Low power design of the physical layer chain of Narrow band LTE uplink receiver

Aya Adel Ismail

Karim Ahmed Alaa

Kareem Ahmed Farouk

Lina Eissa Hassan

Omar Mohamed Abdelkader

Zainab Adel Ibrahim

Under the supervision of

Associate prof. Hassan Mostafa

Si-Vision

A thesis Submitted to the Faculty of

Engineering at Cairo University

in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Science

In

Electronics and Communications Engineering

Faculty of Engineering, Cairo University Giza, Egypt

JULY 2019

Abstract

One of the main requirements to achieve integrity between humans and machines is the need for a common ground where both of them can reach which gave birth to the evolution of IoT (Internet-Of-Things), and by the support of LTE, NB-IoT shortened the path towards such integrity.

One now can deploy a radio-access network with low battery, cover a wide area, efficiently power low-cost devices, and match different spectrum allocations of operators.

With tons of applications like smart metering, monitoring, agriculture, and fleet and logistics management. The optimization for such processes of detecting and reporting different variables like temperature and humidity is now needed as we expand. As for sensor-heavy applications, data rate and latency should be lower.

LTE NB-IoT is a solution that can match these requirements of power, range, and performance.

LTE NB-IoT technology supports a range of data rates for several applications and for several environmental conditions. It depends on the channel quality, or the signal-to-noise ratio and the quantity of resources in certain areas (bandwidth). In addition, each device has a specific power budget, which leads to combine the power of several devices. Focus transmission energy without losing performance to a narrower bandwidth. This efficiency frees up bandwidth for other devices.

In this thesis a digital implementation of the Narrowband Physical Uplink Shared Channel (NPUSCH) receiver which is based on NB-IoT LTE is proposed. The LTE NB-IoT is a new cellular technology introduced in 3GPP Release 13 to support IoT applications.

The standard specifications were first studied well to extract the information needed for implementation. Then a MATLAB model is developed for each block in the chain based on understanding the standard and the models were checked using MATLAB built in functions and assuming the existence of a dummy transmitter.

Then RTL model is implemented for each block in the chain and the RTL results were compared and verified by MATLAB results. The blocks were fully synthesizable on Xilinx ISE and on DC compiler using 130 nm technology. Also area and power for each block are reported.

Acknowledgments

We are using this opportunity to express our gratitude to everyone who supported us throughout the graduation project.

Firstly, we would like to express our sincere gratitude to our supervisor Dr. Hassan Mostafa for the continuous support and guidance.

We also wish to thank our advisors at Si-vision, Eng. Khaled Ismail and Eng. Ahmed Rady for their patience, motivation, and immense knowledge.

Finally, we wish express the gratitude and appreciation we owe to our families for providing us with unfailing support and continuous encouragement throughout our years of study.

This accomplishment would not have been possible without all of you. Thank you.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1. Motivation

The fourth generation of mobile phone standards LTE (Long Term Evolution), developed by the 3GPP (3rd Generation Partnership Project), offers an up to seven times faster upload speed with upload speeds of up to 50mbps. Retrofitting the infrastructure of UMTS (3G) to LTE-Advanced (4G) will not be a hurdle, as the basic scheme of UMTS persists. In addition to the higher capacity, the benefits of 4G LTE are the significantly lower latency times. These play a key role in the smooth retrieval of IOT applications that rely on real-time information, such as data from production systems or traffic information.

Narrow Band IoT (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.

NB-IOT technology also focuses transmission energy without losing performance to a narrower bandwidth. This efficiency frees up bandwidth for other devices. NB-IOT uses tones or subcarriers rather than resource blocks. Its bandwidth is 15 kHz, a relevant difference when compared with a resource block, whose effective bandwidth is 180 kHz.

There are many applications on IOT like smart homes, wearables, traffic management, water distribution, smart grid, connected cars, connected health, … etc.

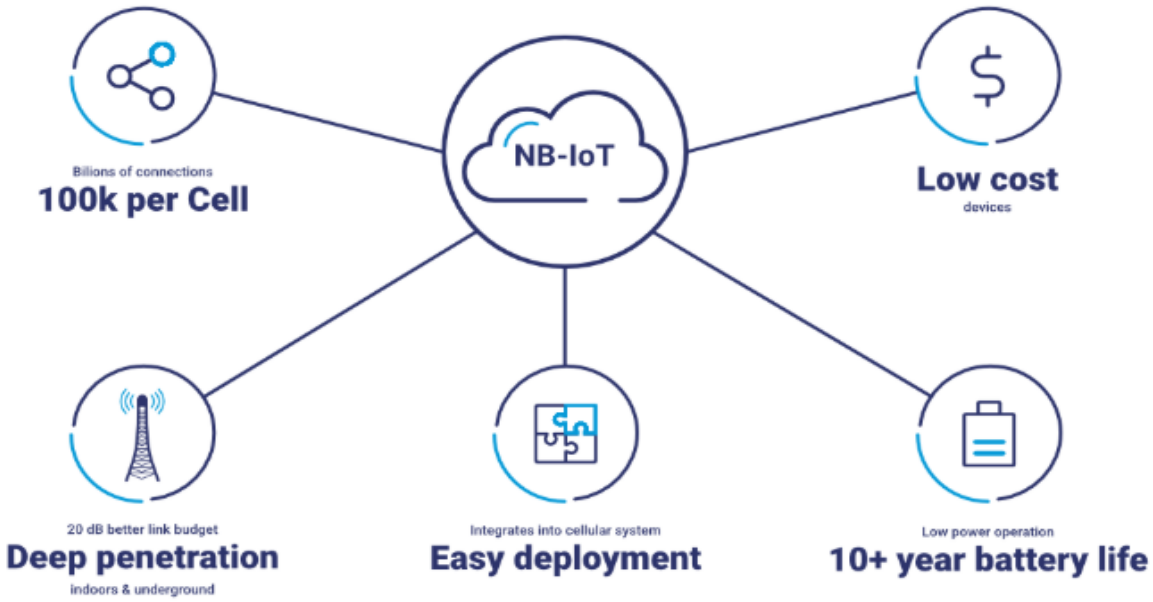


Figure 1.1-1 NB-IOT features

Narrow Band LTE, or NB-LTE is a new suite of technologies being developed by 3GPP, an international conglomerate of Telecommunications Company responsible for developing and maintaining the 4G LTE communications standard, among others.
The Internet of Things (IOT) is the primary application area of NB-LTE technology. "Release 13" simply refers to the platform release number of the release. When 3GPP comes up with a new stable communication platform, they release it to the public. Release 13 is one of these releases, and it outlines the basic communication requirements of NB-LTE technology. NB-LTE is also sometimes referred to as "NB-IOT" or "Narrowband IOT" technology, given its usefulness and many handy applications for the Internet of Things.
Essentially, NB-LTE allows devices to communicate over long distance with cellular networks, without using much bandwidth or power.

1.2. Multiple Access Techniques (OFDM overview)

OFDMA

LTE takes advantage of OFDMA, a multi-carrier scheme that allocates radio resources to multiple users. OFDMA uses Orthogonal Frequency Division Multiplexing (OFDM). For LTE, OFDM splits the carrier frequency bandwidth into many small subcarriers spaced at 15 kHz. OFDMA assigns each user the bandwidth needed for their transmission. Unassigned subcarriers are off, thus reducing power consumption and interference.
OFDMA uses OFDM; however, it is the scheduling and assignment of resources that makes OFDMA distinctive. The OFDM diagram in Fig. 1.3-1 below shows that the entire bandwidth belongs to a single user for a period. In the OFDMA diagram, multiple users are sharing the bandwidth at each point in time.

SC-FDMA

In the uplink, LTE uses a pre-coded version of OFDM called SC-FDMA. SC-FDMA has a lower PAPR (Peak-to-Average Power Ratio) than OFDM. This lower PAPR reduces battery power consumption, requires a simpler amplifier design and improves uplink coverage and cell-edge performance. In SCFDMA, data spreads across multiple subcarriers, unlike OFDMA where each subcarrier transports unique data. The need for a complex receiver makes SC-FDMA unacceptable for the downlink.

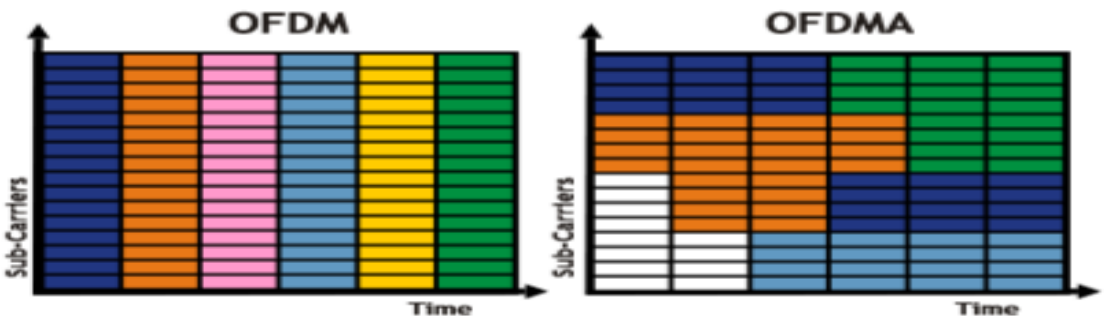


Figure 1.2-1 OFDM vs OFDMA

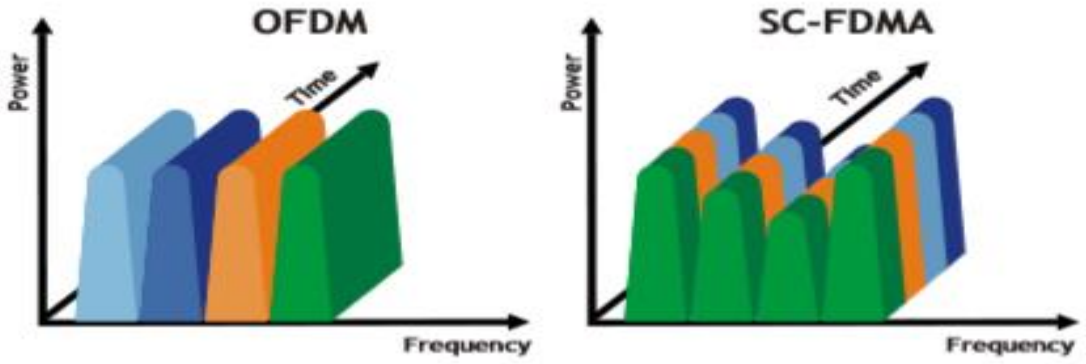


Figure 1.2-2 OFDM vs SC-FDMA

1.3. Frame structure

T_s is the basic time unit for LTE. Time domain fields are typically defined in terms of T_s. T_s is defined as = 1/ (15000 x 2048) seconds or about 32.6 nanoseconds.

- Downlink and uplink transmissions are organized into frames of duration $T_f = 307200T_s$.
- The 10 ms frames divide into 10 sub-frames. Each sub-frame divides into 2 slots of 0.5 ms. In the time domain, a slot is exactly one Resource Block long.
- Two frame types are defined for LTE: Type 1, used in Frequency Division Duplexing (FDD) and Type 2, used in Time Division Duplexing (TDD).
- Type 1 frames consist of 20 slots with slot duration of 0.5 ms.
- Type 2 frames contain two half frames. Depending on the switch period, at least one of the half frames contains a special sub-frame carrying three fields of switch information: Downlink Pilot Time Slot (DwPTS), Guard Period (GP) and Uplink Pilot Time Slot (UpPTS). If the switch time is 10 ms, the switch information occurs only in sub-frame one. If the switch time is 5 ms, the switch information occurs in both half frames, first in sub-frame one, and again in sub-frame six. Sub-frames 0 and 5 and DwPTS are always reserved for downlink transmission. UpPTS and the sub-frame immediately following UpPTS are reserved for uplink transmission. Other sub-frames can be uplink or downlink.

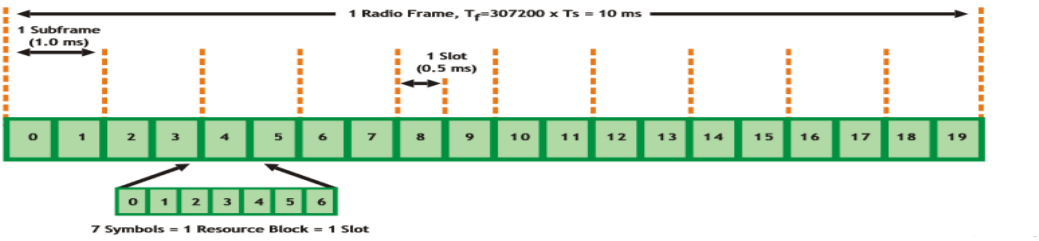


Figure 1.3-1 Type 1 frame

In LTE, ten 1 ms sub-frames compose a 10 ms frame. Each sub-frame divides into two slots. The smallest modulation structure in LTE is the Resource Element. A Resource Element is one 15 kHz subcarrier by one symbol. Resource Elements aggregate into Resource Blocks. A Resource Block has dimensions of subcarriers by symbols. Twelve consecutive subcarriers in the frequency domain and six or seven symbols in the time domain form each Resource Block. The number of symbols depends on the Cyclic Prefix (CP) in use. When a normal CP is used, the Resource Block contains seven symbols. When an extended CP is used, the Resource Block contains six symbols. A delay spread that exceeds the normal CP length indicates the use of extended CP.

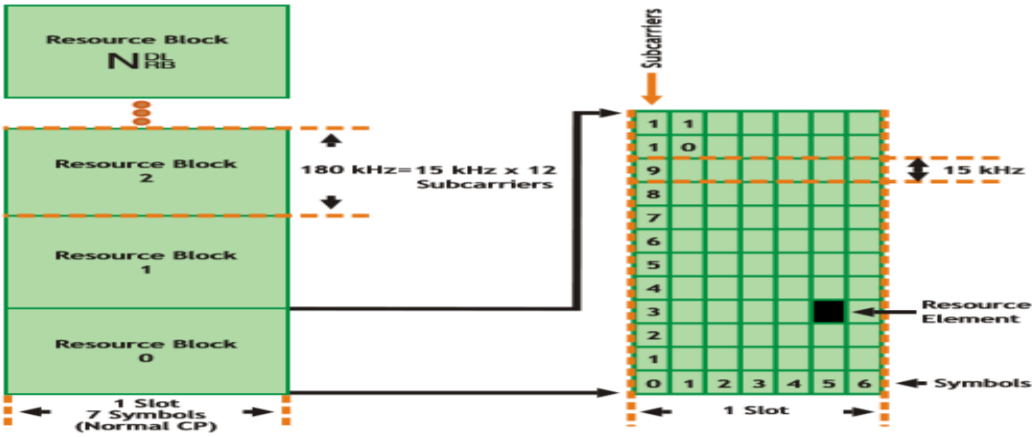


Figure 1.3-2 Relationship between a slot, symbols and Resource Blocks

Channel Bandwidth is the width of the channel as measured from the lowest channel edge to the highest channel edge. The channel edge is the center frequency ± (channel bandwidth/2). Transmission Bandwidth is the number of active Resource Blocks in a transmission. As the bandwidth increases, the number of Resource Blocks increases. The Transmission Bandwidth Configuration is the maximum number of Resource Blocks for the particular Channel Bandwidth.

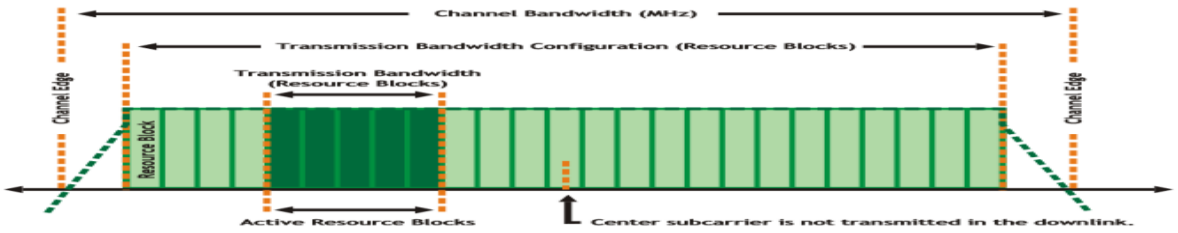


Figure 1.3-3 Relationships between Channel Bandwidth, Transmission Bandwidth Configuration, and Transmission Bandwidth

1.4. Problem description

The target of this project is to implement low power uplink receiver 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 physical layer chain includes but not subjected to the following:

- Scrambling
- Modulation Mapper
- OFDM signal generation
- Turbo Coding
- CRC
- FFT
- Interleaving

The project has gone through the following phases:

- Verilog Training
- Standard and literature reading
- System Modeling using MATLAB
- Testing and verification using Synopsys VCS tool
- RTL Design
- Synthesis using Synopsys Design Compiler
- Prototyping with FPGA (Optional) – Documentation

1.5. Thesis organization

In chapter 1 we give a brief overview about the NB-LTE and its use in the IOT applications. We also give an overview about the multiple access techniques used in LTE.
In chanter 2, we give an overview about the uplink receiver chain and the algorithms chosen for implementation.
In chanter 2, we explained the standard specifications and any assumptions needed for implementation.
In chapter 4, we proposed the architecture and design for each block in the chain. We also showed the MATLAB, RTL and synthesis results on DC compiler.

Chapter 2
Receiver uplink chain and sub-blocks' function

2.1. Chain block diagram

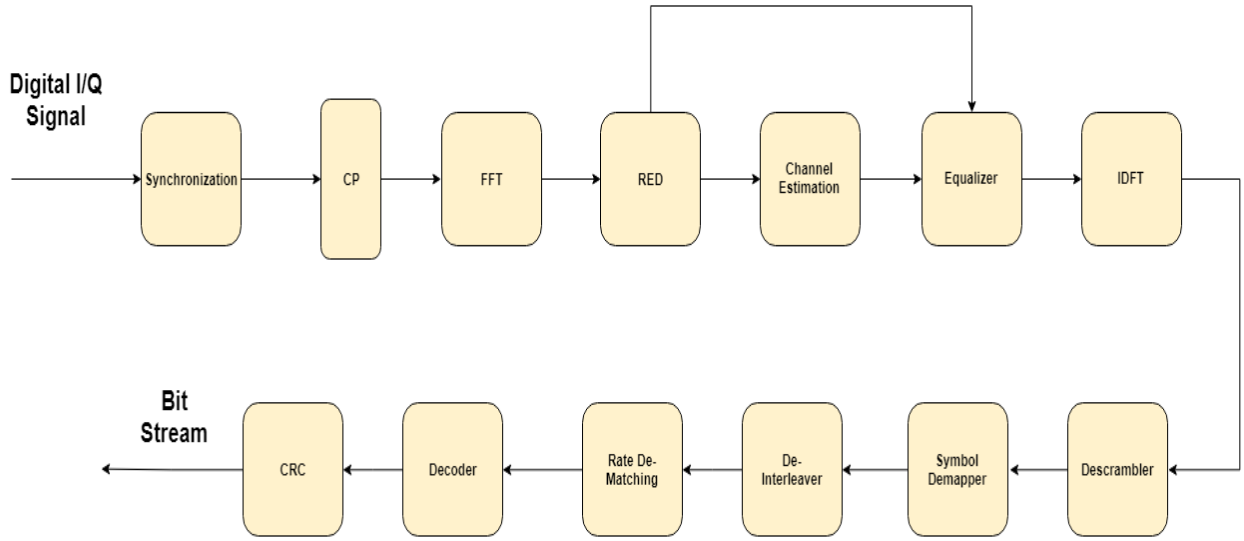


Figure 2.1-1 Uplink receiver chain block diagram

2.2. Synchronization

OFDM systems are very sensitive towards carrier frequency offsets. Carrier frequency offset (CFO) happens due to mismatch of RF oscillator frequency at the transmitter and the receiver, and also due to Doppler shift. The frequency offset causes two problems, one is the reduction of amplitude of the signal and the other is introduction of Inter carrier interference (ICI).

Synchronization of an OFDM signal is required and it consists of two major parts: carrier frequency offset (CFO) and symbol time offset (STO) which is an estimate of when the symbol starts.

The estimation of synchronization error can be performed depending on the type of the training data and can be divided into two parts: pre-FFT synchronization and post-FFT synchronization. Post-FFT synchronization is based on the known pilot data or training data that are inserted in various location of the signal. The known pilot data may be one or two OFDM symbols and we can also name this method as data-aided or preamble-based synchronization and it is performed in the frequency domain as it depends on the location of pilots which is determined after the grid is fully recognizable. On the other hand, the pre-FFT synchronization is based on the cyclic prefix that can be used as training or data and. This method is also known as non-data aided synchronization or cyclic prefix-based synchronization and it is performed in time domain. Both time-domain and frequency-domain synchronization play important roles in correcting carrier frequency offset in OFDM systems.

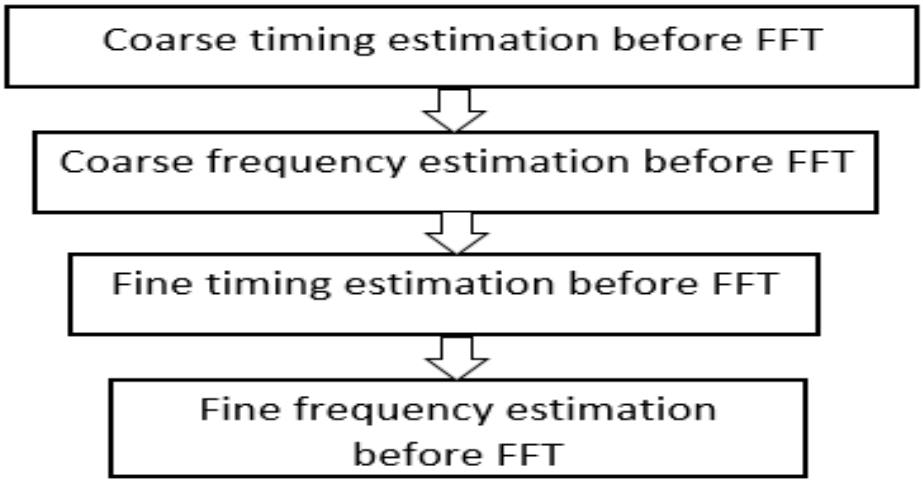


Figure 2.2-1 Synchronization flow in OFDM receiver

2.2.1. Coarse frequency and timing estimation

- System model

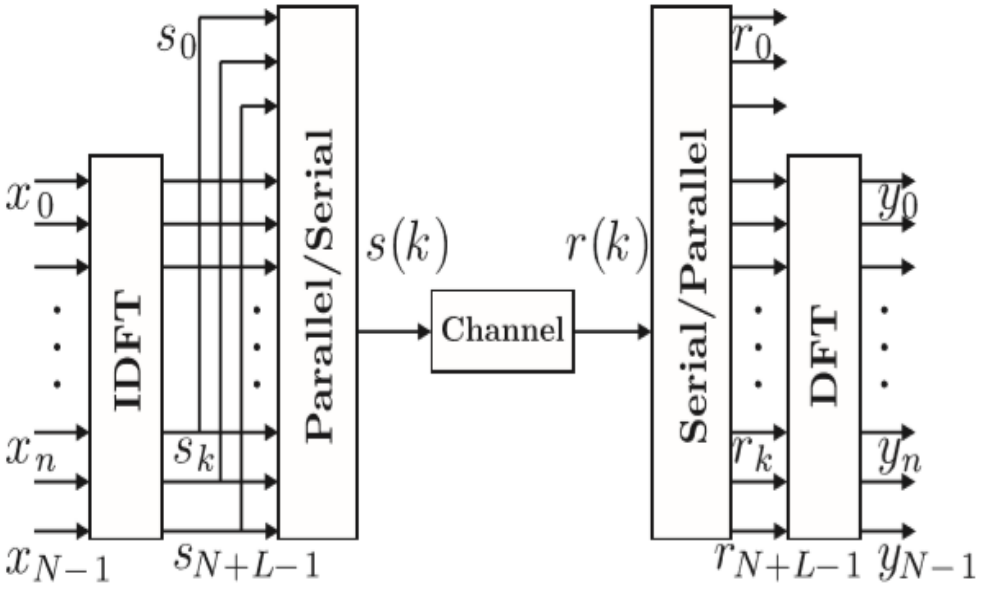


Figure 2-3 The OFDM system model.

The complex data symbols are modulated by inverse discrete Fourier transform on N parallel subcarriers then passed through the channel and demodulated at the receiver by discrete Fourier transform. The insertion of a cyclic prefix results in an equivalent parallel orthogonal channel structure and also decreases the ICI effect.
The length of the transmitted OFDM symbol is (N+L) where N is the FFT points and L is the cyclic prefix length.
In the analysis we consider the following:

1. The channel is nondispersive and that the transmitted signal S(k) is only affected by complex additive white Gaussian noise (AWGN) n(k).
2. The uncertainty in the arrival time of the OFDM symbol and is modelled as a delay in the channel impulse response $\delta(k - \theta)$ where θ is the integer value representing the unknown arrival of the symbol.
3. The uncertainty in carrier frequency (a difference in the local oscillators in the transmitter and receiver gives rise to a shift of all the subcarriers) and is modelled as a complex multiplicative distortion of the received data in the time domain $e^{j2\pi\varepsilon k/N}$, where ε is the frequency shift.

These considerations give the following received signal

$$r(k) = S(k - \theta)\, e^{j2\pi\varepsilon k/N} + n(k) \quad (1)$$

There is an observation here is that r(k) is not a white process even if S(k) approximates a complex Gaussian process whose real and imaginary parts are independent and that's because the cyclic prefix yields a correlation between some pairs of samples, that are spaced N samples apart. But since r(k) has a probabilistic structure, it contains information about the time offset and the frequency offset.

- ML Estimation of Time and Frequency Offset

The maximum likelihood estimation is based on the idea that since the cyclic prefix is a part of the end of the symbol that is appended in its start so if we take a window of L (cyclic prefix length) and correlate it with another window that is N samples apart from the other one and keep doing this and move the window sample by sample, then there will be some time when this correlation gives a maximum value as there has to be some time when the two correlated windows are those that are in the beginning and the end of the symbol. At this time, we can detect the right start of the symbol.

The coarse timing and FFO is obtained from the log likelihood function according to the below set of equations.

$$\gamma\,(n) = \sum_{k=m}^{m+L-1} r(k) r^*(k+N) \quad (2)$$

$$\Phi\,(n) = \sum_{k=m}^{m+L-1} |r(k)|^2 + |r(k+N)|^2 \quad (3)$$

$$\widehat{T_c} = \arg \max_n \{|\gamma\,(n)| - \rho\Phi\,(n)\} \quad (4)$$

$$\hat{\epsilon} = \frac{-1}{2\pi} \angle\, \gamma\,(n) \quad (5)$$

Where γ (n) is the correlation, Φ (n) is the energy, $\widehat{T_c}$ is the estimated time offset and $\hat{\epsilon}$ is the estimated frequency offset.
Since the argument operator arg (.) is performed by using $tan^{-1}(.)$, the range of FFO estimation in this equation is $[\frac{-\pi,+\pi}{2\pi}]$ = [-0.5, +0.5] so that $|\hat{\epsilon}| \leq 0.5$. Hence, this technique is useful for the estimation of Fractional frequency offset and it does not estimate the integer offset.

2.2.2. CORDIC algorithm

CORDIC is an iterative algorithm for the calculation of the rotation of a two-dimensional vector, in linear, circular and hyperbolic coordinate systems, using only add and shift operations. The algorithm can be used for generating sinusoidal waveform, multiplication and division operations, and evaluation of angle of rotation, trigonometric functions and logarithms. It consists of two operating modes, the rotation mode (RM) and the vectoring mode (VM).

In the rotation mode a vector (X, Y) is rotated by an angle θ to obtain a new vector(X', Y'). In every iteration i, fixed angles of the value $\arctan(2^{-1})$ which are stored in a ROM are subtracted or added from/to the remainder angleθ_i.

In the vectoring mode, the magnitude and phase of a vector (X, Y) are computed.

$$x' = x\,cos(\varphi) - y\,sin(\varphi)$$

$$y' = y\,cos(\varphi) + x\,sin(\varphi)$$

Rather than computing $\sin(\varphi)$ directly, we iteratively rotate β towards φ.

Step 1: set β = 45°

Step 2: if φ >= β then

$$\beta = \beta + (45/2)°$$

Else $\beta = \beta - (45/2)°$

$$x' = cos(\varphi)\,[\,x - y\,tan(\varphi)\,]$$

$$y' = cos(\varphi)\,[\,y + x\,tan(\varphi)\,]$$

Allow iterative rotation so that $\tan(\beta) = \pm 2^{-i}$

$$x_{i+1} = cos(\,tan^{-1}(\pm 2^{-i}))\cdot[\,x_i - y_i \cdot d_i \cdot 2^{-i}\,]$$

$$y_{i+1} = cos(\,tan^{-1}(\pm 2^{-i}))\cdot[\,y_i + x_i \cdot d_i \cdot 2^{-i}\,]$$

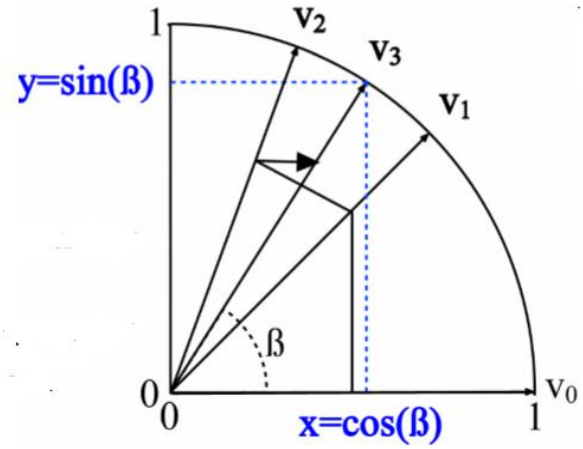
With the rotate direction $d_i = \pm 1$

We know that the cosine is symmetric so

$cos(\ tan^{-1}(2^{-i})) \ = \ cos(\ tan^{-1}(-2^{-i}))$

$cos(\ tan^{-1}(2^{-i}))$ is the gain Ki of an iteration

$k_i = cos(\ tan^{-1}(2^{-i})) = \frac{1}{\sqrt{1+2^{-i}}}$



i	2^{-i}	arctan(2^{-i})*360/2π		$45^* 2^{-i}$
0	1	45*		
1	0.5	26.56505118*	4.065051177*	22.5
2	0.25	14.03624347	0.753717879	11.25
3	0.125	7.125016349	0.106894615	5.625
4	0.0625	3.576334375	0.013826201	2.8125
5	0.03125	1.789910608	0.001743421	1.40625
6	0.015625	0.89517371	0.000218406	0.703125
7	0.0078125	0.447614171	2.73158E-05	0.3515625
8	0.00390625	0.2238105	3.41494E-06	0.17578125
9	0.001953125	0.111905677	4.26882E-07	0.087890625
10	0.000976563	0.055952892	5.33607E-08	0.043945313
11	0.000488281	0.027976453	6.6701E-09	0.021972656
12	0.000244141	0.013988227	8.33763E-10	0.010986328
13	0.00012207	0.006994114	1.0422E-10	0.005493164
14	6.10352E-05	0.003497057	1.30276E-11	0.002746582
15	3.05176E-05	0.001748528	1.62844E-12	0.001373291
16	1.52588E-05	0.000874264	2.03555E-13	0.000686646
17	7.62939E-06	0.000437132	2.54444E-14	0.000343323
18	3.8147E-06	0.000218566	3.18056E-15	0.000171661
19	1.90735E-06	0.000109283	3.97577E-16	8.58307E-05
20	9.53674E-07	5.46415E-05	4.96903E-17	4.29153E-05

We can compute K offline for all n iterations as $k = \prod_n k_i$ and it approaches 0.6037, if n goes to infinity.

Now we will summarize the iterative equations needed for both vectoring and rotation mode.

- Rotation mode

$x_{i+1} \ = \ k_i \cdot [\ x_i - \ y_i \cdot d_i \cdot 2^{-i}\ ]$

$y_{i+1} \ = \ k_i \cdot [\ y_i + \ x_i \cdot d_i \cdot 2^{-i}\ ]$

$z_{i+1} \ = \ [\ z_i - \ \cdot d_i \cdot tan^{-1}(-2^{-i})\ ]$

$d_i = \begin{cases} -1, z_i < 0 \\ +1, otherwise \end{cases}$

After n iterations we get

$$x_n = A_n \cdot [\, x_0 \cos(z_0) - Y_0 \sin(z_0) \,]$$

$$Y_n = A_n \cdot [\, Y_0 \cos(z_0) + X_0 \sin(z_0) \,] \qquad A_n = \prod_{i=0}^{n} \sqrt{1 + 2^{-i}}$$

$$z_n = 0$$

- Vectoring mode

$$x_{i+1} = k_i \cdot [\, x_i - y_i \cdot d_i \cdot 2^{-i} \,] \qquad d_i = \begin{cases} +1, y_i < 0 \\ -1, otherwise \end{cases}$$

$$y_{i+1} = k_i \cdot [\, y_i + x_i \cdot d_i \cdot 2^{-i} \,]$$

$$z_{i+1} = [\, z_i - \cdot d_i \cdot tan^{-1}(-2^{-i}) \,]$$

After n iterations we get

$$x_n = A_n \cdot \sqrt{{x_0}^2 + {y_0}^2}$$

$$Y_n = 0$$

$$z_n = z_0 + tan^{-1}(\frac{y_0}{x_0})$$

The value of $\tan^{-1}(-2^{-i})$ at each iteration can be obtained from the table in the last figure and will later be stored in a LUT (look up table) in the hardware implementation.

2.3. Cyclic prefix removal and offset correction

In this block the cyclic prefix that was added at the transmitter side is removed. Also the frequency offset in the received data is corrected According to the offset estimated in the previous block.

Since we earlier modelled the uncertainty in the offset as $(e^{j2\pi\varepsilon k/N})$ so correcting the offset will be simply done by multiplying the received data by $(e^{-j2\pi\varepsilon k/N})$.

2.4. FFT

FFT is an implementation of Discrete Fourier Transform (DFT) that is a digital way (digital input digital output) to implement Fourier transform which is only defined for continuous input signals. The FFT is an algorithm introduced in 1965, and it implements the DFT in a faster way due to the complexity of the DFT and the simplicity of FFT.

Equation 6 shows the Discrete Fourier Transform. In this equation x (0) …, x(N−1) are the input samples.

$$x(k) = \sum_{n=0}^{n=N-1} x(n) \times e^{-j2\pi n\frac{k}{N}}, k = 0,1, \ldots., N-1 \qquad (6)$$

$$WN = e^{-j2\pi n\frac{k}{N}} \qquad (7)$$

$$x(k) = \sum_{n=0}^{n=N-1} x(n) \times WN \qquad (8)$$

According to Eq. (6), the complexity of the DFT is O (N^2) so it is applicable for simple data size only, but for large data size it will be more complex and can't be implemented on chip so FFT is used instead of DFT because it is more efficient and faster due to its complexity O (Nlog N), where N is the data size.

The radix, r, stands for the number of parts that the input signal will be divided into. The radix-2 algorithm is the simplest and most used form; it divides the input signal into 2 parts. The FFT of the two parts can be calculated separately and can then be combined to form the complete DFT. This dividing into smaller parts is done recursively, requiring the number of samples of the input, N, to be a power of 2.

2.5. RED

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

2.6. Channel estimation

The main function of this Block is to estimate the changes happens in the data bits while it goes through the channel so that we could reverse this changes and detect the data correctly, sure we can't remove the channel effect perfectly but we can at least minimize it effect to the limit required to successfully detect the sent bits.
This happened by sending pilots with the data bits which we be effected also by the channel, and as the receiver also can generate these pilots using upper layer parameters, then if we divide the received pilots on the generated ones we get the channel effect which will be delivered to the equalizer to reverse it.
Also noise estimation is part of this block its function is to get the noise value so that we can calculate signal to noise ratio required by the de-mapper block.
We used the zero forcing technique to estimate the channel. Other techniques like interpolation get more accurate estimation but need higher area and power implementation and as the estimation difference not to large compared to the area and the power needed for it, so we used the implementation of the lower area and power.

2.7. Equalizer

In wireless communication data are sent in radio space, the channel exhibits multipath fading phenomenon which produces inter-symbol interference (ISI) in the signal received at the receiver side. The received signal is a filtered and noise-corrupted version of the transmitted sequence:

$$r_k = s_k \otimes c_k + n_k$$

Where r_k is the received signal, ck is the channel and nk is the noise added (usually it's an AWGN noise) while the symbol ⊗ represents the convolution operation. The chief goal of equalization is to rebuild the actual signal with the help of filter or any other methods and remove the effect of ISI so that the reliability of data transmission is maintained.

A linear equalizer is a filter that can undo the channel effect. Output of the equalizer can be documented as:

$$y_k = r_k \otimes h_{eq}$$

$$y_k = (s_k \otimes c_k) \otimes h_{eq} + n_k \otimes h_{eq}$$

Where $y_k$the output of the equalizer and h_{eq} is the impulse response of the equalizer.

2.8. IDFT

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

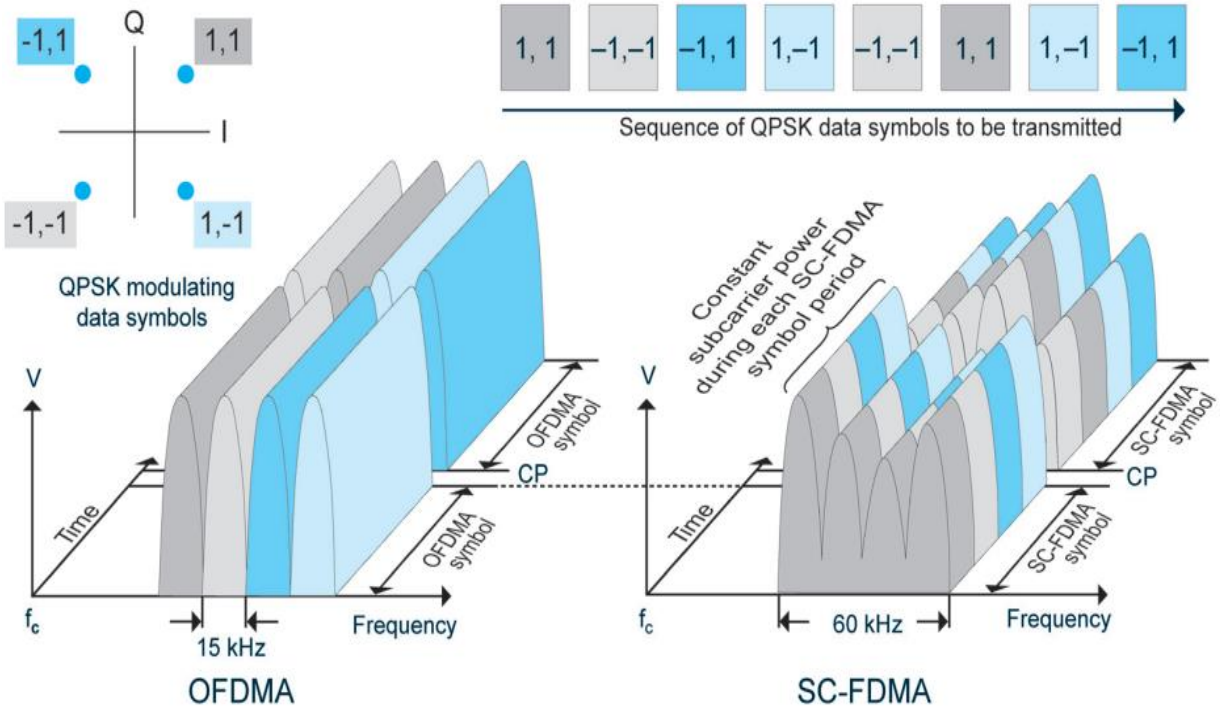


Figure 2.8-1 OFDMA vs SC-FDMA

Uplink uses SC-FDMA which is a modified form of the OFDM with similar throughput performance and complexity, SC-FDMA is viewed as DFT-coded OFDM where time-domain symbols are transformed to frequency domain symbols and then go through the standard OFDM modulation as shown in Figure 2.8-1.

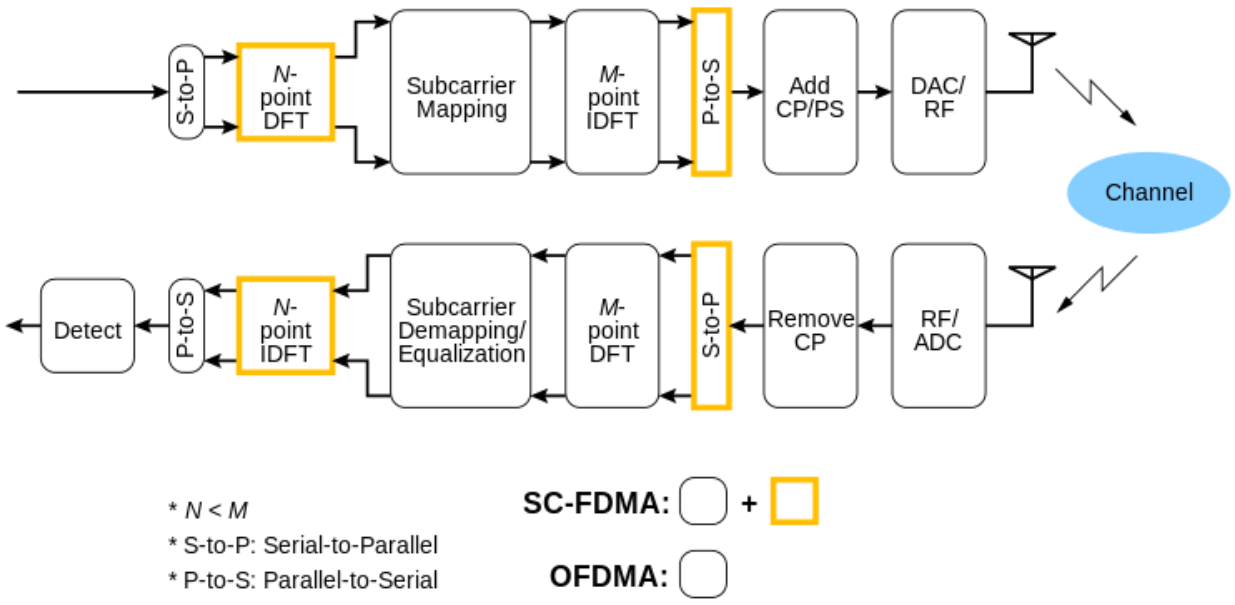


Figure 2.8-2 OFDM vs SC-FDM Block diagram

SC-FDMA has all the advantages of OFDM like robustness against multi-path signal propagation, the block diagram for the SC-FDMA is shown in Figure 2.8-2.

2.9. De-mapper

Symbol De-mapper in communication systems is the transition between complex valued signals into data bits. NB-LTE supports both QPSK and BPSK modulation schemes in the uplink. The De-mapper transforms the received symbols to bits according to symbol location represented by real and imaginary coordinates as shown in the following Figure 2.9-1.

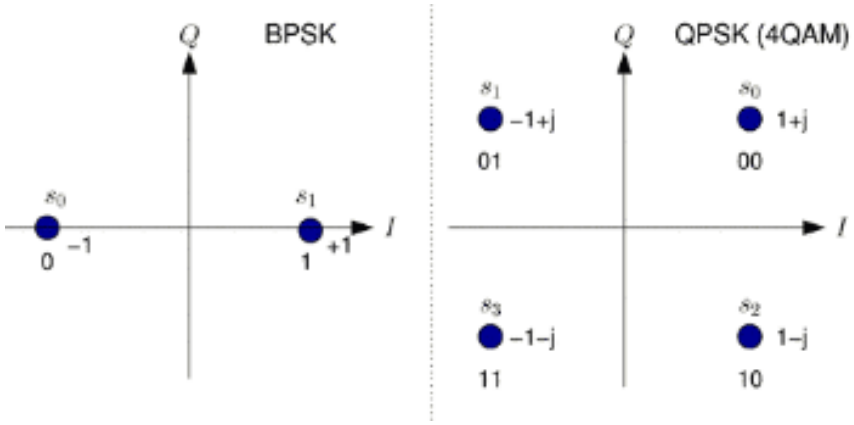


Figure 2.9-1 BPSK and QPSK constellation

The De-mapper output is a soft output which is the log likelihood ratio (LLR) as the input to the turbo decoder must be a soft input.

2.10. Descrambler

Scrambling is very important in communication systems. By using the scrambling code, NodeB can separate signals coming simultaneously from many different UEs and UE can separate signals coming simultaneously from many different NodeB.
The main role of scrambling is to randomize the data before they got modulated by the symbol De-mapper and also to avoid long sequence of zeros or ones as long sequence of zeros or ones cause transmission synchronization problems.

2.11. Data Demultiplexing and Channel De-Interleaver

The main function of this block is to reverse the operations done at the transmitter at the Data multiplexing and Channel interleaver Block.
The main function of Data multiplexing and Channel interleaver Block is control and data multiplexing which is performed such that HARQ-ACK information is present on both slots and is mapped to resources around the demodulation reference signals.
The multiplexing ensures that control and data information are mapped to different modulation symbols.
The interleaver solve the problem of burst errors which happens because of deep fading channels effects relate bits, the error bits after the interleaver are not in the same code word and in each code word a single bit error which is easier to the random error decoder is to correct this single error than the entire burst.
However, in narrowband we have no control data multiplexed; we keep using the block for the interleaver benefits.

2.12. Rate De-Matcher

The main function of this block is to reverse the operations done at the transmitter at the Rate Matching Block.

Rate matching in NPUSCH is a very important block in baseband processing. The main function of the Rate Matching Block is to ensure that the amount of data of the transport channel and the physical channel can adapt each other. The basic principle of rate matching is that when the amount of data of the transport channel is more than that carried by the physical channel, the system performs a punch operation; on the contrary, it operates repeatedly.

It also used to adapt and control the rate as the turbo encoder gives fixed 1/3 rate, we can increase or decrease rate depending on channel quality.

- HARQ and Redundancy Versions

HARQ, which stands for Hybrid Automatic Repeat Request, is an error correction mechanism in LTE based on retransmission of packets, which are detected with error. The functionality of the HARQ can be seen in Fig below. The transmitted packet arrives after a certain propagation delay in receiver. Receiver produces either an ACK for the case of error-free transmission or a NACK, if some errors are detected. The ACK/NACK is produced after some processing time and sent

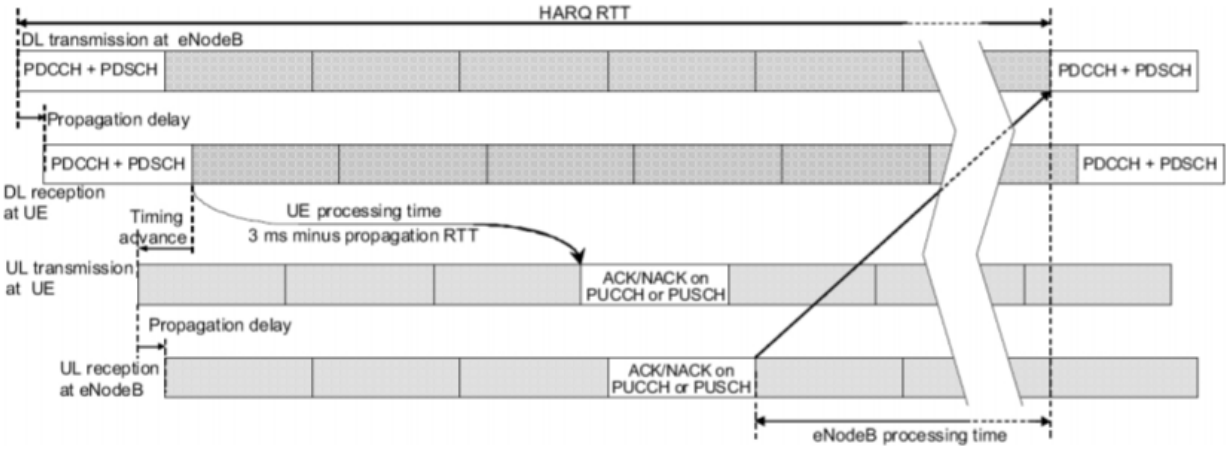


Figure 2.12-1 HARQ mechanism in LTE

back to transmitter and arrives there after a propagation delay. In the case of a NACK, after a certain processing delay in transmitter, the desired packet will be sent again. The bits, which are read out from the circular buffer and sent in each retransmission are different and depend on the position of the RV (Redundancy Version). There are four RVs (0, 1, 2, 3), which define the position of the starting point, where the bits are read out from the circular buffer.

2.13. Turbo Decoder

2.13.1. CHANNEL CODING

In a digital transmission system, error control is achieved by the use of channel coding schemes. Channel coding schemes protect the signal from the effects of channel noise and interference and ensure that the received information is as close as possible to the transmitted information. They help to reduce the BER and improve reliability of information transmission.

Channel coding schemes involve the insertion of redundant bits into the data stream that help to detect and correct bit errors in the received data stream. Due to the addition of the redundant bits, there is a decrease in data rate. Thus the price paid for using channel coding to reduce bit error rate is a reduction in data rate or an expansion in bandwidth.

- **Convolutional codes**

Convolutional codes are designed for real-time error correction. The code converts the entire input stream into one single code-word. The encoded bit depends not only on the current bit but also on the previous bit information

The design of a channel code is always a trade-off between energy efficiency and bandwidth efficiency. Low rate codes having more redundant bits can usually correct more errors. That means that the communication system can operate at lower transmit power, tolerate more interference and noise and transmit at higher data rate.

Thus, the code becomes more energy efficient. However, low rate codes also have a large overhead and have more bandwidth consumption. Also, the decoding complexity of the code also grows exponentially with code length. Thus, low rate codes set high computational requirements to the conventional decoders.

The turbo codes consist of component encoders separated by inter-leaver so that each encoder uses an interleaved version of the same information sequence. It consists of two encoders separated by the inter-leaver. The two encoders recursive systematic convolutional (RSC) encoders used are identical and the code is systematic concatenated in parallel, for better decoding it is required to have high weight (Hamming weight of a code-word is the number of ones that it contains) transmitted code-word because it means that they are more distinct, and thus the decoder will have an easier time distinguishing among them.

Inter-leaver is used to scramble bits before being input to the second encoder. This makes the output of one encoder different from the other encoder. Thus, even if one of the encoders occasionally produces a low-weight, the probability of both the encoders producing a low-weight output is extremely small. This improvement is known as inter-leaver gain. Another purpose of interleaving is to make the outputs of the two encoders uncorrelated from each other. Thus, the exchange of information between the two decoders while decoding yields more reliability.

Both the RSC encoders are of short constraint length in order to avoid excessive decoding complexity. An RSC encoder is typically of rate r = 1/2 and is termed a component encoder. The two component encoders are separated by an inter-leaver. The output of the turbo encoder consists of the systematic input data and the parity outputs from two constituent RSC encoders.

The systematic outputs from the two RSC encoders are not needed because they are identical to each other (although ordered differently) and to the turbo code input. Thus the overall code rate becomes r = 1/3.

Figure 2.13-1 shows the fundamental turbo code encoder.

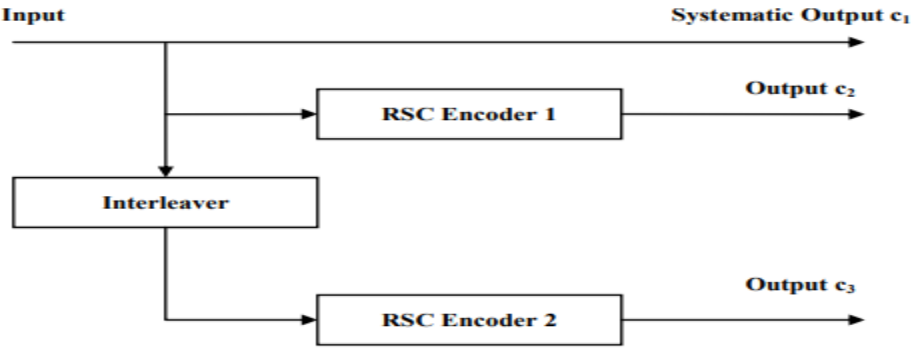


Figure 2.13-1 Fundamental turbo code encoder

The generator matrix of the encoder then becomes

$$G = \left[1, \frac{g2}{g1}\right]$$

Where 1 denotes the systematic output, g2 denotes the feed forward output, and g1 is the feedback to the input of the RSC encoder.
Figure 2.13-2 shows the RSC encoder.

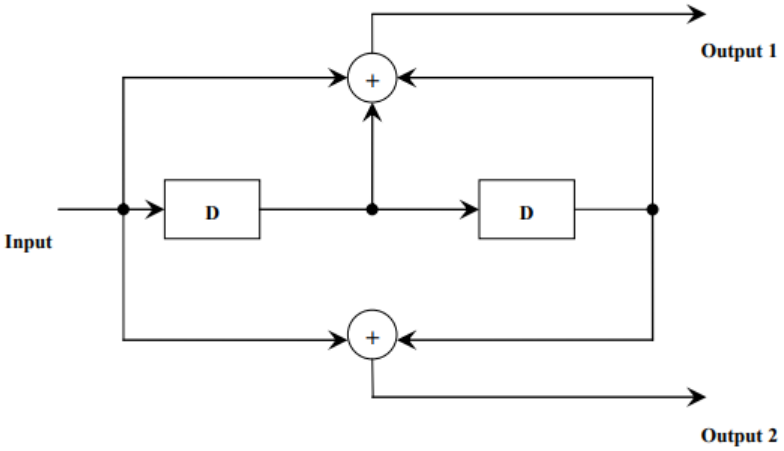


Figure 2.13-2 RSC conventional encoder with r = 1/2.

- **Trellis termination**

Unlike conventional convolutional codes which always use a stream of zeros as tail bits, the tail bits of a RSC depend on the state of the encoder when all the data bits have been encoded. Also because of the presence of inter-leaver between the two encoders, the final states of the two

component encoders will be different. Thus, the trellis termination bits for the two encoders will also be different and an RSC cannot be brought to an all zero state; simply by passing a stream of zeros through it. However, this can be done by using the feedback bit as the encoder input. This is done by using a switch at the input as shown in Figure 2.13-3.

The switch is in position A while encoding the input sequence and is switched to position B at the end of the input bit sequence for termination of trellis. The XOR of the bit with itself will be zero (output of left most XOR) and thus the encoder will return to all zero state

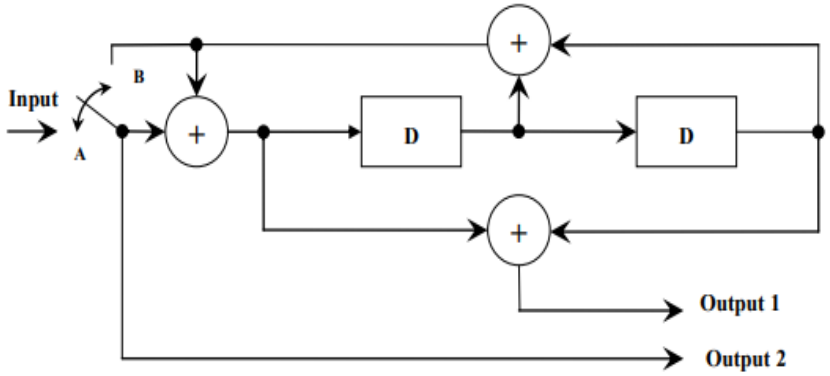


Figure 2.13-3 The trellis termination strategy for RSC encoder.

- **QPP INTERLEAVER**

QPP-based inter-leaver is maximum contention free, implying that the decoding can be parallelized without the risk for contention when different parallel processes are accessing the inter-leaver memory.

2.13.2. TURBO DECODING

Turbo decoding operates on the noisy version of systematic bits and two sets of parity-check bits in two decoding stages to produce estimate of the original transmitted bits.
Each of the two decoding stages uses a SISO decoder to solve the MAP detection problem.
in the following figure the basic structure of the turbo decoder is shown:

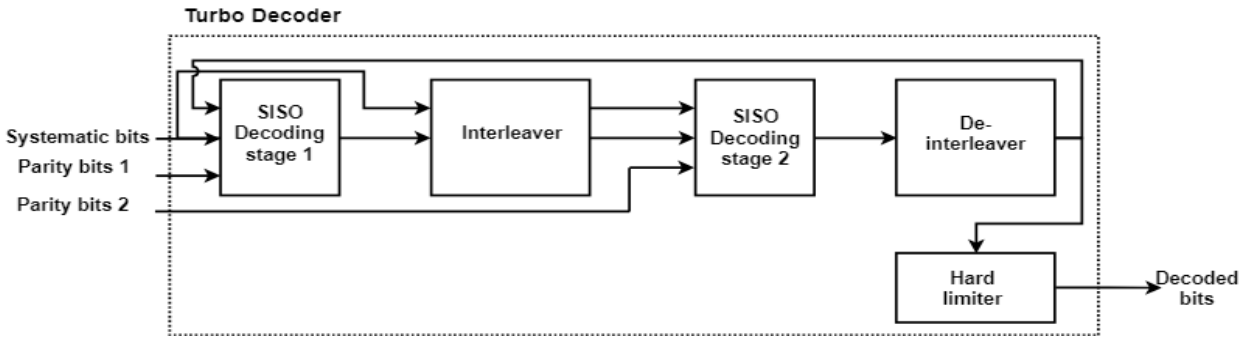


Figure 2.13-4 Turbo Decoder

(1) SISO decoding stages
The SISO decoding stages could be implemented by:
A) BCJR algorithm. B) Modified Viterbi algorithm.
(2) Inter-leaver/de-interleaver
the interleaver used here is a QPP interleaver according to the standard specification.
(3) Hard limiter
to convert the soft output into hard bits.

BCJR	VITERBI
Soft input soft output	The original Viterbi is a soft input hard output algorithm, however it may be modified to be soft input soft output
Has two recursions, one forward and the other backward, both of which involves soft decisions	Has a single forward recursion involves soft decision.
is a MAP decoder that minimize bit error rate by estimating probability of individual bits	is a maximum likelihood sequence estimator that maximize the likelihood function for the whole sequence not each bit
More Complex	Less complex
Has lower BER	Has higher BER
needs a hard limiter to transform its soft output into bits.	the original Viterbi does not need hard limiter as its output is hard bits.

Table 2.13-1 BCJR vs Viterbi

After showing the comparison between the BCJR algorithm and the modified Viterbi algorithm, the BCJR is chosen for the SISO decoding stage to achieve the minimum BER.

BCJR:

BCJR algorithm is an algorithm for maximum a posteriori decoding for error correction codes defined on trellises (principally convolutional codes) to implement the MAP decoder.
The LLR (log likelihood ratio) is defined as:

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

This could be written using joint probability of the received sequence as:

$$L(u_k|y) = ln\left(\frac{\sum_{R1} P(s', s, y)}{\sum_{R0} P(s', s, y)}\right) = ln\left(\frac{\sum_{R1} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}{\sum_{R0} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}\right)$$

Where:

- P (s', s, y) represents the joint probability of receiving the N-bit sequence.
- y the received N-bit sequence.
- s' is the state at time k-1.

- S is the state at time k.
- α is the forward recursion function.
- γ is the branch matric.
- β is the backward recursion function.

The BCJR algorithm contains log operation which is very complex to implement, so some simplified methods are used to calculate LLR thus implement MAP algorithm. In the next section two of those simplified methods will be discussed.

Three new variables will be defined A, B and Γ.

$$\Gamma_k(s',s) = ln\big(\gamma_k(s',s)\big) = ln\, C_k + \frac{u_k L(u_k)}{2} + \frac{L_c}{2}\sum_{l=1}^{n} x_{lk} y_{lk}$$

$$A_k(s) = ln\big(\alpha_k(s)\big) = max_{s'}[A_{k-1}(s') + \Gamma_k(s',s)]$$

$$B_{k-1}(s') = ln\big(\beta_{k-1}(s')\big) = max_s[B_k(s) + \Gamma_k(s',s)]$$

$$max(a,b)\begin{cases} max(a,b) + ln\big(1 + e^{-|a-b|}\big) & log - MAP\ algorithm \\ max(a,b) & max\ log - MAP\ algorithm \end{cases}$$

$$L(u|y) = \max_{R1}[A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)] - \max_{R0}[A_{k-1}(s') + \Gamma_k(s',s) + B_k(s)]$$

Max log-MAP algorithm will be used to implement the turbo decoding BCJR algorithm.

2.14. Cyclic Redundancy Check (CRC)

The function of this block is to detect the errors in the whole transport block by adding 24 redundancy check bits at the end of each transport block at transmitter, while at receiver recalculated 24 bit CRC is compared with last 24 bit.

Chapter 3
Standard Specifications and Assumptions

In this Chapter, the standard specifications and any assumptions taken for each block in the chain are stated according to(3GPP TS 36)

3.1. FFT

In communication systems FFT block is used to convert data from time domain to frequency domain in order to be processed in the whole chain. FFT is used for OFDM and SC-FDMA chains to generate the subcarriers needed to be transmitted in Tx and to recover the symbols again in Rx. SC-FDMA is the modified version of OFDM, it is the same efficiency and complexity but low Peak to Average Power Ratio due to the presence of IDFT and FFT with each other so it is used in Uplink due to the low power of the user equipment. In Downlink OFDM is used because base station operates at high power so power is not a constraint.

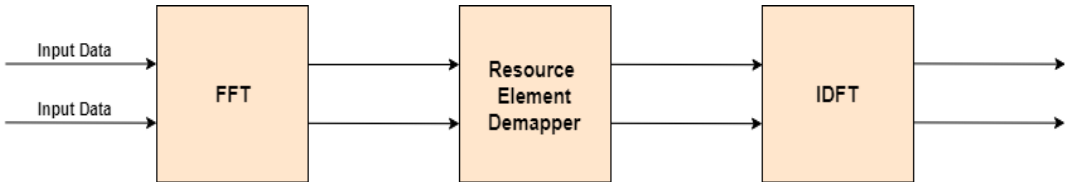


Figure 3.1-1 SC-FDMA chain

- **Assumptions**

LTE in general uses resource blocks; each block is 180 KHz to transmit at high rate (usually in Mbps). Each RB is 15 KHz or 3.75 KHz sub-carrier spacing as mentioned in the 3GPP standard. In Narrow Band LTE we have only one RB per slot due to low power constraint, so the BW is 180 KHz and we use 15KHz. 15 KHz lead us to 12 sub-carriers so we choose the size of the FFT to be 16-bits to support the 12 sub-carriers. Symbol duration is based on the sub-carrier spacing; we can calculate it by knowing the duration of the whole frame of the NB-LTE. Frame duration is 1ms long and it consists of 10 sub-frames, each sub-frame consists of 2 time slots each one is 0.5ms long. Each time slot is 7 SC-FDMA symbols in case of normal Cyclic Prefix and 6 in case of extended Cyclic Prefix and we operate in normal Cyclic Prefix for simplicity. So, 0.5ms over 7 symbols gives symbol duration of 66.7µs and a CP of 5.08µs for first symbol and 4.67µs for the rest of symbols. Each one symbol SC-FDMA consists of 12 symbols so the whole RB is 84 symbols, it gives a data rate of 336Kbps.

3.2. RED

Complex valued symbols $z(0), \ldots, z(M_{symb}^{NPUSCH} - 1)$ which are FFT block outputs shall mapped to a resource element $a_{k,l}$ increasing k (symbols) then increasing l (subcarrier) until reaching N_{slots} symbols, the N_{slots} symbols will be repeated $M_{identical}^{NPUSCH} - 1$ times before continuing mapping as stated in [1].

Where,

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

$$N_{slots} = \begin{cases} 1 & \Delta f = 3.75\text{kHz} \\ 2 & \Delta f = 15\text{kHz} \end{cases}$$

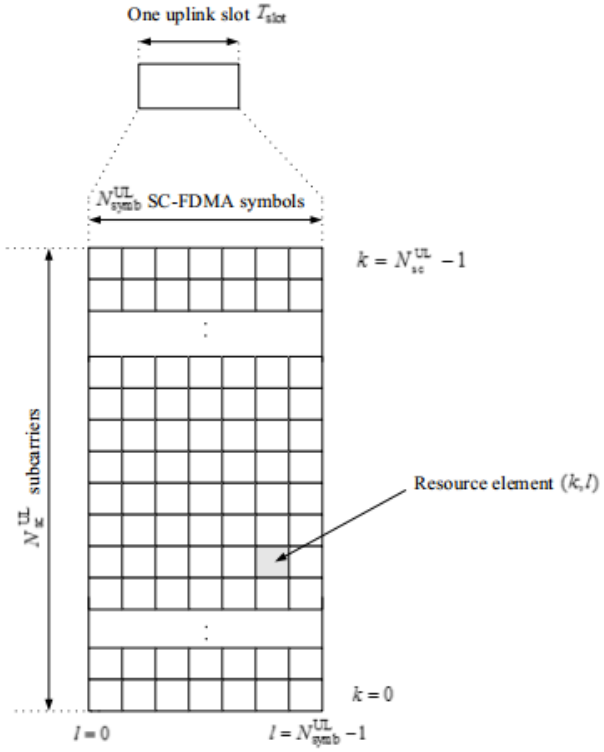


Figure 3.2-1 Resource element (k,l) in the resource grid

The resource allocation information in uplink DCI format N0 for NPUSCH transmission indicates to a scheduled UE as stated in [3].

- A set of contiguously allocated subcarriers (n_{sc}) of a resource unit determined by the Subcarrier indication field in the corresponding DCI,
- A number of resource units (N_{RU}) determined by the resource assignment field in the corresponding DCI according to Table 3.2-1.
- A repetition number (N_{REP}) determined by the repetition number field in the corresponding DCI according to Table 3.2-2.

For NPUSCH transmission with subcarrier spacing $\Delta f = 15$ kHz, the subcarrier indication field (I_{sc}) in the DCI determines the set of contiguously allocated subcarriers (n_{sc}) according to Table 3.2-3.

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

Table 3.2-1 Number of resource units (N_{RU}) for NPUSCH

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

Table 3.2-2 Number of repetitions (N_{Rep}) for NPUSCH

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

Table 3.2-3 Set of allocated sub-carriers

In this design the carrier spacing is assumed to be 15 kHz with no repetition which is left for future work.

3.3. Channel estimation

Standard supports different number of Demodulation reference signals (pilots)
1, 3,6and 12 each has its own parameter to be generated

Demodulation reference signal

- The reference signal sequence $\bar{r}_u(n)$ for $N_{sc}^{RU} = 1$ is defined by

$$\bar{r}_u(n) = \frac{1}{\sqrt{2}}(1+j)(1-2c(n))w(n \bmod 16), \quad 0 \le n < M_{rep}^{NPUSCH} N_{slots}^{UL} N_{RU}$$

Where the binary sequence $c(n)$ shall be initialized with $c_{init} = 35$ at the start of the NPUSCH transmission. Where $u = N_{ID}^{Ncell} \bmod 16$ for NPUSCH format 1if group hopping is not enabled.

u	$w(0),\ldots,w(15)$															
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1	1	-1
2	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1
3	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1	1	-1	-1	1
4	1	1	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1
5	1	-1	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1
6	1	1	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1	1	1
7	1	-1	-1	1	-1	1	1	-1	1	-1	-1	1	-1	1	1	-1
8	1	1	1	1	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1
9	1	-1	1	-1	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1
10	1	1	-1	-1	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1
11	1	-1	-1	1	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1
12	1	1	1	1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	1	1
13	1	-1	1	-1	-1	1	-1	1	-1	1	-1	1	1	-1	1	-1
14	1	1	-1	-1	-1	-1	1	1	-1	-1	1	1	1	1	-1	-1
15	1	-1	-1	1	-1	1	1	-1	-1	1	1	-1	1	-1	-1	1

Table 3.3-1 Definition of W (n)

The reference signal sequence for NPUSCH format 1 is given by:

$$r_u(n) = \bar{r}_u(n)$$

The reference signal sequence for NPUSCH format 2 is given by

$$r_u(3n+m) = \bar{w}(m)\bar{r}_u(n), \quad m = 0,1,2$$

Where the sequence index chosen according to $\left(\sum_{i=0}^{7} c(8n_s+i)2^i\right) \bmod 3$ with $c_{\text{init}} = N_{\text{ID}}^{\text{Ncell}}$.

Reference signal sequence for $N_{\text{sc}}^{\text{RU}} > 1$

The reference signal sequences $r_u(n)$ for $N_{\text{sc}}^{\text{RU}} > 1$ is defined by a cyclic shift α of a base sequence according to

$$r_u(n) = e^{j\alpha n} e^{j\phi(n)\pi/4}, \quad 0 \le n < N_{\text{sc}}^{\text{RU}},$$

If group hopping is not enabled, the base sequence index u is given by higher layer parameters threeTone-BaseSequence, sixTone-BaseSequence, and twelveTone-BaseSequence for $N_{\text{sc}}^{\text{RU}} = 3$, $N_{\text{sc}}^{\text{RU}} = 6$, and $N_{\text{sc}}^{\text{RU}} = 12$, respectively. If not signalled by higher layers, the base sequence is given by

$$u = \begin{cases} N_{\text{ID}}^{\text{Ncell}} \bmod 12 & \text{for } N_{\text{sc}}^{\text{RU}} = 3 \\ N_{\text{ID}}^{\text{Ncell}} \bmod 14 & \text{for } N_{\text{sc}}^{\text{RU}} = 6 \\ N_{\text{ID}}^{\text{Ncell}} \bmod 30 & \text{for } N_{\text{sc}}^{\text{RU}} = 12 \end{cases}$$

The cyclic shift α for $N_{\text{sc}}^{\text{RU}} = 3$ and $N_{\text{sc}}^{\text{RU}} = 6$ is derived from higher layer parameters threeTone-CyclicShift and sixTone-CyclicShift, respectively, as defined in Table below. For $N_{\text{sc}}^{\text{RU}} = 12$, $\alpha = 0$.

u	$\phi(0),\phi(1),\phi(2)$		
0	1	-3	-3
1	1	-3	-1
2	1	-3	3
3	1	-1	-1
4	1	-1	1
5	1	-1	3
6	1	1	-3
7	1	1	-1
8	1	1	3
9	1	3	-1
10	1	3	1
11	1	3	3

Table 3.3-2 Definition of ϕ (n) for Nsc =3

u	$\phi(0),\ldots,\phi(5)$					
0	1	1	1	1	3	-3
1	1	1	3	1	-3	3
2	1	-1	-1	-1	1	-3
3	1	-1	3	-3	-1	-1
4	1	3	1	-1	-1	3
5	1	-3	-3	1	3	1
6	-1	-1	1	-3	-3	-1
7	-1	-1	-1	3	-3	-1
8	3	-1	1	-3	-3	3
9	3	-1	3	-3	-1	1
10	3	-3	3	-1	3	3
11	-3	1	3	1	-3	-1
12	-3	1	-3	3	-3	-1
13	-3	3	-3	1	1	-3

Table 3.3-3 Definition of ϕ (n) for Nsc =6

u	$\varphi(0),\ldots,\varphi(11)$											
0	-1	1	3	-3	3	3	1	1	3	1	-3	3
1	1	1	3	3	3	-1	1	-3	-3	1	-3	3
2	1	1	-3	-3	-3	-1	-3	-3	1	-3	1	-1
3	-1	1	1	1	1	-1	-3	-3	1	-3	3	-1
4	-1	3	1	-1	1	-1	-3	-1	1	-1	1	3
5	1	-3	3	-1	-1	1	1	-1	-1	3	-3	1
6	-1	3	-3	-3	-3	3	1	-1	3	3	-3	1
7	-3	-1	-1	-1	1	-3	3	-1	1	-3	3	1
8	1	-3	3	1	-1	-1	-1	1	1	3	-1	1
9	1	-3	-1	3	3	-1	-3	1	1	1	1	1
10	-1	3	-1	1	1	-3	-3	-1	-3	-3	3	-1
11	3	1	-1	-1	3	3	-3	1	3	1	3	3
12	1	-3	1	1	-3	1	1	1	-3	-3	-3	1
13	3	3	-3	3	-3	1	1	3	-1	-3	3	3
14	-3	1	-1	-3	-1	3	1	3	3	3	-1	1
15	3	-1	1	-3	-1	-1	1	1	3	1	-1	-3
16	1	3	1	-1	1	3	3	3	-1	-1	3	-1
17	-3	1	1	3	-3	3	-3	-3	3	1	3	-1
18	-3	3	1	1	-3	1	-3	-3	-1	-1	1	-3
19	-1	3	1	3	1	-1	-1	3	-3	-1	-3	-1
20	-1	-3	1	1	1	1	3	1	-1	1	-3	-1
21	-1	3	-1	1	-3	-3	-3	-3	-3	1	-1	-3
22	1	1	-3	-3	-3	-3	-1	3	-3	1	-3	3
23	1	1	-1	-3	-1	-3	1	-1	1	3	-1	1
24	1	1	3	1	3	3	-1	1	-1	-3	-3	1
25	1	-3	3	3	1	3	3	1	-3	-1	-1	3
26	1	3	-3	-3	3	-3	1	-1	-1	3	-1	-3
27	-3	-1	-3	-1	-3	3	1	-1	1	3	-3	-3
28	-1	3	-3	3	-1	3	3	-3	3	3	-1	-1
29	3	-3	-3	-1	-1	-3	-1	3	-3	3	1	-1

Table 3.3-4 Definition of Definition of ϕ (n) for Msc = Nsc

$N_{sc}^{RU}=3$		$N_{sc}^{RU}=6$	
threeTone-CyclicShift	α	sixTone-CyclicShift	α
0	0	0	0
1	$2\pi/3$	1	$2\pi/6$
2	$4\pi/3$	2	$4\pi/6$
		3	$8\pi/6$

Table 3.3-5 Definition of α

3.4. IDFT

For each symbol $x(0), x(1), \dots, x(M_{symb}^{layer}-1)$ will be divided into $M_{symb}^{layer}/M_{sc}^{NPUSCH}$ sets each set represents a SC-FDMA pre-coded by the following equation

$$y(l \cdot M_{sc}^{NPUSCH}+k)=\frac{1}{\sqrt{M_{sc}^{NPUSCH}}}\sum_{i=0}^{M_{sc}^{NPUSCH}-1} x(l \cdot M_{sc}^{NPUSCH}+i)e^{-j\frac{2\pi ik}{M_{sc}^{NPUSCH}}}$$

$$k=0,\dots,M_{sc}^{NPUSCH}-1\ (idft\ points\ depend\ on\ M_{sc}^{NPUSCH})$$

$$l=0,\dots,\frac{M_{symb}^{layer}}{M_{sc}^{NPUSCH}}-1$$

Resulting in a block of complex-valued symbols $y(0), \dots, y(M_{symb}^{layer}-1)$.
Where $M_{sc}^{NPUSCH}=N_{sc}^{RU}$ and, N_{sc}^{RU} is the number of allocated subcarriers can be determined from the Table 3.9.

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

Table 3.4-1 Number of subcarriers per RU

Resource units are used to describe the mapping of the NPUSCH to resource elements. A resource unit is defined as $N_{slots}^{UL} * N_{symbols}^{UL}$ consecutive SC-FDMA symbols in the time domain and, N_{sc}^{RU} consecutive subcarriers in the frequency domain, where N_{slots}^{UL}, $N_{symbols}^{UL}$, and N_{sc}^{RU}, are given by Table 3-9.
In our design we assumed working with NPUSCH format 1 with 15 KHz spacing.

3.5. Mapper

According to 3GPP Narrow band LTE standard Release 14 the modulation scheme used for uplink is either BPSK or QPSK depends on the channel condition to minimize the bit error rate (BER).

- BPSK:

Modulating a single bit into a symbol and map it to the constellation according to the following Table:

b(i)	I	Q
0	1/√2	1/√2
1	-1/√2	-1/√2

Table 3.5-1 BPSK real and imaginary values

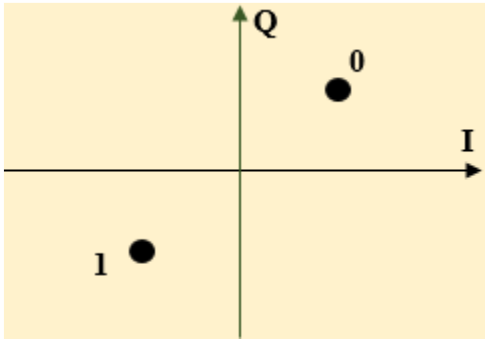


Figure 3.5-1 BPSK constellation

- QPSK:

Modulating 2 bits to a symbol and map it to the constellation according to the following Table:

b(i),b(i+1)	I	Q
00	1/√2	1/√2
01	1/√2	-1/√2
10	-1/√2	1/√2
11	-1/√2	-1/√2

Table 3.5-2 QPSK real and imaginary values

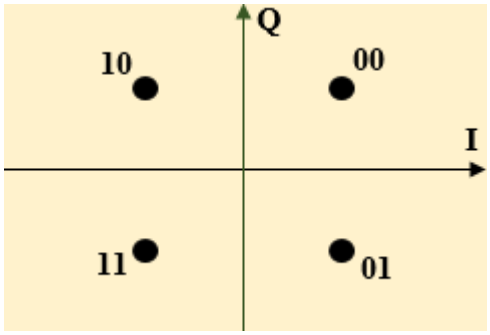


Figure 3.5-2 QPSK constellation

3.5.1. Clock Domain Crossing

A clock domain crossing occurs whenever data is transferred from a flip-flop driven by one clock to a flip-flop driven by another clock. This will occur if there is a change in phase, frequency or both of them. Leading to a meta-stability region between the 2 clock domains.

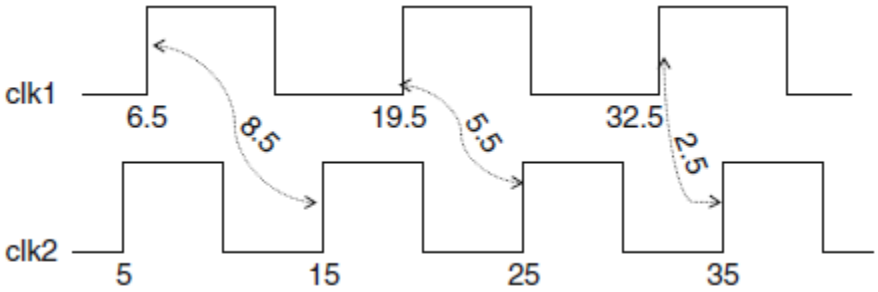


Figure 3.5-3 Example of 2 clock domains

Meta-Stability:

One of the biggest problems due to CDC is meta-stability because the time available for a signal keeps changing for each edge pair. Hence, even if the timing is met for some specific pairs of clock edges, there is quite a likelihood that for some other pair of edges, the setup or hold might not be met. Also, the extent of violation would keep varying across different pairs. Thus, sometimes the setup and hold requirements would be met with sufficient slack, sometimes they would be just met, sometimes they would be just violated and sometimes, there would be gross violations.

Figure 3.5-4 Meta-stability region

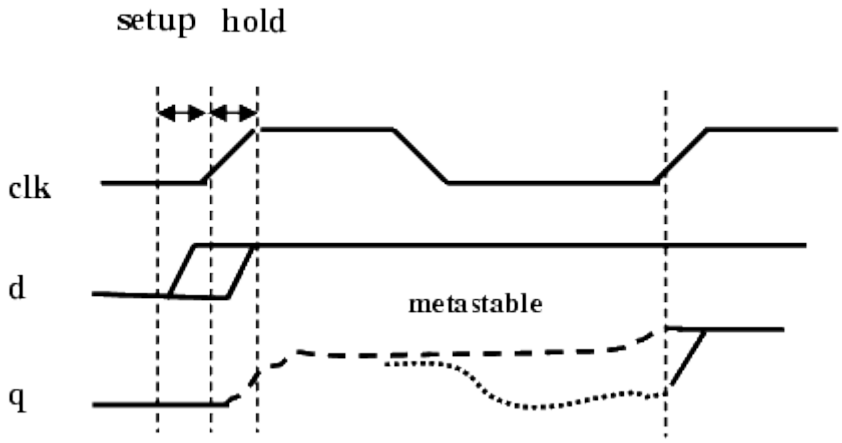


Figure 3.5-5 Meta-stability effect

Problems of Meta-stability region:

- Setup and hold time violations which lead to data corruption.
- Short circuit path is established between supply and ground if an input to a set of CMOS transistors is some way between '0' and '1'.
- Error propagation along the receiver chain.

Proposed solution to Clock Domain Crossing (CDC) problem:

- Synchronizer (two flip-flops).
- RAM
- FIFO

3.5.1.1. Synchronizer

Synchronization is the solution to the above problem as the main responsibility of a synchronizer is to allow sufficient time such that any meta-sable output can settle down to a stable value in the destination clock domain. The most common synchronizer used by designers is Double Flop (2-FF) synchronizers.

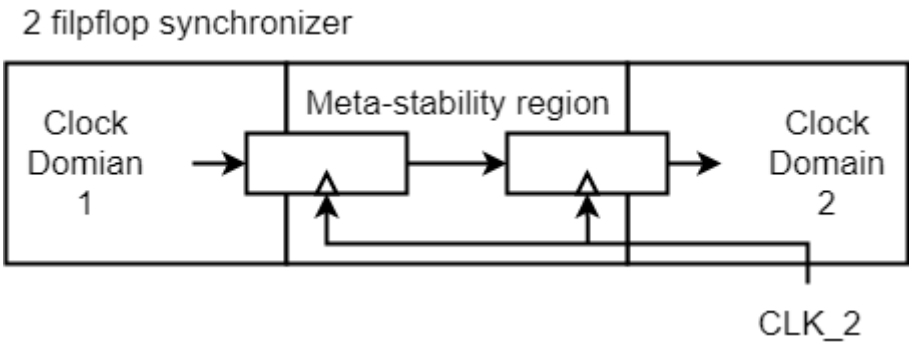


Figure 3.5-6 Two flip-flop synchronizers

The main problem of the synchronizer is the data loss due to the change in frequency between the 2 clock domains.

Let’s consider 2 scenarios for the data loss problem:

A. Slow to fast crossing:

Considering a situation, where the source flop is generating the data at a lower frequency. And, the destination flop is getting triggered by a faster clock. In this case, before the source flop generates another data, the destination flop would have sampled the previous data. Thus, for slow to fast crossing, there might not be a risk of data loss. However, if the destination clock is only marginally faster than the source clock, the data loss risk would still be there. This happens because once the edges of the two clocks are almost aligned they will come very close together for next several cycles.

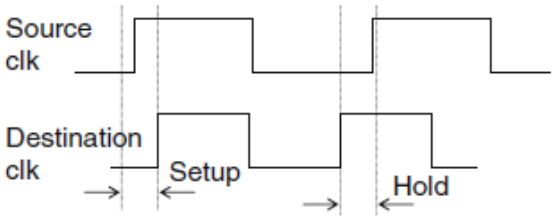


Figure 3.5-7 Setup and hold time violations due to CDC

to ensure no loss of data, it is required to take care of all kinds of uncertainty. The factors that cause this uncertainty are:

- Setup requirement of the destination flop.
- Hold requirement of the destination flop.
- Clock Jitter for both launch and capture clock.
- Path delay differential for fastest and slowest path from the launch to the capture.

B. Fast to slow crossing:

Consider a situation, where the source flop is generating the data at a higher frequency. And, the destination flop is getting triggered by a slower clock. In this case, before the destination flop captures a data, the source clock would have launched the next data. Thus, for fast to slow crossing, there is always a risk that only intermediate data might get captured, and, several data might get lost.

3.5.1.2. FIFO

Data Loss can be prevented using FIFO (First in First out) based mechanism. FIFO based mechanism is very useful for either of the two situations:

- The two clocks have very close frequency.
- The data launch is in bursts, i.e. after launching several data in consecutive cycles, it then becomes quiet for several cycles. In such cases, even if the launch clock is faster than the capture clock, the FIFO based mechanism can be found to be useful. The data launched in the bursts keep getting stored in the FIFO. While, the launch side becomes quiet, the capture side keeps picking up the data stored in the FIFO.

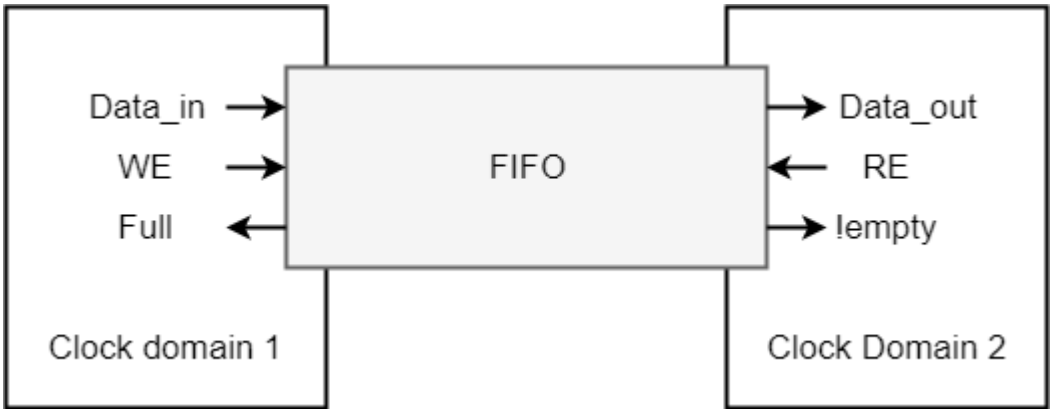


Figure 3.5-8 FIFO as a solution to CDC

3.5.1.3. FIFO structure

The FIFO is a 2 port memory which input and the output bit stream is in the same sequence. However, the address of this memory cannot be accessed from its interfaces. The data is entered or read each clock cycle using Read and Write enables. It also has 2 signals to indicate that either the FIFO is full or empty to prevent writing or reading from it if there is no free memory to write or no data to read.

3.5.1.4. FIFO Depth

Size of the FIFO basically refers to the amount of data available at a given time. In asynchronous FIFO this depends on both read and write clock domain frequencies and number of data written and read (data rate). Data rate can vary depending on the two-clock domain operation and requirement and of course frequency. The worst case condition is the maximum data rate difference between read and write clock. This can happen when data rate of writing operation is maximum and for read operation data rate is minimum.

$$FIFO\ Size = D_{max} - \frac{D_{max} * F_{read} * D_{read}}{F_{write} * D_{write}}$$

Where:

- D_{max} is the maximum number of words that could be written.
- F_{read} is the read frequency.
- F_{write} is the write frequency.
- D_{read} is the number of words that is read each clock cycle.
- D_{write} is the number of words that is written each clock cycle.

In the case of NB-LTE uplink receiver the FIFO based solution is used.

3.6. Scrambler

Scrambler mainly consists of two linear feedback shift registers which simply generating an L=31- Golden Sequence C(n) by 2 paths of flip flops which initialized by two different values as shown in figure 3.6-1.

For each code word q , the block of bits $b^{(q)}(0),...,b^{(q)}(M_{\text{bit}}^{(q)}-1)$, where $M_{\text{bit}}^{(q)}$ is the number of bits transmitted in code word q on the physical uplink shared channel in one sub frame, shall be scrambled with a UE-specific scrambling sequence prior to modulation, resulting in a block of scrambled bits $\tilde{b}^{(q)}(0),...,\tilde{b}^{(q)}(M_{\text{bit}}^{(q)}-1)$.

$c^{(q)}(i)$ Is the scrambling sequence and to get the required output, the first LFSR shall be initialized with$x_1(0) = 1, x_1(n) = 0$, n=1,2,….30.

While the second LFSR shall be initialized with

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

Where n_{RNTI} corresponds to the RNTI associated with the PUSCH transmission.

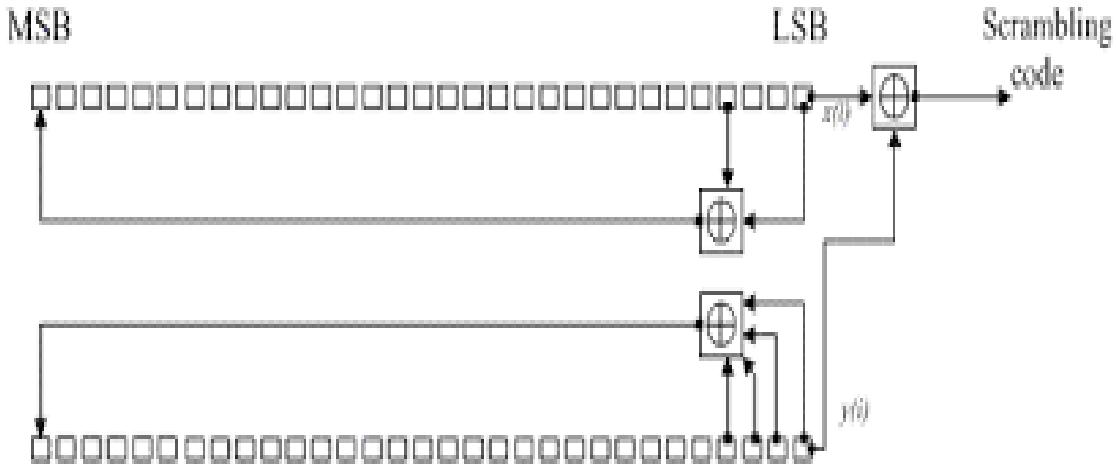


Figure 3.6-1 Scrambler architecture according to standard

3.7. Data multiplexing and Channel Interleaver

In narrowband interleaving is applied per resource unit without any control information in order to apply a time-first rather than frequency-first mapping, where the input sequence is the portion of e for a resource unit instead of f

1. Data and control multiplexing:

The inputs to the data and control multiplexing are the coded bits of the control information denoted by $q_0, q_1, q_2, q_3, ..., q_{N_L \cdot Q_{CQI}-1}$ and the coded bits of the UL-SCH denoted by $f_0, f_1, f_2, f_3, ..., f_{G-1}$. The output of the data and control multiplexing operation is denoted by $\underline{g}_0, \underline{g}_1, \underline{g}_2, \underline{g}_3, ..., \underline{g}_{H'-1}$, where $H = (G + N_L \cdot Q_{CQI})$ and $H' = H/(N_L \cdot Q_m)$, and where $\underline{g}_i$, $i = 0,...,H'-1$ are column vectors of length $(Q_m \cdot N_L)$. H is the total number of coded bits allocated for UL-SCH data and CQI/PMI information across the N_L transmission layers of the transport block.
In narrowband Qm is 1 for π/2-BPSK and 2 for π/4-QPSK, NL=1.
We will skip the control part so Q_{cQi}will be take value Zero.

in the case of single transport block transmission, and assuming that N_L is the number of layers onto which the UL-SCH transport block is mapped, the control information and the data shall be multiplexed as follows:
Set i, j, k to 0

While $i < G$ -- place the data
$\underline{g}_k = [\, f_i \cdots f_{i+Q_m \cdot N_L - 1}]^T$
$i = i + Q_m \cdot N_L$
$k = k + 1$
end while

2. Channel interleaver

The input to the channel interleaver are denoted by $\underline{g}_0, \underline{g}_1, \underline{g}_2, \ldots, \underline{g}_{H'-1}$, $\underline{q}_0^{RI}, \underline{q}_1^{RI}, \underline{q}_2^{RI}, \ldots, \underline{q}_{Q'_{RI}-1}^{RI}$ and $\underline{q}_0^{ACK}, \underline{q}_1^{ACK}, \underline{q}_2^{ACK}, \ldots, \underline{q}_{Q'_{ACK}-1}^{ACK}$. In case where more than one UL-SCH transport block are transmitted in a sub-frame of an UL cell, the HARQ-ACK and RI information are multiplexed with data on both UL-SCH transport blocks. The number of modulation symbols per layer in the sub-frame is given by $H'_{total} = H' + Q'_{RI}$. The output bit sequence from the channel interleaver is derived as follows:

(1) Assign $C_{mux} = \left(N_{\text{symb}}^{\text{UL}} - 1\right) N_{\text{slots}}^{\text{UL}}$ to be the number of columns of the matrix. The columns of the matrix are numbered 0, 1, 2,…, $C_{mux} - 1$ from left to right. $N_{\text{symb}}^{\text{UL}}$ is the number of SC-FDMA symbols for NPUSCH in a UL resource unit.

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

(3) Write the input vector sequence, for k = 0, 1,…, $H' - 1$, into the $\left(R_{mux} \times C_{mux}\right)$ matrix by sets of $\left(Q_m \cdot N_L\right)$ rows starting with the vector $\underline{y}_0$ in column 0 and rows 0 to $\left(Q_m \cdot N_L - 1\right)$:

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

The pseudocode is as follows:
Set i, k to 0.
While k < H',

if $\underline{y}_i$ is not assigned to RI symbols

$\underline{y}_i = \underline{g}_k$

k = k + 1

end if

i = i+1

end while

(4) The output of the block interleaver is the bit sequence read out column by column from the $(R_{mux} \times C_{mux})$ matrix. The bits after channel interleaving are denoted by $h_0, h_1, h_2, ..., h_{H+N_L \cdot Q_{RI}-1}$.

3.8. Rate De-Matching

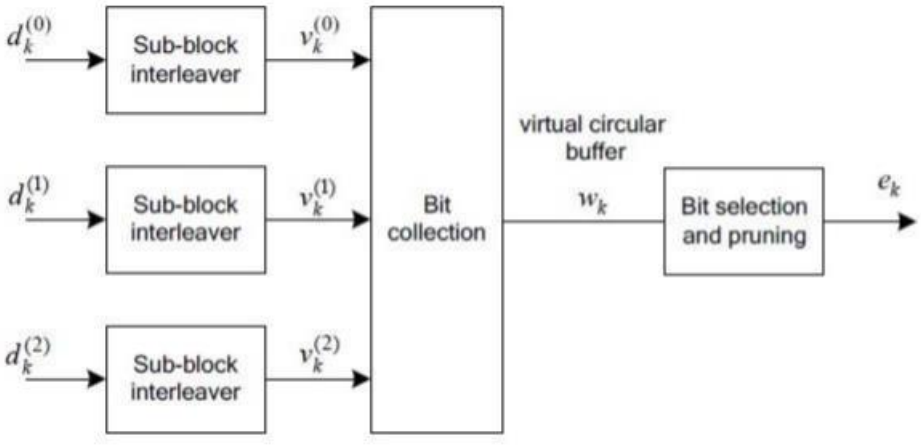


Figure 3.8-1 Rate matching Block diagram

The Block is divided into the main blocks

1. Sub-block interleaver: is defined for each output stream from Turbo coding. The streams include a systematic bit stream dk (0), a parity bit stream dk (1) and an interleaved parity stream dk (2) coming from turbo-encoder.

The bit stream $d_k^{(0)}$ is interleaved according to the first sub-block interleaver with an output sequence defined as $v_0^{(0)}, v_1^{(0)}, v_2^{(0)}, ..., v_{K_\Pi-1}^{(0)}$ and where $K_\Pi = \left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right)$

The bit stream $d_k^{(1)}$ is interleaved according to the second sub-block interleaver with an output sequence defined as $v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, ..., v_{K_\Pi-1}^{(1)}$.

The bit stream $d_k^{(2)}$ is interleaved according to the third sub-block interleaver with an output sequence defined as $v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, ..., v_{K_\Pi-1}^{(2)}$.

The bits input to the block interleaver are denoted by $d_0^{(i)}, d_1^{(i)}, d_2^{(i)}, ..., d_{D-1}^{(i)}$, where D is the number of bits. The output bit sequence from the block interleaver is derived as follows:

(1) Assign $C_{subblock}^{TC} = 32$ to be the number of columns of the matrix. The columns of the matrix are numbered 0, 1, 2,…, $C_{subblock}^{TC} - 1$ from left to right.

(2) Determine the number of rows of the matrix $R_{subblock}^{TC}$, by finding minimum integer $R_{subblock}^{TC}$ such that:

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

The rows of rectangular matrix are numbered 0, 1, 2,…, $R_{subblock}^{TC} - 1$ from top to bottom.

(3) If $\left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right) > D$, then $N_D = \left(R_{subblock}^{TC} \times C_{subblock}^{TC} - D\right)$ dummy bits are padded such that y_k = <NULL> for k = 0, 1,…, N_D - 1. Then, $y_{N_D+k} = d_k^{(i)}$, k = 0, 1,…, D-1, and the bit sequence y_k is written into the $\left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right)$ matrix row by row starting with bit y_0 in column 0 of row 0:

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

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

(4) Perform the inter-column permutation for the matrix based on the pattern $\langle P(j) \rangle_{j \in \{0,1,\ldots C_{subblock}^{TC}-1\}}$ that is shown in Table 3.8-1, where P(j) is the original column position of the j-th permuted column. After permutation of the columns, the inter-column permuted $\left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right)$ matrix is equal to

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

(5) The output of the block interleaver is the bit sequence read out column by column from the inter-column permuted $\left(R_{subblock}^{TC} \times C_{subblock}^{TC}\right)$ matrix. The bits after sub-block interleaving are denoted by $v_0^{(i)}, v_1^{(i)}, v_2^{(i)}, \ldots, v_{K_\Pi-1}^{(i)}$, where $v_0^{(i)}$ corresponds to $y_{P(0)}$, $v_1^{(i)}$ to $y_{P(0)+C_{subblock}^{TC}}$ … .

For $d_k^{(2)}$:

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

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

The permutation function P is defined in Table 3.8-1.

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

Table 3.8-1 Inter-column permutation pattern for sub-block interleave

2. Bit collection: is used to concatenate the three bit streams $vk\pi$ (0) , $vk\pi$ (1) and vk(2) which represent the systematic bit stream, parity bit stream and interleaved parity stream respectively together at the circular buffer.

(1) The circular buffer of length $K_w = 3K_\Pi$ for the r-th coded block is generated as follows:

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

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

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

3. Bit selection: extracts consecutive bits from the circular buffer to the extent that fits into the assigned physical resource. Combined with the Turbo coding, the circular buffer can puncture or repeat the collected coded bits to achieve an alterable channel coding rate under different scenarios.

Denoting by E the rate matching output sequence length for the r-th coded block, and rv_{idx} the redundancy version number for this transmission (rv_{idx} = 0, 1, 2 or 3), the rate matching output bit sequence is e_k , k = 0,1,..., $E-1$.

Define by G the total number of bits available for the transmission of one transport block.

Set $G' = G/(N_L \cdot Q_m)$

N_L is equal to the number of layers a transport block is mapped onto

Set $\gamma = G' \bmod C$, where C is the number of code blocks

if $r \le C - \gamma - 1$

set $E = N_L \cdot Q_m \cdot \lfloor G'/C \rfloor$

else

set $E = N_L \cdot Q_m \cdot \lceil G'/C \rceil$

end if

In narrowband C=1

Set $k_0 = R_{subblock}^{TC} \cdot \left(2 \cdot \left\lceil \frac{N_{cb}}{8R_{subblock}^{TC}} \right\rceil \cdot rv_{idx} + 2 \right)$, where $R_{subblock}^{TC}$ the number of rows is

Set k = 0 and j = 0

while {k < E}

if $w_{(k_0+j) \bmod N_{cb}} \neq <NULL>$

$e_k = w_{(k_0+j) \bmod N_{cb}}$

k = k +1

end if

j = j +1

end while

$N_{cb} = K_w$ for UL-SCH

3.9. Encoder

According to 3GPP narrowband LTE IOT Standard release 14 for UL-SCH channel codding is turbo coding with rate 1/3 as show in Table 3.9-1

TrCH	Coding scheme	Coding rate
UL-SCH	Turbo coding	1/3
DL-SCH		
PCH		
MCH		
SL-SCH		
SL-DCH		
BCH	Tail biting convolutional coding	1/3
SL-BCH		

Table 3.9-1 Usage of channel coding scheme and coding rate

Turbo Encoder is consisting of two recursive Convolutional Encoder and one Turbo internal interleaver, the overall code rate is approximately r = 1/3. Figure 3.9-1 shows a 3GPP turbo encoder.

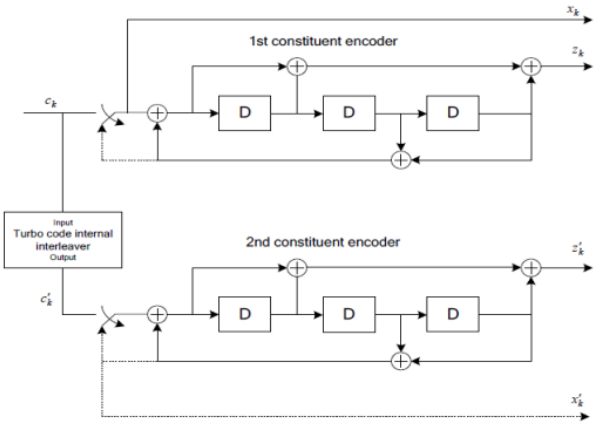


Figure 3.9-1 Structure of rate 1/3 turbo encoder (dotted lines apply for trellis termination only)

Convolutional Encoder

The two convolutional encoders used in the Turbo code are identical with generator polynomials

$$G(D) = \left[1, \frac{g1(D)}{g_0(D)}\right]$$

$$g_0(D) = 1 + D^2 + D^3$$

$$g_1(D) = 1 + D \quad + D^3$$

The data bits are transmitted together with the parity bits generated by two constituent convolutional encoders. Prior to encoding, both the convolutional encoders are set to all zero state, each shift register is filled with zeros. The turbo encoder consists of an internal inter-leaver which interleaves the input data bits c_1, c_2 …….c_k to c'_1, c'_2 ……. c'_k which are then input to the second constituent encoder. Thus, the data is encoded by the first encoder in the natural order and by the second encoder after being interleaved. The systematic output of the second encoder is not used and thus the output of the turbo coder is serialized combination of the systematic bits c_k, parity bits from the first (upper) encoder Z_K and parity bits from the second encoder Z'_k for k = 1, 2, …K.
After all the data bits K have been encoded, trellis termination is performed by passing tail bits from the constituent encoders bringing them to all zeros state. To achieve this, the switches in Figure 3.13 are moved in the down position. Because of the interleaver, the states of both the constituent encoders will usually be different, so the tail bits will also be different and need to be dealt separately.

For tails bits the transmitted bit stream includes not only the tail bits $\{X_{k+1}, X_{k+2}, X_{k+3}\}$ corresponding to the upper encoder but also tail bits corresponding to the lower encoder $\{X'_{k+1}, X'_{k+2}, X'_{k+3}\}$. In addition to these six tail bits, six corresponding parity bits $\{Z_{k+1}, Z_{k+2}, Z_{k+3}\}$ and $\{Z'_{k+1}, Z'_{k+2}, Z'_{k+3}\}$ for the upper and lower encoder respectively are also transmitted. First, the switch in the upper (first) encoder is brought to lower (flushing) position and then the switch in the lower (second) encoder. The tail bits are then transmitted at the end of the encoded data frame. According to 3GPP release 14 the tail bits' sequence is:

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

For number of input bits K, the total length of the encoded bit sequence now becomes 3K+12, 3K being the coded data bits and 12 being the tail bits. The code rate of the encoder is thus
r = K / (3K+12). However, for large size of input K, the fractional loss in code rate due to tail bits in negligible and thus, the code rate is approximated at 1/3.

Internal inter-leaver

The bits input to the turbo code internal inter-leaver are denoted by C_0, C_1, C_2, … C_{k-1}, where K is the number of input bits = TBS + 24 CRC bits.

The relationship between the output index (i) and the input index i according to the following equation is:

$$\pi(i) = (f1 * i + f2 * i^2)\% \, k$$

Where $f1$ and $f2$ are constants depending on K (TBS + 24 CRC bits), Table 3.9-2 shows standard values for $f1$ and $f2$ for every K, the size of the input data ranges from 40 to 2560 bits for NB-LTE.

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

Table 3.9-2 NB-LTE standard inter-leaver constants

3.10. Cyclic Redundancy Check (CRC)

According to Release 14 specifications, the parity bits are generated by one of the following cyclic generator polynomials:

- $g_{CRC24A}(D) = [D^{24} + D^{23} + D^{18} + D^{17} + D^{14} + D^{11} + D^{10} + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1]$ and;
- $g_{CRC24B}(D) = [D^{24} + D^{23} + D^6 + D^5 + D + 1]$ for a CRC length L = 24 and;
- $g_{CRC16}(D) = [D^{16} + D^{12} + D^5 + 1]$ for a CRC length L = 16.
- $g_{CRC8}(D) = [D^8 + D^7 + D^4 + D^3 + D + 1]$ for a CRC length of L = 8.

Where $g_{CRC24A}(D)$ is added after each code block, $g_{CRC24B}(D)$ is added only when code block segmentation is applied, g_{CRC16} is attached to the Master Information Block g_{CRC8} and is employed by (MIB) and Downlink Control Information (DCI) messages some uplink channels (PUCCH and PUSCH) for transmitting Channel Quality Indicator (CQI) information.

For NB-IOT maximum ULSH (uplink shared channel) with maximum TBS=2536 bits while the maximum transmitted code block size Z before segmentation =6144 therefore "g_{CRC24A}" polynomial equation is used.

Figure 3.10-1 shows transport block segmentation to code blocks and $g_{CRC24A}(D)$ is added to each code block.

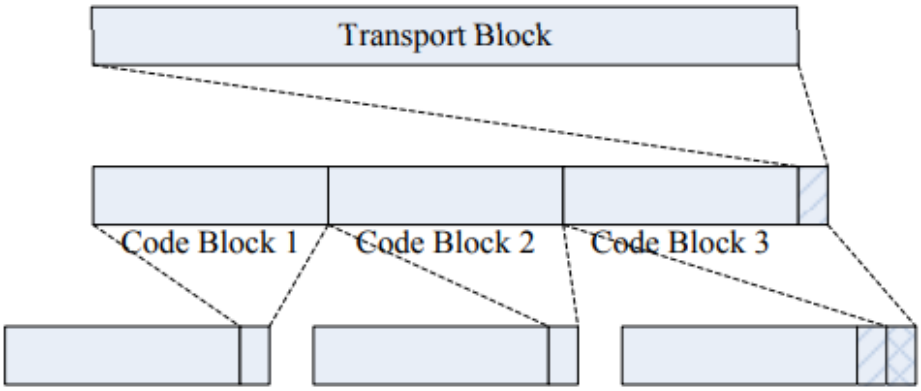


Figure 3.10-1 Block segment and CRC attached

3.11. Equalizer

In our project we split the equalizer block into two blocks, channel estimation block and equalization block. Equalization block in this case is very simple, it's just a divider. It reflects the channel effect by divide the symbols corrupted by the channel estimated through channel estimation block. As we mentioned we have a power and cost constraints so we use memory-based architecture to use one complex divider. Complex divider is done using restoring algorithm. Equalizer takes input symbols from resource element De-mapper block and the channel information from the channel estimation block and divides symbols by channel pilots to produce the correct symbols to IDFT block.

In LTE uplink grid as shown in Figure 3.11-1, pilots are generated in the 4th symbol in each slot so we save 1st three symbols in memory and wait for channel information from channel estimation. When receiving channel information in equalizer we saved it in memory and wait for other 3 symbols to be generated.

Symbols/Sub-carriers	Symbol1	Symbol2	Symbol3	Pilots	Symbol5	Symbol6	Symbol7
Sub-carrier1							
Sub-carrier2							
Sub-carrier3							
Sub-carrier4							
Sub-carrier5							
Sub-carrier6							
Sub-carrier7							
Sub-carrier8							
Sub-carrier9							
Sub-carrier10							
Sub-carrier11							
Sub-carrier12							

Figure 3.11-1 Uplink Grid for each slot in LTE

We assumed a stair case channel for equalization, we receive the channel in symbol 4 only so we can assume it a linear channel and give each symbol a factor depending on the space between the channel or we simply assumed that the whole slot sees the same channel (coherence time is bigger than the slot time) and this is the stair case approximation, the next slot will face another channel factor so we save the channel for each slot.

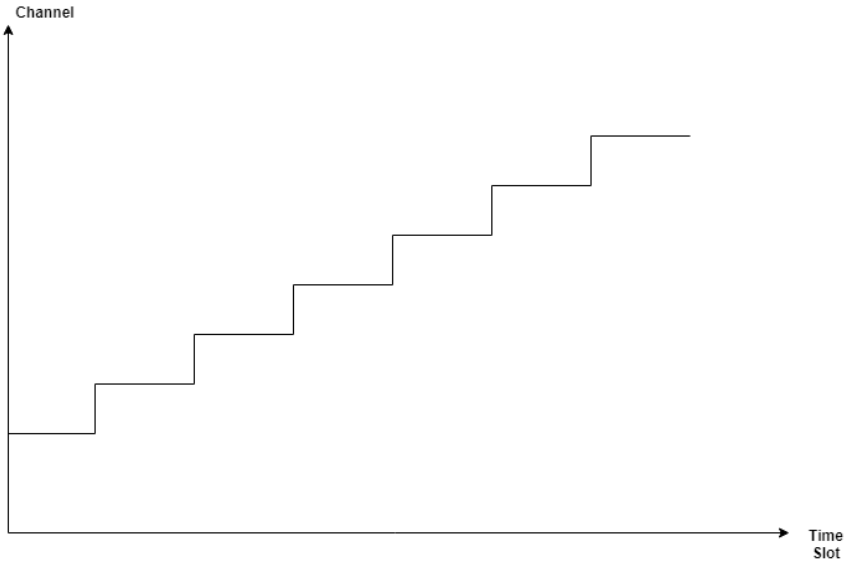


Figure 3.11-2 Stair case approximation for equalization process

Chapter 4
Design Architecture and interfaces

4.1. Synchronization (Time and frequency offset estimation)

4.1.1. Top level

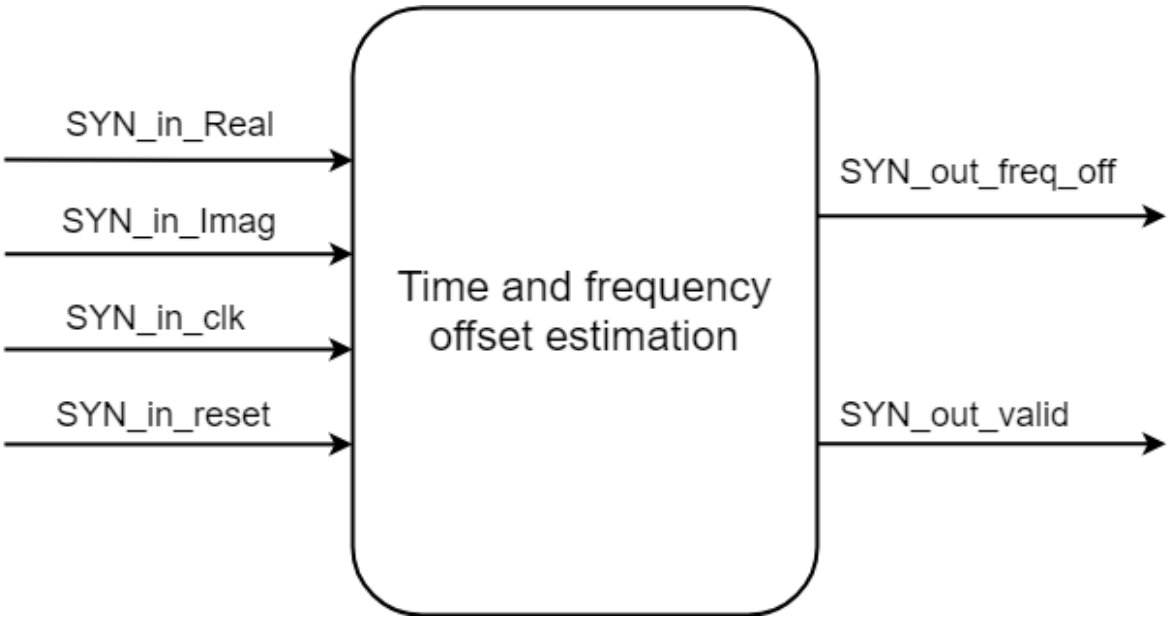


Figure 4.1-1 Time and frequency offset estimation Top Level

4.1.2. Block interface

Signal name	Direction	Description	Size (bits)
SYN_in_Real	input	Real input data	16
SYN_in_Imag	input	Imaginary input data	16
SYN_in_CLK	input	clock	1
SYN_in_reset	input	reset signal	1
SYN_out_freq_off	output	Estimated time offset	20
SYN_out_valid	output	Valid out to the correction block	1

Table 4.1-1 Time and frequency offset estimation interface signals

4.1.3. Architecture

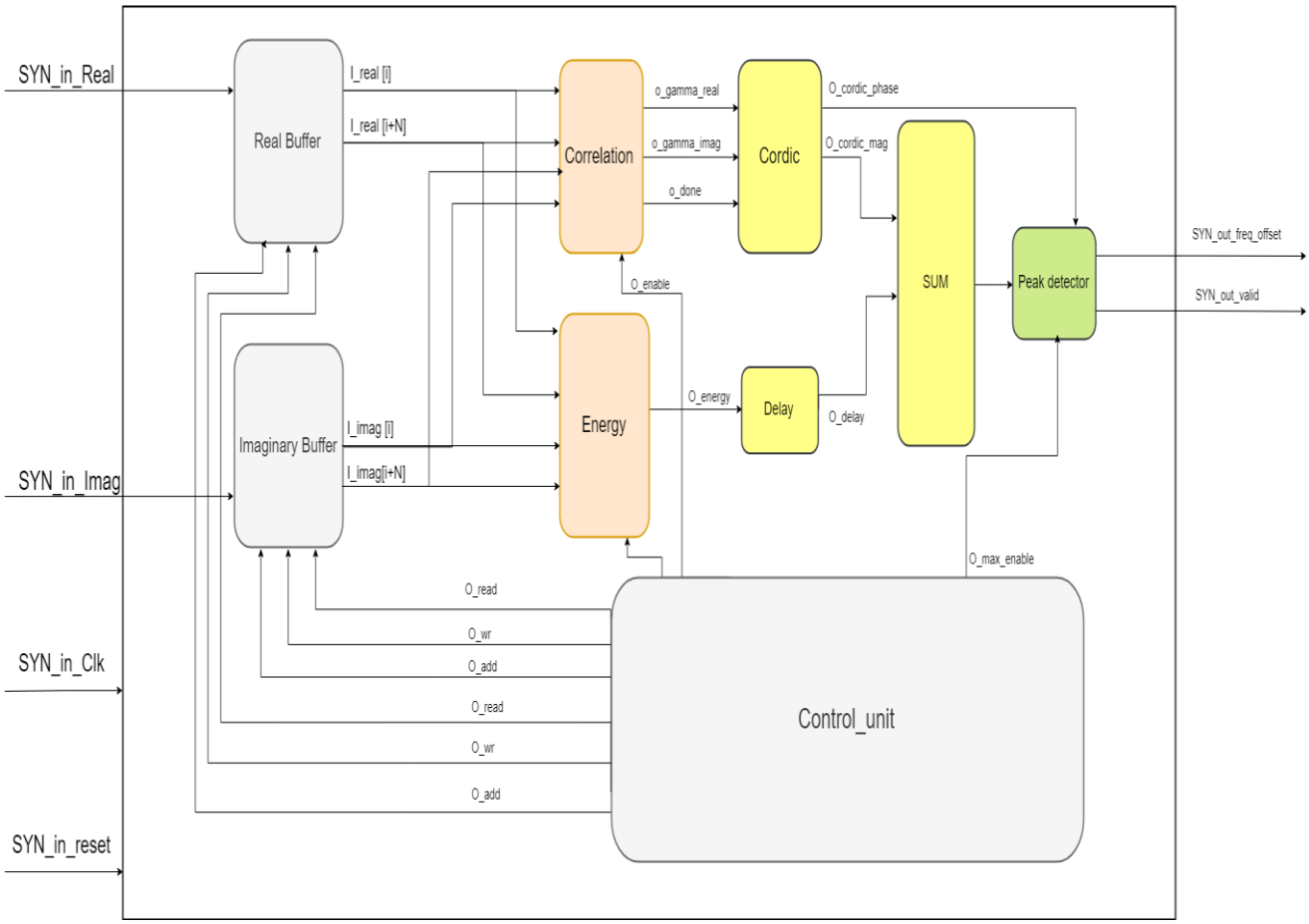


Figure 4.1-2 Architecture of the time and frequency offset estimation block

4.1.4. Operation

- First of all, the incoming data (real and imaginary) is collected in the real and imaginary buffers.
- According to the standard, the cyclic prefix length is (L=2) so the buffer is designed to out four values I_real [i], I_real[i+1], I_real[i+N+1], I_real[i+N+2]
  Where N is the FFT points (symbol length).
- The correlation block performs equation (2) in chapter 2 on the real and imaginary data coming from the real and imaginary buffers.
- The energy block performs equation (3) on the real and imaginary data coming from the real and imaginary buffers.
- If we look at equation (4) and equation (5) we will notice that it's required to get the magnitude and phase of the correlation out and here comes the rule of the CORDIC block in its vectoring mode.
- The out of the CORDIC blocks takes a specific number of cycles according to the number of bits and accuracy needed as will be discussed later, but to perform equation (4) we need the output from the CORDIC and the output from the energy block to be input at the SUM block at the same time and since the output of the energy block comes earlier, we need a delay block.

- The peak detector block gets the maximum of the input data according to equation (4) and then gets the phase according to this value and this will be the frequency offset estimated.

4.1.5. Sub blocks design

4.1.5.1. Correlation sub block

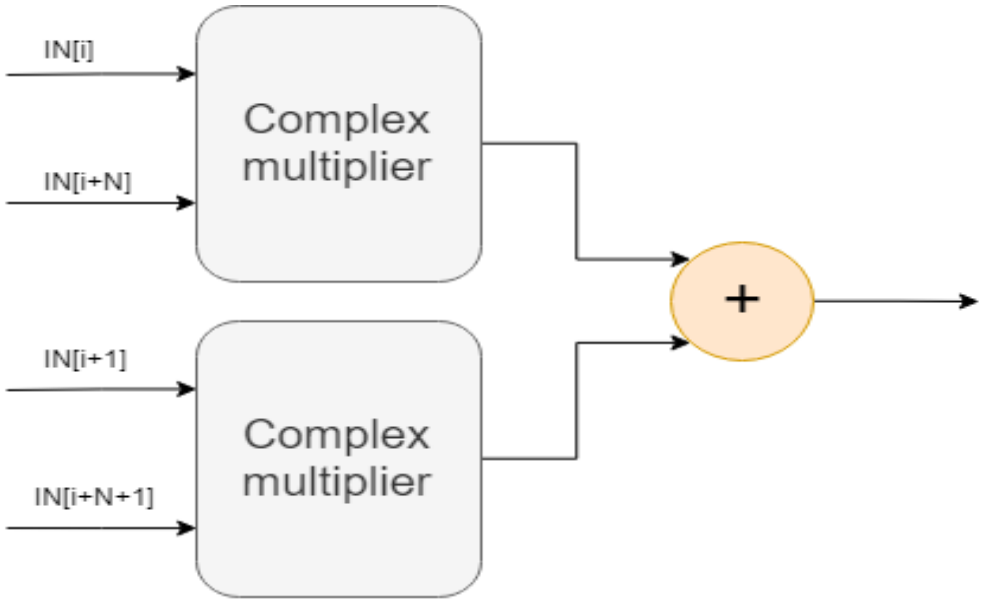


Figure 4.1-3 Correlation block design

4.1.5.2. Complex multiplier

For multiplying two complex numbers (real_1 + j imag_1) and(real_2 + j imag_2)

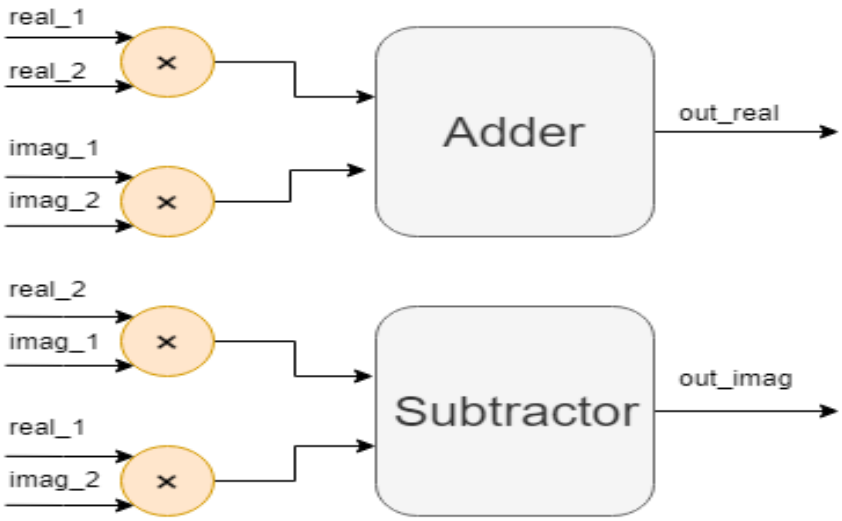


Figure 4.1-4 Complex multiplier design

4.1.5.3. CORDIC sub-block

As mentioned before, the CORDIC is an iterative algorithm so the output value of each iteration depends on the input from the last iteration.

The number of iterations is choosed according to the number of bits and accuracy needed. For example, if the number of bits is 16 bits then we will not need more than 15 iterations to get a very high accuracy. But we can get an acceptable accuracy using less number of iterations.

The CORDIC can be designed using two approaches:

1. A unit for each iteration

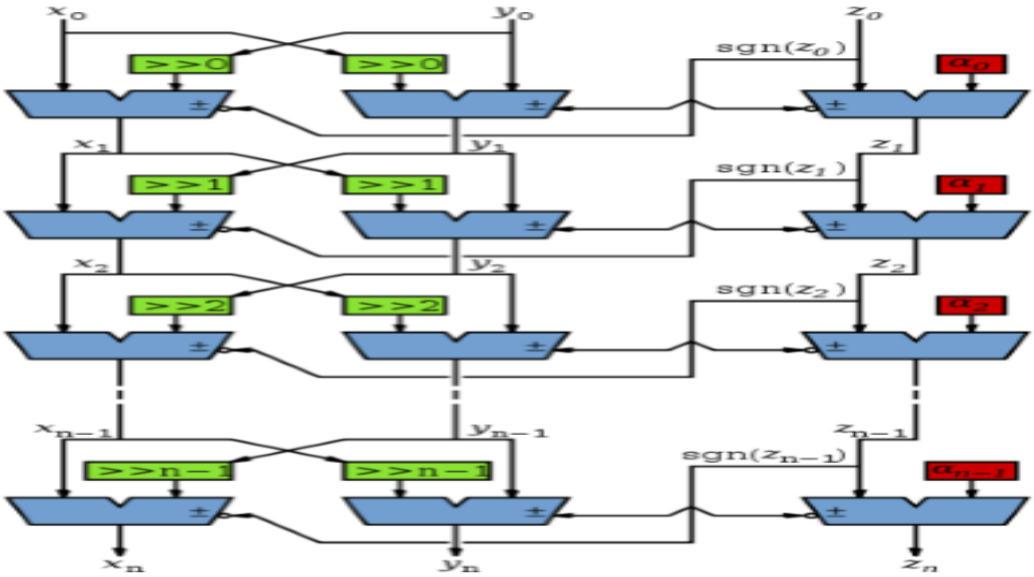


Figure 4.1-5 First Internal architecture of CORDIC sub block

The cost for this implementation is:

- Three ADD/SUB ALU units for each iteration.
- Shift operations: hardwired.

But typically, with a pipeline registers after each iteration, we can get a very high throughput.

2. Only one unit and feedback.

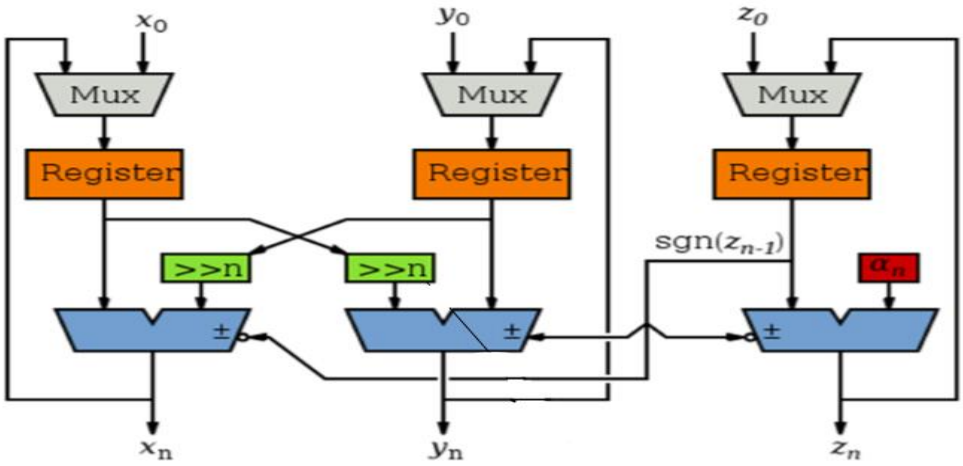


Figure 4.1-6 Second Internal architecture of CORDIC sub block

The cost for this implementation is:

- Very low throughput (n times less).
- The shifter is variable and costs logic.

But of course, this design has lower area compared to the first one.

4.1.6. Results

4.1.6.1. CORDIC results

The first approach is the one implemented in the frequency estimation block. The CORDIC block was implemented first on MATLAB to see the effect of increasing the number of iterations on the error.

NITERS	Real	ERROR	Imag	ERROR
1	−0.7072	0.8909	3.5354	0.3033
2	−2.2136	0.6156	2.8459	0.3861
3	−1.4573	0.1407	3.2980	0.0659
4	−1.8551	0.2571	3.0917	0.1404
5	−1.6586	0.0605	3.2013	0.0308
6	−1.5579	0.0402	3.2517	0.0196
7	−1.6084	0.0104	3.2268	0.0053
8	−1.5833	0.0148	3.2394	0.0073
9	−1.5958	0.0022	3.2332	0.0011
10	−1.6021	0.0040	3.2299	0.0022
11	−1.5990	0.0010	3.2314	0.0006
12	−1.5974	0.0006	3.2323	0.0002
13	−1.5981	0.0001	3.2318	0.0003
14	−1.5985	0.0005	3.2316	0.0005
15	−1.5984	0.0004	3.2318	0.0003

Figure 4.1-7 MATLAB results for CORDIC block

4.1.6.2. CORDIC RTL results

The design was tested with 80 complex numbers.

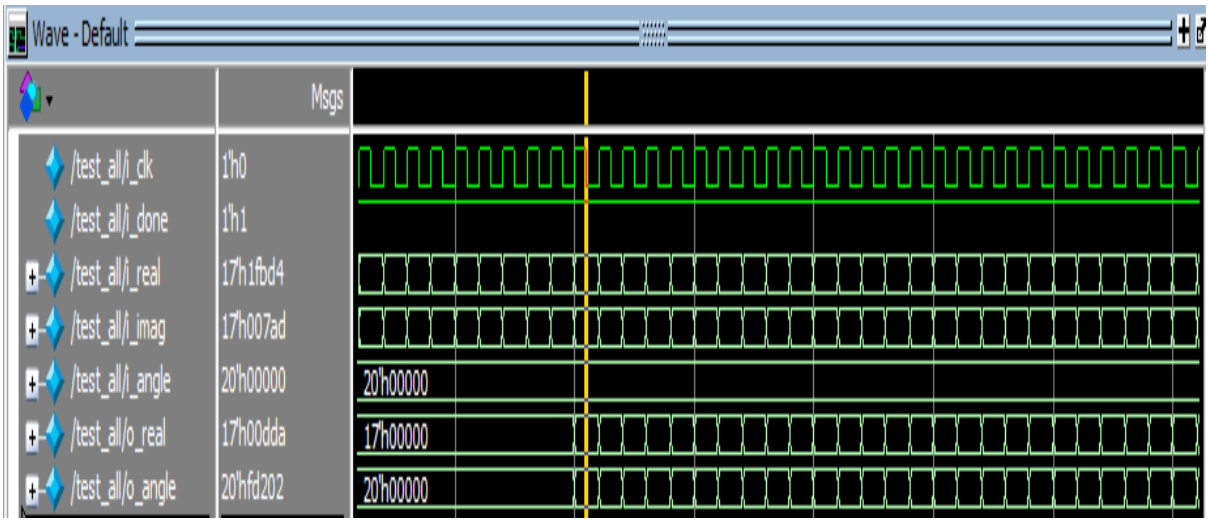


Figure 4.1-8 RTL results of the CORDIC block

The last figure shows inputs and outputs of the CORDIC block. Each output takes 9 cycles (number of iterations +1).

- The inputs and outputs (real and imaginary) are represented in 17 bits (5 bits integer and 12 fraction) as this is the out from the correlation block.
- The input and output angles are represented in 20 bits (12 bits integer and 8 bits fraction).

The outputs are taken from the RTL and transformed from fixed point into decimal then compared to MATLAB and the results were as follows.

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.8640	0.6557	1.4186	0.5904	0.8501	0.7660	0.5024	2.0653	0.8622	0.1673	0.5070	0.3149

Figure 4.1-9 Ideal out magnitude

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.8657	0.6560	1.4238	0.5925	0.8516	0.7681	0.5042	2.0742	0.8645	0.1687	0.5088	0.3164

Figure 4.1-10 RTL out magnitude

	1	2	3	4	5	6	7	8	9	10
1	-45.9742	44.9579	-170.5546	5.5700	85.4637	156.2048	-172.6988	-159.0247	98.2822	-38.8988

Figure 4.1-11 Ideal out phase

	1	2	3	4	5	6	7	8	9	10
1	-45.9922	44.9609	-170.6016	5.3750	85.4141	156.2266	-172.8906	-159.0938	98.1719	-39.3672

Figure 4.1-12 RTL out phase

The average error between the ideal outputs and the RTL outputs for magnitude and phase is calculated.

- Average error for magnitude = 0.0032
- Average error for phase = 0.1443

4.1.6.3. Block results

To test the whole block, we have to assume that there's a dummy transmitter that sends the data and that the data has passed through a channel and suffered from time and frequency shift.

The block was first implemented on MATLAB and it was found that it gives very good and accurate results for the timing offset if the FFT points and cyclic prefix length are very large.

For example,

- FFT points = 1024
- Cyclic prefix length = 128
- Number of symbols = 7
- Assumed frequency offset of 0.25
- Assumed timing offset of 4

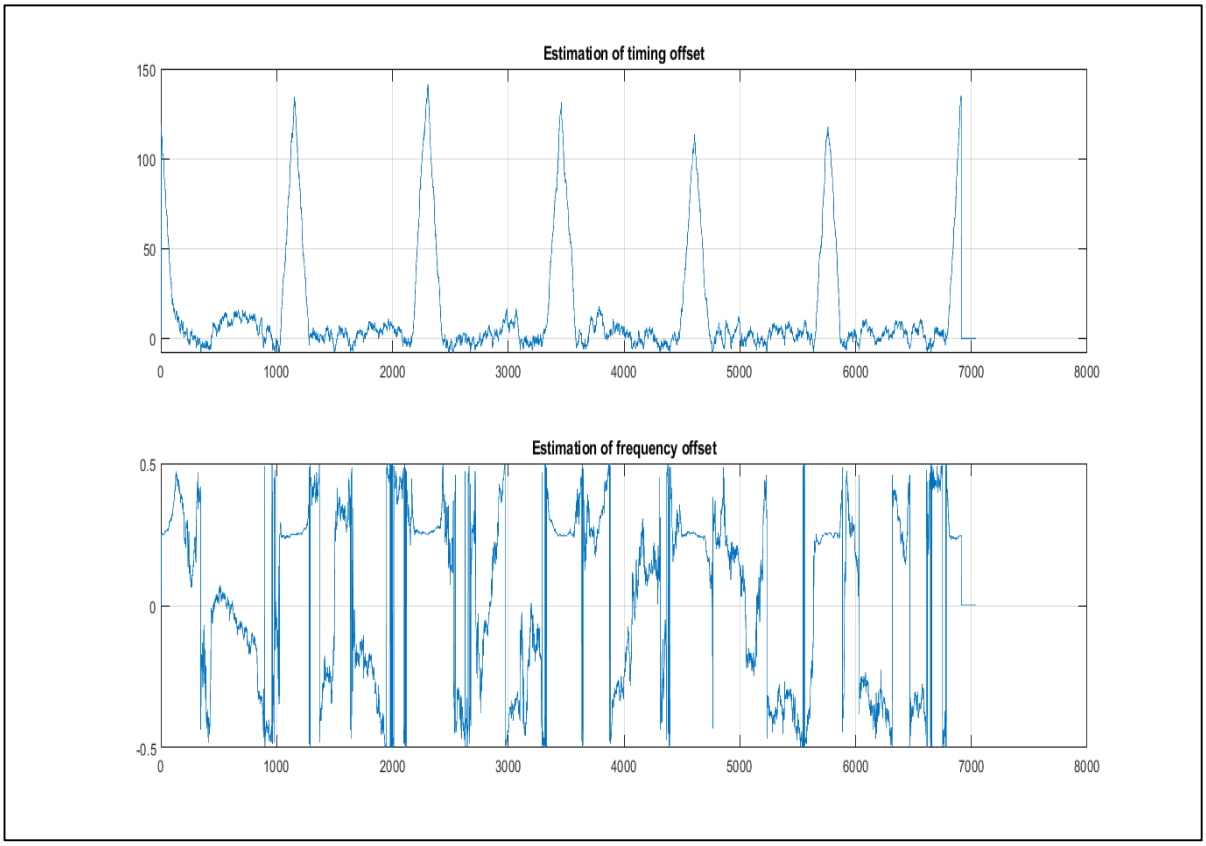


Figure 4.1-13 MATLAB results for time and frequency offset estimation

The peaks in the last figure defines the start of every symbol and the corresponding value on the frequency plot represents the frequency offset and it's the same for each symbol.

In our case, the FFT points and the cyclic prefix length are not very large so the correlation doesn't give very accurate results for timing offset that's why the block is implemented in RTL with only offset estimation and it's assumed that there is no timing offset.

Frequency offset estimated from MATLAB = 0.2591 rad = 14.8 deg.

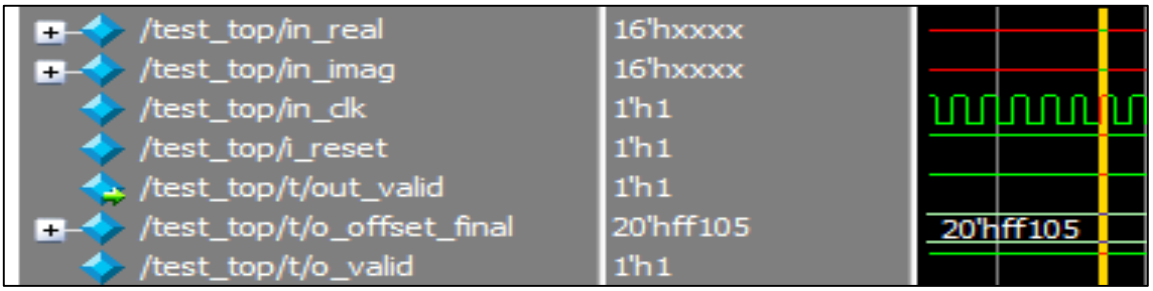


Figure 4.1-14 RTL results of the synchronization block

Frequency offset = 20'hff105 (fixed point representation of 12 bits integer and 8 bits fraction).

Frequency offset in decimal = -14.98 deg.

The output is negative to be input directly to the next block.

4.1.6.4. Synthesis results

The block was synthesized on Xilinx and DC compiler and the results were as follows.

```
Number of ports:                  153
Number of nets:                   373
Number of cells:                   30
Number of references:              10

Combinational area:       365021.756733
Noncombinational area:     69765.364134
Net Interconnect area:      4402.000000

Total cell area:          434787.120867
Total area:               439189.120867
```

Figure 4.1-15 Synchronization block area report

```
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  = 995.7347 uW   (67%)
  Net Switching Power  = 490.6813 uW   (33%)
                         ---------
Total Dynamic Power    =   1.4864 mW  (100%)

Cell Leakage Power     = 361.7212 uW
```

Figure 4.1-16 Synchronization block power report

```
 ------------------------------------------------------------------------
 data required time                                              99.58
 data arrival time                                              -16.82
 ------------------------------------------------------------------------
 slack (MET)                                                     82.76
```

Figure 4.1-17 Synchronization block total slack

4.2. CP removal and offset correction architecture

4.2.1. Top level

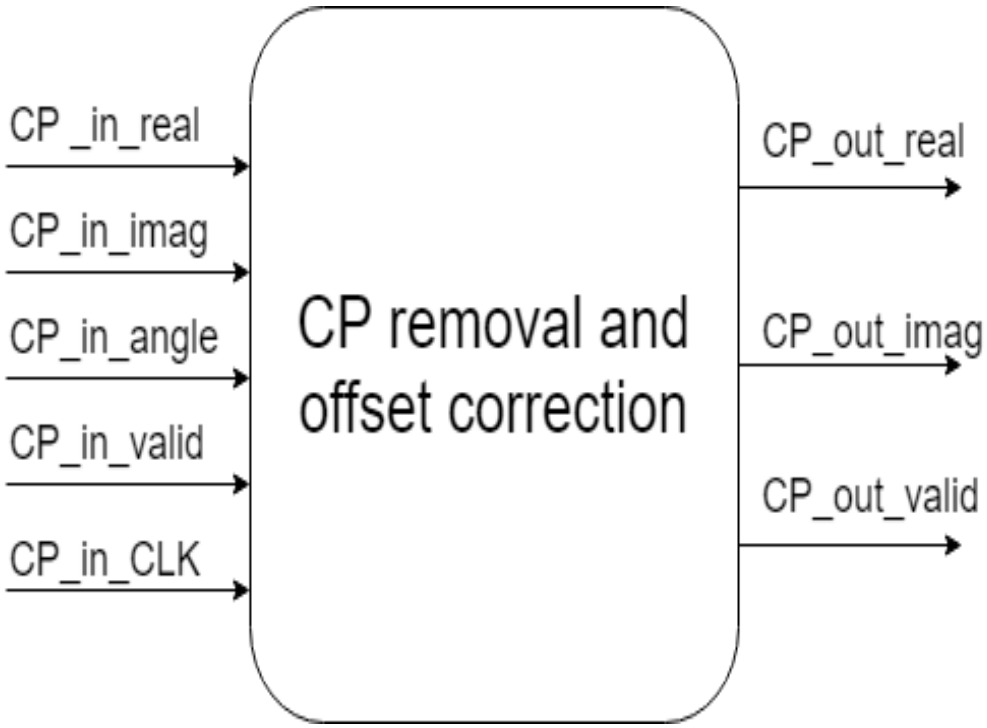


Figure 4.2-1 CP removal and offset correction Top Level

4.2.2. Block interface

Signal name	Direction	Description	Size (bits)
CP_in_real	input	Real input data	16
CP_in_imag	input	Imaginary input data	16
CP_in_angle	input	Frequency offset estimated from synchronization block	20
CP_in_valid	input	Valid input indicated that the offset is valid	1
CP_in_CLK	output	clock	1
CP_out_real	output	Real output to the FFT block	20
CP_out_imag	output	Imaginary output to the FFT block	16
CP_out_valid	output	Valid out to the FFT block	16

Table 4.2-1 CP removal and offset correction interface signals

4.2.3. Architecture

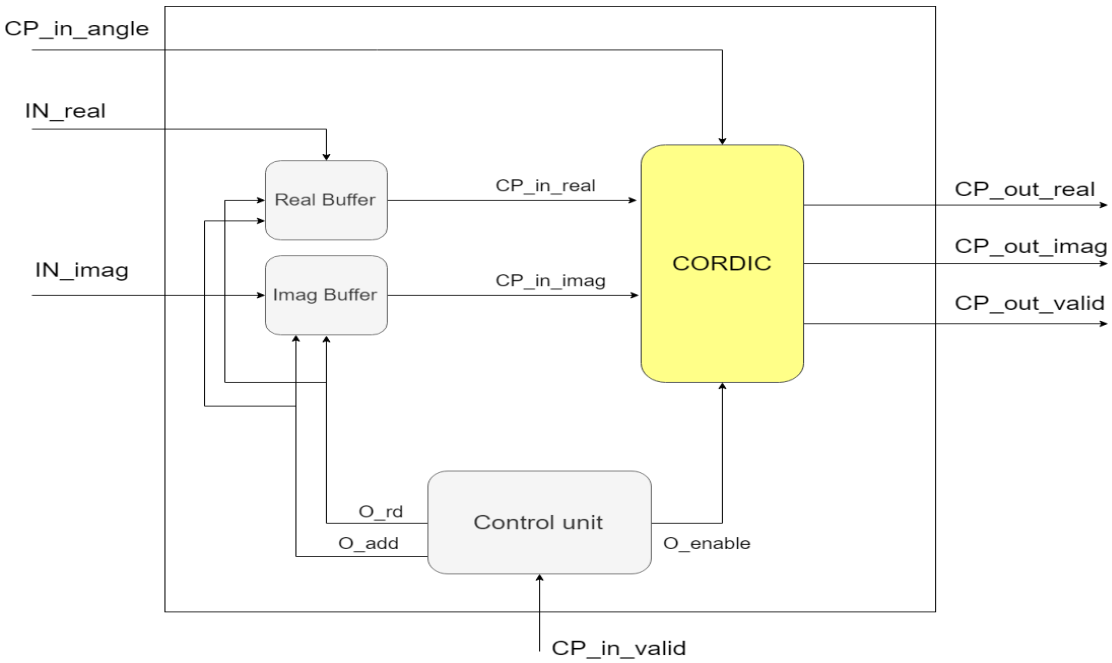


Figure 4.2-2 Architecture of CP removal and offset correction block

4.2.4. Operation

- The cyclic prefix is removed by controlling which address to read from the real and imaginary buffers.
- To correct the offset, we have to multiply the input by $(e^{-j2\pi\varepsilon k/N})$ and here comes the rule of the CORDIC block in its rotation mode.

4.2.5. Sub blocks design

The CORDIC block has the same design mentioned before with an additional unit that is responsible for pre rotation of the input angle to be in the range of

$(-\frac{\pi}{4}, \frac{\pi}{4})$ as the CORDIC block works only if the angle of rotation is in this range.

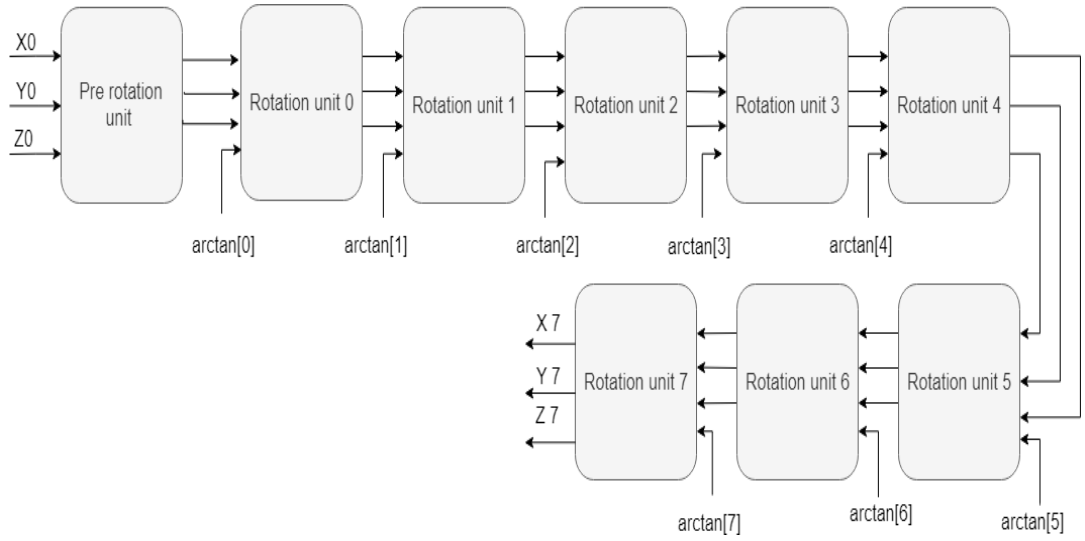


Figure 4.2-3 CORDIC sub block internal design

4.2.6. Results

The block was integrated with the frequency estimation block and tested with the output of it then the results were compared to MATLAB.

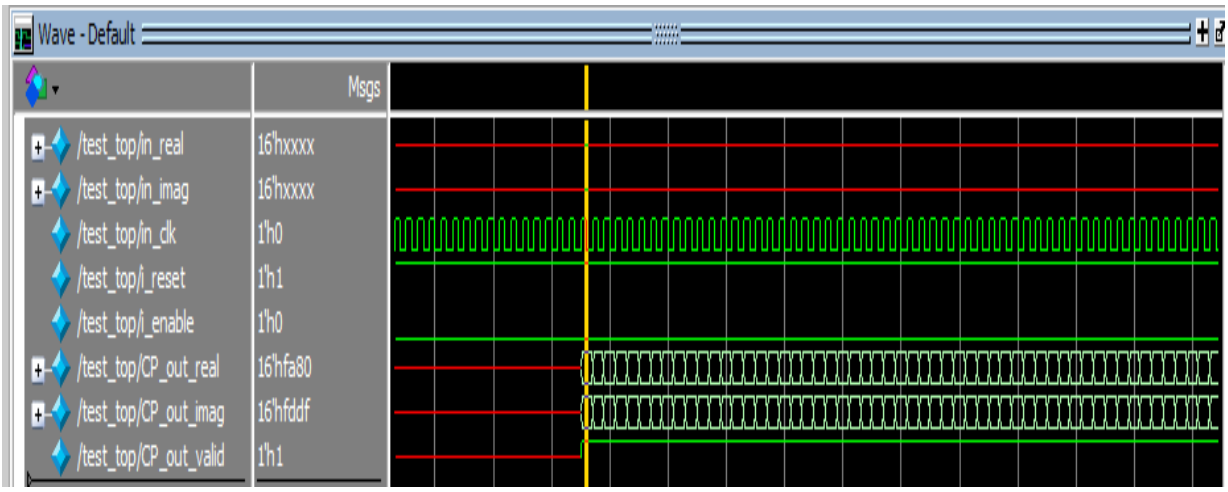


Figure 4.2-4 Timing diagram for the RTL output after CP removal and offset correction

	1	2	3	4	5	6	7	8	9	10	11	12
1	-0.3393	0.2166	0.4691	-0.3463	-1.1036	0.0885	0.4116	1.7123	0.6558	0.0691	0.7204	0.6893

Figure 4.2-5 Ideal real out

	1	2	3	4	5	6	7	8	9	10	11	12
1	-0.3438	0.2148	0.4666	-0.3481	-1.1135	0.0876	0.4131	1.7161	0.6563	0.0715	0.7217	0.6892

Figure 4.2-6 RTL real out

	1	2	3	4	5	6	7	8	9	10	11	12
1	-0.1306	-0.4069	-0.9657	0.9654	-0.9394	0.1839	0.6934	-0.2415	0.2850	1.6603	0.6759	0.0536

Figure 4.2-7 Ideal imaginary out

	1	2	3	4	5	6	7	8	9	10	11	12
1	-0.1331	-0.4109	-0.9712	0.9678	-0.9407	0.1838	0.6936	-0.2483	0.2839	1.6650	0.6753	0.0518

Figure 4.2-8 RTL imaginary out

The average error is calculated for the real and imaginary outputs.

- Average error for real = 0.0034
- Average error for imaginary = 0.0027

4.3. Fast Fourier Transform (FFT)

4.3.1. Top level

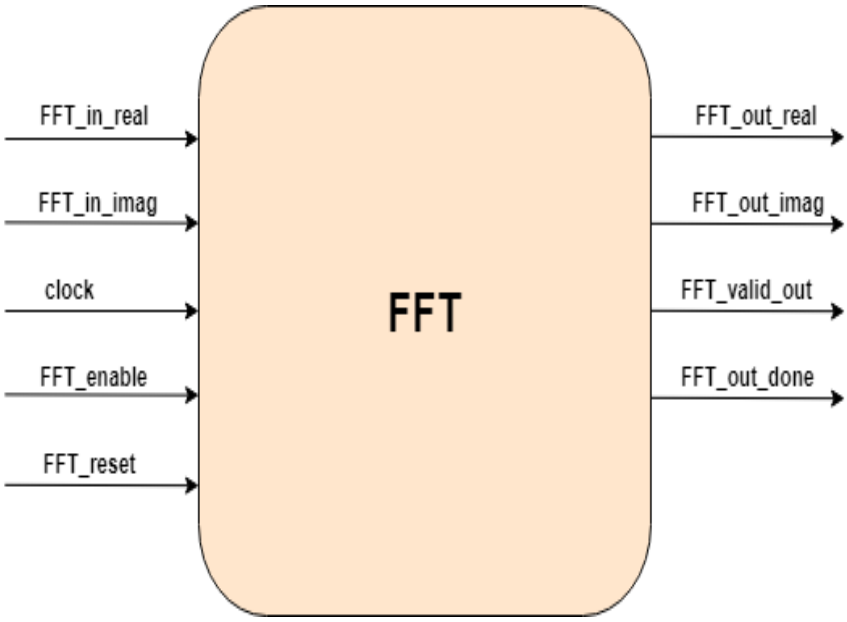


Figure 4.3-1 FFT Top Level

4.3.2. Block Interface

Signal name	Direction	Description	Size
FFT_in_real	Input	FFT input real part	16
FFT_in_imag	Input	FFT input imaginary part	16
clock	Input	FFT input clock	1
FFT_enable	Input	FFT block enable	1
FFT_reset	Input	FFT block reset	1
FFT_out_real	Output	FFT output real part	16
FFT_out_imag	Output	FFT output imaginary part	16
FFT_valid_out	Output	Signal indicates that output is valid	1
FFT_out_done	Output	Signal indicates that output is done	1

Table 4.3-1 FFT block interface signals

Algorithm

FFT has a lot of algorithms to be implemented but since our project is NB-IOT LTE we have a power and cost constraints so we choose the Cooley-Tukey algorithm in our design as it provides low cost and complexity.

Cooley-Tukey algorithm is a divide and conquer algorithm. Divide means break the given problem into sub problems of the same size and Conquer means recursively solve these sub problems. It combines the answers in at the end of the algorithm and generate the output. Number of stages to generate the output is $\log_2$ N where N is the data size.

Two methods to compute Cooley-Tukey algorithm:

1- Decimation in time.
2- Decimation in frequency.

Those two methods give nearly same throughput but differ in input/output pattern.

1- Decimation in time

FFT can be performed using DFT of even and odd points. Its input is out of order and its output is in order.

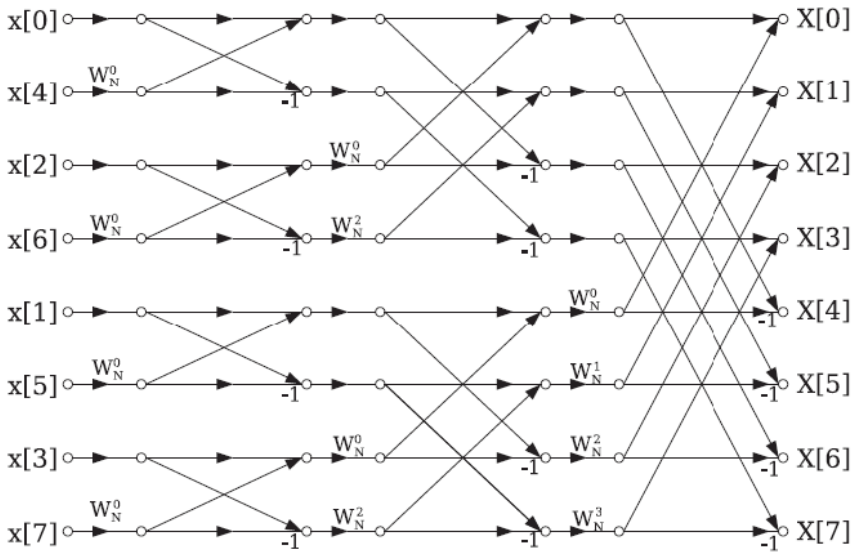


Figure 4.3-2 Decimation in time divide and conquer algorithm

2- Decimation in frequency

FFT can be decomposed using a first-half/second-half approach. Its input is ordered and its output is out of order.

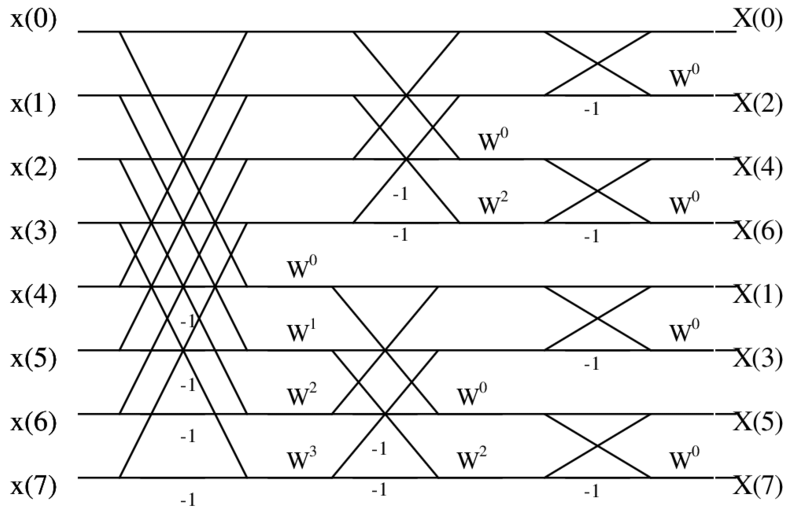


Figure 4.3-3 Decimation in frequency divide and conquer algorithm

We choose decimation in frequency for easier integration between blocks due to ordered input and we can re-arrange the output through output buffer.

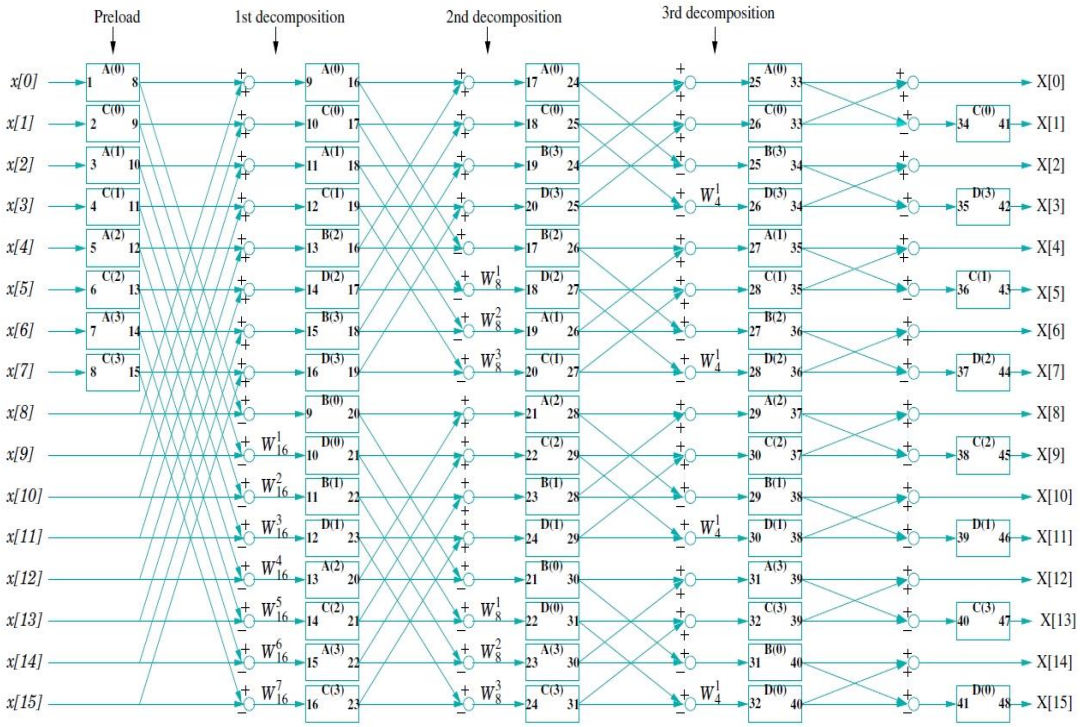


Figure 4.3-4 Time/space-embedded (TSE) signal flow graph of the 16-point memory-based FFT

Figure 4.3-4 shows the signal flow graph of our architecture. Output is in order due to ordering in buffering phase. Next section shows the detailed architecture and operation for FFT block.

4.3.3. Architecture

A faster way to implement the FFT is to use 16 multipliers to compute each stage in one cycle results in a $\log_2 N$ cycles and this is the fastest way but it has a large power consumption and large cost due to the 16 multipliers. In our project NB-IOT power and cost are constraints so we won't use 16 multipliers we will use only one multiplier to reduce the power and cost.
We choose memory-based architecture as shown in Figure 4.3-5. This is a lower speed method but speed is not our constraint.

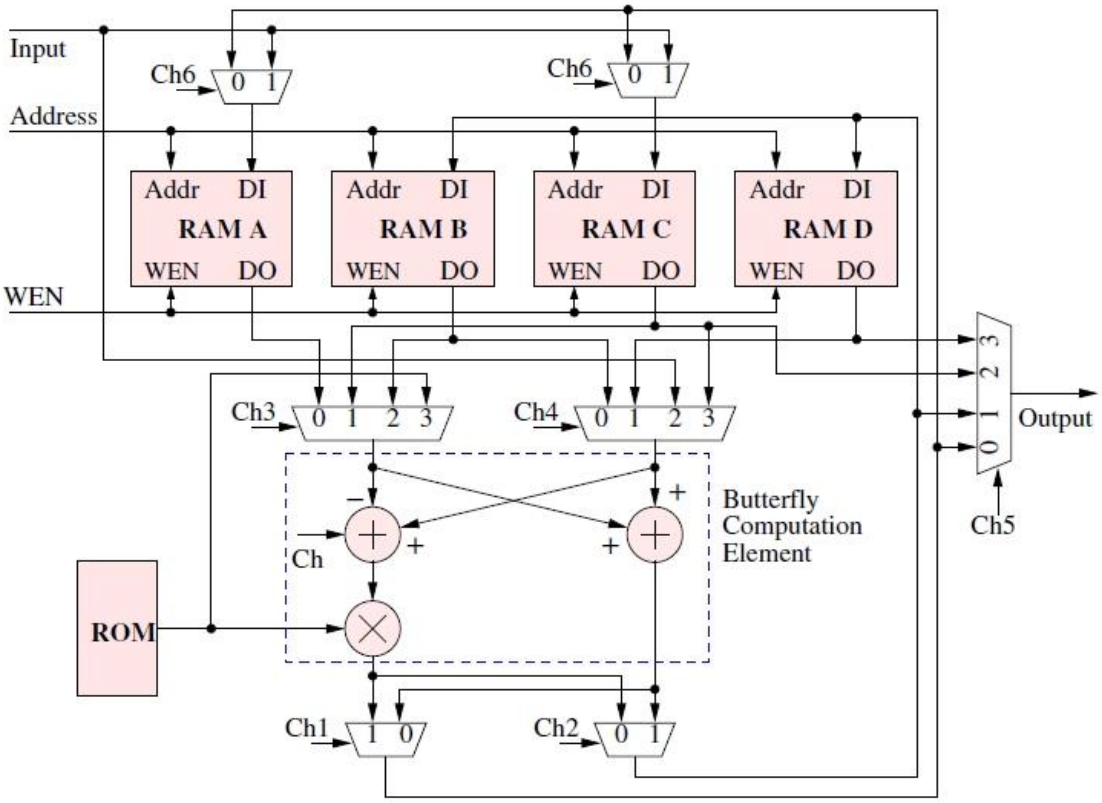


Figure 4.3-5 FFT memory-based architecture

4.3.4. Operation

- Preload Phase: load data x[2r] and x[2r+1] into the r^{th} address of RAM A and RAM C, respectively.
  Where r=0,1…, $\frac{N}{4} - 1$ and N is the number of points for FFT operations.
- Decomposition phases: ($(\log_2 N) - 1$) decomposition stages are required for an N-point FFT processor. In the first decomposition stage, data is read from address $r = 0$ to address $r = \frac{N}{2} - 1$ of RAM A and RAM C, and rest of data is read from the external input buffer. In the second stage computation data is read from the four RAMs. The N-point FFT is decomposed into two N/2-point FFTs, the upper and the lower FFT. Upper and lower are the same procedure with different inputs. The subsequent decomposition operation can be executed in a similar fashion until the last decomposition operation is completed.
- Buffering phase: this phase reorders the output sequence through the RAMs.

The below figure shows the whole operation for the memory-based FFT.

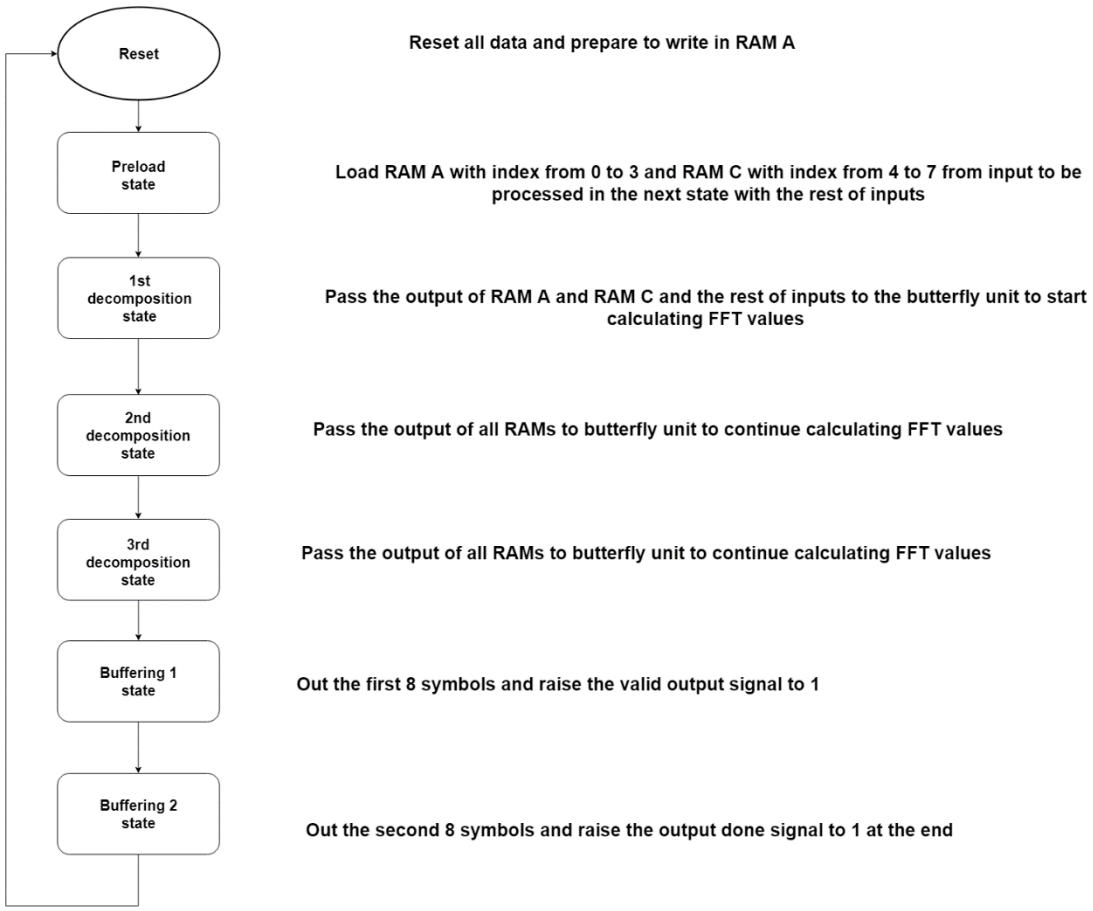


Figure 4.3-6 Flow chart of the Finite state machine control unit for the proposed 16-bit FFT

4.3.5. Results

We use fixed point representation to can deal with floating point numbers. Floating point representation is more complex because of the infinite probability of the point place, so we need a complex processor to can deal with a floating point accurately 100%. In our project NB-IOT we have constraints on power and cost so we used fixed point. We choose 12-bits fraction and 4-bits integer based on best accuracy and range of numbers will be around one so 4-bits will fit well.

Real values

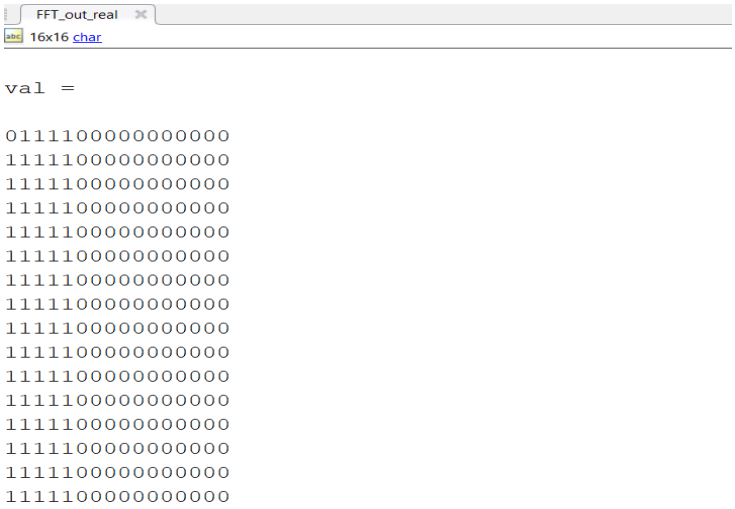


Figure 4.3-7 Real Values for FFT output

Imaginary values

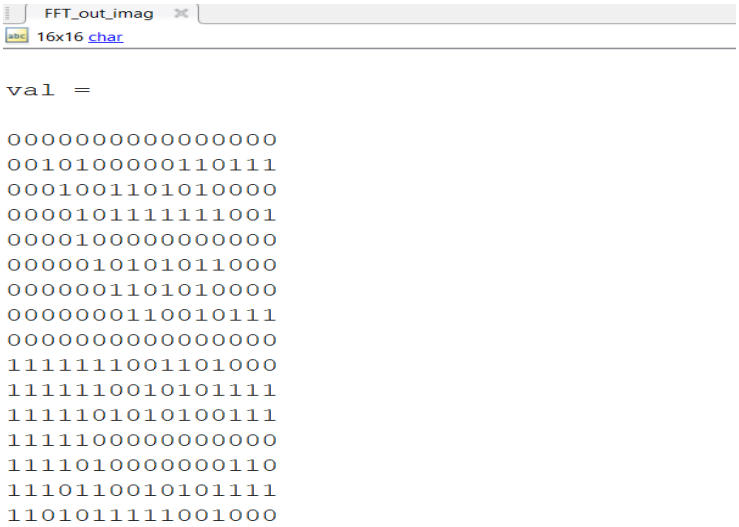


Figure 4.3-8 Imaginary Values for FFT output

We can see that the outputs are nearly the same, due to fixed point accuracy there is some error but it's nearly zero.

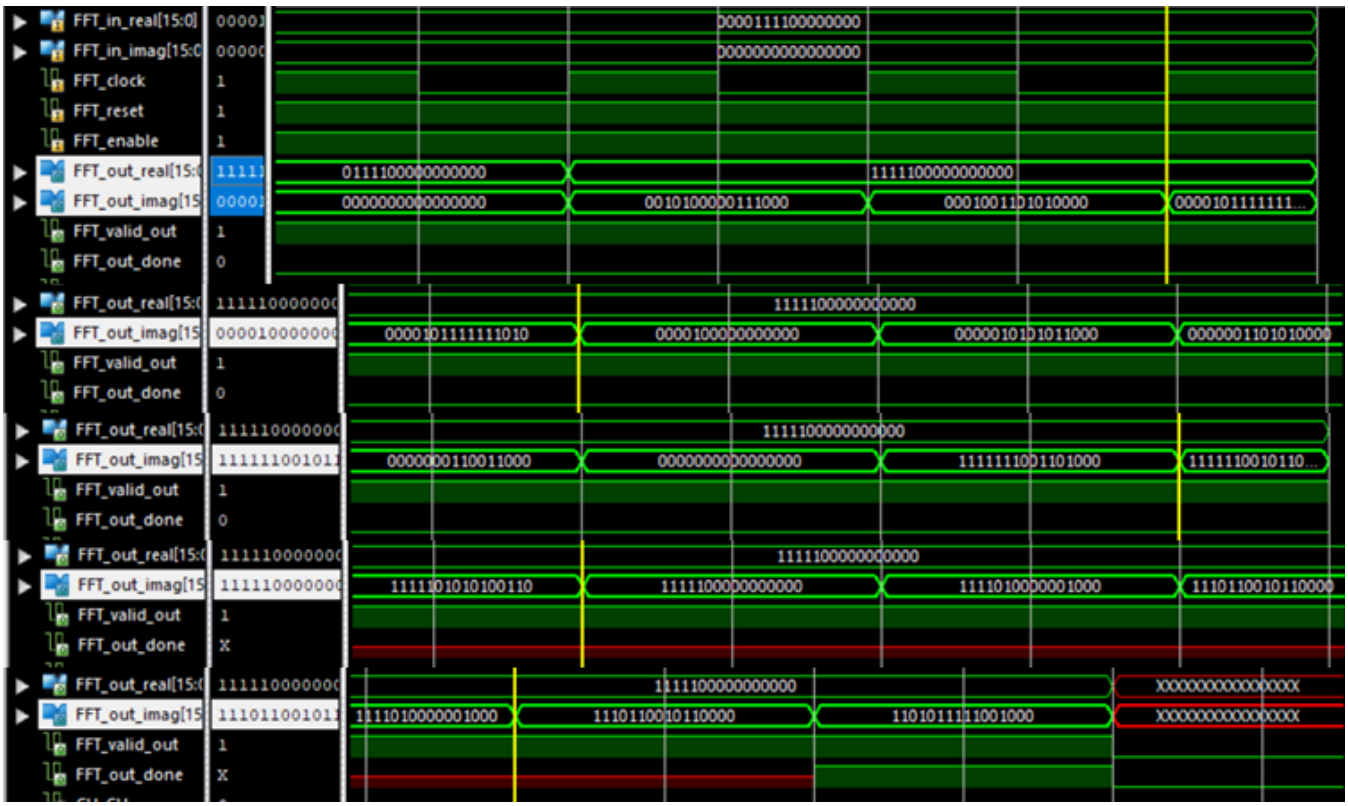


Figure 4.3-9 Result of RTL

4.3.6. Synthesis Results

FFT is synthesised using Xilinx ISE and Synopsis Design Complier at clock equal to 10MHz.

The result are as follows for Area, Power and Timing.

```
****************************************
Report : area
Design : FFT
Version: B-2008.09
Date   : Thu Oct 10 00:14:16 2013
****************************************

Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File: /root/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_ss_1p08v_125c.db)

Number of ports:              69
Number of nets:              479
Number of cells:              13
Number of references:         10

Combinational area:       46530.248307
Noncombinational area:      162.384602
Net Interconnect area:       76.000000

Total cell area:          46692.632909
Total area:               46768.632909
```

Figure 4.3-10 Area report for the 16-bits FFT

```
****************************************
Report : power
        -analysis_effort low
Design : FFT
Version: B-2008.09
Date   : Thu Oct 10 00:14:20 2013
****************************************


Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File: /root/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_ss_1p08v_125c.db)


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

Design        Wire Load Model            Library
------------------------------------------------
FFT                   ForQA              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  =  43.7327 uW   (72%)
  Net Switching Power  =  16.7523 uW   (28%)
                         ---------
Total Dynamic Power    =  60.4850 uW  (100%)

Cell Leakage Power     =  37.9122 uW
```

Figure 4.3-11 Power report for the 16-bits FFT

```
clock FFT_clock (rise edge)                          100.00      100.00
clock network delay (ideal)                            0.00      100.00
clock uncertainty                                     -0.25       99.75
output external delay                                 -4.00       95.75
data required time                                                95.75
--------------------------------------------------------------------------
data required time                                                95.75
data arrival time                                                -18.20
--------------------------------------------------------------------------
slack (MET)                                                       77.55
```

Figure 4.3-12 Timing report for the 16-bits FFT

4.4. Resource Elements De-mapper

4.4.1. Top level

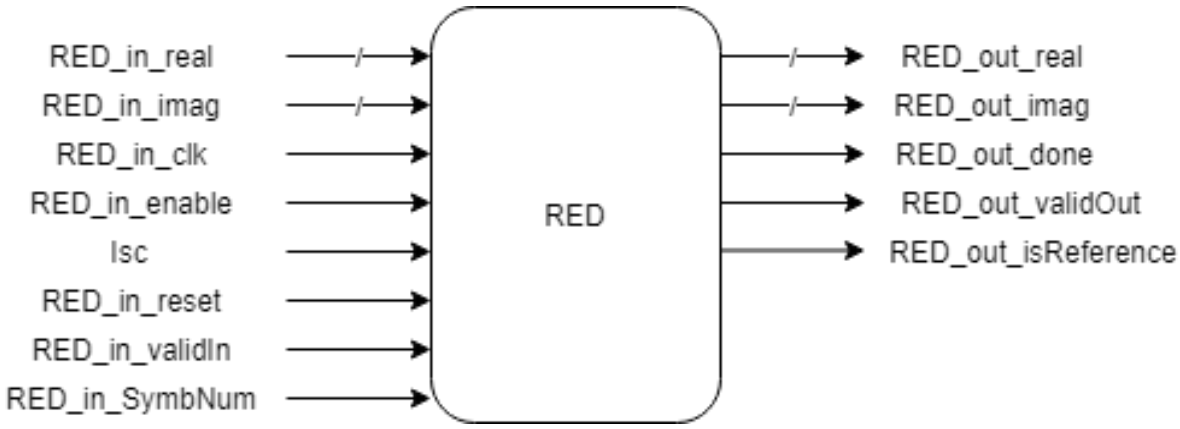


Figure 4.4-1 Resource elements de-mapper Block Top-level

4.4.2. Block interface

Signal Name	Direction	Description	Size
RED_in_real	input	Resource element De-mapper input real part	16
RED_in_imag	input	Resource element De-mapper input imaginary part	16
RED_in_clock	input	Resource element De-mapper input clock	1
RED_in_reset	input	RED reset	1
RED_in_enable	input	RED enable	1
Isc	input	allocated set of subcarriers from an upper layer	6
RED_out_real	output	RED output real part	16
RED_out_img	output	RED output imaginary part	16
RED_out_isRefrence	output	to determine if the signal is a reference signal or data	1
RED_out_done	output	RED done operation	1
RED_in_SymbolNum	input	current SCFDMA symbol number in the slot	2

Table 4.4-1 Resource elements De-mapper interface signals

4.4.3. Architecture

The Resource Elements De-mapper consists of A single RAM and a control unit as shown in Figure 4.4.1-1 which gives the control signals according and addresses to the given upper layer input I_{sc}.

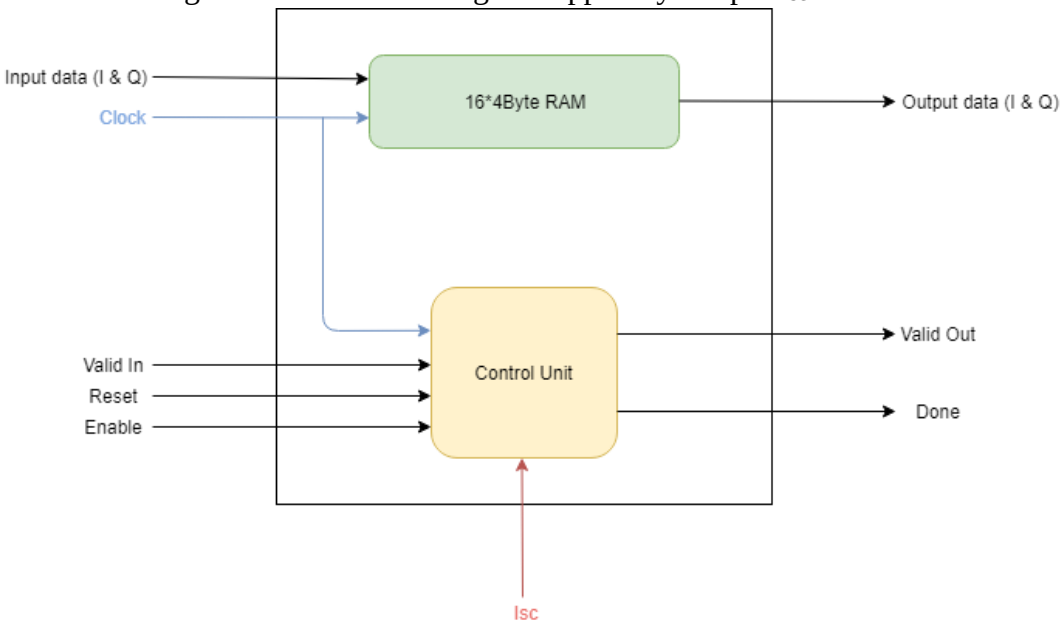


Figure 4.4-2 Resource elements de-mapper Architecture

4.4.4. Operation

The resource elements de-mapper stores the FFT output and chooses the correct set of allocated subcarriers as mentioned before according to the Table 3.2-3. It operates serially by collecting the symbols provided by FFT then output the correct set of subcarriers to the next block also is tells the Channel Estimation block if the data is a reference signal in order for it to get them.

4.4.5. Results

Shown below the results of the RTL simulation and MATLAB model results for several cases

At Isc = 3

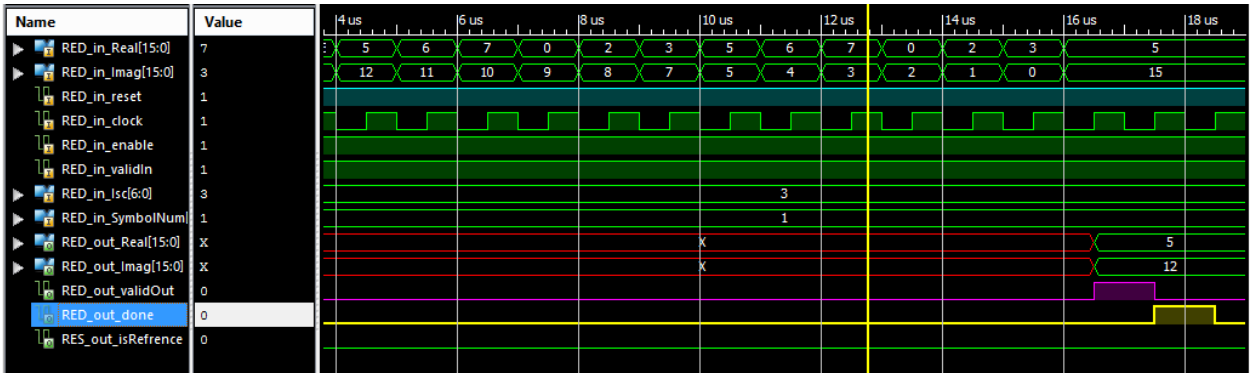


Figure 4.4-3 RTL output waveforms for RED block at single subcarrier mode

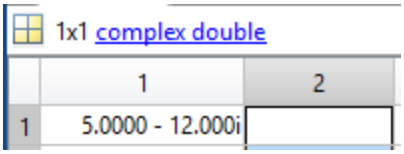


Figure 4.4-4 MATLAB output for RED model at single subcarrier mode

At Isc = 12

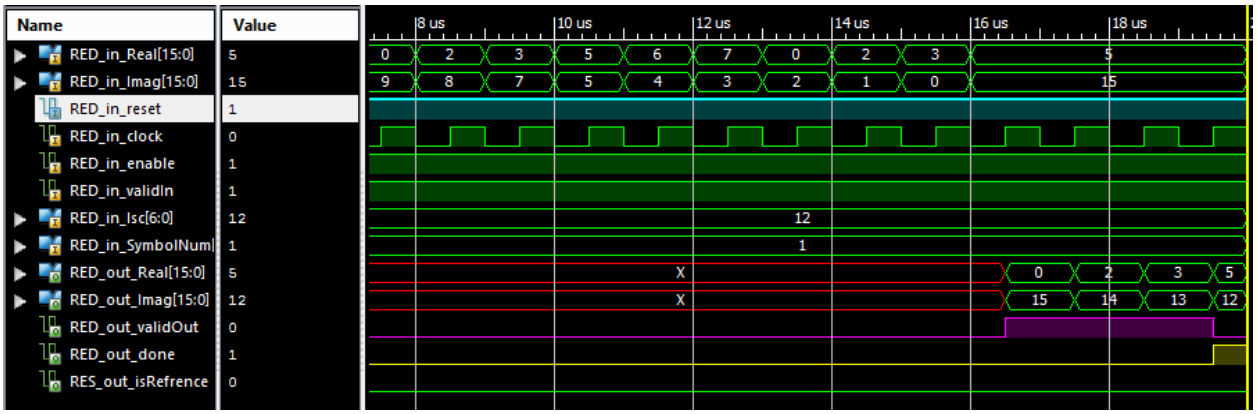


Figure 4.4-5 RTL output waveforms for RED block at 3 subcarriers mode

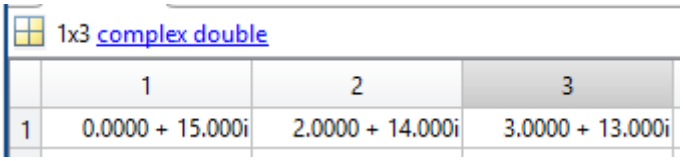


Figure 4.4-6 MATLAB output for RED model at 3 subcarriers mode

At Isc = 16

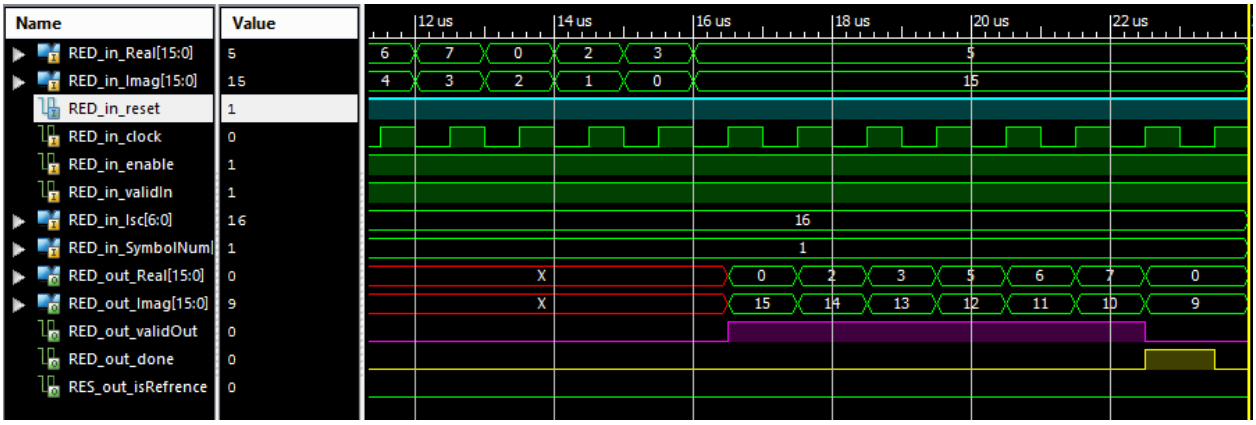
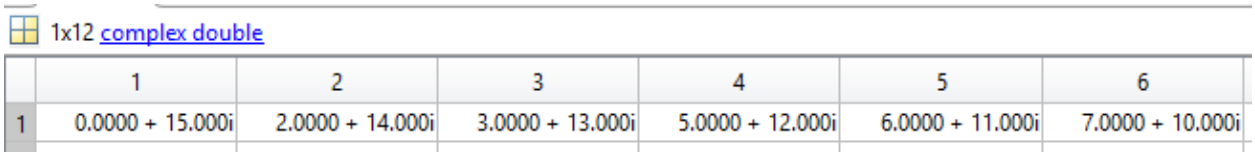


Figure 4.4-7 RTL output waveforms for RED block at 6 subcarriers mode



1x12 complex double

	1	2	3	4	5	6
1	0.0000 + 15.000i	2.0000 + 14.000i	3.0000 + 13.000i	5.0000 + 12.000i	6.0000 + 11.000i	7.0000 + 10.000i

Figure 4.4-8 MATLAB output for RED model at 6 subcarriers mode

At Isc = 18

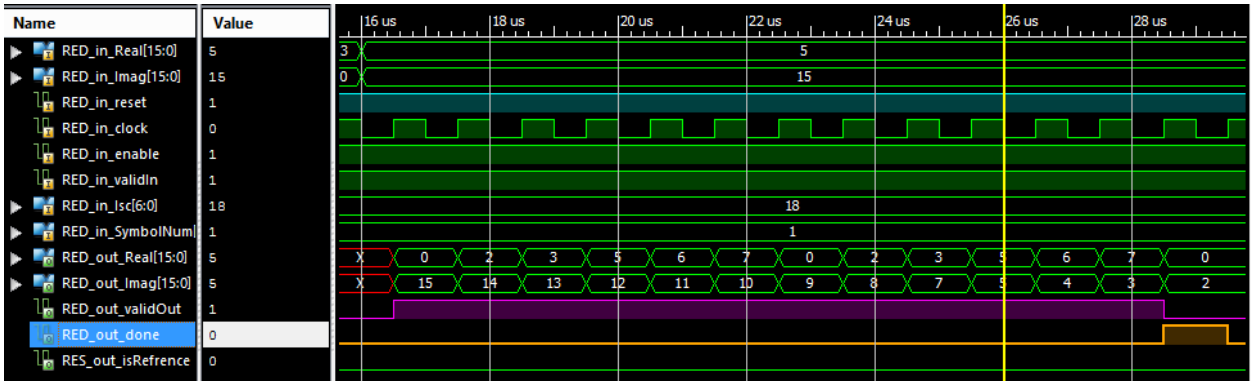
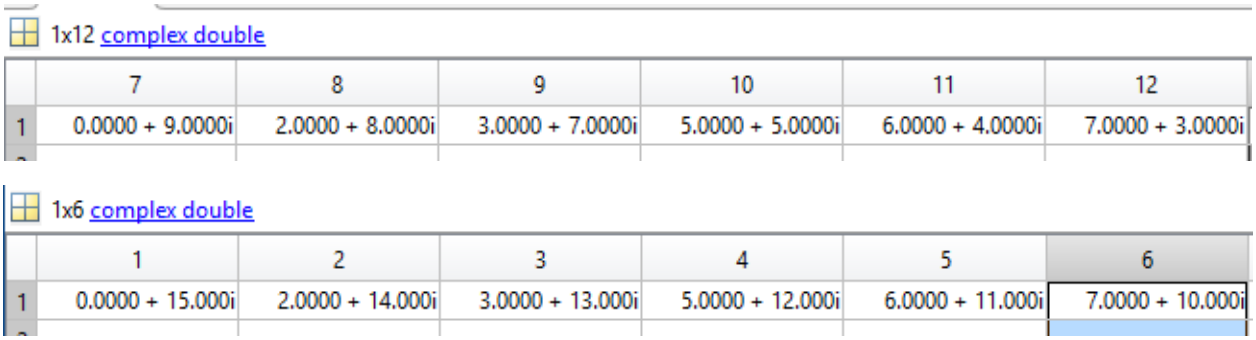


Figure 4.4-9 RTL output waveforms for RED block at 12 subcarriers mode



1x12 complex double

	7	8	9	10	11	12
1	0.0000 + 9.0000i	2.0000 + 8.0000i	3.0000 + 7.0000i	5.0000 + 5.0000i	6.0000 + 4.0000i	7.0000 + 3.0000i

1x6 complex double

	1	2	3	4	5	6
1	0.0000 + 15.000i	2.0000 + 14.000i	3.0000 + 13.000i	5.0000 + 12.000i	6.0000 + 11.000i	7.0000 + 10.000i

Figure 4.4-10 MATLAB output for RED model at 12 subcarriers mode

4.4.6. Synthesis

```
Number of ports:                 81
Number of nets:                2140
Number of cells:               2094
Number of references:            43

Combinational area:       26041.548358
Noncombinational area:    11781.119968
Net Interconnect area:    1004.000000

Total cell area:          37822.668327
Total area:               38826.668327
```

Figure 4.4-11 Area report for RED Block

```
data arrival time                                          0.31
-----------------------------------------------------------------
(Path is unconstrained)
```

Figure 4.4-12 Timing report for RED Block

```
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  = 624.3515 uW   (67%)
  Net Switching Power  = 306.3182 uW   (33%)
                         ---------
Total Dynamic Power    = 930.6697 uW  (100%)

Cell Leakage Power     =  29.0298 uW
```

Figure 4.4-13 Power report for RED Block

4.5. Channel estimation

4.5.1. Top level

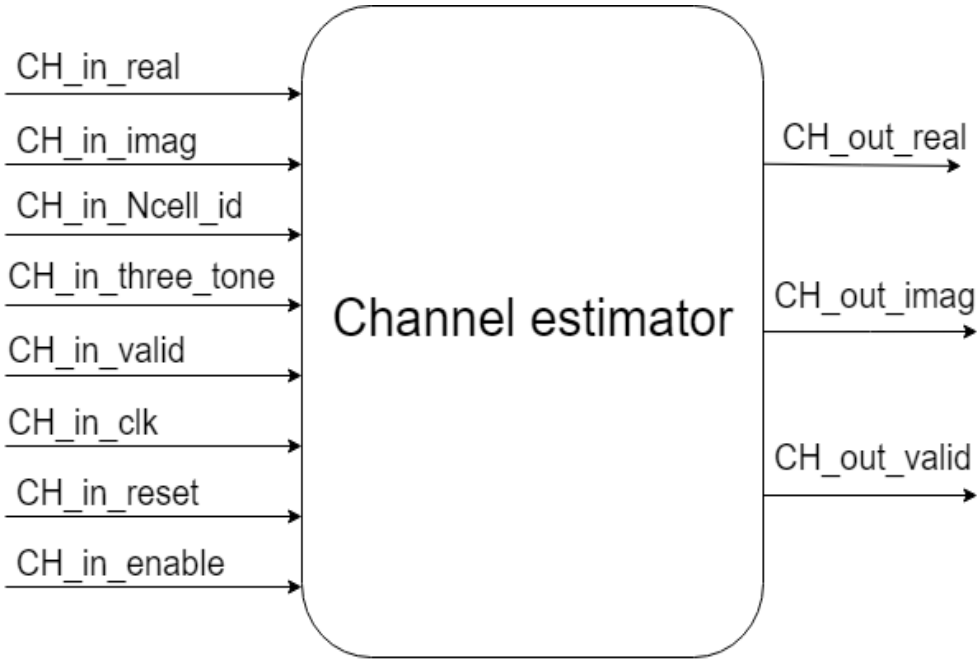


Figure 4.5-1 Channel estimator Top Level

4.5.2. Block interface

Signal name	Direction	Description	Size (bits)
CH_in_real	input	Real input from resource element De-mapper	16
CH_in_imag	input	Imaginary input from resource element De-mapper	16
CH_in_Ncell_id	input	Cell ID(upper layer parameter)	9
CH_in_three_tone	input	First slot (upper layer)	20
CH_in_valid	input	Indicates that the inputs from resource element De-mapper are valid	1
CH_in_clk	input	clock	1
CH_in_reset	input	reset signal	1
CH_in_enable	input	Enable signal	1
CH_out_real	output	Real outputs for estimated channel	16
CH_out_imag	output	Imaginary outputs for estimated channel	16
CH_out_valid	output	Input to the equalizer that indicates that the channel is estimated and ready	1

Table 4.5-1 Channel estimator interface signals

4.5.3. Architecture

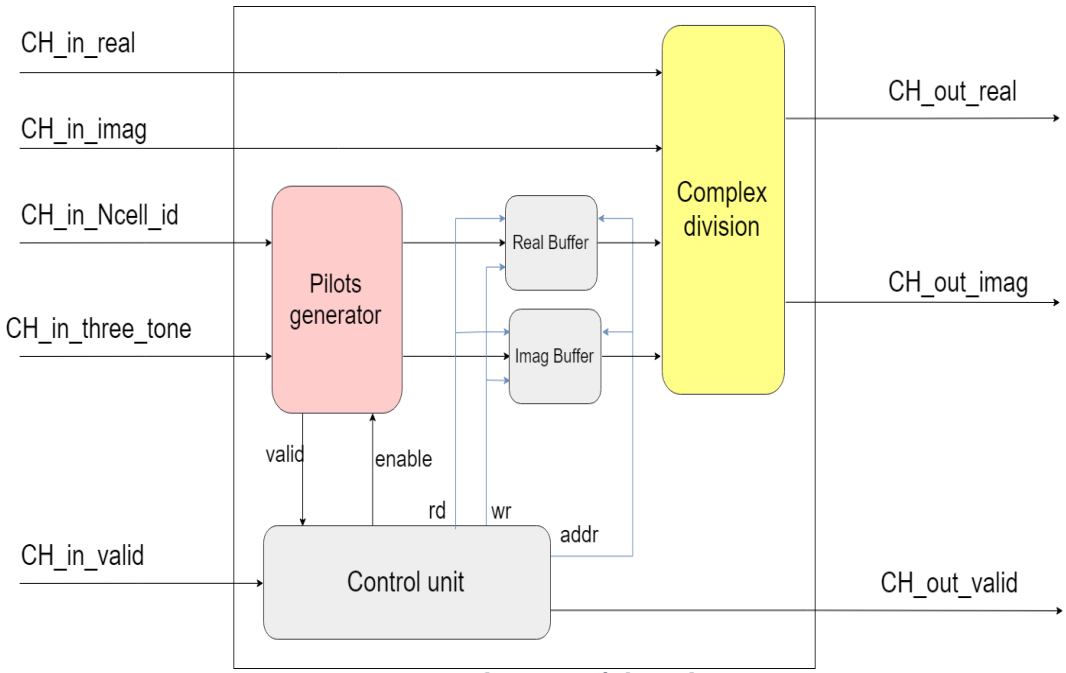


Figure 4.5-2 Architecture of channel estimator

4.5.4. Operation

- When the (CH_in_valid) signal is high, this means that the values of the pilots from the resource element De-mapper are ready and valid.
- The control unit outputs enable signal to the pilot's generator so it starts calculating the pilots according to the upper layer inputs.
- When the pilots are done the pilot, generator outputs a valid signal to the control unit.
- The control unit enables the write signal into the real and imaginary buffers.
- When all pilots are generated, the control unit enables the read signal from the real and imaginary buffers and the complex divider starts to divide the incoming pilots by the generated pilots to get an estimate of the channel at the pilots' positions.

4.5.5. Sub blocks design

4.5.5.1. Pilots generator sub block

an exponential equation and to implement this we used the CORDIC block in its rotation mode with the real input equals to(one) , the imaginary input equals to(zero) and the angle input comes from a combinational unit that calculates the angle according to the upper layer inputs.

The Φ (n) Tables mentioned before in the standard part are stored in LUTs.

The block was first designed with a CORDIC block for each pilot so the pilots were generated at the same time which means high throughput (speed) but of course very high area and power consumption.

So, the block was then designed with only one CORDIC block so the area is reduced but each pilot is generated each nine cycles.

4.5.6. Results

4.5.6.1. MATLAB and RTL

The block was tested to generate pilots at the following parameters:

- NcellID = 1
- Three tone cyclic shift = 0

1	2	3
0.7071 + 0.7071i	-0.7071 - 0.7071i	0.7071 - 0.7071i

Figure 4.5-3 MATLAB results for three pilots

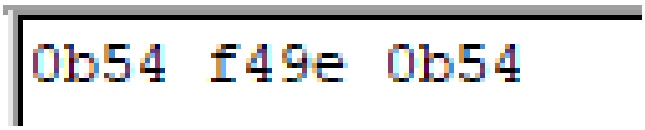


Figure 4.5-4 RTL results for real part of the three pilots

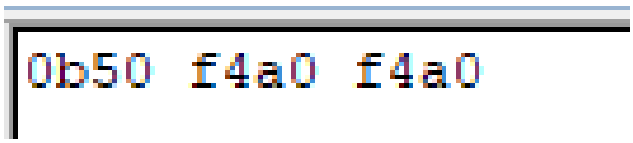


Figure 4.5-5 RTL results of the imaginary part of the three pilots

4.5.6.2. Synthesis results

```
Combinational area:       494912.962578
Noncombinational area:    29975.255543
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:          524888.218121
```

Figure 4.5-6 Channel estimation area report

```
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  =   2.1591 mW   (82%)
  Net Switching Power  = 462.2240 uW   (18%)
                         ---------
Total Dynamic Power    =   2.6213 mW  (100%)

Cell Leakage Power     = 426.8418 uW
```

Figure 4.5-7 Channel estimation power report

4.6. Equalizer

4.6.1. Top level

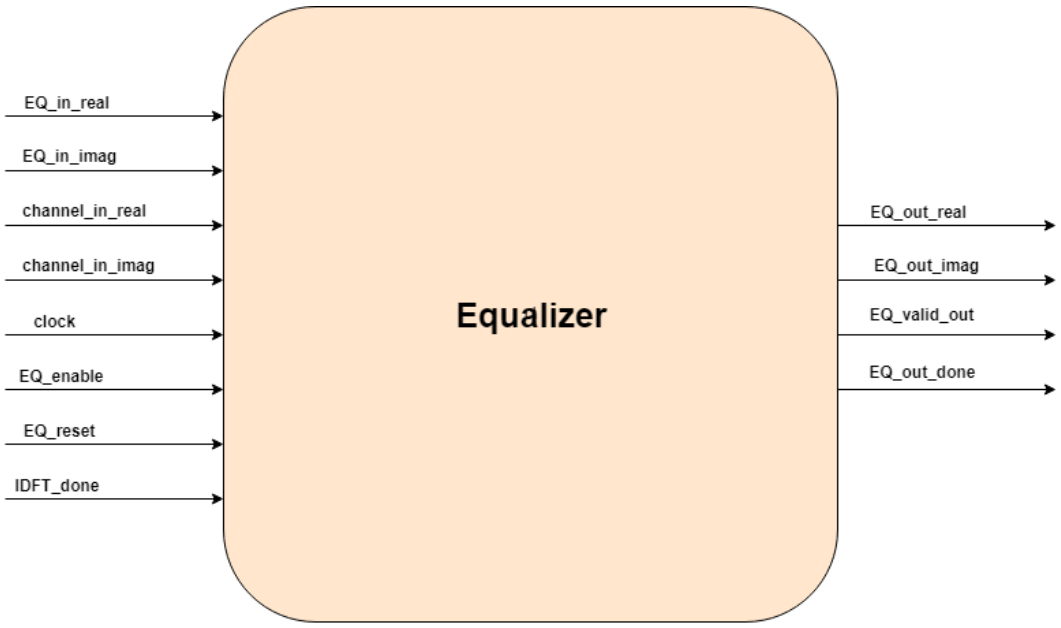


Figure 4.6-1 Equalizer block Top Level

4.6.2. Block interface

Signal	Direction	Description	Size
EQ_in_real	Input	Equalizer input real part	12
EQ_in_imag	Input	Equalizer input imaginary part	12
Channel_in_real	Input	Channel Information real part	12
Channel_in_imag	Input	Channel Information imaginary part	12
Clock	Input	Equalizer clock	1
EQ_enable	Input	Equalizer enable	1
EQ_reset	Input	Equalizer reset	1
IDFT_done	Input	Input from next block to handle integration between them	1
EQ_out_real	Output	Equalizer output real part	12
EQ_out_imag	Output	Equalizer output imaginary part	12
EQ_valid_out	Output	Signal indicates the valid output	1
EQ_out_done	Output	Signal indicates the end of the output	1

Table 4.6-1 Equalizer interface signals

4.6.3. Architecture

We design our own architecture, memory-based architecture is simply a memory and divider with multiplexing between symbols and a control Unit to control the whole design.

We tried 2 approaches, first approach rate is higher than the next block, IDFT, so when we started integration this architecture failed because IDFT work symbol by symbol so we modified it to fit with IDFT block.

First Architecture

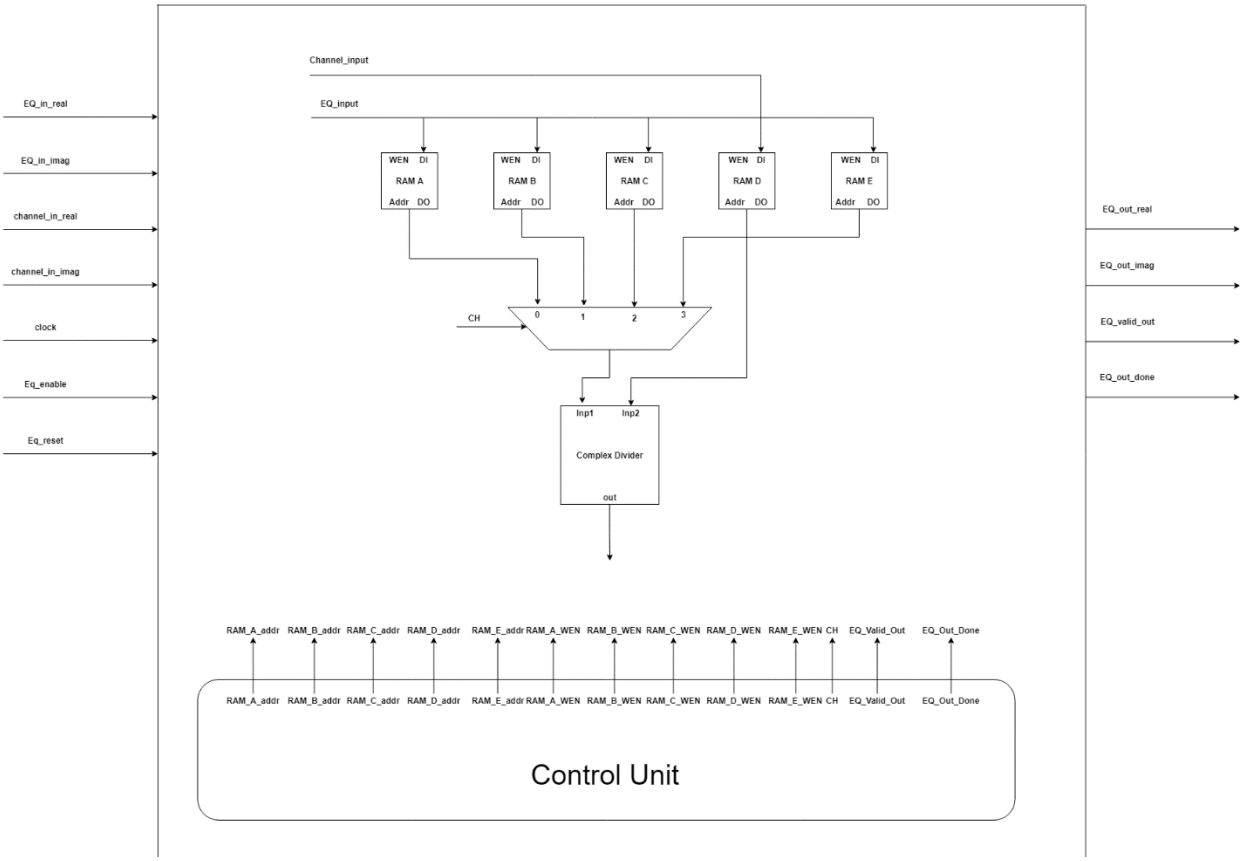


Figure 4.6-2 Architecture of the first approach for Equalizer

Control unit

Control unit is simple in the first approach; it saves the 1st three symbols waiting for the channel information to be stored. Symbol 5 is then stored and symbol 1 is out through the complex divider. Figure 4.6-3 shows the whole states of the control unit of the equalizer in first approach.

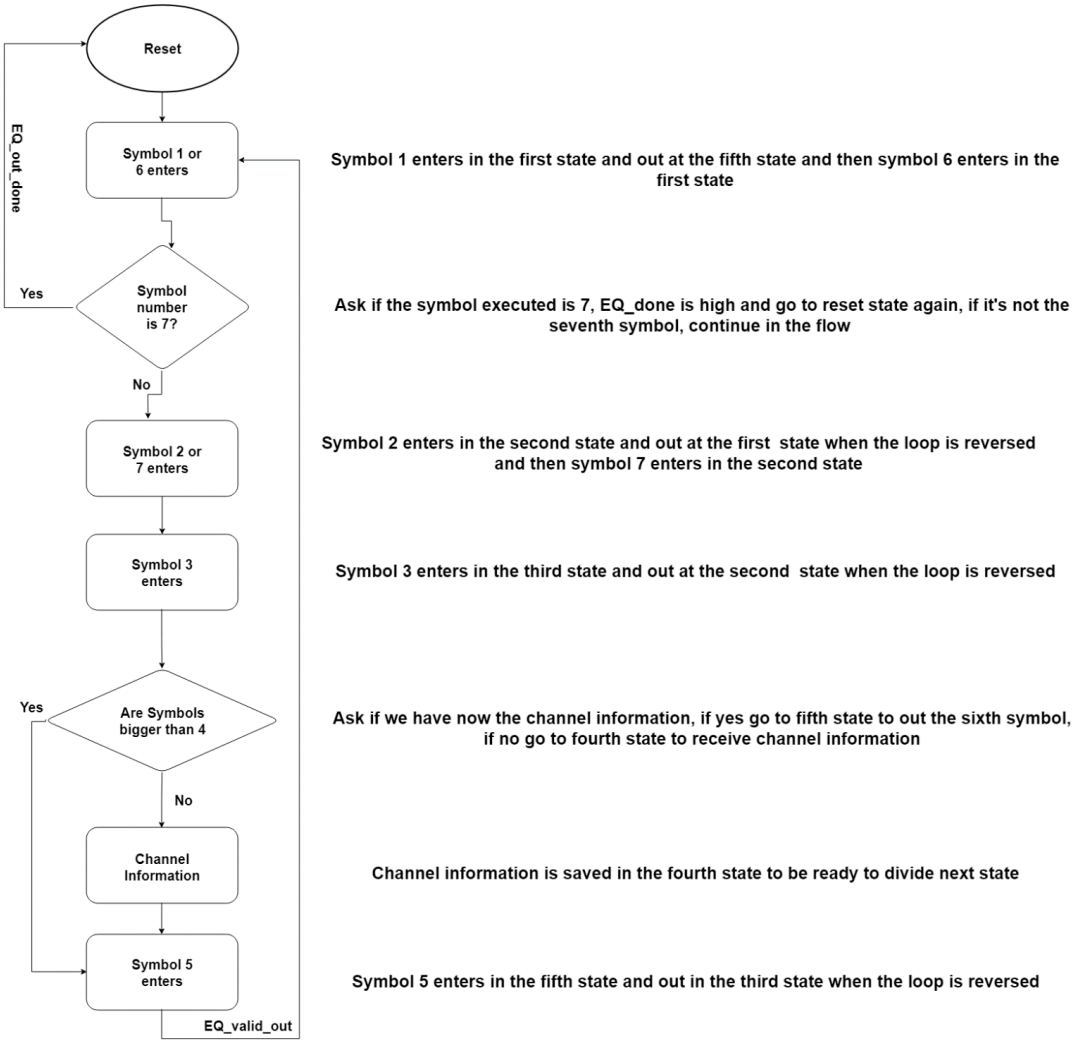


Figure 4.6-3 Flow chart for the control unit of the first approach of Equalizer

As we said, this approach out the symbols in sequential way and the IDFT block needs one symbol at a specific time because IDFT is memory-based so it takes more than one cycle to generates the output.

Second approach is to save the 7 symbols (6 symbols and the channel information), out the first symbol and wait for IDFT to end and out symbol 2 and so on. Figure 4.6-4 shows the architecture of the second approach which is the final approach of the Equalizer and Figure 4.6-5 shows the control unit of the second approach.

Second Architecture (Final Architecture)

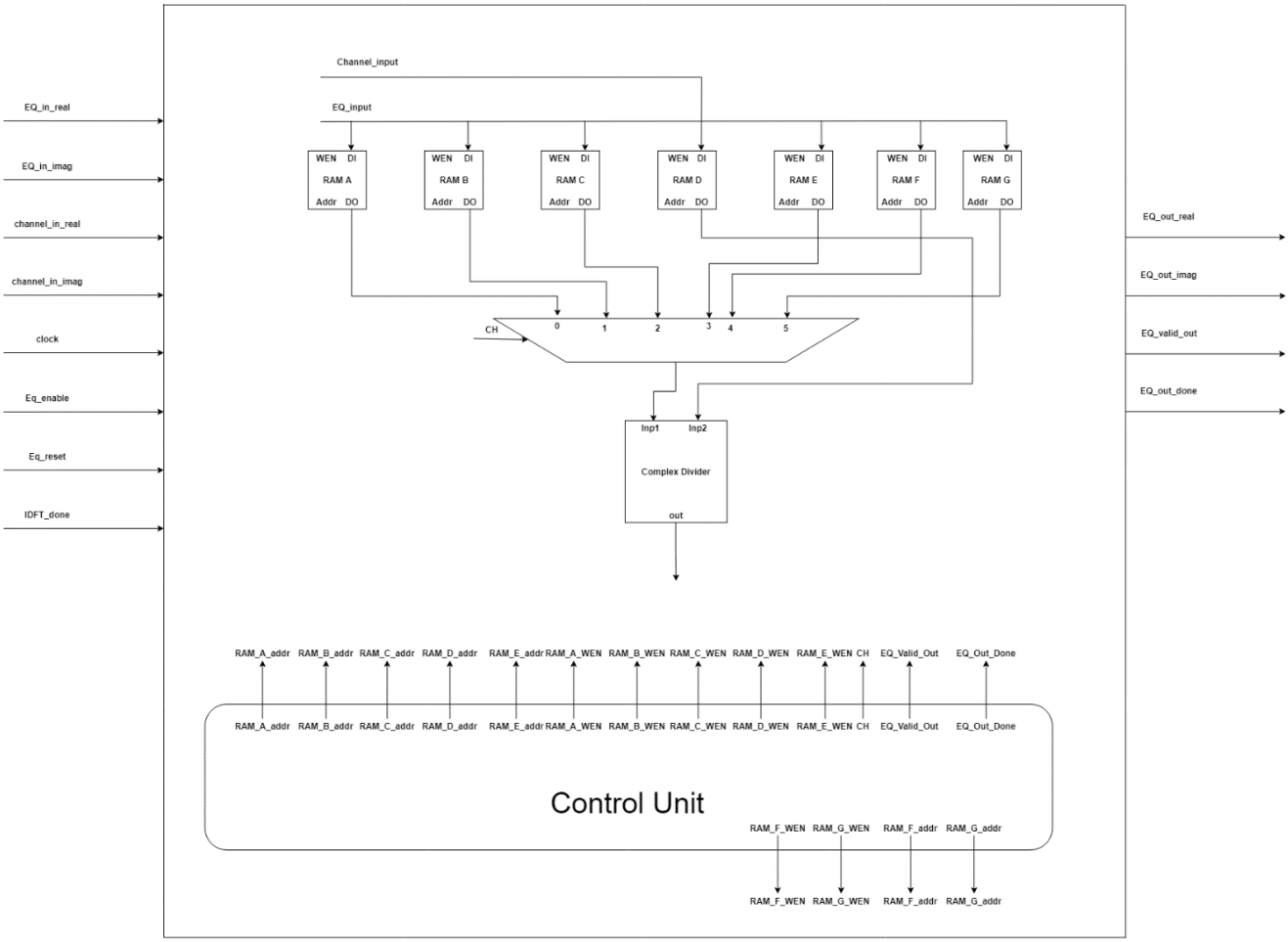


Figure 4.6-4 The Final Architecture of the Equalizer

Final approach is memory-based too but with a waiting state as shown in Figure 4.6-5. It allows the equalizer to wait for IDFT to be done and send the current symbol and so on.

Control Unit

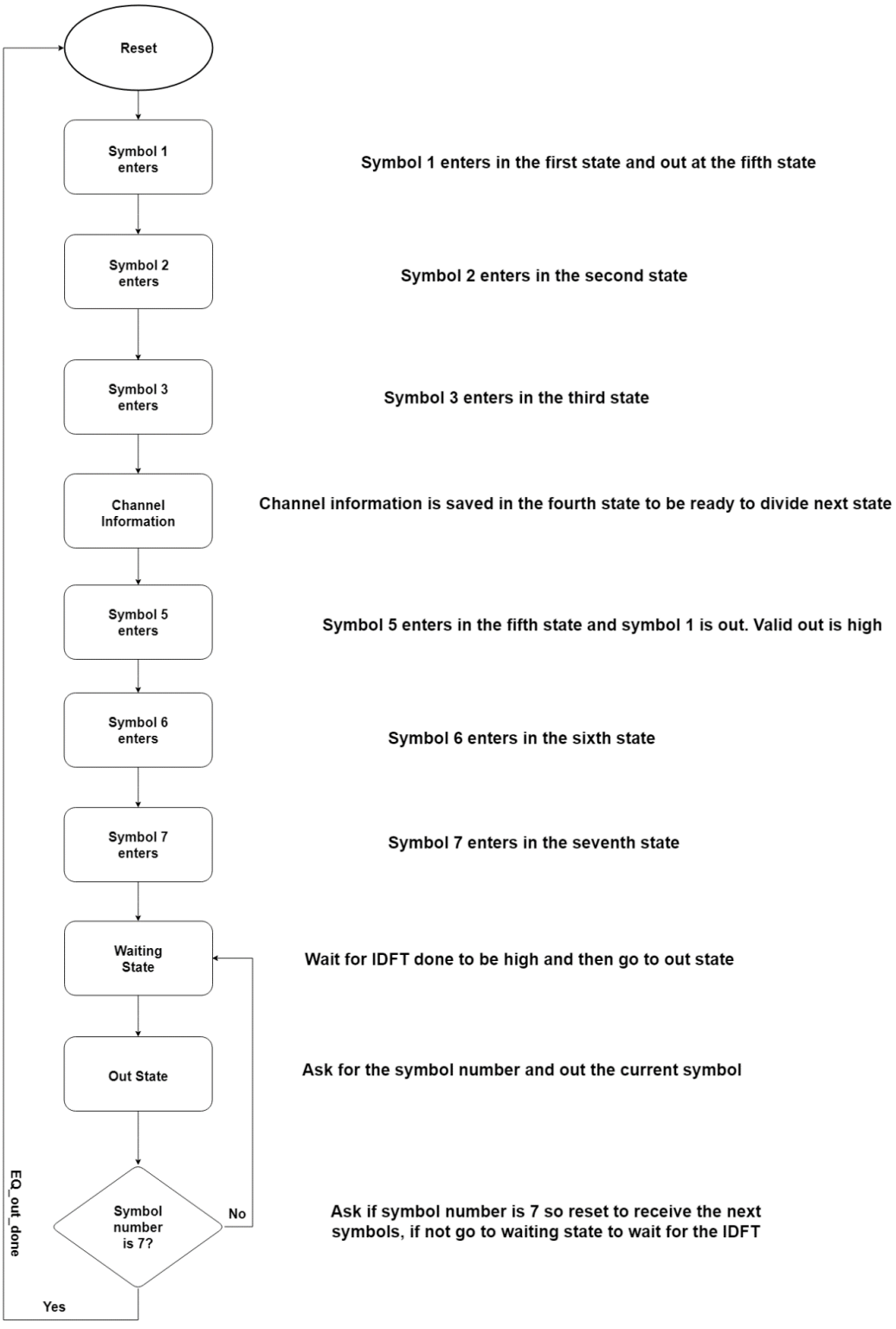


Figure 4.6-5 The Control Unit for the final approach of the equalizer

4.6.4. Synthesis results

```
****************************************
Report : area
Design : Equalizer_new
Version: B-2008.09
Date   : Tue Jul  2 19:22:50 2019
****************************************

Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File: /root/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_ss_1p08v_125c.db)

Number of ports:              102
Number of nets:               383
Number of cells:               10
Number of references:           4

Combinational area:       441704.944806
Noncombinational area:     267.110901
Net Interconnect area:     116.000000

Total cell area:          441972.055707
Total area:               442088.055707

Information: This design contains black box (unknown) components. (RPT-8)
1
```

Figure 4.6-6 Area report for the Equalizer

Area unit is μm^2. Area is very big due to the area of complex divider. Complex Divider consists of 6 multipliers and a lot of adders so it takes a big area. In general, the Equalizer block is the most complex block in the whole chain due to its function, it predicts and undo the channel effect.

```
****************************************
Report : power
        -analysis_effort low
Design : Equalizer_new
Version: B-2008.09
Date   : Tue Jul  2 19:22:59 2019
****************************************


Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File: /root/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/scmetro_tsmc_cl013g_rvt_ss_1p08v_125c.db)


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

Design        Wire Load Model            Library
------------------------------------------------
Equalizer_new          ForQA             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  = 799.6776 uW   (58%)
  Net Switching Power  = 580.0759 uW   (42%)
                         ---------
Total Dynamic Power    =   1.3798 mW  (100%)

Cell Leakage Power     = 369.5430 uW
```

Figure 4.6-7 Power report for the Equalizer

```
Divider/sub_46_I44/U74/Y (NAND2BX2M)                    0.16      94.43 r
Divider/sub_46_I44/U76/Y (NAND4X2M)                     0.16      94.59 f
Divider/sub_46_I44/U73/Y (CLKINVX2M)                    0.10      94.69 r
Divider/sub_46_I44/U71/Y (NAND4X2M)                     0.14      94.83 f
Divider/sub_46_I44/U3/Y (OAI2B1X2M)                     0.17      94.99 r
Divider/sub_46_I44/U26/Y (NOR2X2M)                      0.06      95.05 f
Divider/sub_46_I44/U211/Y (AOI31X2M)                    0.18      95.23 r
Divider/sub_46_I44/U8/Y (OAI21BX2M)                     0.10      95.33 f
Divider/sub_46_I44/U33/Y (AOI2B1X2M)                    0.13      95.46 r
Divider/sub_46_I44/U30/Y (NOR2X2M)                      0.07      95.52 f
Divider/sub_46_I44/U34/Y (NAND2XLM)                     0.08      95.60 r
Divider/sub_46_I44/U35/Y (NAND2X1M)                     0.07      95.67 f
Divider/sub_46_I44/DIFF[43] (divider_WIDTH44_DW01_sub_2448)
                                                        0.00      95.67 f
Divider/U32/Y (INVXLM)                                  0.04      95.72 r
Divider/o_real[0] (divider_WIDTH44)                     0.00      95.72 r
EQ_op_real[0] (out)                                     0.00      95.72 r
data arrival time                                                 95.72

clock EQ_clock (rise edge)                            100.00     100.00
clock network delay (ideal)                             0.00     100.00
clock uncertainty                                      -0.25      99.75
output external delay                                  -4.00      95.75
data required time                                                95.75
--------------------------------------------------------------------------
data required time                                                95.75
data arrival time                                                -95.72
--------------------------------------------------------------------------
slack (MET)                                                        0.03
```

Figure 4.6-8 Timing report for the Equalizer

4.7. Inverse Discrete Fourier Transform (IDFT)

4.7.1. Top level

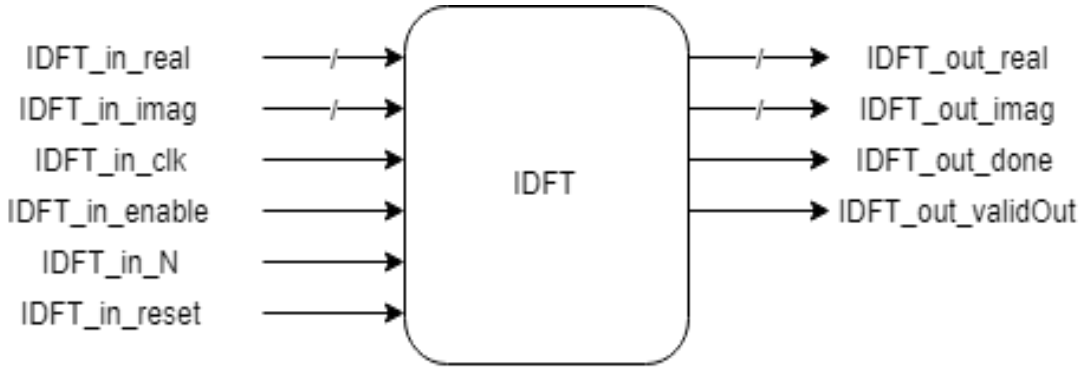


Figure 4.7-1 IDFT block Top-level

4.7.2. Block interface

Signal Name	Direction	Description	Size
IDFT_in_real	input	IDFT input real part	16
IDFT_in_imag	input	IDFT input imaginary part	16
IDFT_in_clock	input	IDFT input clock	1
IDFT_in_reset	input	IDFT reset block	1
IDFT_in_enable	input	IDFT enable block	1
IDFT_in_N	input	Choose between 1/3/6/12-IDFT operation	2
IDFT_out_real	output	IDFT output real part	16
IDFT_out_imag	output	IDFT output imaginary part	16
IDFT_out_done	output	IDFT done operation	1
IDFT_out_validOut	output	IDFT Output is valid	1

Table 4.7-1 IDFT Interface signals

4.7.3. Architecture

Memory based architecture as in [13] was implemented as it is the most suitable choice for low power implementation which is the main goal in case of IoT systems, this reduces the area, power, and test cost. Memory-based architecture usually performs the FFT in serial, i.e. one butterfly operation at a time instead of more than one in parallel, and this result in a low area cost for implementing the

memory-based FFT processor and also using single port memories which requires lower power than dual port memories the design was modified to perform IDFT.

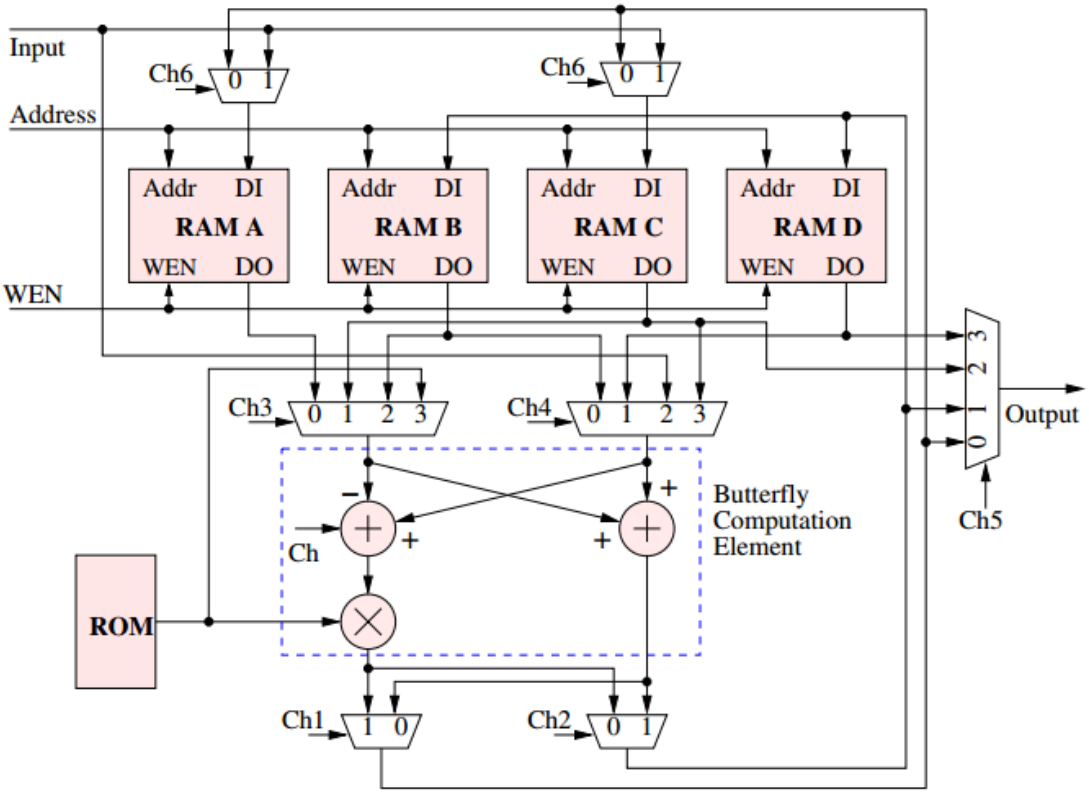


Figure 4.7-2 IDFT Memory based Architecture

As shown in figure 4.7-2, Data path of this design consists of four single port RAMs to store and compute intermediate results of the 3,6,12-point IFFT, 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 A-B. However, if Ch=1, then the adder executes B-A. A controller, generates the controlling signals for the multiplexers and the four RAMs.

The subtraction unit was modified by adding another control signal which shifts right (multiplies by 1/2) the subtrahend to satisfy the 2^{nd} stage in radix-3 which implicated that the input to the subtraction unit i.e. subtrahend should be multiplied by 1/2.

Another modification is a division unit at the output to contribute the division in the IDFT equation controlled by signal N which chooses the mode of operation 1/3/6/12-IDFT which simply is a multiplexer and a multiplier by 1/3 working as the following pseudo code.

```
If N=0
        Bypass the data to output directly
Else if N=1 (divide by 3)
        Pass the data to multiply by 1/3 then to output
Else if N=2 (divide by 6)
        Pass the data shifted right once to multiply by 1/3 then to output
Else if N=3 (divide by 12)
        Pass the data shifted right twice to multiply by 1/3 then to output
```

4.7.4. Operation

As discussed before the NB-IoT supports different number of subcarriers in format 1, 1-IDFT, 3-IDFT, 6-IDFT, and 12-IDFT therefor a mixed radix algorithm from radix-2 and radix-3 will be implemented as was stated in [13].
Figure 4.7-3 and 4.7-4 show the signal flow graph of radix-2 and radix-3.

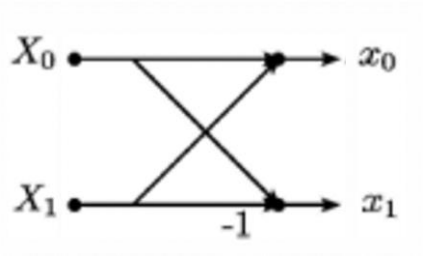


Figure 4.7-3 Radix-2 SFG

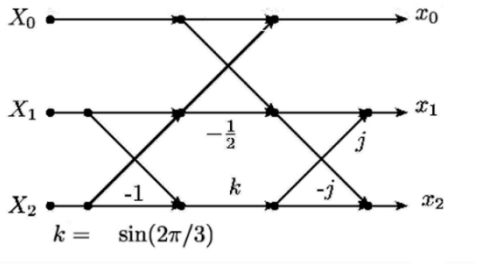


Figure 4.7-4 Radix-3 SFG

Taking 4 bits for integer value and 12 bits for fraction value for twiddle factors and output for high SQNR.

SFG for 3-IDFT, 6-IDFT and 12-IDFT are shown in Figures 4.7-5, 4.7-6 and 4.7-7 respectively.

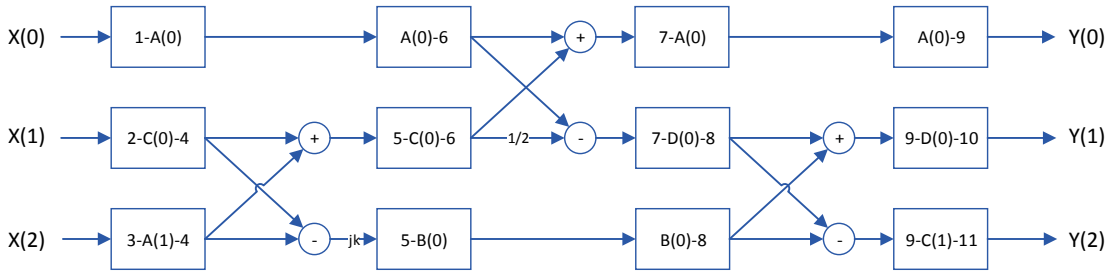
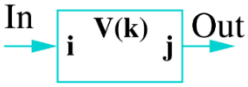


Figure 4.7-5 3-IDFT SFG

Legend:



Data in is written into the address k of the memory V at the i^{th} clock and the data of the address k of the memory V is read to Out at the j^{th} clock.

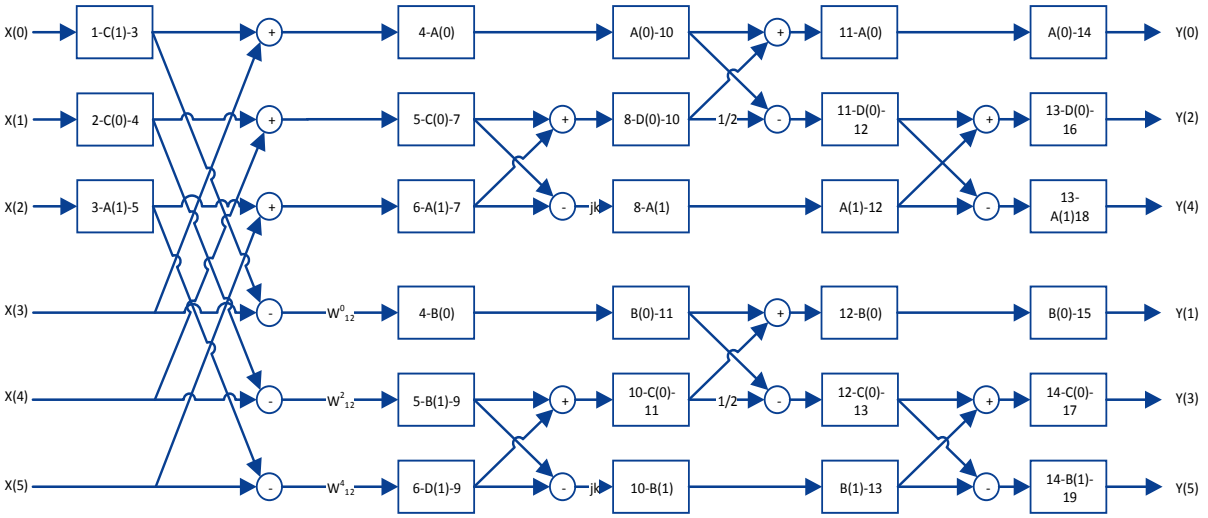


Figure 4.7-6 6-IDFT SFG

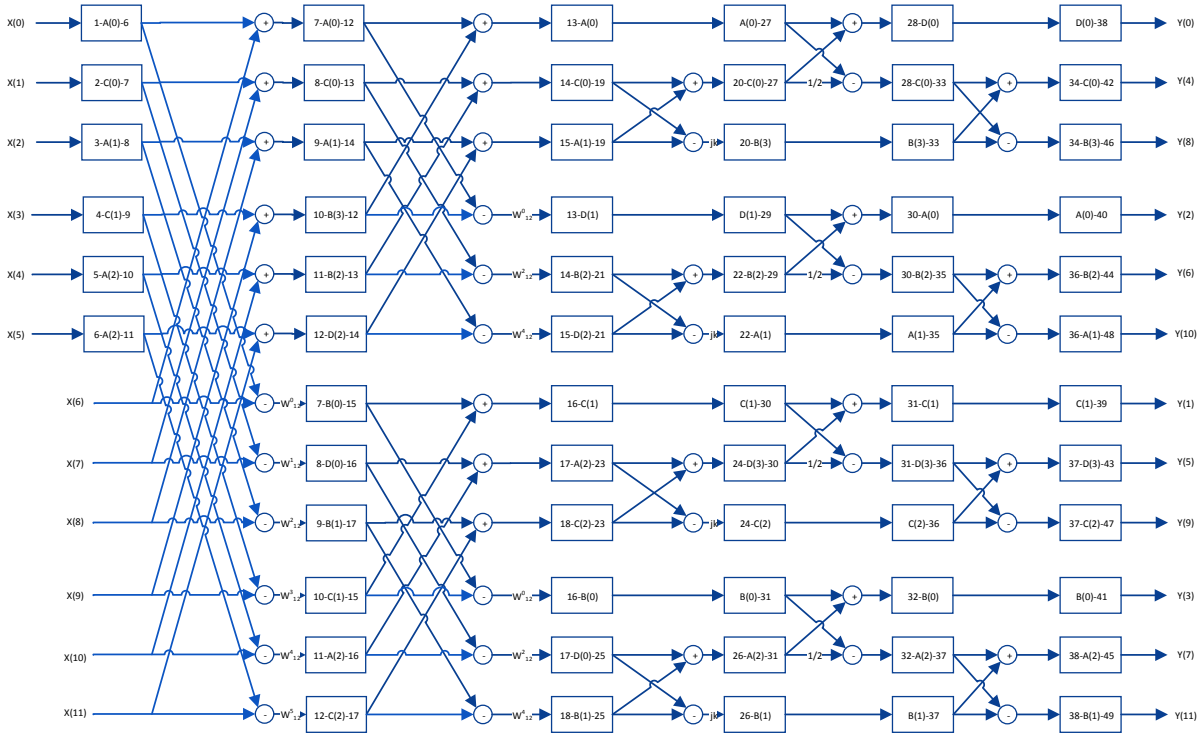


Figure 4.7-7 12-IDFT SFG

The following FSM in Figure 4.7-8 shows the operation of the IDFT block and how the 1/3/6/12-IDFT was implemented.

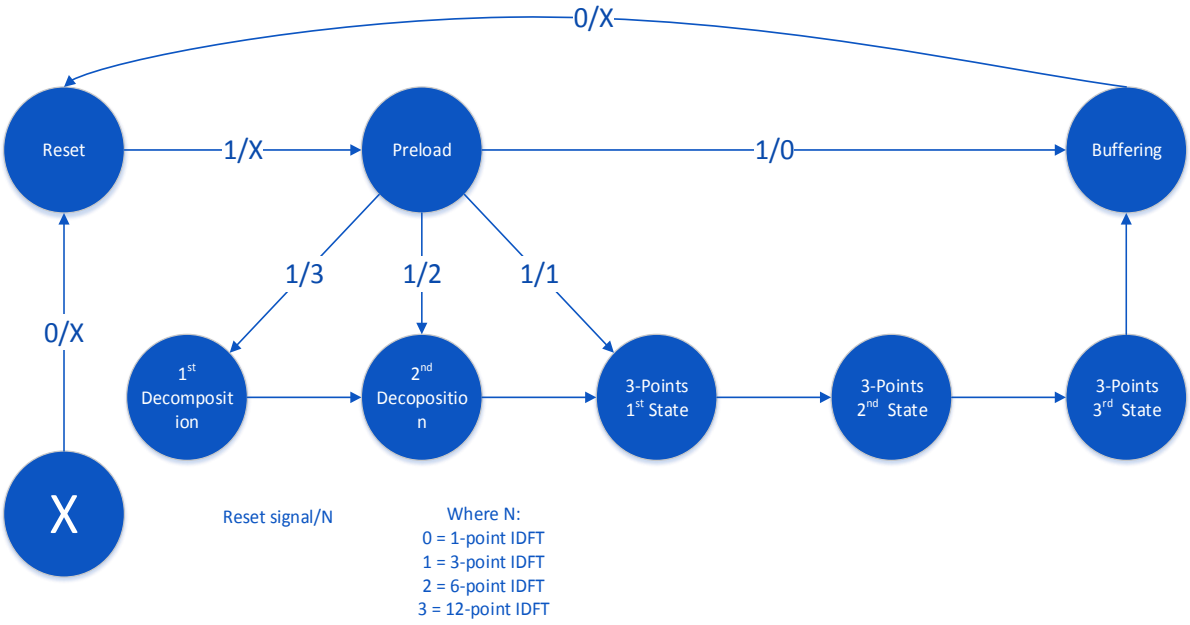


Figure 4.7-8 IDFT control unit FSM

States:

Preload state: the needed input data are loaded in memory.

Decomposition states: the radix-2 stages in case of 12-IDFT 2 stages in case of 6-IDFT just 1 stage.

3-Points states: the radix-3 stages which are included in all cases except

1-IDFT.

Buffering state: the output is reordered and delivered to next block in the chain.

4.7.5. Results

Several test cases were conducted and compared with MATLAB model shown in the figures below the output of the simulator at 1/3/6/12-IDFT.

12-IDFT simulation results on RTL and MATLAB.

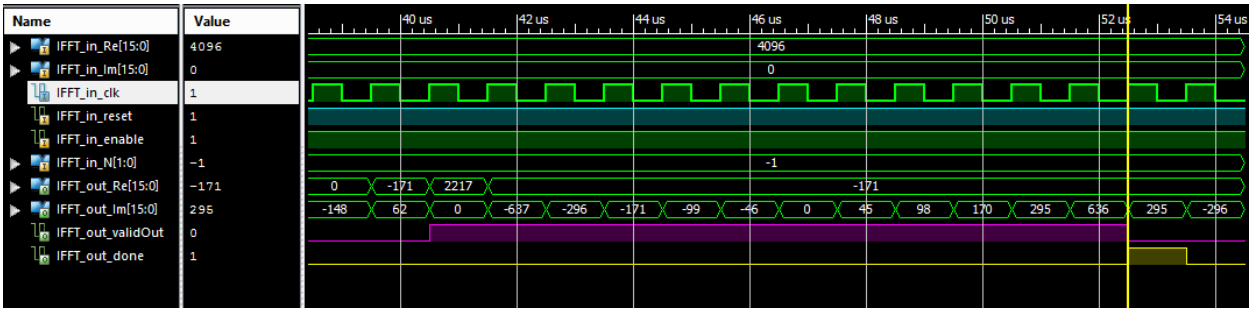


Figure 4.7-9 RTL output waveforms of IDFT block at 12-IDFT operation

1x12 complex double

	7	8	9	10	11	12
1	-171.00 + 0.0000i	-171.00 + 45.000i	-171.00 + 98.000i	-171.00 + 170.00i	-171.00 + 295.00i	-171.00 + 636.00i

1x12 complex double

	1	2	3	4	5	6
1	2218.0 + 0.0000i	-171.00 - 637.00i	-171.00 - 296.00i	-171.00 - 171.00i	-171.00 - 99.000i	-171.00 - 46.000i

Figure 4.7-10 MATLAB output of IDFT model at 12-IDFT operation

6-IDFT simulation results on RTL and MATLAB

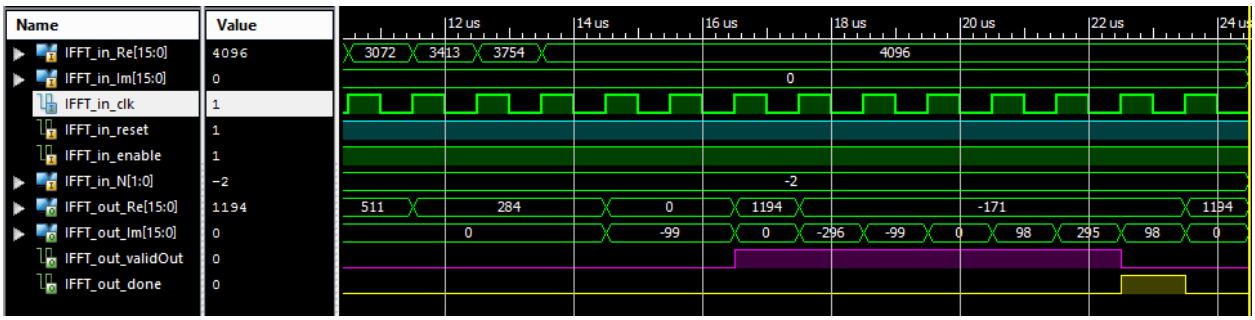


Figure 4.7-11 RTL output waveforms of IDFT block at 6-IDFT operation

1x6 complex double

	1	2	3	4	5	6
1	1194.0 + 0.0000i	-171.00 - 296.00i	-171.00 - 99.000i	-171.00 + 0.0000i	-171.00 + 98.000i	-171.00 + 295.00i

Figure 4.7-12 MATLAB output of IDFT model at 6-IDFT operation

3-IDFT simulation results on RTL and MATLAB

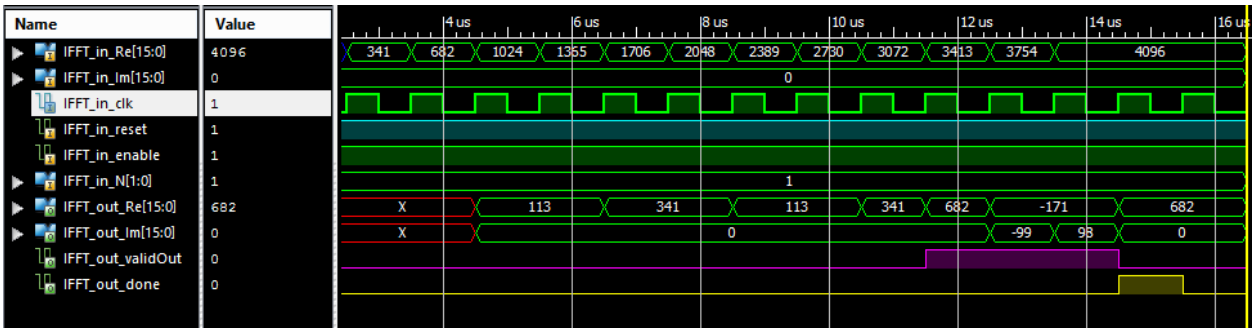


Figure 4.7-13 MATLAB output of IDFT model at 3-IDFT operation

1x3 complex double

	1	2	3
1	682.00 + 0.0000i	-171.00 - 99.000i	-171.00 + 98.000i

Figure 4.7-14 RTL output waveforms of IDFT block at 3-IDFT operation

1-IDFT simulation results on RTL and MATLAB

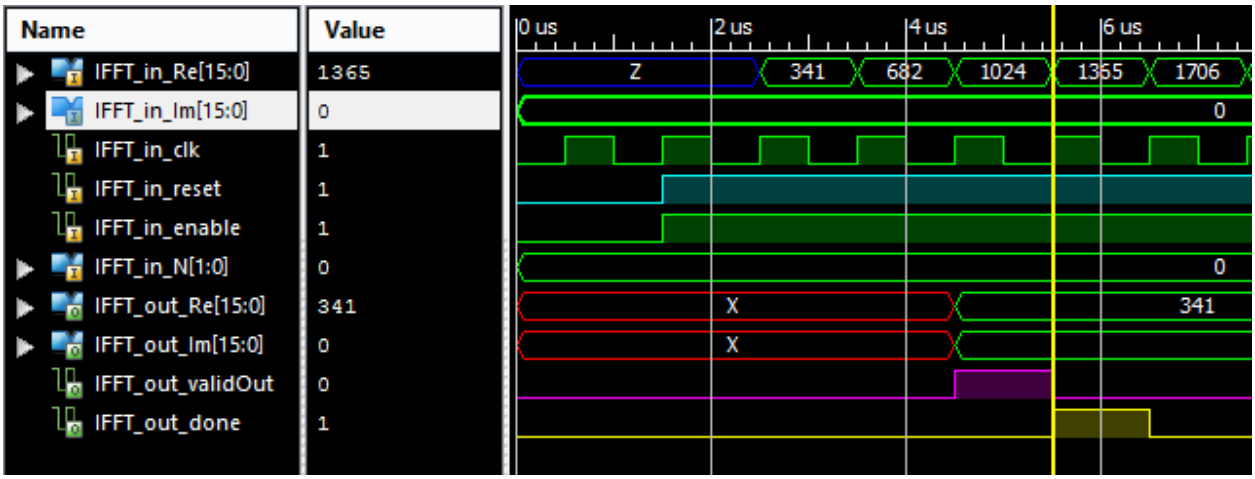


Figure 4.7-15 RTL output waveforms of IDFT block at 1-IDFT operation

1x1 double

	1
1	341

Figure 4.7-16 MATLAB output of IDFT model at 1-IDFT operation

4.7.6. Synthesis

Shown below the area, power and timing reports

```
Number of ports:                71
Number of nets:                513
Number of cells:                15
Number of references:           15

Combinational area:       64112.500082
Noncombinational area:    15251.208467
Net Interconnect area:      713.000000

Total cell area:          79363.708550
Total area:               80076.708550
```

Figure 4.7-17 Area report of IDFT Block

```
---------------------------------------------------------------------------
data required time                                                   99.55
data arrival time                                                   -22.52
---------------------------------------------------------------------------
slack (MET)                                                          77.04
```

Figure 4.7-18 Timing report of IDFT Block

```
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  = 178.6911 uW   (95%)
  Net Switching Power  =   9.4807 uW    (5%)
                         ---------
Total Dynamic Power    = 188.1718 uW  (100%)

Cell Leakage Power     =  61.7314 uW
```

Figure 4.7-19 Power Report for IDFT Block

4.8. De-mapper & FIFO

4.8.1. De-mapper

4.8.1.1. Top level

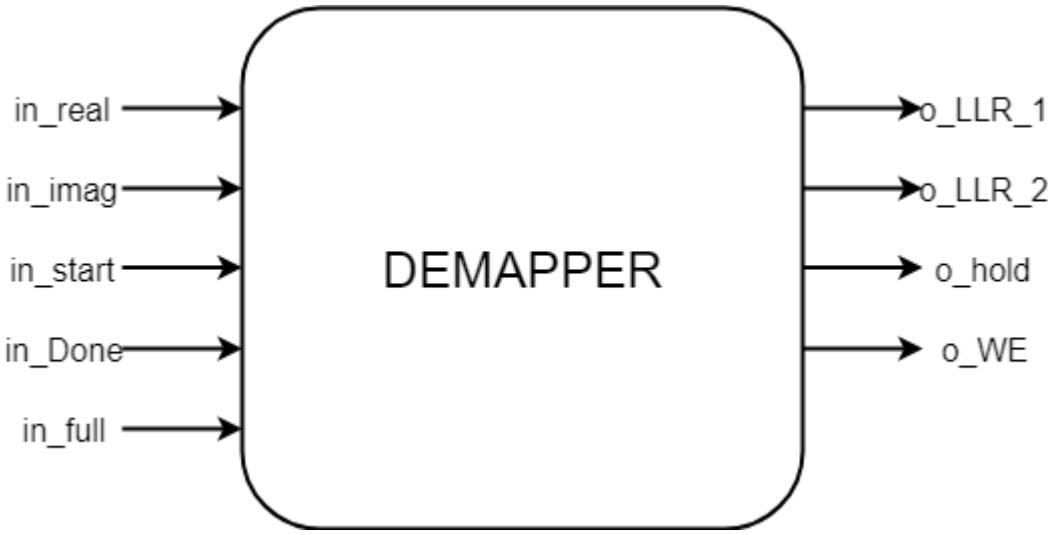


Figure 4.8-1 De-mapper block diagram

4.8.1.2. Block interfaces

Signal Name	Direction	Description	Size
in_real	input	real part input from IDFT	8
in_imag	input	imag part input from IDFT	8
in_start	input	indicates valid data	1
in_Done	input	indicates that data transmission is over	1
in_full	input	From FIFO to hold operation	1
o_LLR_1	output	bit-0 soft output to FIFO to be saved	8
o_LLR_2	output	bit-1 soft output to FIFO to be saved	8
o_hold	output	from De-mapper to IDFT to hold operation	1
o_WE	output	to enable writing data to FIFO	1

Table 4.8-1 De-mapper interfaces

4.8.1.3. Operation

This block is a combinational block that takes I and Q values from IDFT and output the probability of the bit represented in 8 bits. Negative values represent zeros probability and positive values represent one's probability. Taking the advantage of symmetry in the QPSK and BPSK constellations, the imaginary part (Q) can be considered as the probability of bit [0] and the real part to be considered as the probability of bit (1). So, for BPSK only the real part is containing information, however for QPSK both real and imaginary are containing information.

4.8.1.4. Simulation Results

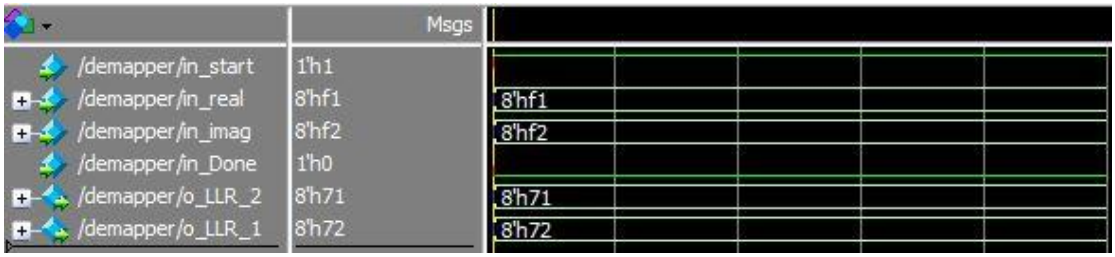


Figure 4.8-2 De-mapper RTL results

4.8.1.5. Synthesis results

```
Combinational area:          15.360000
Noncombinational area:        0.000000
Net Interconnect area:       undefined  (Wire load has zero net area)

Total cell area:             15.360000
Total area:                  undefined
```

Figure 4.8-3 Area Report of the De-mapper

```
                          Cell      Driven Net   Tot Dynamic      Cell
                          Internal  Switching    Power            Leakage
Cell                      Power     Power        (% Cell/Tot)     Power      Attrs
--------------------------------------------------------------------------------
U4                        4.234e-04 7.259e-03    7.68e-03 (6%)    2099.4250
U5                        1.407e-04 7.332e-03    7.47e-03 (2%)    1206.7000
U6                        1.407e-04 7.332e-03    7.47e-03 (2%)    1206.7000
--------------------------------------------------------------------------------
Totals (3 cells)           704.743nW  21.923uW     22.627uW (3%)     4.513nW
```

Figure 4.8-4 Power report of the De-mapper

4.8.2. FIFO

4.8.2.1. Top level

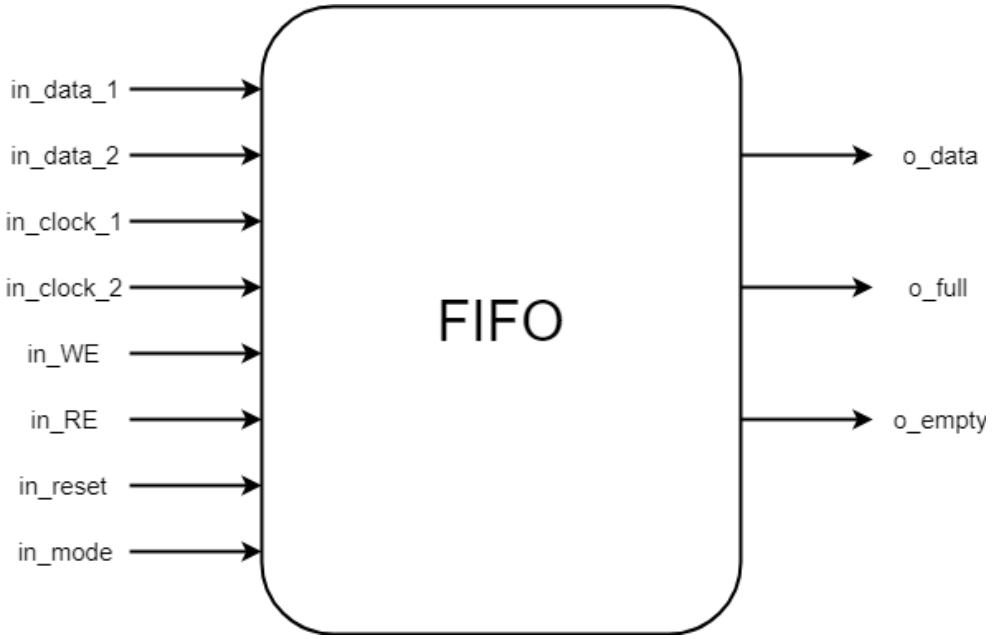


Figure 4.8-5 FIFO top level

4.8.2.2. Block interfaces

Signal Name	Direction	Description	Size
in_data_1	input	input data from De-mapper (real real)	8
in_data_2	input	input data from De-mapper (imag part)	8
in_clock_1	input	write clock (slow one)	1
in_clock_2	input	read clock (fast one)	1
in_WE	input	Write enable	1
in_RE	input	Read enable	1
in_mode	input	0 for BPSK and 1 for QPSK	1
in_reset	input	reset signal (active low)	1
o_data	output	Output data to the descrambler	8
o_full	output	to indicates that the FIFO is full	1
o_empty	output	to indicates that the FIFO is empty	1

Table 4.8-2 FIFO interfaces

4.8.2.3. Architecture

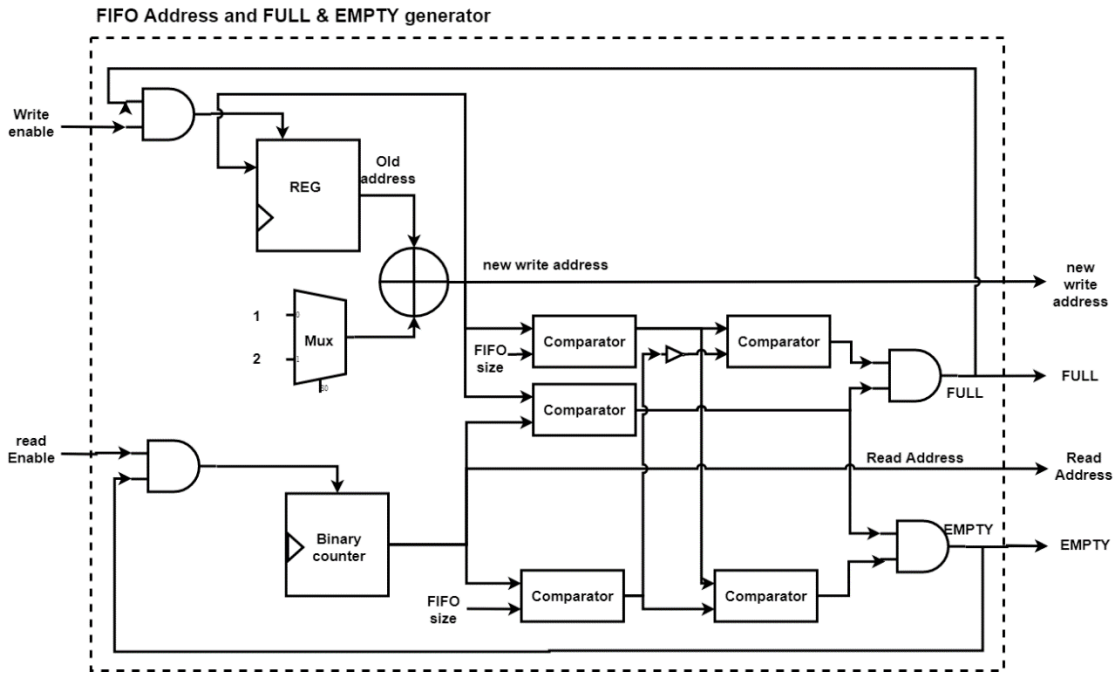


Figure 4.8-6 FIFO Architecture

4.8.2.4. Operation

The FIFO block is added to solve the CDC problem as the QPSK contains 2 bits so the rate is different before and after the de-mapper. The de-mapper converts the symbol into 2 bits and writes them in the FIFO in parallel, as the FIFO has 2 input ports. Then the descrambler activates the read enable to read the data bits one by one using higher frequency clock.

The FIFO has 2 signals to indicate either empty, full or none of them. When the FIFO is full a hold signal is activated to hold the de-mapper until the descrambler read some data. Also the descrambler operation is stopped if the FIFO is empty. The FIFO is a circular FIFO to make the best use of the memory.

4.8.2.5. Simulation results

Comparing the FIFO input with the output operating with different frequencies:

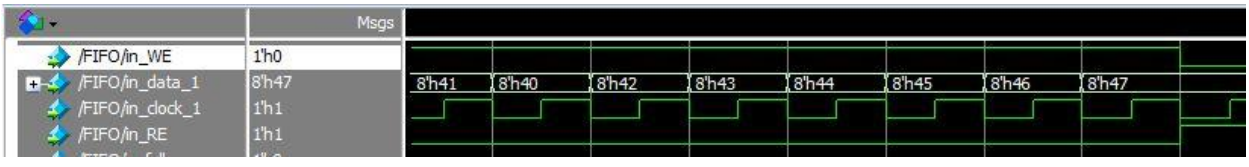


Figure 4.8-7 FIFO input data

Figure 4.8-8 FIFO output

4.8.2.6. Synthesis report

```
Combinational area:       4047.360044
Noncombinational area:    5148.159824
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          9195.519869
Total area:                 undefined
```

Figure 4.8-9 FIFO area report

Cell	Cell Internal Power	Driven Net Switching Power	Tot Dynamic Power (% Cell/Tot)	Cell Leakage Power	Attrs
read_loc_reg[0]	3.104e-04	1.768e-05	3.28e-04 (95%)	8576.6055	
read_loc_reg[1]	2.825e-04	1.503e-05	2.98e-04 (95%)	8576.6816	
o_data_reg[0]	2.718e-04	6.287e-05	3.35e-04 (81%)	7941.3418	
o_data_reg[1]	2.714e-04	6.135e-05	3.33e-04 (82%)	7941.7642	
o_data_reg[3]	2.713e-04	6.126e-05	3.33e-04 (82%)	7941.0991	
o_data_reg[4]	2.713e-04	6.104e-05	3.32e-04 (82%)	7941.5391	
Totals (461 cells)	25.662uW	N/A	N/A (N/A)	2.645uW	

Figure 4.8-10 FIFO power report

```
clock clk_0 (rise edge)                          100.00      100.00
clock network delay (ideal)                        0.00      100.00
clock uncertainty                                 -0.05       99.95
Mem_array_reg[0][0]/CK (DFECHD)                    0.00       99.95 r
library setup time                                -0.20       99.75
data required time                                            99.75
-----------------------------------------------------------------------
data required time                                            99.75
data arrival time                                             -1.46
-----------------------------------------------------------------------
slack (MET)                                                   98.29
```

Figure 4.8-11 FIFO timing report for clock 1

```
clock clk_1 (rise edge)                          50.00       50.00
clock network delay (ideal)                        0.00      50.00
clock uncertainty                                 -0.05       49.95
turn_read_reg/CK (DFECHD)                          0.00       49.95 r
library setup time                                -0.11       49.84
data required time                                            49.84
-----------------------------------------------------------------------
data required time                                            49.84
data arrival time                                             -1.23
-----------------------------------------------------------------------
slack (MET)                                                   48.62
```

Figure 4.8-12 FIFO timing report for clock 2

4.9. Descrambler

4.9.1. Top level

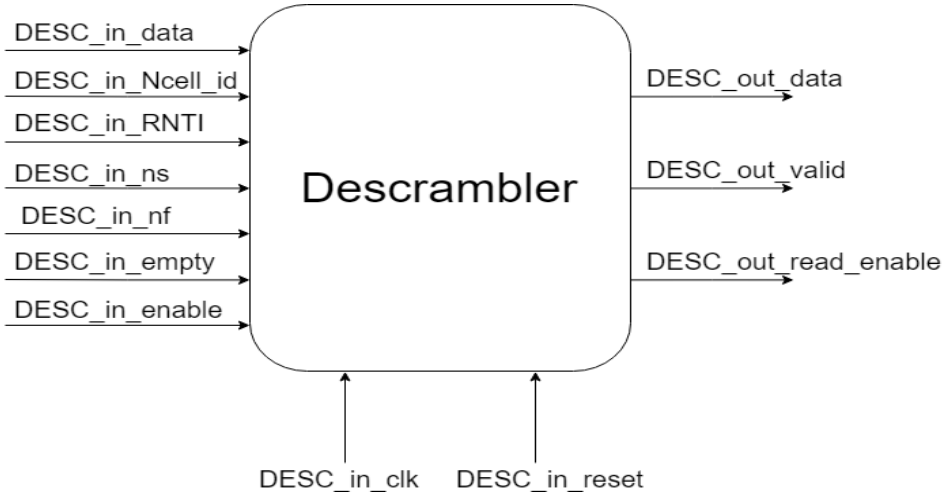


Figure 4.9-1 Descrambler block diagram

4.9.2. Block interface

Signal name	Direction	Description	Size (bits)
DESC_in_data	input	input bits from De-mapper	8
DESC_in_Ncell_id	input	Cell ID(upper layer parameter)	9
DESC_in_RNTI	input	Radio Network Identifier(upper layer)	16
DESC_in_ns	input	First slot (upper layer)	4
DESC_in_nf	input	First frame(upper layer)	1
DESC_in_clk	input	clock	1
DESC_in_reset	input	reset signal	1
DESC_in_empty	input	Input from De-mapper indicates that the FIFO is not empty	1
DESC_in_enable	input	Enable signal	1
DESC_out_data	output	Out data to the DE rate matching	8
DESC_out_valid	output	Indicates that the out data is valid	1
DESC_out_read_enable	output	Read enable signal to the De-mapper to enable reading from the FIFO	1

Table 4.9-1 Descrambler interface signals

4.9.3. Architecture

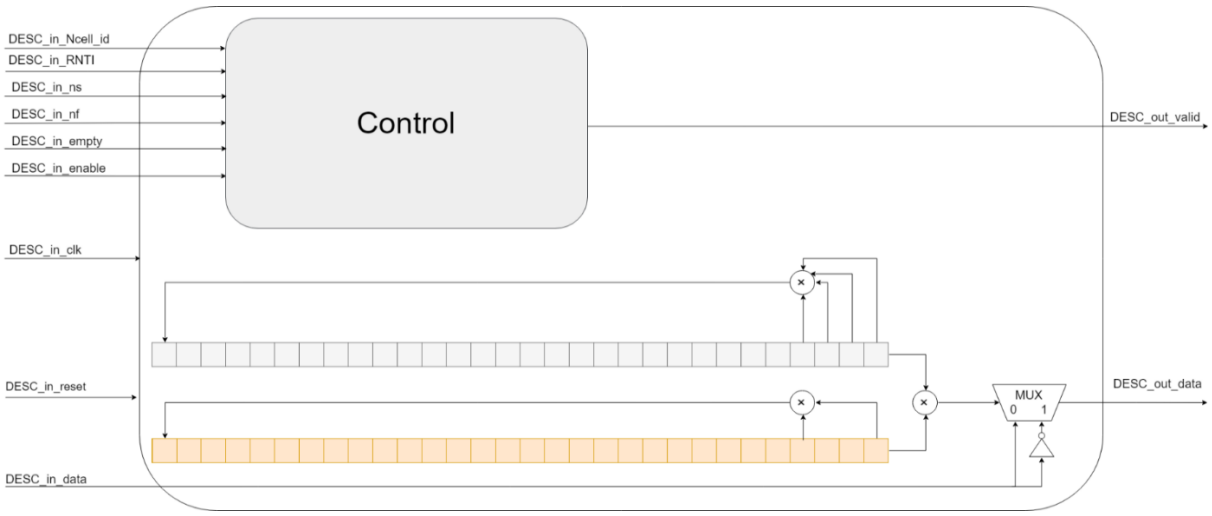


Figure 4.9-2 Scrambler architecture

4.9.4. Operation

- The control part calculates the initializations for each LFSR according to the upper layer input parameters.
- The difference in the scrambler between the uplink and downlink is that in the uplink the descrambler receives soft input from the De-mapper not hard input and it also outputs soft output.
- The Golden sequence is generated bit by bit by xoring the first two bits of the LFSRs
- Instead of xor gate at the output, there's a MUX that outputs the input as it is or flipped according to the scrambling bit.
- Polynomial of LFSR1 = $1 + D + D^2 + D^3 + D^{31}$
- Polynomial of LFSR2 = $1 + D^3 + D^{31}$

4.9.5. Results

4.9.5.1. RTL results

The descrambler was tested at the following upper layer parameters

- RNTI =65535
- NcellID = 504
- nf = 1
- ns = 28

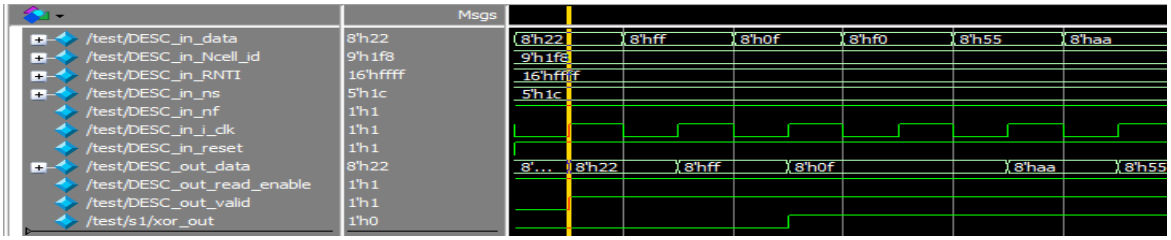


Figure 4.9-3 RTL results of descrambler block

4.9.5.2. Synthesis results

```
Combinational area:          878.994915
Noncombinational area:      2058.048328
Net Interconnect area:       123.000000

Total cell area:            2937.043243
Total area:                 3060.043243
```

Figure 4.9-4 Descrambler area report

```
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  =  17.9604 uW   (98%)
  Net Switching Power  = 404.7792 nW    (2%)
                         ---------
Total Dynamic Power    =  18.3652 uW  (100%)

Cell Leakage Power     =   1.5331 uW
```

Figure 4.9-5 Descrambler power report

```
 ------------------------------------------------------------
 data required time                                    99.43
 data arrival time                                     -4.00
 ------------------------------------------------------------
 slack (MET)                                           95.43
```

Figure 4.9-6 Descrambler total slack

4.10. Data De-multiplexing and Channel De-interleaver

4.10.1. Top level

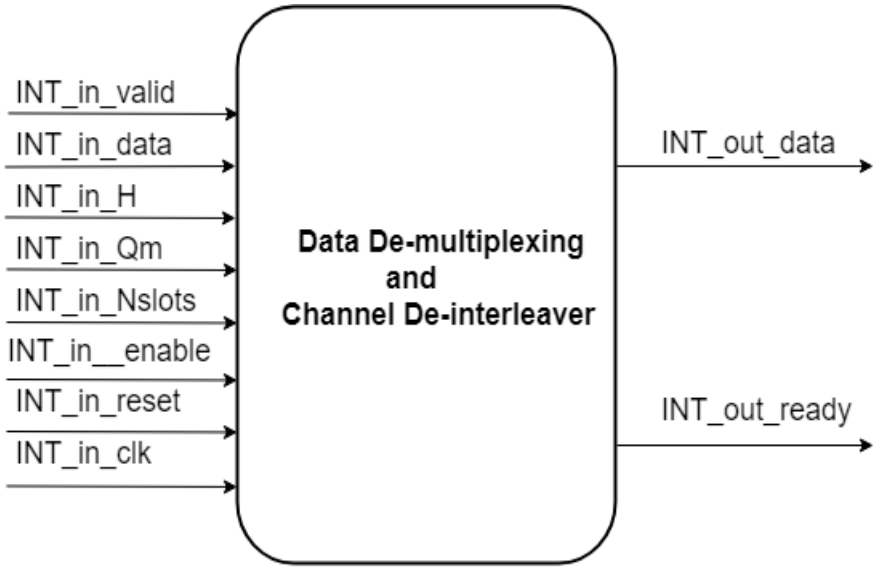


Figure 4.10-1 Data De-multiplexing and Channel De-interleaver block diagram

4.10.2. Block interface

Signal name	Direction	Description	Size
INT_in_data	input	Data In for channel de-interleaver	8
INT_in_H	input	The number of modulation symbols	13
INT_in_Qm	input	Modulation order parameter	1
INT_in_slots	input	The number of columns of the matrix	4
INT_in_reset	input	Reset for channel de-interleaver	1
INT_in_clk	input	Clock for channel de-interleaver	1
INT_in__enable	input	Enable signal	1
INT_in__valid	input	valid data signal	1
INT_out_data	output	Data Out for channel de-interleaver	8
INT_out_ready	output	Ready flag	1

Table 4.10-1 Data De-multiplexing and Channel De-interleaver interface signals

4.10.3. Architecture

4.10.3.1. Data flow according to standard description

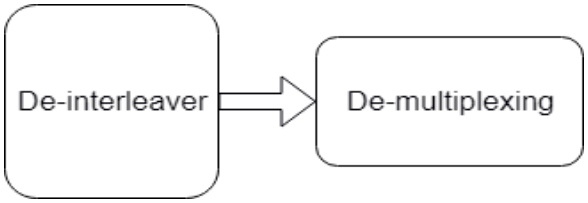


Figure 4.10-2 Basic Data flow for Data De-multiplexing and Channel De-interleaver according to standard

Design consists of:

- De-interleaver Block supports BPSK and QPSK modes, also handles variable number of columns and rows.
- De-multiplexing Block takes one stream in BPSK Case and pass it on, and takes two streams and combined them in one in QPSK case.

4.10.3.2. Our proposed Architecture

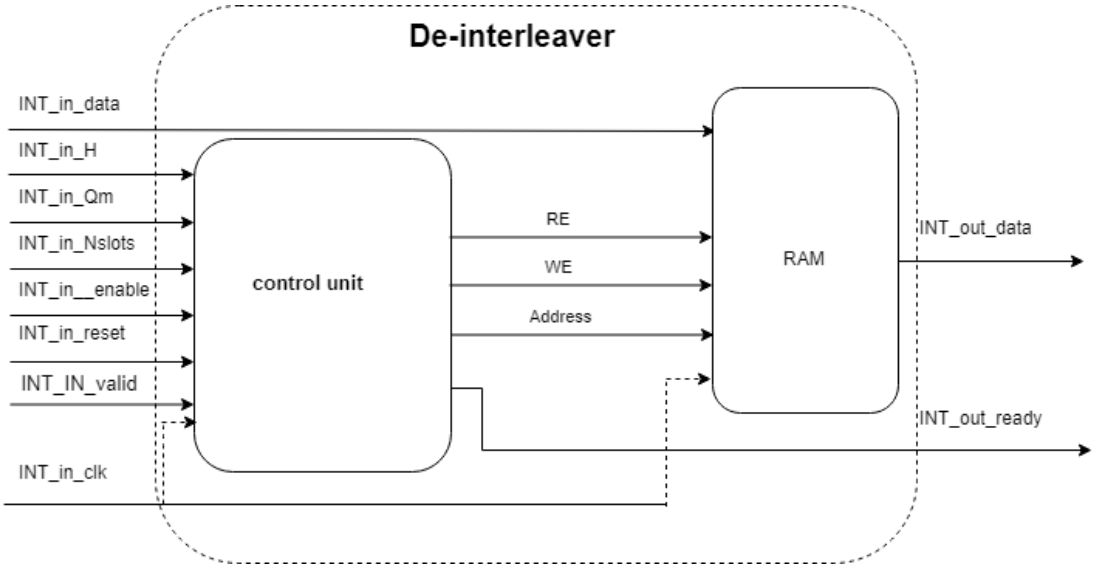


Figure 4.10-3 Our proposed architecture for Data De-multiplexing and Channel De-interleaver

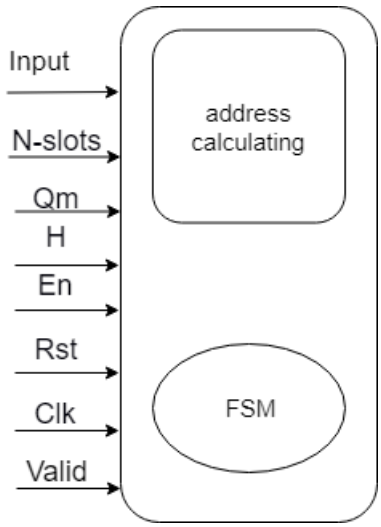


Figure 4.10-4 Architecture control unit for Data De-multiplexing and Channel De-interleaver

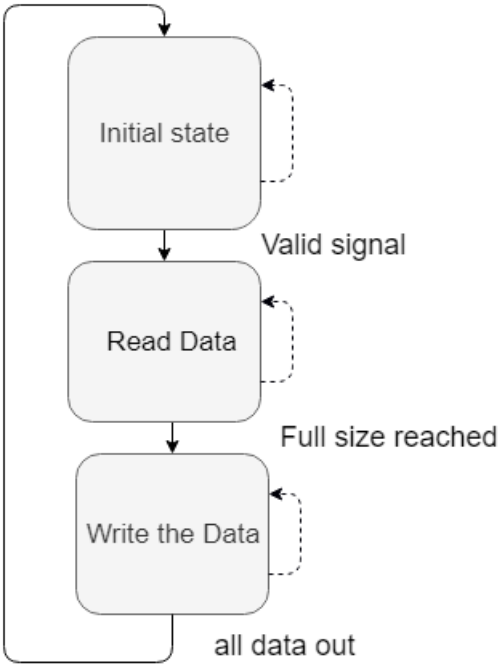


Figure 4.10-5 Flow chart of control unit for Data De-multiplexing and Channel De-interleaver

The Design consists of:

- One RAM support Data size starts from 144*8bit and multiple of that size
  As 144 referred to the data exists in one frame and * (8 bits) to support soft Data.
- Control unit to perform the de-multiplexing and de-interleaving operations.

4.10.3.2.1. Control unit

Control unit consists of:

- Finite State Machine (FSM): is the brain of the block and it handles controlling signal for all the memory and address calculating unit required in each state.
- Address calculating unit: create the address or manipulate the existing address to calculate the new address.
  Multi-operation could happen on the address at the same clock cycle.
  Containing adders/subtractor, comparators and one multiplier Block
  This unit changes the de-multiplexing and de-interleaving sequence according to the upper layer parameters Qm which determine BPSK or QPSK Case and the Number of slots which change number of columns and rows.

4.10.3.2.2. Structural issues in our proposed architecture

1. We merge the de-multiplexing process and the de-interleaving process together.
2. We calculate the address in BPSK case every clock cycle, and in QPSK case we calculated it at the first clock cycle and write the data of the second clock cycle at the address+1 then we calculate the new address again at the new first clock cycle.

4.10.3.3. Another proposed Architecture

Same pervious architecture but we write in the memory column by column and read it row by row to reduce the complexity of address calculations but this design need S/P converter at the input data path and P/S converter at the output data path.

But such solution will need that memory to has maximum dimension in both columns and rows and such memory will consume large area and power and there is need for that as there always large part of it will never be used .

4.10.4. Operation according to previous FSM

1. Get the valid signal and the upper layer parameters.
2. Calculate the number of columns and the rows .
3. Start filling the RAM according the sequence required in this case of parameters.
4. If the RAM is fully filled start the write process .
5. If all data out wait till you get new valid signal and upper layer parameters to start working again .

4.10.5. Challenges and enhancement

- Merge the de-multiplexing process and the de-interleaving process together decrease power and area.

4.10.6. Testing Results

Results in decimal form

4.10.6.1. Test case1

- QM=1(QPSK case)
- Number of slots=2 (columns=12)
- H=144

4.10.6.1.1. RTL Simulation result

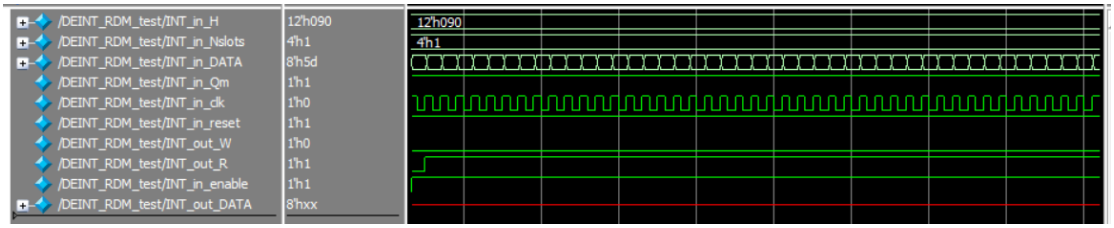


Figure 4.10-6 RTL wave form for test case 1 in Data De-multiplexing and Channel De-interleaver

0	1	2	13	14	25	26	37	38	49	50	61	62	73	74	85	86	97	98	109	110
20	121	122	133	134	3	4	15	16	27	28	39	40	51	52	63	64	75	76	87	88
40	99	100	111	112	123	124	135	136	5	6	17	18	29	30	41	42	53	54	65	66
60	77	78	89	90	101	102	113	114	125	126	137	138	7	8	19	20	31	32	43	44
80	55	56	67	68	79	80	91	92	103	104	115	116	127	128	139	140	9	10	21	22
100	33	34	45	46	57	58	69	70	81	82	93	94	105	106	117	118	129	130	141	142
120	11	12	23	24	35	36	47	48	59	60	71	72	83	84	95	96	107	108	119	120
140	131	132	143	144	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.10-7 RTL memory data for test case 1 in Data De-multiplexing and Channel De-interleaver

4.10.6.1.2. MATLAB Simulation result

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	2	13	14	25	26	37	38	49	50	61	62	73	74	85	86	97	98	109	110
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	121	122	133	134	3	4	15	16	27	28	39	40	51	52	63	64	75	76	87	88
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	99	100	111	112	123	124	135	136	5	6	17	18	29	30	41	42	53	54	65	66
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	77	78	89	90	101	102	113	114	125	126	137	138	7	8	19	20	31	32	43	44
	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	55	56	67	68	79	80	91	92	103	104	115	116	127	128	139	140	9	10	21	22
	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
1	33	34	45	46	57	58	69	70	81	82	93	94	105	106	117	118	129	130	141	142
	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
1	11	12	23	24	35	36	47	48	59	60	71	72	83	84	95	96	107	108	119	120
	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
1	131	132	143	144																

Figure 4.10-8 MATLAB memory data for test case 1 in Data De-multiplexing and Channel De-interleaver

4.10.6.2. Test case2

- QM=0(BPSK case)
- Number of slots=4 (columns=24)
- H=144

4.10.6.2.1. RTL Simulation result

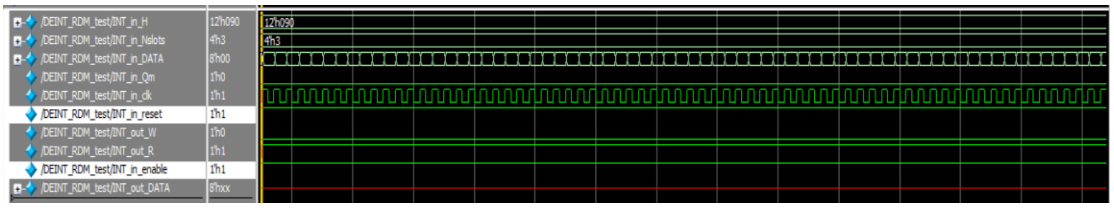


Figure 4.10-9 RTL wave form for test case 2 in Data De-multiplexing and Channel De-interleaver

0	1	7	13	19	25	31	37	43	49	55	61	67	73	79	85	91	97	103	109	115
20	121	127	133	139	2	8	14	20	26	32	38	44	50	56	62	68	74	80	86	92
40	98	104	110	116	122	128	134	140	3	9	15	21	27	33	39	45	51	57	63	69
60	75	81	87	93	99	105	111	117	123	129	135	141	4	10	16	22	28	34	40	46
80	52	58	64	70	76	82	88	94	100	106	112	118	124	130	136	142	5	11	17	23
100	29	35	41	47	53	59	65	71	77	83	89	95	101	107	113	119	125	131	137	143
120	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
140	126	132	138	144	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.10-10 RTL memory data for test case 2 in Data De-multiplexing and Channel De-interleaver

4.10.6.2.2. MATLAB Simulation result

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	7	13	19	25	31	37	43	49	55	61	67	73	79	85	91	97	103	109	115
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	121	127	133	139	2	8	14	20	26	32	38	44	50	56	62	68	74	80	86	92
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	98	104	110	116	122	128	134	140	3	9	15	21	27	33	39	45	51	57	63	69
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	75	81	87	93	99	105	111	117	123	129	135	141	4	10	16	22	28	34	40	46
	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
1	52	58	64	70	76	82	88	94	100	106	112	118	124	130	136	142	5	11	17	23
	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
1	29	35	41	47	53	59	65	71	77	83	89	95	101	107	113	119	125	131	137	143
	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140
1	6	12	18	24	30	36	42	48	54	60	66	72	78	84	90	96	102	108	114	120
	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
1	126	132	138	144																

Figure 4.10-11 MATLAB memory data for test case 2 in Data De-multiplexing and Channel De-interleaver

4.10.7. Synthesis result

4.10.7.1. Area Report

```
Combinational area:       3920.764448
Noncombinational area:    1232.004892
Net Interconnect area:      73.000000

Total cell area:          5152.769340
Total area:               5225.769340
```

Figure 4.10-12 De-interleaver area report

4.10.7.2. Power Report

```
  Cell Internal Power  =  12.0314 uW    (5%)
  Net Switching Power  = 223.4794 uW   (95%)
                         ---------
Total Dynamic Power    = 235.5108 uW  (100%)

Cell Leakage Power     =   3.3741 uW
```

Figure 4.10-13 De-interleaver power report

4.10.7.3. Timing Report

```
clock INT_in_clk (rise edge)           100.00     100.00
clock network delay (ideal)              0.00     100.00
clock uncertainty                       -0.25      99.75
u1/counter2_reg[1]/CK (DFFRQX1M)         0.00      99.75 r
library setup time                      -0.33      99.42
data required time                                 99.42
-----------------------------------------------------------
data required time                                 99.42
data arrival time                                  -5.06
-----------------------------------------------------------
slack (MET)                                        94.35
```

Figure 4.10-14 De-interleaver time report

4.11. Rate De-matching for Turbo Decoder

4.11.1. Top level

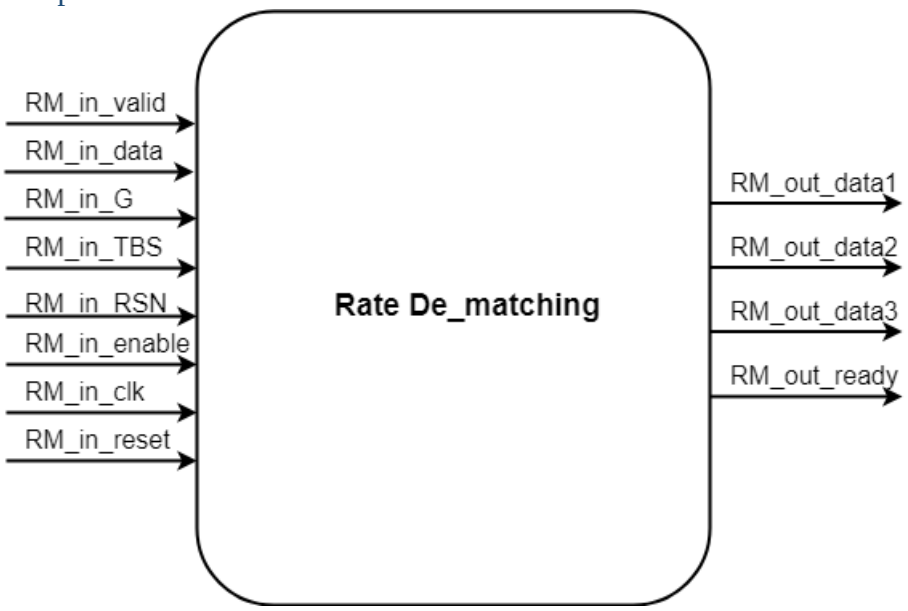


Figure 4.11-1 Rate De-matching block diagram

4.11.2. Block interface

Signal name	Direction	Description	Size
RM_in_data	input	Input data for rate de- matching	8
RM_in_G	input	The total number of actual transmitted data bits	13
RM_in_TB	input	Transport block size for each interleaver	12
RM_in_RSN	input	Retransmission number for each token at the DataIn port (rvidx)	1
RM_in_clk	input	Clock for rate de-matching	1
RM_in_reset	input	Reset for rate de-matching	1
RM_in_valid	input	valid data signal	1
RM_in_enable	input	Enable signal	1
RM_out_data1	output	First Output data after rate de-matching	8
RM_out_data2	output	second Output data after rate de- matching	8
RM_out_data3	output	Third Output data after rate de-matching	8
RM_out_ready	output	Ready flag	1

Table 4.11-1Rate De-matching interface signals

4.11.3. Architecture

4.11.3.1. Architecture according to standard description

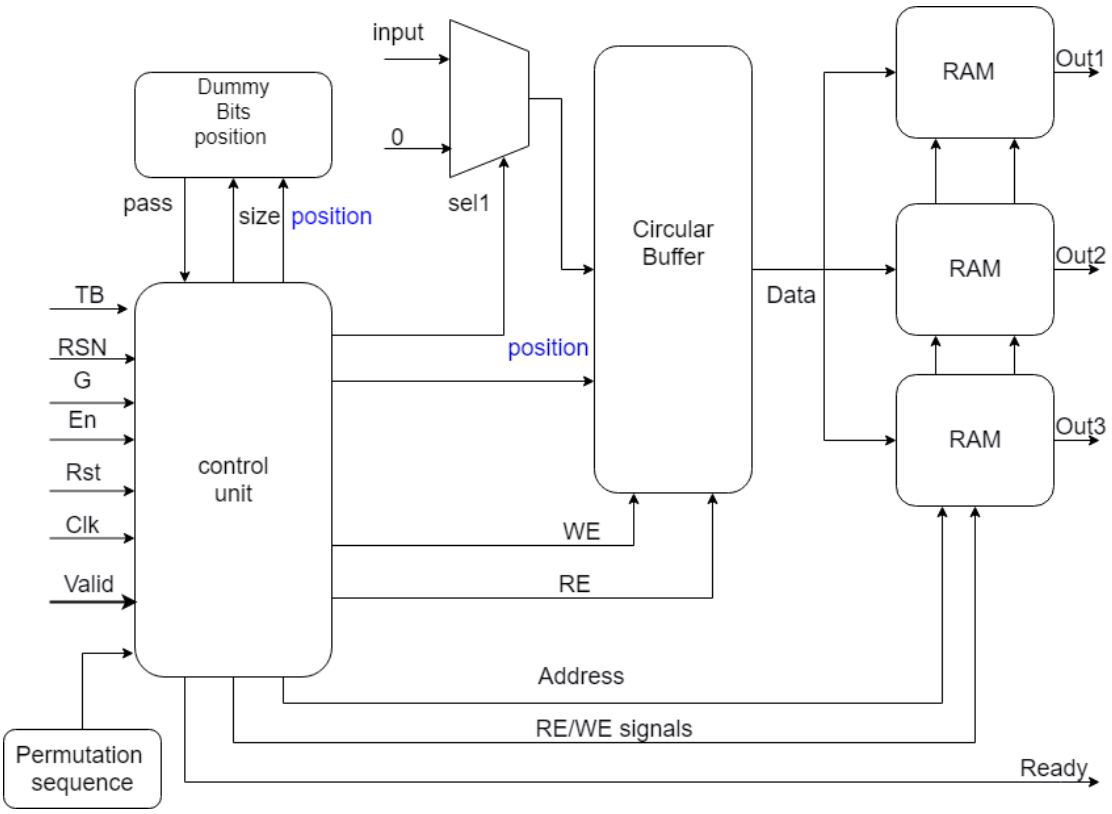


Figure 4.11-2 Rate De-matching architecture according to standard

This design consists of :

- Control unit handles the position calculation for the buffer and addresses calculation for the three RAMs so that the deinterleaving and demultiplixing process has been excuted ,also handles their control signals.
- Three RAMs to handle interleaving process each of size ((2560+4)*8) to support larger(TB)added to the tail bits and to pass soft data(8bits needed for each input)
- The circular buffer concatenate the three stream so the size of it will be (3*larger(TB)+12bit)*8bits =((3*2560)+12)*8= 61,536 bit.
- We need extra block which to memorize the dummy bits position to avoid write in the buffer in this position .

- Permutation sequence block to store in the sequence mentioned in the standard
- Input multiplexer to fill the empty postion by zeros and no external input enter here the perivous block should stop the sending process.

4.11.3.2. Our proposed Architecture

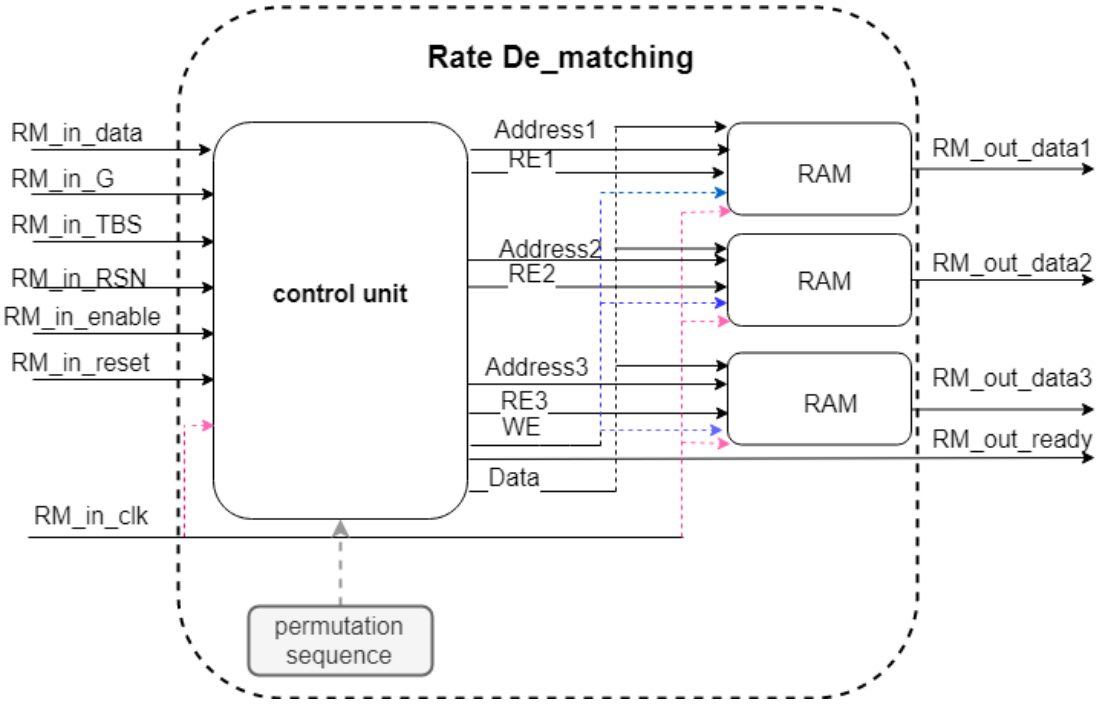


Figure 4.11-3 Our proposed architecture for Rate De-matching

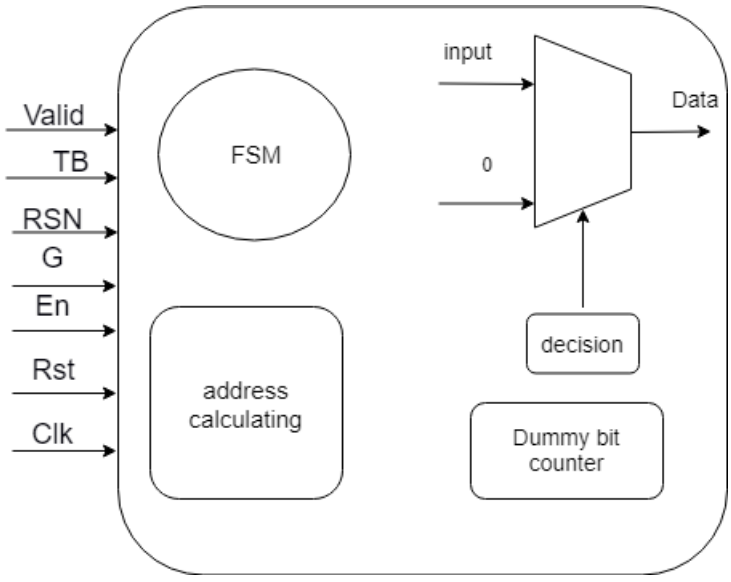


Figure 4.11-4 Architecture control unit for Rate De-matching

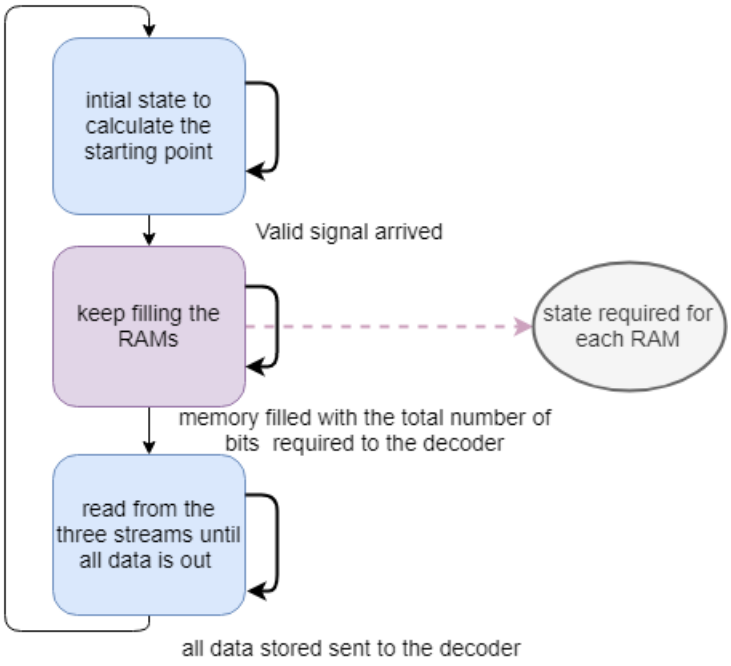


Figure 4.11-5 Flow chart of control unit for Rate De-matching

The Design consists of:

- Three RAMs one for each sub block interleaver of size 2564 *8 each to support maximum transport block size and passing soft data.
- Control unit to perform the bit selection, bit collection and de-interleaving operations.
- Permutation sequence Block to save the sequence mentioned in the standard

4.11.3.2.1. Control unit

Control unit consists of:

- Finite State Machine (FSM): is the brain of the block and it handles controlling signal for all the other units required in each state.
- Address calculating unit: create the address or manipulate the existing address by addition, subtraction and shifting to calculate the new address.
  Multi-operation could happen on the address at the same cycle.
  This unit is that is directly connected to the permutation sequence Block.
- Dummy bit counter to count the skipping cycle to be able to know when the zero filling of the data starts.
- Input multiplexer same usage as explained before.

4.11.3.2.2. Structural issues in our proposed architecture

1. In this design we removed the Circular buffer to reduce the huge area and power consumed by the buffer.

So control unit now handles only the addresses for the three RAMs and their control signals.
But that also introduce a difficulty of the starting point of the bit selection
To solve this problem, we trace the starting point equation

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

We get that it takes only two values as rv_{idx} takes also two values only
So when rv_{idx}=0, K_0=2$R_{subblock}^{TC}$, this means we start filling in the first RAM From third permutation column
And when rv_{idx}=2, K_0=50$R_{subblock}^{TC}$, this means we start filling in the second RAM From eighteenth permutation column because each sub block interleaver consist of 32column only.

2. We calculate the address so the deinterleaving and demultiplixing process has been excuted
   So now we know that the Dummy bits exist at the begin of the memory so we only need for it extra counter to get the correct input data size.
   We skip writing any thing in the memory at the Dummy bit cycles as we do not care about it also when we read the data.
   But we Still have to stop the reading process at their adddresses cycle and this cause a gap between this block and the block before so extra control is needed to disable the block before and this problem exist also in the previous design according to the standard
3. Also to minimize the combinational in the third equation we trace it and it gives same Permutation sequence saved +1 but in range of 5 bits only which means if 32 is reached make it 0,And then do the deinterleaving operation as the other two streams

4.11.3.3. Another proposed Architecture

Same pervious architecture but we write in the memory column by column and read it row by row to reduce the complexity of address calculations but this design need S/P converter at the input data path and P/S converter at the output data path.

4.11.4. Operation according to previous FSM

6. Get the valid signal and the upper layer parameters
7. Calculate the number of rows and the null size
8. get the starting point according to the upper layer input RSN
9. Start filling the RAMs according to the standard descripition of the filling sequence,While doing so keep count the null cycles
10. If you reached the intial 1/3 rate that the decoder require to start it's operation , start sending the three streams to the decoder
11. If all data out wait till you get new valid signal and upper layer parameters to start working again .

4.11.5. Challenges and enhancement

- All equations are executed without making any multiplication or divisions only used operations are addition, subtraction and shifting.
- Removing the circular buffer and the Dummy position Blocks introduce diffeculity to the control unit block but improve both power and area .

4.11.6. Testing Results

Results in decimal form

4.11.6.1. Test case1(No Zero filling)

- TB=40(TBS = 44)
- G = 132
- RSN = 0

4.11.6.1.1. RTL Simulation result

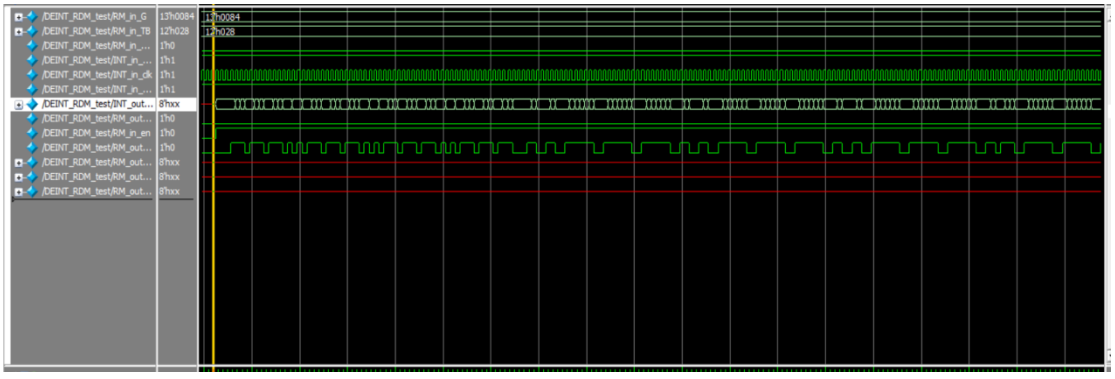


Figure 4.11-6 RTL wave form for test case 1 in Rate De-matching

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
20	25	15	86	76	2	134	73	63	38	28	109	99	71	121	50	40	14	4	85	75
40	1	133	62	52	37	27	98	88	72	122	61	51	26	16	97	87	13	3	74	64
60	49	39	110	100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.11-7 RTL memory1 data for test case 1 in Rate De-matching

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
20	41	21	31	12	5	127	137	118	77	57	67	48	111	91	101	81	29	9	19	142
40	135	115	125	106	65	45	55	36	123	103	113	94	53	33	43	24	17	139	7	130
60	89	69	79	60	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.11-8 RTL memory2 data for test case 1 in Rate De-matching

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
20	93	42	22	32	23	6	128	138	129	78	58	68	59	112	92	102	82	30	10	20
40	11	136	116	126	117	66	46	56	47	124	104	114	105	54	34	44	35	18	140	8
60	141	90	70	80	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.11-9 RTL memory3 data for test case 1 in Rate De-matching

4.11.6.1.2. MATLAB Simulation result

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	25	15	86	76	2	134	73	63	38	28	109	99	71	121	50	40	14	4	85	75
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	1	133	62	52	37	27	98	88	72	122	61	51	26	16	97	87	13	3	74	64
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	49	39	110	100																

Figure 4.11-10 MATLAB memory1 data for test case 1 in Rate De-matching

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	41	21	31	12	5	127	137	118	77	57	67	48	111	91	101	81	29	9	19	142
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	135	115	125	106	65	45	55	36	123	103	113	94	53	33	43	24	17	139	7	130
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	89	69	79	60																

Figure 4.11-11 MATLAB memory2 data for test case 1 in Rate De-matching

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	93	42	22	32	23	6	128	138	129	78	58	68	59	112	92	102	82	30	10	20
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	11	136	116	126	117	66	46	56	47	124	104	114	105	54	34	44	35	18	140	8
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	141	90	70	80																

Figure 4.11-12 MATLAB memory1 data for test case 1 in Rate De-matching

4.11.6.2. Test case2(Zero filling)

- TB=40(TBS = 44)
- G = 100
- RSN = 0

4.11.6.2.1. RTL Simulation result

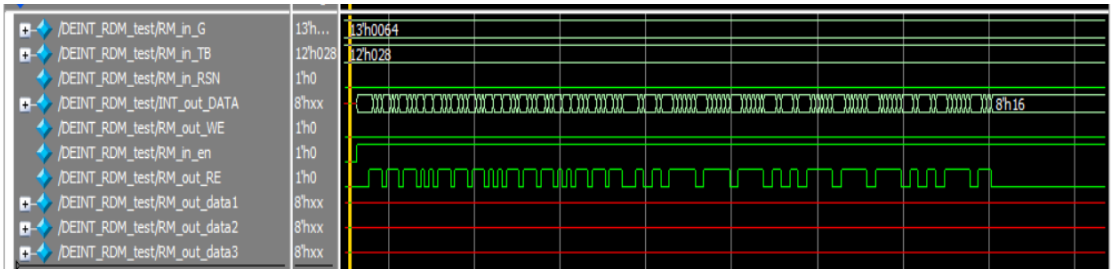


Figure 4.11-13 RTL wave form for test case 2 in Rate De-matching

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
20	25	15	86	76	2	134	73	63	38	28	109	99	0	121	50	40	14	4	85	75
40	1	133	62	52	37	27	98	88	0	122	61	51	26	16	97	87	13	3	74	64
60	49	39	110	100	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.11-14 RTL memory1 data for test case 2 in Rate De-matching

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
20	41	21	31	0	5	127	137	0	77	0	67	0	111	91	101	0	29	9	19	0
40	135	115	125	0	65	0	55	0	123	103	113	0	53	0	43	0	17	139	7	0
60	89	0	79	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.11-15 RTL memory2 data for test case 2 in Rate De-matching

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
20	0	42	22	32	0	6	128	138	0	78	0	68	0	112	92	102	0	30	10	20
40	0	136	116	126	0	66	0	56	0	124	104	114	0	54	0	44	0	18	140	8
60	0	90	0	80	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 4.11-16 RTL memory2 data for test case 3 in Rate De-matching

4.11.6.2.2. MATLAB Simulation result

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	25	15	86	76	2	134	73	63	38	28	109	99	0	121	50	40	14	4	85	75
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	1	133	62	52	37	27	98	88	0	122	61	51	26	16	97	87	13	3	74	64
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	49	39	110	100																

Figure 4.11-17 MATLAB memory1 data for test case 2 in Rate De-matching

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	41	21	31	0	5	127	137	0	77	0	67	0	111	91	101	0	29	9	19	0
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	135	115	125	0	65	0	55	0	123	103	113	0	53	0	43	0	17	139	7	0
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	89	0	79	0																

Figure 4.11-18 MATLAB memory2 data for test case 2 in Rate De-matching

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	42	22	32	0	6	128	138	0	78	0	68	0	112	92	102	0	30	10	20
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	0	136	116	126	0	66	0	56	0	124	104	114	0	54	0	44	0	18	140	8
	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
1	0	90	0	80																

Figure 4.11-19 MATLAB memory3 data for test case 2 in Rate De-matching

4.11.7. Synthesis result

```
Combinational area:       7123.741902
Noncombinational area:    2553.438999
Net Interconnect area:     141.000000

Total cell area:          9677.180901
Total area:               9818.180901
```

Figure 4.11-20 Rate de-matcher area report

```
  Cell Internal Power  =  23.6598 uW    (3%)
  Net Switching Power  = 802.5718 uW   (97%)
                         ---------
Total Dynamic Power    = 826.2316 uW  (100%)

Cell Leakage Power     =   6.3077 uW
```

Figure 4.11-21 Rate de-matcher power report

```
clock RM_in_clk (rise edge)              100.00     100.00
clock network delay (ideal)                0.00     100.00
clock uncertainty                         -0.25      99.75
u4/address1_reg[8]/CK (DFFRQX2M)           0.00      99.75 r
library setup time                        -0.32      99.43
data required time                                   99.43
-------------------------------------------------------------
data required time                                   99.43
data arrival time                                    -7.04
-------------------------------------------------------------
slack (MET)                                          92.40
```

Figure 4.11-22 Rate de-matcher timing report

4.11.8. Extra interconnection control unit

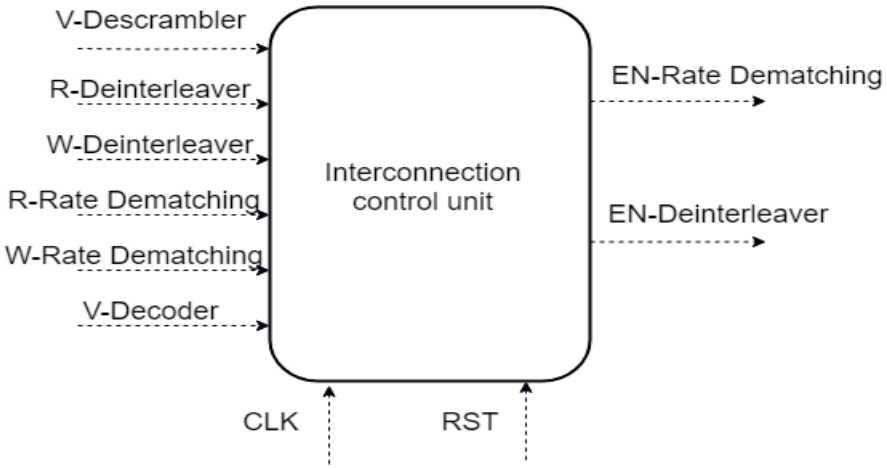


Figure 4.11-23 Extra interconnection control unit block diagram

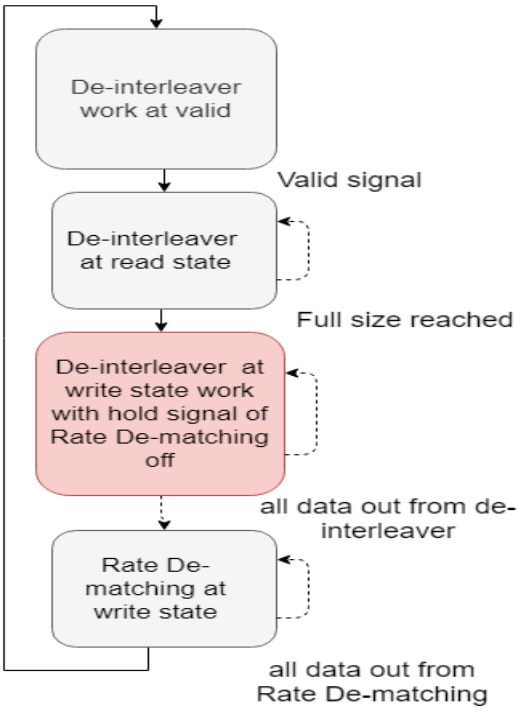


Figure 4.11-24 Flow chart of extra interconnection control unit between Rate De-matching and Data De-multiplexing and Channel De-interleaver

The control unit consist of FSM only which handle the enable signals for the two blocks attached to it.

This Finite state machine focuses on the writing state in the Data De-multiplexing and Channel De-interleaver which is the Reading state in Rate De-matching block.

4.12. Turbo Decoder

4.12.1. Top level

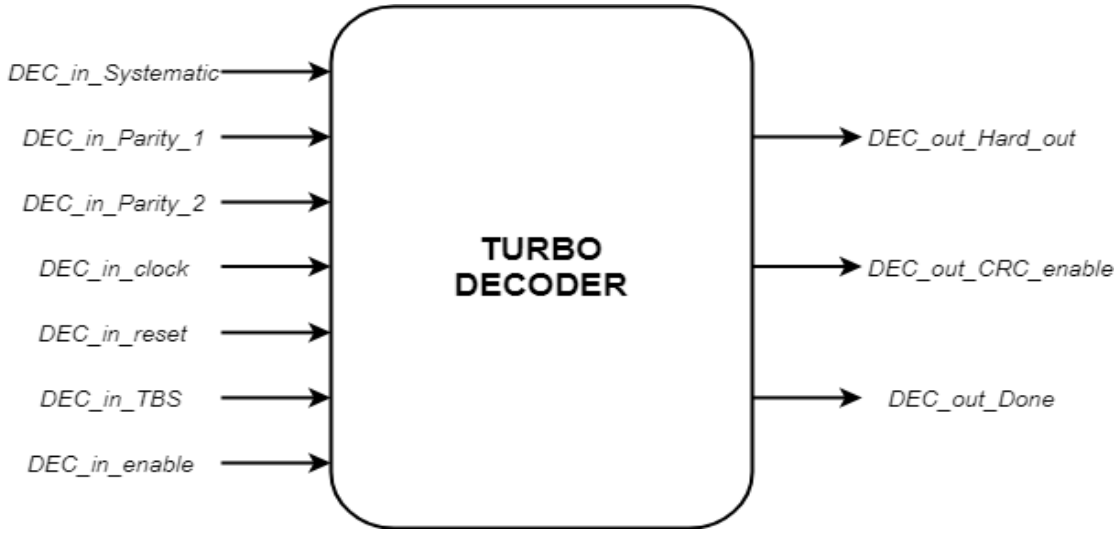


Figure 4.12-1 Turbo Decoder block diagram

4.12.2. Block interfaces

Signal name	Direction	Description	Size
DEC_in_Systematic	input	systematic vectors received from rate de-matcher	8
DEC_in_Parity_1	input	parity1 vectors received from rate de-matcher	8
DEC_in_Parity_2	input	parity2 vectors received from rate de-matcher	8
DEC_in_TBS	input	Block size for interleaver and De-interleaver	12
DEC_in_enable	input	handshake with rate de-matcher to start decoding	1
DEC_in_clock	input	System clock	1
DEC_in_reset	input	Asynchronous reset	1
DEC_out_Hard_out	output	Stream output bits	1
DEC_out_Done	output	Decoding is done	1
DEC_out_CRC_enable	output	handshake with CRC block	1

Table 4.12-1 Turbo decoder interface signals

4.12.3. Architecture

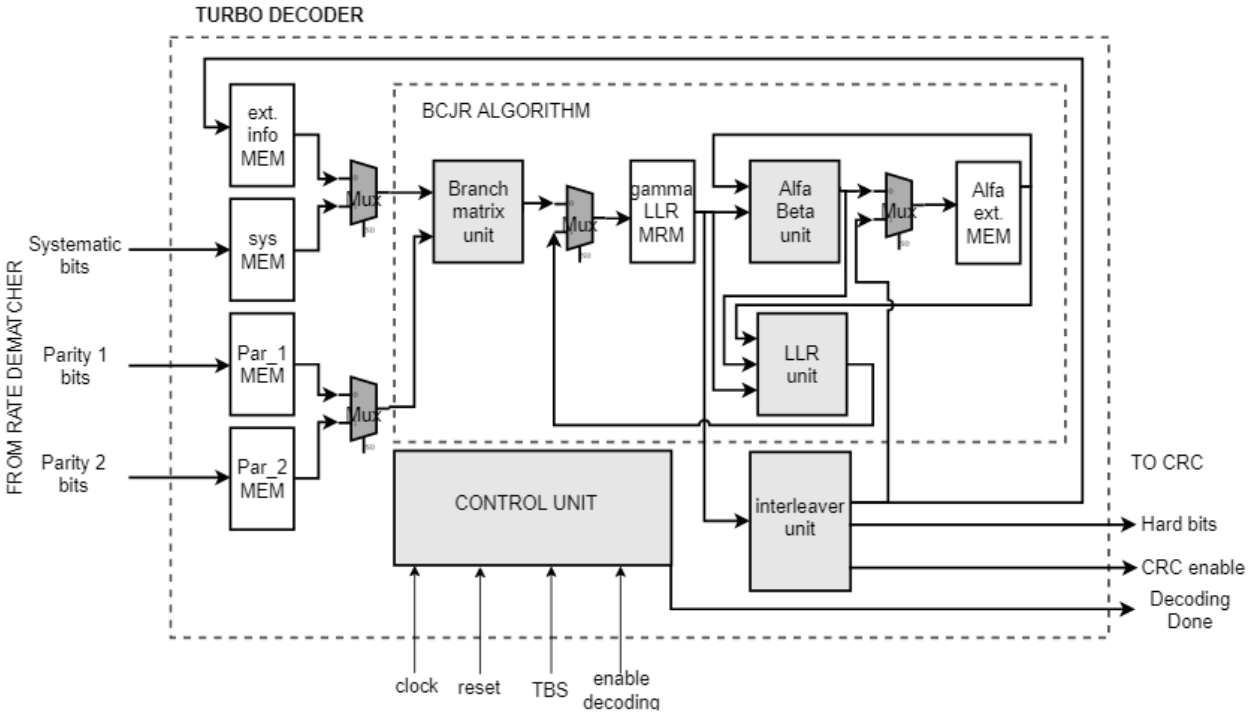


Figure 4.12-2 Turbo decoder architecture

4.12.4. Sub-Blocks & operation

4.12.4.1. Branch matric (γ)

Branch matric (γ) is the conditional probability that the received symbol is γ_k at time k and the current state is $S_k = S$, knowing that the state from which the connecting branch came was $S_{k-1} = S'$.

The trellis structure used by the RSC decoder is shown in Figure 4.12-3. Each state has two branches leaving it, one corresponding to an input one and one for input zero. Solid lines indicate data one and dotted lines indicate data zero. The branches indicate which next state can be reached from a particular state.

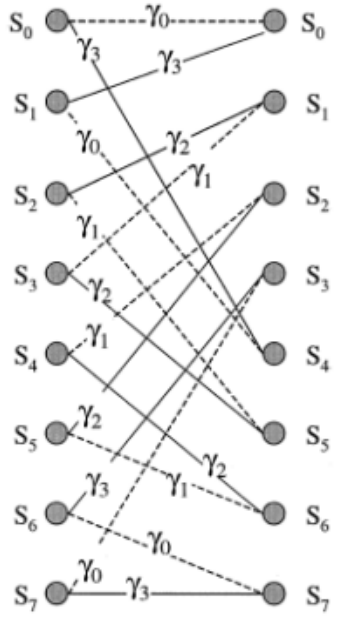


Figure 4.12-3 8 state trails diagram

The branch metric connecting state S_i (previous state, on left) and state S_j (present state, on right) is denoted as γ_{ij} . The branch metric depends on the data bit $X(i,j)$ as well as the parity bit $Z(i,j)$ associated with the branch. The branch metric is given as:

$$\gamma_{ij} = sys * X(i,j) + par * Z(i,j) + X(i,j) * EXT \quad (1)$$

Where sys and par are received soft value from channel and EXT is the extrinsic information from previous decoding stage initially is zero.

The RSC encoder being rate r=1/2, only four distinct branch metrics are possible:

$$\gamma_0 = sys + par + EXT \qquad X(i,j) = 1, Z(i,j) = 1$$

$$\gamma_1 = sys - par + EXT \qquad X(i,j) = 1, Z(i,j) = 0$$

$$\gamma_2 = -sys + par - EXT \qquad X(i,j) = 0, Z(i,j) = 1$$

$$\gamma_3 = -sys - par - EXT \qquad X(i,j) = 0, Z(i,j) = 0$$

Figure 4.12-4 shows diagram of branch matric unit of decoder 1 and 2 stages, input is multiplexed for each decoder stage.

After the calculation of the branch metrics, they should be stored in RAM modules to be used later in calculation of LLRs and backward State Metric Block β

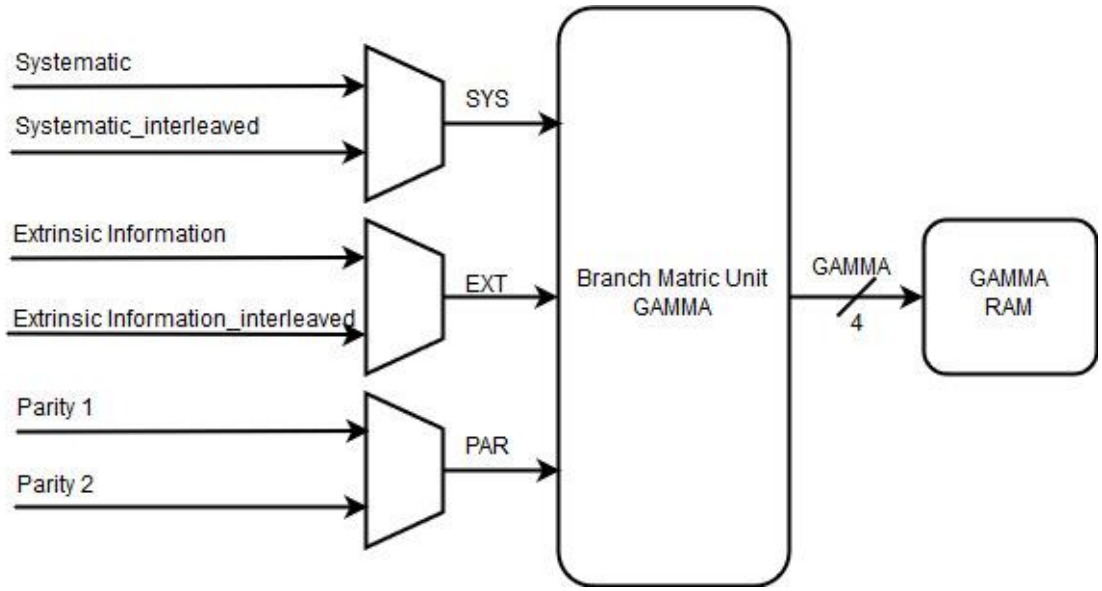


Figure 4.12-4 Branch Matric Unit block diagram

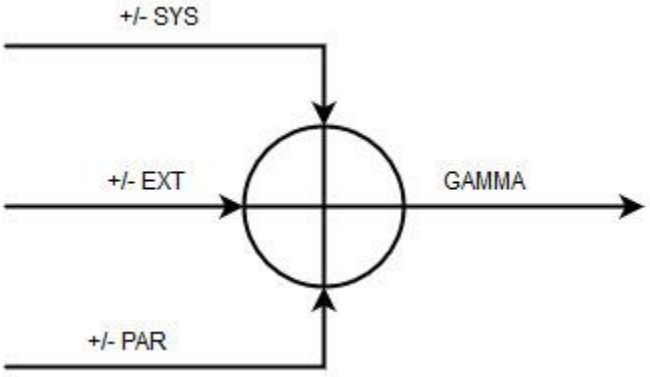


Figure 4.12-5 GAMMA equation

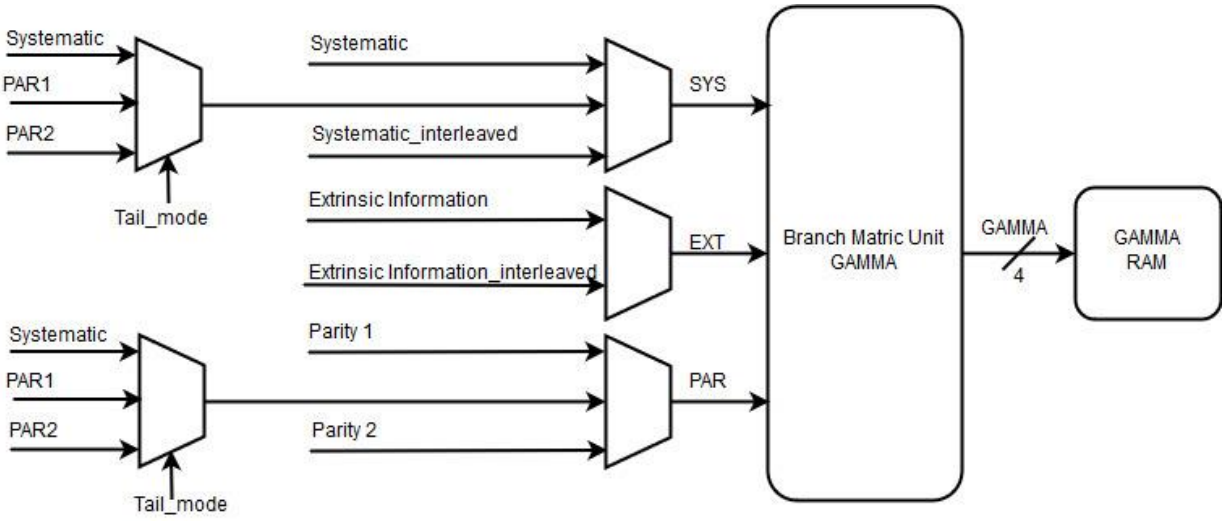


Figure 4.12-6 Branch Metric Unit modification foe tails

Trellis termination

Tail bits are included at the end of each block to force trellis diagram to reach zero state, Tails are used to ensure the initial value for backward state metric β_0 to be the highest probability which is one and states from β_1 to β_7 have zero probability.

Without using tails, we can assume equal probability for backward state metric initial value.

According to the standard tail bits are transmitted in different order unlike original data, for code-word with length K and original data from 0 to k-1, received systematic data= K+4, received parity1 data= K+4 and received parity2 data= K+4.

Each 4 extra data include systematic or parity1 or parity2 or systematic interleaved tails therefore systematic, parity1, parity2 and systematic interleaved must be reorders before decoding.

Systematic received tail bits' location

$$Systematic\ tail1 = systematic_K\ ,$$

$$Systematic\ tail2 = Parity2\ _K,$$

$$Systematic\ tail3 = Parity1\ _{K+1}$$

Parity 1 received tail bits' location

$$Parity1\ tail1 = Parity1_K\ ,$$

$$Parity1\ tail2 = Systematic\ _K,$$

$$Parity1\ tail3 = Parity2_{K+1}$$

Parity 2 received tail bits' location

$$Parity2\ tail1 = Parity1_{K+2}\ ,$$

$$Parity2\ tail2 = Systematic\ _{K+3},$$

$$Parity2\ tail3 = Parity\ 2_{K+3}$$

Systematic interleaved received tail bits' location

$$Systematic\_interleaved\ tail1 = systematic_{K+2},$$

$$Systematic\ tail2 = Parity2\ _{K+2},$$

$$Systematic\ tail3 = Parity1\ _{K+3}$$

Figure 4.12-6 shows added Multiplexer at systematic and parity input to Branch metric unit for tails bits and its control signals are controlled using control unit.

4.12.4.2. Forward and Backward State Metric ($\boldsymbol{\alpha}, \boldsymbol{\beta}$)

Forward α estimation of state probabilities indicates probability of each state in case of moving in the forward direction in the trellis diagram, While Backward state probability of a certain state at a certain time indicates probability of transition to this state given a certain received code-word after this time. The calculation of the backward state probabilities is similar to that of forward state probabilities.

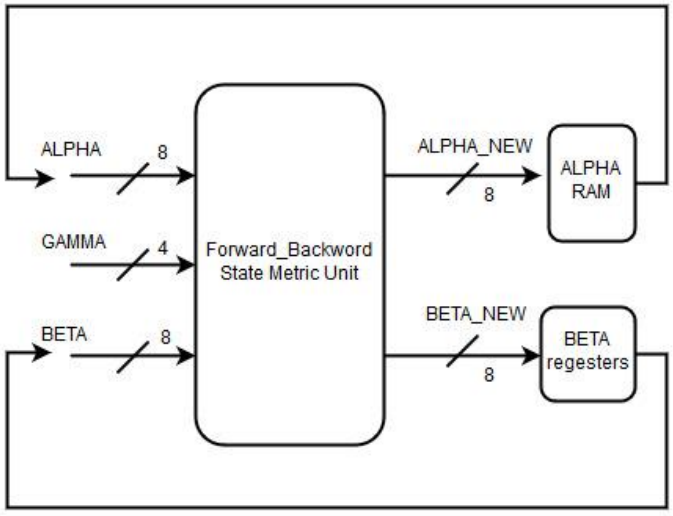


Figure 4.12-7 Forward and Backward Metric unit diagram

$$\alpha_K(S_K) = \max^*(\alpha_{K-1}(S_{K-1}) + \gamma(S_{K-1}, S_K))$$

$$\beta_K(S_K) = \max^*(\beta_{K+1}(S_{K+1}) + \gamma(S_{K+1}, S_K))$$

According to previous equations the computation is the same but the state transitions are different.
Therefore, MUXs are used to multiplex between α and β and between γ_K and γ_{K+1}.
Those equations are implemented on 8 states.
The previous figure shows forward and backward calculation unit for one state.
Due to RSC encoder starts from zero state therefor initial value for Forward states $\alpha_0 = 1, \alpha_1$ to $\alpha_7 = 0$ at time 0, Same for Backward states due to trails termination encoding ends ant zero state $\beta_0 = 1, \beta_1$ to $\beta_7 = 0$ at time $K + 1$.
Probability of 1 in log scale is represented by zero and Probability of 0 in log scale is represented by -4.
Normalization is done by comparing all 8 states values and subtracting each value by the maximum value, the key idea is that the main concern is not in the value of the state metric itself, but in the value of the difference between the state metrics.

The main drawback in implementing state metrics is the recursive computation. This may lead to an arithmetic overflow. To avoid overflow, a large number of bits is needed for representation of state metrics. This means more area, hardware resources, higher storage requirements, and increased delay therefore limitation on certain range [-4:4] is done.

Also the drawback of state metric normalization is the increase in the critical path of the state metric unit. It is considered the bottleneck of the SISO decoder that limits the maximum frequency of operation. The critical path implies Addition, comparison, MUX, and normalization which includes both comparison and subtraction.

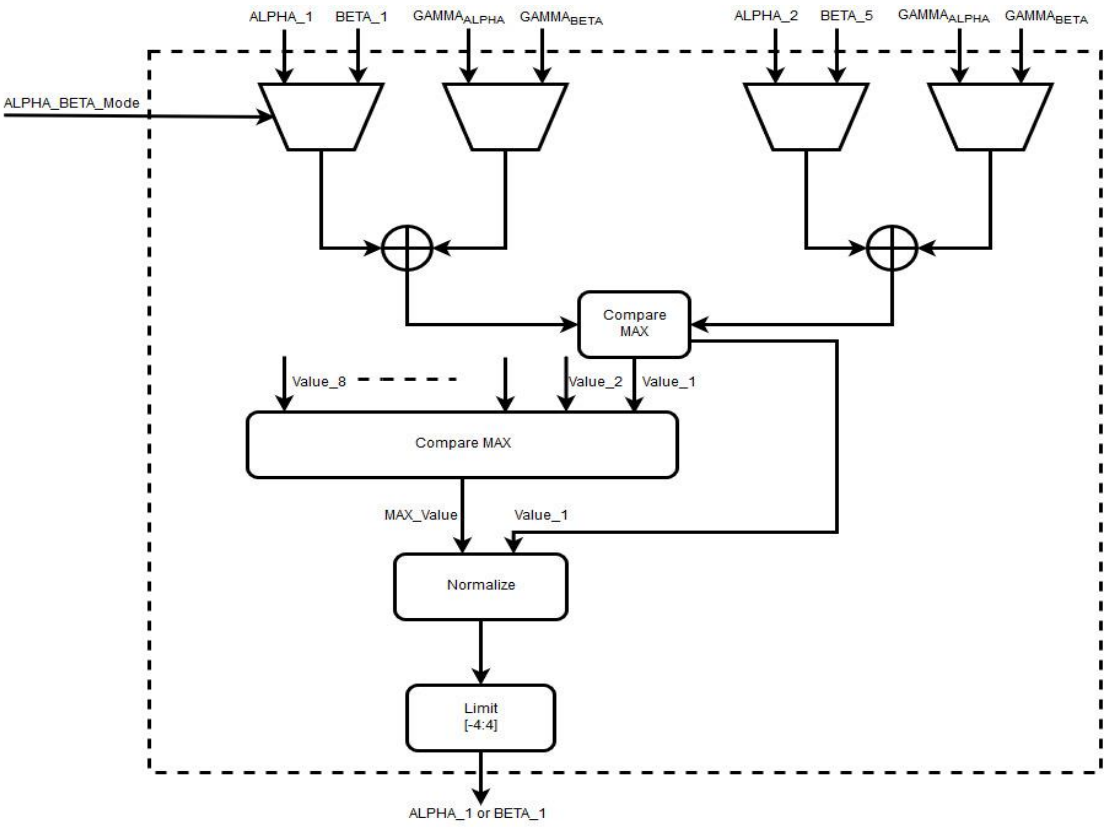


Figure 4.12-8 State 1 metric unit

4.12.4.3. Interleaver

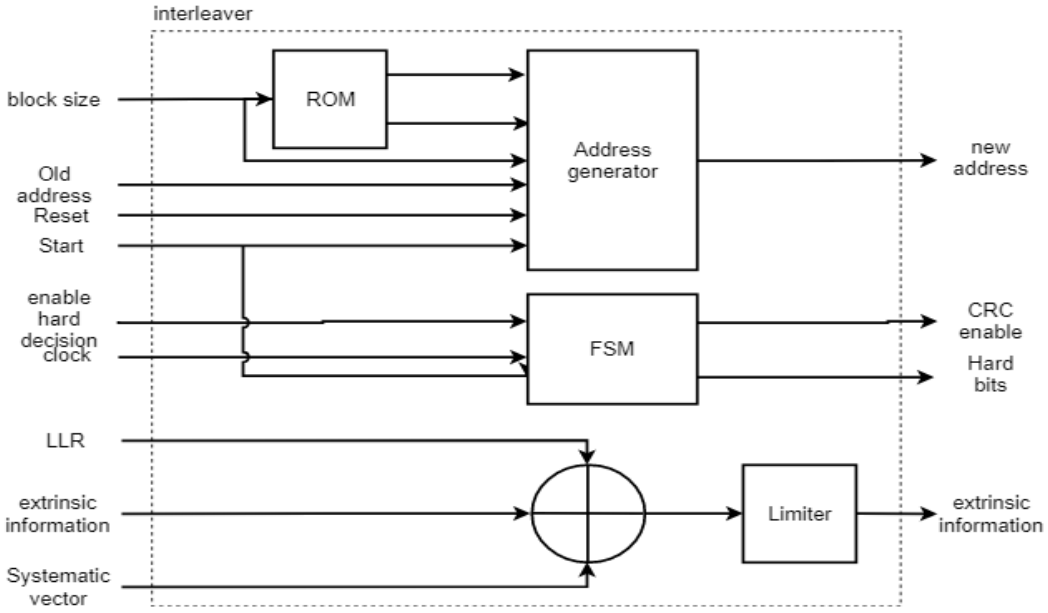


Figure 4.12-9 Interleaver unit architecture

The interleaver block consists of 4 sub-blocks:

1. Address generator.
2. Extrinsic information calculation.
3. Hard limiter and CRC enable.
4. ROM

1. Address generator:

As shown in figure 4.12-9 the address generator sub-block is a QPP interleaver that performs the equation specified in the standard which is:

$$\pi(i) = (f1 * i + f2 * i^2)\% \, k$$

The equation is done using 3 multiplications and mod operation. To reduce the number of multiplication and replace it with shift and addition operations a recursive way of calculation is used according to the following equation:

$$\text{Address}[i] = \{2*\text{Address}\,[i-1] - \text{Address}\,[i-2] + 2*F\_2\} \bmod \{\text{block\_size}\}$$

Where:

- F_2 is a constant that depends on the block size.
- Block_size is the data length and it is received from upper layer.

To generate the new address, the mod operation must be used. And it is not synthesizable, so it is replaced by the following equation that contains subtraction, multiplication and division:

$$A \bmod B = A - A/B$$

Where A/B is an integer division implemented using Restoring Division algorithm.

2. Extrinsic information calculation:

It is a combinational circuit of subtractor and limiter, to calculate the extrinsic information needed for the second decoding stage and limit it by {4, -4}.

3. Hard limiter and CRC enable:

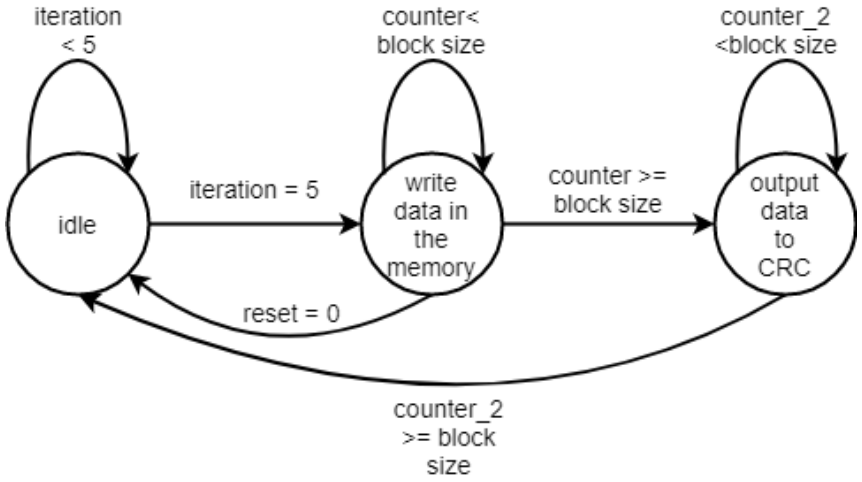


Figure 4.12-10 Hard limiter finite state machine

The hard limiter is a finite state machine of 3 states:

1. State 1: idle state as the decoder is not enabled yet or it is not performing the last iteration.
2. State 2: write data in the memory state, in which the hard limiter takes the MSB of the sequence, invert it and write it in a column memory sequentially.
3. State 3: output data to CRC sequentially at each clock cycle along with the CRC enable signal.

4.12.4.4. LLR

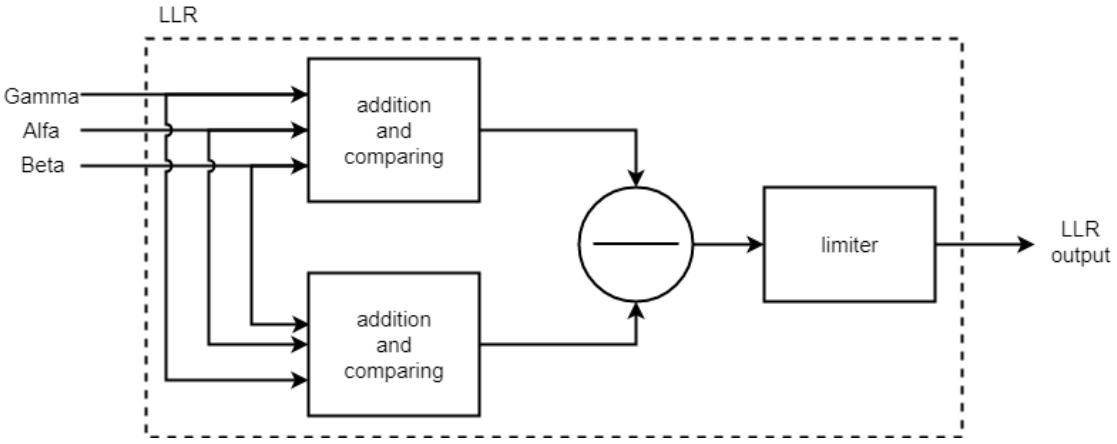


Figure 4.12-11 LLR unit architecture

As shown in the figure 4.12-11, The LLR unit is a combinational unit that performs addition and comparison for the values of the branch matric and Forward & backward recursions to produce the LLR (log likelihood ratio) and limits the output to {4, -4} which represents infinity for the decoder.

4.12.4.5. Control unit

Control unit is used to control decoding flow between sub-blocks Input memories (Systematic, parity1, parity2)

- Branch Metric unit multiplexer's selections (tails bits, decoder1 or decoder2 mode)
- Gamma γ memory
- State metric unit multiplexer's selections (α or β mode)
- Alpha α memory
- Inter-leaver and de-interleaver enable
- Number of decoding iterations

Figure 4.12-13 shows BCJR decoding flowchart and FSM, starting from storing data from rate de-matcher then controlling MUXs for GAMMA, ALPHA and BETA calculation and storing in RAMs after processing on total block calculating LRR and extrinsic information start and finally data is ready to interleave for second decoder stage.
After second decoding stage LLR and extrinsic information de-interleaved to start new iteration.

Figure 4.12-14 shows Turbo decoder control flow starting from first decoding stage to calculate extrinsic information which is interleaved and passed to second decoder stage to calculate extrinsic information which is passed to first decoder stage and LLR which is interleaved and used to calculate final hard decision value this is done after N times decoding iterations.

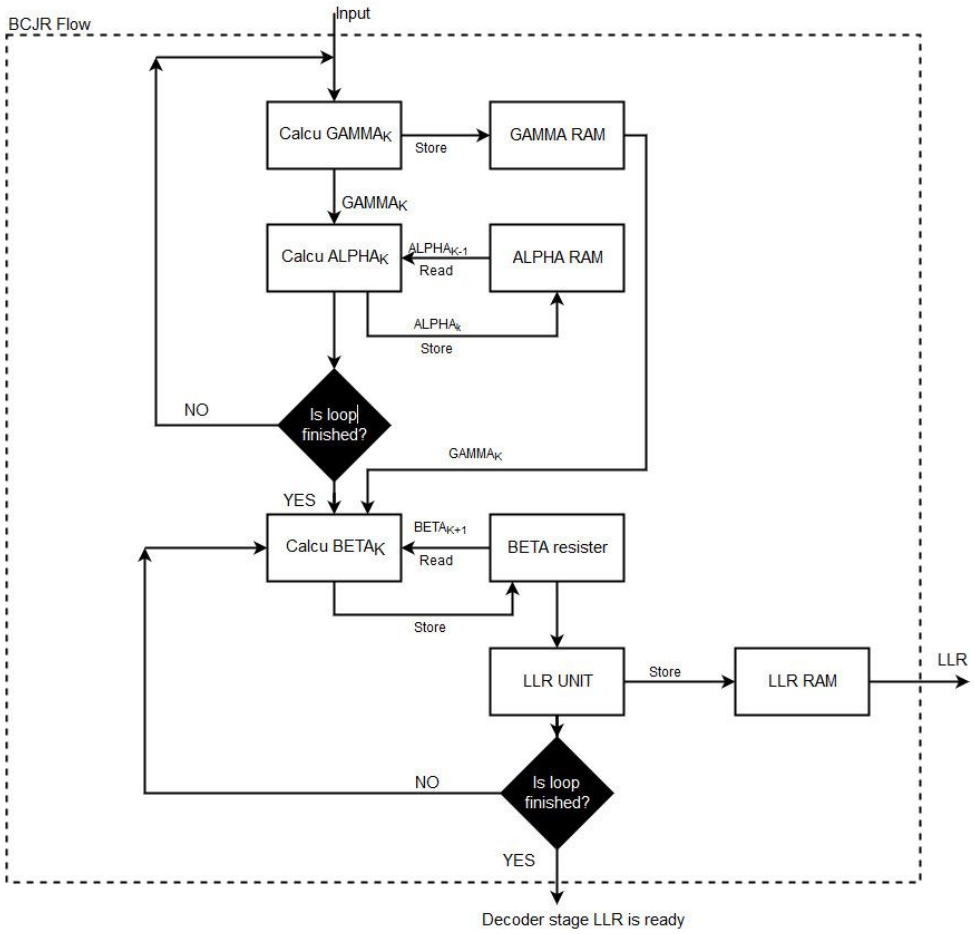


Figure 4.12-12 RSC decoder stage flow

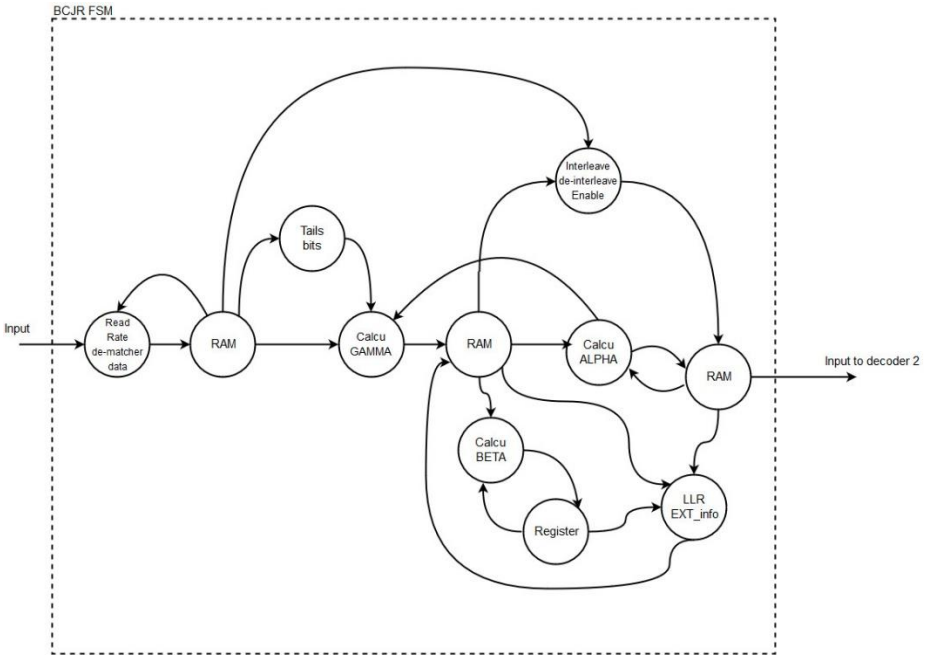


Figure 4.12-13 BCJR FSM

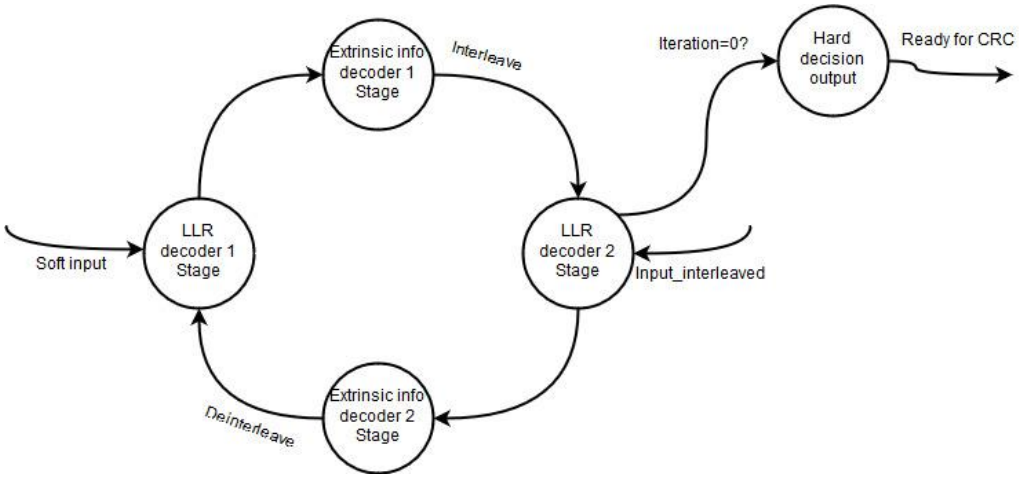


Figure 4.12-14Turbo decoder control flow

4.12.5. Results

4.12.5.1. Matlab Results

The turbo code was simulated for frame size K = 2560 over a AWGN channel.

Figure 4.12-15 shows BER for un-coded bits and BER for turbo encoded bits, the number of decoder iterations was chosen to be 5.
For low SNR BER for turbo is worse than un-coded bits, as SNR increases BER for turbo has great improvement.

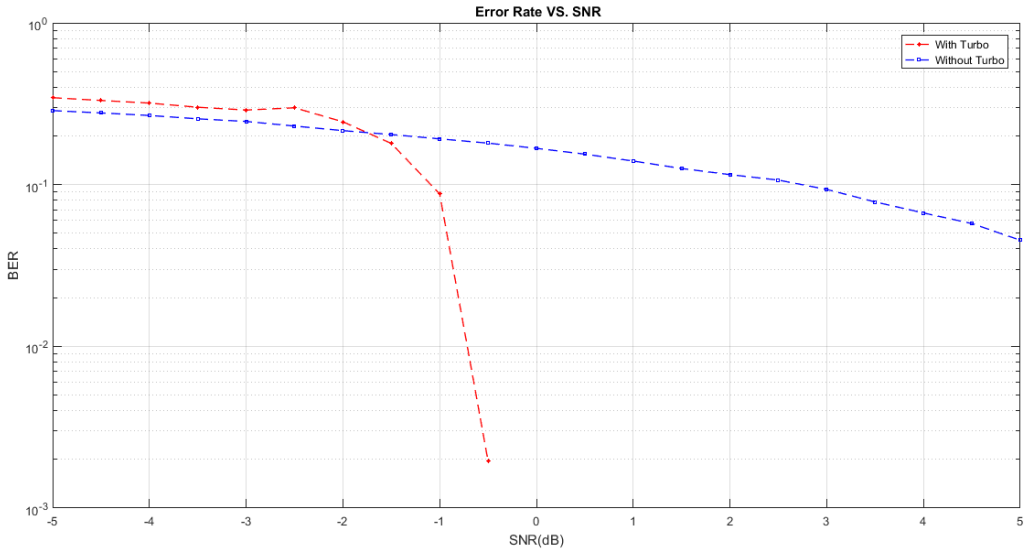


Figure 4.12-15 BER using turbo decoder Vs without decoding

The SNR range was used from -5 to 1 dB, 4 integer bits and 3 fraction bits. The number of decoder iterations was chosen to be 5.

Figure 4.12-16 shows BER for decoding iterations from 1 to 7, for small SNR as number of iteration increases it has small effect on BER, for SNR greater than -2 dB as number of iteration increases BER decreases.

It can be seen that as the number of iteration increases, the BER performance improves. However, the rate of improvement decreases.

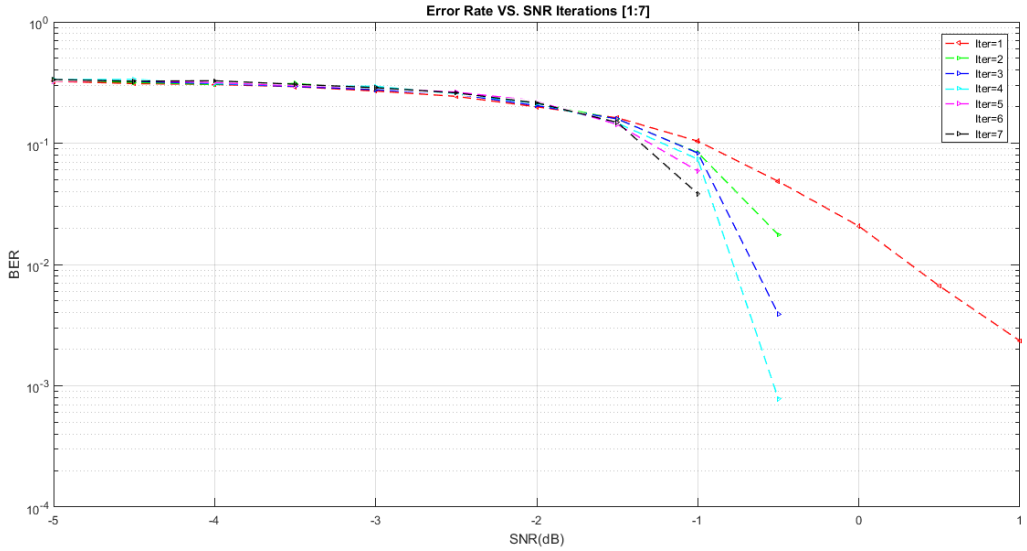


Figure 4.12-16 BER Vs SNR for different decoding iterations

The SNR range was used from -5 to -0.5 dB, 8-bit word length with 1 bit for sign, N integer bits and (7-N) fraction bits. The number of decoder iterations was chosen to be 5.

Figure 4.12-17 and 4.12-18 shows quantization error due to integer and fraction bits, Quantization error is large for small integer bits and high fraction bits and also for high integer bits and small fraction bits as.

For 8-bit word length with 4 integer bits and 3 fraction bits' quantization error has smallest quantization error.

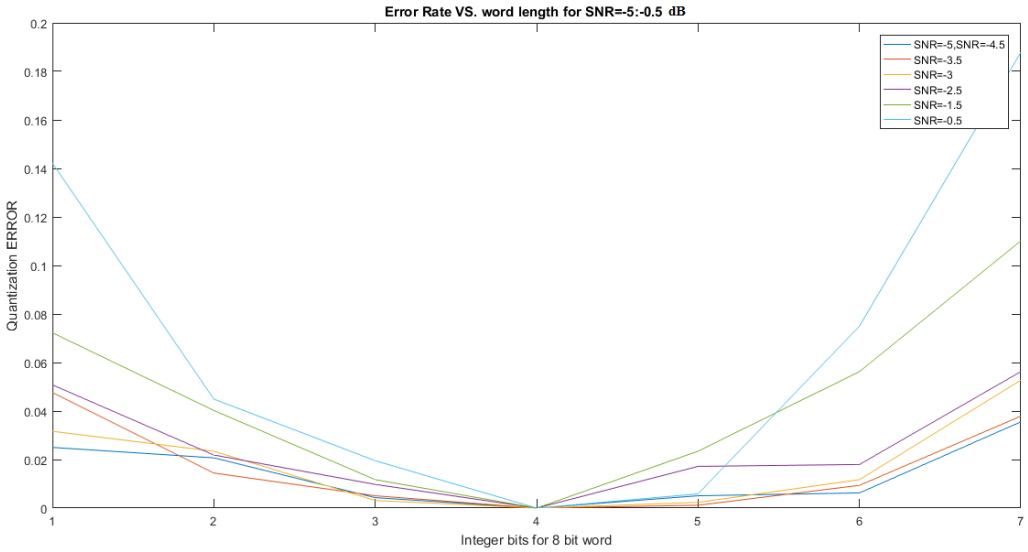


Figure 4.12-17 Effect of integer and fraction bits on decoding error at different values SNR

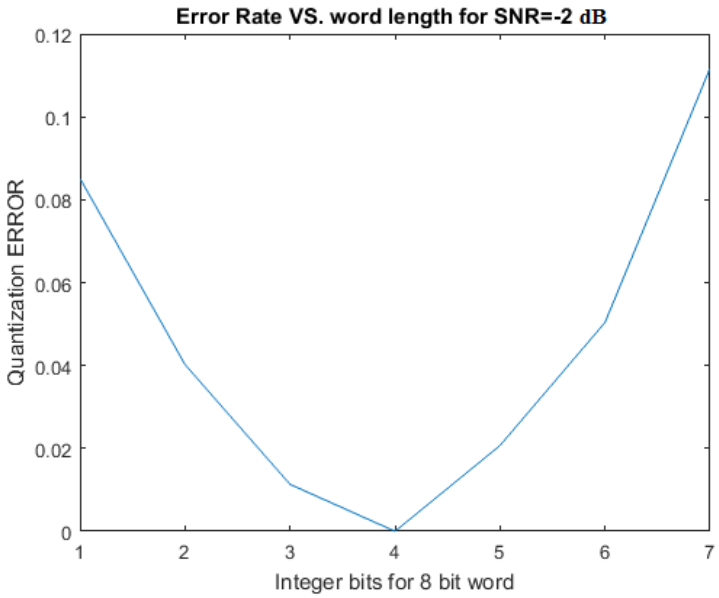


Figure 4.12-18 Effect of integer and fraction bits on decoding error at SNR=-2 dB

The SNR range was used from -5 to 1 dB, 4 integer bits and fraction bits from 1 to 5 bits. The number of decoder iterations was chosen to be 5.
Figure 4.12-19 shows the effect of increasing number of fraction bits on decoding performance, as number of fraction bits increases BER improves

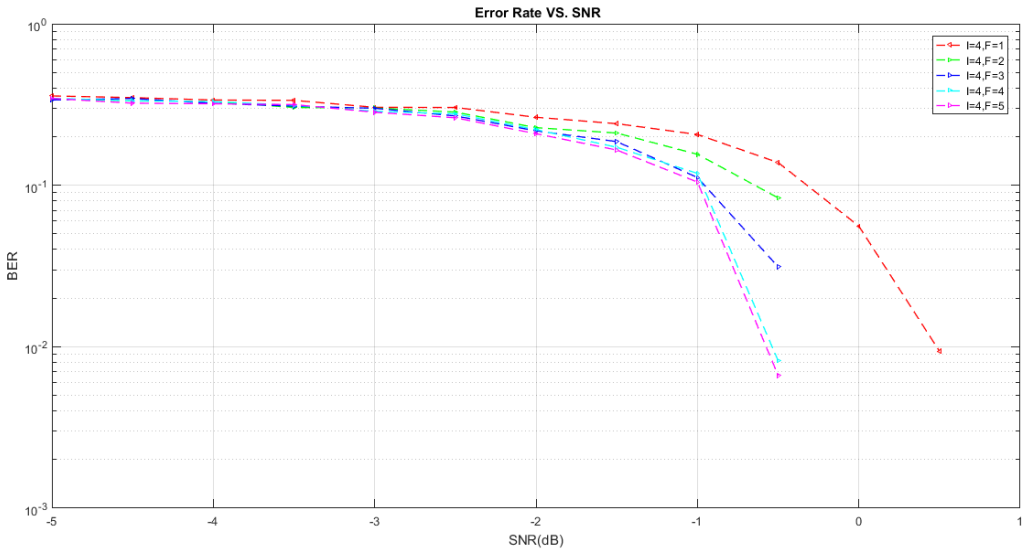


Figure 4.12-19 BER Vs SNR for 4 integer bits and different fraction bits

4.12.5.2. RTL Results

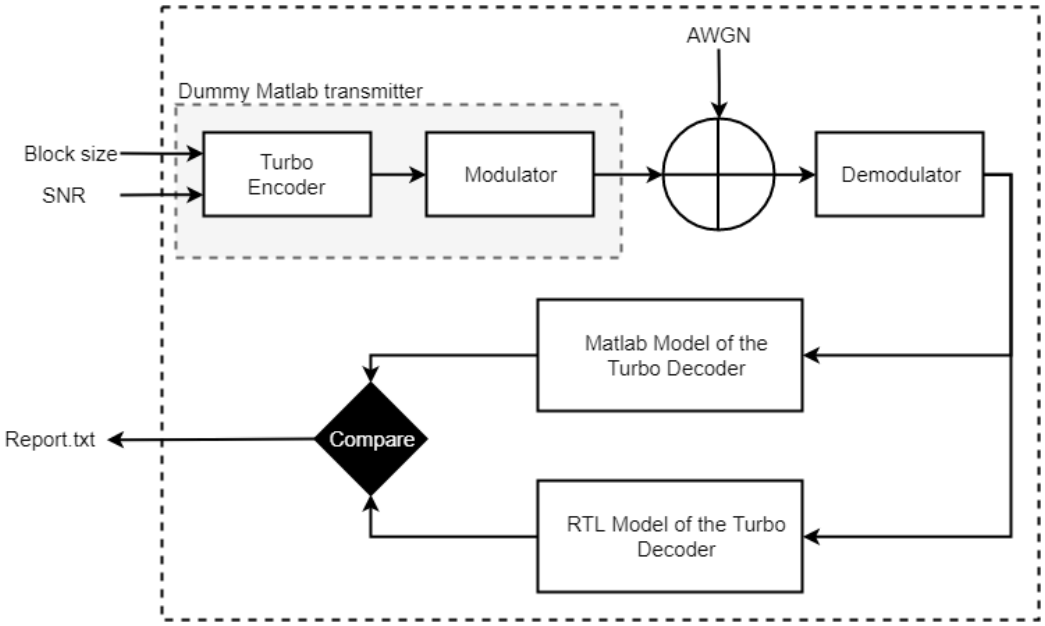


Figure 4.12-20 Testing Technique block diagram

As shown in figure 4.12-30 the block is tested and verified by creating a Matlab dummy transmitter that consists of a turbo encoder and BPSK/QPSK modulator. The output data of this transmitter is added to the channel noise then demodulated to be passed to two paths one is the Matlab turbo decoder model and the other is the RTL turbo decoder model, coded in Verilog, then the results is compared between the Matlab and the RTL model and the BER is calculated.

The output of the testing plan is a Report.txt file that contains:

1. The Herd bits output from the decoder.
2. Number of errors compared to the transmitted bits.
3. Bit error rate.
4. Number of errors compared to Matlab model of the turbo decoder.

A sample of the output file is shown in the next figure, tested for data block size of 128 and SNR of -2dB:

```
1
0
0
1
0
Errors =          18
BER =         14 percent
Errors with matlab =          0
```

Figure 4.12-21 Sample of the report file

Comparing Simulation results with RTL results for a block size of 40 and SNR of -1dB:

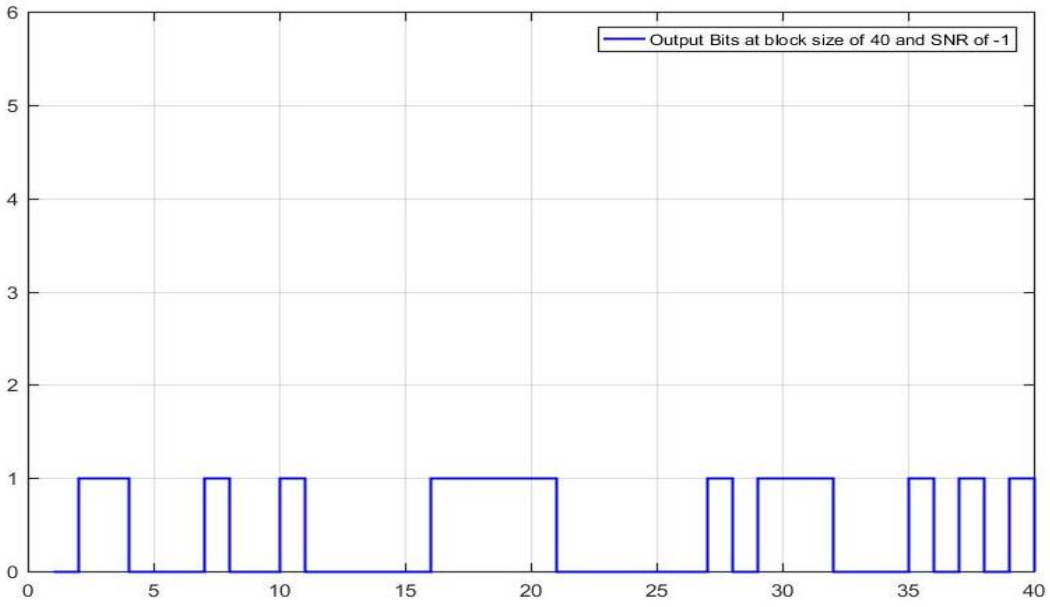


Figure 4.12-22 Decoder output (Matlab)

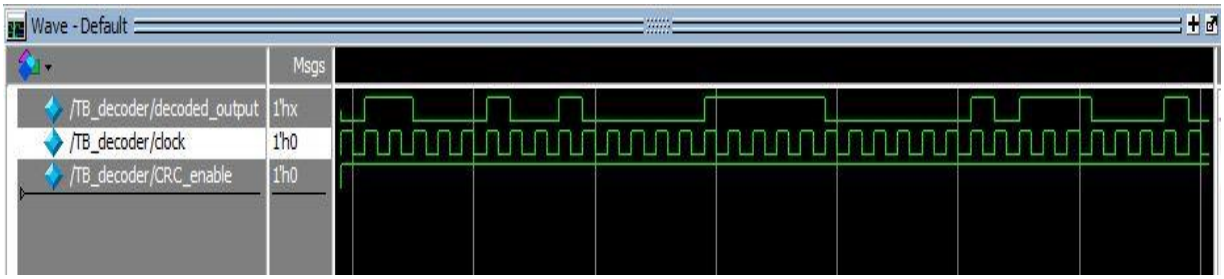


Figure 4.12-23 Decoder output (RTL)

4.12.6. Synthesis Results

```
Number of ports:                42
Number of nets:                496
Number of cells:                11
Number of references:           10

Combinational area:       49423.359728
Noncombinational area:    2960.639996
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:          52383.999723
Total area:                 undefined
```

Figure 4.12-24 Turbo decoder area report

Cell	Cell Internal Power	Driven Net Switching Power	Tot Dynamic Power (% Cell/Tot)	Cell Leakage Power	Attrs
interleaver	0.0461	N/A	N/A (N/A)	9615116.0000	h
LLR_value	0.0251	N/A	N/A (N/A)	3966410.0000	h
CU	6.812e-03	N/A	N/A (N/A)	1636939.3750	h
BM	1.966e-03	N/A	N/A (N/A)	743354.0000	
Totals (11 cells)	79.961uW	N/A	N/A (N/A)	15.962uW	

Figure 4.12-25 Turbo decoder power report

```
clock clk (rise edge)                                  100.00     100.00
clock network delay (ideal)                              0.00     100.00
clock uncertainty                                       -0.05      99.95
interleaver/pre_index_1_reg[11]/CK (DFERBCHD)            0.00      99.95
library setup time                                      -0.13      99.82
data required time                                                 99.82
-------------------------------------------------------------------------
data required time                                                 99.82
data arrival time                                                 -29.05
-------------------------------------------------------------------------
slack (MET)                                                        70.77
```

Figure 4.12-26 Turbo decoder timing report

4.13. Cyclic Redundancy Check (CRC)

4.13.1. Top level

This division is commonly implemented using LFSR circuits. Implementing Internal or Galois LFSR, takes the highest order bit (MSB) as the feedback term that is feedback into the relevant flip-flops through the XOR gates placed between the flip-flops. This form is the more popular because it is faster.

When input data is ready and CRC enable is high CRC starts its function when CRC enable falls from high to low this indicates that all code word enters CRC block, CRC_out_error is one if error exists otherwise is zero and CRC_out_valid indicates that CRC_out_error signal is valid to be read or not this happened when decoder and CRC are done.

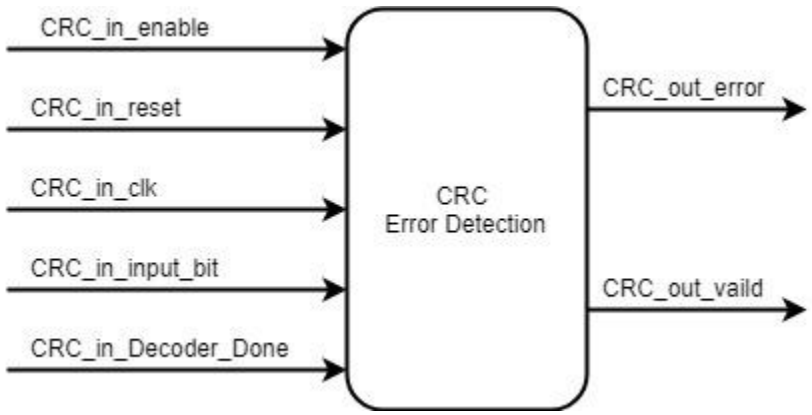


Figure 4.13-1 CRC block diagram

4.13.2. Block Interface

Signal name	Direction	Description	Size
CRC_in_enable	Input	Indicates that input data is valid	1
CRC_in_reset	Input	Asynchronous reset, Resets LFSR and output	1
CRC_in_clk	Input	System Clock	1
CRC_in_input_bit	Input	Input stream	1
CRC_in_Decoder_Done	Input	Indicates that decoder had finished, used for CRC_vaild output	1
CRC_out_error	Output	Indicates error exists or not	1
CRC_out_valid	Output	Indicates that CRC_out_error signal valid to be read or not	1

Table 4.13-1 CRC interface signals

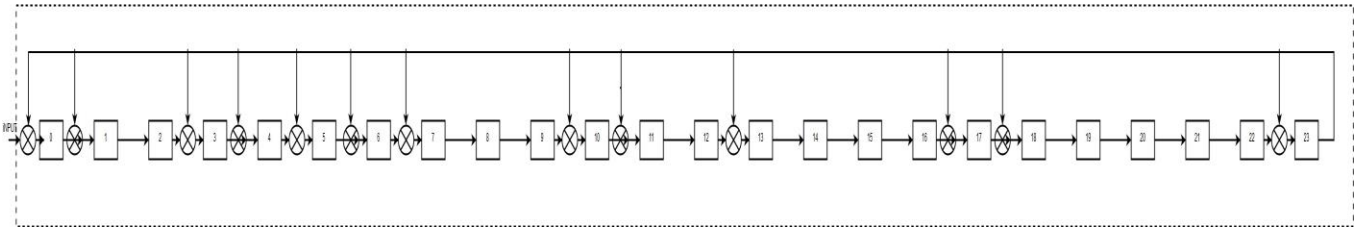


Figure 4.13-2 CRC shift register

4.13.3. Simulation Results

Figure 4.13-3 and 4.13-4 Simulation on data stream output from decoder with and without error and CRC_out_error is taken into consideration only when CRC_valid is high.

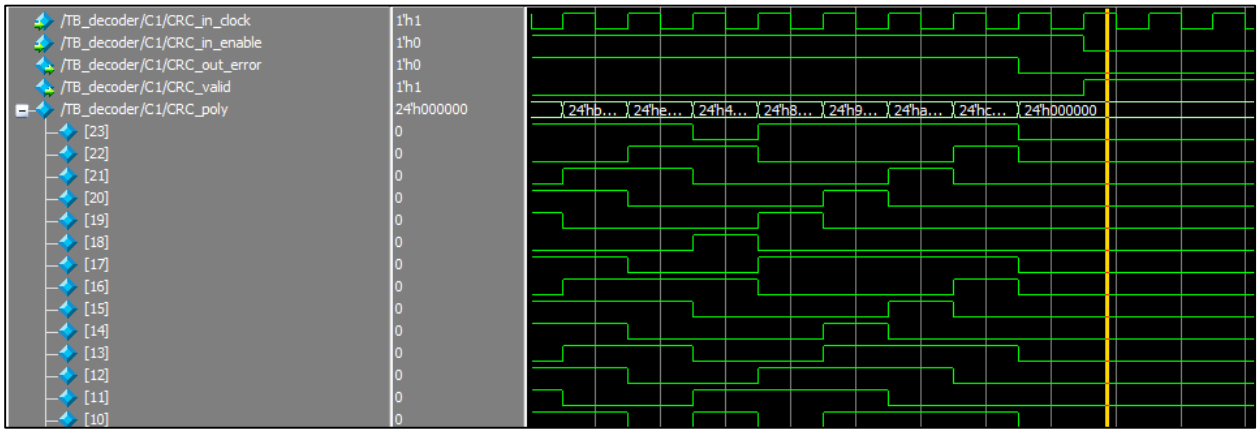


Figure 4.13-3 CRC output with zero error detection output

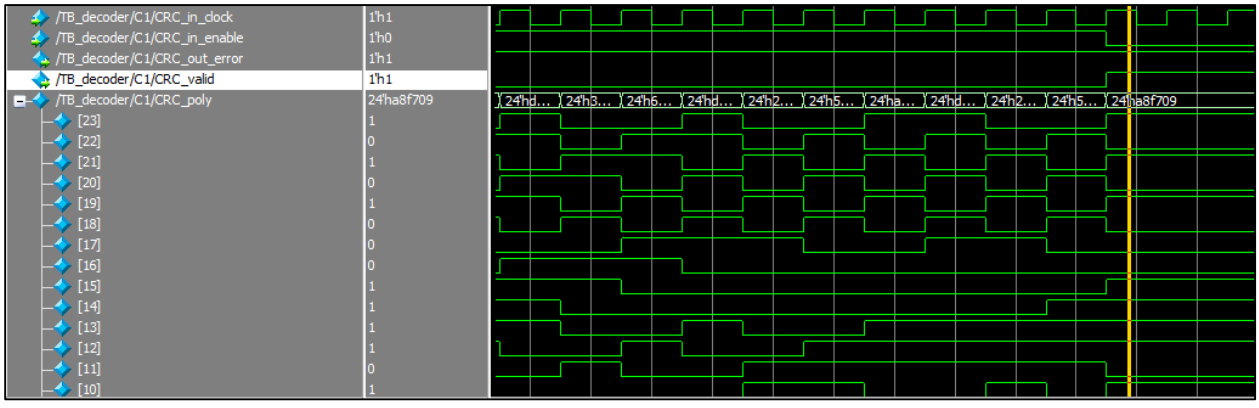


Figure 4.13-4 CRC output with high error detection output

4.13.4. Synthesis Results

4.13.4.1. Area Report

```
Library(s) Used:

    scmetro_tsmc_cl013g_rvt_ss_1p08v_125c (File: /root/tsmc_fb_cl013g_sc/aci/sc-m/synopsys/
scmetro_tsmc_cl013g_rvt_ss_1p08v_125c.db)

Number of ports:                7
Number of nets:               100
Number of cells:               95
Number of references:          12

Combinational area:        411.845002
Noncombinational area:     647.184992
Net Interconnect area:      32.000000

Total cell area:          1059.029994
Total area:               1091.029994
```

Figure 4.13-5 CRC area report

4.13.4.2. Power Report

```
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  =   4.1812 uW   (98%)
  Net Switching Power  =  99.8925 nW    (2%)
                         ---------
Total Dynamic Power    =   4.2811 uW  (100%)

Cell Leakage Power     = 471.7893 nW
```

Figure 4.13-6 CRC power report

Chapter 5
Integration and Conclusion

Integration

First, we integrated each two blocks and most of the blocks explained in chapter 4 were already tested with the output of other blocks. Then we had four separated parts of the chain.

- The first one includes (Synchronization, Offset correction, FFT)
- The second part includes (Resource element De-mapper, channel estimation and equalizer)
- The third part includes (IDFT, De-mapper and descrambler)
- The forth part includes (De-Interleaver, Rate De-Matching, Decoder and CRC)

Each part of them was integrated, synthesized, tested and compared with MATLAB.
Then we integrated the four parts together to get the full chain and made Shure that signals go correctly from one block to the other one.

Conclusion

In this thesis a full chain for narrow band LTE uplink receiver was proposed. The full Architecture for each block was explained. Each block was modelled by MATLAB and was RTL implemented. Each block was tested and the results were verified and compared with MATLAB. The blocks were fully synthesized on Xilinx ISE and on DC compiler and the area and power were reported. Finally, all blocks were integrated and tested.
Out future plan is based on testing more cases for the full chain, optimizing in the blocks to get less area and power consumption and test the chain on FPGA.

References

[1] 3GPP Technical Specifications 36.211, “Physical channel and modulation.

[2] 3GPP Technical Specifications 36.212, “Physical channel and modulation.

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

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

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

[6] M. Sandell, J.-J. van de Beek, and P. O. Borjesson, “ML Estimation of Timing and Frequency Offset in Multicarrier Systems,” Signal Processing, no. April, 1996.

[7] N. P. Samavedam, “Mobile Cell Search and Synchronization in LTE,” vol. 4, no. 5, p. 75, 2011.

[8] B. Lakshmi and A. S. Dhar, “CORDIC architectures: A survey,” VLSI Des., vol. 2010, 2010.

[9] S. Dirlik, “A comparison of FFT processor designs,” 2013.

[10] P. S. Pariyal, D. M. Koyani, D. M. Gandhi, S. F. Yadav, D. J. Shah, and A. Adesara, “Comparison based Analysis of Different FFT Architectures,” Int. J. Image, Graph. Signal Process., vol. 8, no. 6, pp. 41–47, 2016.

[11] M. Eroglu S., Toprak S., Urgan O, MD, Ozge E. Onur, MD, Arzu Denizbasi, MD, Haldun Akoglu, MD, Cigdem Ozpolat, MD, Ebru Akoglu, Digital Signal Processing with Field Programmable Gate Arrays, vol. 33. 2012.

[12] Yu, M. H. Yen, P. A. Hsiung, and S. J. Chen, “A low-power 64-point pipeline FFT/IFFT processor for OFDM applications,” IEEE Trans. Consum. Electron., vol. 57, no. 1, pp. 40–45, 2011.

[13] 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.