padded such that y_k = < *NULL* > for k = 0, 1,…, N_D - 1.

   Then, $y_{N_D+k} = d_k^{(i)}$, for k = 0, 1,…, D - 1, and the bit sequence y_k is written into the $\left(R_{subblock}^{CC} \times C_{subblock}^{CC}\right)$ matrix row by row starting with bit y_0 in column 0 of row 0:

$$\begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_{C_{subblock}^{CC}-1} \\ y_{C_{subblock}^{CC}} & y_{C_{subblock}^{CC}+1} & y_{C_{subblock}^{CC}+2} & \cdots & y_{2C_{subblock}^{CC}-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}} & y_{(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}+1} & y_{(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}+2} & \cdots & y_{(R_{subblock}^{CC}\times C_{subblock}^{CC}-1)} \end{bmatrix}$$

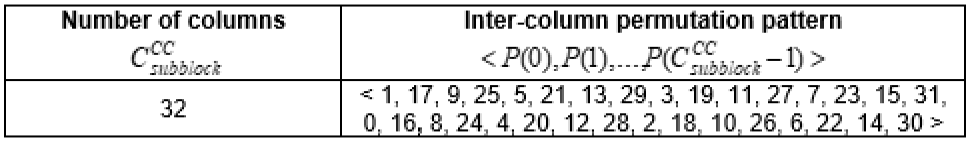
4. Perform the inter-column permutation for the matrix based on pattern $\langle P(j)\rangle_{j\in\{0,1,\ldots C_{subblock}^{CC}-1\}}$ that is shown in table 1, where $P(j)$ is the original column position of the j-th permuted column.

   After permutation of the columns, the inter-column permuted$\left(R_{subblock}^{CC}\times C_{subblock}^{CC}\right)$ matrix is equal to:

$$\begin{bmatrix} y_{P(0)} & y_{P(1)} & y_{P(2)} & \cdots & y_{P(C_{subblock}^{CC}-1)} \\ y_{P(0)+C_{subblock}^{CC}} & y_{P(1)+C_{subblock}^{CC}} & y_{P(2)+C_{subblock}^{CC}} & \cdots & y_{P(C_{subblock}^{CC}-1)+C_{subblock}^{CC}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{P(0)+(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}} & y_{P(1)+(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}} & y_{P(2)+(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}} & \cdots & y_{P(C_{subblock}^{CC}-1)+(R_{subblock}^{CC}-1)\times C_{subblock}^{CC}} \end{bmatrix}$$

5. The output of the block interleaver is the bit sequence read out column by column from the inter-column permuted $\left(R_{subblock}^{CC}\times C_{subblock}^{CC}\right)$ matrix.

   The bits after sub-block interleaving are denoted by$v_0^{(i)}, v_1^{(i)}, v_2^{(i)}, \ldots, v_{K_\Pi-1}^{(i)}$, where v_o^i corresponds to $y_{p(0)}$, v_1^i to $y_{P(0)+C_{subblock}^{CC}}$ … and $K_\Pi = \left(R_{subblock}^{CC}\times C_{subblock}^{CC}\right)$



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

Figure 27: inter-column permutation pattern for sub-block interleaver

2. Bit collection

The circular buffer of length $K_w = 3K_\Pi$ is generated as follows:

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

$w_{K_\Pi+k} = v_k^{(1)}$ for k = 0,…, $K_\Pi - 1$

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

3. Bit selection and data transmission

Denoting by E the rate matching output sequence length, the rate matching output bit sequence is e_k, k = 0, 1,.., $E-1$.

```
Set $k$ = 0 and $j$ = 0
while {$k$ < $E$}
    if $w_{j\ mod\ K_w} \neq < NULL >$
        $e_k = w_{j\ mod\ K_w}$
        $k$ = $k$ +1
    end if
    $j$ = $j$ +1
end while
```

3.4. Scrambler

According to 3GPP Narrowband LTE standard release 14, Scrambler implementation is based on pseudo random sequence that generated from two 31-bit linear feedback shift register (LFSR) and XOR gates. The two sequences of the two LFSRs get XORed and the output is called a Gold sequence.

The sequence of each LFSR is configure by the following polynomials:

- $x_1(n) = 1 + D^3 + D^0$
- $x_2(n) = 1 + D^3 + D^2 + D^1 + D^0$

In order to get the required output each LFSR must be initialized so

- $x_1(n)$ will be initialized as follow:
$$x_1(0) = 1,\ x_1(n) = 0\ where\ n = 1,2,3, \dots \dots ,30$$
- $x_2(n)$ will be initialized as follow:
$$C_{init} = \sum_{i=0}^{30} x_2(i) * 2^i$$
$$C_{init} = n_{RNTI} * 2^{14} + n_f mod 2 * 2^{13} + \left\lfloor \frac{n_s}{2} \right\rfloor * 2^9 + N_{ID}^{cell}$$

Then to work properly there is an initial 1600 shift cycles done to induce more randomness for the initial state of the two LFSRs, so $N_c = 1600$ then the two outputs of the LFSRs get XORed to generate the gold sequence $c(n)$ so the output gold sequence is given by:

$$c(n) = \left(x_1(n+N_c) + x_2(n+N_c)\right) mod2$$

$$x_1(n+31) = \left(x_1(n+3) + x_1(n)\right) mod2$$

$$x_2(n+31) = \left(x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)\right) mod2$$

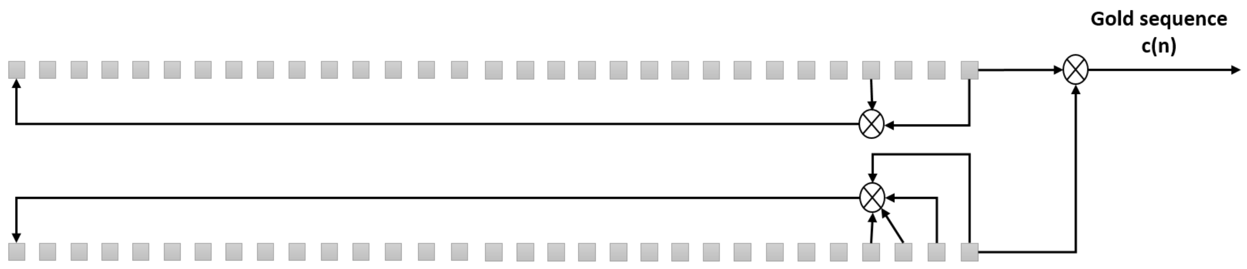


Figure 28: Gold sequence generation

3.5. Modulation Mapper

According to 3GPP Narrowband LTE standard release 14 there are two main schemes of modulation for NB-IoT, QPSK and BPSK, both are supported for uplink and only QPSK is supported for downlink so we will use QPSK only. QPSK is a modulation scheme that maps every two bits to one of four symbols on the constellation (complex-valued modulation symbol S = I + jQ) as shown in the figure below.

b_i , b_{i+1}	**I**	**Q**
00	$1/\sqrt{2}$	$1/\sqrt{2}$
01	$1/\sqrt{2}$	$-1/\sqrt{2}$
10	$-1/\sqrt{2}$	$1/\sqrt{2}$
11	$-1/\sqrt{2}$	$-1/\sqrt{2}$

Figure 29: QPSK modulation mapping

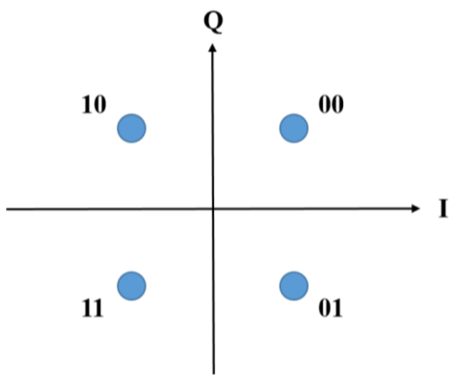


Figure 30: QPSK Constellation

3.6. Resource Element Mapper

3.6.1. NRS Generation

NRS value depends on three variables l, n_s, $and\ m$ so that the value of NRS changes according to any variation of these parameters.

l: $indicates\ the\ number\ of\ symbols\ that\ contain\ the\ NRS$ and it is $equal$ 5,6

ns: $indicates\ the\ number\ of\ the\ recieved\ slot\ its\ range$ [0,9]

$m = 0,1,\dots,2N_{RB}^{maxDL} - 1 \quad , \quad for\ narrow\ band,$

$N_{RB}^{maxDL} = 1\ as\ there\ is\ only\ one\ resource\ block$

$Therefore\ m = 0,1$

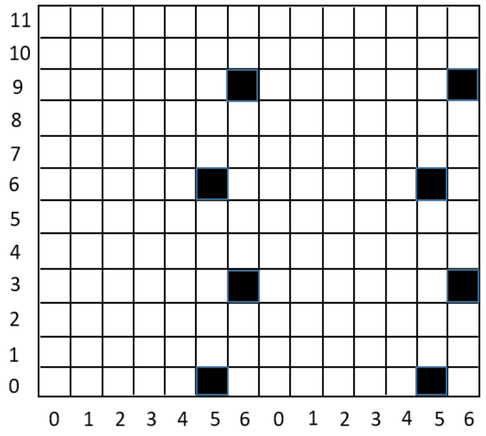


Figure 31: Sub-frame (two slots) of the radio frame

$$NRS\ values\ equation: \ r_{l,n_s}(m) = \frac{1}{\sqrt{2}}\big(1 - 2c(2m)\big) + j\frac{1}{\sqrt{2}}\big(1 - 2c(2m+1)\big)$$

Where C refers to the pseudo random sequence generation defined by a length-31 Gold Sequence:

$$c(n) = \big(x_1(n + N_c) + x_2(n + N_c)\big)\ mod2$$

$$x_1(n + 31) = \big(x_1(n+3) + x_1(n)\big)mod2$$

$$x_2(n + 31) = \big(x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)\big)mod2$$

$The\ initialization\ of\ the\ first\ m - sequence\ is\ given\ by:$

$$x_1(0) = 1 \quad and \quad x_1(n) = 0\ , n = 1,2,\dots,30$$

$The\ initialization\ of\ the\ second\ m - sequence\ is\ given\ by:$

$$C_{init} = 2^{10}(7(n_s + 1) + l + 1)\big(2N_{ID}^{cell} + 1\big) + 2N_{ID}^{cell} + N_{cp}$$

$where\ N_c = 1600\ \ ,\ \ N_{cp} = 1\ for\ normal\ cyclic\ prefix$

3.6.2. Narrowband Reference Signals Location (NRS index)

$$k = 6m + (v + v_{shift}) \bmod 6$$

$$l = N_{symb}^{DL} - 2 \ , \qquad N_{symb}^{DL} - 1 \ = 5,6 \ \ in\ each\ slot$$

k: *refers to the place of a pilot to be in which row, ranged from* 0 *to* 11 *as row refers to the sub − carrier*

l: *refers to the symbol number, ranged from* 0 *to* 6

N_{symb}^{DL}: *refers to the number of symbols in one slot for downlink and equal to* 7

$$v = \begin{cases} 0 & if\ l = 5 \\ 3 & if\ l = 6 \end{cases}$$

$$v_{shift} = N_{ID}^{cell} \bmod 6$$

$$m = 0,1, \dots, 2N_{RB}^{\max DL} - 1 \quad for\ narrow\ band\ N_{RB}^{\max DL} = 1\ as\ there\ is\ only\ one\ resource\ block$$

$$so\ m = 0,1$$

3.6.3. Narrowband Primary Synchronization Signals (NPSS)

The base sequence in frequency domain $d_l(n)$ *is defined as follows*

$$d_l(n) = \boldsymbol{S(l)\,.\,e^{-j\frac{\pi u n(n+1)}{11}}} \quad , \quad \boldsymbol{u = 5} \quad , \quad \boldsymbol{n = 0, 1, \dots, 10}$$

The code cover in time domain $S(l)$ *is defined as follows*

$$S(0{:}10) = \{1,1,1,1,-1,-1,1,1,1,-1,1\}$$

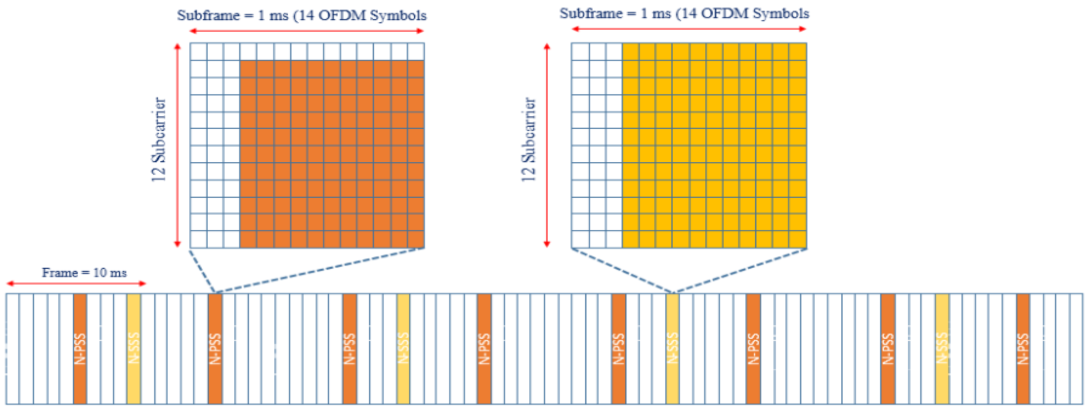


Figure 32: Mapping of NPSS in the frame

3.6.4. Data Rate Calculation for NPDSCH

- $\boldsymbol{For\ any\ NPDSCH\ (actual\ rate)}$

$$Max.Rate = \frac{7\ OFDM * 12\ subcarriers * 2\ (QPSK)}{0.5\ ms\ (slot\ time)} = 336\ Kbps$$

- $\boldsymbol{Useful\ data\ rate}$

  - $For\ our\ TBS = 960\ \ ,\ coding\ ^1/_3\ \ \rightarrow\ \ 960 * 3 = 2880\ \ ,\ QPSK\ \rightarrow$ $1440\ symbols$
    $Since\ each\ slot\ can\ transmit\ [(7 * 12) - 4\ (pilot\ symbols)] = 80\ symbols$
    $Therefore\ we\ need\ ceil\left\lceil\frac{1440}{80}\right\rceil = 18\ slots = 9\ subframes$

$$Data\ Rate = \frac{960}{9\ ms + 1\ ms} = 96\ Kbps$$

  - $For\ TBS = 296\ \ ,\ coding\ ^1/_3\ \ \rightarrow\ \ 296 * 3 = 888\ \ ,\ QPSK\ \rightarrow\ \ 444\ symbols$
    $Since\ each\ slot\ can\ transmit\ [(7 * 12) - 4\ (pilot\ symbols)] = 80\ symbols$
    $Therefore\ we\ need\ ceil\left\lceil\frac{444}{80}\right\rceil = 6\ slots = 3\ subframes$

$$Data\ Rate = \frac{296}{3\ ms} = 98.666\ Kbps$$

  - $For\ Max.TBS = 2536\ \ ,\ coding\ ^1/_3\ \ \rightarrow\ \ 2536 * 3 = 7608\ \ ,\ QPSK\ \rightarrow$ $3804\ symbols$
    $Since\ each\ slot\ can\ transmit\ [(7 * 12) - 4\ (pilot\ symbols)] = 80\ symbols$
    $Therefore\ we\ need\ ceil\left\lceil\frac{3804}{80}\right\rceil = 48\ slots = 24\ subframes$

$$Data\ Rate = \frac{2536}{24\ ms + 3\ ms} = 93.926\ Kbps$$

3.7. IFFT & CP

LTE supports different channel bandwidths and as a result a different number of resource blocks can be supported. Where the resource block is the smallest unit of resources that can be allocated to a user. The resource block is 180 kHz wide in frequency and 1 slot long in time. In frequency, resource blocks are 12 x 15 kHz, where 12 is the number of the subcarriers where each one is 15 kHz spacing. In NB-IOT we have only one resource block because the targeted devices is not expected to operate with huge data rates as in the LTE.

As in DL we use only QPSK and we have a sub-frame of length 0.5ms which consists of 7 OFDM symbols each with 12 sub-carrier, this constrained the data rate to be 336 Kbps which is not huge as in LTE which may reach 1Gbps in DL and 500 Mbps in UL.

As we have only 12 subcarriers in NB-IOT so we choose the IFFT size to be 16-point to reduce power, area and latency as it is the main goal of the NB-IOT, In the 16-point first 12 subcarriers are carrying data and the last four are padded zeroes which act as a guard band for the transmitted data.

In LTE there is two types of cyclic prefix, normal cyclic prefix where this configuration has 7 OFDM symbol within the resource block and extended cyclic prefix which has 6 OFDM symbol, NB-IOT (downlink) (Release 14) only supports the normal cyclic prefix where OFDM symbol duration is 66.7 µs and a CP of 5.02 µs for first symbol and 4.67 µs for rest of OFDM symbols. So the OFDM symbol length for first OFDM symbol is 71.18 µs and the remaining 6 symbols are 71.39 which leads us to the total length of the slot (resource block) which is 0.5 ms.

We implemented the I and Q to be 16-bit for each, to fit the range of numbers we deal with, and we notice that the last 4 bits of each 16 bits of the IFFT output is redundant, so we can consider the 2 DACs (next block after IFFT) are 12-bit, one for I and the other for Q.

Chapter Four

4. Design Implementation

4.1. CRC

4.1.1. Block Diagram

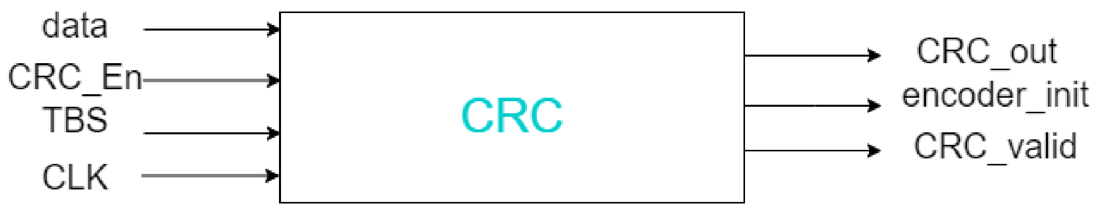


Figure 33: CRC top level

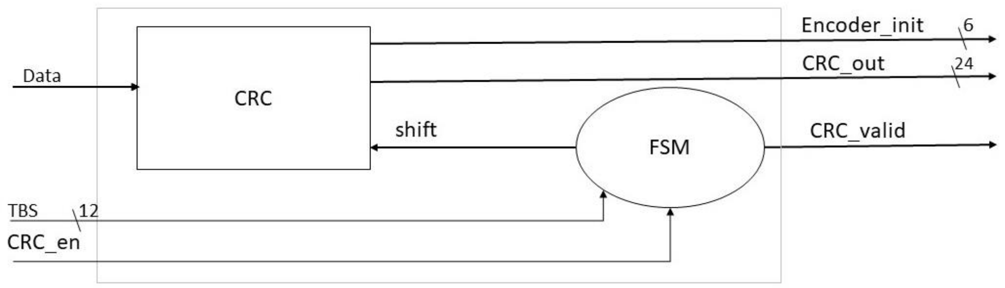


Figure 34: CRC Architecture

4.1.2. Interface Table

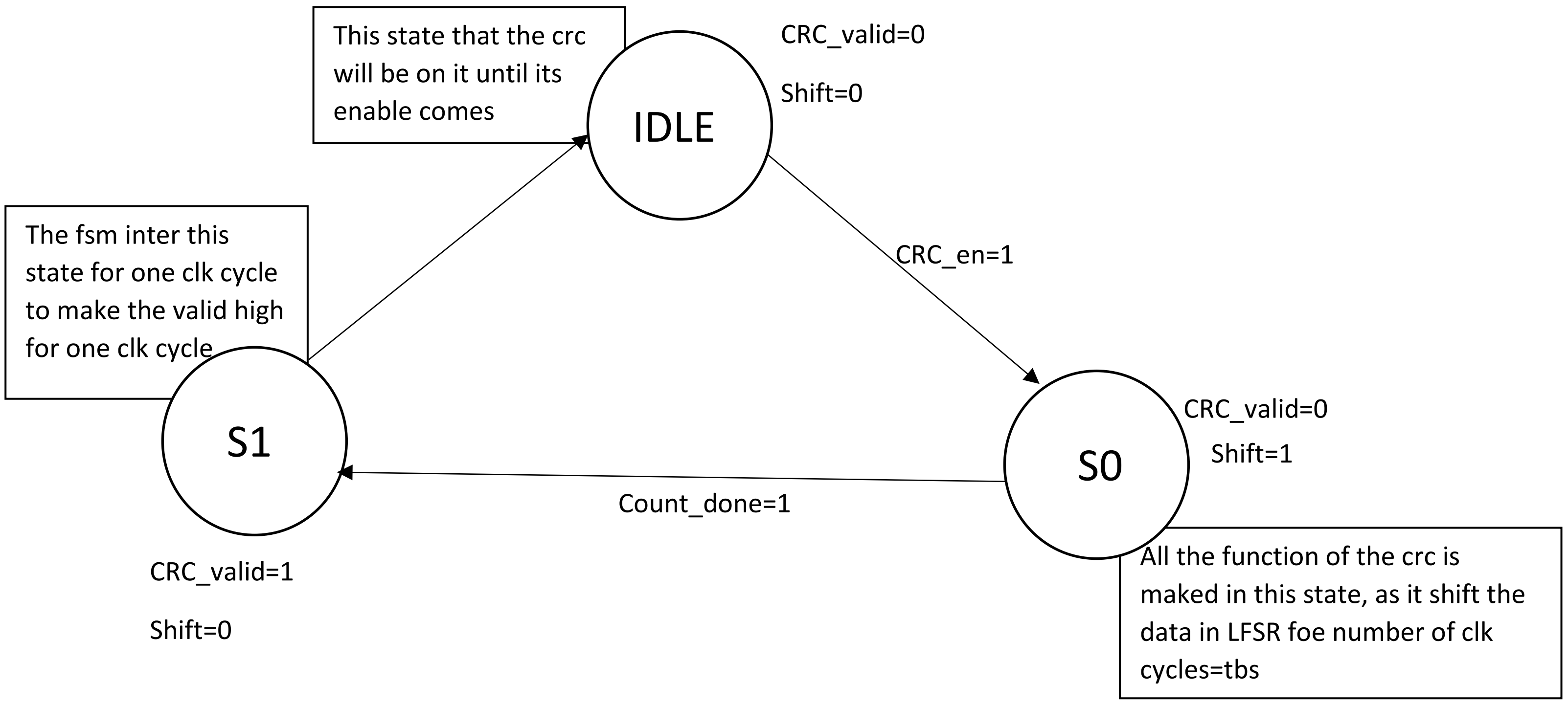
Port	Direction	Width	Description
clk	input	1 bit	System clk
rst_n	input	1 bit	Block reset
data	input	1 bit	serial input data its size is TBS
CRC_En	input	1 bit	When its high, the CRC will start to work, always high while the block is working or for one clk cycle the system will work
TBS	input	12 bits	The length of data, comes from the upper layer, its max value is 2536
CRC_out	output	24 bits	the CRC bit comes out when valid is heigh, its name in the code is mem_elem
CRC_valid	output	1 bit	it represents the data is ready in the CRC and it becomes high for one clk cycle
Encode_init	output	6 bits	the last 6 bits in the CRC bit we out this bit in parallel to the encoder

Table 1: Port description of CRC

4.1.3. Operation

- First, initialize the register with zero values.
- Second, we have the equation of LFSR so we designed the shift register as described in chapter 3.
- Third, shift the data by TBS
- Fourth, send the bits found in the reg to the RAM to shift them after the data, and the last 6 bit of output is also sent to the encoder to initialize its registers

4.1.4. FSM


This state that the crc will be on it until its enable comes
IDLE
CRC_valid=0
Shift=0
CRC_en=1
The fsm inter this state for one clk cycle to make the valid high for one clk cycle
S1
S0
CRC_valid=0
Shift=1
Count_done=1
CRC_valid=1
Shift=0
All the function of the crc is maked in this state, as it shift the data in LFSR foe number of clk cycles=tbs

State Table

Current State	Status signal	Next State	Control Signals
IDLE	If (CRC_en==1)	S0	CRC_valid=0 , Shift=0
S0	If (count_done==1)	S1	CRC_valid=0 , Shift=1
S1	(Next Cycle)	IDLE	CRC_valid=1 , Shift=0

Table 2: State Table of CRC FSM

4.1.5. Simulation results

1) Comparing our MATLAB function with the built-in function
Here we generate the data stream and calculate its CRC values then compare this bit with the output of the built-in function

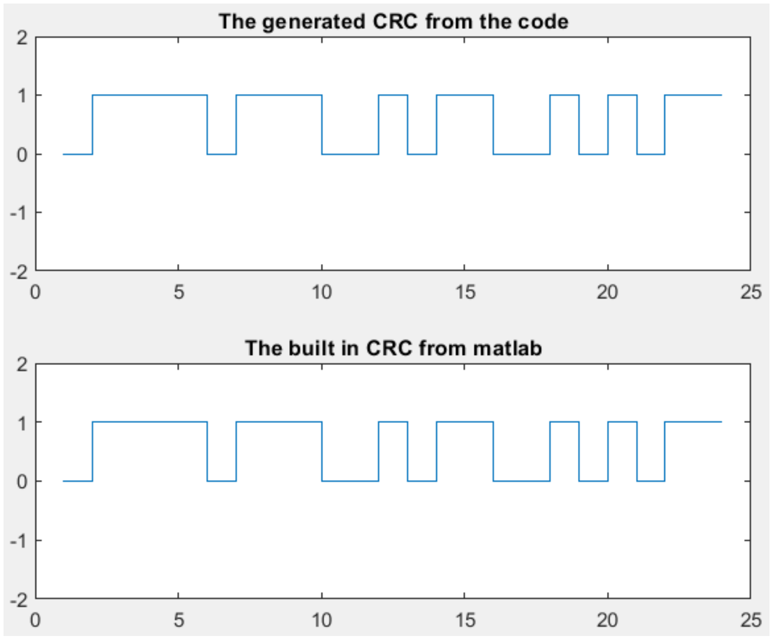
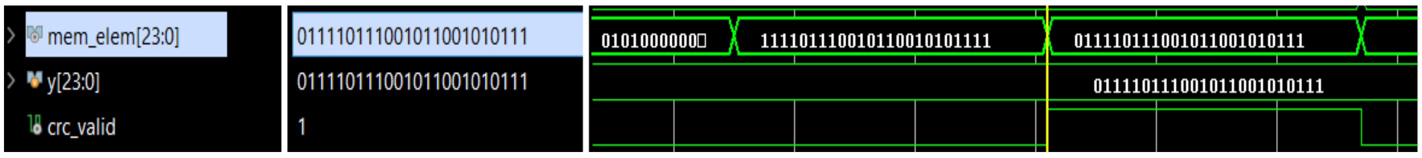


Figure 35: Generated & built-in MATLAB functions comparison of CRC

2) Comparing MATLAB with RTL

As shown in the figure below, when the CRC_valid is high so the data in mem_elem is the correct CRC bits (the first signal), and y signal contain the CRC bits that generated from MATLAB so they are the same.

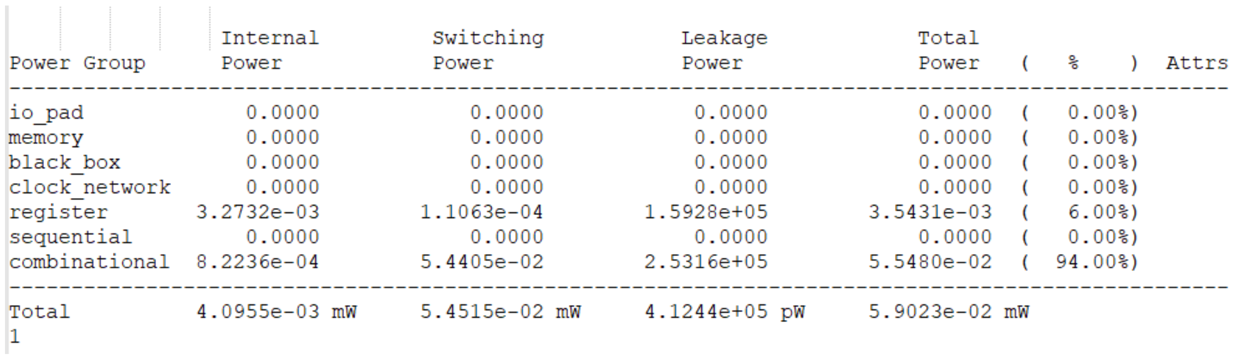


```
mem_elem[23:0]   011110111001011001010111   01010000000  111101110010110010101111  011110111001011001010111
y[23:0]          011110111001011001010111                                           011110111001011001010111
crc_valid        1
```

Figure 36: Correct CRC bits in RTL as MATLAB

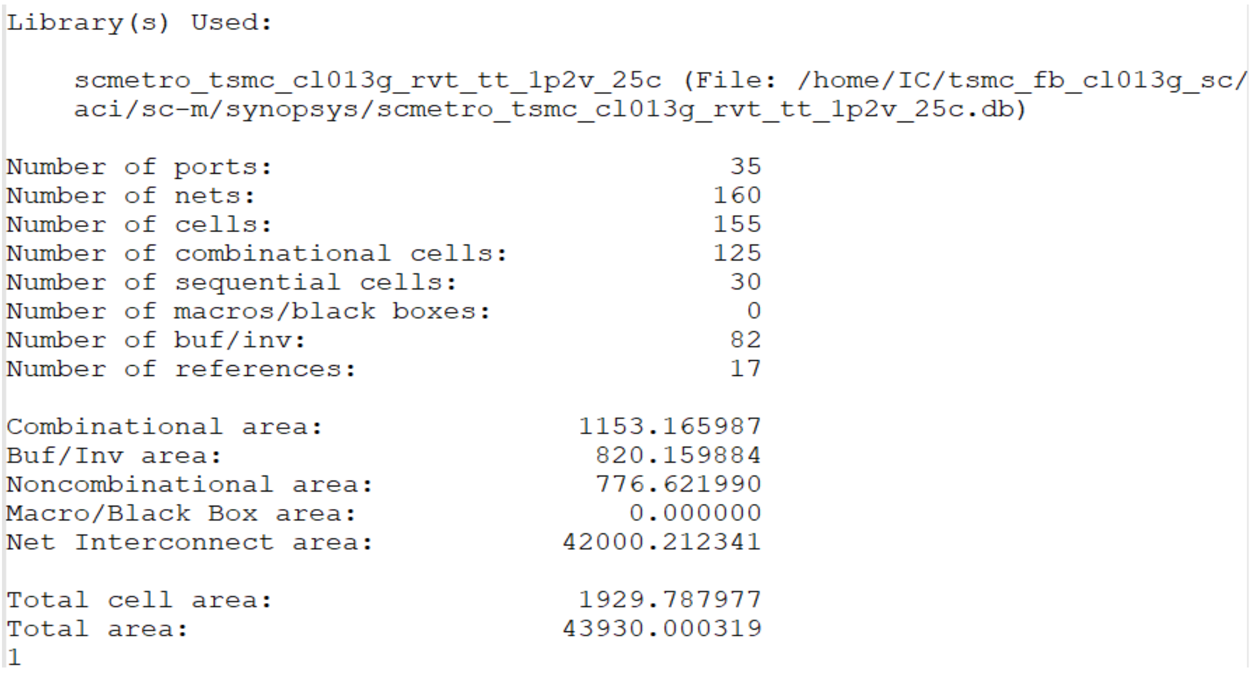
Finaly we generate multiple test cases to cover all the corner cases in matlab and compare it with verilog code outputs

4.1.6. Synthesis Results



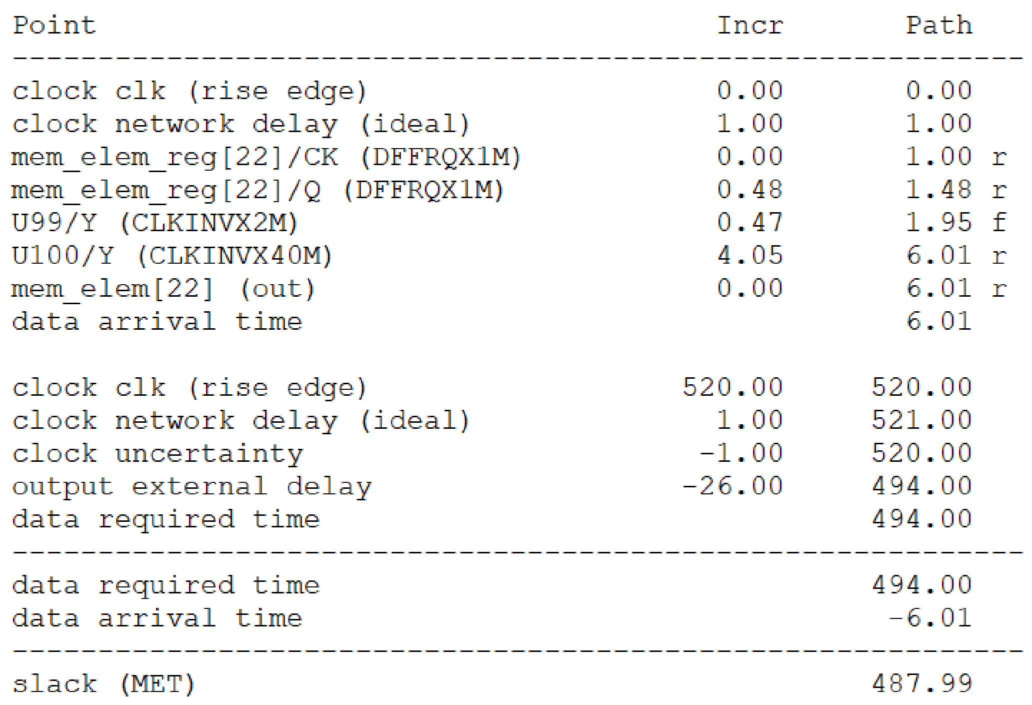
```
                 Internal          Switching           Leakage             Total
Power Group      Power             Power               Power               Power      (   %    )  Attrs
---------------------------------------------------------------------------------------------------
io_pad             0.0000            0.0000              0.0000              0.0000  (   0.00%)
memory             0.0000            0.0000              0.0000              0.0000  (   0.00%)
black_box          0.0000            0.0000              0.0000              0.0000  (   0.00%)
clock_network      0.0000            0.0000              0.0000              0.0000  (   0.00%)
register         3.2732e-03        1.1063e-04          1.5928e+05          3.5431e-03  (   6.00%)
sequential         0.0000            0.0000              0.0000              0.0000  (   0.00%)
combinational    8.2236e-04        5.4405e-02          2.5316e+05          5.5480e-02  (  94.00%)
---------------------------------------------------------------------------------------------------
Total            4.0955e-03 mW     5.4515e-02 mW       4.1244e+05 pW       5.9023e-02 mW
1
```

Figure 37: CRC Power report



```
Library(s) Used:

    scmetro_tsmc_cl013g_rvt_tt_1p2v_25c (File: /home/IC/tsmc_fb_cl013g_sc/
    aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_tt_1p2v_25c.db)

Number of ports:                               35
Number of nets:                               160
Number of cells:                              155
Number of combinational cells:                125
Number of sequential cells:                    30
Number of macros/black boxes:                   0
Number of buf/inv:                             82
Number of references:                          17

Combinational area:                   1153.165987
Buf/Inv area:                          820.159884
Noncombinational area:                 776.621990
Macro/Black Box area:                    0.000000
Net Interconnect area:               42000.212341

Total cell area:                      1929.787977
Total area:                          43930.000319
1
```

Figure 38: CRC Area report



```
Point                                          Incr       Path
-----------------------------------------------------------------
clock clk (rise edge)                          0.00       0.00
clock network delay (ideal)                    1.00       1.00
mem_elem_reg[22]/CK (DFFRQX1M)                 0.00       1.00 r
mem_elem_reg[22]/Q (DFFRQX1M)                  0.48       1.48 r
U99/Y (CLKINVX2M)                              0.47       1.95 f
U100/Y (CLKINVX40M)                            4.05       6.01 r
mem_elem[22] (out)                             0.00       6.01 r
data arrival time                                         6.01

clock clk (rise edge)                        520.00     520.00
clock network delay (ideal)                    1.00     521.00
clock uncertainty                             -1.00     520.00
output external delay                        -26.00     494.00
data required time                                      494.00
-----------------------------------------------------------------
data required time                                      494.00
data arrival time                                        -6.01
-----------------------------------------------------------------
slack (MET)                                             487.99
```

Figure 39: Timing_max (setup time) report of CRC

4.1.7. Latency

The number of clock cycles from the enable signal to the valid signal to be high = TBS

4.2. Encoder

4.2.1. Block Diagram

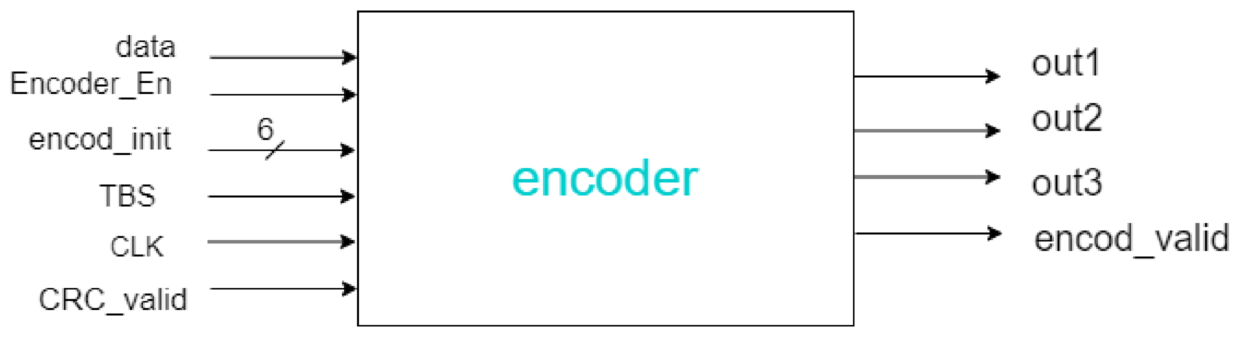


Figure 40: Tail-bit convolutional encoder top level

4.2.2. Interface Table

Port	Direction	Width	Description
CLK	input	1 bit	System clock
rst_n	input	1 bit	Block reset
data	input	1 bit	Serial input data its size is TBS comes from the serializer
Encoder_En	input	1 bit	When its high the encoder will work, can work for one clock cycle or more but it should be high after the crc_valid signal is high
Encoder_initial	input	6 bits	The last 6 bits in the CRC bit we take this bit in parallel to the encoder, they are ready when crc_valid is high
TBS	input	12 bits	The length of data comes from the upper layer; its max value is 2536
CRC_valid	input	1 bit	it represents the data is ready in the CRC and it becomes high for one clock cycle
Out1	output	1 bit	The first encoded output (G1)
Out2	output	1 bit	The second encoded output (G2).
Out3	output	1 bit	The third encoded output (G3).
Encoder_valid	output	1 bit	Valid signal becomes high when the data is ready to come out, becomes high after 1 clock from the encoder_en

Table 3: Port description table of Tail-bit Conv. encoder

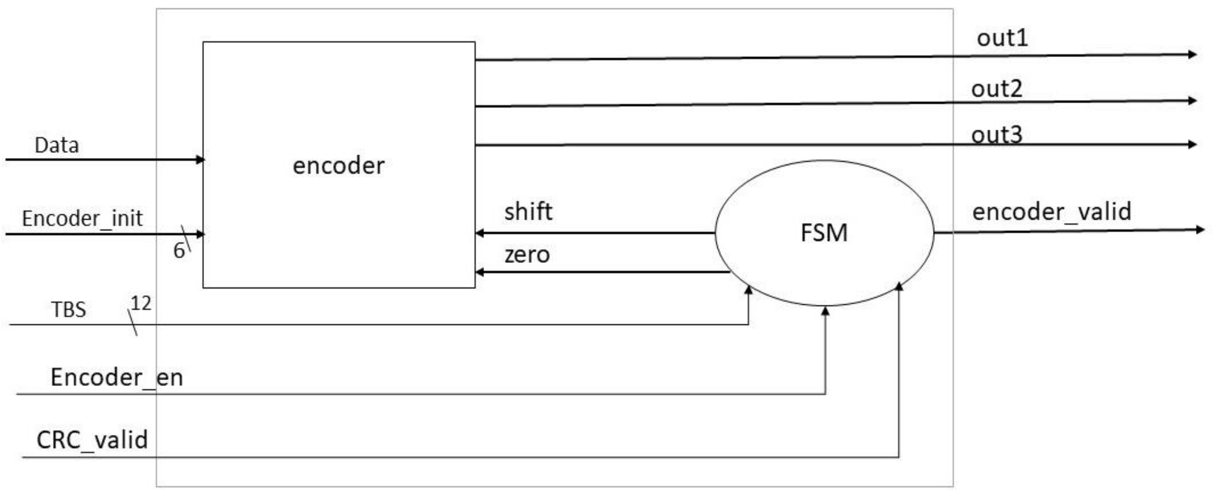
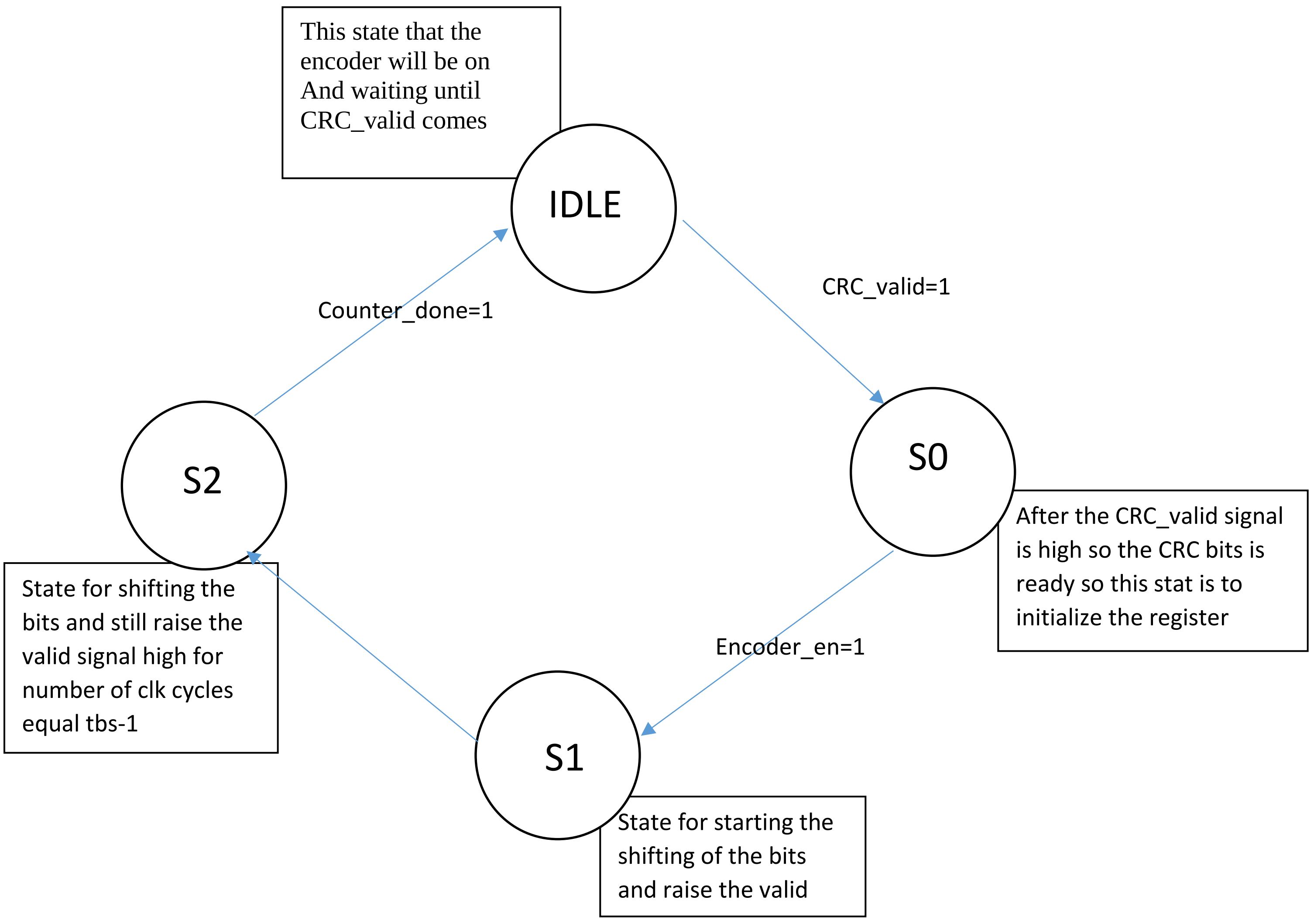


Figure 41: tail-bit conv. encoder architecture

4.2.3. FSM


This state that the encoder will be on And waiting until CRC_valid comes
IDLE
CRC_valid=1
Counter_done=1
S2
S0
After the CRC_valid signal is high so the CRC bits is ready so this stat is to initialize the register
State for shifting the bits and still raise the valid signal high for number of clk cycles equal tbs-1
Encoder_en=1
S1
State for starting the shifting of the bits and raise the valid

Current State	**Status signal**	**Next State**	**Control Signals**

IDLE	If (crc_valid==1)	S0	Shift = 0
			Zero = 0
			encoder_valid = 0

S0	If (encoder_en==1)	S1	Shift = 0
			Zero = 1
			encoder_valid = 0

S1	(Next Cycle)	S2	Shift = 1
			Zero = 0
			encoder_valid = 0

S2	If (count_done==1)	IDLE	Shift = 1
			Zero = 0;
			encoder_valid = 1

Table 4: State Table of the encoder's FSM

4.2.4. Operation

- First, initialize the register with last 6 bits of data
- Second, shift the data by the number of the bits equal TBS and raising the valid signal high.
- Third, the data enter serially so each clock cycle one bit entered and three bits comes out until all data out from encoder

4.2.5. Simulation results

1) Comparing MATLAB with built in

Here we generate the data stream and calculate three encoder output streams values then compare this stream with the output of the built-in function then print it in a file, and this is the output of the comparison in MATLAB

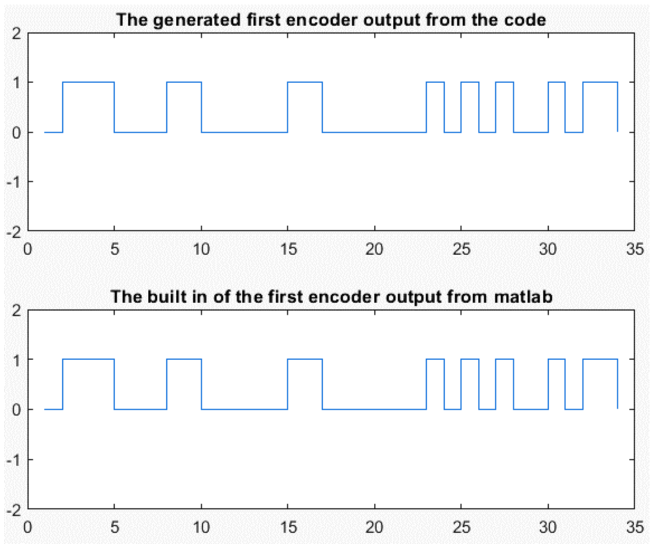


Figure 43: Generated & built-in first encoder output stream

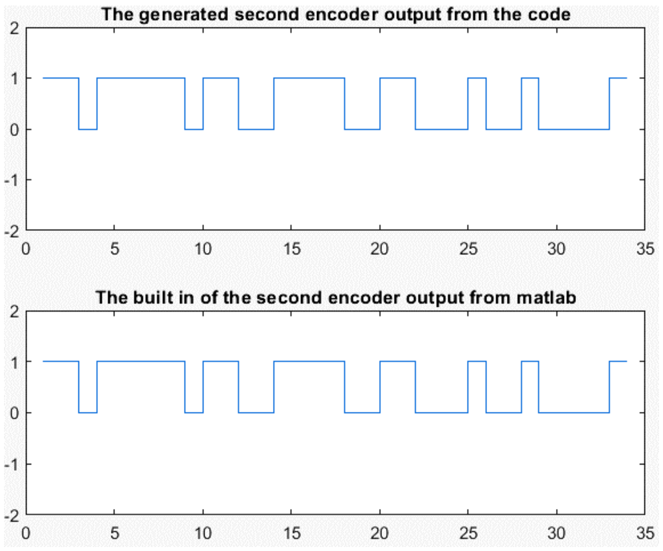


Figure 42: Generated & built-in second encoder output stream

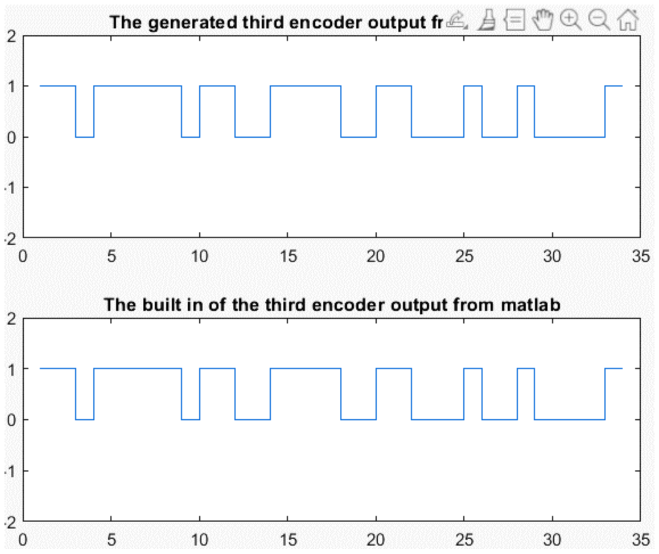
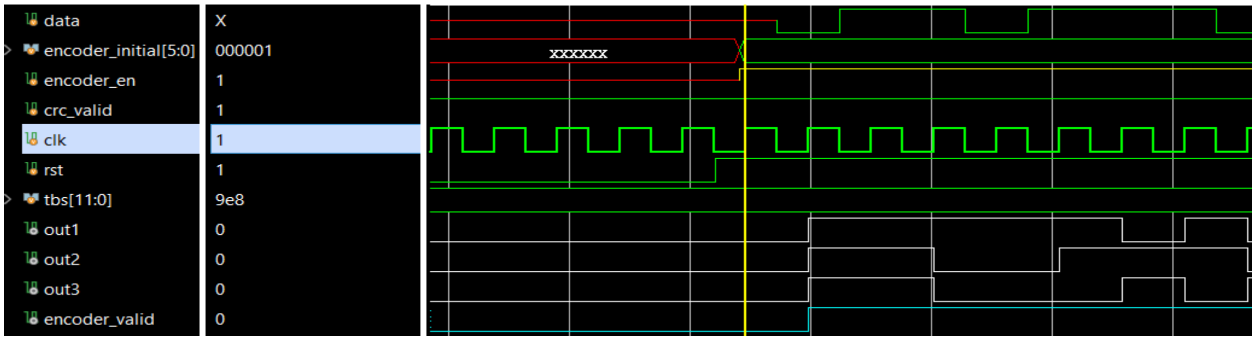


Figure 44: Generated & built-in third encoder output stream

2) Comparing matlab with verilog

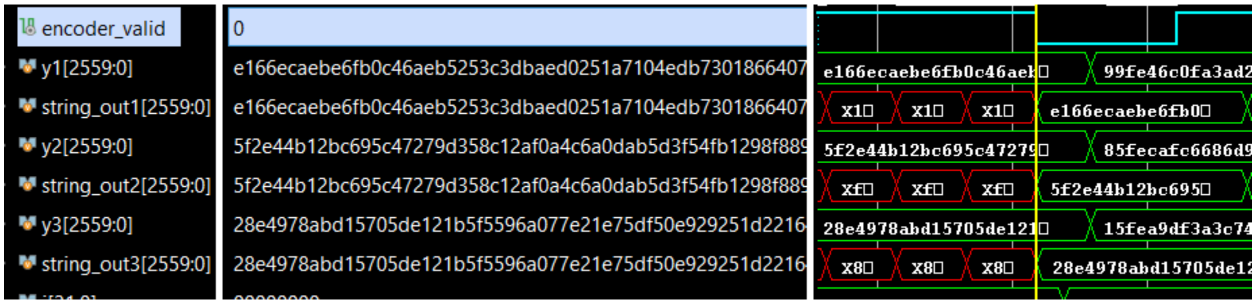
The figure below shows that the block started to generate the three outputs and assert the encoder_valid high after one clock cycle from the encoder_en as expected



data X
encoder_initial[5:0] 000001
encoder_en 1
crc_valid 1
clk 1
rst 1
tbs[11:0] 9e8
out1 0
out2 0
out3 0
encoder_valid 0

Figure 45: Encoder output streams and valid signal

As the data output is serial, so we collect **out1** in **string_out1** and **out2** in **string_out2** and **out3** in **string_out3** and compare when the encoder_valid signal comes low the **string_out** with the data generated from MATLAB, so we compare **string_out1** with **y1** and so on.

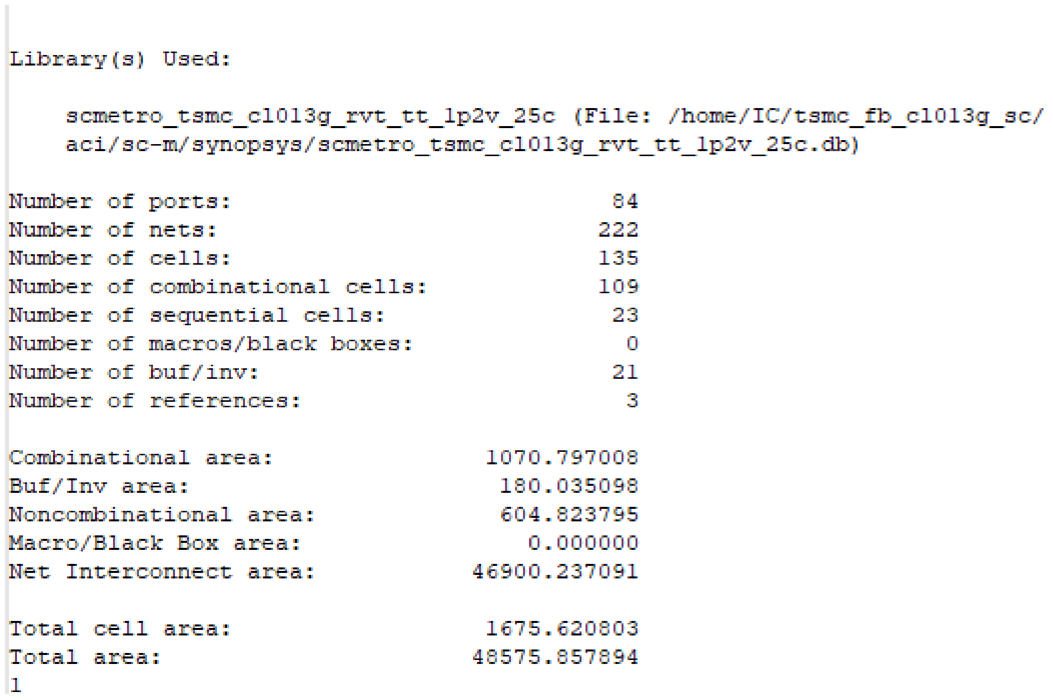


encoder_valid 0
y1[2559:0] e166ecaebe6fb0c46aeb5253c3dbaed0251a7104edb7301866407
string_out1[2559:0] e166ecaebe6fb0c46aeb5253c3dbaed0251a7104edb7301866407
y2[2559:0] 5f2e44b12bc695c47279d358c12af0a4c6a0dab5d3f54fb1298f889
string_out2[2559:0] 5f2e44b12bc695c47279d358c12af0a4c6a0dab5d3f54fb1298f889
y3[2559:0] 28e4978abd15705de121b5f5596a077e21e75df50e929251d2216
string_out3[2559:0] 28e4978abd15705de121b5f5596a077e21e75df50e929251d2216

Figure 46: Comparing MATLAB and RTL results

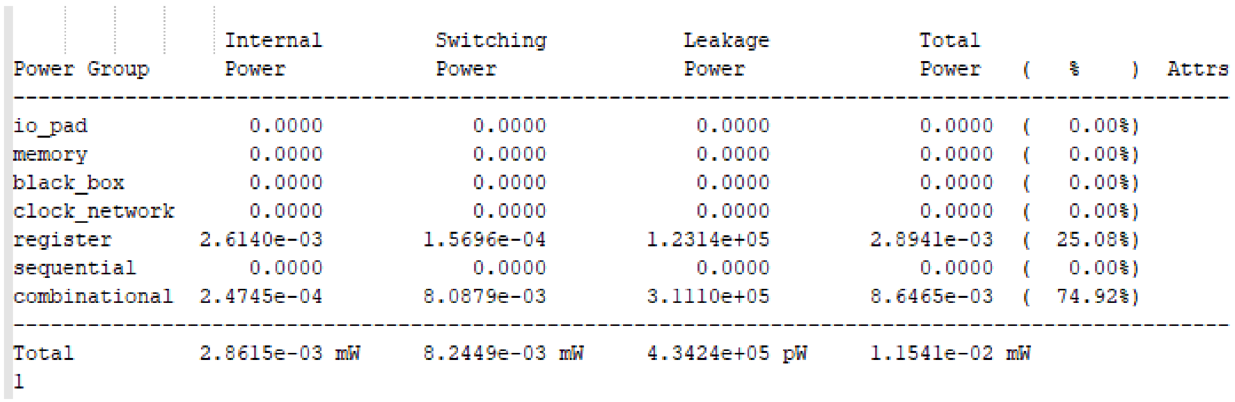
Finaly we generate multiple test cases to cover all the corner cases in matlab and compare it with verilog code outputs

4.2.6. Synthesis Results



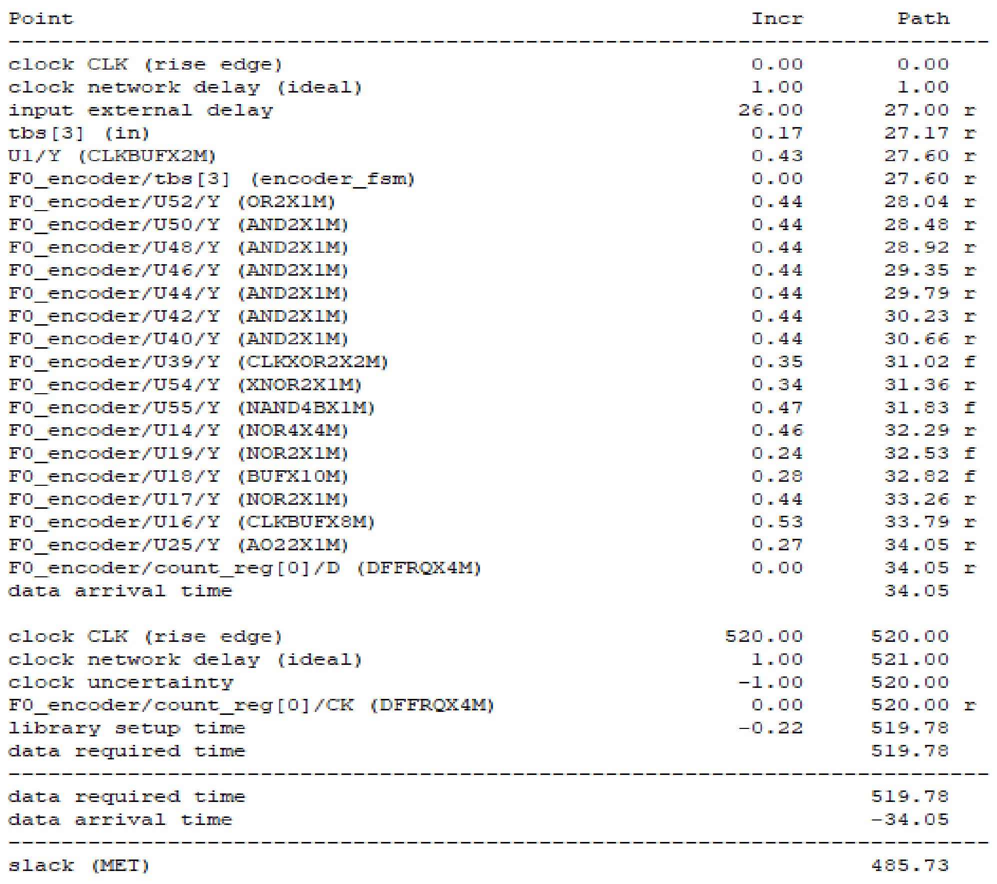
```
Library(s) Used:

    scmetro_tsmc_cl013g_rvt_tt_1p2v_25c (File: /home/IC/tsmc_fb_cl013g_sc/
    aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_tt_1p2v_25c.db)

Number of ports:                           84
Number of nets:                           222
Number of cells:                          135
Number of combinational cells:            109
Number of sequential cells:                23
Number of macros/black boxes:               0
Number of buf/inv:                         21
Number of references:                       3

Combinational area:               1070.797008
Buf/Inv area:                      180.035098
Noncombinational area:             604.823795
Macro/Black Box area:                0.000000
Net Interconnect area:           46900.237091

Total cell area:                  1675.620803
Total area:                      48575.857894
1
```

Figure 47: Encoder Area report



```
                Internal        Switching           Leakage          Total
Power Group     Power           Power               Power            Power    (   %    )  Attrs
---------------------------------------------------------------------------------------------------
io_pad            0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory            0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box         0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network     0.0000           0.0000            0.0000            0.0000  (   0.00%)
register       2.6140e-03       1.5696e-04        1.2314e+05       2.8941e-03  (  25.08%)
sequential        0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational  2.4745e-04       8.0879e-03        3.1110e+05       8.6465e-03  (  74.92%)
---------------------------------------------------------------------------------------------------
Total          2.8615e-03 mW    8.2449e-03 mW     4.3424e+05 pW    1.1541e-02 mW
1
```

Figure 48: Encoder power report



```
Point                                                   Incr       Path
-----------------------------------------------------------------------
clock CLK (rise edge)                                   0.00       0.00
clock network delay (ideal)                             1.00       1.00
input external delay                                   26.00      27.00 r
tbs[3] (in)                                             0.17      27.17 r
U1/Y (CLKBUFX2M)                                        0.43      27.60 r
F0_encoder/tbs[3] (encoder_fsm)                         0.00      27.60 r
F0_encoder/U52/Y (OR2X1M)                               0.44      28.04 r
F0_encoder/U50/Y (AND2X1M)                              0.44      28.48 r
F0_encoder/U48/Y (AND2X1M)                              0.44      28.92 r
F0_encoder/U46/Y (AND2X1M)                              0.44      29.35 r
F0_encoder/U44/Y (AND2X1M)                              0.44      29.79 r
F0_encoder/U42/Y (AND2X1M)                              0.44      30.23 r
F0_encoder/U40/Y (AND2X1M)                              0.44      30.66 r
F0_encoder/U39/Y (CLKXOR2X2M)                           0.35      31.02 f
F0_encoder/U54/Y (XNOR2X1M)                             0.34      31.36 r
F0_encoder/U55/Y (NAND4BX1M)                            0.47      31.83 f
F0_encoder/U14/Y (NOR4X4M)                              0.46      32.29 r
F0_encoder/U19/Y (NOR2X1M)                              0.24      32.53 f
F0_encoder/U18/Y (BUFX10M)                              0.28      32.82 f
F0_encoder/U17/Y (NOR2X1M)                              0.44      33.26 r
F0_encoder/U16/Y (CLKBUFX8M)                            0.53      33.79 r
F0_encoder/U25/Y (AO22X1M)                              0.27      34.05 r
F0_encoder/count_reg[0]/D (DFFRQX4M)                    0.00      34.05 r
data arrival time                                                 34.05

clock CLK (rise edge)                                 520.00     520.00
clock network delay (ideal)                             1.00     521.00
clock uncertainty                                      -1.00     520.00
F0_encoder/count_reg[0]/CK (DFFRQX4M)                   0.00     520.00 r
library setup time                                     -0.22     519.78
data required time                                               519.78
-----------------------------------------------------------------------
data required time                                               519.78
data arrival time                                                -34.05
-----------------------------------------------------------------------
slack (MET)                                                      485.73
```

Figure 49: Encoder Timing_max (setup time) report

4.2.7. Latency

The number of clock cycles from the enable signal to the valid signal to be high = 1

4.3. Rate Matcher

We have designed the rate matcher block through two designs which are slightly different in some features, hardware and algorithm.

The Second Design is more optimized and improved than the first design.

4.3.1. Block Diagram

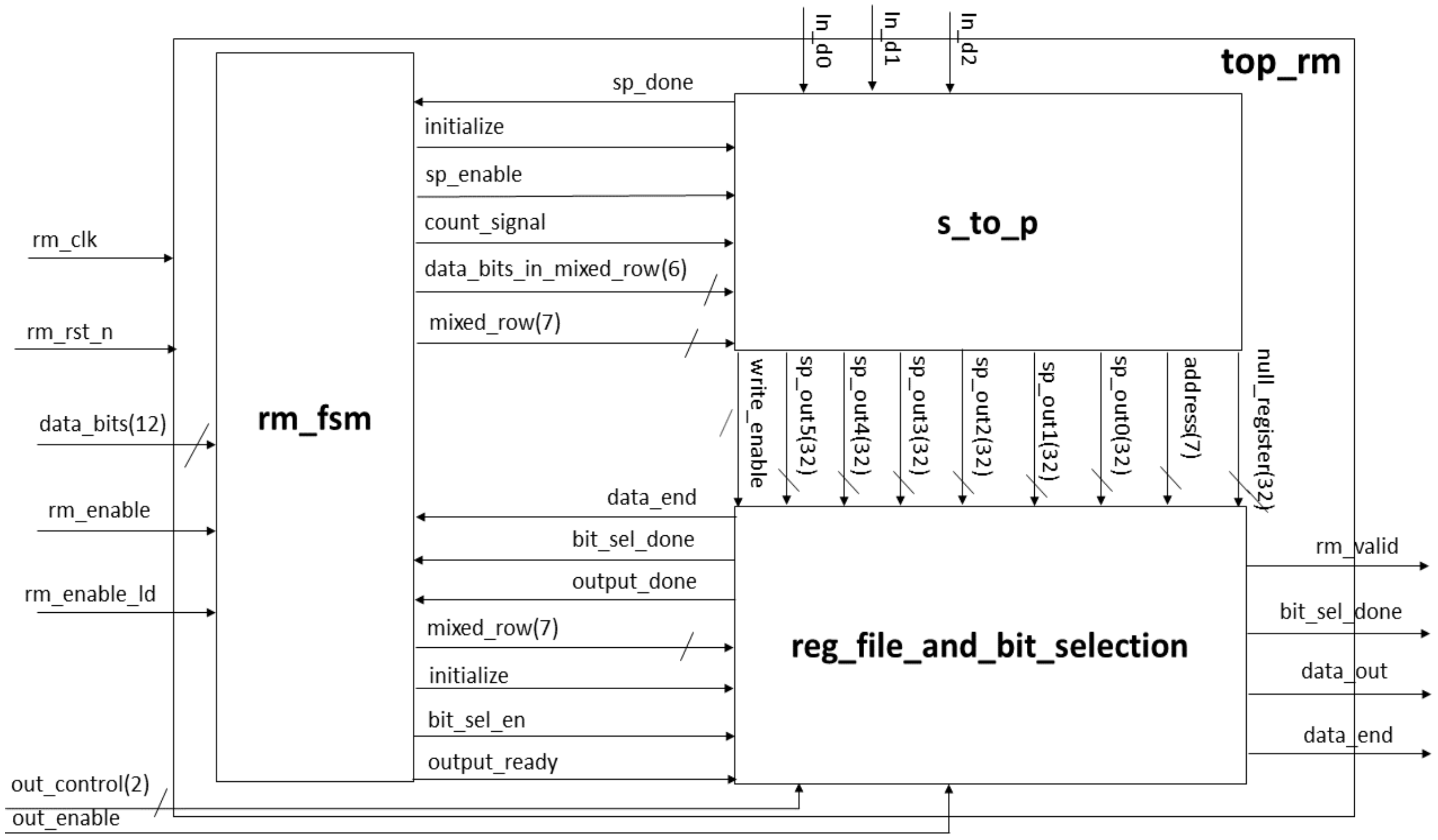


Figure 50: Rate Matcher main architecture

4.3.2. Interface Table

Port	Direction	Width	Description
rm_clk	Input	1 bit	Operating clock.
rm_rst_n	Input	1 bit	Reset the whole block.
rm_enable_ld	Input	1 bit	Enable block to read input stream (Valid from tail biting encoder)
In_d0	Input	1 bit	First input stream which is d0 output from the tail biting convolution encoder.
In_d1	Input	1 bit	Second input stream which is d1 output from the tail biting convolution encoder.
In_d2	Input	1 bit	Third input stream which is d2 output from the tail biting convolution encoder.
data_bits	Input	12 bits	Upper layer parameter to determine transport block size in addition to 24 bits of CRC.
out_enable	Input	1 bit	Signal that enables output stream to get out
out_control	Input	2 bits	Determine if output stream will be 24 bits ➔ 2'b11 20 bits ➔ 2'b01
data_out	Output	1 bit	Output bit stream from the block.
rm_valid	Output	1 bit	Valid output enables for the next block to start reading the RM output.
data_end	Output	1 bit	Indicates that all data bits are got out and the block is waiting for new input stream.
bit_sel_done	Output	1 bit	Indicates that the selection of 24 bits or 20 bits is done and the block is ready to get out the output data

Table 5: Rate Matcher Interface Table

4.3.3. General Steps of Operation

1. **Serial to parallel and interleaving**
   In this step, input stream from tail bit encoder is registered in 32-bit register according to look-up table defined in the standard to allow interleaving of input bits.
2. **Bit collection**
   Every 32 bits which are interleaved and registered is then send to written in a register file (80*32)
   So, we have 3 register files for 3 input streams

Maximum TBS is 2536 bits in addition to 24 bits CRC, then maximum depth*width of register file is 80*32 = 2560.

3. **Bit selection**
   In this step, we select data bits from the register file column by column and get rid of dummy bits which were padded before in the register file to complete the empty places in each register file.
   NOTE ➔ the difference between the first design and the second one is mainly based on this step and how to optimize it.
4. **Output stage**

In this step, output is ready to get out and wait for enable signal, also control signal is important here to inform the block about number of bits needed to get out "24 bits or 20 bits" according to demand of the system.

4.3.4. First design

We will talk briefly about the first design and its idea of operation then; we will go to the second design as it is the final design we have reached and the more optimized and efficient design.

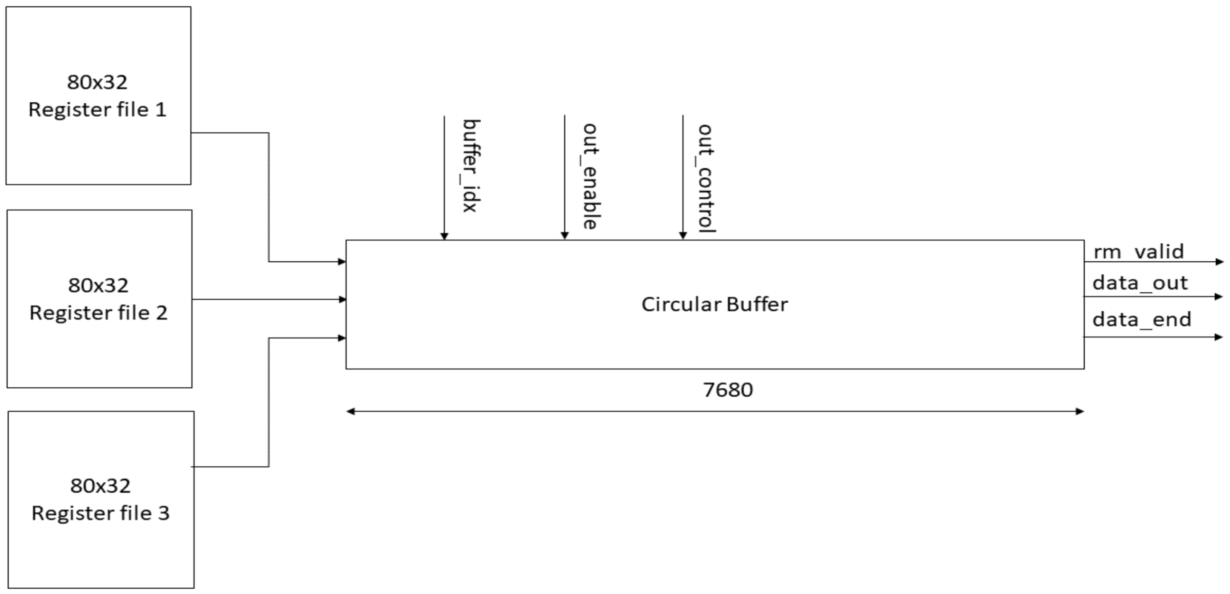


Figure 51: Illustrative architecture of first design

4.3.4.1. Operation

1. After bit collection stage, when all data bits are saved in the register files and the input streams from the encoder have been finished, bit selection stage starts.
2. Data is selected from the first register file bit by bit each clock cycle column by column.
3. The first bit is selected from the row which includes both data bits and dummy bits and we differentiate between them using the null register which we have designed in the first stage of the rate matcher block, In short, the null register is 32 bits register corresponds to the mixed

row in each register file *"the row which includes both data bits and dummy bits",* if any bit of the 32 bits is "1" therefore there is a data bit in the corresponding index and if any bit of the 32 bits is "0" therefore there is a dummy bit in the corresponding index and we will not select it. Look at figure 53.

4. After the selection from the register file, we repeat the previous steps with register file 2 then 3 and all selected bits are saved in the circular buffer from the least significant bit to the most one.
5. When output enable becomes high ➔ rm_valid gets high and data started to get out according to the value of out_control
6. Output data is 24 serial bits stream or 20 bits only, this according to the LTE narrow band IoT resource block which demands 12 QPSK symbols in the first 5 OFDM symbols and 10 QPSK symbols with 2 reference signals symbols in OFDM symbol 6 and 7. Look at figure 52

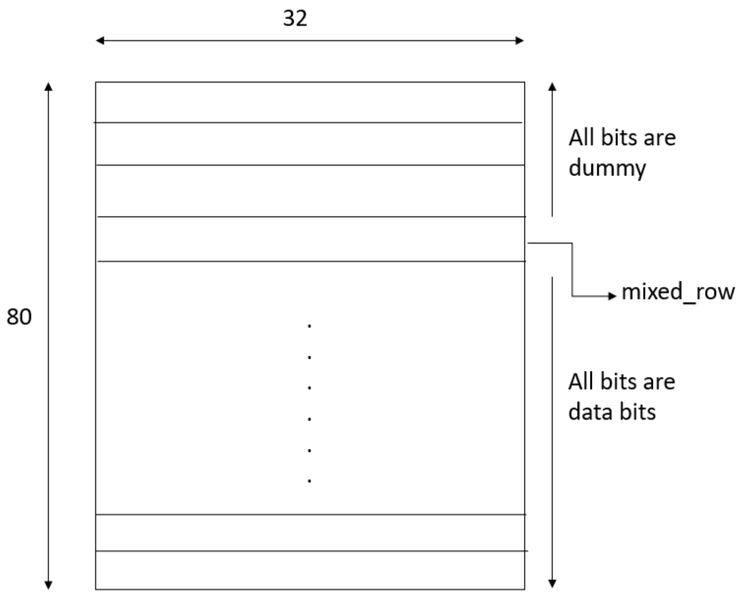


Figure 53: Register File Implementation

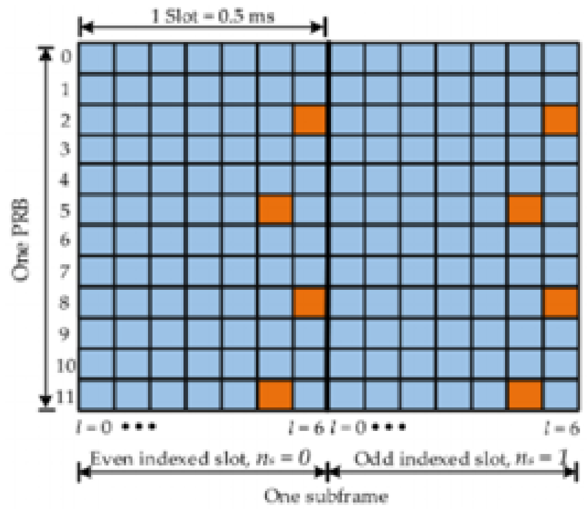


Figure 52: Resource Allocation

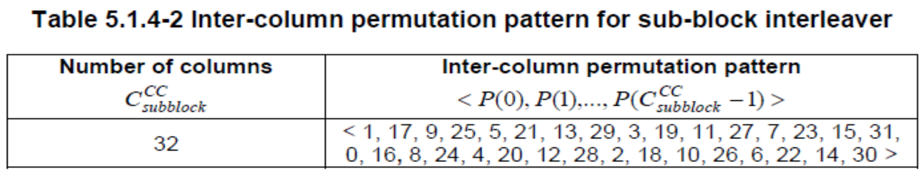


Table 5.1.4-2 Inter-column permutation pattern for sub-block interleaver

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

Figure 54: The used Permutation look-up table

4.3.4.2. Disadvantages of the design

i. ***The large latency***

Latency of design #1 = (TBS + 24) + (3*(TBS+24))

➔ The first term is during the input bits stream which we can't deal with.

➔ The second term is due to the bit selection from the register files and saving data bits in the large circular buffer from 3 register files.

ii. ***The large area and power***

➔ The circular buffer is a large register consists of 7680 flip flops in addition to the 3 register files, so we have about 15k flip flops in the design.

➔ Also, number of LUTs is large appears after synthesizing this block.

4.3.5. Second design

4.3.5.1. Idea of optimization

- We have a problem from the first design regarding the large latency appears in bit selection stage in addition to the large area represented in the large circular.

- We also discovered that the IFFT block *"the last block in our system's chain"* needs about 137 clock cycles to operate on 1 OFDM symbol only and during these clock cycles all blocks in the chain are halted waited for IFFT to finish.

- So, we can benefit from this by selecting and registering the output bits either 24 or 20 bits from the register files in small registers for a maximum 30 clock cycles instead of the large circular buffer and wait for IFFT to finish then enable signal becomes high to get out the output then registering the data bits again in small registers and wait and so on.

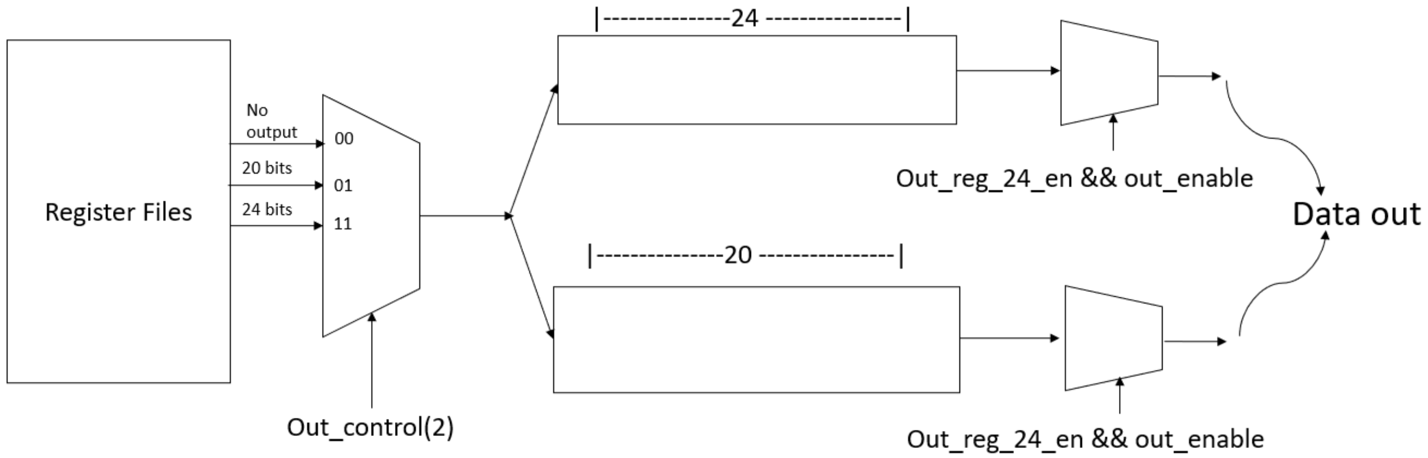


Figure 55: Illustrative architecture of second design

4.3.5.2. Operation

1) The s_to_p module

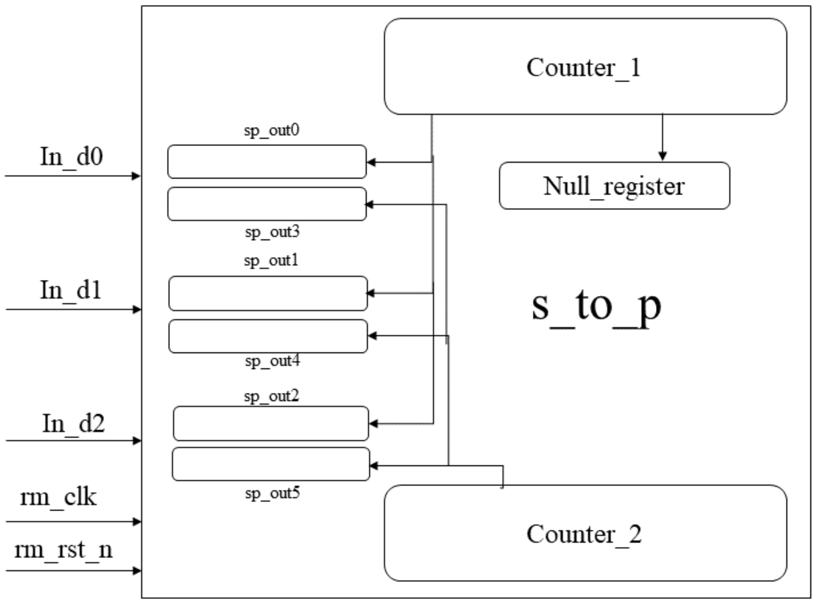


Figure 56: Serial-to-parallel module

1. Data input streams are registered in 6 registers (2 for each stream, one for the first row in the register file which is the mixed row including data bits and dummy bits)
   - ➔ Sp_out0, sp_out1 and sp_out2 are for the mixed row using counter 1 counts from 0 to the number of data bits in the mixed row
   - ➔ Sp_out0, sp_out1 and sp_out2 are for the rest below rows in the register files using counter 2 counts from 0 to 32 until data input streams are finished
2. Registering data input bits is based on the look-up table.
3. At the same time of counter 1 counting, the bit of the corresponding location in null register becomes 1.
4. At the beginning address is equal to the position of the mixed row then it is incremented.
5. When write enable is high, we wrote in the register files.

2) Register files and bit selection module

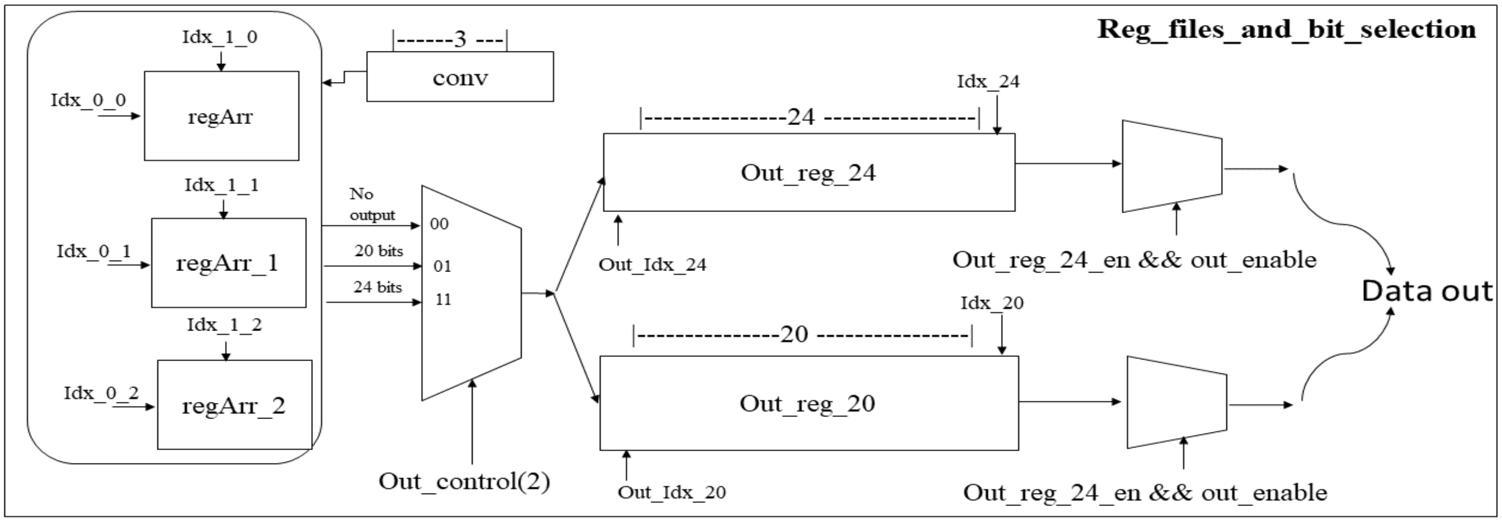


Figure 57: Register files and bit selection module

1. After the input data are completely registered in the register files and we ready to select data bits, we wait until out_control signal changes its value to 01 ➔ select bits and save them in the 24-bit registers or 11 ➔ select bits and save them in the 20-bits register.
2. The block then is halted until out_enable becomes high for 1 clock cycle then rm_valid becomes high and data_out stream gets data out either from the 24-bit register or 20-bit register according to:
   i. Out_reg_24_en ➔ data is got out from 24-bit register.
   ii.Out_reg_20_en ➔ data is got out from 20-bit register.
3. Each register file *"regArr, regArr_1 and regArr_2"* has index for column and index for rows which are incremented during selection of bits
4. Idx_0_0, Idx_0_1 and Idx_0_2 ➔ each starts from the mixed row until row 79.
5. Idx_1_0, Idx_1,1 and Idx_1,2 ➔ each starts from column 0 to column 31.
6. When the first register file indices reach their last positions "idx_0_0 ➔ 79, Idx_1_0 ➔ 31, conv register changes it's value to "01" and selection from the second register file starts, when the second register file indices reach their last positions, conv changes it's value to "11" then to "10 after the third register file indices reach their last positions and by this all data has been selected.
7. ***bit_sel_en*** ➔ to enable the step of selection of data bits and get rid of null bits by the help of null register and register data bits in the 24- or 20-bits registers.
8. ***bit_sel_done*** ➔ when the step of bit selection is finished.
9. ***Output_ready*** ➔ from rate matcher finite state machine to be ready for output stage when output enable becomes high.
10. ***Output_done*** ➔ when all either 24 or 20 bits are got out.
11. ***Data_end*** ➔ signal sent to finite state machine when all data in the register files have got out to go to IDLE state to be ready for a new input stream.

3) The Finite State Machine

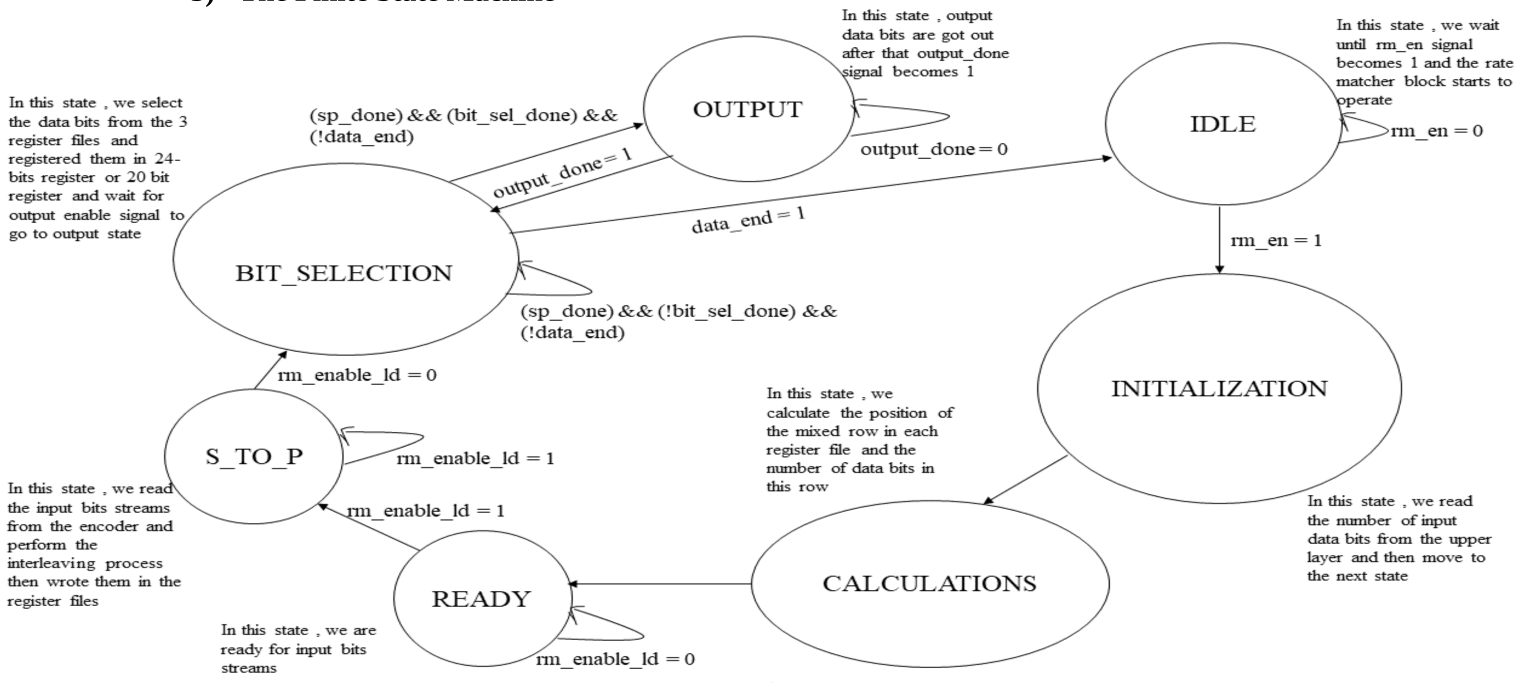

OUTPUT
IDLE
BIT_SELECTION
INITIALIZATION
S_TO_P
READY
CALCULATIONS
In this state , output data bits are got out after that output_done signal becomes 1
In this state , we wait until rm_en signal becomes 1 and the rate matcher block starts to operate
In this state , we select the data bits from the 3 register files and registered them in 24-bits register or 20 bit register and wait for output enable signal to go to output state
(sp_done) && (bit_sel_done) && (!data_end)
output_done = 1
output_done = 0
rm_en = 0
data_end = 1
rm_en = 1
(sp_done) && (!bit_sel_done) && (!data_end)
rm_enable_ld = 0
rm_enable_ld = 1
rm_enable_ld = 1
rm_enable_ld = 0
In this state , we calculate the position of the mixed row in each register file and the number of data bits in this row
In this state , we read the input bits streams from the encoder and perform the interleaving process then wrote them in the register files
In this state , we read the number of input data bits from the upper layer and then move to the next state
In this state , we are ready for input bits streams

Figure 58: Rate Matcher State Diagram

Current state	**Next state**	**Description of the current state**
IDLE	• INITIALIZATION ➔ if rm_en = 1 • IDLE ➔ if rm_en = 0	Waiting for the rate matcher block to operate or a new input stream is available.
INITIALIZATION	• CALCULATIONS	Reading data_bits which is the number of data input bits ➔ TBS + 24.
CALCULATIONS	• READY	Calculating:- • The position of the mixed row in the register files. • The number of data bits in the mixed row.
READY	• S_TO_P ➔ rm_enable_ld = 1 • READY ➔ rm_enable_ld = 0	The block is ready for input streams from tail bit convolutional encoder.
S_TO_P	• BIT_SELECTION ➔ rm_enable_ld = 0 • S_TO_P ➔ rm_enable_ld = 1	Convert the input streams from serial input to 32 parallel bits registers, registered according to look-up table then saved in the register files.
BIT_SELECTION	• BIT_SELECTION ➔ (sp_done) && (!bit_selection_done) && (!data_end) • OUTPUT ➔ (sp_done) && (bit_selection_done) && (!data_end) • IDLE ➔ data_end = 1	Selecting data bits from the register files and waiting for the output enable. Getting rid of the dummy bits during selection.
OUTPUT ➔ **(101)**	BIT_SELECTION ➔ output_done = 1	Data_bits are got out.

Table 6: Rate Matcher State Table

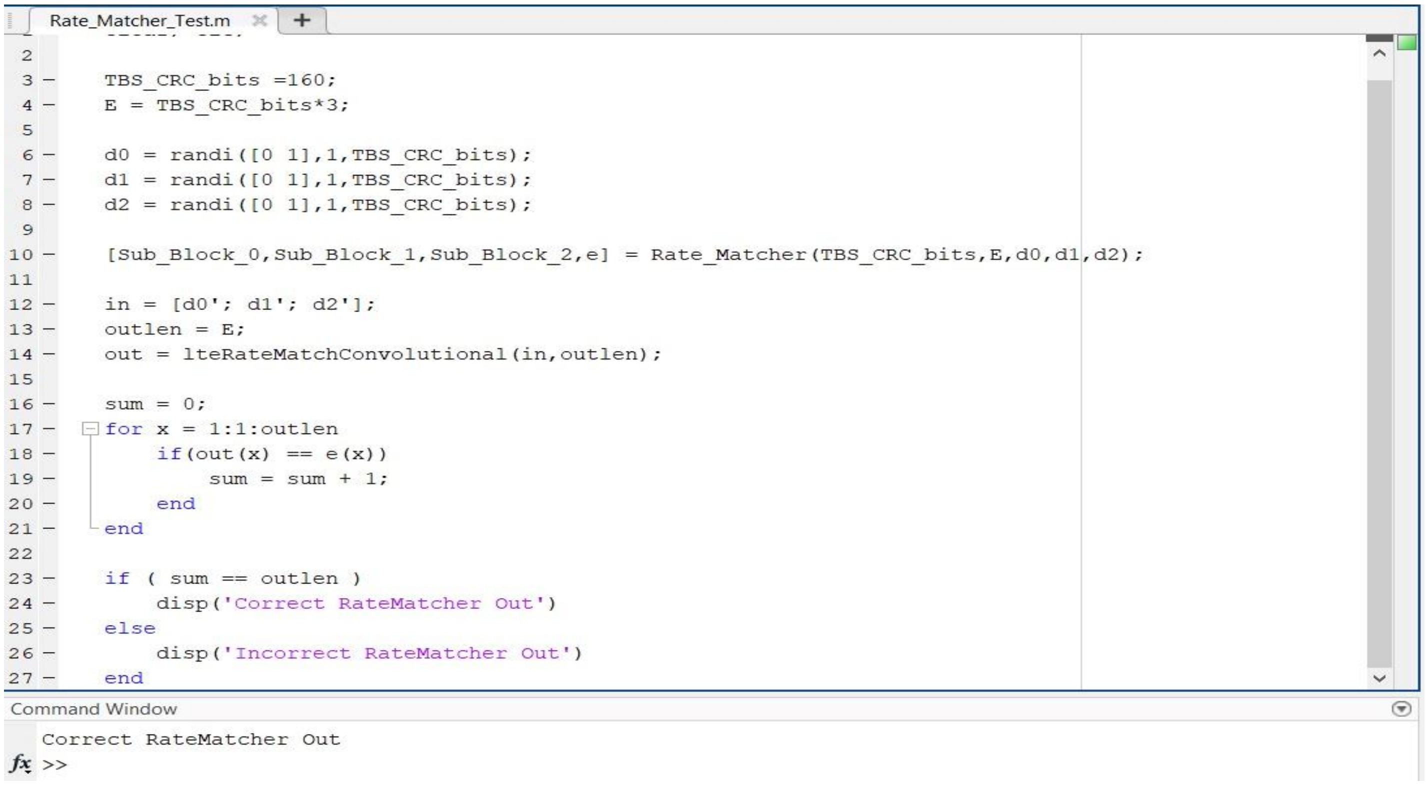
4.3.5.3. Enhancements and improvements in the second design: -

1. **The small latency**
   Compared to the first design latency here is very small.
   Latency of design #2 = (TBS + 24) + (24 or 25)

2. **The small area and power**
   As the absence of the large buffer allow us to have a maximum of 7k flip flips not 15k used in the register files also there is a large decrease in the number of LUTs.

4.3.6. Simulation results

We generate a simple MATLAB code define the algorithm of the rate matcher for convolutional encoding and generate a random bit stream *"d0, d1 and d2"*, then we compared the output from our generated code with the built-in function "**lteRateMatcherConvolutional**" and the result was that the two outputs are identical and our generated algorithm is correct.



Rate_Matcher_Test.m

```
TBS_CRC_bits =160;
E = TBS_CRC_bits*3;

d0 = randi([0 1],1,TBS_CRC_bits);
d1 = randi([0 1],1,TBS_CRC_bits);
d2 = randi([0 1],1,TBS_CRC_bits);

[Sub_Block_0,Sub_Block_1,Sub_Block_2,e] = Rate_Matcher(TBS_CRC_bits,E,d0,d1,d2);

in = [d0'; d1'; d2'];
outlen = E;
out = lteRateMatchConvolutional(in,outlen);

sum = 0;
for x = 1:1:outlen
    if(out(x) == e(x))
        sum = sum + 1;
    end
end

if ( sum == outlen )
    disp('Correct RateMatcher Out')
else
    disp('Incorrect RateMatcher Out')
end
```

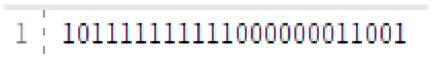
Command Window

```
Correct RateMatcher Out
>>
```

Figure 59: comparison between MATLAB built in and our code

RTL simulations

We have tested 2 different streams, the first input stream was 360 bits from the encoder and compared the output bits by the output bits from the MATLAB, after all data bits have got out and the block returned to IDLE state, we have tested another input stream which was 160 bits and applied the previous steps again.



```
1   101111111111000000011001
```

Figure 61: first 24 bits in MATLAB

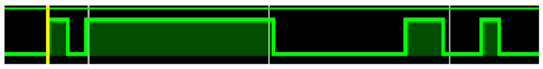
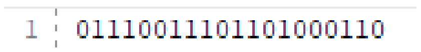


Figure 60: first 24 bits in RTL



```
1   01110011101101000110
```

Figure 63: first 20 bits in MATLAB

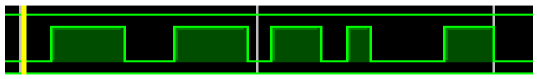
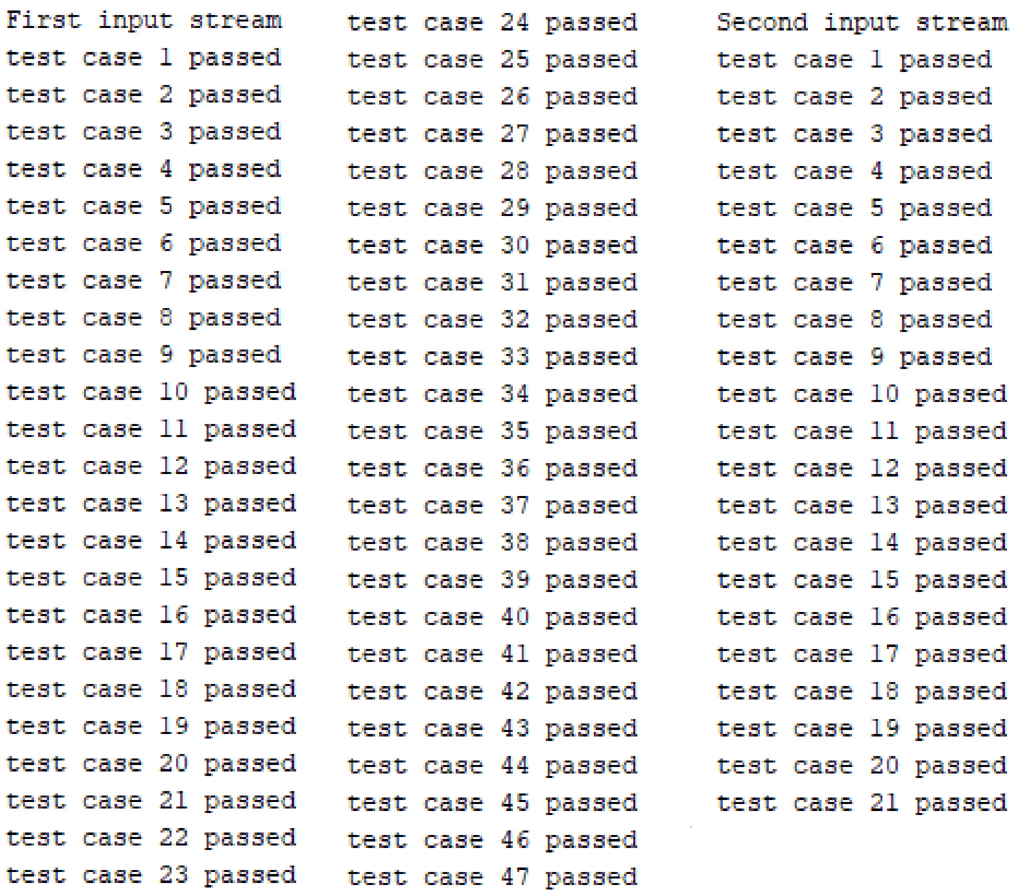


Figure 62: first 20 bits in RTL

Final Results



```
First input stream      test case 24 passed     Second input stream
test case 1 passed      test case 25 passed     test case 1 passed
test case 2 passed      test case 26 passed     test case 2 passed
test case 3 passed      test case 27 passed     test case 3 passed
test case 4 passed      test case 28 passed     test case 4 passed
test case 5 passed      test case 29 passed     test case 5 passed
test case 6 passed      test case 30 passed     test case 6 passed
test case 7 passed      test case 31 passed     test case 7 passed
test case 8 passed      test case 32 passed     test case 8 passed
test case 9 passed      test case 33 passed     test case 9 passed
test case 10 passed     test case 34 passed     test case 10 passed
test case 11 passed     test case 35 passed     test case 11 passed
test case 12 passed     test case 36 passed     test case 12 passed
test case 13 passed     test case 37 passed     test case 13 passed
test case 14 passed     test case 38 passed     test case 14 passed
test case 15 passed     test case 39 passed     test case 15 passed
test case 16 passed     test case 40 passed     test case 16 passed
test case 17 passed     test case 41 passed     test case 17 passed
test case 18 passed     test case 42 passed     test case 18 passed
test case 19 passed     test case 43 passed     test case 19 passed
test case 20 passed     test case 44 passed     test case 20 passed
test case 21 passed     test case 45 passed     test case 21 passed
test case 22 passed     test case 46 passed
test case 23 passed     test case 47 passed
```

Figure 64: RTL final simulation results

Waveforms

As shown in the figure below, when active low reset becomes "0", the whole returns to IDLE state and we start reading input when encoder valid becomes high.

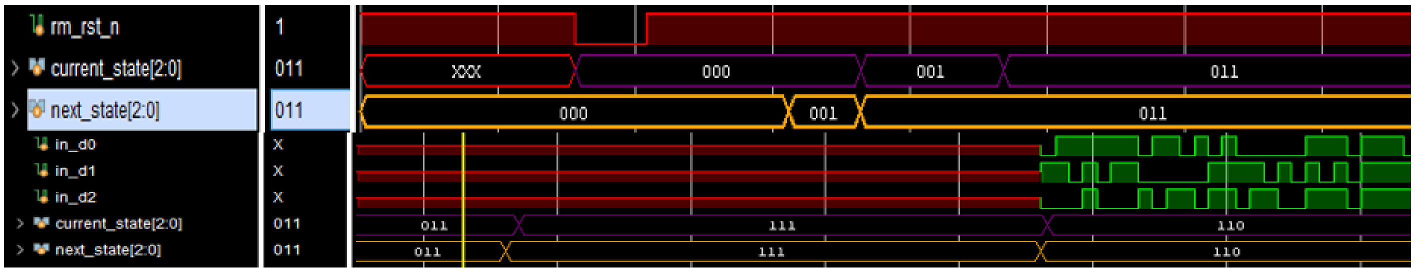


Figure 65: states transition from reset to input read

Reading 360 bits in serial from tail bit convolutional encoder through 3-bit streams in_d0, in_d1 and in_d2 when rm_enable_ld is high.

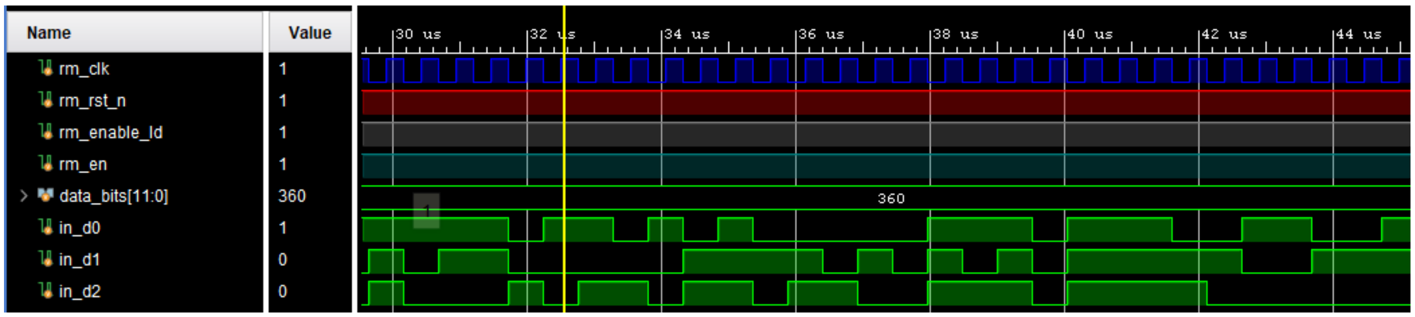


Figure 66: Input stream reading

When out_control changes for 1 clock cycle to "11", this means that we will select 24 bits and be ready for out_enable to become high for output stage.

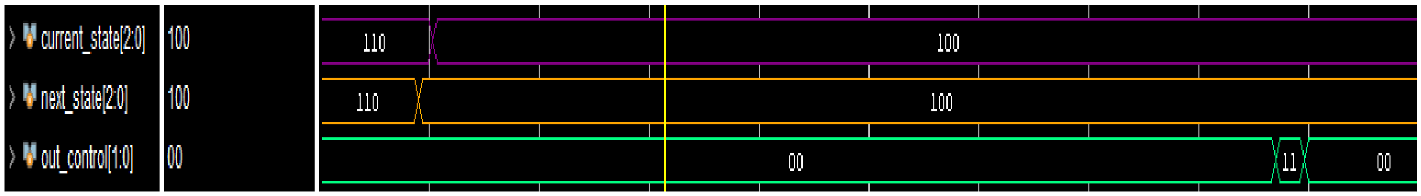


Figure 67: Bit selection and out_control new value

When out_enable becomes high, rm_valid becomes high and data is got out serially.

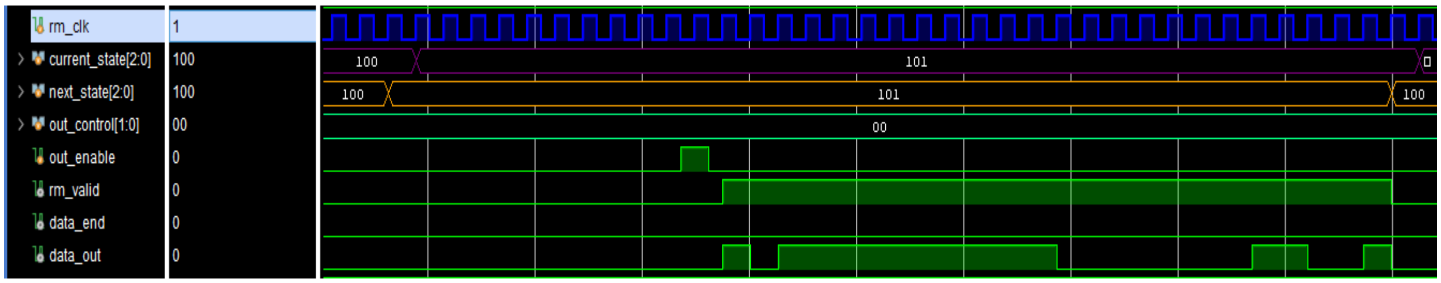


Figure 68: Output state

At the end of the last output stream data_end flag becomes high indicates that all data in the register files are got out and the rate matcher block will return to IDLE state for a new input stream.

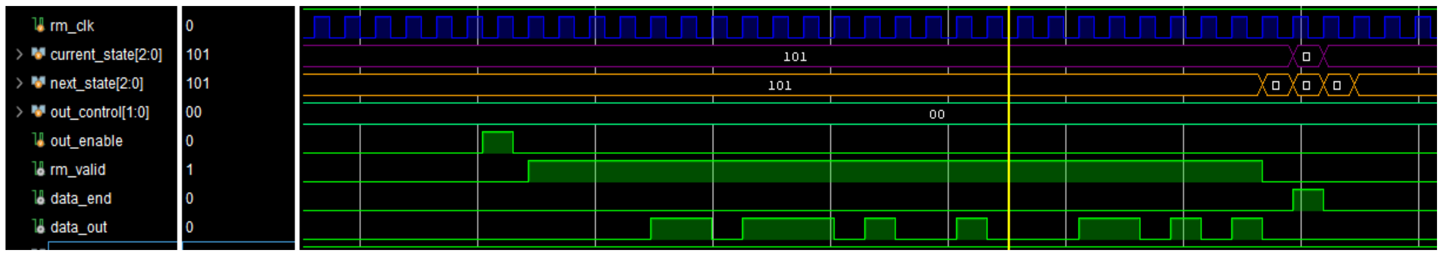
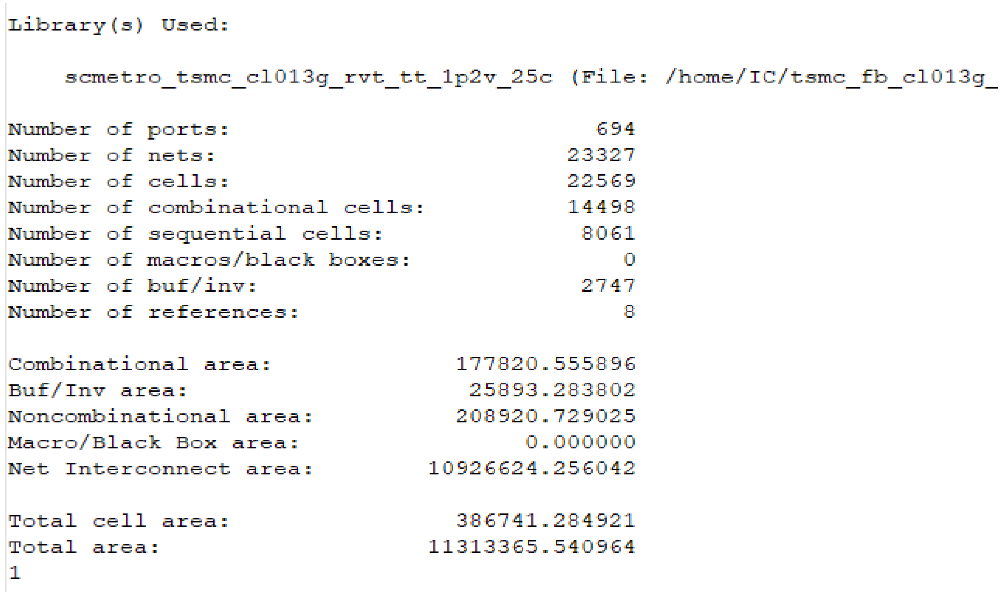


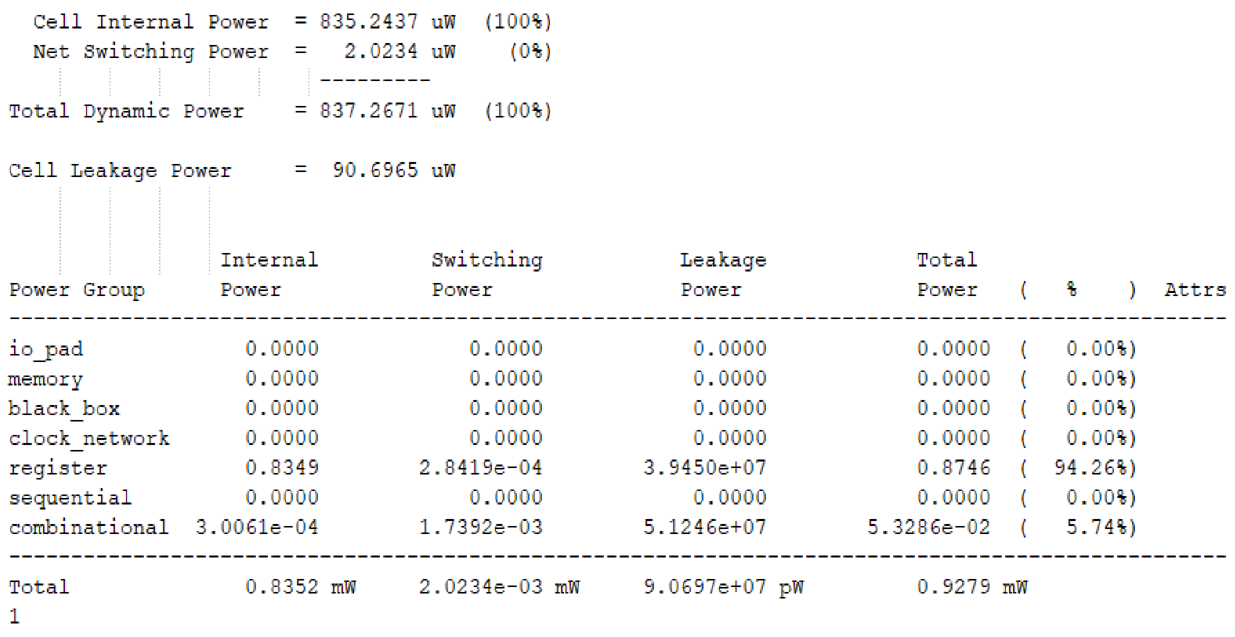
Figure 69: Last output data

4.3.7. Synthesis Results



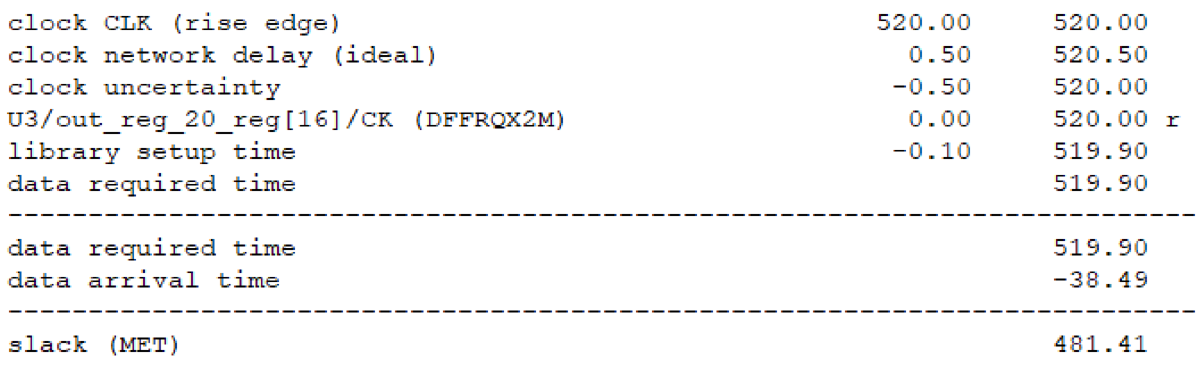
```
Library(s) Used:

    scmetro_tsmc_cl013g_rvt_tt_1p2v_25c (File: /home/IC/tsmc_fb_cl013g_

Number of ports:                          694
Number of nets:                         23327
Number of cells:                        22569
Number of combinational cells:          14498
Number of sequential cells:              8061
Number of macros/black boxes:               0
Number of buf/inv:                       2747
Number of references:                       8

Combinational area:              177820.555896
Buf/Inv area:                     25893.283802
Noncombinational area:           208920.729025
Macro/Black Box area:                 0.000000
Net Interconnect area:         10926624.256042

Total cell area:                 386741.284921
Total area:                    11313365.540964
1
```

Figure 70: Rate Matcher Area report



```
  Cell Internal Power  = 835.2437 uW  (100%)
  Net Switching Power  =   2.0234 uW    (0%)
                         ---------
Total Dynamic Power    = 837.2671 uW  (100%)

Cell Leakage Power     =  90.6965 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000            0.0000  (   0.00%)
register           0.8349       2.8419e-04        3.9450e+07            0.8746  (  94.26%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational  3.0061e-04       1.7392e-03        5.1246e+07        5.3286e-02  (   5.74%)
--------------------------------------------------------------------------------------------------
Total              0.8352 mW    2.0234e-03 mW     9.0697e+07 pW         0.9279 mW
1
```

Figure 71: Rate Matcher Power report



```
clock CLK (rise edge)                                  520.00     520.00
clock network delay (ideal)                              0.50     520.50
clock uncertainty                                       -0.50     520.00
U3/out_reg_20_reg[16]/CK (DFFRQX2M)                      0.00     520.00 r
library setup time                                      -0.10     519.90
data required time                                                519.90
---------------------------------------------------------------------------
data required time                                                519.90
data arrival time                                                 -38.49
---------------------------------------------------------------------------
slack (MET)                                                       481.41
```

Figure 72: Rate Matcher Timing_max (setup time) report

4.4. Scrambler

4.4.1. Block Diagram

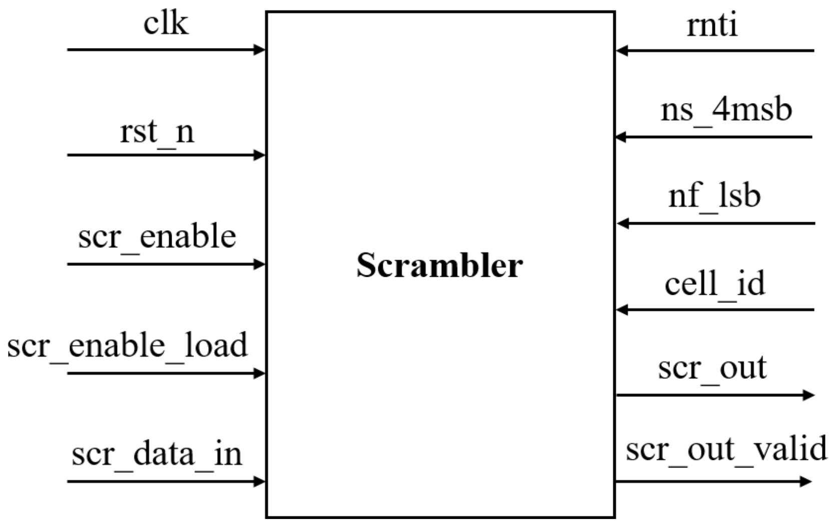


Figure 73: Scrambler Block Diagram

4.4.2. Interface Table

Port	Direction	Width	Description
clk	Input	1 bit	System clock.
rst_n	Input	1 bit	Block reset.
scr_enable	Input	1 bit	Block enable
scr_enable_load	Input	1 bit	Enable block by valid out of rate matcher to start reading the input bit stream.
scr_data_in	Input	1 bit	Input bit stream coming from rate matcher.
rnti	Input	16 bit	Radio Network Temporary ID given from upper layer.
ns_4msb	Input	4 bit	First slot number 4msb given from upper layer.
nf_lsb	Input	1 bit	First system frame number LSB given from upper layer.
cell_id	Input	9 bit	Physical layer cell identity given from upper layer.
scr_out	Output	1 bit	Output bit scrambled by the gold sequence.
scr_out_valid	Output	1 bit	Signal high with the output bit for the next block to know that is a valid output to read.

Table 7: Interface table of scrambler

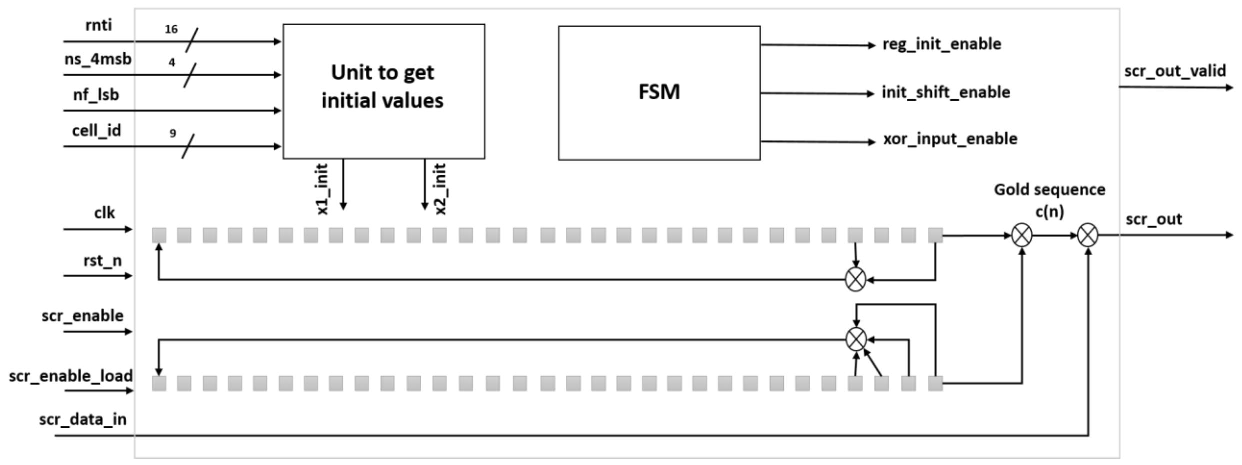


Figure 74: Scrambler architecture

4.4.3. Operation

There is a finite state machine controlling the states which the scrambler will be work in it so:

- After reset, the scrambler goes to an initialization state to assert the initial values of the LFSRs using the upper layer parameters.
- After LFSRs initialization is done, shift state is enabled to start the 1600 of shifting to have the gold sequence ready and that take 1600 clock cycles.
- After the 1600 cycle of shifting, the block is waiting in its state for a valid input coming from rate matcher to start reading the input.
- Then when the valid become high the block start reading the input and XORing it with the gold sequence generated by the two LFSRs to generate the output.

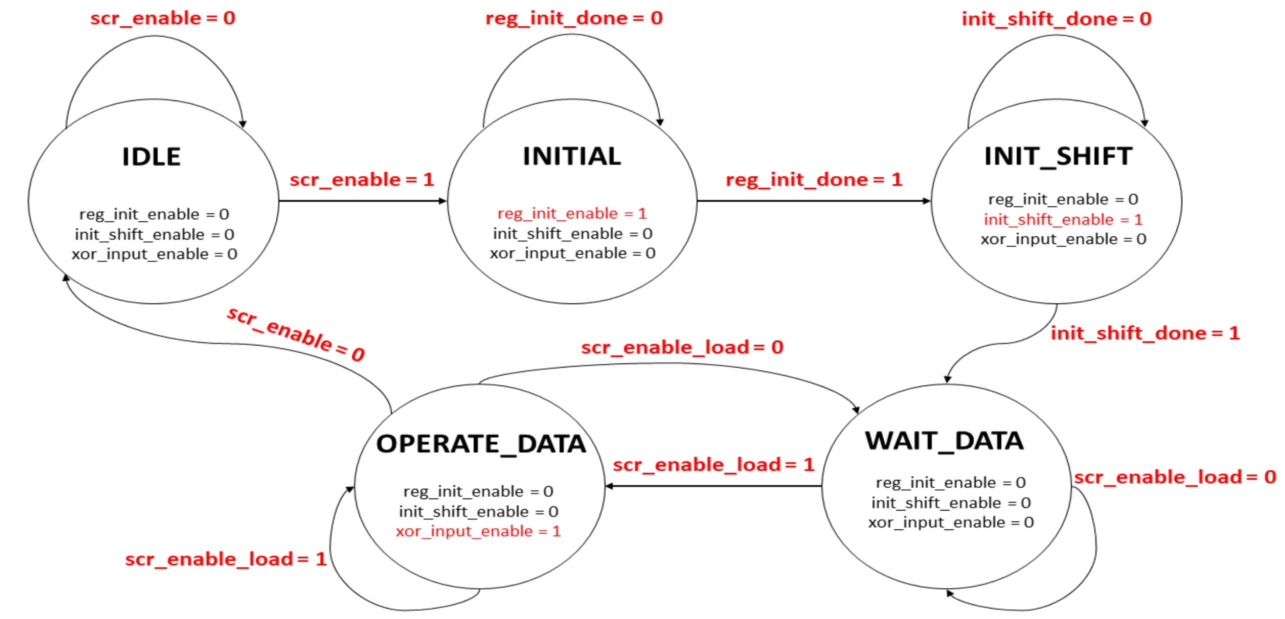


Figure 75: Scrambler FSM

4.4.4. Challenges

- First challenge was calculating the initial value of x2 as there is more than one multiplication but it was noticed that all multiplication are by number of base 2 so it can just performed using shift and concatenation only.
- Second challenge is the large number of input ports because of upper layer parameters used in x2 initial value calculation so as an optimization not all bits were taken as input for some parameters as for first system frame number n_f it is consists of 10 bits but only LSB bit is used in the design as it's the result of $n_f mod2$, and for the first slot number it consists of 5 bits but only 4 MSBs is used in the design as they are the result of $\lfloor {}^{n_s}/_2 \rfloor$.

4.4.5. Future work:

We can optimize in the number of cycles for the initial shift state and instead of using 1600 cycle we can use only 1 clock cycle and that is done by calculating the indices of the bits in the LFSR we need to XOR together to get the value of each bit in the LFSR after 1600 cycle using the following equations:

$$x_1(n+31) = \big(x_1(n+3) + x_1(n)\big)mod2$$

$$x_2(n+31) = \big(x_2(n+3) + x_2(n+2) + x_2(n+1) + x_2(n)\big)mod2$$

So we substitute $n = 1600$ and then get the indices for XORing for each bit in LFSR so the indices will be from 0 to 31.

This method will be apply only on x_2 as the initial value before shift changes according to the upper layer parameters but x_1 have a constant initial value so we can calculate its value after 1600 cycle and write it every time we need to initialize it.

4.4.6. Simulation results

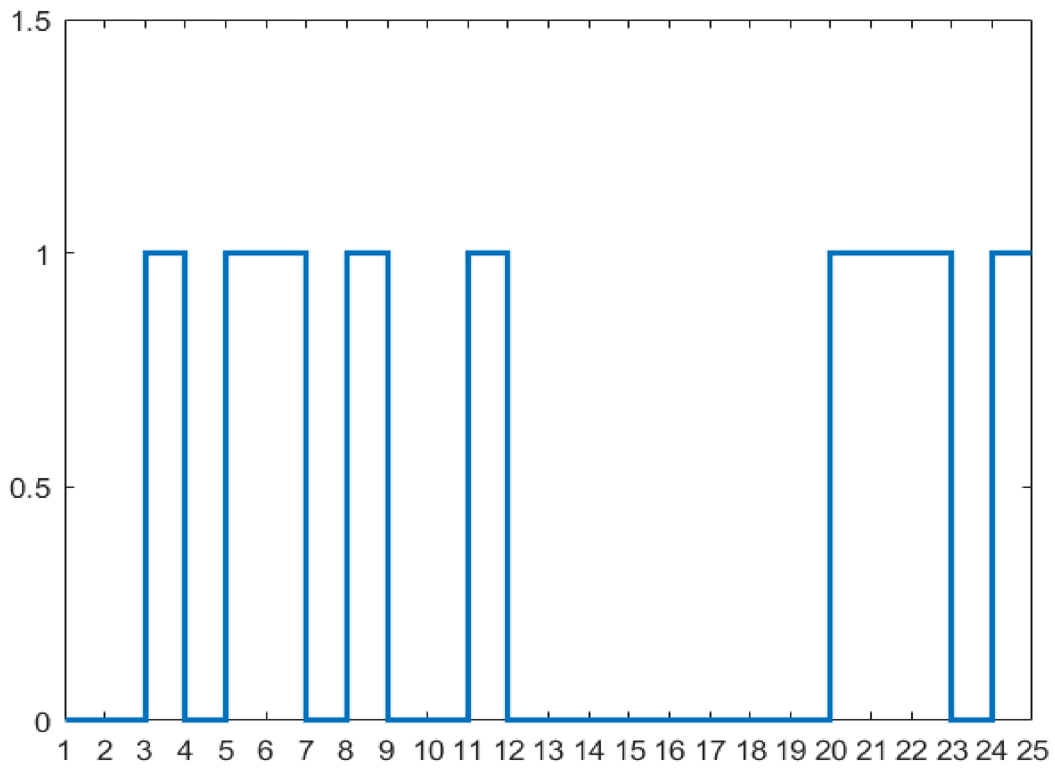
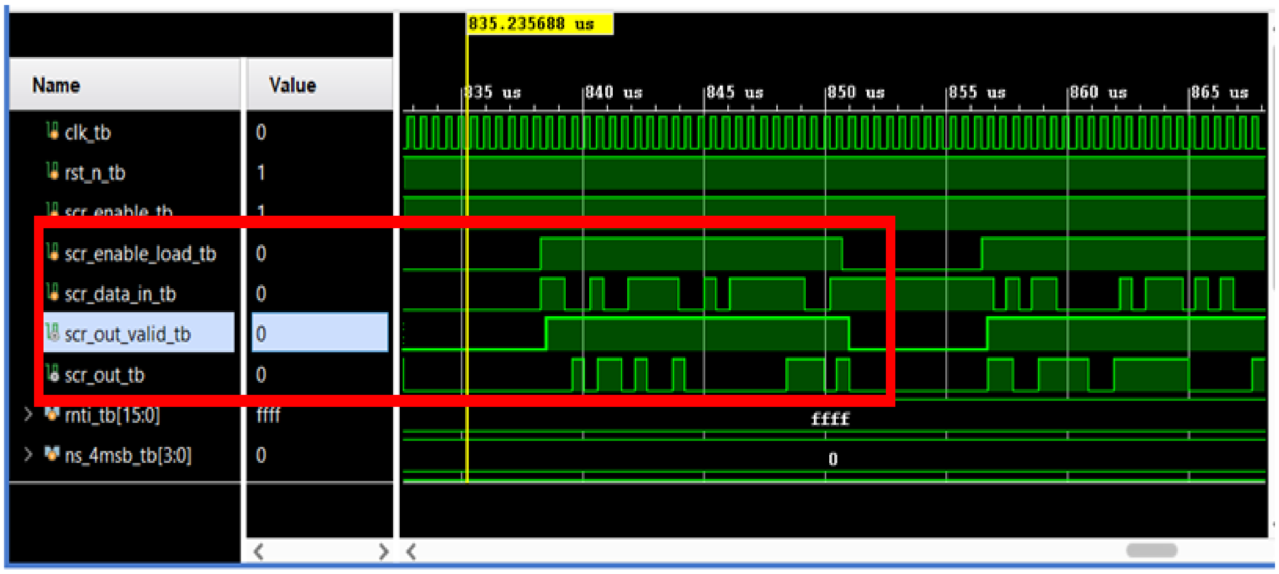


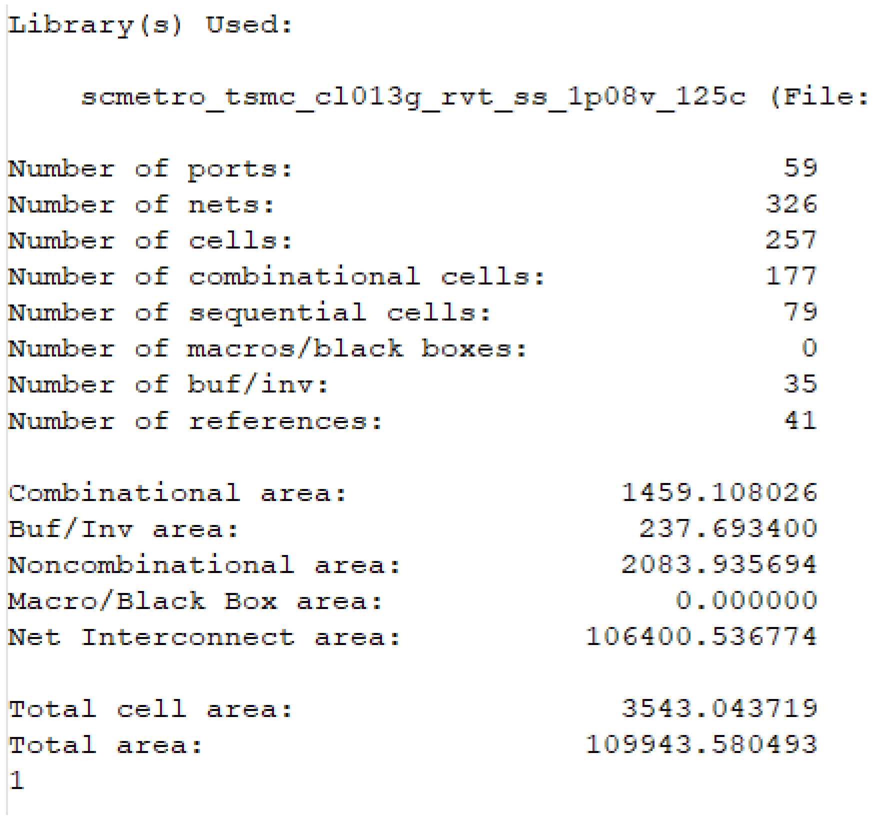
Figure 76: MATLAB output for scrambler NcellID = 65535, nf = 1, ns = 0



Name	Value
clk_tb	0
rst_n_tb	1
scr_enable_tb	1
scr_enable_load_tb	0
scr_data_in_tb	0
scr_out_valid_tb	0
scr_out_tb	0
rnti_tb[15:0]	ffff
ns_4msb_tb[3:0]	0

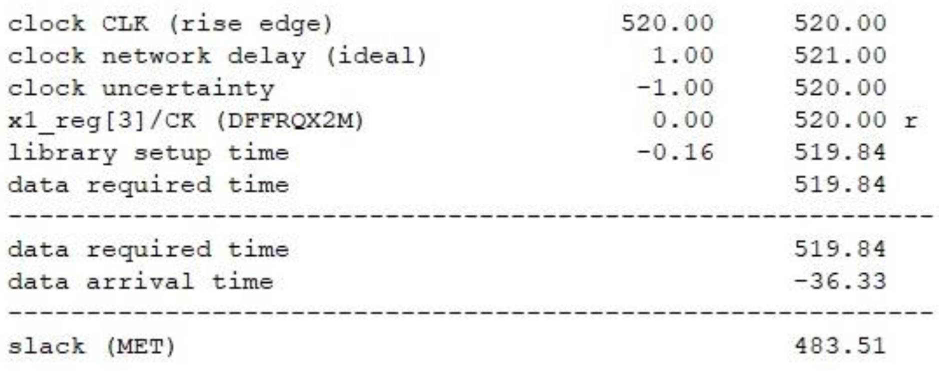
Figure 77: RTL output for scrambler NcellID = 65535, nf = 1, ns = 0

4.4.7. Synthesis Results



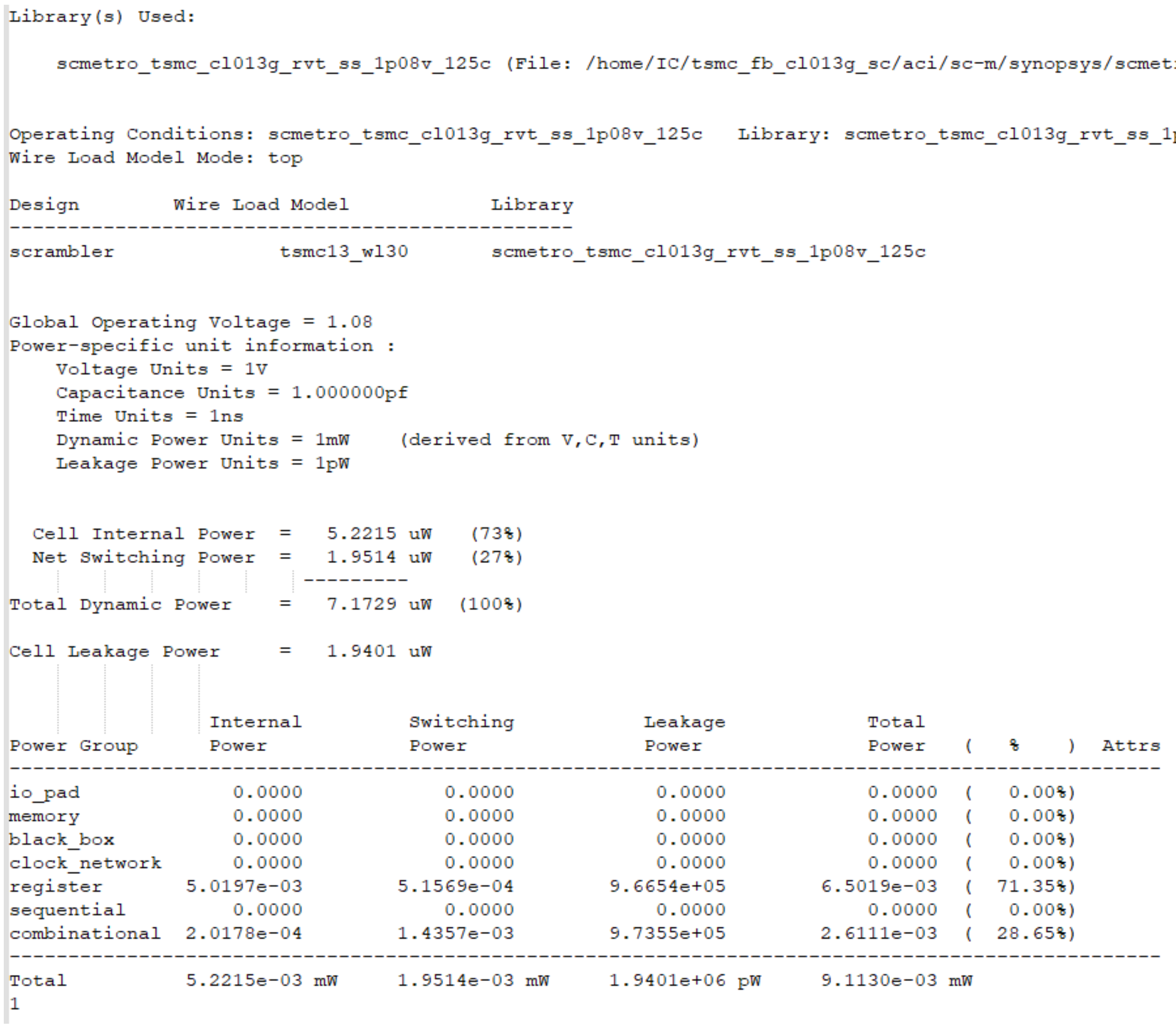
```
Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File:

Number of ports:                           59
Number of nets:                           326
Number of cells:                          257
Number of combinational cells:            177
Number of sequential cells:                79
Number of macros/black boxes:               0
Number of buf/inv:                         35
Number of references:                      41

Combinational area:               1459.108026
Buf/Inv area:                      237.693400
Noncombinational area:            2083.935694
Macro/Black Box area:                0.000000
Net Interconnect area:          106400.536774

Total cell area:                  3543.043719
Total area:                     109943.580493
1
```

Figure 78: Scrambler area report



```
clock CLK (rise edge)                  520.00     520.00
clock network delay (ideal)              1.00     521.00
clock uncertainty                       -1.00     520.00
x1_reg[3]/CK (DFFRQX2M)                  0.00     520.00 r
library setup time                      -0.16     519.84
data required time                                519.84
-----------------------------------------------------------
data required time                                519.84
data arrival time                                 -36.33
-----------------------------------------------------------
slack (MET)                                       483.51
```

Figure 79: Scrambler Timing_max (setup time) report



```
Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File: /home/IC/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetr


Operating Conditions: scmetro_tsmc_cl013g_rvt_ss_1p08v_125c   Library: scmetro_tsmc_cl013g_rvt_ss_1p
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
scrambler              tsmc13_wl30       scmetro_tsmc_cl013g_rvt_ss_1p08v_125c


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   5.2215 uW   (73%)
  Net Switching Power  =   1.9514 uW   (27%)
                         ---------
Total Dynamic Power    =   7.1729 uW  (100%)

Cell Leakage Power     =   1.9401 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000             0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000             0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000             0.0000  (   0.00%)
clock_network      0.0000           0.0000            0.0000             0.0000  (   0.00%)
register         5.0197e-03       5.1569e-04        9.6654e+05         6.5019e-03  (  71.35%)
sequential         0.0000           0.0000            0.0000             0.0000  (   0.00%)
combinational    2.0178e-04       1.4357e-03        9.7355e+05         2.6111e-03  (  28.65%)
--------------------------------------------------------------------------------------------------
Total            5.2215e-03 mW    1.9514e-03 mW     1.9401e+06 pW      9.1130e-03 mW
1
```

Figure 80: Scrambler power report

4.5. Modulation Mapper

4.5.1. Block Diagram

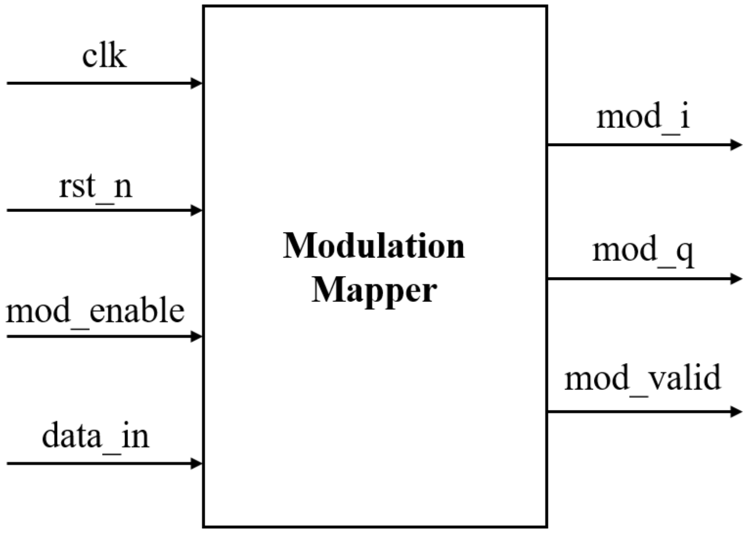


Figure 81: Modulation Mapper Block Diagram

4.5.2. Interface Table

Port	Direction	Width	Description
clk	Input	1 bit	System clock.
rst_n	Input	1 bit	Block reset.
mod_enable	Input	1 bit	Enable the block by the valid out of scrambler to start reading the input bit stream.
data_in	Input	1 bit	Input bit stream coming from scrambler.
mod_i	Output	16 bit	Real component of the output symbol.
mod_q	Output	16 bit	Imaginary component of the output symbol.
mod_valid	Output	1 bit	Signal high with the output symbol for the next block to know that is a valid output to read.

Table 8: Modulator interface table

4.5.3. Operation

As we use QPSK modulation scheme so the block maps each two bits to their corresponding symbol so it takes 2 clock cycles to generate one symbol to the next block, the symbol consists of 16-bits the most 5 bits for integer and the fraction part is 11 bits so for

- $+1/\sqrt{2} = 0000010110101000$
- $-1/\sqrt{2} = 1111101001011000$

We design the modulation mapper with two approaches:

➢ The first approach was introduced using the information of the incoming “tbs” from upper layer :
  - The first incoming bit is stored in a 1-bit register “temp”, when the second input bit comes, the multiplexing takes place and generate the corresponding symbol.
  - Similarly, the third bit is stored and compared along with the fourth bit, and so on.
  - Generally, in the odd clock cycles, the incoming bit is stored. In the even clock cycles, the stored bit is compared along with the incoming bit in this even cycle to generate the output symbol.
  - That was achieved by a counter which counts till the upcoming “tbs” and by using the condition that if the LSB of the counter is one then it is an odd cycle, so it’s expected to store the bit for one more cycle, otherwise it is an even cycle, so it’s expected to have an output symbol.
  - Since each 2 bits are mapped to the corresponding symbol, therefore it takes 2 clock cycles to generate one symbol to the next block.

➢ The second approach was introduced without using the information of the incoming “tbs” from upper layer:

  But using a bit “is_odd” which toggles every clock cycle:
    - **If it is low:** then the incoming bit is stored.
    - **If it is high:** then the incoming bit is compared along with the previous one to get multiplexed to generate the output symbol.

At the end we use the second approach as we didn’t use the 12-bit counter of the tbs and use 1-bit toggle every cycle only.

As a conclusion the first bit is used to generate the real part of the symbol (I) and the second bit is used to generate the imaginary part of the symbol (Q) as shown in the figure below.

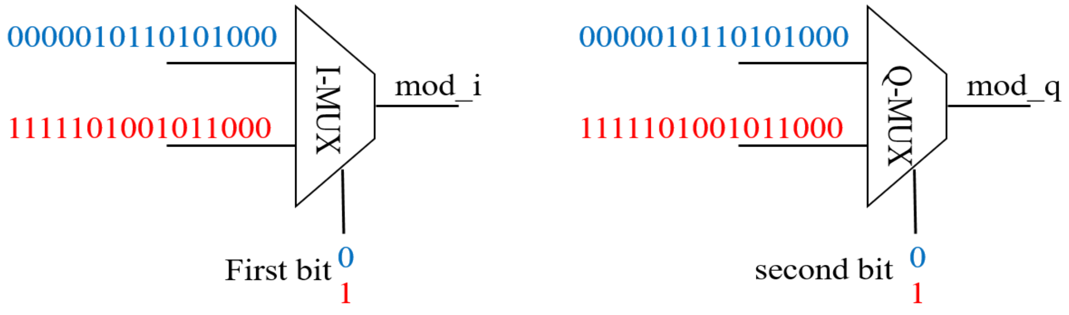


Figure 82: Modulation Mapper symbol generation

4.5.4. Simulation results

We tested the four cases of the QPSK modulator and all pass in MATLAB and RTL simulation so there is an example for one of the cases.

For input stream: “1 0”

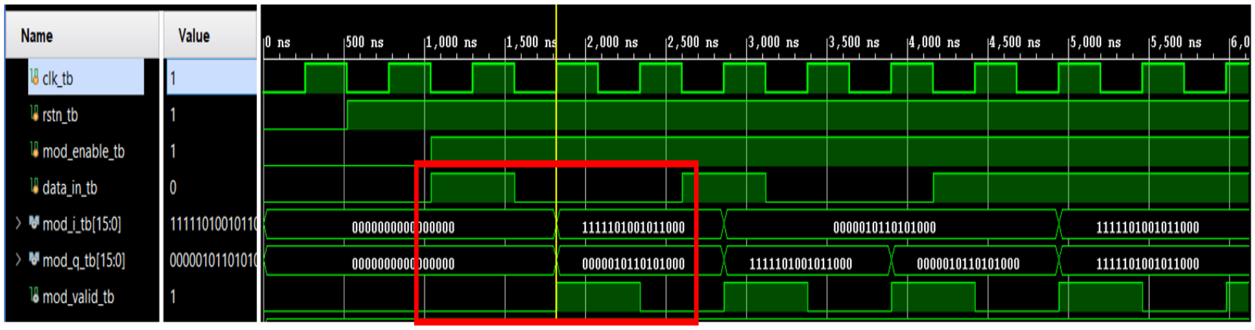
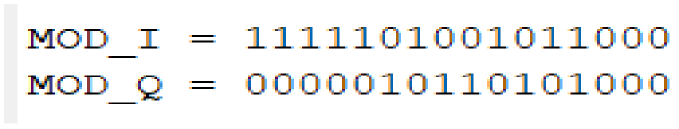


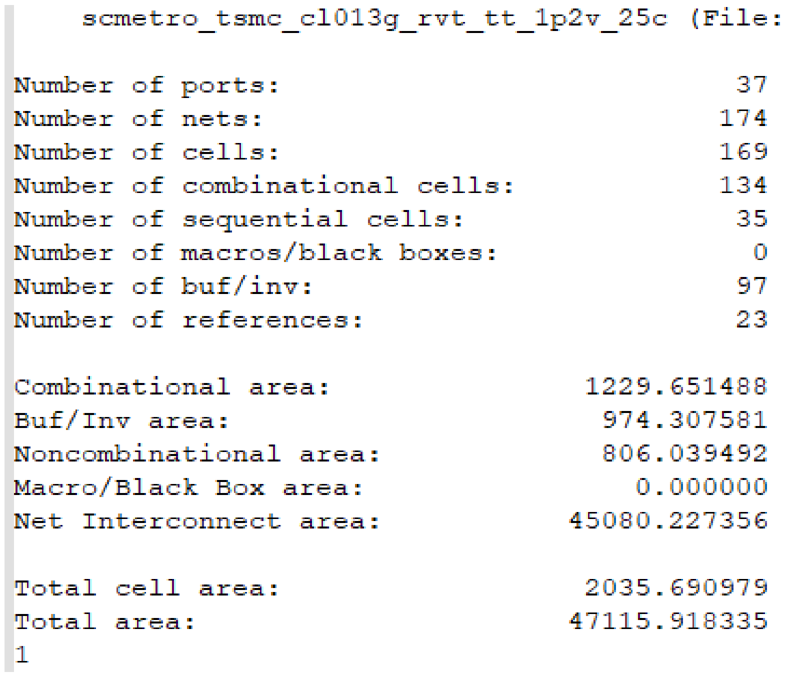
Figure 83: Modulation Mapper RTL output



```
MOD_I = 1111101001011000
MOD_Q = 0000010110101000
```

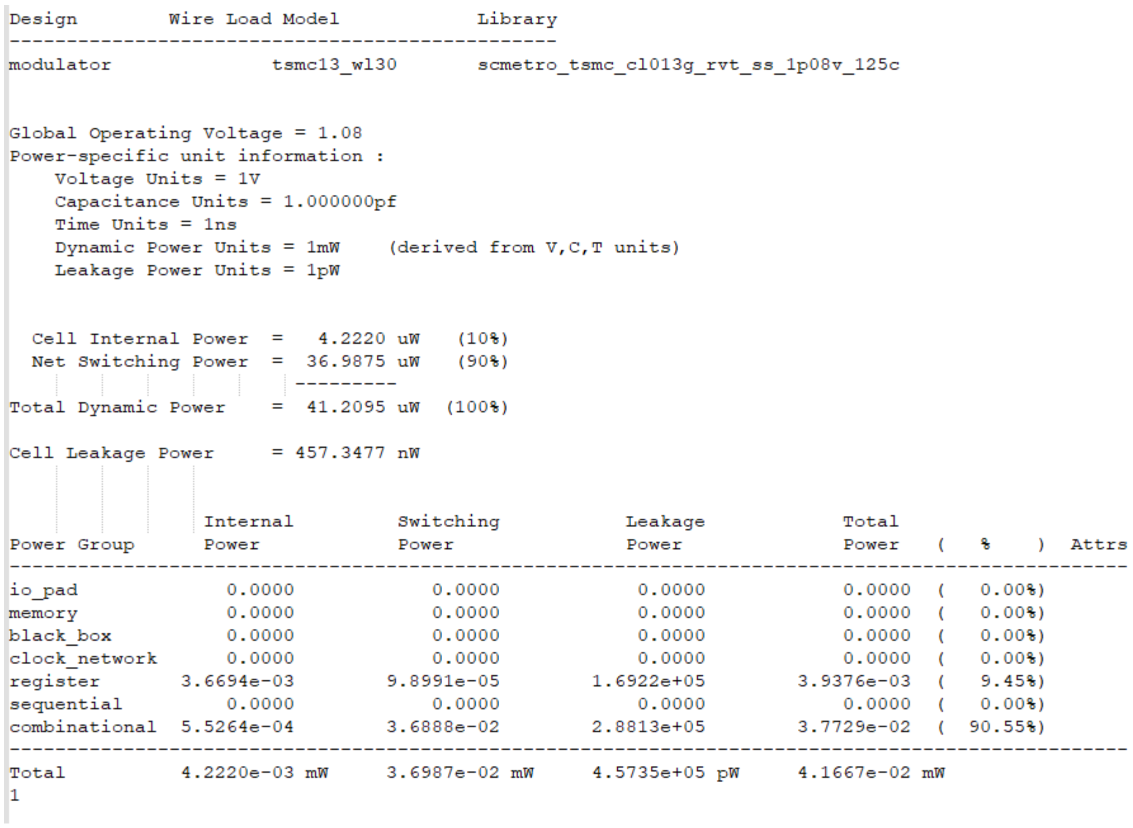
Figure 84: Modulation Mapper MATLAB output

4.5.5. Synthesis results



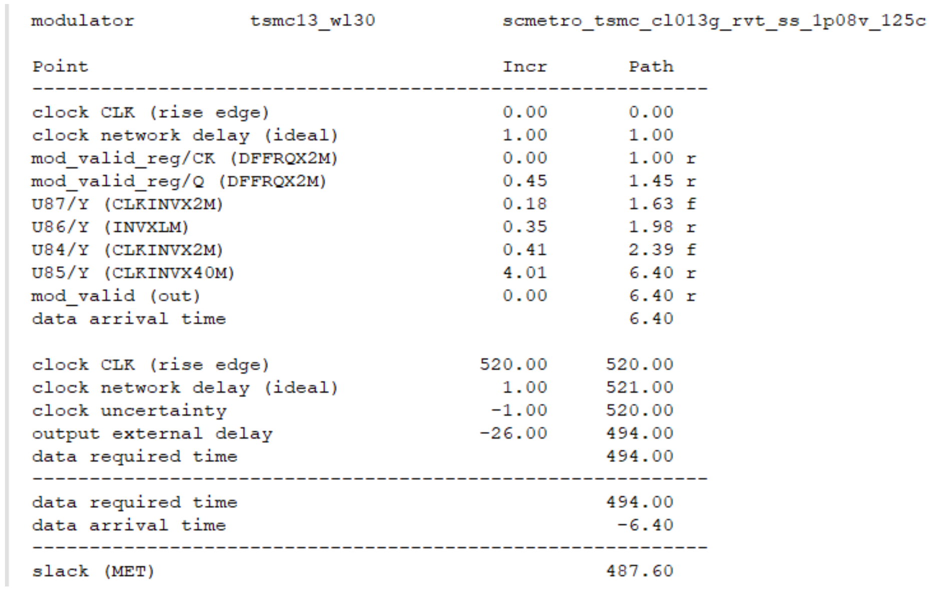
```
    scmetro_tsmc_cl013g_rvt_tt_1p2v_25c (File:

Number of ports:                           37
Number of nets:                           174
Number of cells:                          169
Number of combinational cells:            134
Number of sequential cells:                35
Number of macros/black boxes:               0
Number of buf/inv:                         97
Number of references:                      23

Combinational area:               1229.651488
Buf/Inv area:                      974.307581
Noncombinational area:             806.039492
Macro/Black Box area:                0.000000
Net Interconnect area:           45080.227356

Total cell area:                  2035.690979
Total area:                      47115.918335
1
```

Figure 85: Modulation Mapper area report



```
Design         Wire Load Model            Library
------------------------------------------------
modulator              tsmc13_wl30        scmetro_tsmc_cl013g_rvt_ss_1p08v_125c


Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   4.2220 uW   (10%)
  Net Switching Power  =  36.9875 uW   (90%)
                         ---------
Total Dynamic Power    =  41.2095 uW  (100%)

Cell Leakage Power     = 457.3477 nW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad              0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory              0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box           0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network       0.0000           0.0000              0.0000             0.0000  (   0.00%)
register         3.6694e-03       9.8991e-05          1.6922e+05         3.9376e-03  (   9.45%)
sequential          0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational    5.5264e-04       3.6888e-02          2.8813e+05         3.7729e-02  (  90.55%)
--------------------------------------------------------------------------------------------------
Total            4.2220e-03 mW    3.6987e-02 mW       4.5735e+05 pW      4.1667e-02 mW
1
```

Figure 86: Modulation Mapper power report



```
modulator          tsmc13_wl30           scmetro_tsmc_cl013g_rvt_ss_1p08v_125c

Point                                        Incr       Path
------------------------------------------------------------
clock CLK (rise edge)                        0.00       0.00
clock network delay (ideal)                  1.00       1.00
mod_valid_reg/CK (DFFRQX2M)                  0.00       1.00 r
mod_valid_reg/Q (DFFRQX2M)                   0.45       1.45 r
U87/Y (CLKINVX2M)                            0.18       1.63 f
U86/Y (INVXLM)                               0.35       1.98 r
U84/Y (CLKINVX2M)                            0.41       2.39 f
U85/Y (CLKINVX40M)                           4.01       6.40 r
mod_valid (out)                              0.00       6.40 r
data arrival time                                       6.40

clock CLK (rise edge)                      520.00     520.00
clock network delay (ideal)                  1.00     521.00
clock uncertainty                           -1.00     520.00
output external delay                      -26.00     494.00
data required time                                    494.00
------------------------------------------------------------
data required time                                    494.00
data arrival time                                      -6.40
------------------------------------------------------------
slack (MET)                                           487.60
```

Figure 87: Modulation mapper Timing_max (setup time) report

4.6. Resource Element Mapper

4.6.1. NRS Generation

4.6.1.1. Block Diagram

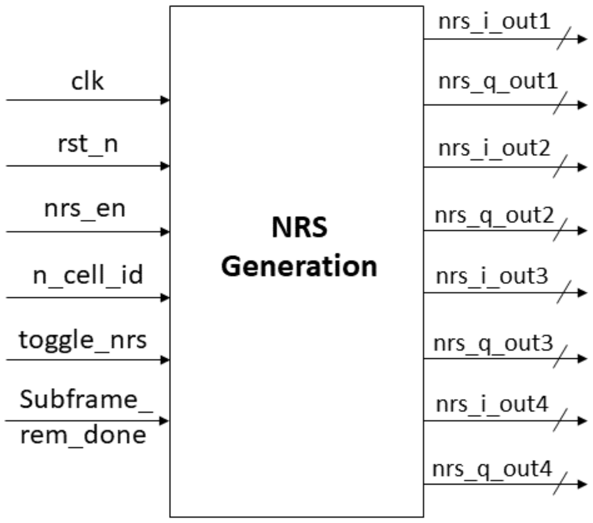


Figure 88: Block Interface of NRS generation

4.6.1.2. Interface Table

Port	Type	width	Description
clk	Input	1 bit	System clock
rst_n	Input	1 bit	Block reset
nrs_enable	Input	1 bit	Block enable
subframe_rem _done	Input	1 bit	Comes from element mapper indicating that a sub-frame is done
toggle_nrs	Input	1 bit	Comes from element mapper to decide which part of the pilots memory will be read
n_cell_id	Input	9 bits	Physical layer cell identity given from upper layer
nrs_i_out1	Output	16 bits	In-phase output symbol of the narrow band reference signal
nrs_q_out1	Output	16 bits	Quadrature-phase output symbol of the narrow band reference signal
nrs_i_out2	Output	16 bits	In-phase output symbol of the narrow band reference signal
nrs_q_out2	Output	16 bits	Quadrature-phase output symbol of the narrow band reference signal
nrs_i_out3	Output	16 bits	In-phase output symbol of the narrow band reference signal
nrs_q_out3	Output	16 bits	Quadrature-phase output symbol of the narrow band reference signal
nrs_i_out4	Output	16 bits	In-phase output symbol of the narrow band reference signal
nrs_q_out4	Output	16 bits	Quadrature-phase output symbol of the narrow band reference signal

Table 9: Interface Table of NRS generation

4.6.1.3. Operation

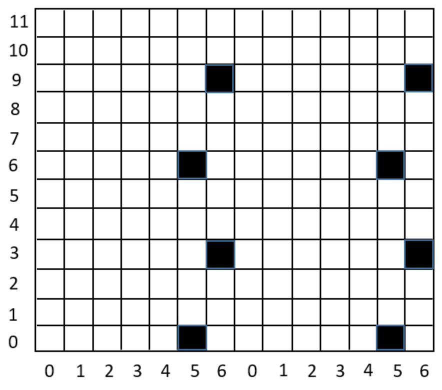


Figure 89: Two slots of the frame

- Each slot includes 4 pilot symbols as shown in fig.89, we want to have the reference signals of a complete sub-frame at a time.

- This was achieved by repeating the (31-bit X2 LFSR) 4 times as shown in fig.91 :
  - X2_1 : gives NRS of first slot & L = 5
  - X2_2 : gives NRS of first slot & L = 6
  - Similarly for X2_3 & X2_4 for the second slot

- All these 4 LFSRs are initialized in parallel whenever the nrs_enable becomes high according to the standard initialization equation.

- Then shift takes place in the X1 LFSR and in the X2 LFSRs according to equation (4) and is done for 1600 cycles. This can be noted in **INITIAL SHIFT** state in the FSM. Then the LSB of X2 is xored with the LSB of X1 to get the gold sequence as shown in fig.90

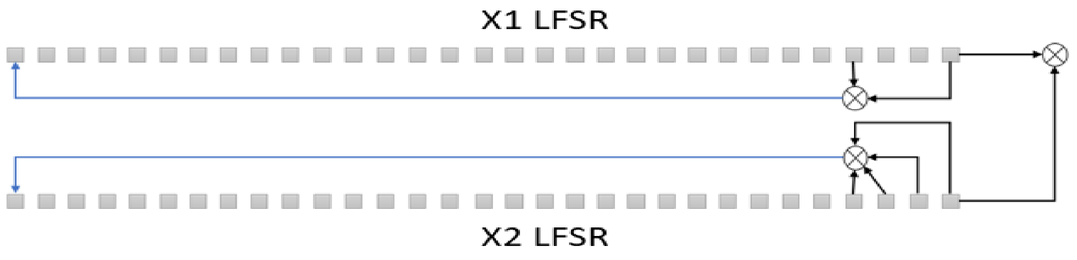


Figure 90: The two LFSRs of the NRS values

- Here comes the role of the 4-bit registers (c_n_1 , c_n_2 , c_n_3 , c_n_4) which store the four output bits from the generated sequence due to xoring X1 with (X2_1 , X2_2 , X2_3 , X2_4) respectively as shown in fig.91.

- After storing in the 4-bit registers is done, we go into a state which waits the subframe_rem_done signal from the element mapper module, then we go into muxing state which uses each 2 bits of the 4-bit registers as selection lines to generate the 16-bit symbol. Otherwise, we wait in this state.

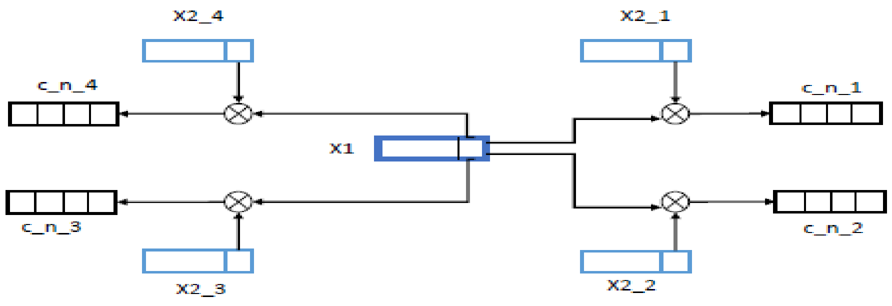


Figure 91: The 4-bit registers & LFSRs configuration

- The generated symbols are stored in a register file “pilots” to store the pilots of the whole subframe and using the “toggle_nrs” signal we can select the pilots of any of the two slots.
- A signal “toggle_nrs” comes from element mapper to decide which part of the pilots memory will be read depending on the slot (first or last 4 locations of pilots memory).
- All the previous steps are to be repeated once a signal “subframe_rem_done” becomes high which comes from the element mapper.

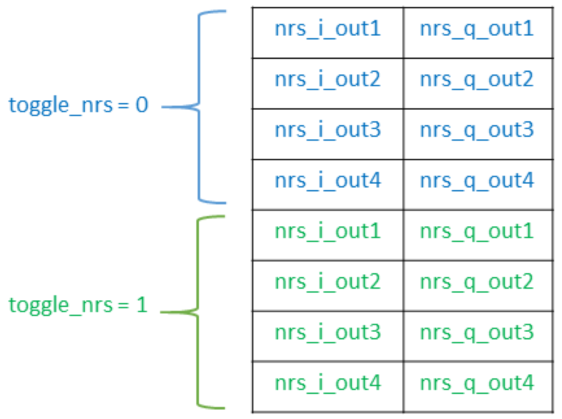


Figure 92: Pilots memory elements

4.6.1.4. FSM

A finite state machine is needed to organize the steps of generating the pilots, an illustrative summarized diagram of the controller status & control signals are shown in the below figure

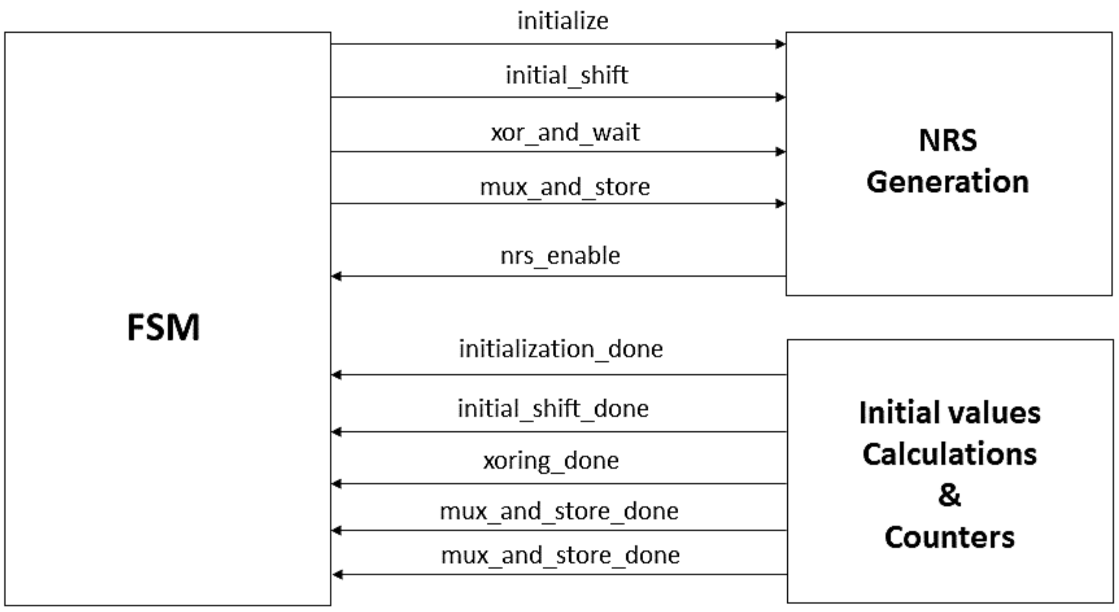


Figure 93: illustrative diagram of datapath & controller of NRS values

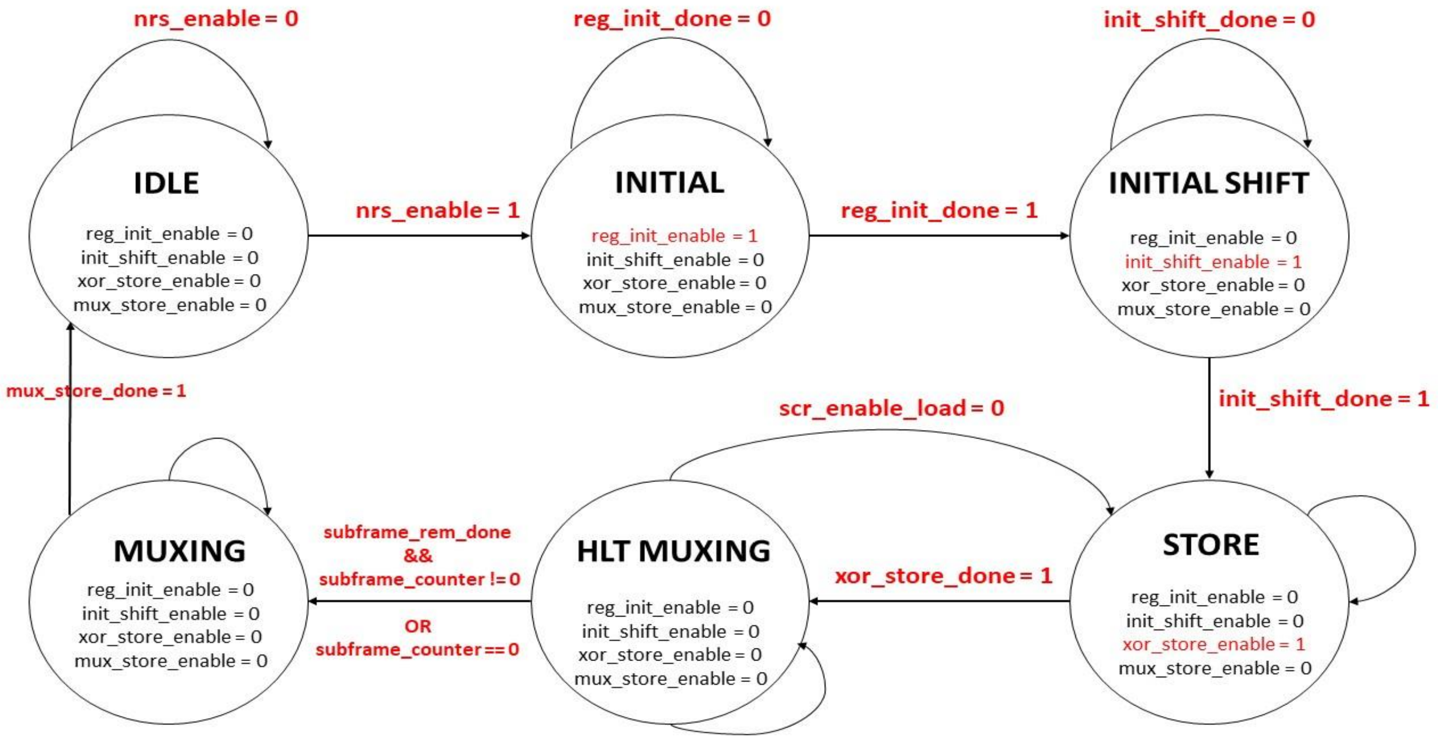


Figure 94: NRS_Generation FSM State Diagram

Current State	Condition (Status)	Next State	Output (Control)
IDLE	NRS Generation block is enabled	INITIAL	Initializing the LFSRs is enabled
INITIAL	Initializing the LFSRs is done	INITIAL_SHIFT	Initial 1600 shift cycles counter is enabled
INITIAL_SHIFT	Initial 1600 shift cycles counter is done	STORE	XORING & storing in 4-bit registers are enabled
STORE	XORING & storing in 4-bit registers are done	HLT_MUXING	XORING & storing in 4-bit registers are disabled
HLT_MUXING	A subframe is done AND we are not in the first subframe OR We are in the first subframe	MUXING	-
MUXING	Muxing each 2-bits & Storing in the pilots memory are done	IDLE	-
	Muxing each 2-bits & Storing in the pilots memory are not done	MUXING	Muxing & Storing in the pilots memory is enabled

Table 10: State Table of NRS Generation FSM

4.6.2. NRS Location

4.6.2.1. Block Diagram

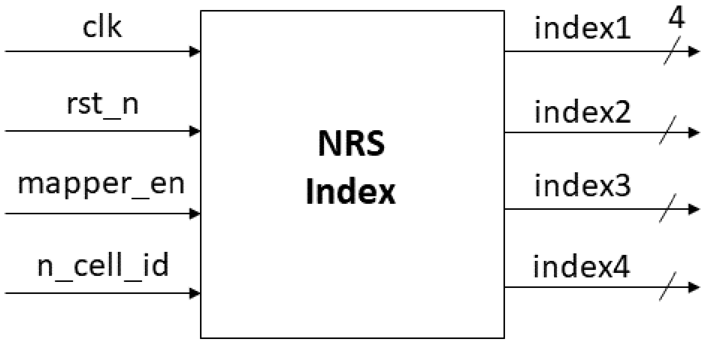


Figure 95: Interface Diagram of NRS Index

4.6.2.2. Interface Table

Port	Direction	Width	Description
clk	Input	1 bit	System clock
rst_n	Input	1 bit	Block reset
mapper_en	Input	1 bit	Block enable
n_cell_id	Input	9 bits	Physical layer cell identity given from upper layer
index1	Output	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
index2	Output	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
index3	Output	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
index4	Output	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier

Table 11: Interface Table of NRS Index

4.6.2.3. Operation

- Since the algorism of the code that generate the index contains complex operations like multiplication and mod of constant value, so we replace them with the equivalent values.
  - First: generate the mod operation for all the expected values of the Ncell and save it in a memory (we will have 6 output values as we make mod6)
  - Second: generate the values of the index0, index1, index2 and index3 from their equations in MATLAB so each output of the mode operation has its own 4 index values.

  - So when the Ncell enter the design we will find the mode output from a memory inside, then this mode value have the 4 values of the index which is the output of the design.

4.6.3. NPSS

4.6.3.1. Block Diagram

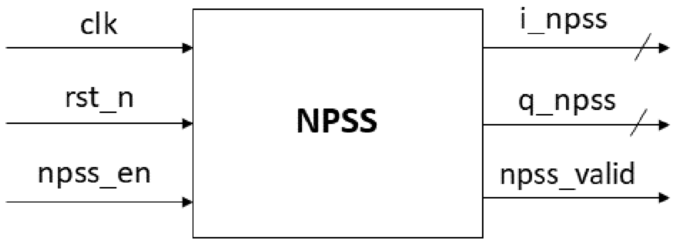


Figure 96: Interface Diagram of NPSS

4.6.3.2. Interface Table

Port	Direction	Width	Description
clk	Input	1 bit	System clock
rst_n	Input	1 bit	Block reset
npss_en	Input	1 bit	Block enable comes from element mapper module
i_npss	Output	16 bits	Narrowband Primary Synchronization signal input in-phase symbol from the NPSS module
q_npss	Output	16 bits	Narrowband Primary Synchronization signal input quadrature-phase symbol from the NPSS module
npss_valid	Output	1 bit	Valid signal with output NPSS symbol

Table 12: Interface Table of NPSS

4.6.3.3. Operation

- The idea is that the equation contains exponential operation, so it is calculated using MATLAB and then replaced with a ROM in RTL containing the values.

- When the enable signal is given to the block, it starts to read all the values in the ROM at every two clock cycles to enter the resource element mapper with the same rate like modulator

4.6.4. Element Mapper

4.6.4.1. Interface Table

Port	Type	Width	Description
clk	Input	1 bit	System clock
rst_n	Input	1 bit	Block reset
mapper_en	Input	1 bit	Block enable
mod_valid	Input	1 bit	Valid signal with output symbol sent by modulator
mod_i	Input	16 bits	In-phase input symbol from the modulator
mod_q	Input	16 bits	Quadrature-phase input symbol from the modulator
nrs_i_out1	Input	16 bits	In-phase input symbol from the NRS generation
nrs_q_out1	Input	16 bits	Quadrature-phase input symbol from the NRS generation
nrs_i_out2	Input	16 bits	In-phase input symbol of the narrow band reference signal
nrs_q_out2	Input	16 bits	Quadrature-phase input symbol from the NRS generation
nrs_i_out3	Input	16 bits	In-phase input symbol from the NRS generation
nrs_q_out3	Input	16 bits	Quadrature-phase input symbol from the NRS generation
nrs_i_out4	Input	16 bits	In-phase input symbol from the NRS generation
nrs_q_out4	Input	16 bits	Quadrature-phase input symbol from the NRS generation
index1	Input	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
index2	Input	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
index3	Input	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
index4	Input	4 bits	Determines the location of a pilot to be in which row, ranged from 0 to 11 where the row refers to the subcarrier
i_npss	Input	16 bits	Narrowband Primary Synchronization signal input in-phase symbol from the NPSS module
q_npss	Input	16 bits	Narrowband Primary Synchronization signal input quadrature-phase symbol from the NPSS module
ifft_valid	Input	1 bit	Valid signal sent from the IFFT block
ifft_ready	Input	1 bit	Sent from IFFT to indicates it is ready to receive the next OFDM symbol
rm_ctrl_en	Input	1 bit	Control signal sent from the system's FSM to make the block send rm_ctrl signal to rate matcher
rm_data out_en	Input	1 bit	Control signal sent from the system's FSM to resource element mapper to make the rate matcher start sending data

rem_stop	Input	1 bit	Control signal sent from the system's FSM when the whole operation is done to stop the IFFT block
rm_ctrl	Output	1 bit	Control signal sent to rate matcher to send 20 or 24 bits
rm_data_out	Output	1 bit	Control signal sent to rate matcher and FSM to make rate matcher start sending data
i_out	Output	16 bits	In-phase output QPSK symbol sent to IFFT
q_out	Output	16 bits	Quadrature-phase output QPSK symbol sent to IFFT
hlt_npss	Output	1 bit	Signal sent to FSM to stop sending bits until NPSS is done
npss_en	Output	1 bit	Signal sent to enable the NPSS module
toggle_nrs	Output	1 bit	Signal sent to NRS generation module to decide which part of the pilots memory will be read depending on the slot (first or last 4 locations)
Subframe rem_done	Output	1 bit	Signal sent to NRS generation indicating that a sub-frame is done to generate the new pilot symbols for the next subframe
Ifft_enable load	Output	1 bit	Signal sent to enable the IFFT one cycle before sending the symbols
ifft_stop	Output	1 bit	Sent to ifft when the rem_stop signal reaches from the FSM

Table 13: Interface Table of Element Mapper

4.6.4.2. Operation

The element mapper module is considered the brain of the resource allocation in NB-IoT LTE. It takes the responsibility of the actual mapping of data or pilots or synchronization, also it has status/control signals from/to the system's finite state machine, rate matcher, and IFFT.

It is mainly composed of memories and different counters to help in the mapping process and here is a brief description of some of these counters

- **Counting Sub-carriers (Frequency domain) :**
  - It counts the rows of the NB-IoT physical resource block (PRB).
  - Each two clock cycles, a QPSK symbol is stored inside a memory and the counter resets automatically.
  - In case of NRS, the counter counts are reduced by 2.

- **Counting OFDM Symbols (Time domain) :**
  - It counts the columns of the NB-IoT physical resource block (PRB).
  - It counts the 7 OFDM symbols whether these symbols represents data symbols from the modulator (first 5 OFDM symbols) or pilot symbols from NRS generation & Index modules (last 2 OFDM symbols) or synchronization symbols from NPSS.

  - When one slot is done, toggle a bit is sent to NRS generation module to select the pilot symbols for this slot.

- **Slot Counter :**

  - It is needed to know the slot number which will be used to map the NPSS symbols in their locations and also to send a signal indicates that the subframe is done for NRS generation module to exit from the state of “HLT_MUXING” and store the new pilots.

- **Output Counter :**

  - When the IFFT block sends “ifft_ready” signal, the “ifft_enable_load” will be sent to IFFT and is high for one clock cycle, so in order to send the 16 QPSK symbols we need a signal to be high for 16 clock cycles “load_ifft”, this counter counts these 16 output cycles.
  - For the first QPSK symbol, the IFFT doesn’t send “ifft_ready”, so when the first symbol is ready it will be sent directly and the signal “first_symbol_ready” is high when the first column is done.

- **NPSS Counter** and others ..

4.6.5. Resource Element Mapper TOP LEVEL

4.6.5.1. Block Interface

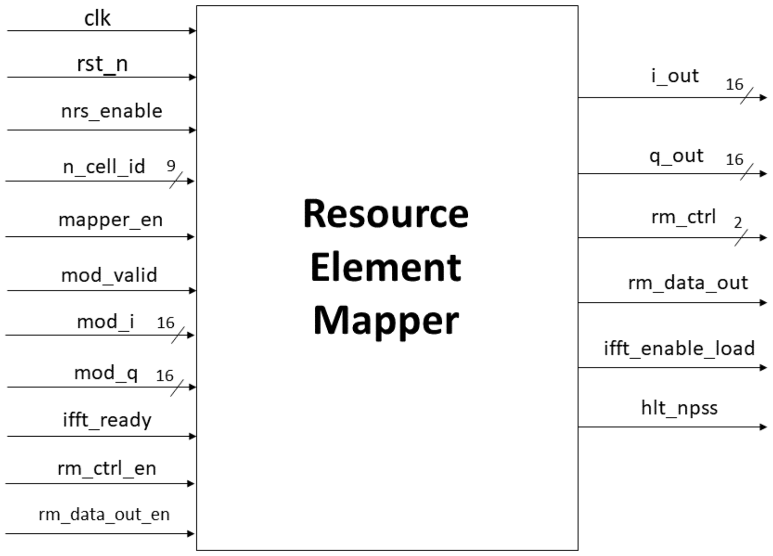


Figure 97: Block Interface of Resource Element Mapper

4.6.5.2. Architecture

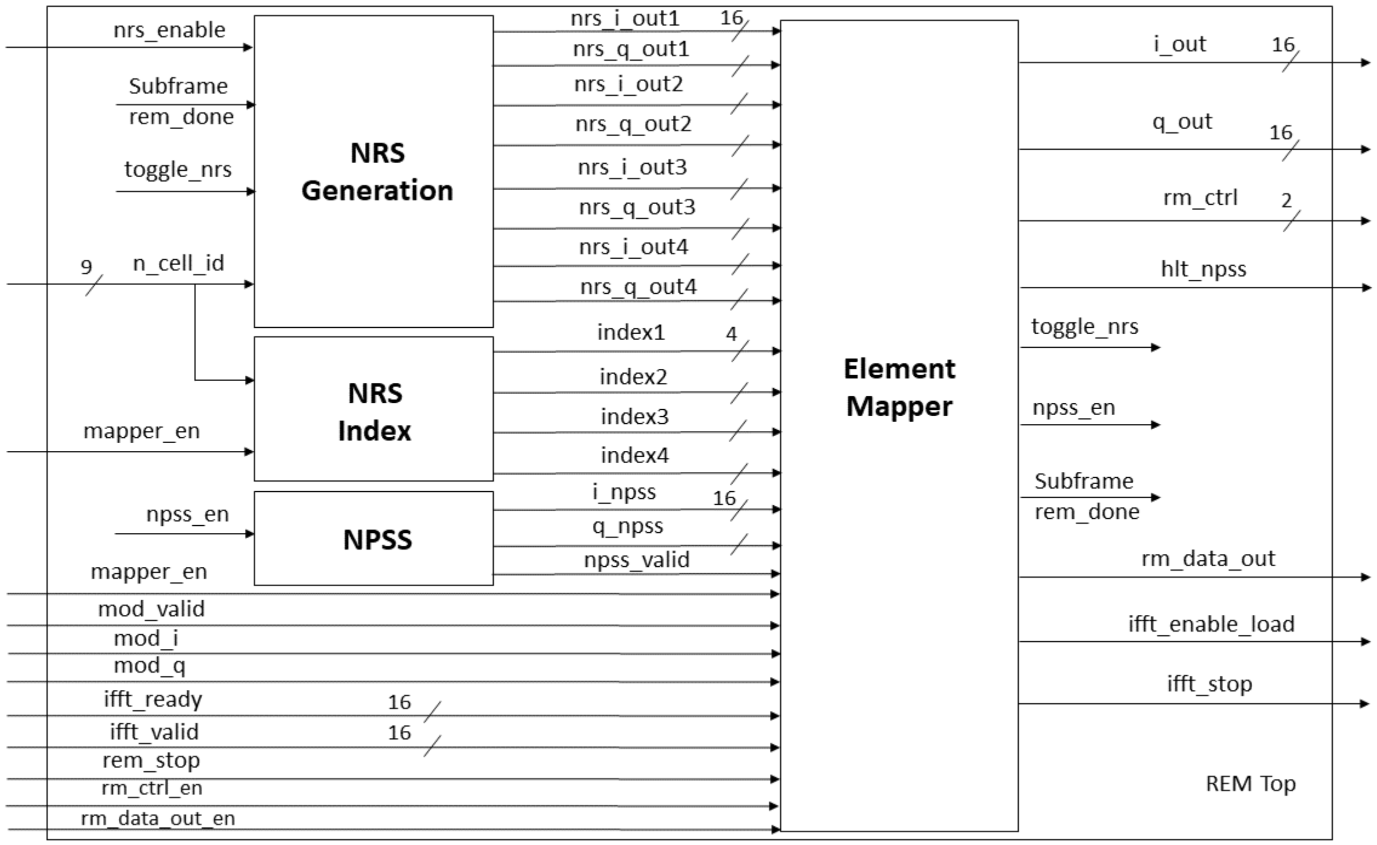


Figure 98: Top Level of Resource Element Mapper

4.6.5.3. Interface Table

Port	Direction	Width	Description
clk	Input	1 bit	System clock
rst_n	Input	1 bit	Block reset
nrs_enable	Input	1 bit	NRS Generation module enable enable
mapper_en	Input	1 bit	Block enable (for NRS index & Element mapper) modules
n_cell_id	Input	9 bits	Physical layer cell identity given from upper layer
mod_valid	Input	1 bit	Valid from modulator block accompanied with the sent symbol
rm_ctrl_en	Input	1 bit	From FSM to make the block send rm_ctrl signal to rate matcher
rm_data_out_en	Input	1 bit	From FSM to make the block send rm_data_out signal to rate matcher
mod_i	Input	16 bits	In-phase input QPSK symbol from modulator
mod_q	Input	16 bits	Quadrature-phase input QPSK symbol from modulator
ifft_ready	Input	1 bit	Ready signal from IFFT (ready to receive a symbol)
ifft_valid	Input	1 bit	Valid signal sent from the IFFT block
rem_stop	Input	1 bit	Control signal sent from the system’s FSM when the whole operation is done to stop the IFFT block
i_out	Output	16 bits	In-phase output QPSK symbol to IFFT
q_out	Output	16 bits	Quadrature-phase output QPSK symbol to IFFT
hlt_npss	Output	1 bit	Signal sent to FSM to stop sending bits until NPSS is done
rm_ctrl	Output	2 bits	Control signal sent to rate matcher to know whether sent 20 or 24 bits
rm_data_out	Output	1 bit	To rate matcher and FSM to make rate matcher start sending data
ifft_enable_load	Output	1 bit	Signal sent to the FSM to enable the IFFT to be ready to receive them
ifft_stop	Output	1 bit	Sent to ifft when the rem_stop signal reaches from the FSM

Table 14: Interface Table of REM top level

4.6.5.4. Complete Frame Scenario

- At first, the incoming QPSK symbols from the modulator are stored in a memory element, since each symbol is formed after 2 clock cycles, therefore the OFDM symbol takes 24 clock cycles to be stored.

- After one cycle, the OFDM symbol is stored in another memory waiting for the “ifft_ready” to send the In-phase & Quadrature symbols. This operation lasts for the first 5 OFDM symbols in the first slot.

- The last 2 OFDM symbols includes NRS symbols (pilots) and only 10 QPSK symbols from modulator to complete these OFDM symbols.

- The previous operation lasts for the first 10 slots (from 0 to 9).

- In slots 10 & 11, NPSS symbols are mapped with no pilots.

- In slots 12 to 19, the same mapping of QPSK symbols & NRS symbols takes place.

- From the system’s FSM, this block uses three status signals as an input :

    - **rm_ctrl_en** : the resource element mapper sends the number of bits configuration “rm_ctrl” (24 or 20 bits) to the Rate Matcher for one clock cycle.
    - **rm_data_out_en** : the signal “rm_data_out” is sent to the Rate Matcher in one of the following cases :
      a) When “rm_data_out_en” is high **and** “ifft_ready” is high

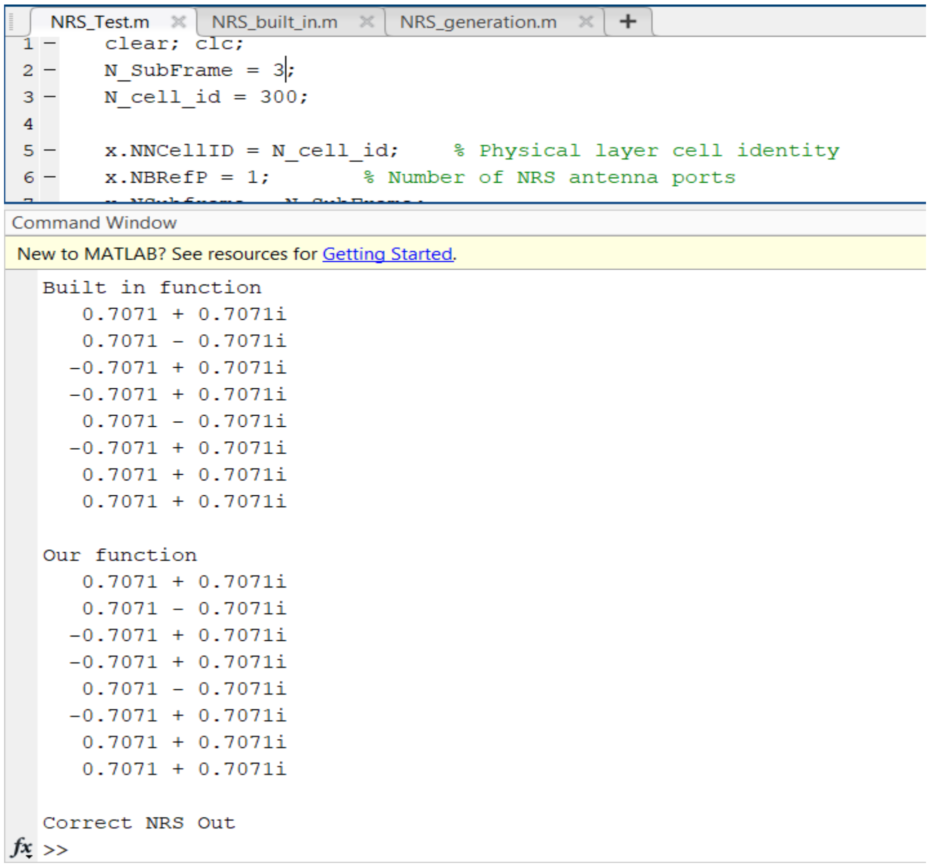
      b) When “rm_data_out_en” is high **and** (we’re mapping the first two OFDM symbols)
    - **rem_stop :** to stop the IFFT block when the whole operation is done

- To the system’s FSM, this block sends three control signals as an output :
    - **hlt_npss :** to stop sending data from modulator until NPSS is mapped.
    - **ifft_enable_load :** to enable the IFFT to be ready to receive the outcoming symbols from REM.
    - **rm_data_out :** to make rate matcher start sending data bits according the bit-configuration.

➢ **Block Latency :** 1625 clock cycles

4.6.6. Simulation results of NRS Generation

First, we compared our function to the MATLAB built-in function and it results in identical pilots values at cell id = 300 & sub-frame number 3.



NRS_Test.m | NRS_built_in.m | NRS_generation.m

```
    clear; clc;
    N_SubFrame = 3;
    N_cell_id = 300;

    x.NNCellID = N_cell_id;     % Physical layer cell identity
    x.NBRefP = 1;         % Number of NRS antenna ports
```

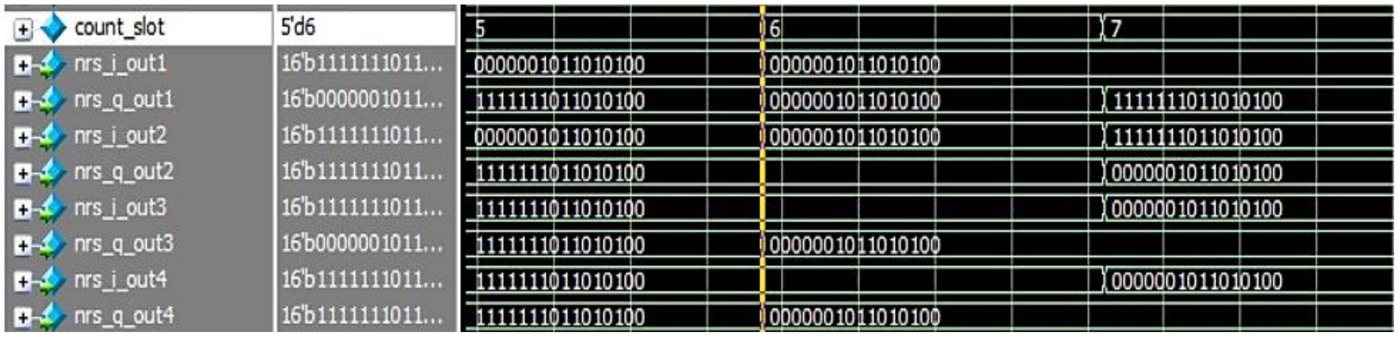
Command Window

New to MATLAB? See resources for Getting Started.

```
Built in function
   0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 + 0.7071i
   0.7071 + 0.7071i

Our function
   0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
  -0.7071 + 0.7071i
   0.7071 - 0.7071i
  -0.7071 + 0.7071i
   0.7071 + 0.7071i
   0.7071 + 0.7071i

Correct NRS Out
fx >>
```

Figure 99: MATLAB comparison of NRS values at cell_id=300 & subframe=3

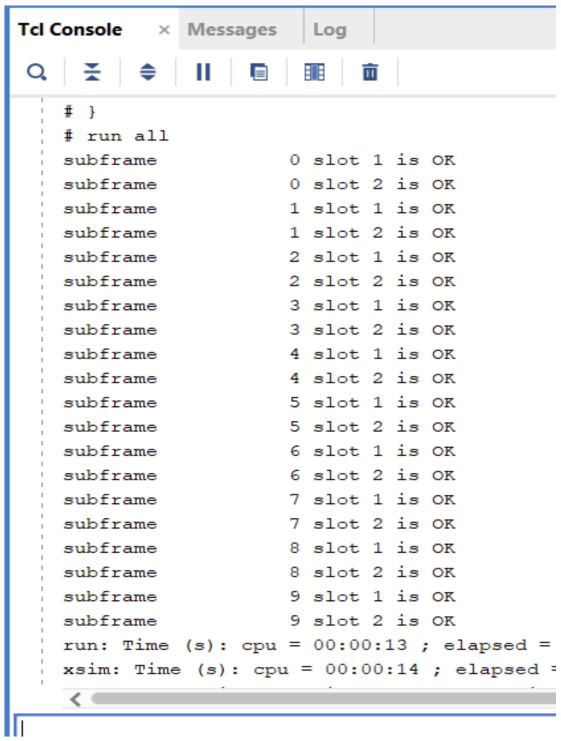
After that, the RTL was tested using the same parameters (cell id = 300 & sub-frame number 3) and it is clear that at slot 6 (subframe 3) the pilots sign is identical to that generated from Matlab.



```
count_slot   5'd6               5                   6                   7
nrs_i_out1   16'b1111111011...  0000001011010100    0000001011010100
nrs_q_out1   16'b0000001011...  1111111011010100    0000001011010100    1111111011010100
nrs_i_out2   16'b1111111011...  0000001011010100    0000001011010100    1111111011010100
nrs_q_out2   16'b1111111011...  1111111011010100                        0000001011010100
nrs_i_out3   16'b1111111011...  1111111011010100                        0000001011010100
nrs_q_out3   16'b0000001011...  1111111011010100    0000001011010100
nrs_i_out4   16'b1111111011...  1111111011010100                        0000001011010100
nrs_q_out4   16'b1111111011...  1111111011010100    0000001011010100
```

Figure 100: RTL results of NRS values at cell_id=300 & subframe=3

Finally, a whole frame (10 sub-frames) is tested to check its pilots values, we used our MATLAB function to verify our RTL using automated test-bench.

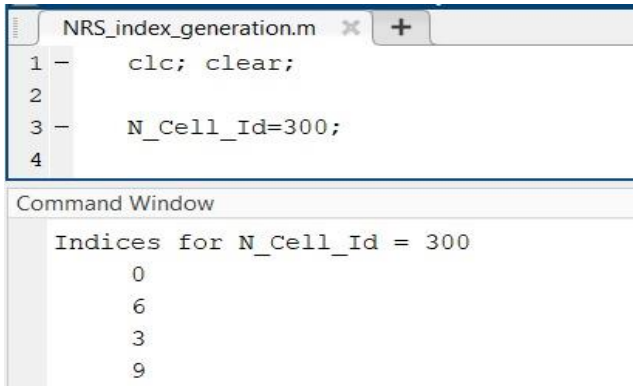


```
# }
# run all
subframe          0 slot 1 is OK
subframe          0 slot 2 is OK
subframe          1 slot 1 is OK
subframe          1 slot 2 is OK
subframe          2 slot 1 is OK
subframe          2 slot 2 is OK
subframe          3 slot 1 is OK
subframe          3 slot 2 is OK
subframe          4 slot 1 is OK
subframe          4 slot 2 is OK
subframe          5 slot 1 is OK
subframe          5 slot 2 is OK
subframe          6 slot 1 is OK
subframe          6 slot 2 is OK
subframe          7 slot 1 is OK
subframe          7 slot 2 is OK
subframe          8 slot 1 is OK
subframe          8 slot 2 is OK
subframe          9 slot 1 is OK
subframe          9 slot 2 is OK
run: Time (s): cpu = 00:00:13 ; elapsed =
xsim: Time (s): cpu = 00:00:14 ; elapsed =
```

Figure 101: Testing a wholre frame pilots using automated testbench

4.6.7. Simulation results of NRS Index

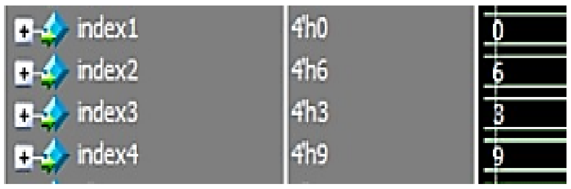
We compared the MATLAB function with the generated values from RTL at the same cell ID and it is noticeable that the indices are identical



```
clc; clear;

N_Cell_Id=300;
```

```
Indices for N_Cell_Id = 300
     0
     6
     3
     9
```

Figure 103: Matlab results of NRS indices at cell_id=300



index1	4'h0	0
index2	4'h6	6
index3	4'h3	3
index4	4'h9	9

Figure 102: RTL results of NRS indices at cell_id=300

4.6.8. Simulation results of the TOP module

In the figure below, we can notice that the output QPSK data symbols from the modulator are mapped and delivered to the IFFT when it requests that symbol

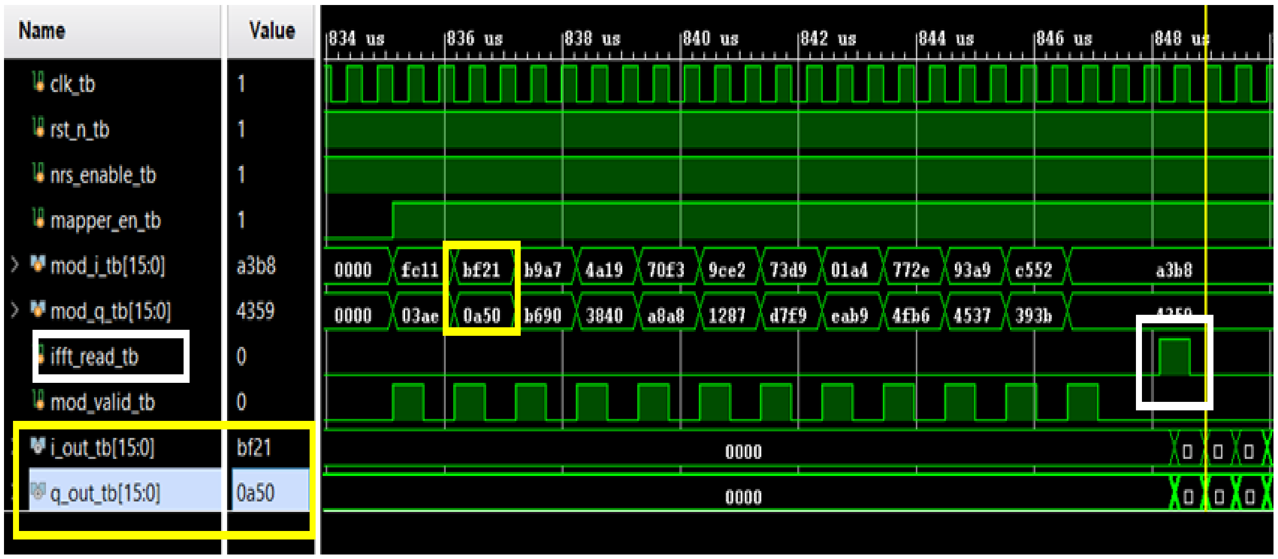
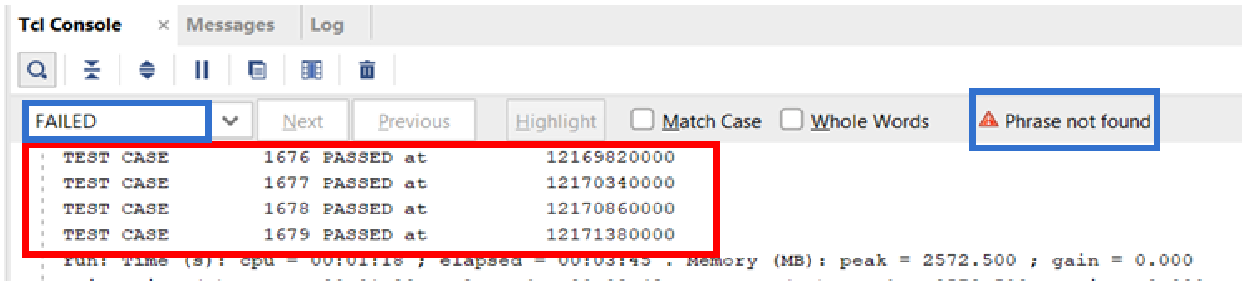


Figure 104: Output of the REM is the QPSK symbols requested by the IFFT

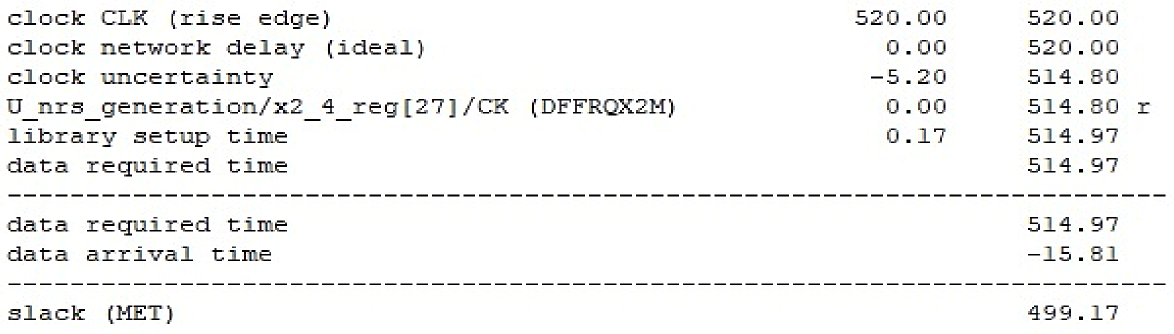
In the figure below, a whole frame (20x7x12 = 1680 QPSK symbols) is tested using automated test-bench and all test cases passed which includes (Data, Pilots, and Synchronization)



```
TEST CASE         1676 PASSED at          12169820000
TEST CASE         1677 PASSED at          12170340000
TEST CASE         1678 PASSED at          12170860000
TEST CASE         1679 PASSED at          12171380000
run: Time (s): cpu = 00:01:18 ; elapsed = 00:03:45 . Memory (MB): peak = 2572.500 ; gain = 0.000
```

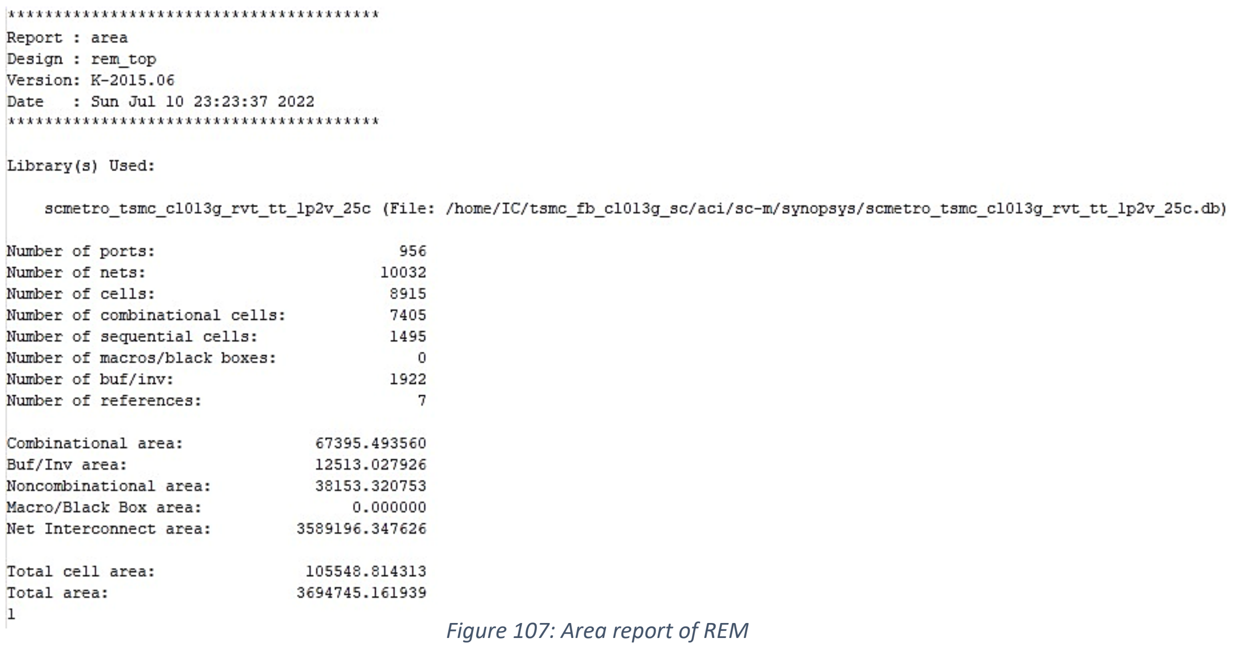
Figure 105: Test cases for a complete frame using automated testbench

4.6.9. Synthesis Results



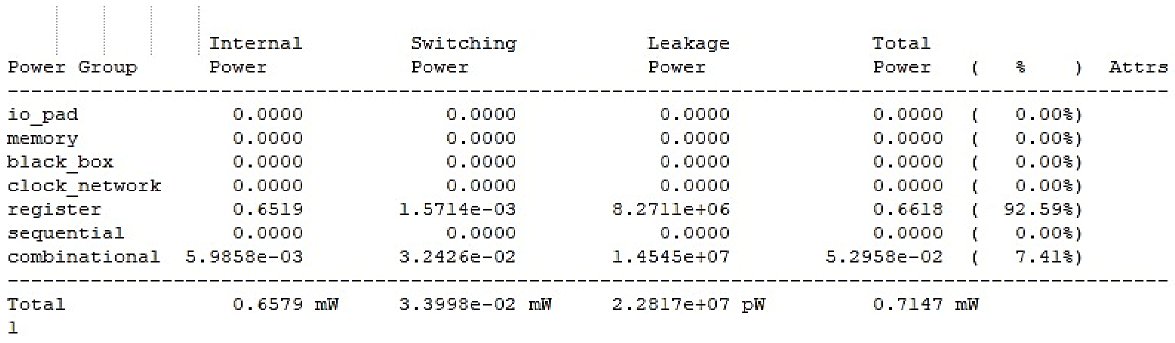
```
clock CLK (rise edge)                                   520.00     520.00
clock network delay (ideal)                               0.00     520.00
clock uncertainty                                        -5.20     514.80
U_nrs_generation/x2_4_reg[27]/CK (DFFRQX2M)               0.00     514.80 r
library setup time                                        0.17     514.97
data required time                                                 514.97
---------------------------------------------------------------------------
data required time                                                 514.97
data arrival time                                                  -15.81
---------------------------------------------------------------------------
slack (MET)                                                        499.17
```

Figure 106: Timing_max (setup time) report of REM



```
****************************************
Report : area
Design : rem_top
Version: K-2015.06
Date   : Sun Jul 10 23:23:37 2022
****************************************

Library(s) Used:

    scmetro_tsmc_cl013g_rvt_tt_1p2v_25c (File: /home/IC/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_tt_1p2v_25c.db)

Number of ports:                          956
Number of nets:                         10032
Number of cells:                         8915
Number of combinational cells:           7405
Number of sequential cells:              1495
Number of macros/black boxes:               0
Number of buf/inv:                       1922
Number of references:                       7

Combinational area:             67395.493560
Buf/Inv area:                   12513.027926
Noncombinational area:          38153.320753
Macro/Black Box area:               0.000000
Net Interconnect area:        3589196.347626

Total cell area:               105548.814313
Total area:                   3694745.161939
1
```

Figure 107: Area report of REM



```
                  Internal         Switching           Leakage            Total
Power Group       Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad               0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory               0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box            0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network        0.0000           0.0000              0.0000             0.0000  (   0.00%)
register             0.6519       1.5714e-03          8.2711e+06             0.6618  (  92.59%)
sequential           0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational    5.9858e-03       3.2426e-02          1.4545e+07         5.2958e-02  (   7.41%)
--------------------------------------------------------------------------------------------------
Total                0.6579 mW    3.3998e-02 mW       2.2817e+07 pW          0.7147 mW
1
```

Figure 108: Power report of REM

4.7. IFFT & CP

IFFT is an algorithm used to compute IDFT to convert signals from frequency domain to time domain. IDFT is a useful operation but computing it directly from its definition is not a practical method, so we go to IFFT which rapidly computes the operation by breaking IDFT into smaller IDFTs (dived and conquer algorithm). Which decreases the complexity of computing from $O(N^2)$ of IDFT to $O(N \log_2 N)$ of IFFT where N is the data size.

IFFT equation $X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{j2\pi kn/N}\ where\ k = 0, \ldots, N-1,$

The most common algorithm of IFFT is the Cooley–Tukey algorithm, where radix-2 is the simplest form of the algorithm which is used in this thesis, there are two forms in radix-2 which are decimation in time (DIT) and decimation in frequency(DIF). For DIT the input is bit reversed while the output is a natural order. While DIF the input is in natural order while the output is bit reversed order, and the DIF butterfly is slightly different from DIT where in DIF the complex multiplication takes place after the add-subtract operation. But both requires N $\log_2$N operation to compute the DFT. Both algorithms can done in-place and both need to perform bit reversal at some place during the computation, the below figures show the two forms, in our thesis we choose DIT algorithm.

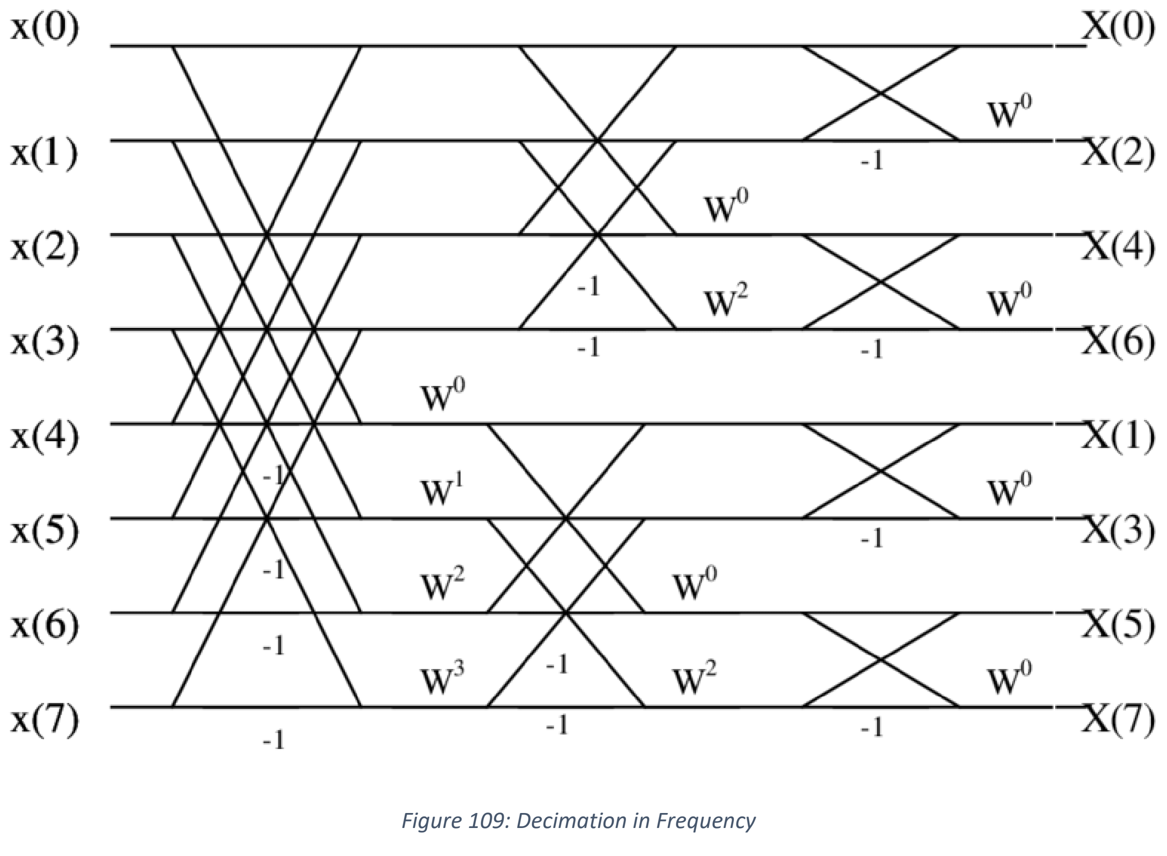


Figure 109: Decimation in Frequency

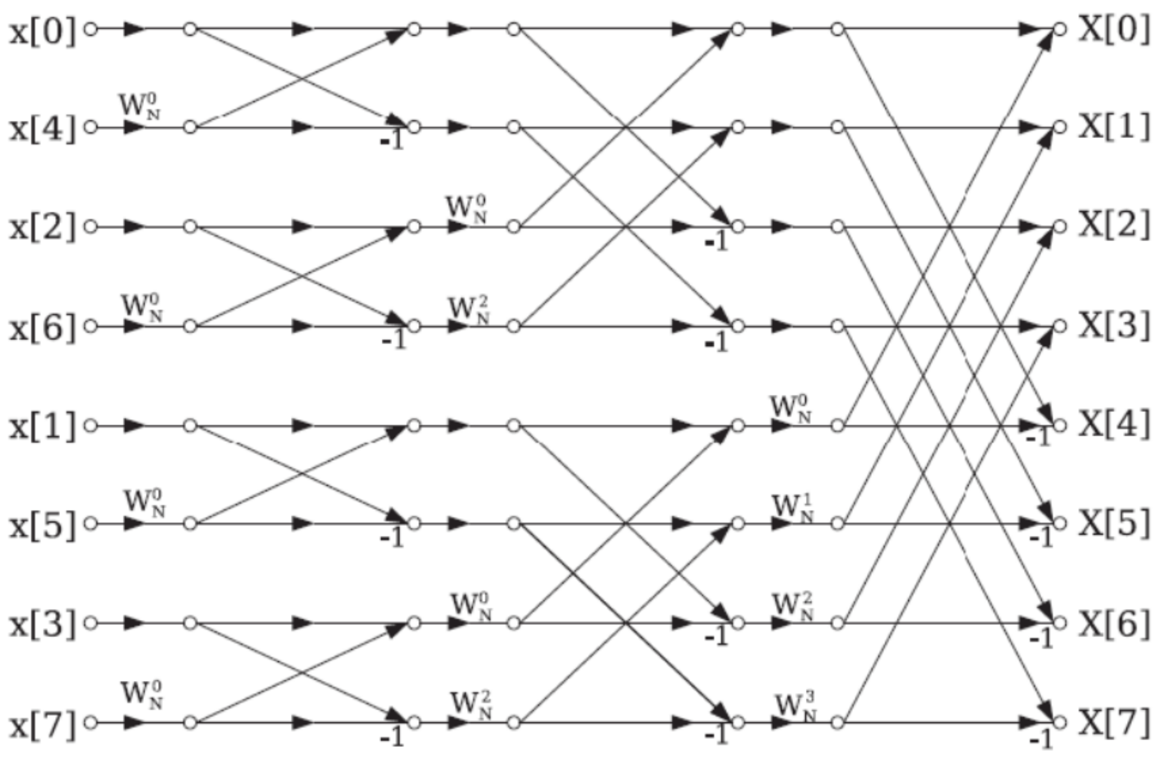


Figure 110: Decimation in Time

The most common architecture to implement the IFFT is Memory-based architecture and pipeline architecture we will take single-delay feedback (SDF) as an example of the pipeline architecture, here is a comparison between them.

Point of Comparison	Memory-based	SDF
Complex multipliers	1	3
Complex Adders	2	8
RAM Size (Depth)	16	15
latency(cycles)	34	16

Table 15: Comparison between Memory-based and SDF architectures

This comparison based on 16-point IFFT radix-2.

Memory-based architecture

In NB-IOT the target devices are intended to operate for a long time, so we want to implement a design has a long-life battery with low power consumption, so we choose Memory-based architecture to be implemented due its small area which will reflect in power consumption unlike the single-delay feedback architecture which consume more power.

This architecture consists of four single port RAMs, seven multiplexers, two adders, one multiplier, one ROM, and one controller. The four single-port RAMs are used for buffering the computational data. The multiplexers are responsible for switching the data flow between the storage and arithmetic components. The adder and multiplier execute the computation of the two-point IFFT.

The ROM stores the twiddle factors. The addend and augend of the left adder can be changed by controlling the Ch signal. For example, if Ch=0, then the adder executes. However, if Ch=1, then the adder executes. The controller, which is not shown in Figure 111, generates the controlling signals for the multiplexers and the four RAMs, eventually, the output is shift right by 16 as the IFFT equation stated.

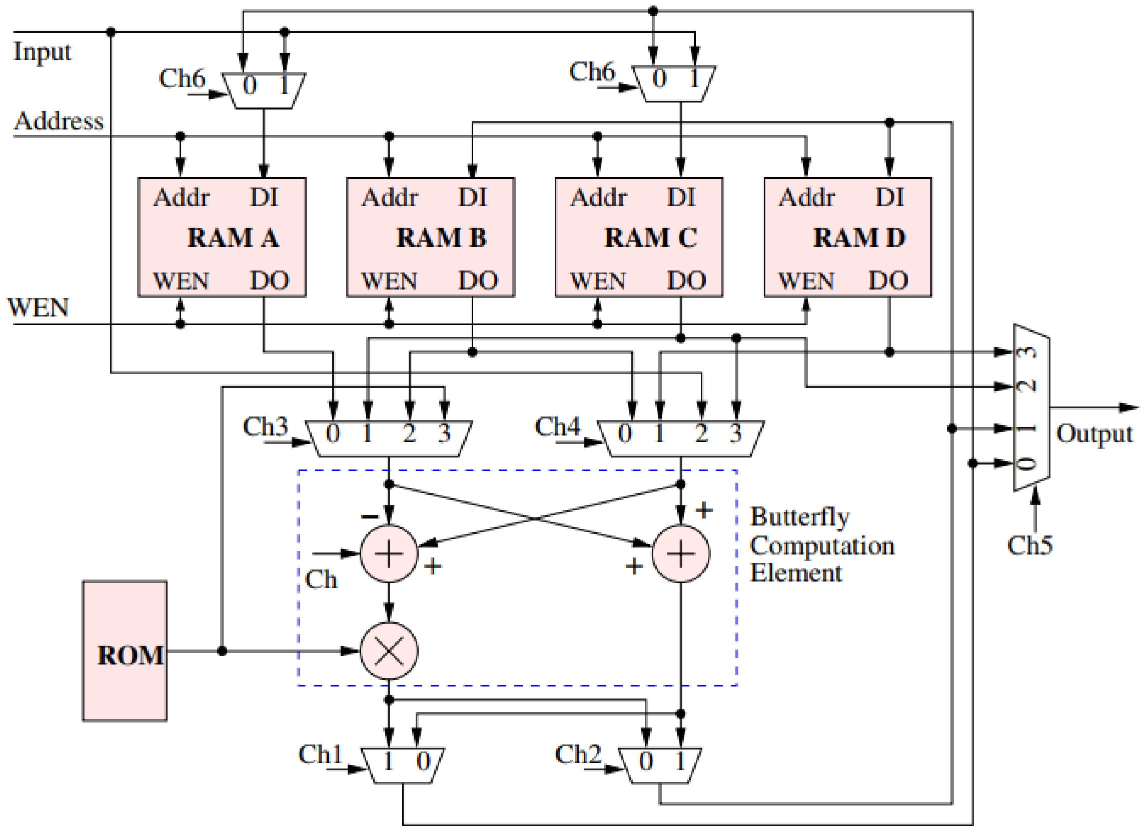


Figure 111: Memory-based Architecture

The signal flow graph explains the operation of the proposed memory-based FFT processor as shown in Figure 112.

At first 8 cycles the first 8 frequency domain signals are stored in memory banks A & C, then in each cycle a frequency domain signal is added and subtracted from the corresponding signal which is stored previously in the memory banks, this will lead to get out 2 signals each cycle, so the second 8 cycles will generate 16 output signal from the first stage where the last 8 signals are multiplied with their corresponding twiddle factor as shown in the signal flow graph which is follow the DIT butterfly algorithm. The process is going on with divide and conquer algorithm so in the second stage the upper 8 signals operated separately from the low 8 signals and so on until x[0] is got out in the cycle number 34 and the rest of the output will get out continuously until x[15] is got out at cycle number 48.

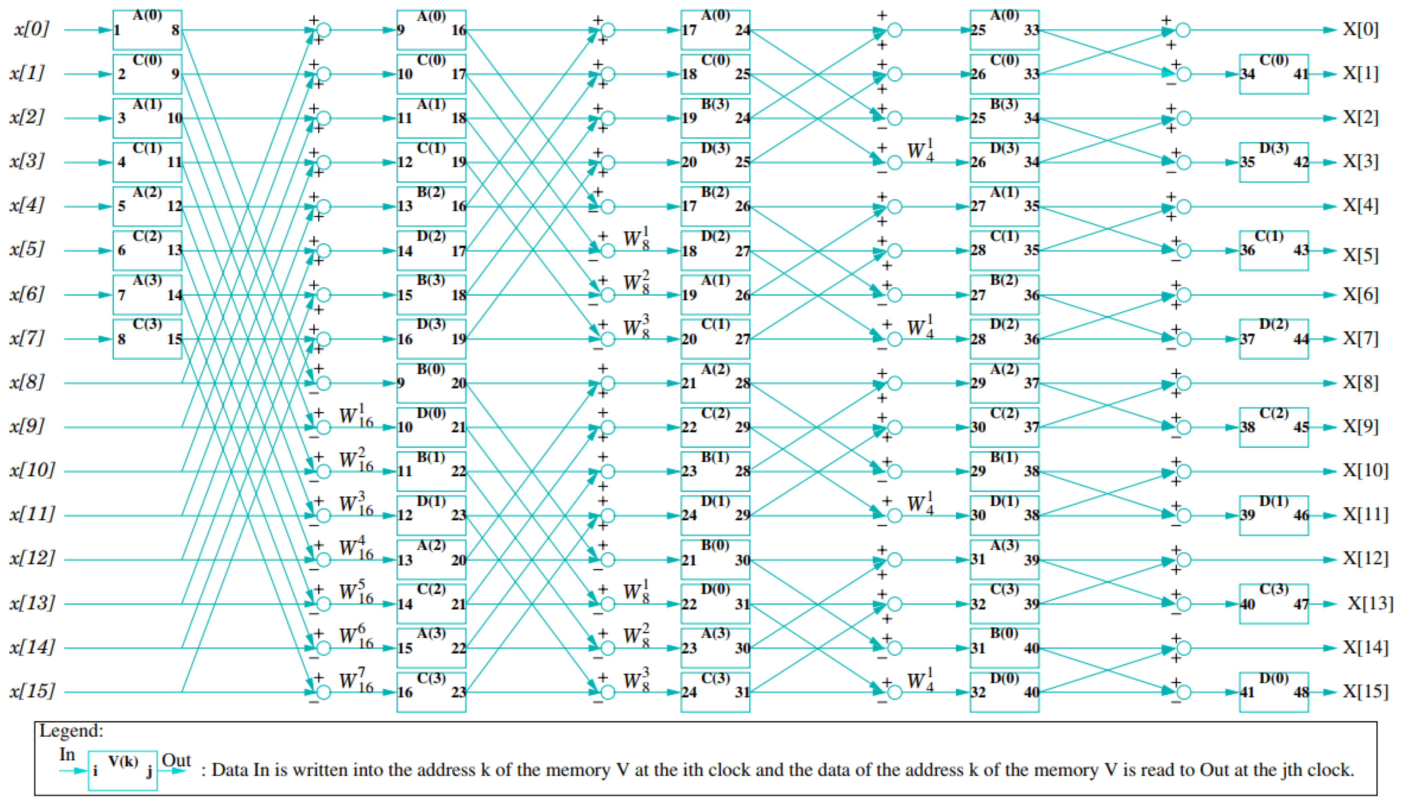


Figure 112: Signal Flow Graph for 16-point FFT

As there are nearly 16 control signals for {Ch, Ch1, Ch2, Ch3, Ch4, Ch5, Ch6, Address A/B/C/D, Write_enable A/B/C/D, Address for ROM} each vary in each cycle from the 48 working cycles so we have nearly 768 different values for the control signals which will not be a good idea to write them value by value in the FSM of the IFFT.

So we write the required value in a text file, then by a PYTHON we write a script to read this text file to generate an automated Verilog code, so any required modification we just change in the text file and re-run the PYTHON script to generate the modified Verilog code.

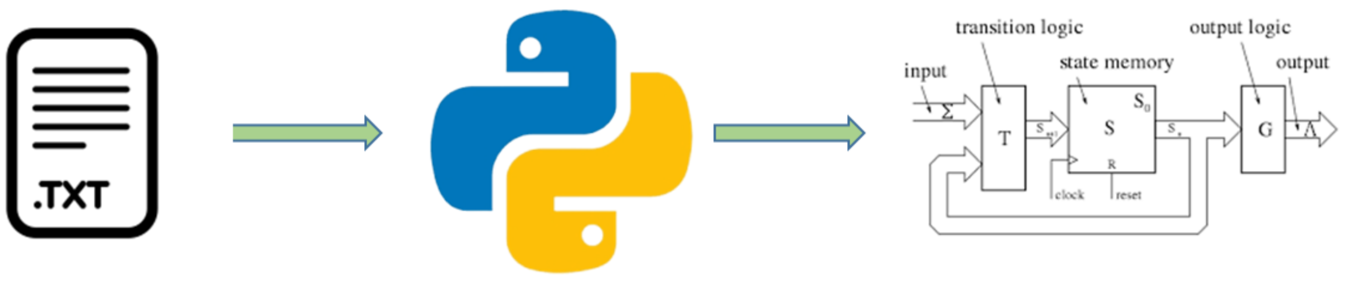


Figure 113: Automation

4.7.1. Block Interface

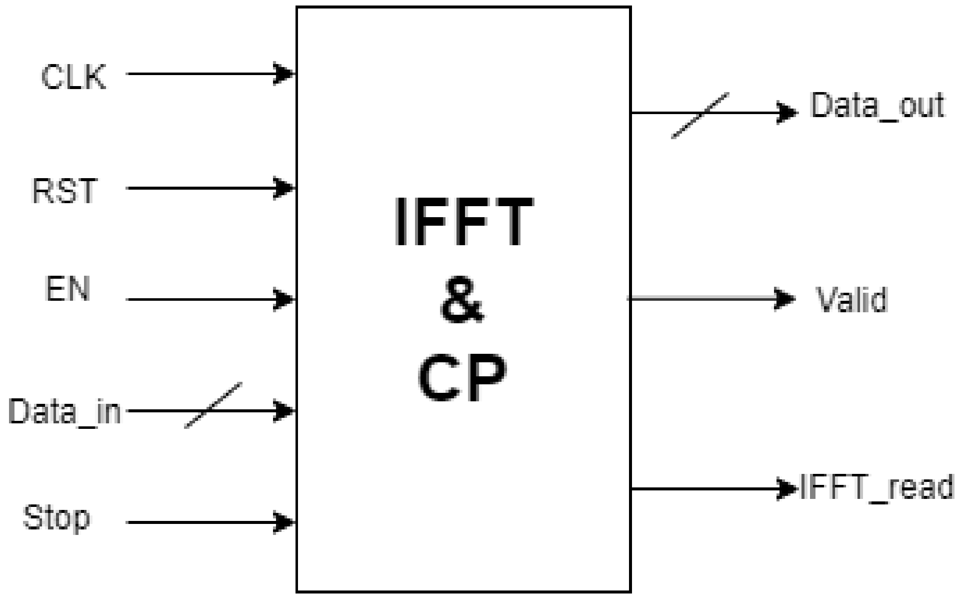


Figure 114: IFFT & CP Block Diagram

4.7.2. Interfaces Table

Port	Direction	Width	Description
CLK	Input	1-bit	System clock
RST	Input	1-bit	Asynchronous negative reset
EN	Input	1-bit	Enable signal to operate the IFFT, it’s connected from the REM
Data_in	Input	32-bit	It’s concatenation of the I and Q of the input data from the REM, first 16 bits are the I while the last 16 bits are the Q
Stop	Input	1-bit	It’s the signal which indicate that there no more data, so the valid will be zero after extracting the current OFDM symbo
Data_out	output	32-bit	It’s concatenation of the I and Q of the output data from the REM, first 16 bits are the I while the last 16 bits are the Q
Valid	output	1-bit	It’s the signal which indicates that the output data is valid to be read by the DAC
IFFT_read	output	1-bit	To read a new data from the REM.

Table 16: Block Interfaces

Architecture

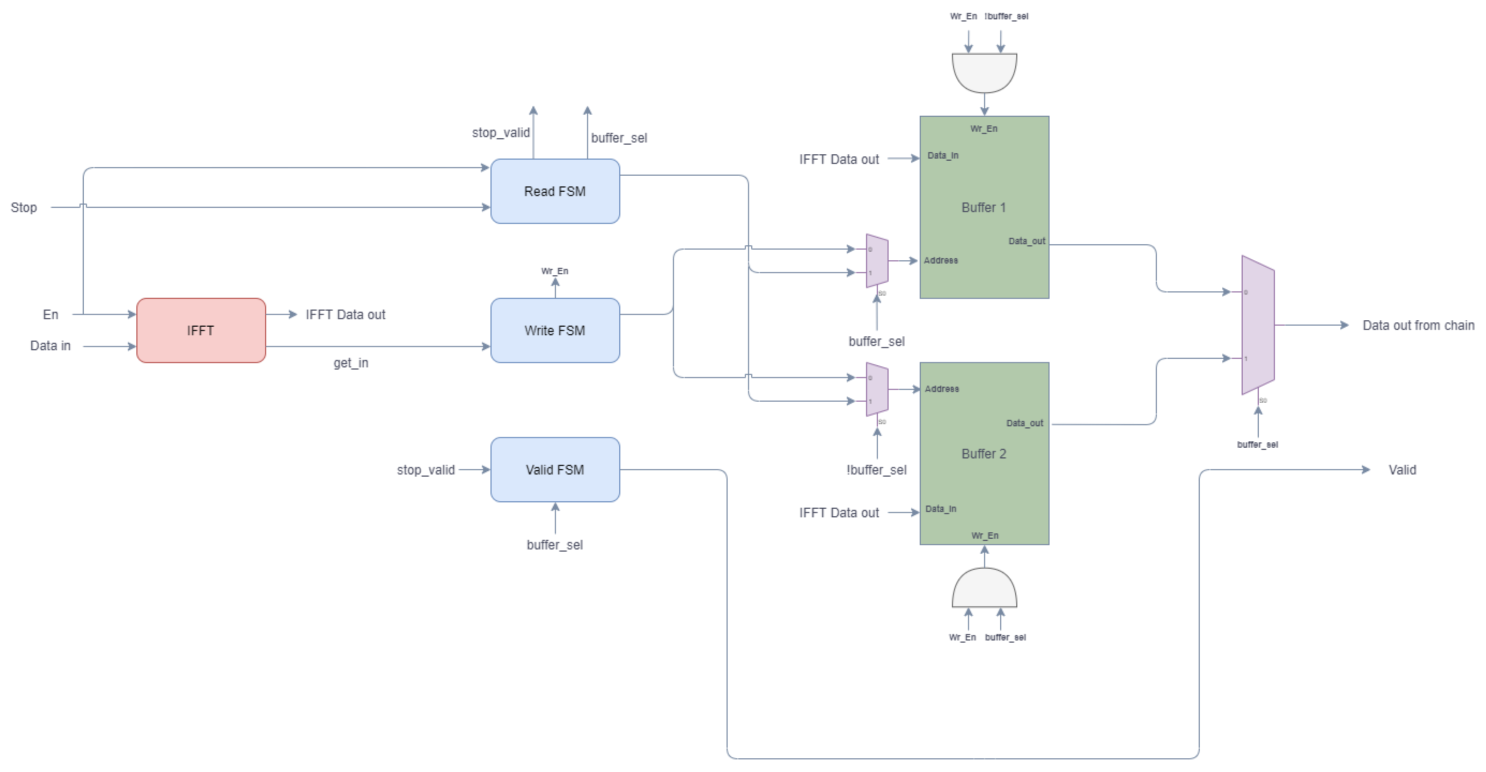


Figure 115: IFFT & CP Architecture

4.7.3. Blocks Description

The IFFT block which takes the input data and convert it to time domain with the pre-mentioned Memory-based architectures which is controlled by a FSM that is embedded in the IFFT block.

The write FSM is responsible for writing the output data from IFFT in the two buffers with an ordered manner.

The Read FSM is responsible for read the data from the buffer with their pre-arranged manner, and implicitly it's the block which add the cyclic prefix as it get the end of the data at the start of the OFDM symbol and then extract data in sequence.

The Valid FSM is responsible for indicating the output data validity, to be read by the next block which is the DAC.

The Output buffers are implemented for two reasons, first to reorder the output data as they are got out with a bit reversal order as we use DIT algorithm (input are ordered while the output is bit reversed), second reason to add the cyclic prefix as the data will be buffered so we can copy the end of the data to be get out as a prefix for the OFDM symbol.

4.7.4. Operation

First "EN" signal is triggered to be high then the data is get in from the next cycle of "EN" triggering until the 16 symbol enter the IFFT block, as mentioned before the first 8 cycles of the operation the IFFT is only storing the symbols in RAM A and RAM B, then from the 9th cycle the addition, subtraction and multiplication are applied on the data, till reach the 34th cycle at which the data are get out from the IFFT sample by sample, at the same time the "get_in" signal is triggered to activate the WRITE FSM to start write the data in the output buffer1.

Till now the "valid" signal is still zero, and the data is stored in output buffer 1, then new data are coming after 137 cycles from the first data (we will clarify why 137 cycle below), the "IFFT_read" is the signal which is responsible for claiming new data, then the new data take the same process as the previous data, while the stored data are getting out from the upper output buffer by the control of the READ FSM, and now the valid signal is activated to indicate that the output data is valid to be read.

The data in buffer 1 are read out in 137 cycles as follow, first 9 cycles are cyclic prefix while the next 128 cycles are useful data, the 2nd input data are stored in the output buffer 2 when they are processed by the IFFT block this storing is done within the 137 cycles of read the data in buffer 1, and the process go on as described, one buffer is storing while the other is extracting the data, they work alternately.

"buffer_sel" signal plays a vital role in alternate between reading and writing in the output buffers, this signal is generated from the READ FSM through an internal counter.

Eventually, when the data in the chain are read a "stop" signal will be activated from the system FSM, so the IFFT extract the last OFDM symbol then the "valid" is deactivated.

Figure 116 describes the time line of the expected system output.

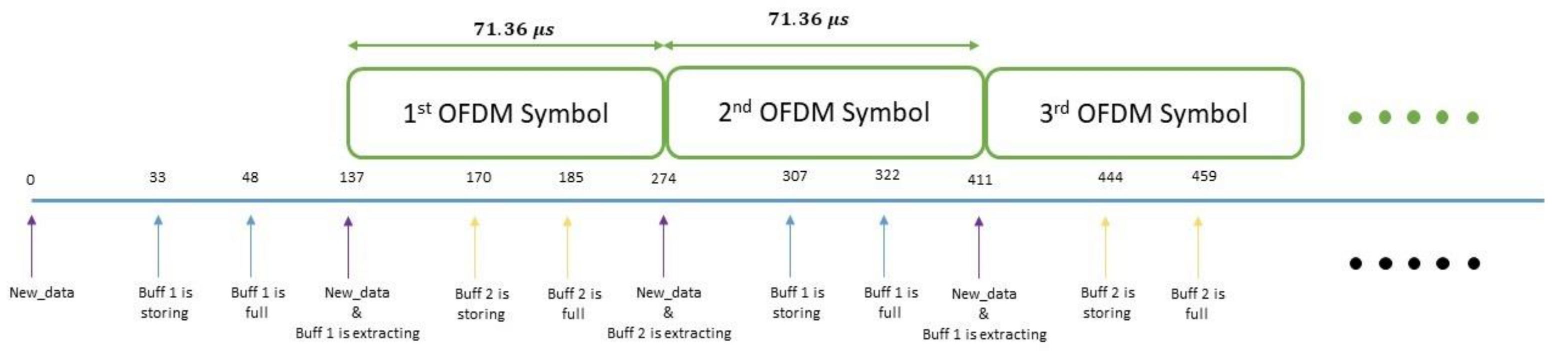


Figure 116: IFFT & CP time-line

The reason why we choose 137, is to achieve the cyclic prefix time which is determined in the standard, and here is a clarification.

Assume ‘x’ is repeated samples of the cyclic prefix, and ‘y’ is repeated samples of the data.

According to the standard, cyclic prefix duration is 4.69 µs while data duration is 66.67 µs

$$\frac{x}{x+16y} = \frac{4.69}{4.69+66.67}$$

To obtain these durations, we use the above equation, and by trial and error we get that

x = 9, when y = 8

This means that the total number of cycles to send an OFDM symbol is 9+16*8 = 137 cycle, the first 9 cycles is for cyclic prefix and the last 128 cycles is for data, so we’ll copy the tail of the OFDM symbol to the beginning of it, as shown in Figure 117.

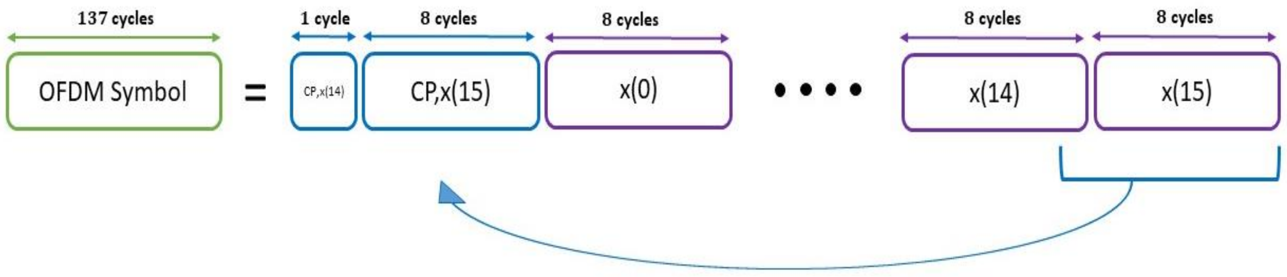


Figure 117: OFDM Symbol's 137 cycles

4.7.5. Clock Determination

As the IFFT is the last block in the chain, so it’s the bottleneck which determine the output rate, so to achieve the OFDM symbol duration according to the standard which is 71.36µs.

$$137 * T_{clk} = 71.36\mu s$$

$$f_{clk} = 1.92MHz$$

We operate the whole chain with this frequency and depend on halting some blocks by the system FSM if these blocks can operate with higher frequency.

4.7.6. Future Work

In this thesis we present the IFFT and CP design which extracts the data and stands for 8 cycles for each sample, which is not a practical situation especially from the communication wise, so it’s recommended to implement 128-point IFFT instead of 16-point, by insert the data in 12 symbols to IFFT (as it is the allocated number of subcarriers according to the standard) and the other symbols are padded with zero, then copy the last 9 samples of the 128 output samples from the IFFT as a cyclic prefix, it’s the ideal situation so the design is ready to get out as a real product.

4.7.7. Simulation Results

Before presenting the simulation and results, we should first present the verification procedure which facilitate the way to simulations and get accurate results.

1- We generate random complex numbers using MATLAB and export these numbers in two formats fixed point format and floating point format, all the generated input are combinations of ±1/sqrt(2) because our modulation scheme is QPSK, as shown in Figures 118, 119.

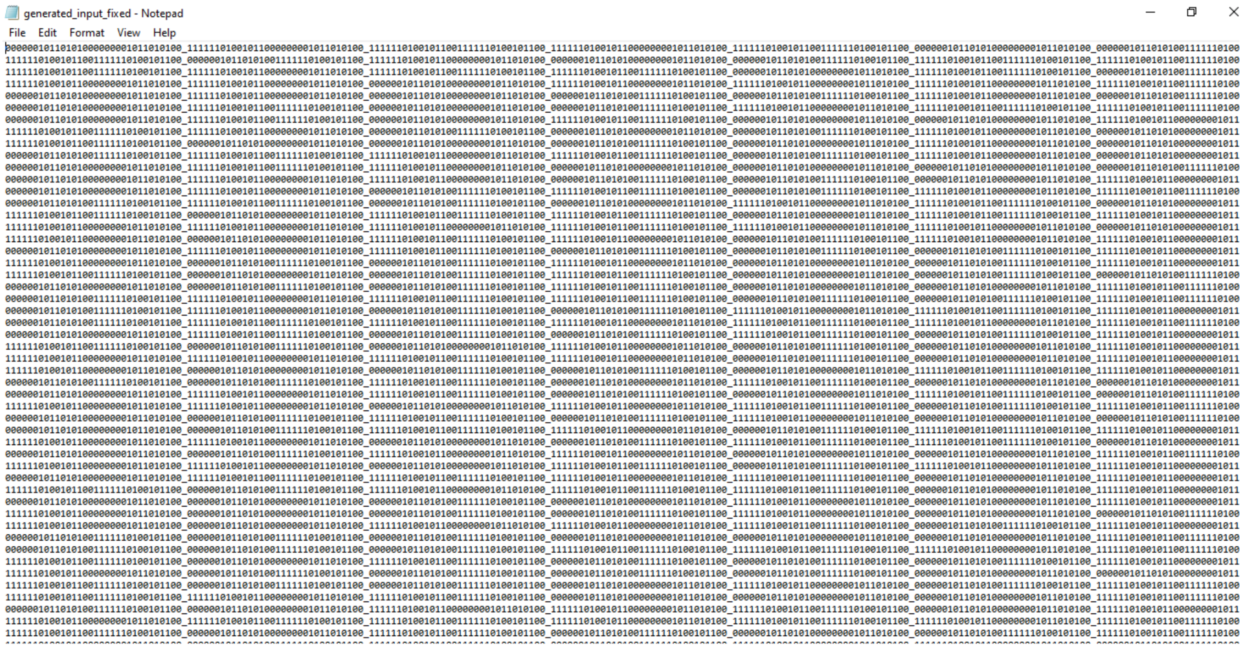


Figure 118: Generated Input (fixed)

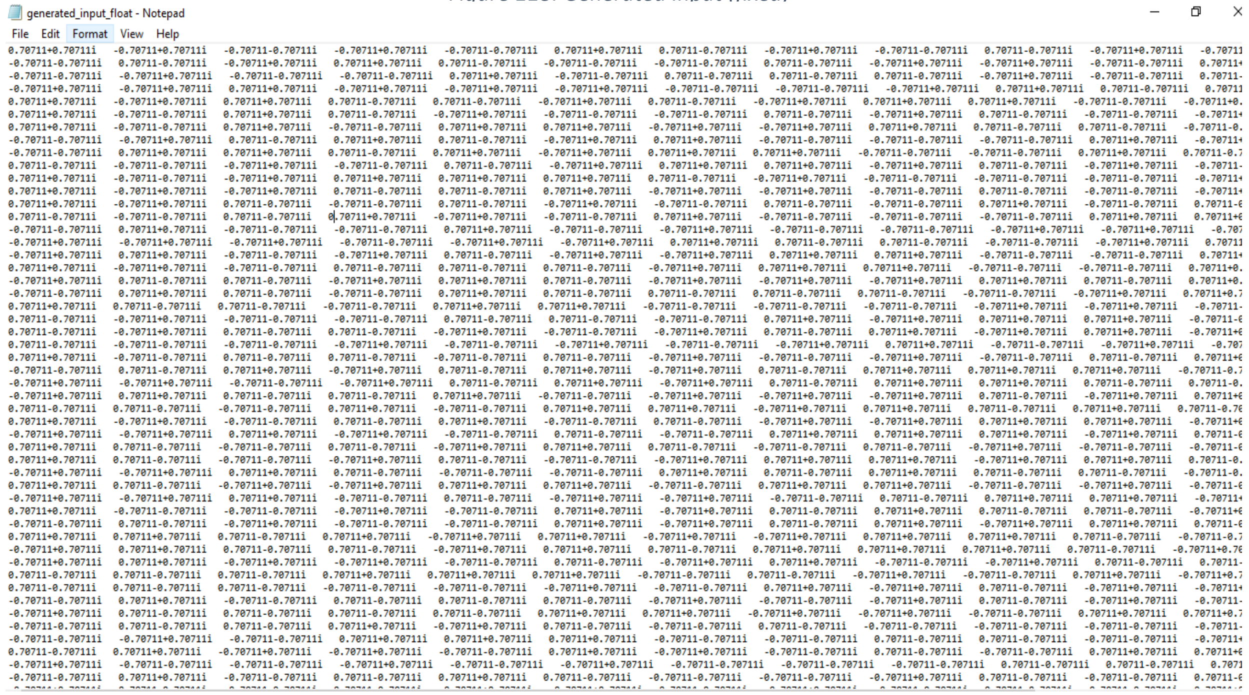
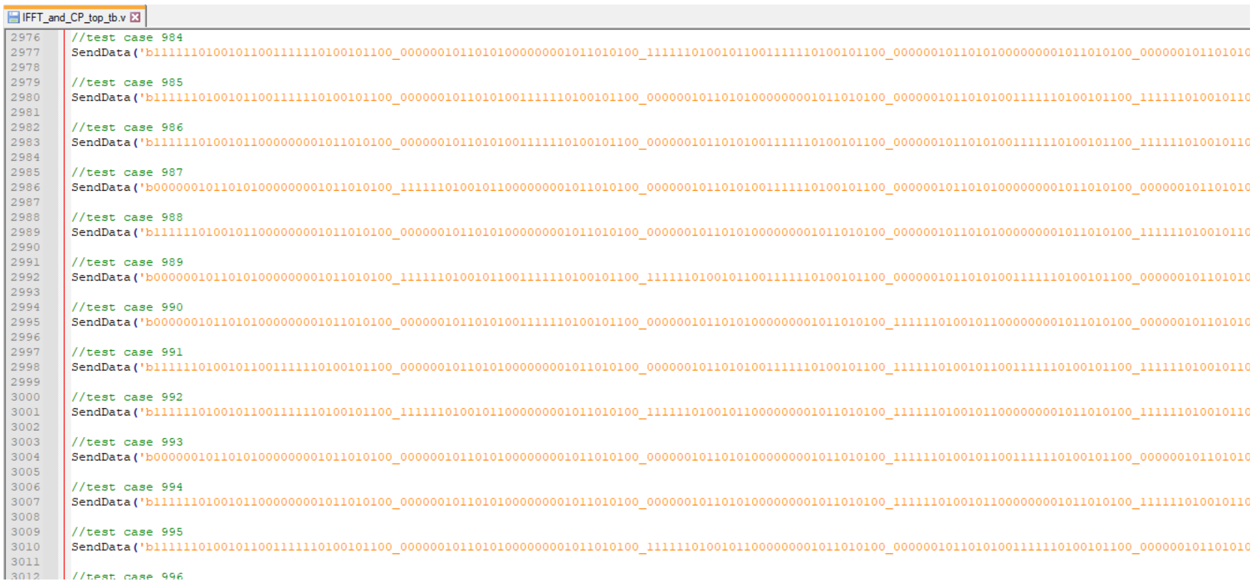


Figure 119: Generated Input (float)

2- We take the generated input which is in fixed format to a **Python** script so it generates a Verilog code that can be inserted in the testbench as in Figure 120.

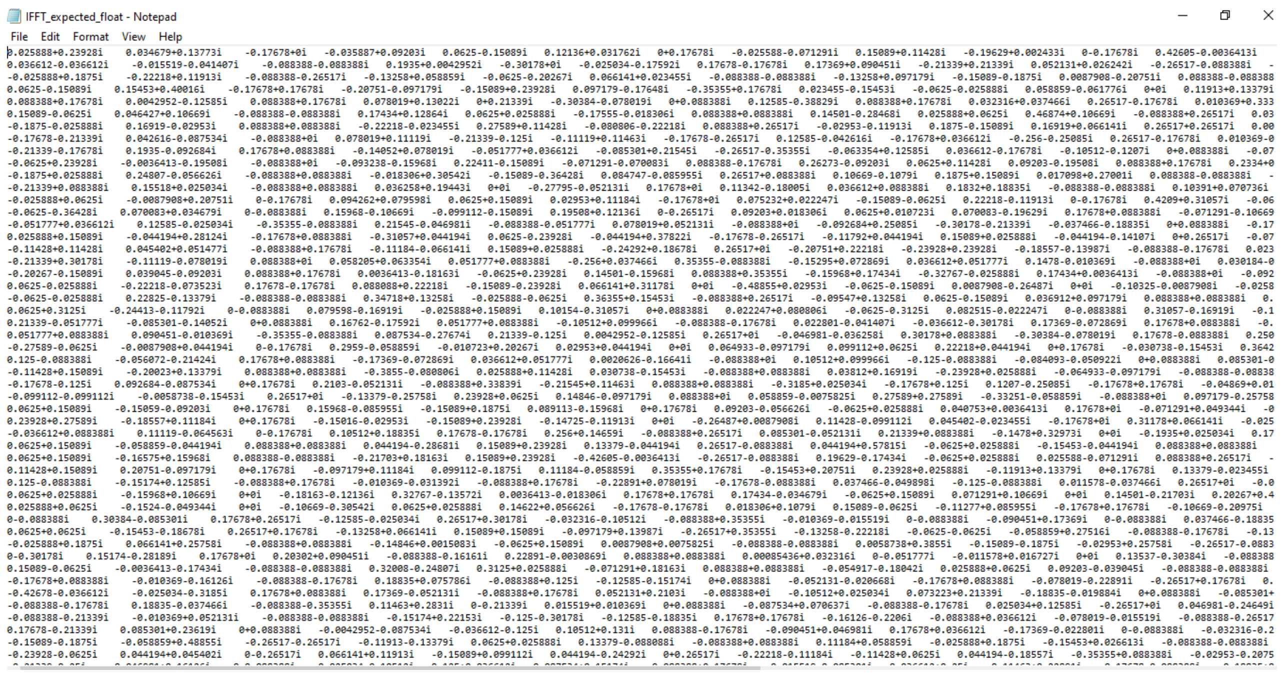


IFFT_and_CP_top_tb.v

```
//test case 984
SendData('b11111101001011001111110100101100_00000010110101000000001011010100_11111101001011001111110100101100_00000010110101000000001011010100_000000101101010

//test case 985
SendData('b11111101001011001111110100101100_00000010110101001111110100101100_00000010110101000000001011010100_00000010110101001111110100101100_111111010010110

//test case 986
SendData('b11111101001011000000001011010100_00000010110101001111110100101100_00000010110101001111110100101100_00000010110101001111110100101100_111111010010110

//test case 987
SendData('b00000010110101000000001011010100_11111101001011000000001011010100_00000010110101001111110100101100_00000010110101000000001011010100_000000101101010

//test case 988
SendData('b11111101001011000000001011010100_00000010110101000000001011010100_00000010110101000000001011010100_00000010110101000000001011010100_111111010010110

//test case 989
SendData('b00000010110101000000001011010100_11111101001011001111110100101100_11111101001011001111110100101100_00000010110101001111110100101100_000000101101010

//test case 990
SendData('b00000010110101000000001011010100_00000010110101001111110100101100_00000010110101000000001011010100_11111101001011000000001011010100_000000101101010

//test case 991
SendData('b11111101001011001111110100101100_00000010110101000000001011010100_00000010110101001111110100101100_11111101001011001111110100101100_111111010010110

//test case 992
SendData('b11111101001011001111110100101100_11111101001011000000001011010100_11111101001011000000001011010100_11111101001011000000001011010100_111111010010110

//test case 993
SendData('b00000010110101000000001011010100_00000010110101000000001011010100_00000010110101000000001011010100_11111101001011001111110100101100_000000101101010

//test case 994
SendData('b11111101001011000000001011010100_00000010110101000000001011010100_00000010110101000000001011010100_11111101001011000000001011010100_111111010010110

//test case 995
SendData('b11111101001011001111110100101100_00000010110101000000001011010100_11111101001011000000001011010100_00000010110101001111110100101100_000000101101010

//test case 996
```

Figure 120: Generated Test-bench fragment

3- We also generate the expected output from the IFFT in a text file in floating point format as shown in Figure 121, here there is no need to generate the fixed format.



IFFT_expected_float - Notepad

```
0.025888+0.23928i   0.034679+0.13773i   -0.17678+0i   -0.035887+0.09203i   0.0625-0.15089i   0.12136+0.031762i   0+0.17678i   -0.025588-0.071291i   0.15089+0.11428i   -0.19629+0.002433i   0-0.17678i   0.42605-0.0036413i
0.036612-0.036612i   -0.015519-0.041407i   -0.088388-0.088388i   0.1935+0.0042952i   -0.30178+0i   -0.025034-0.17592i   0.17678-0.17678i   0.17369+0.090451i   -0.21339+0.21339i   0.052131+0.026242i   -0.26517-0.088388i   -
-0.025888+0.1875i   -0.22218+0.11913i   -0.088388-0.26517i   -0.13258+0.058859i   -0.0625-0.20267i   0.066141+0.023455i   -0.088388-0.088388i   -0.13258+0.097179i   -0.15089-0.1875i   0.0087908-0.20751i   0.088388-0.088388
0.0625-0.15089i   0.15453+0.40016i   -0.17678+0.17678i   -0.20751-0.097179i   -0.15089+0.23928i   0.097179-0.17648i   -0.35355+0.17678i   0.023455-0.15453i   -0.0625-0.025888i   0.058859-0.061776i   0+0i   0.11913+0.13379i
0.088388+0.17678i   0.0042952-0.12585i   0.088388+0.17678i   0.078019+0.13022i   0+0.21339i   -0.30384-0.078019i   0+0.088388i   0.12585-0.38829i   0.088388+0.17678i   0.032316+0.037466i   0.26517-0.17678i   0.010369+0.333
0.15089-0.0625i   0.046427+0.10669i   -0.088388-0.088388i   0.17434+0.12864i   0.0625+0.025888i   -0.17555-0.018306i   0.088388+0.088388i   0.14501-0.28468i   0.025888+0.0625i   0.46874+0.10669i   -0.088388+0.26517i   0.03
-0.1875-0.025888i   0.16919-0.02953i   0.088388+0.088388i   -0.22218-0.023455i   0.27589+0.11428i   -0.080806-0.22218i   0.088388+0.26517i   -0.02953-0.11913i   0.1875-0.15089i   0.16919+0.066141i   0.26517+0.26517i   0.00
-0.17678-0.21339i   0.042616-0.087534i   -0.088388+0i   0.078019+0.11119i   -0.21339-0.125i   -0.11119+0.11463i   -0.17678-0.26517i   0.12585-0.042616i   -0.17678+0.036612i   -0.256-0.25085i   0.26517-0.17678i   0.010369-0
-0.21339-0.17678i   0.1935-0.092684i   0.17678+0.088388i   -0.14052+0.078019i   -0.051777+0.036612i   -0.085301+0.21545i   -0.26517-0.35355i   -0.063354+0.12585i   0.036612-0.17678i   -0.10512-0.1207i   0+0.088388i   -0.07
-0.0625+0.23928i   -0.0036413-0.19508i   -0.088388+0i   -0.093238-0.15968i   0.22411-0.15089i   -0.071291-0.070083i   0.088388-0.17678i   0.26273-0.09203i   0.0625+0.11428i   0.09203-0.19508i   0.088388+0.17678i   0.2334+0
-0.1875+0.025888i   0.24807-0.056626i   -0.088388+0.088388i   -0.018306+0.30542i   -0.15089-0.36428i   0.084747-0.085955i   0.26517+0.088388i   0.10669-0.1079i   0.1875+0.15089i   0.017098+0.27001i   0.088388-0.088388i   -
-0.21339+0.088388i   0.15518+0.025034i   -0.088388+0.088388i   0.036258+0.19443i   0+0i   -0.27795-0.052131i   0.17678+0i   0.11342-0.18005i   0.036612+0.088388i   0.1832+0.18835i   -0.088388-0.088388i   0.10391+0.070736i
-0.025888+0.0625i   -0.0087908+0.20751i   0-0.17678i   0.094262+0.079598i   0.0625+0.15089i   0.02953+0.11184i   -0.17678+0i   0.075232+0.022247i   -0.15089-0.0625i   0.22218-0.11913i   0-0.17678i   0.4209+0.31057i   -0.06
-0.0625-0.36428i   0.070083+0.034679i   0-0.088388i   0.15968-0.10669i   -0.099112-0.15089i   0.19508+0.12136i   0-0.26517i   0.09203+0.018306i   0.0625+0.010723i   0.070083-0.19629i   0.17678+0.088388i   -0.071291-0.10669
-0.051777+0.036612i   0.12585-0.025034i   -0.35355-0.088388i   0.21545-0.046981i   -0.088388-0.051777i   0.078019+0.052131i   -0.088388+0i   -0.092684+0.25085i   -0.30178-0.21339i   -0.037466-0.18835i   0+0.088388i   -0.17
0.025888+0.15089i   -0.044194+0.28124i   -0.17678+0.088388i   -0.31057+0.044194i   0.0625-0.23928i   -0.044194+0.37822i   -0.17678-0.26517i   -0.11792+0.044194i   0.15089+0.025888i   -0.044194-0.14107i   0+0.26517i   -0.07
-0.11428+0.11428i   0.045402+0.051477i   -0.088388+0.17678i   -0.11184-0.066141i   0.15089+0.025888i   -0.24292+0.18678i   0.26517+0i   -0.20751+0.22218i   -0.23928+0.23928i   -0.18557-0.13987i   -0.088388-0.17678i   0.023
-0.21339+0.30178i   -0.11119-0.078019i   0.088388+0i   0.058205+0.063354i   0.051777+0.088388i   -0.256+0.037466i   0.35355-0.088388i   -0.15295+0.072869i   0.036612+0.051777i   0.1478-0.010369i   -0.088388+0i   0.030184-0
-0.20267-0.15089i   0.039045-0.09203i   0.088388+0.17678i   0.0036413-0.18163i   -0.0625+0.23928i   0.14501-0.15968i   0.088388+0.35355i   -0.15968+0.17434i   -0.32767-0.025888i   0.17434+0.0036413i   -0.088388+0i   -0.092
0.0625-0.025888i   -0.22218-0.073523i   0.17678-0.17678i   0.088088+0.22218i   -0.15089-0.23928i   0.066141+0.31178i   0+0i   -0.48855+0.02953i   -0.0625-0.15089i   0.0087908-0.26487i   0+0i   -0.10325-0.0087908i   -0.0258
-0.0625-0.025888i   0.22825-0.13379i   -0.088388-0.088388i   0.34718+0.13258i   -0.025888-0.0625i   0.36355+0.15453i   -0.088388+0.26517i   -0.09547+0.13258i   0.0625-0.15089i   0.036912+0.097179i   0.088388+0.088388i   0.
0.0625+0.3125i   -0.24413-0.11792i   0-0.088388i   0.079598-0.16919i   -0.025888+0.15089i   0.10154-0.31057i   0+0.088388i   0.022247+0.080806i   -0.0625-0.3125i   0.082515-0.022247i   0-0.088388i   0.31057-0.16919i   -0.1
0.21339-0.051777i   -0.085301-0.14052i   0+0.088388i   0.16762-0.17592i   0.051777+0.088388i   -0.10512+0.099966i   -0.088388-0.17678i   0.022801-0.041407i   -0.036612-0.30178i   0.17369-0.072869i   0.17678+0.088388i   -0.
0.051777+0.088388i   0.090451-0.010369i   -0.35355-0.088388i   0.087534-0.27674i   0.21339-0.125i   0.0042952-0.12585i   0.26517+0i   -0.046981-0.036258i   0.30178+0.088388i   -0.30384-0.078019i   0.17678-0.088388i   0.250
-0.27589-0.0625i   -0.0087908+0.044194i   0-0.17678i   0.2959-0.058859i   -0.010723+0.20267i   0.02953+0.044194i   0+0i   0.064933-0.097179i   0.099112+0.0625i   0.22218+0.044194i   0+0.17678i   -0.030738-0.15453i   0.3642
0.125-0.088388i   -0.056072-0.21424i   0.17678+0.088388i   -0.17369-0.072869i   0.036612+0.051777i   0.0020626-0.16641i   -0.088388+0i   0.10512+0.099966i   -0.125-0.088388i   -0.084093-0.050922i   0+0.088388i   0.085301-0
-0.11428+0.15089i   -0.20023+0.13379i   0.088388+0.088388i   -0.3855-0.080806i   0.025888+0.11428i   0.030738-0.15453i   -0.088388+0.088388i   0.03812+0.16919i   -0.23928+0.025888i   -0.064933-0.097179i   -0.088388-0.08838
-0.17678-0.125i   0.092684-0.087534i   0+0.17678i   0.2103-0.052131i   -0.088388+0.33839i   -0.21545+0.11463i   0.088388+0.088388i   -0.3185+0.025034i   -0.17678+0.125i   0.1207-0.25085i   -0.17678+0.17678i   -0.04869+0.01
-0.099112-0.099112i   -0.0058738-0.15453i   0.26517+0i   -0.13379-0.25758i   0.23928+0.0625i   0.14846-0.097179i   0.088388+0i   0.058859-0.0075825i   0.27589+0.27589i   -0.33251-0.058859i   -0.088388+0i   0.097179-0.25758
0.0625+0.15089i   -0.15059-0.09203i   0+0.17678i   0.15968-0.085955i   -0.15089+0.1875i   0.089113-0.15968i   0+0.17678i   0.09203-0.056626i   -0.0625+0.025888i   0.040753+0.0036413i   0.17678+0i   -0.071291+0.049344i   -0
0.23928+0.27589i   -0.18557+0.11184i   0+0.17678i   -0.15016-0.02953i   -0.15089+0.23928i   -0.14725-0.11913i   0+0i   -0.26487+0.0087908i   0.11428-0.099112i   0.045402-0.023455i   -0.17678+0i   0.31178+0.066141i   -0.025
-0.036612+0.088388i   0.11119-0.064563i   0-0.17678i   0.10512+0.18835i   0.17678-0.17678i   0.256+0.14659i   -0.088388+0.26517i   0.085301-0.052131i   0.21339+0.088388i   -0.1478+0.32973i   0+0i   -0.1935+0.025034i   0.17
0.0625+0.15089i   -0.058859-0.044194i   0.088388+0.088388i   0.044194-0.28681i   0.15089+0.23928i   0.13379-0.044194i   0.26517-0.088388i   0.044194+0.57815i   -0.0625+0.025888i   -0.15453-0.044194i   0.088388+0.088388i
0.0625+0.15089i   -0.16575+0.15968i   0.088388-0.088388i   -0.21703+0.18163i   0.15089+0.23928i   -0.42605-0.0036413i   -0.26517-0.088388i   0.19629-0.17434i   -0.0625+0.025888i   0.025588-0.071291i   0.088388+0.26517i   -
0.11428+0.15089i   0.20751-0.097179i   0+0.17678i   -0.097179+0.11184i   0.099112-0.1875i   0.11184-0.058859i   0.35355+0.17678i   -0.15453+0.20751i   0.23928+0.025888i   -0.11913+0.13379i   0+0.17678i   0.13379-0.023455i
0.125-0.088388i   -0.15174+0.12585i   -0.088388+0.17678i   -0.010369-0.031392i   -0.088388+0.17678i   -0.22891+0.078019i   -0.17678-0.088388i   0.037466-0.049898i   -0.125-0.088388i   0.011578-0.037466i   0.26517+0i   -0.0
0.0625+0.025888i   -0.15968+0.10669i   0+0i   -0.18163-0.12136i   0.32767-0.13572i   0.0036413-0.018306i   0.17678+0.17678i   0.17434-0.034679i   -0.0625+0.15089i   0.071291+0.10669i   0+0i   0.14501-0.21703i   0.20267+0.4
0.025888+0.0625i   -0.1524-0.049344i   0+0i   -0.10669-0.30542i   0.0625+0.025888i   0.14622+0.056626i   -0.17678-0.17678i   0.018306+0.1079i   0.15089-0.0625i   -0.11277+0.085955i   -0.17678+0.17678i   -0.10669-0.20975i
0-0.088388i   0.30384-0.085301i   0.17678+0.26517i   -0.12585-0.025034i   0.26517+0.30178i   -0.032316-0.10512i   -0.088388+0.35355i   -0.010369-0.015519i   0-0.088388i   -0.090451+0.17369i   0-0.088388i   0.037466-0.18835
0.0625+0.0625i   -0.15453-0.18678i   0.26517+0.17678i   -0.13258+0.066141i   0.15089+0.15089i   -0.097179+0.13987i   -0.26517+0.35355i   -0.13258-0.22218i   -0.0625-0.0625i   -0.058859+0.27516i   -0.088388-0.17678i   -0.13
-0.025888+0.1875i   0.066141+0.25758i   -0.088388+0.088388i   -0.14846+0.0015083i   -0.0625+0.15089i   0.0087908+0.0075825i   -0.088388-0.088388i   0.0058738+0.3855i   -0.15089-0.1875i   -0.02953+0.25758i   -0.26517-0.0883
0-0.30178i   0.15174-0.28189i   0.17678+0i   0.20302+0.090451i   -0.088388-0.16161i   0.22891-0.0030869i   0.088388+0.088388i   0.00085436+0.032316i   0-0.051777i   -0.011578+0.016727i   0+0i   0.13537-0.30384i   -0.088388
0.15089-0.0625i   -0.0036413-0.17434i   -0.088388-0.088388i   0.32008-0.24807i   0.3125+0.025888i   -0.071291+0.18163i   0.088388+0.088388i   -0.054917-0.18042i   0.025888+0.0625i   0.09203-0.039045i   -0.088388-0.088388i
-0.17678+0.088388i   -0.010369-0.16126i   -0.088388-0.17678i   0.18835+0.075786i   -0.088388+0.125i   -0.12585-0.15174i   0+0.088388i   -0.052131-0.020668i   -0.17678+0.088388i   -0.078019-0.22891i   -0.26517+0.17678i   0.
-0.42678-0.036612i   -0.025034-0.3185i   0.17678+0.088388i   0.17369-0.052131i   -0.088388+0.17678i   0.052131+0.2103i   -0.088388+0i   -0.10512+0.025034i   0.073223+0.21339i   -0.18835-0.019884i   0+0.088388i   -0.085301+
-0.088388-0.17678i   0.18835-0.037466i   -0.088388-0.35355i   0.11463+0.2831i   0-0.21339i   0.015519+0.010369i   0+0.088388i   -0.087534+0.070637i   -0.088388-0.17678i   0.025034+0.12585i   -0.26517+0i   0.046981-0.24649i
-0.088388-0.21339i   -0.010369+0.052131i   -0.088388-0.088388i   -0.15174+0.22153i   -0.125-0.30178i   -0.12585-0.18835i   0.17678+0.17678i   -0.16126-0.2206i   -0.088388+0.036612i   -0.078019-0.015519i   -0.088388-0.26517
0.17678-0.21339i   0.085301+0.23619i   0+0.088388i   -0.0042952-0.087534i   -0.036612-0.125i   0.10512+0.131i   0.088388-0.17678i   -0.090451+0.046981i   0.17678+0.036612i   -0.17369-0.022801i   0-0.088388i   -0.032316-0.2
-0.15089-0.1875i   -0.058859+0.48855i   -0.26517-0.26517i   -0.11913-0.13379i   0.0625+0.025888i   0.13379-0.088088i   -0.088388+0.088388i   0.11184+0.058859i   -0.025888+0.1875i   -0.15453+0.026613i   -0.088388-0.088388i
-0.23928-0.0625i   0.044194+0.045402i   0-0.26517i   0.066141+0.11913i   -0.15089+0.099112i   0.044194-0.24292i   0+0.26517i   -0.22218-0.11184i   -0.11428+0.0625i   0.044194-0.18557i   -0.35355+0.088388i   -0.02953-0.2075
```

Figure 121: Expected Output (float)

4- Then we write a testbench which generates the output data when valid is high in a text file, which for sure in fixed format as we deal with a digital system, then this file is passed to a MATLAB script to convert it to a floating format.

5- Eventually, we pass the file which generated from the testbench (generated output file) and the expected file which is generated from MATLAB (expected output file) and compare them to see the final results. (We will present the results at the end of this section).

Waveforms

Figure 122 below shows that "EN" is activated then input data are read by the IFFT, at the same time the 2nd buffer is extracting the data (as discussed before).

The "valid" signal is activated with the beginning of the OFDM symbol.

The highlighted samples are the cyclic prefix which is 9 cycles (1 cycle for x(14) and 8 cycles for x(15)) then the data are extracted in arranged sequence from x(0) to x(15).

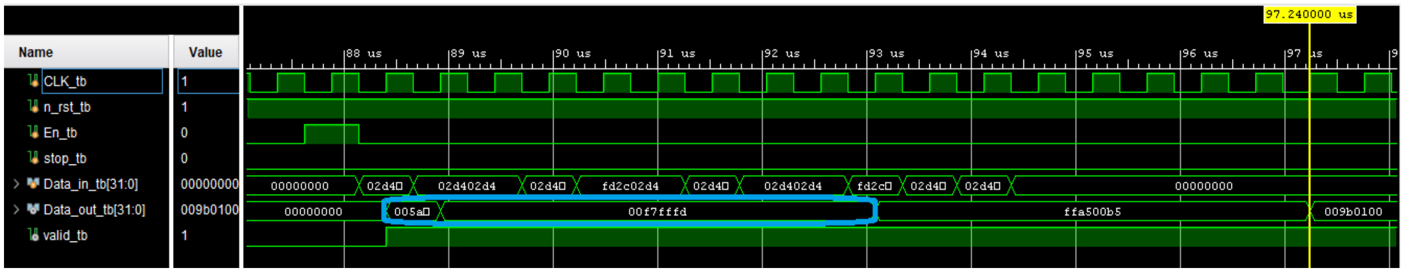


Figure 122: IFFT output to clarify the CP

Figure 123 below shows that the cyclic prefix duration is 4.69µs as stated in the standard.

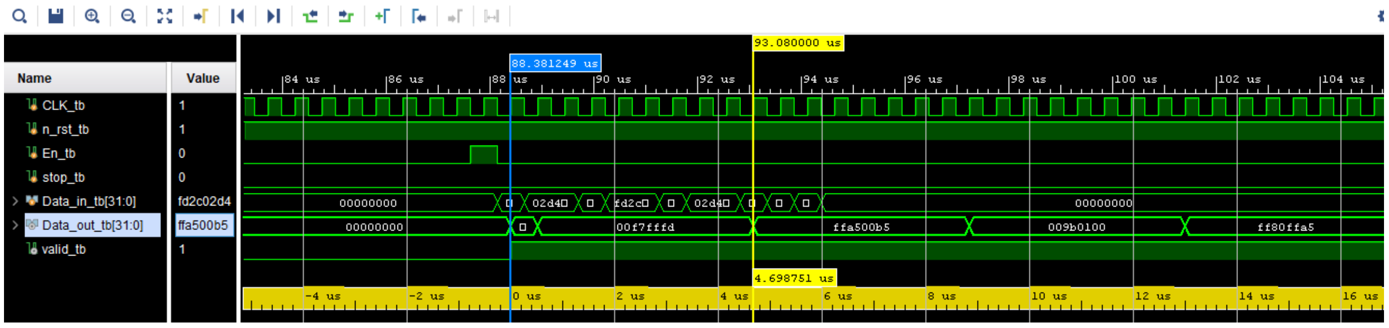


Figure 123: CP duration

Figure 124 shows the OFDM symbol duration is 71.3 which is expected as we operate the system with frequency 1.92MHz and extract the output in 137 cycles.

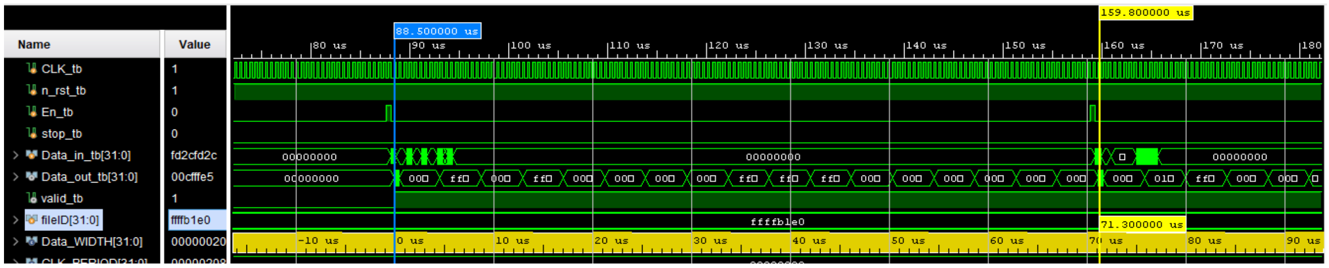


Figure 124: IFFT output to clarify the OFDM duration

Figure 125 shows the “Data_out” is extracted without any bubbles (in continues manner), first 9 cycles are the cyclic prefix while the last 128 cycles are the useful data, each is extracted in 8 cycles.

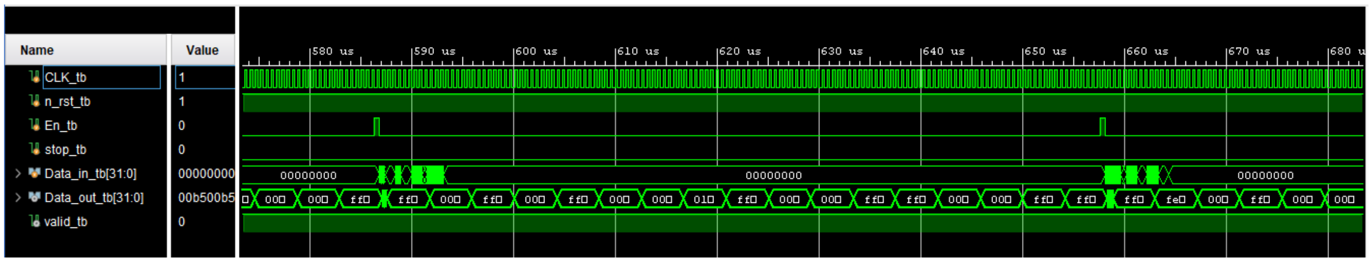


Figure 125: IFFT output to clarify the continuity of the output

Figure 126 shows IFFT operation for 20 OFDM symbols.

Also a “stop” signal is activated at the end of the last OFDM symbol, and it’s shown that when the OFDM symbol completely extracted the valid signal is deactivated.

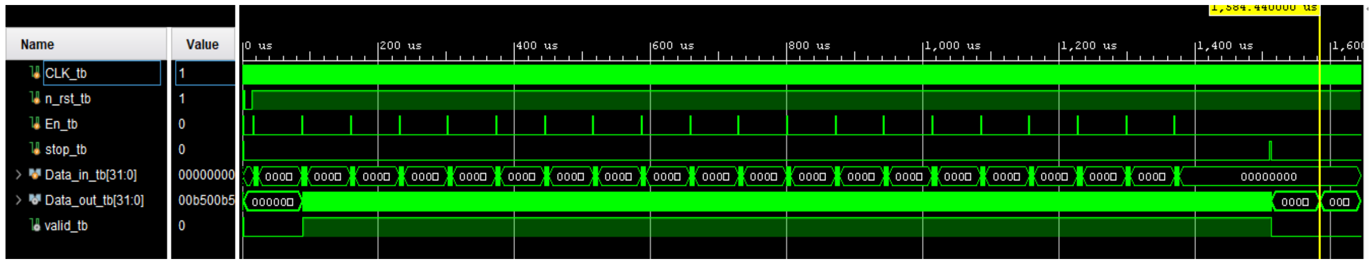


Figure 126: IFFT output for 20 OFDM symbols

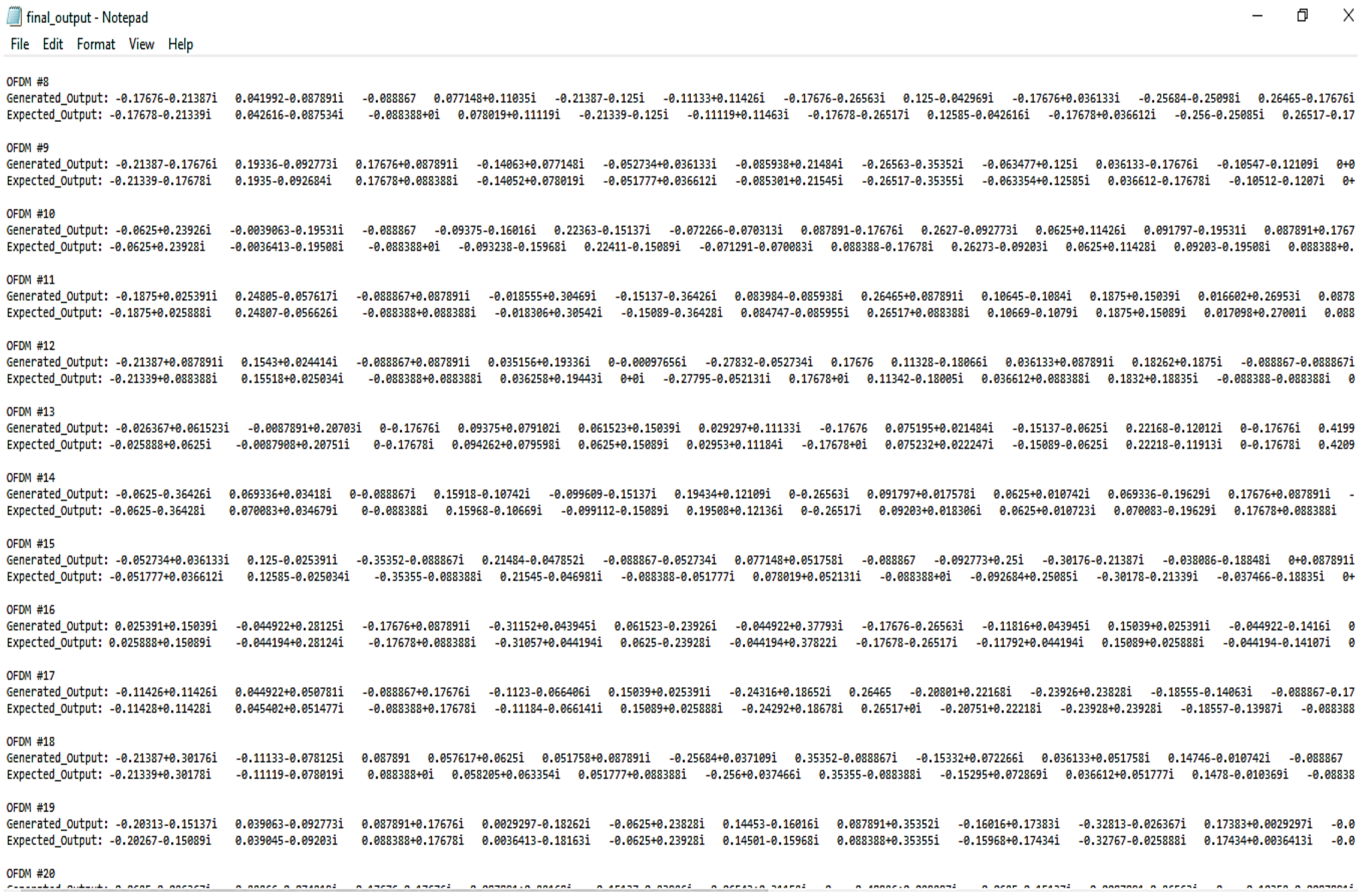
Results

And finally the comparison file, as we see the expected output is nearly equals the generated output, they are not exactly equal each other's due to two reasons:

1- Quantization error as we only represent the data in 16 bits (for either real part or imaginary part).

2- Truncation of multiplication output which is expected to be 32 bits as we multiply 16 bits with each other, but we have a limited RAMs with 16 bits width, which force us to truncate the data to be 16 bits.

If we increase the length of the represented data bits we will consume more power as larger multiplier and adders are needed and also we will need more area as the RAMs will be wider, but our goal from the beginning is to get the minimum power and area as possible.



```
OFDM #8
Generated_Output: -0.17676-0.21387i   0.041992-0.087891i   -0.088867   0.077148+0.11035i   -0.21387-0.125i   -0.11133+0.11426i   -0.17676-0.26563i   0.125-0.042969i   -0.17676+0.036133i   -0.25684-0.25098i   0.26465-0.17676i
Expected_Output: -0.17678-0.21339i   0.042616-0.087534i   -0.088388+0i   0.078019+0.11119i   -0.21339-0.125i   -0.11119+0.11463i   -0.17678-0.26517i   0.12585-0.042616i   -0.17678+0.036612i   -0.256-0.25085i   0.26517-0.17

OFDM #9
Generated_Output: -0.21387-0.17676i   0.19336-0.092773i   0.17676+0.087891i   -0.14063+0.077148i   -0.052734+0.036133i   -0.085938+0.21484i   -0.26563-0.35352i   -0.063477+0.125i   0.036133-0.17676i   -0.10547-0.12109i   0+0
Expected_Output: -0.21339-0.17678i   0.1935-0.092684i   0.17678+0.088388i   -0.14052+0.078019i   -0.051777+0.036612i   -0.085301+0.21545i   -0.26517-0.35355i   -0.063354+0.12585i   0.036612-0.17678i   -0.10512-0.1207i   0+

OFDM #10
Generated_Output: -0.0625+0.23926i   -0.0039063-0.19531i   -0.088867   -0.09375-0.16016i   0.22363-0.15137i   -0.072266-0.070313i   0.087891-0.17676i   0.2627-0.092773i   0.0625+0.11426i   0.091797-0.19531i   0.087891+0.1767
Expected_Output: -0.0625+0.23928i   -0.0036413-0.19508i   -0.088388+0i   -0.093238-0.15968i   0.22411-0.15089i   -0.071291-0.070083i   0.088388-0.17678i   0.26273-0.09203i   0.0625+0.11428i   0.09203-0.19508i   0.088388+0.

OFDM #11
Generated_Output: -0.1875+0.025391i   0.24805-0.057617i   -0.088867+0.087891i   -0.018555+0.30469i   -0.15137-0.36426i   0.083984-0.085938i   0.26465+0.087891i   0.10645-0.1084i   0.1875+0.15039i   0.016602+0.26953i   0.0878
Expected_Output: -0.1875+0.025888i   0.24807-0.056626i   -0.088388+0.088388i   -0.018306+0.30542i   -0.15089-0.36428i   0.084747-0.085955i   0.26517+0.088388i   0.10669-0.1079i   0.1875+0.15089i   0.017098+0.27001i   0.088

OFDM #12
Generated_Output: -0.21387+0.087891i   0.1543+0.024414i   -0.088867+0.087891i   0.035156+0.19336i   0-0.00097656i   -0.27832-0.052734i   0.17676   0.11328-0.18066i   0.036133+0.087891i   0.18262+0.1875i   -0.088867-0.088867i
Expected_Output: -0.21339+0.088388i   0.15518+0.025034i   -0.088388+0.088388i   0.036258+0.19443i   0+0i   -0.27795-0.052131i   0.17678+0i   0.11342-0.18005i   0.036612+0.088388i   0.1832+0.18835i   -0.088388-0.088388i   0

OFDM #13
Generated_Output: -0.026367+0.061523i   -0.0087891+0.20703i   0-0.17676i   0.09375+0.079102i   0.061523+0.15039i   0.029297+0.11133i   -0.17676   0.075195+0.021484i   -0.15137-0.0625i   0.22168-0.12012i   0-0.17676i   0.4199
Expected_Output: -0.025888+0.0625i   -0.0087908+0.20751i   0-0.17678i   0.094262+0.079598i   0.0625+0.15089i   0.02953+0.11184i   -0.17678+0i   0.075232+0.022247i   -0.15089-0.0625i   0.22218-0.11913i   0-0.17678i   0.4209

OFDM #14
Generated_Output: -0.0625-0.36426i   0.069336+0.03418i   0-0.088867i   0.15918-0.10742i   -0.099609-0.15137i   0.19434+0.12109i   0-0.26563i   0.091797+0.017578i   0.0625+0.010742i   0.069336-0.19629i   0.17676+0.087891i   -
Expected_Output: -0.0625-0.36428i   0.070083+0.034679i   0-0.088388i   0.15968-0.10669i   -0.099112-0.15089i   0.19508+0.12136i   0-0.26517i   0.09203+0.018306i   0.0625+0.010723i   0.070083-0.19629i   0.17678+0.088388i

OFDM #15
Generated_Output: -0.052734+0.036133i   0.125-0.025391i   -0.35352-0.088867i   0.21484-0.047852i   -0.088867-0.052734i   0.077148+0.051758i   -0.088867   -0.092773+0.25i   -0.30176-0.21387i   -0.038086-0.18848i   0+0.087891i
Expected_Output: -0.051777+0.036612i   0.12585-0.025034i   -0.35355-0.088388i   0.21545-0.046981i   -0.088388-0.051777i   0.078019+0.052131i   -0.088388+0i   -0.092684+0.25085i   -0.30178-0.21339i   -0.037466-0.18835i   0+

OFDM #16
Generated_Output: 0.025391+0.15039i   -0.044922+0.28125i   -0.17676+0.087891i   -0.31152+0.043945i   0.061523-0.23926i   -0.044922+0.37793i   -0.17676-0.26563i   -0.11816+0.043945i   0.15039+0.025391i   -0.044922-0.1416i   0
Expected_Output: 0.025888+0.15089i   -0.044194+0.28124i   -0.17678+0.088388i   -0.31057+0.044194i   0.0625-0.23928i   -0.044194+0.37822i   -0.17678-0.26517i   -0.11792+0.044194i   0.15089+0.025888i   -0.044194-0.14107i   0

OFDM #17
Generated_Output: -0.11426+0.11426i   0.044922+0.050781i   -0.088867+0.17676i   -0.1123-0.066406i   0.15039+0.025391i   -0.24316+0.18652i   0.26465   -0.20801+0.22168i   -0.23926+0.23828i   -0.18555-0.14063i   -0.088867-0.17
Expected_Output: -0.11428+0.11428i   0.045402+0.051477i   -0.088388+0.17678i   -0.11184-0.066141i   0.15089+0.025888i   -0.24292+0.18678i   0.26517+0i   -0.20751+0.22218i   -0.23928+0.23928i   -0.18557-0.13987i   -0.088388

OFDM #18
Generated_Output: -0.21387+0.30176i   -0.11133-0.078125i   0.087891   0.057617+0.0625i   0.051758+0.087891i   -0.25684+0.037109i   0.35352-0.088867i   -0.15332+0.072266i   0.036133+0.051758i   0.14746-0.010742i   -0.088867
Expected_Output: -0.21339+0.30178i   -0.11119-0.078019i   0.088388+0i   0.058205+0.063354i   0.051777+0.088388i   -0.256+0.037466i   0.35355-0.088388i   -0.15295+0.072869i   0.036612+0.051777i   0.1478-0.010369i   -0.08838

OFDM #19
Generated_Output: -0.20313-0.15137i   0.039063-0.092773i   0.087891+0.17676i   0.0029297-0.18262i   -0.0625+0.23828i   0.14453-0.16016i   0.087891+0.35352i   -0.16016+0.17383i   -0.32813-0.026367i   0.17383+0.0029297i   -0.0
Expected_Output: -0.20267-0.15089i   0.039045-0.09203i   0.088388+0.17678i   0.0036413-0.18163i   -0.0625+0.23928i   0.14501-0.15968i   0.088388+0.35355i   -0.15968+0.17434i   -0.32767-0.025888i   0.17434+0.0036413i   -0.0

OFDM #20
```

Figure 127: Final Comparison File

4.7.8. Fraction bits number determination

Real numbers can be represented in digital systems by two representation:

1) **Floating-point representation:** The floating-point representation breaks the number into two parts, the left-hand side is a signed, fixed-point number known as a mantissa and the right-hand side of the number is known as the exponent.

$$Mantissa * 2^{exponent}$$

It's a complex representation which need a complex units to deal with to get very accurate results, which will lead to consumption of more power and increasing in area, while our target in NB-IOT is to achieve minimum power and area with a good performance.

2) **Fixed-point representation:** Fixed-point numbers having decimal points at the right end of the number are treated as integers because the fixed-point numbers having decimal points at the left end of the number are treated as fractions.

We can represent these numbers using:

- Signed representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 1's complement representation: range from $-(2^{(k-1)}-1)$ to $(2^{(k-1)}-1)$, for k bits.
- 2's complementation representation: range from $-(2^{(k-1)})$ to $(2^{(k-1)}-1)$, for k bits.

2's complementation representation is preferred in computer system because of unambiguous property and easier for arithmetic operations.

In this thesis we work with 2's complementation representation (fixed-point).

To know how number will represent either the integer bits or the fraction bits, we should now first the range of numbers which we deal with, which will be constrained by the IFFT output.

We generate random QPSK symbols and apply IFFT operation using MATLAB to determine the ranges of IFFT output, we get that the number ranges is between -0.65 and +0.65, by putting some margin we will assume that the range is between -0.7 and +0.7.

Since, IFFT equation is $X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{j2\pi kn/N}$, the pre-mentioned range is determined after divide the summation by N, so before this division the range was -0.7*N and +0.7*N which is -11.2 and +11.2, hence this range will appear in the intermediate values in the butterfly unit, so at least we must allocate 5-bit for integer to fit this range.

By taking the range ±0.7, we run a MATLAB script to calculate the error due to quantization, to determine the fraction bits according to the error.

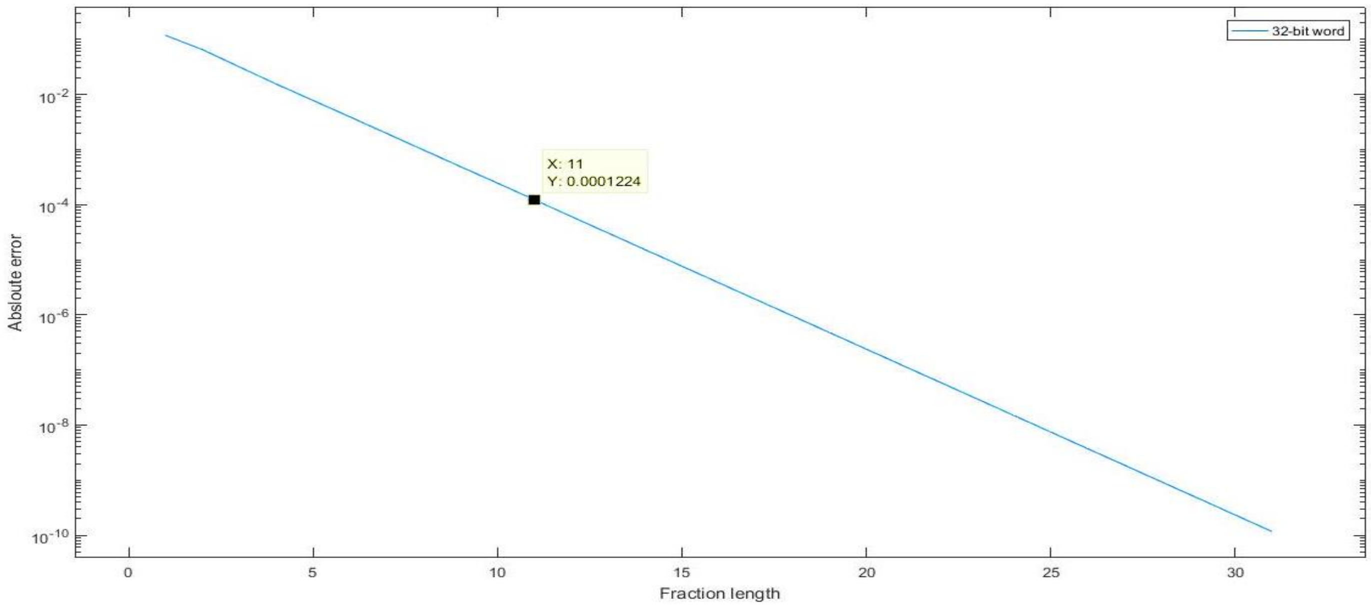


Figure 128: Quantization error vs Fraction length

To calculate this error, we generate range of numbers between -0.7 and 0.7, then re-represent these numbers using a 32-bit word with different fraction length as shown in Figure 128.

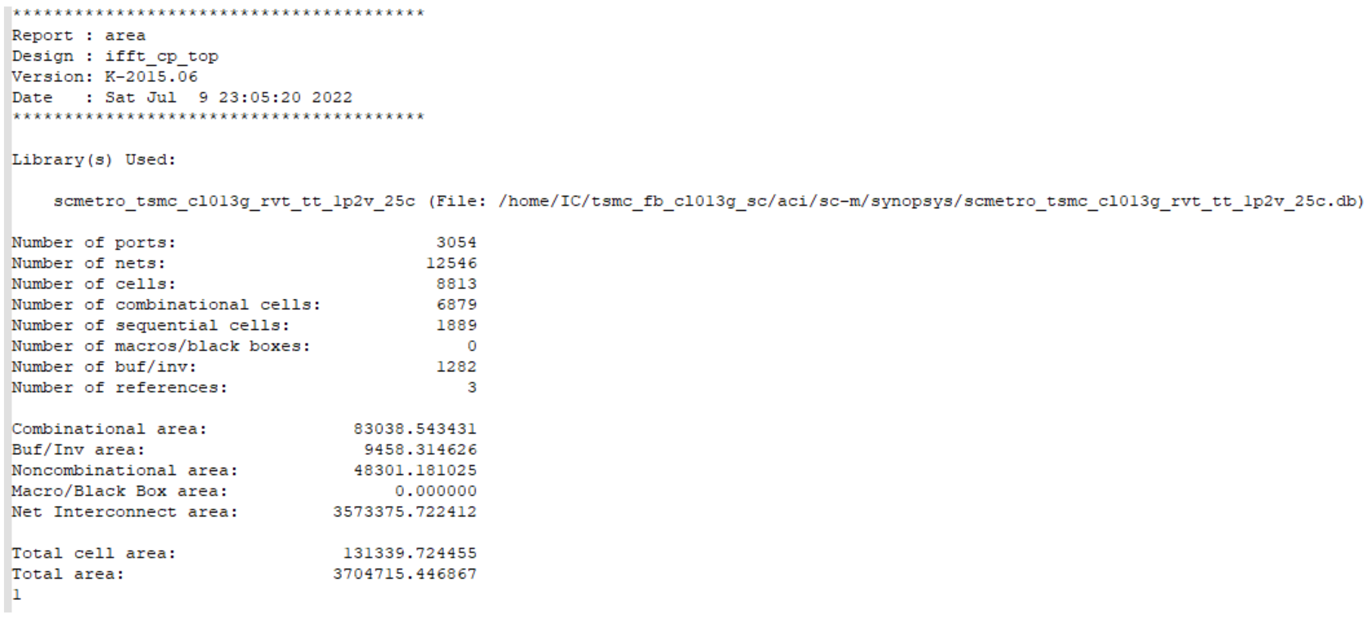
Then we get the error due to the quantization and get the mean of error at each fraction length, then plot the above curve.

We noted that the error is decreased exponentially with the fraction bits, and we will get more accurate result as we increase the number of bits to represent the fraction part, but on the other hand we will consume more power and more area, while our target in NB-IOT is low power and area, so we assumed that 11 bits is sufficient to represent the fraction part, which corresponds to a mean error of 0.0001224, so the total bits to represent our real number are 16 bits in fixed-point representation

4.7.9. Synthesis Results

Area report in Figure 129, Report separates the area of combinational and sequential logics, also the interconnect area (wiring area) which has the biggest share.

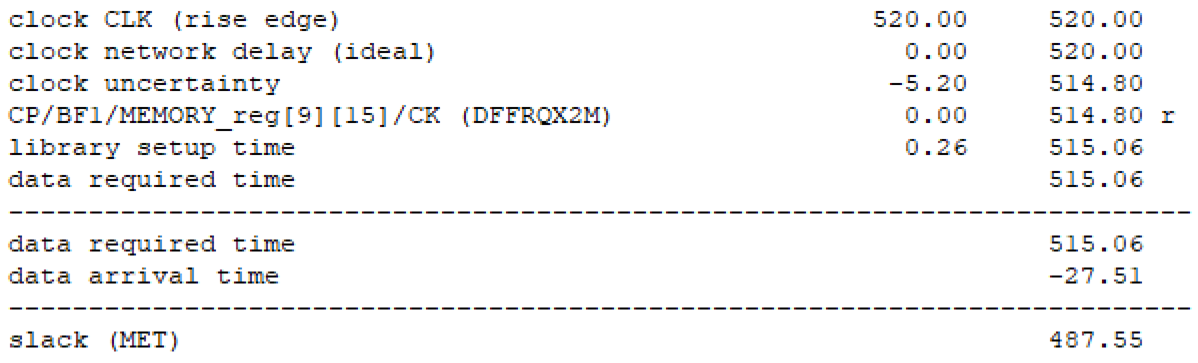
Total cell area is also reported which include sequential and combinational logics area.



```
****************************************
Report : area
Design : ifft_cp_top
Version: K-2015.06
Date   : Sat Jul  9 23:05:20 2022
****************************************

Library(s) Used:

    scmetro_tsmc_cl013g_rvt_tt_1p2v_25c (File: /home/IC/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_tt_1p2v_25c.db)

Number of ports:                         3054
Number of nets:                         12546
Number of cells:                         8813
Number of combinational cells:           6879
Number of sequential cells:              1889
Number of macros/black boxes:               0
Number of buf/inv:                       1282
Number of references:                       3

Combinational area:              83038.543431
Buf/Inv area:                     9458.314626
Noncombinational area:           48301.181025
Macro/Black Box area:                0.000000
Net Interconnect area:         3573375.722412

Total cell area:                131339.724455
Total area:                    3704715.446867
1
```

Figure 129: Area report of IFFT & CP

As shown in Figure 130, the timing of the design is met with too large slack, which indicates that the design may operate at a higher frequency, but we are constrained to this frequency to achieve standard specification.



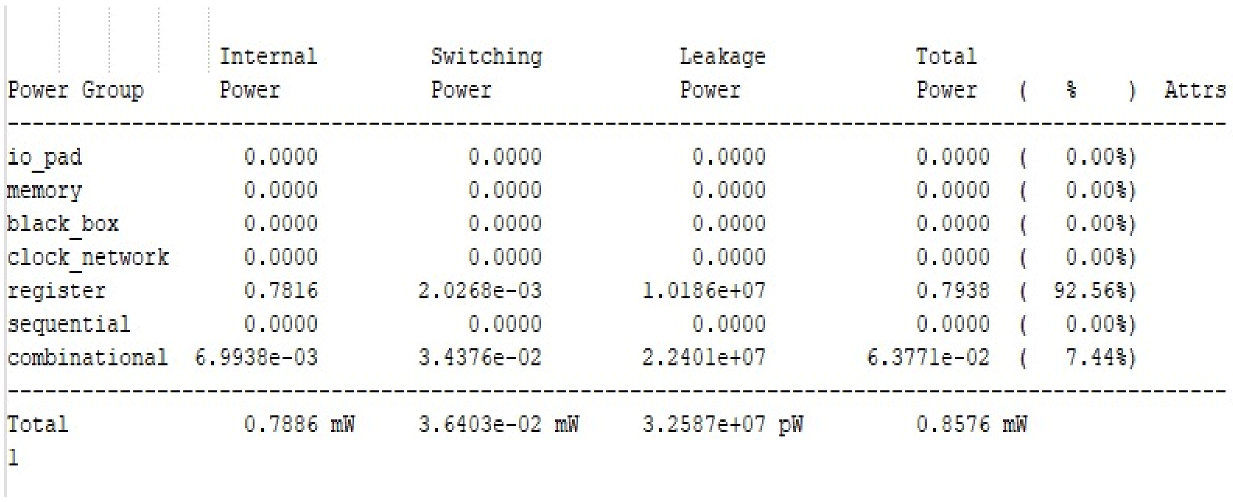
```
clock CLK (rise edge)                              520.00     520.00
clock network delay (ideal)                          0.00     520.00
clock uncertainty                                   -5.20     514.80
CP/BF1/MEMORY_reg[9][15]/CK (DFFRQX2M)               0.00     514.80 r
library setup time                                   0.26     515.06
data required time                                            515.06
---------------------------------------------------------------------------
data required time                                            515.06
data arrival time                                             -27.51
---------------------------------------------------------------------------
slack (MET)                                                   487.55
```

Figure 130: Timing_max (setup time) report of IFFT & CP

Switching power is caused by the charging and discharging of the load capacitance in the circuit when the circuit output switches from high to low and from low to high during the normal operation of CMOS gates, where $p_{switching} = \frac{1}{2} \propto C_L VDD^2 f$

Internal power is power dissipated within the boundary of a cell. During switching, a circuit dissipates internal power by the charging or discharging of any existing capacitances internal to the cell. Internal power includes power dissipated by a momentary short circuit between the P and N transistors of a gate, called short-circuit power.

Leakage power is considered a static power which is defined as the power consumption due to constant current from V_{DD} to GND in the absence of switching activity.



```
                Internal         Switching           Leakage            Total
Power Group     Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad            0.0000           0.0000              0.0000             0.0000  (   0.00%)
memory            0.0000           0.0000              0.0000             0.0000  (   0.00%)
black_box         0.0000           0.0000              0.0000             0.0000  (   0.00%)
clock_network     0.0000           0.0000              0.0000             0.0000  (   0.00%)
register          0.7816        2.0268e-03          1.0186e+07            0.7938  (  92.56%)
sequential        0.0000           0.0000              0.0000             0.0000  (   0.00%)
combinational  6.9938e-03       3.4376e-02          2.2401e+07        6.3771e-02  (   7.44%)
--------------------------------------------------------------------------------------------------
Total             0.7886 mW     3.6403e-02 mW       3.2587e+07 pW         0.8576 mW
1
```

Figure 131: Power report of IFFT & CP

Chapter Five

5. System Integration

In this chapter we will discuss briefly the NB-IoT integrated transmitter results.

We integrated the system in a hierarchical approach, where each 2 successive blocks are combined in a top level and so on till we reach the state that we had two halves of the chain, upper layer register file, and the controller.

The final step was to integrate the system as shown in the figure below, which took a lot of time to achieve the standard specs.

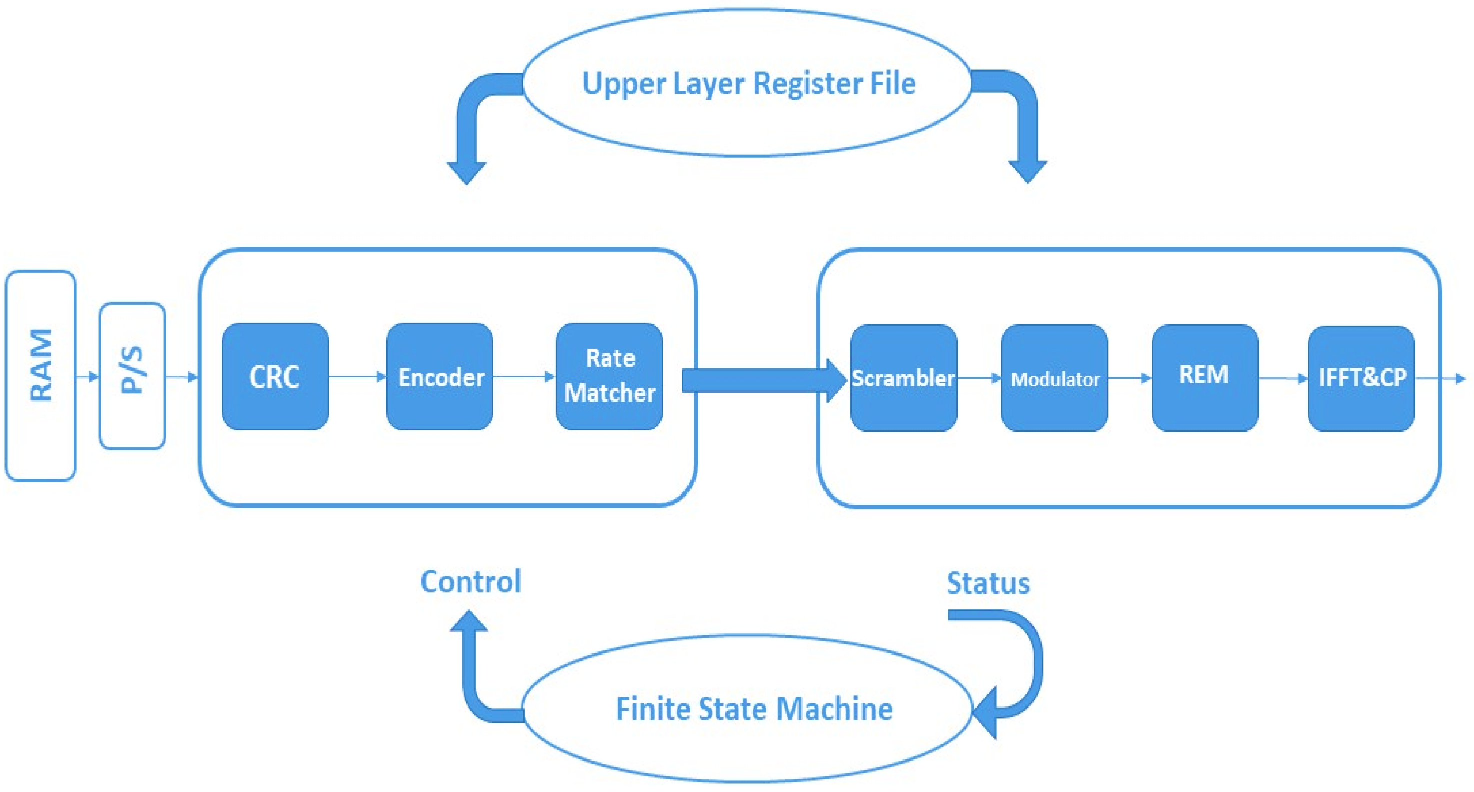


Figure 132: Illustrative Diagram of our integration approach

5.1. NB-IoT LTE Transmitter TOP LEVEL

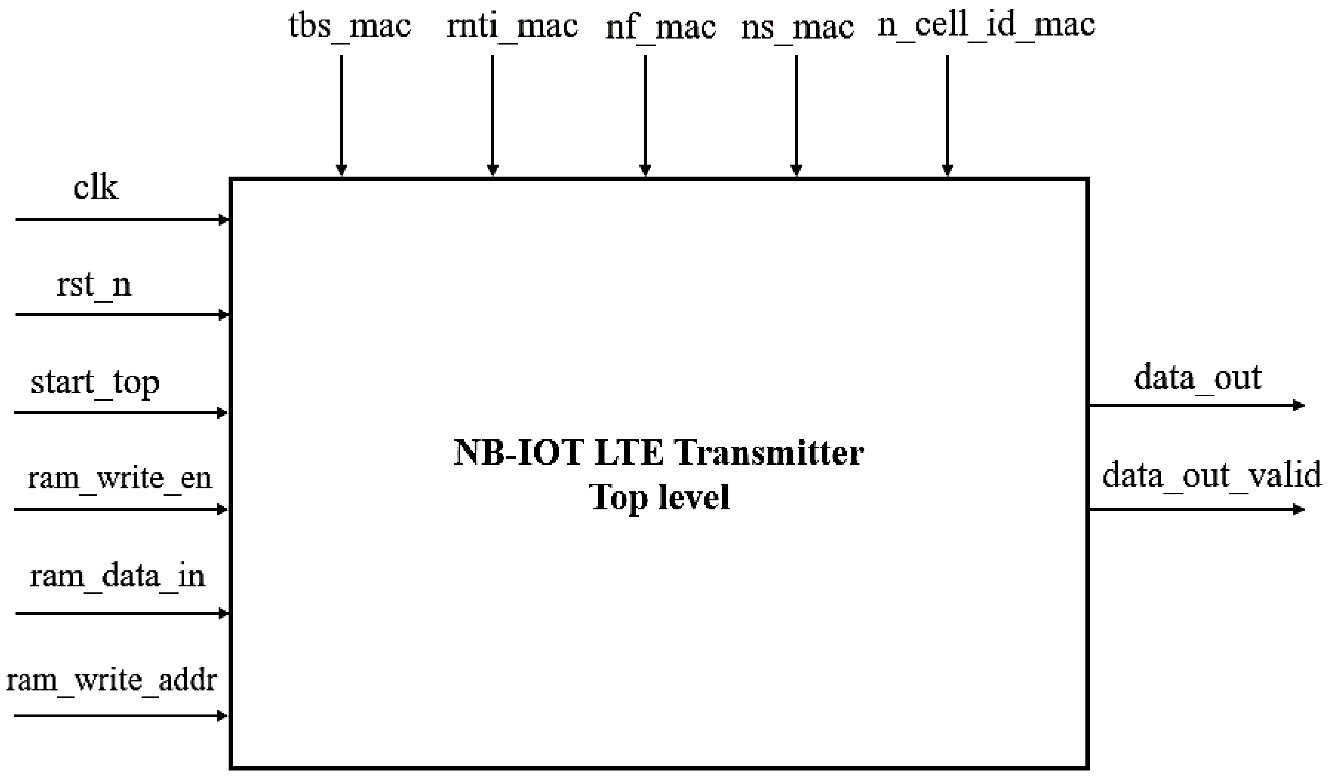


Figure 133: NB-IoT Tx Top Level

5.2. Top Level Interface Table

Port	Type	Width	Description
clk	Input	1 bit	System clock
rst_n	Input	1 bit	Active Low Asynchronous Chip reset
start_top	Input	1 bit	Interface Signal that starts or enables the chip
ram_write_en	Input	1 bit	Write enable to write the upper layer data into the RAM
ram_data_in	Input	32 bits	Data written into the RAM from the upper layer
ram_write_addr	Input	7 bits	Address of data RAM from the upper layer
tbs_mac	Input	12 bits	Transport Block Size, its maximum value is 2536
rnti_mac	Input	16 bits	Radio Network Temporary ID given from upper layer.
nf_mac	Input	10 bits	First system frame number given from upper layer.
ns_mac	Input	5 bits	First slot number given from upper layer.
n_cell_id_mac	Input	9 bits	Physical layer cell identity given from upper layer.
data_out	Output	32 bits	Output I & Q symbols from the Transmitter chain
data_out_valid	Output	1 bit	Valid output, asserted high with the output symbols

Table 17: Top Level Interface Table

5.3. Detailed NB-IoT LTE Transmitter Chain

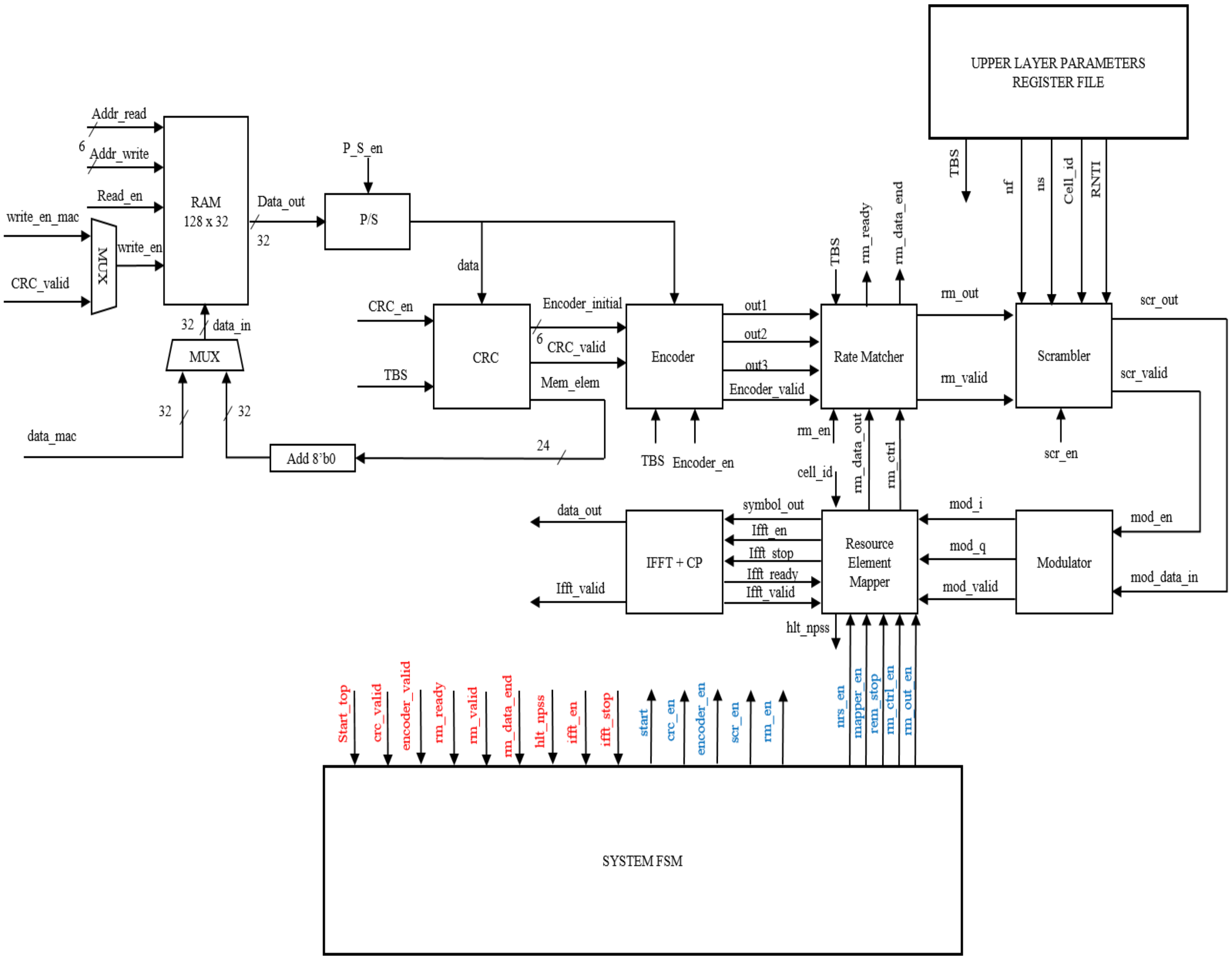


Figure 134: Detailed Architecture

5.4. System's Finite State Machine

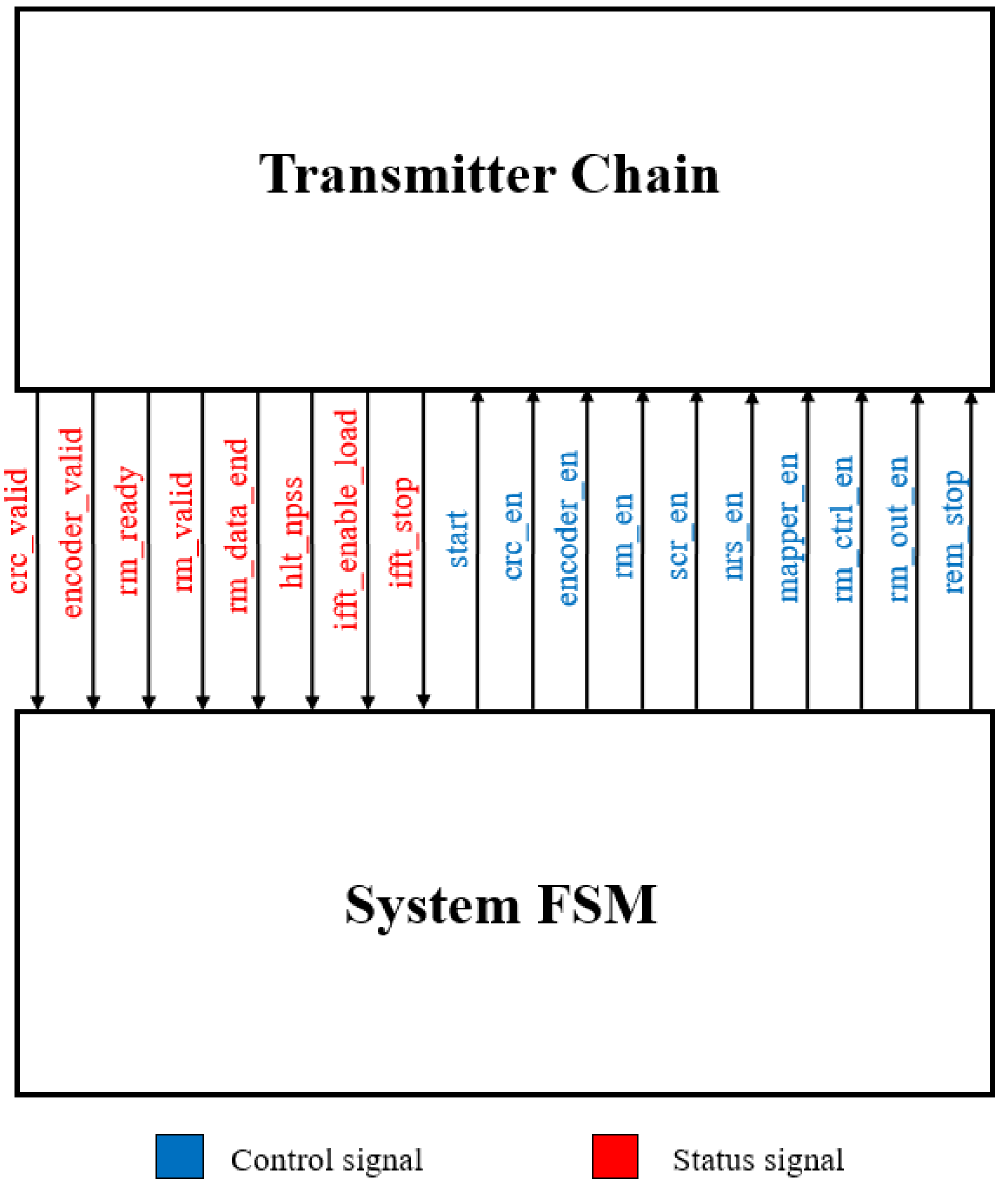
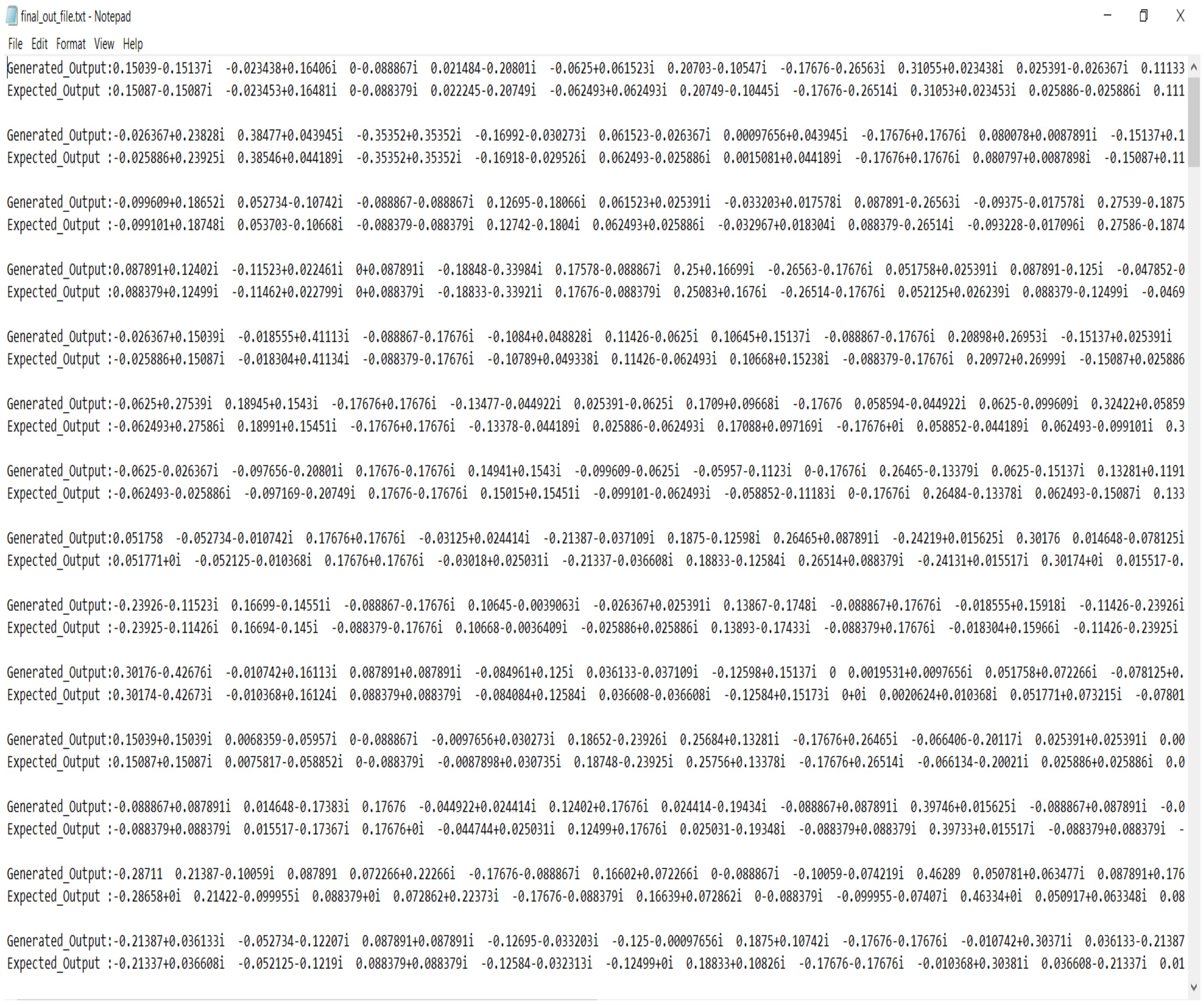


Figure 135: Datapath & Controller Diagram

5.5. Simulation Results

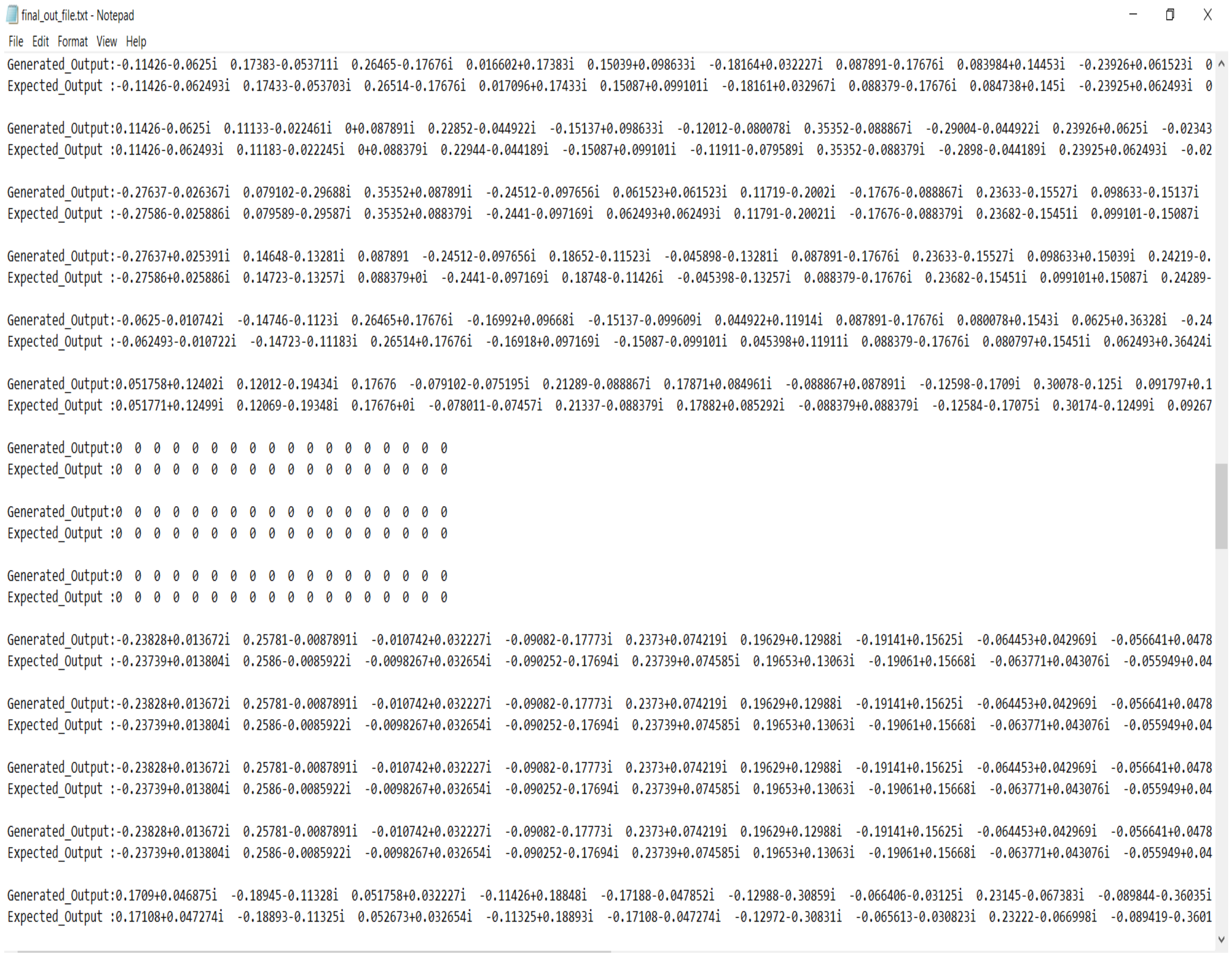
First, we integrated the MATLAB functions of the whole chain by generating an input stream inside the RAM which then interfaces with the P/S to propagate through the whole chain.
The generated output from the RTL is verified using the integrated MATLAB expected output they are almost identical.



```
Generated_Output:0.15039-0.15137i  -0.023438+0.16406i  0-0.088867i  0.021484-0.20801i  -0.0625+0.061523i  0.20703-0.10547i  -0.17676-0.26563i  0.31055+0.023438i  0.025391-0.026367i  0.11133
Expected_Output :0.15087-0.15087i  -0.023453+0.16481i  0-0.088379i  0.022245-0.20749i  -0.062493+0.062493i  0.20749-0.10445i  -0.17676-0.26514i  0.31053+0.023453i  0.025886-0.025886i  0.111

Generated_Output:-0.026367+0.23828i  0.38477+0.043945i  -0.35352+0.35352i  -0.16992-0.030273i  0.061523-0.026367i  0.00097656+0.043945i  -0.17676+0.17676i  0.080078+0.0087891i  -0.15137+0.1
Expected_Output :-0.025886+0.23925i  0.38546+0.044189i  -0.35352+0.35352i  -0.16918-0.029526i  0.062493-0.025886i  0.0015081+0.044189i  -0.17676+0.17676i  0.080797+0.0087898i  -0.15087+0.11

Generated_Output:-0.099609+0.18652i  0.052734-0.10742i  -0.088867-0.088867i  0.12695-0.18066i  0.061523+0.025391i  -0.033203+0.017578i  0.087891-0.26563i  -0.09375-0.017578i  0.27539-0.1875
Expected_Output :-0.099101+0.18748i  0.053703-0.10668i  -0.088379-0.088379i  0.12742-0.1804i  0.062493+0.025886i  -0.032967+0.018304i  0.088379-0.26514i  -0.093228-0.017096i  0.27586-0.1874

Generated_Output:0.087891+0.12402i  -0.11523+0.022461i  0+0.087891i  -0.18848-0.33984i  0.17578-0.088867i  0.25+0.16699i  -0.26563-0.17676i  0.051758+0.025391i  0.087891-0.125i  -0.047852-0
Expected_Output :0.088379+0.12499i  -0.11462+0.022799i  0+0.088379i  -0.18833-0.33921i  0.17676-0.088379i  0.25083+0.1676i  -0.26514-0.17676i  0.052125+0.026239i  0.088379-0.12499i  -0.0469

Generated_Output:-0.026367+0.15039i  -0.018555+0.41113i  -0.088867-0.17676i  -0.1084+0.048828i  0.11426-0.0625i  0.10645+0.15137i  -0.088867-0.17676i  0.20898+0.26953i  -0.15137+0.025391i
Expected_Output :-0.025886+0.15087i  -0.018304+0.41134i  -0.088379-0.17676i  -0.10789+0.049338i  0.11426-0.062493i  0.10668+0.15238i  -0.088379-0.17676i  0.20972+0.26999i  -0.15087+0.025886

Generated_Output:-0.0625+0.27539i  0.18945+0.1543i  -0.17676+0.17676i  -0.13477-0.044922i  0.025391-0.0625i  0.1709+0.09668i  -0.17676  0.058594-0.044922i  0.0625-0.099609i  0.32422+0.05859
Expected_Output :-0.062493+0.27586i  0.18991+0.15451i  -0.17676+0.17676i  -0.13378-0.044189i  0.025886-0.062493i  0.17088+0.097169i  -0.17676+0i  0.058852-0.044189i  0.062493-0.099101i  0.3

Generated_Output:-0.0625-0.026367i  -0.097656-0.20801i  0.17676-0.17676i  0.14941+0.1543i  -0.099609-0.0625i  -0.05957-0.1123i  0-0.17676i  0.26465-0.13379i  0.0625-0.15137i  0.13281+0.1191
Expected_Output :-0.062493-0.025886i  -0.097169-0.20749i  0.17676-0.17676i  0.15015+0.15451i  -0.099101-0.062493i  -0.058852-0.11183i  0-0.17676i  0.26484-0.13378i  0.062493-0.15087i  0.133

Generated_Output:0.051758  -0.052734-0.010742i  0.17676+0.17676i  -0.03125+0.024414i  -0.21387-0.037109i  0.1875-0.12598i  0.26465+0.087891i  -0.24219+0.015625i  0.30176  0.014648-0.078125i
Expected_Output :0.051771+0i  -0.052125-0.010368i  0.17676+0.17676i  -0.03018+0.025031i  -0.21337-0.036608i  0.18833-0.12584i  0.26514+0.088379i  -0.24131+0.015517i  0.30174+0i  0.015517-0.

Generated_Output:-0.23926-0.11523i  0.16699-0.14551i  -0.088867-0.17676i  0.10645-0.0039063i  -0.026367+0.025391i  0.13867-0.1748i  -0.088867+0.17676i  -0.018555+0.15918i  -0.11426-0.23926i
Expected_Output :-0.23925-0.11426i  0.16694-0.145i  -0.088379-0.17676i  0.10668-0.0036409i  -0.025886+0.025886i  0.13893-0.17433i  -0.088379+0.17676i  -0.018304+0.15966i  -0.11426-0.23925i

Generated_Output:0.30176-0.42676i  -0.010742+0.16113i  0.087891+0.087891i  -0.084961+0.125i  0.036133-0.037109i  -0.12598+0.15137i  0  0.0019531+0.0097656i  0.051758+0.072266i  -0.078125+0.
Expected_Output :0.30174-0.42673i  -0.010368+0.16124i  0.088379+0.088379i  -0.084084+0.12584i  0.036608-0.036608i  -0.12584+0.15173i  0+0i  0.0020624+0.010368i  0.051771+0.073215i  -0.07801

Generated_Output:0.15039+0.15039i  0.0068359-0.05957i  0-0.088867i  -0.0097656+0.030273i  0.18652-0.23926i  0.25684+0.13281i  -0.17676+0.26465i  -0.066406-0.20117i  0.025391+0.025391i  0.00
Expected_Output :0.15087+0.15087i  0.0075817-0.058852i  0-0.088379i  -0.0087898+0.030735i  0.18748-0.23925i  0.25756+0.13378i  -0.17676+0.26514i  -0.066134-0.20021i  0.025886+0.025886i  0.0

Generated_Output:-0.088867+0.087891i  0.014648-0.17383i  0.17676  -0.044922+0.024414i  0.12402+0.17676i  0.024414-0.19434i  -0.088867+0.087891i  0.39746+0.015625i  -0.088867+0.087891i  -0.0
Expected_Output :-0.088379+0.088379i  0.015517-0.17367i  0.17676+0i  -0.044744+0.025031i  0.12499+0.17676i  0.025031-0.19348i  -0.088379+0.088379i  0.39733+0.015517i  -0.088379+0.088379i  -

Generated_Output:-0.28711  0.21387-0.10059i  0.087891  0.072266+0.22266i  -0.17676-0.088867i  0.16602+0.072266i  0-0.088867i  -0.10059-0.074219i  0.46289  0.050781+0.063477i  0.087891+0.176
Expected_Output :-0.28658+0i  0.21422-0.099955i  0.088379+0i  0.072862+0.22373i  -0.17676-0.088379i  0.16639+0.072862i  0-0.088379i  -0.099955-0.07407i  0.46334+0i  0.050917+0.063348i  0.08

Generated_Output:-0.21387+0.036133i  -0.052734-0.12207i  0.087891+0.087891i  -0.12695-0.033203i  -0.125-0.00097656i  0.1875+0.10742i  -0.17676-0.17676i  -0.010742+0.30371i  0.036133-0.21387
Expected_Output :-0.21337+0.036608i  -0.052125-0.1219i  0.088379+0.088379i  -0.12584-0.032313i  -0.12499+0i  0.18833+0.10826i  -0.17676-0.17676i  -0.010368+0.30381i  0.036608-0.21337i  0.01
```

Figure 138: Generated & Expected final output

Also, we make sure that the NB-synch signals are mapped successfully in the integration step, where the NPSS occupies 11 OFDM symbols and the rest are assumed to be zeros.



```
Generated_Output:-0.11426-0.0625i  0.17383-0.053711i  0.26465-0.17676i  0.016602+0.17383i  0.15039+0.098633i  -0.18164+0.032227i  0.087891-0.17676i  0.083984+0.14453i  -0.23926+0.061523i  0
Expected_Output :-0.11426-0.062493i  0.17433-0.053703i  0.26514-0.17676i  0.017096+0.17433i  0.15087+0.099101i  -0.18161+0.032967i  0.088379-0.17676i  0.084738+0.145i  -0.23925+0.062493i  0

Generated_Output:0.11426-0.0625i  0.11133-0.022461i  0+0.087891i  0.22852-0.044922i  -0.15137+0.098633i  -0.12012-0.080078i  0.35352-0.088867i  -0.29004-0.044922i  0.23926+0.0625i  -0.02343
Expected_Output :0.11426-0.062493i  0.11183-0.022245i  0+0.088379i  0.22944-0.044189i  -0.15087+0.099101i  -0.11911-0.079589i  0.35352-0.088379i  -0.2898-0.044189i  0.23925+0.062493i  -0.02

Generated_Output:-0.27637-0.026367i  0.079102-0.29688i  0.35352+0.087891i  -0.24512-0.097656i  0.061523+0.061523i  0.11719-0.2002i  -0.17676-0.088867i  0.23633-0.15527i  0.098633-0.15137i
Expected_Output :-0.27586-0.025886i  0.079589-0.29587i  0.35352+0.088379i  -0.2441-0.097169i  0.062493+0.062493i  0.11791-0.20021i  -0.17676-0.088379i  0.23682-0.15451i  0.099101-0.15087i

Generated_Output:-0.27637+0.025391i  0.14648-0.13281i  0.087891  -0.24512-0.097656i  0.18652-0.11523i  -0.045898-0.13281i  0.087891-0.17676i  0.23633-0.15527i  0.098633+0.15039i  0.24219-0.
Expected_Output :-0.27586+0.025886i  0.14723-0.13257i  0.088379+0i  -0.2441-0.097169i  0.18748-0.11426i  -0.045398-0.13257i  0.088379-0.17676i  0.23682-0.15451i  0.099101+0.15087i  0.24289-

Generated_Output:-0.0625-0.010742i  -0.14746-0.1123i  0.26465+0.17676i  -0.16992+0.09668i  -0.15137-0.099609i  0.044922+0.11914i  0.087891-0.17676i  0.080078+0.1543i  0.0625+0.36328i  -0.24
Expected_Output :-0.062493-0.010722i  -0.14723-0.11183i  0.26514+0.17676i  -0.16918+0.097169i  -0.15087-0.099101i  0.045398+0.11911i  0.088379-0.17676i  0.080797+0.15451i  0.062493+0.36424i

Generated_Output:0.051758+0.12402i  0.12012-0.19434i  0.17676  -0.079102-0.075195i  0.21289-0.088867i  0.17871+0.084961i  -0.088867+0.087891i  -0.12598-0.1709i  0.30078-0.125i  0.091797+0.1
Expected_Output :0.051771+0.12499i  0.12069-0.19348i  0.17676+0i  -0.078011-0.07457i  0.21337-0.088379i  0.17882+0.085292i  -0.088379+0.088379i  -0.12584-0.17075i  0.30174-0.12499i  0.09267

Generated_Output:0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Expected_Output :0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

Generated_Output:0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Expected_Output :0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

Generated_Output:0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Expected_Output :0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

Generated_Output:-0.23828+0.013672i  0.25781-0.0087891i  -0.010742+0.032227i  -0.09082-0.17773i  0.2373+0.074219i  0.19629+0.12988i  -0.19141+0.15625i  -0.064453+0.042969i  -0.056641+0.0478
Expected_Output :-0.23739+0.013804i  0.2586-0.0085922i  -0.0098267+0.032654i  -0.090252-0.17694i  0.23739+0.074585i  0.19653+0.13063i  -0.19061+0.15668i  -0.063771+0.043076i  -0.055949+0.04

Generated_Output:-0.23828+0.013672i  0.25781-0.0087891i  -0.010742+0.032227i  -0.09082-0.17773i  0.2373+0.074219i  0.19629+0.12988i  -0.19141+0.15625i  -0.064453+0.042969i  -0.056641+0.0478
Expected_Output :-0.23739+0.013804i  0.2586-0.0085922i  -0.0098267+0.032654i  -0.090252-0.17694i  0.23739+0.074585i  0.19653+0.13063i  -0.19061+0.15668i  -0.063771+0.043076i  -0.055949+0.04

Generated_Output:-0.23828+0.013672i  0.25781-0.0087891i  -0.010742+0.032227i  -0.09082-0.17773i  0.2373+0.074219i  0.19629+0.12988i  -0.19141+0.15625i  -0.064453+0.042969i  -0.056641+0.0478
Expected_Output :-0.23739+0.013804i  0.2586-0.0085922i  -0.0098267+0.032654i  -0.090252-0.17694i  0.23739+0.074585i  0.19653+0.13063i  -0.19061+0.15668i  -0.063771+0.043076i  -0.055949+0.04

Generated_Output:-0.23828+0.013672i  0.25781-0.0087891i  -0.010742+0.032227i  -0.09082-0.17773i  0.2373+0.074219i  0.19629+0.12988i  -0.19141+0.15625i  -0.064453+0.042969i  -0.056641+0.0478
Expected_Output :-0.23739+0.013804i  0.2586-0.0085922i  -0.0098267+0.032654i  -0.090252-0.17694i  0.23739+0.074585i  0.19653+0.13063i  -0.19061+0.15668i  -0.063771+0.043076i  -0.055949+0.04

Generated_Output:0.1709+0.046875i  -0.18945-0.11328i  0.051758+0.032227i  -0.11426+0.18848i  -0.17188-0.047852i  -0.12988-0.30859i  -0.066406-0.03125i  0.23145-0.067383i  -0.089844-0.36035i
Expected_Output :0.17108+0.047274i  -0.18893-0.11325i  0.052673+0.032654i  -0.11325+0.18893i  -0.17108-0.047274i  -0.12972-0.30831i  -0.065613-0.030823i  0.23222-0.066998i  -0.089419-0.3601
```

Figure 139: NPSS in final comparison file

- **When the system starts "start_top = 1", the data propagates through the P/S then CRC then encoder and the following waveform shows that the rate matcher has received its 3 input streams successfully**

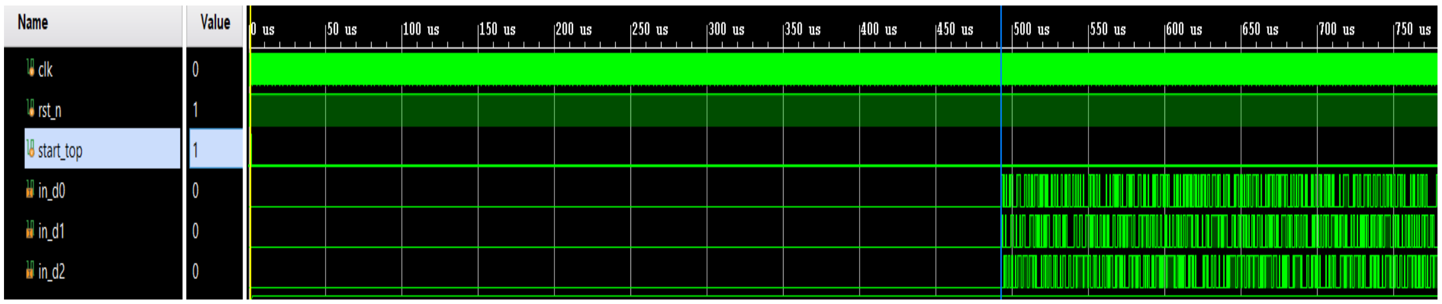


Figure 140: Integration of the first half of the chain

- **When the Resource Element Mapper sends a "stop" signal to the IFFT indicating that it is the last symbol to be sent, the IFFT receives that symbol and de-asserts the "data_valid" signal. Now the Resource Element Mapper can de-assert the "stop" signal after this successful handshaking.**

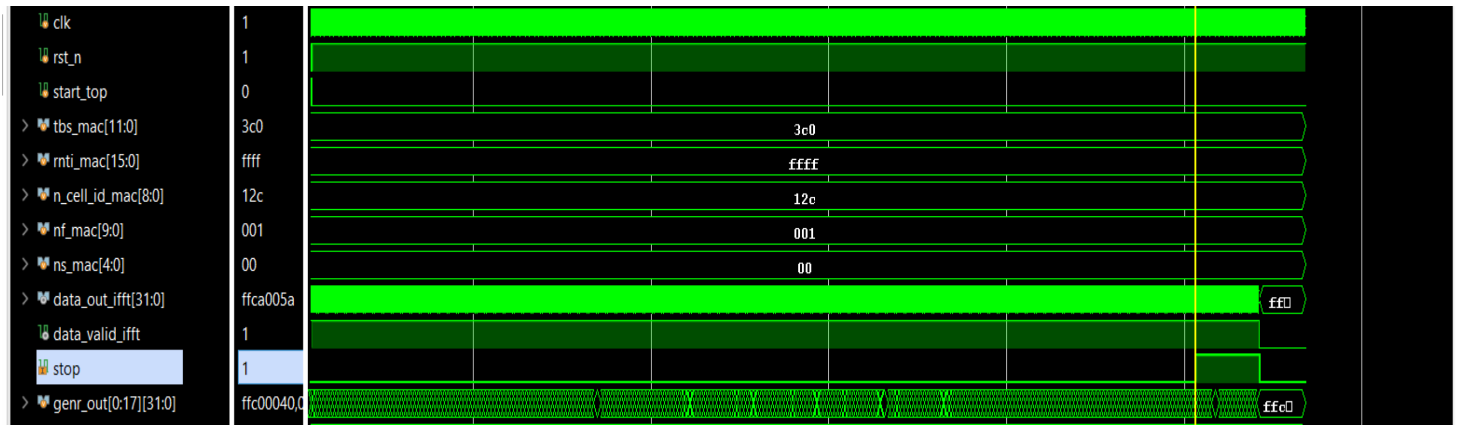


Figure 141: Handshaking between Resource Element Mapper and IFFT

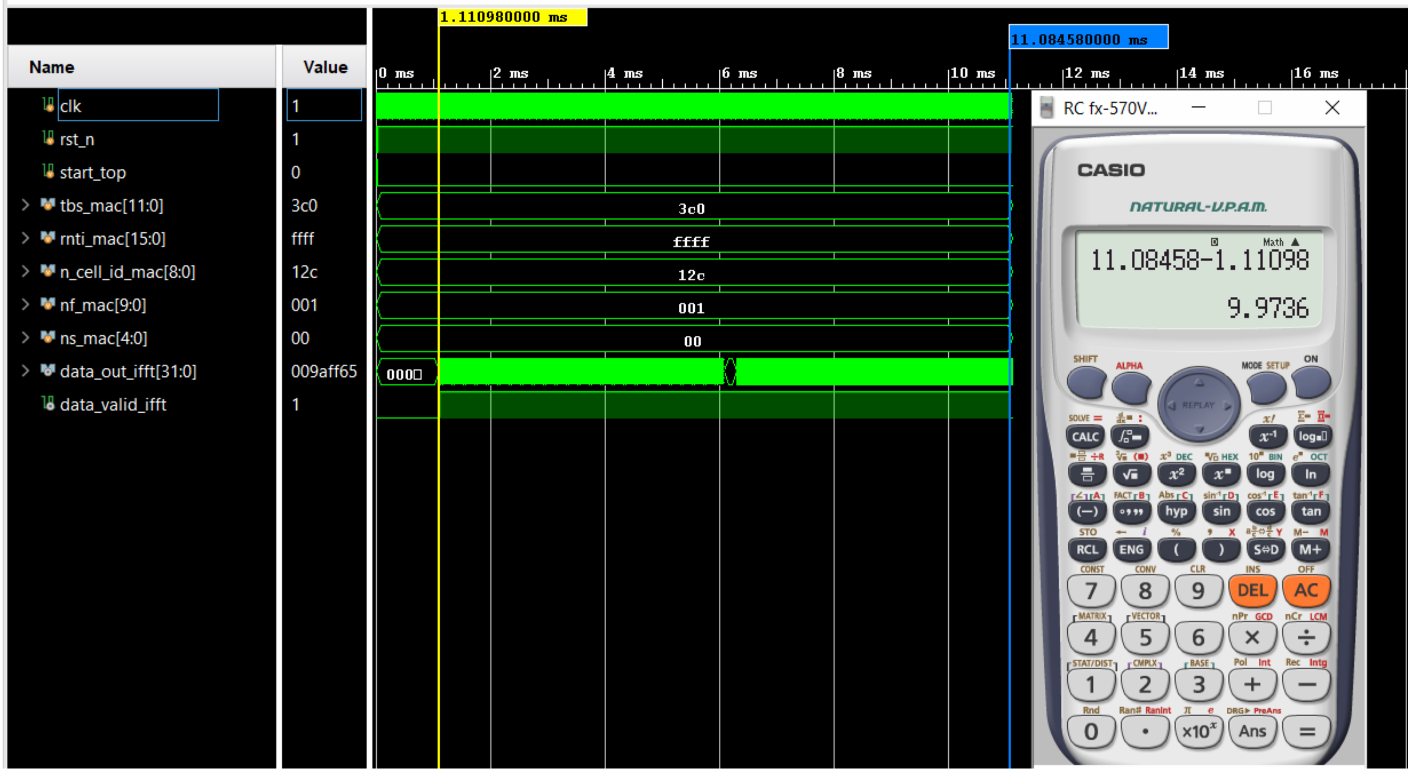
5.6. System Latency

- **Calculated :**

Latency = 1 (RAM) + 1 (P/S) + [TBS] (CRC) + 1 (RAM) + 1 (P/S) + 1 (Encoder) + [TBS+25] (Rate Matcher) + 1 (Scrambler) + 1 (Modulator) + 24 (REM) + 137 (IFFT)

For TBS = 960, Latency = $\frac{2113*520}{10^6} = \mathbf{1.09876}\ \boldsymbol{ms}$

- **From Waveform :**
  The latency is **1.11098 ms** (difference between "data_out_ifft" assertion and de-assertion) Also the below figure shows that we achieved the specified time by the standard for the NPDSCH for one frame



Name	Value
clk	1
rst_n	1
start_top	0
tbs_mac[11:0]	3c0
rnti_mac[15:0]	ffff
n_cell_id_mac[8:0]	12c
nf_mac[9:0]	001
ns_mac[4:0]	00
data_out_ifft[31:0]	009aff65
data_valid_ifft	1

```
11.08458-1.11098
9.9736
```

Figure 142: The frame time specified by the 3GPP

5.7. Synthesis Results

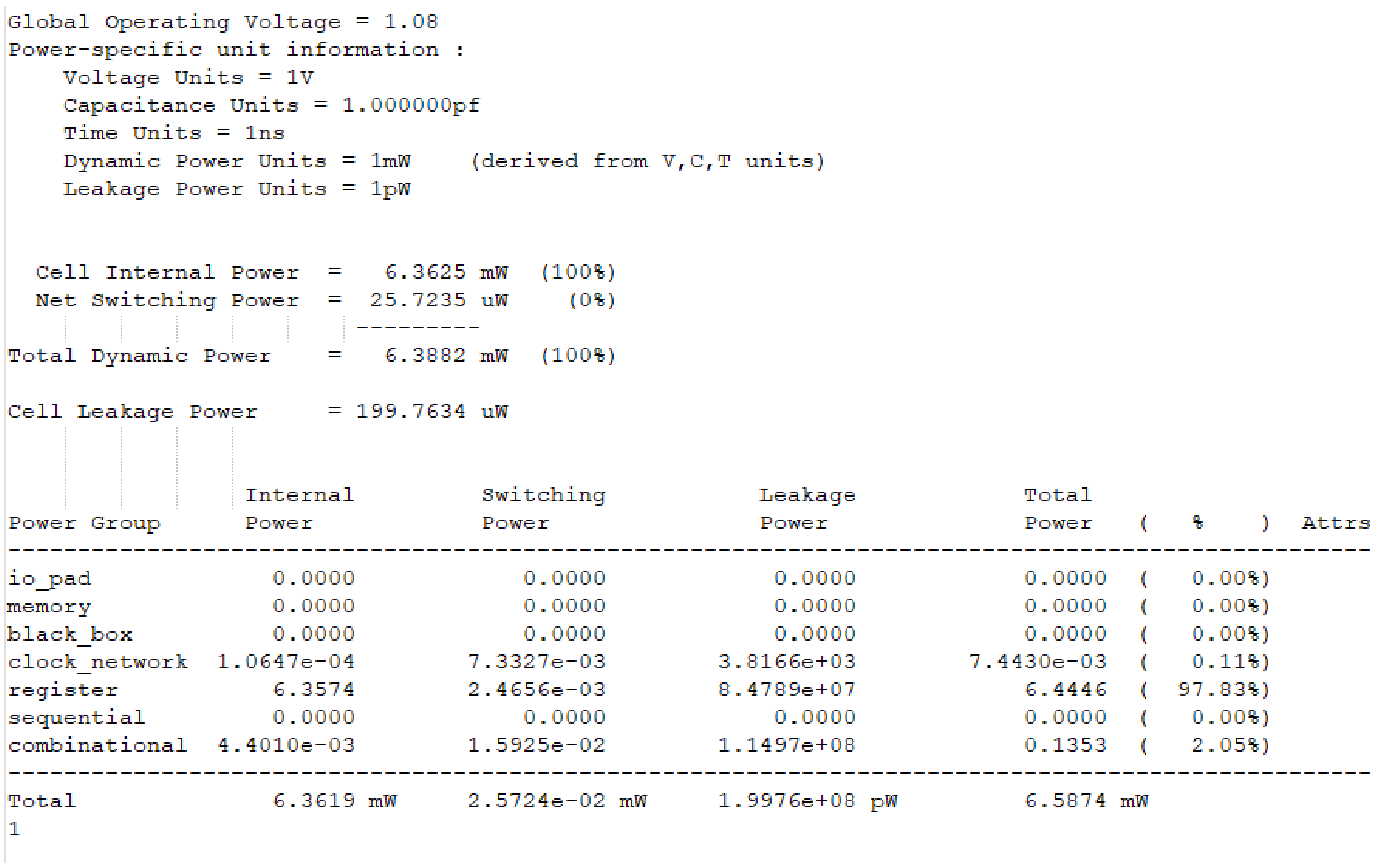
5.7.1. Whole Chain ASIC Synthesis Results

Synthesis is the process that translate and map RTL code written in HDL into a technology specific gate-level netlist, optimized for a set of pre-defined constraints.

You start with: behavioral RTL design, standard cell library & set of design constraints.

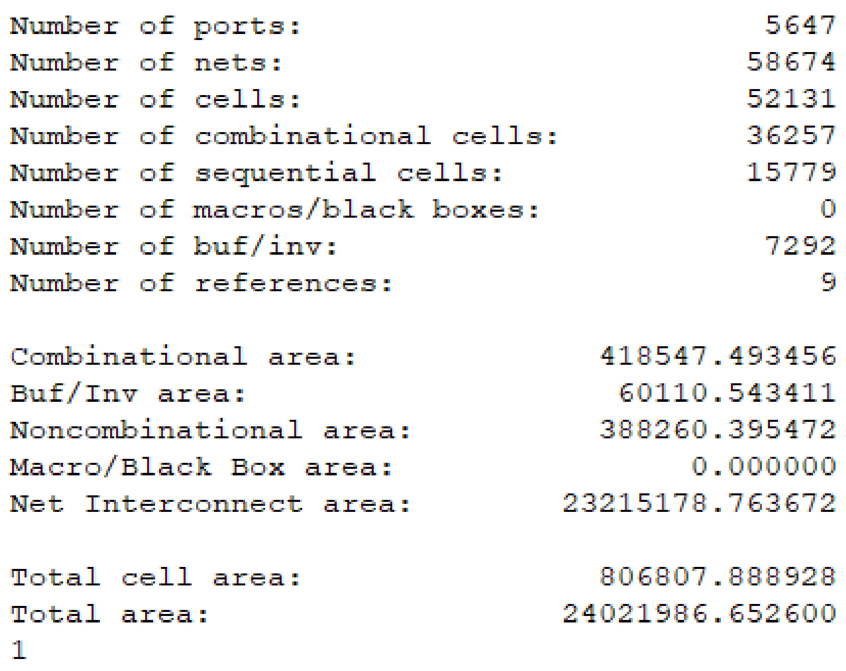
You finish with: gate-level netlist, mapped to the standard cell library Timing, Area and Power reports.

In this thesis we work with TSMC 130 nm technology, we operate at 1.92MHz, such factors are critical in determination of area, power and speed of the system.



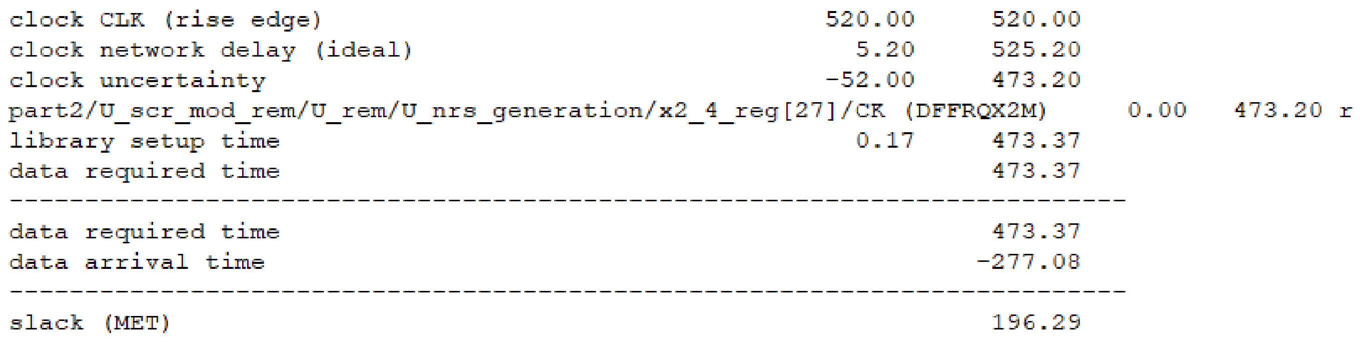
```
Global Operating Voltage = 1.08
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


  Cell Internal Power  =   6.3625 mW  (100%)
  Net Switching Power  =  25.7235 uW    (0%)
                         ---------
Total Dynamic Power    =   6.3882 mW  (100%)

Cell Leakage Power     = 199.7634 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power    (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000            0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000            0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000            0.0000            0.0000            0.0000  (   0.00%)
clock_network  1.0647e-04        7.3327e-03        3.8166e+03        7.4430e-03  (   0.11%)
register           6.3574        2.4656e-03        8.4789e+07            6.4446  (  97.83%)
sequential         0.0000            0.0000            0.0000            0.0000  (   0.00%)
combinational  4.4010e-03        1.5925e-02        1.1497e+08            0.1353  (   2.05%)
--------------------------------------------------------------------------------------------------
Total              6.3619 mW     2.5724e-02 mW     1.9976e+08 pW         6.5874 mW
1
```

Figure 143: Total Power for the integrated system



```
Number of ports:                          5647
Number of nets:                          58674
Number of cells:                         52131
Number of combinational cells:           36257
Number of sequential cells:              15779
Number of macros/black boxes:                0
Number of buf/inv:                        7292
Number of references:                        9

Combinational area:              418547.493456
Buf/Inv area:                     60110.543411
Noncombinational area:           388260.395472
Macro/Black Box area:                 0.000000
Net Interconnect area:         23215178.763672

Total cell area:                 806807.888928
Total area:                    24021986.652600
1
```

Figure 144: Area report for the integrated system



```
clock CLK (rise edge)                                      520.00     520.00
clock network delay (ideal)                                  5.20     525.20
clock uncertainty                                          -52.00     473.20
part2/U_scr_mod_rem/U_rem/U_nrs_generation/x2_4_reg[27]/CK (DFFRQX2M)      0.00    473.20 r
library setup time                                           0.17     473.37
data required time                                                    473.37
--------------------------------------------------------------------------------
data required time                                                    473.37
data arrival time                                                    -277.08
--------------------------------------------------------------------------------
slack (MET)                                                           196.29
```

Figure 145: Timing_max (setup) report for the integrated system

5.7.2. Summary for Power and Area results

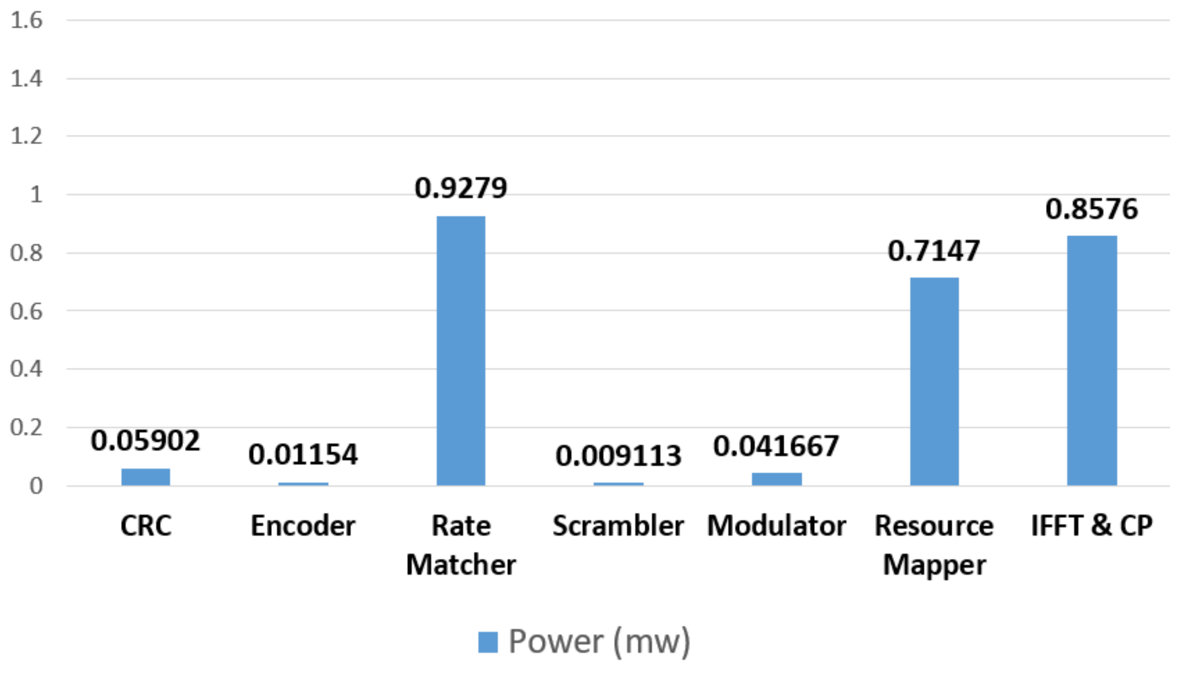


Figure 146: Power Results for all blocks

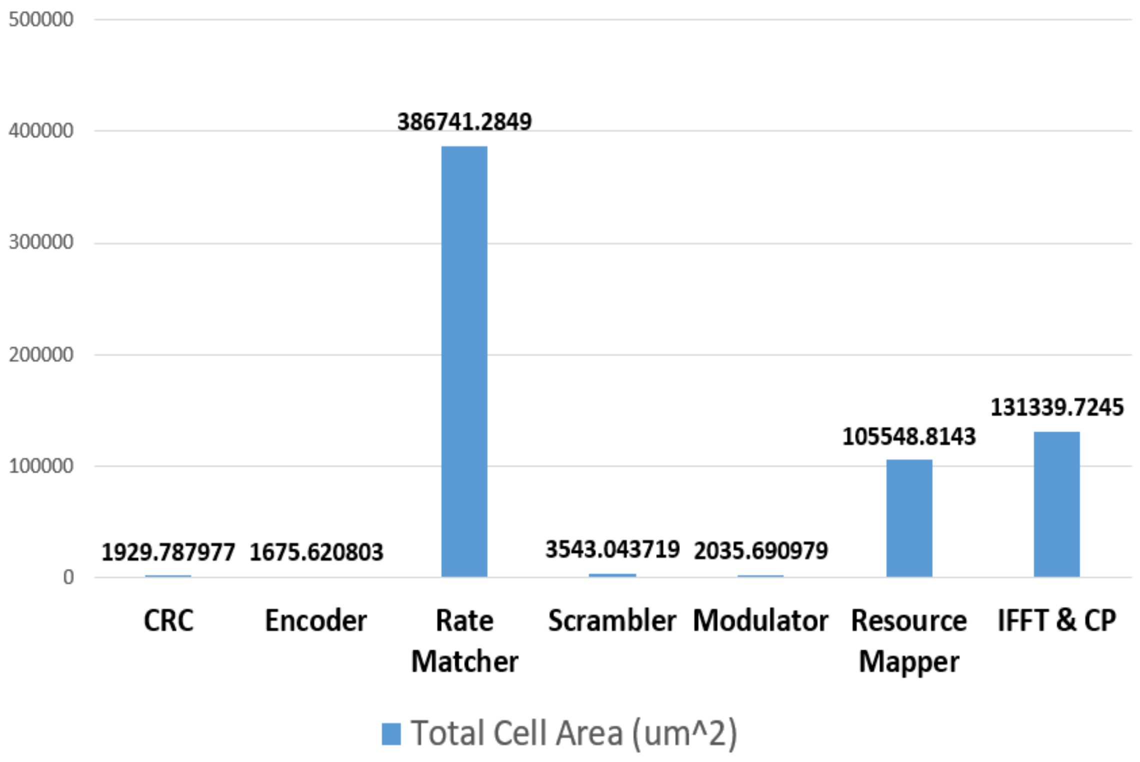
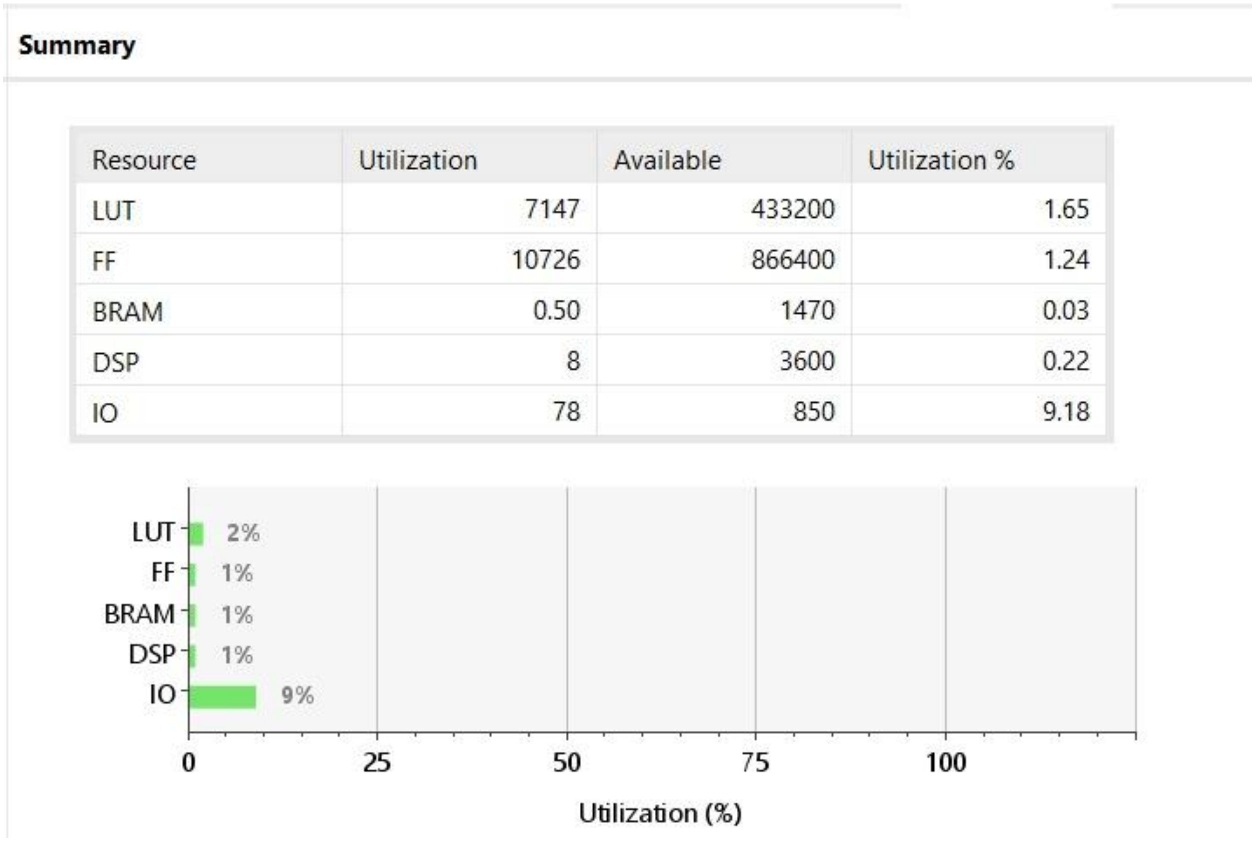


Figure 147: Area Results for all blocks

5.8. Full System Vivado Utilization for Vertex-7


Summary

Resource	Utilization	Available	Utilization %
LUT	7147	433200	1.65
FF	10726	866400	1.24
BRAM	0.50	1470	0.03
DSP	8	3600	0.22
IO	78	850	9.18

Figure 148: : Vivado Utilization for Vertex-7 for the integrated chain

Chapter Six

6. FPGA Implementation

The whole chain is now implemented on the Vertex-7 FPGA and we interface with the kit through the differential clock pins as shown in Figure 149.

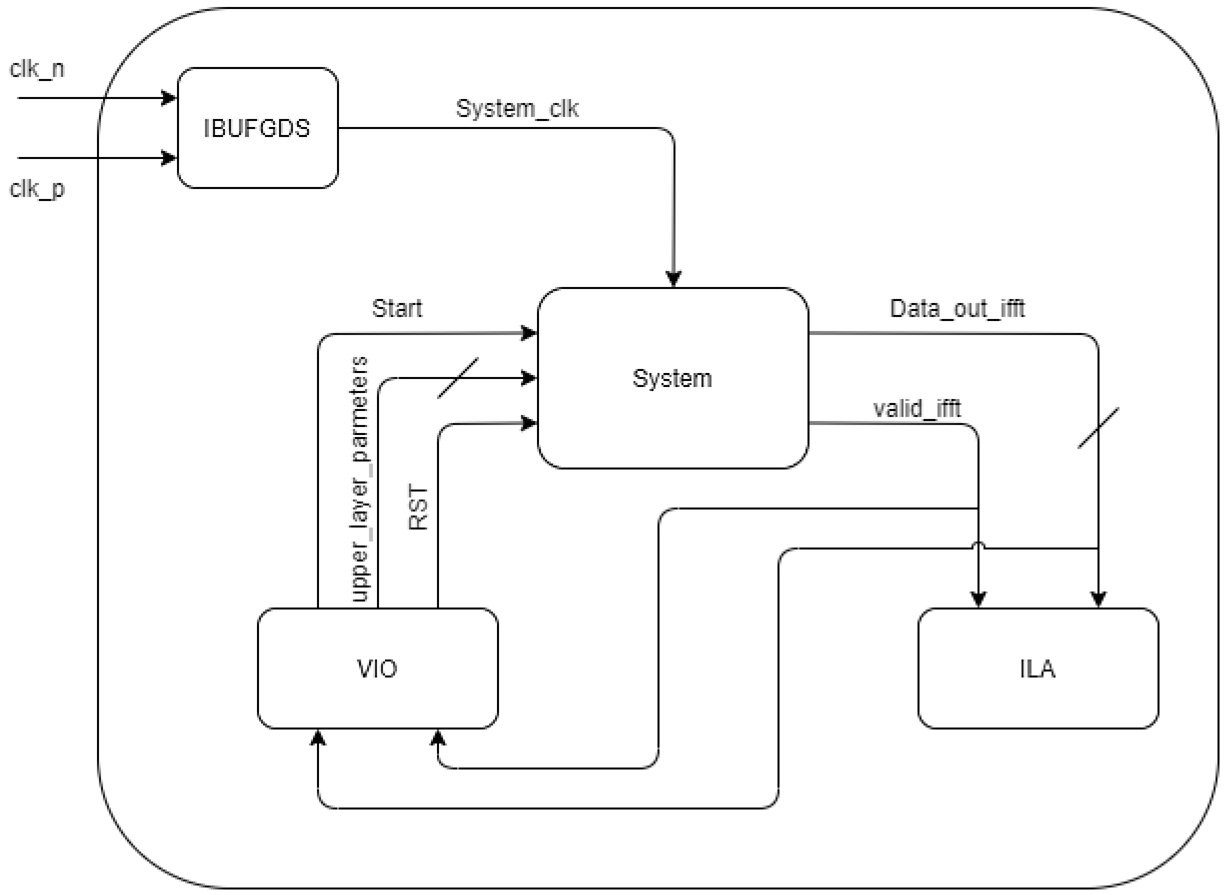


Figure 149: FPGA Architecture

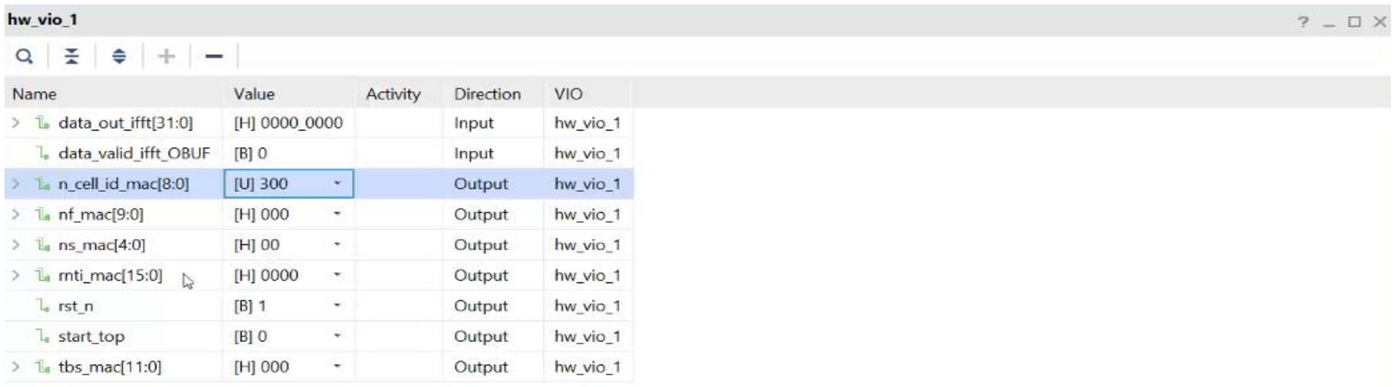
The IBFGDS is a block used to convert differential pair clock signal into a single clock signal.

The virtual input/output (VIO) core is a customizable core "that both monitor and drive internal FPGA signals in real time.

The integrated logic analyzer (ILA) feature allows you to perform in-system debugging of post-implemented design on an FPGA device you can use it to monitor signals in design.

We modify the system so it has an embedded ROM stores all the data instead of write them in a testbench, and only a "start" signal is triggered so the system starts to read the data from the ROM to pass throw the chain until get out from the IFFT to be captured by the ILA ip.

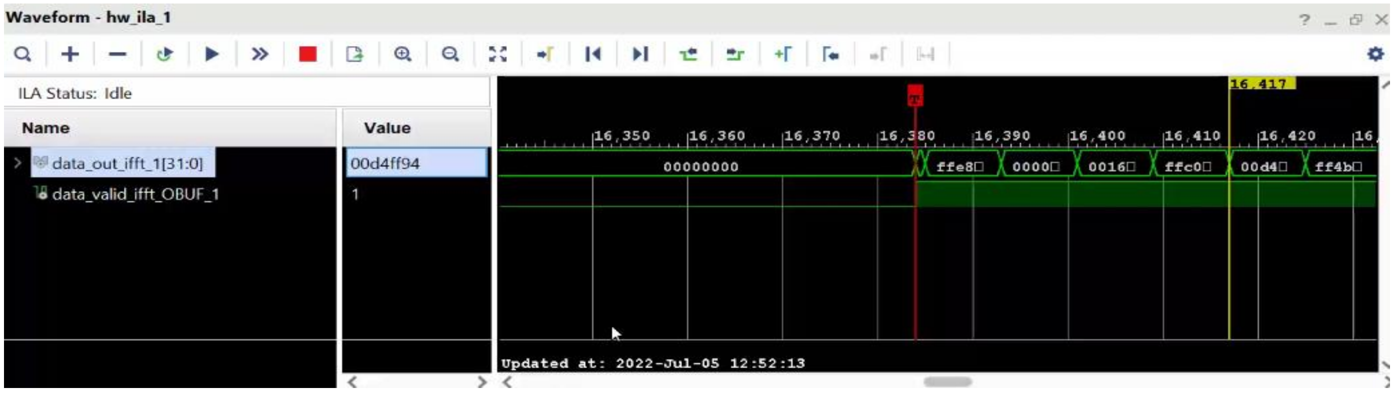
To run the system we reset, enter the upper layer parameters then trigger the “start” signal, all this signals are triggered through the VIO in the GUI in Vivado as shown in Figure 150.


hw_vio_1

Name	Value	Activity	Direction	VIO
data_out_ifft[31:0]	[H] 0000_0000		Input	hw_vio_1
data_valid_ifft_OBUF	[B] 0		Input	hw_vio_1
n_cell_id_mac[8:0]	[U] 300		Output	hw_vio_1
nf_mac[9:0]	[H] 000		Output	hw_vio_1
ns_mac[4:0]	[H] 00		Output	hw_vio_1
rnti_mac[15:0]	[H] 0000		Output	hw_vio_1
rst_n	[B] 1		Output	hw_vio_1
start_top	[B] 0		Output	hw_vio_1
tbs_mac[11:0]	[H] 000		Output	hw_vio_1

Figure 150: VIO

Then the ILA capture the data, and draw a wave form as shown in Figure 151.


Waveform - hw_ila_1

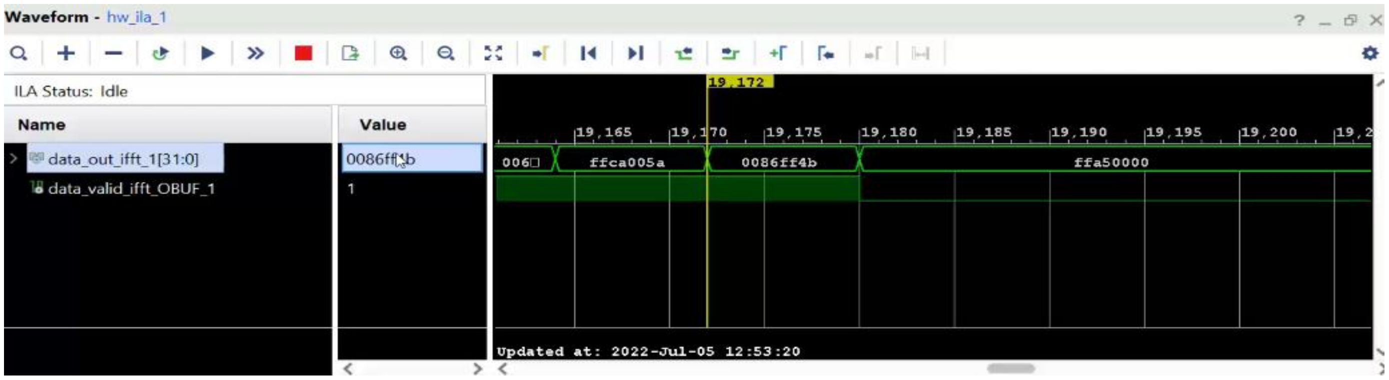
ILA Status: Idle

Name	Value
data_out_ifft_1[31:0]	00d4ff94
data_valid_ifft_OBUF_1	1

Updated at: 2022-Jul-05 12:52:13

Figure 151: ILA waveform (at the beginning)

Then when the chain finishes processing on data, the valid signal is deactivated as expected, see Figure 152.


Waveform - hw_ila_1

ILA Status: Idle

Name	Value
data_out_ifft_1[31:0]	0086ff4b
data_valid_ifft_OBUF_1	1

Updated at: 2022-Jul-05 12:53:20

Figure 152: ILA waveform (at the end)

Conclusion

In this thesis we introduced the design and implementation of the physical layer of downlink shared channel for the narrow-band IoT LTE Transmitter chain. The full transmitter chain was introduced then we started with some theory concepts then the 3GPP Release-14 in which our design meets this standard, after that we walked through the MATLAB modeling for the transmitter’s blocks and RTL implementation which is verified using the MATLAB’s built-in functions. Then we integrated the whole system including the RAM which stores the incoming transport block size, also the register file which contains the upper layer parameters. Finally we synthesized the blocks and the integrated system on ASIC DC compiler using tsmc 130 nm technology and reported slacks, area, and power, then we implemented the system of Vertex-7 FPGA.

References

[1] Evolution of NB-IoT and its implementation, simnovus.

[2] core.ac.uk/download/pdf/154281786.pdf

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

[4] ieeexplore.ieee.org/abstract/document/231911

[5] 3GPP Technical Specifications 36.212, “Physical channel and modulation.

[6] 3GPP Technical Specifications 36.213, “Physical channel and modulation.

[7] www.semanticscholar.org/paper/Performance-of-a-fixed-delay-decoding-scheme-for-Sung-Kim/f4892d662af80524d7649c20cec42c0975c21cf1

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

[9] ieeexplore.ieee.org/document/8306882

[10] International Journal of Computer Applications (0975 – 8887) Volume 116 – No. 7, April 2015

[11] tutorialspoint.com/what-is-fixed-point-representation-in-computer-architecture

[12] ieeexplore.ieee.org/document/9194757

[13] tutorialspoint.com/what-is-floating-point-representation-in-computer-architecture

[14] rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/lte/content/lte_overview

[15] ieeexplore.ieee.org/document/5231183

[16] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation (Release 14) V14.7.0 (2018-06)

[17] 3GPP TS 36.201 V14.1.0 (2017-03)

[18] onelab-eg.com/wp-content/uploads/2020/06/PDF.pdf