A

Dissertation

on

Realization of Address Generator for WiMAX using FPGA

submitted in

partial fulfilment of the requirement

for the award of the degree of

Master of Technology

in

VLSI Design and Embedded System

Submitted
by

Akshay Khalatkar
University Roll No. 2K12/VLS/01

Under the Guidance of

Dr. Asok Bhattacharya
Professor (Retd.),
Department of Electronics and Communication Engineering
Delhi Technological University

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

2012-2014

Electronics and Communication Engineering Department
Delhi Technological University
Delhi-110042
www.dce.edu

Certificate

This is to certify that the dissertation titled "**Realization of Address Generator for WiMAX Using FPGA**" is a bonafide record of work done by **Akshay Khalatkar, Roll No. 2K12/VLS/01** at **Delhi Technological University** for partial fulfilment of the requirements for the degree of Master of Technology in VLSI and Embedded System Design. This project was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Date: _____

(Dr. Asok Bhattacharya)
Professor (Retd.)
Department of Electronics and Communication Engineering
Delhi Technological University

ACKNOWLEDGEMENTS

I would like to express my deep sense of respect and gratitude to my project supervisor **Dr. Asok Bhattacharya,** Professor (Retd.), Electronics and Communication Engineering Department, DTU for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success.

I am also grateful to **Prof. Rajeev Kapoor**, HOD, Electronics and Communication Engineering Department, DTU for his immense support.

A special thanks to **Dr.S.Indu**, Associate Professor, Electronics and Communication Engineering Department, DTU for giving me valuable guidance and support. Her enormous knowledge and investigation has helped me unconditionally to solve various problems. I am also grateful to **Dr.Neeta Pandey** , Associate Professor, Electronics and Communication Engineering Department, DTU for her constant motivation throughout my thesis work.

I would also like to acknowledge Delhi Technological University for providing the right academic resources and environment for this work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents and my colleagues for constantly encouraging me during the completion of work.

Akshay Khalatkar
University Roll no: 2K12/VLS/01
M.Tech (VLSI and Embedded System Design)
Department of Electronics & Communication Engineering
Delhi Technological University
Delhi – 110042

ABSTRACT

The demand for mobile broadband access for internet and multimedia based applications has given rise to scalable network architecture which can provide such services at a lower cost to the operators and the end users. The IEEE 802.16e standard or the WiMAX (Worldwide Interoperability for Microwave Access) standard provides a solution to the Broadband Wireless Access with an extended feature of mobility support.

In the WiMAX transceiver structure a channel coding block is employed in order to prevent the errors during transmission and correct them on the receiver end. The interleaver block in this module performs an important role to reduce the effect of burst errors by spreading the sequential or the adjacent data words through several transmitted bursts. This is done by a sequence of permutation steps, and on the receiver side the reverse operation of these steps is carried out in order to recover the data.

The floor function required for these permutations is difficult to implement on FPGA. So a simple mathematical algorithm has been shown in this work eliminating the use of floor function. Also the modulus 3 function has been proposed using a set of binary parallel adders which can be used as an alternative for ROM circuitry.

In this work, an address generation circuitry for WiMAX deinterleaver has been designed using Verilog HDL and the results have been simulated on ISim Simulator. The proposed work enhances the operating frequency and gives a low complexity circuit as compared to earlier work in terms of performance parameters for FPGA.

KEYWORDS: WiMAX, Field Programmable Gate Arrays (FPGA), Deinterleaver, transceiver, Verilog, Broadband Wireless Access (BWA).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

SDU	Service Data Units
CID	Connector Identifier
WiMAX	Worldwide Interoperability for Microwave Access
NLOS	non-Line Of Sight
MAC	Medium Access Control
OFDMA	Orthogonal Frequency Division Based Multiple Access
BTC	Block Turbo Codes
CTC	Convolutional Turbo Codes
OSI	Open Systems Interface
BPSK	Binary Phase Shift Keying Technique
QPSK	Quadrature Phase Shift Keying Technique
QAM	Quadrature Amplitude Modulation
OFDM	Orthogonal Frequency Division Multiplexing
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
SSI	Self Symbol Interference
ISI	Inter Symbol Interference
SNR	Signal to Noise Ratio
PRBS	Pseudo Random Binary Sequence Generator
FEC	Forward Error Correction Coding
LDPC	Low Density Parity Check Codes

CHAPTER 1

INTRODUCTION

1.1 Motivation

The increase in demand for mobile broadband access for internet and multimedia based applications has given rise to scalable network architecture to provide such services at a lower cost to operators and end users.

The IEEE 802.16 standard or the Worldwide Interoperability for Microwave Access commonly known as WiMAX provides a solution to Broadband Wireless Access. Initially the IEEE 802.16 standard, published in 2001 supported a frequency range of 10-66 Ghz with a theoretical maximum bandwidth of 120 Mb/s and maximum transmission range was estimated to be 50 km [1]. The initial standard supported only the line of sight (LOS) transmission and did not seem to be suitable for urban areas. An alternative to this standard IEEE 802.16a-2003, supported non-LOS transmission (NLOS) and provided OFDM at PHY layer. These earlier drafts had an interoperability problem due to many profiles and PHY layer specifications. The earlier amendments focused on fixed applications (802.16/a/d). In addition to this the IEEE 802.16e standard has provided mobility support.

The IEEE 802.16e standard defines the services and protocols related with the medium access control and physical layer of the OSI reference model. In this model, the physical layer is in charge of transmission of information over a physical medium (electromagnetic waves) and accurate reception of the encoded information. In this aspect a channel coding block is designed to prevent the errors while transmission and reception of signals and correct them in a wireless communication channel [3]. The interleaver block is useful for spreading sequential data words or packets through several transmitted bursts. The process is done by permutation sequence steps described in accordance to IEEE 802.16e standard. The deinterleaver module is responsible for performing the reverse steps of permutation as done by the interleaver block.

In IEEE 802.16e standard block interleaver is typically used with an address generator for generation of addresses where the data bits are to read or written as per the required operation. The address generation requires floor function which is associated with the implementation of the permutation steps, which is difficult to implement in FPGA. In the work described in this thesis, an efficient algorithm for generation of addresses for deinterleaver is designed and a new design for modulus 3 function using a set of binary parallel ideas has been proposed. The design has been verified by using Xilinx ISE Design suite (Version 14.3) with simulation results verified on ISim Simulator.

1.2 Related Work

For the hardware implementation of the address generator for WiMAX deinterleaver very a small number of works are available in the literature. The work in [13] shows that incoming data streams can be grouped into the block to decrease the frequency of memory access in a deinterleaver module by use of a conventional look-up table (LUT)-based address generator for WiMAX.

The work done in [18] by Khater et.al showed a VHDL implementation of address generator for channel interleaver. This work had a drawback that it was done with only code rate of ½. The address generation circuitry for the channel interleaver was improved by the work done by B.K.Upadhaya et.al[15] in which they proposed a finite state machine based address generator for the same interleaver architecture for all code rates with all the modulation schemes used in IEEE 802.16e standards.

The translation of interleaver permutation steps into 2-D functions was done by Asghar and Liu in their work [14] to claim for efficient hardware architecture. But this work did not clearly explain issues related to 64-quadrature amplitude modulation. The design was improved by B.K.Upadhaya et.al [16] where the authors have given low complexity architecture by reducing the algorithm complexity for different modulation schemes in WiMAX standards.

This thesis work improves the earlier design in [16] by introducing a modulus 3 architecture by using a set of binary parallel adders and improving the operating frequency from the earlier design. A comparative study of the performance results

obtained with the work in [16] and the conventional LUT-based technique confirms the improvement in operating frequency and an efficient use of logic resources.

1.3 Objective and Scope of the Project

The objective of this thesis is to design an address generator for WiMAX deinterleaver using Verilog HDL and to verify the results using the ISim Simulator. These results which are obtained are compared with earlier work done on different performance parameters for FPGA.

1.4 Organization of Thesis

The thesis outline is as follows:

Chapter 2: This chapter provides overview of WiMAX technology where the objectives and the features of WiMAX are illustrated.

Chapter 3: This chapter provides an insight to the physical layer of the WiMAX where the different modulation technique and the transmission and reception modules are studied.

Chapter 4: This chapter gives an approach to design of (de)interleaver for WiMAX and also explores the algorithm for the design of address generator for WiMAX deinterleaver.

Chapter 5: In this chapter the digital design of address generator for WiMAX deinterleaver is modeled in Verilog HDL and the simulation result is obtained and performance comparison of FPGA parameter has been done.

Chapter 6: This chapter deals with the conclusion from the interpreted results and explores the future work for this technology.

CHAPTER 2

WiMAX Overview

2.1 Introduction to WiMAX

2.1.1 Objectives

The key technical objectives with which the WiMAX technology was built comprises of the following [4]:

- Orthogonal frequency division based multiple access (OFDMA) which is used as key modulation scheme in WiMAX technology. The modulation scheme also has a scalable bandwidth depending on the type of radio link present at a given location.

- Use of advanced antenna technologies which allowed beam forming and diversity with the use of space time coding and spatial multiplexing.

- Physical layer (PHY) design done so as to provide fast link adaptation and merged with fast time and frequency scheduling.

- Flat-IP Network Architecture which provides the key benefits of reduced system latency along with flexible decoupled radio access and core network evolution. This supports traditional operator managed as well as new internet services.

- Use of open standard interface to enable the key feature of interoperability in multivendor environment. In order to extend this feature based on the service providers requirements the WiMAX forum also defined end to end architectures and protocols to interact with other standards such as Third Generation Partnership Project (3GPP), DSL Forum, and Open Mobile Alliance.

2.1.2 WiMAX Features

WIMAX (Worldwide Interoperability for Microwave Access) refers to the broadband standard that ensures high bandwidth over a long-data transmission and is based on IEEE 802.16 standard. The IEEE 802.16 is the working group of IEEE 802 dedicated to

Broadband Wireless Access whose aim is to provide standards for high data rate WMAN (Wireless Metropolitan Area Networks). Thus the term WiMAX is a term which used very widely with IEEE 802.16 products [2]. This standard specifies the services and protocols related to the lower 2 layers (Data Link and Physical) of the OSI reference model. This is depicted in figure 2.1. Here the data link is subdivided into two layers which are Logical Link Control (LLC) and Medium Access Layer (MAC).

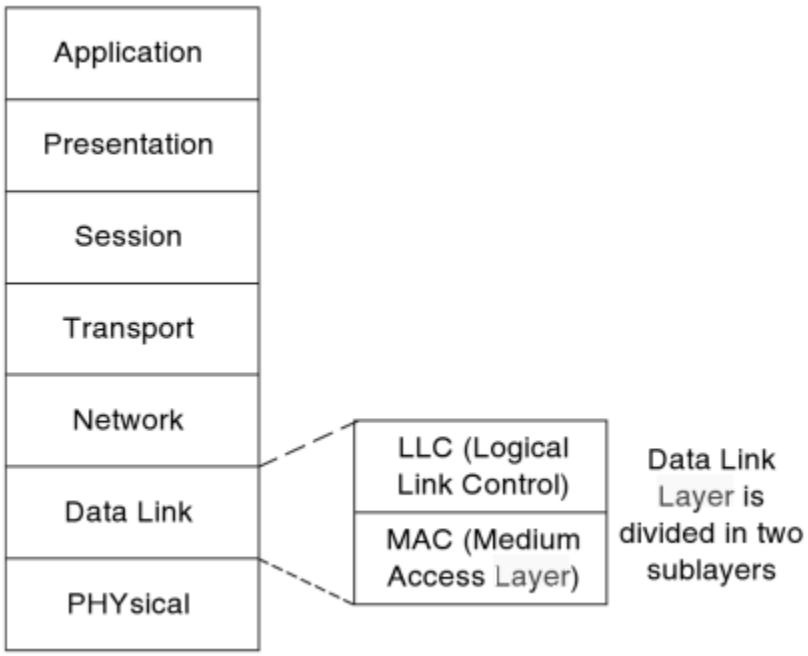

Figure 2.1: Seven Layer OSI Model. In IEEE 802.16 only lower 2 layers are defined [2].

The BWA has a much larger range as compared to WLAN WiFi and compromises of two variants:

- IEEE 802.16-2004 which comprised of fixed wireless access WMAN
- IEEE 802.16e which is an amendment to the earlier act and included mobility and then fast handover

2.2 PHYsical Layer

In WiMAX the data is transmitted at very high speed using the electromagnetic waves at a particular operating frequency via the air interface. The physical layer is responsible for establishment of connection between two sides which are uplink and downlink. The primary responsibility of this layer is efficient bit transmission. All the physical

parameters which are the type of modulation/demodulation, transmission power and signal characteristics are defined by this layer. The PHY layer is based on orthogonal frequency division multiplexing in which multiple access is ensured by assigning subset of subcarriers to each individual user. The OFDM systems is based on assignment of data into parallel sub-streams at a reduced data rate, each modulated and transmitted on separate orthogonal carrier.The WiMAX standard has frequency band consideration of 2-66 GHz. The frequency band is divided into two parts:

- **10-66 GHz Licensed Band -** This band addresses Line of Sight LOS operation and the multipath effect is very less. The channel size varies between 25-28 MHz.

- **2-11 GHz Licensed Band and License Exempt Band** – This band addresses additional functionality which support LOS and NLOS environment in which the effects of multipath propagation are alleviated. Some of the additional features include dynamic frequency selection to mitigate the effect of interference in the license exempt band.

The five modes are tabulated in the table 3.1 with their descriptions. In this the major duplexing modes i.e. Time Division Duplexing (TDD) and Frequency Division Duplexing (FDD) are included with 802.16 systems.

2.3 Medium Access Control (MAC) Layer

The MAC layer is subdivided into three parts:

- Service Specific Convergence Sublayer (CS) - The basic functionality of this layer is to map the data from the upper layers into the appropriate MAC Service Data Units (SDU). This is done with the help of apt MAC Service Flow Identifier (SFID) and Connector ID (CID).
- MAC Common Part Sublayer (MAC CPS) - This layer handles the core function of system access, allocation of bandwidth and connection establishment. This layer is also responsible for Quality of Service (QoS)
- Security Sublayer- This layer handles the functionality of authentication and secure key exchange. The data encryption protocol is applied for both ways for

the connection between SS and BS. In this protocol pairings of encryption of data and different protocols for authentication are included.

Figure 2.2 depicts all the layers of IEEE 802.16 reference model [1].

Designation	Frequency	Duplexing Mode	Notes
WirelessMAN-SC	10–66 GHz	TDD, FDD	Single Carrier
WirelessMANSCa	2–11 GHz Licensed band	TDD, FDD	Single Carrier for NLOS
WirelessMAN OFDM	2–11 GHz Licensed band	TDD, FDD	OFDM for NLOS operation
WirelessMAN OFDMA	2–11 GHz Licensed band	TDD, FDD	OFDM Broken into subgroups to provide multiple access in a single frequency band
WirelessHUMAN	2–11 GHz Licensed Exempt	TDD	May be SC, OFDM, OFDMA. Must include Dynamic Frequency Selection to mitigate interference

Table 2.1: IEEE 802.16 air interface nomenclature and description [6]

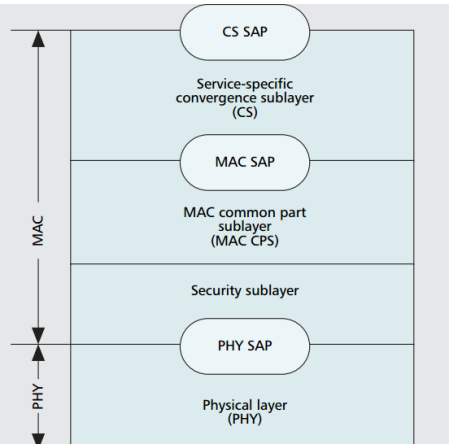

Figure 2.2: IEEE 802.16 Reference Model[1]

CHAPTER 3

WiMAX Physical Layer

3.1 Digital Modulation Techniques

For the wireless communication systems, digital modulation techniques are used. In this the analog signal is modulated with an incoming digital stream to send the message over a given medium which can be fibre or radio link. The digital modulation techniques provide more information capacity and compatibility with digital data services. It also has an advantage of higher data security with a better quality communications combined with quicker system availability.

3.1.1 Binary Phase Shift Keying Technique (BPSK)

Binary Phase Shift Keying is a technique which uses binary digital modulation that is one modulation symbol per bit. This technique uses phase variation to encode the bits and each modulation symbol is equal to one phase. The phase of the modulated signal is π or $-\pi$ according to the value of data bit which is 1 or 0 respectively. This ensures high noise immunity and robust modulation. The depiction of binary phase shift keying is as as per the following figure.

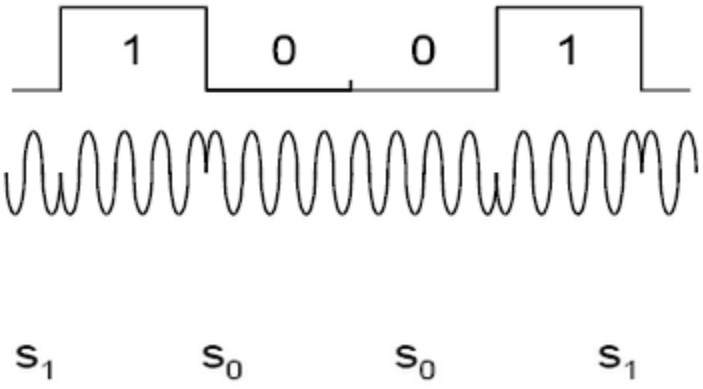

Figure 3.1: BPSK Signal for symbol 1001

The equation for BPSK is given as

$$v_{BPSK}(t) = b(t)\sqrt{2P}cos2\pi f_c t \text{, where } 0 < t < T \quad (3.1)$$

Where b(t) = +1 or -1

f_c =Carrier Frequency

T = bit duration

The constellation diagram for BPSK is as shown below. This shows that the signal phase can take values 0 orπ.

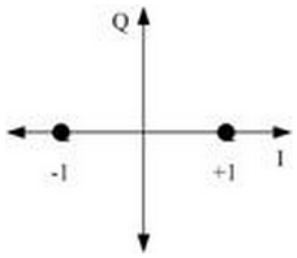

Figure 3.2: BPSK Constellation [3]

3.1.2 Quadrature Phase Shift Keying Technique (QPSK)

In QPSK 2-bit modulation symbols are considered. This is done in order to ensure higher spectral efficiency modulation. The decision at the receiver between two symbol combinations is less easy than a decision between '0' and '1' as in BPSK. Therefore the QPSK has less noise resistance than the BPSK technique.

$$v_{QPSK}(t) = \sqrt{2P}\cos(2\pi f_c t + \varphi(t)) \quad (3.2)$$

Where $\varphi(t) = 135°, 45°, -45°, -135°$

The QPSK modulation scheme is illustrated in Figure 3.3. For the constellation diagram the four symbols are placed in four quadrants of I-Q curve. Thus the four combinations (00, 01, 10, 11) can be seen from the Figure 3.4.

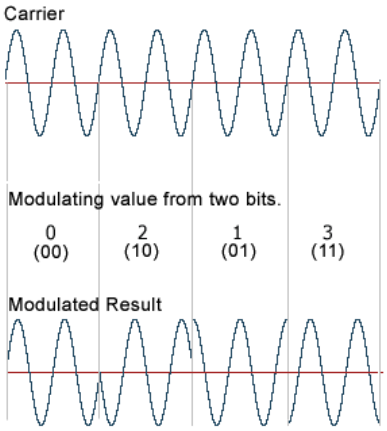

Figure 3.3: QPSK Modulation [3]

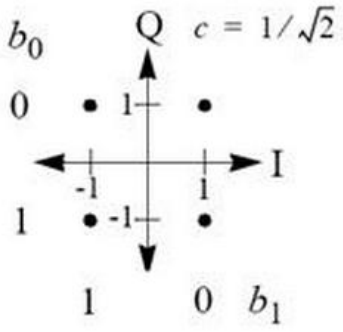

Figure 3.4: QPSK Constellation [3]

3.1.3 Quadrature Amplitude Modulation (QAM) – (16-QAM and 64-QAM)

The Quadrature modulation modulates the amplitude of two sinusoidal carriers on the basis of the digital sequence which is to be sent and the two carriers are in out of phase with each other +π/2. Thus QAM-4 and QPSK are same modulation. The 16-QAM(4-bits per symbol) and 64-QAM (6-bits per symbol) modulation are both used in WiMAX.

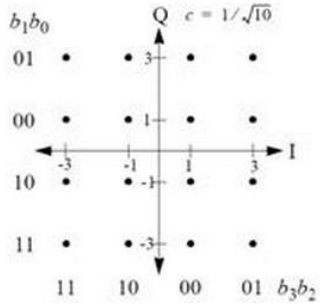

Figure 3.5: 16-QAM Constellation [3]

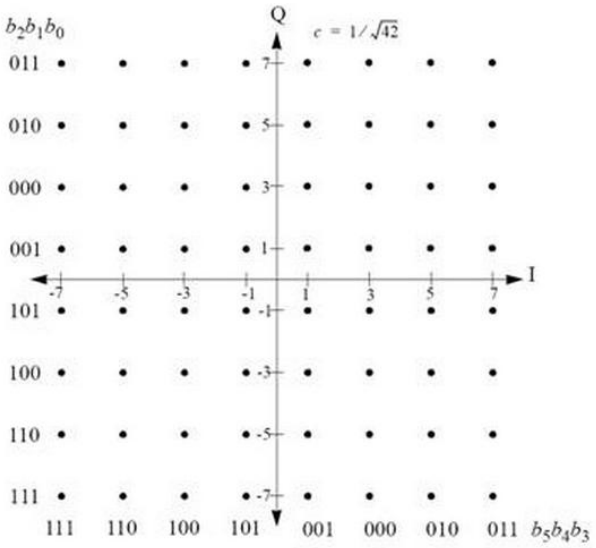

Figure 3.6: 64-QAM Constellation [3]

$$v_{QAM}(t) = A_k sin2\pi f_c t + B_k cos2\pi f_c t \quad (3.3)$$

The performance of various modulation schemes [12] can be seen from the bit error rate versus signal to noise ratio graph shown in figure 3.7. It can be seen that 64-QAM has the highest bit error rate when the signal to noise ratio increases.

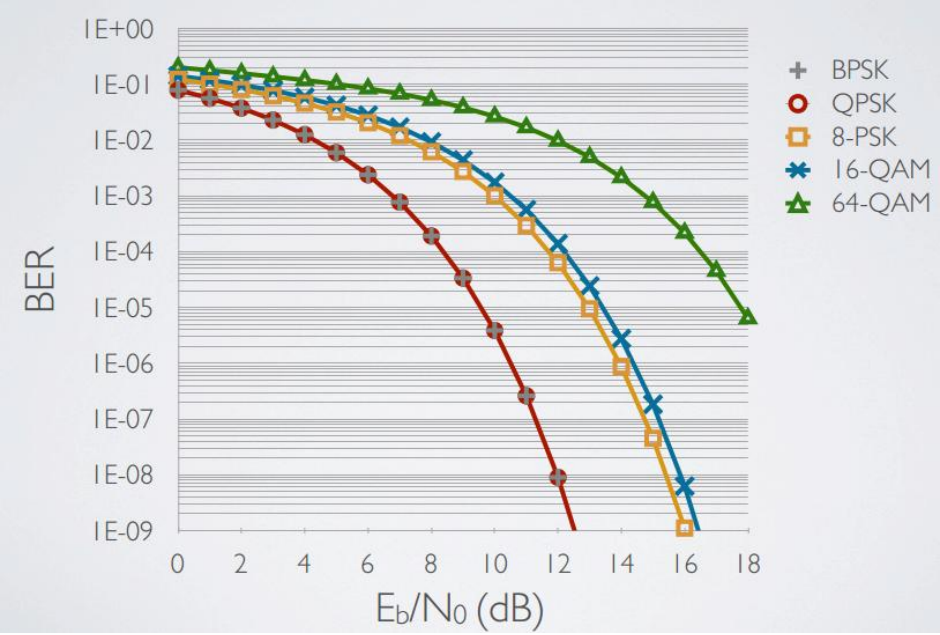

Figure 3.7: Bit Error Rate versus SNR of different modulation schemes [12]

3.2 Orthogonal Frequency Division Multiplexing (OFDM)

OFDM is a method of transmitting information over a physical channel in which multiple carriers are utilized to transmit a large number of symbols concurrently, and also distributing information blocks containing certain number of bits to a certain number of carriers.

The OFDM concept comes from multi-carrier modulation in which each input bit stream is divided into several parallel bit streams. These are used to modulate the sub carriers as depicted in the following Figure 3.8.

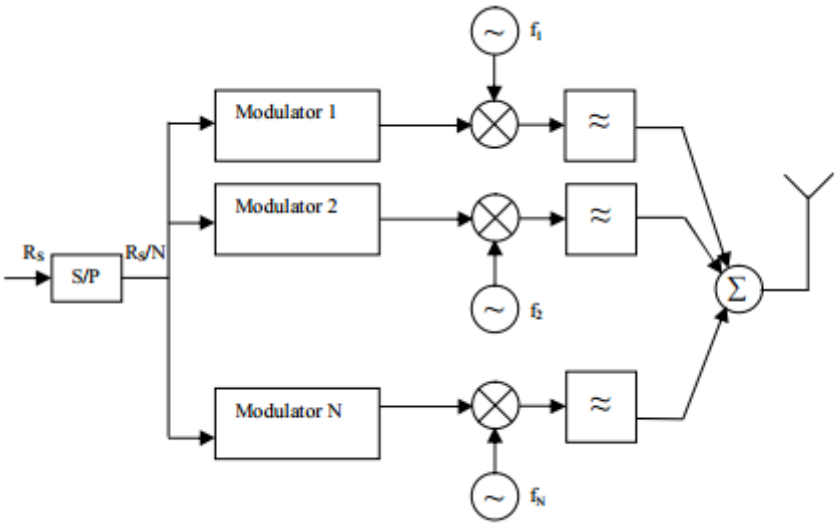

Figure 3.8: Multi Carrier Modulation [6]

Here each sub-carrier is separated by a guard band so that they do not interfere with each other. Subsequently on the receiver end the band pass filters are employed to separate the obtained spectrum of each sub carrier. In OFDM the above technique is optimized by employing densely spaced orthogonal sub-carriers with overlapping spectrums. This eliminates the use of band pass filters on the receiver end since the sun-carriers are orthogonal in nature. This also ensures that the bandwidth is used more efficiently and the inter carrier interference (ICI) is also reduced. The following figure shows the comparison of the FDM and OFDM technique. The savings in bandwidth is seen from this figure. The orthogonality is achieved by taking Fast Fourier Transform of the input stream.

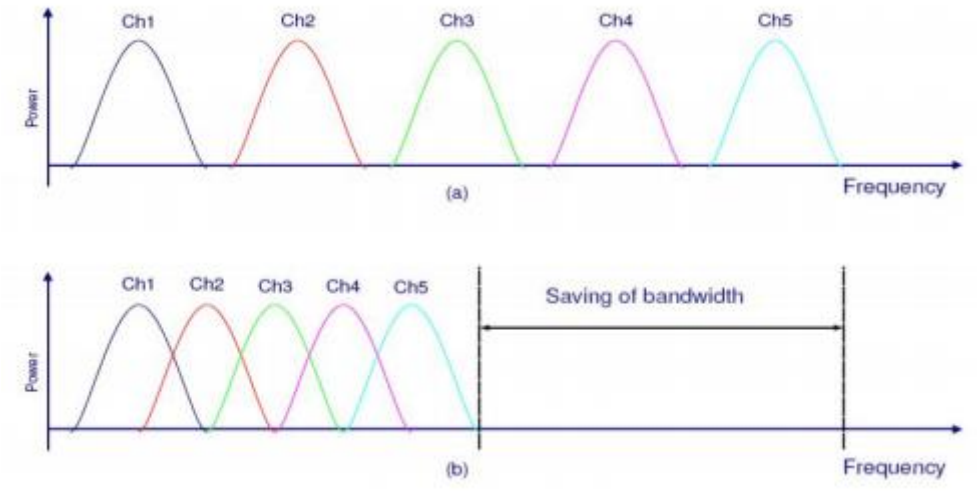

Figure 3.9: Comparison of FDM and OFDM [5]

The advantages of OFDM include high spectrum efficiency, resistance towards multipath interference which becomes particularly useful in wireless communication and the noise interference becomes easy to filter since those frequencies affected by noise can be disabled if need arises. The upstream and downstream speed becomes scalable by allotting different number of carriers for each purpose.

One of the key advantages of using multiple subcarriers is that because each carrier operates at a relatively low bit rate, the length of each symbol is relatively long. For example, if we want to send a million bits per second over a single baseband channel, then the duration of each bit should be under a microsecond. This imposes severe barrier on synchronization and removal of multipath interference. The same million bits per second when spread among N subcarriers, the duration of each bit can be increased by a factor of N, and hence the constraints of timing and multipath sensitivity are greatly relaxed. For non-stationary objects, the Doppler Effect on signal timing is another factor that causes difficulties for some other modulation schemes. The more intricate the technique and the greater the information bandwidth that is number of useful bits that can be transmitted when using 1 Hz bandwidth, the more the systems are prone to disturbing effects (such as fading, noise, transmitter and receiver imperfections).

Data transmission is large as compared to FDM as OFDM follows multicarrier modulation. For this, OFDM splits high data bits into low data bits and sends each sub-stream in several parallel sub-channels, known as OFDM subcarriers. These subcarriers are orthogonal to each other and the each subcarrier bandwidth is much lesser than the total bandwidth. Inter Symbol Interference is reduced in OFDM technique as the symbol time T_s of each sub-channel is higher than the channel delay spread ς.

3.2.1 OFDM Implementation

In order to efficiently implement OFDM algorithm into chip Cooley and Tuckey [7] gave an algorithm for efficient FFT calculation, by virtue of which the implementation of FFT and IFFT algorithm on a chip became possible.

The Discrete Fourier Transform (DFT) and the Inverse Discrete Fourier Transform (IDFT) are used to obtain the OFDM signal. The DFT of a signal is taken over a discrete time signal and transformation gives a discrete or M number of components in the frequency domain.

The formula for DFT is given for a M-length DFT as:

$$X_k = \frac{1}{\sqrt{M}} X(e^{\frac{j2\pi k}{M}}) \tag{3.4}$$

$$X_k = \frac{1}{\sqrt{M}} \sum_{i=-\infty}^{\infty} x_i e^{-\frac{j2\pi ki}{M}} \tag{3.5}$$

The inverse DFT is given to recover x[n] from X_k given as

$$x[n] = \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} X_i e^{\frac{j2\pi ni}{M}} \tag{3.6}$$

For $n = 0.1,2 \ldots . M - 1.$

Here we assume that $x[n] = 0$ for n<0 and n> M-1.

For system implementation the OFDM make use of FFT and IFFT blocks which are mathematical equivalents of the DFT and IDFT as given by the previous equations.

Now if we consider a discrete time situation, in any symbol interval M symbols are to be transmitted over M carriers.

If the symbols are represented $X_0, X_1, X_2 \ldots X_{M-1}$ and the M carriers are represented by $\emptyset_0[n], \emptyset_1[n], \emptyset_2[n] \ldots \emptyset_{M-1}[n]$ then the resultant signal is given by

$$x[n] = X_0\emptyset_0[n] + X_1\emptyset_1[n] + X_2\emptyset_2[n] + \ldots . X_{M-1}\emptyset_{M-1}[n] \quad (3.7)$$

So the block diagram is as shown in the Figure 3.5.

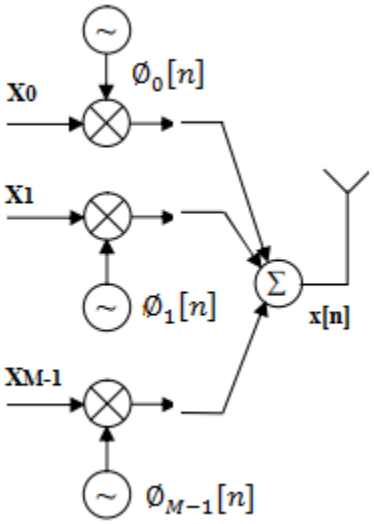

Figure 3.10: OFDM Signal Generation

If the rows of IDFT matrix are taken as carriers

$$\emptyset_i[n] = \frac{1}{\sqrt{M}}(e^{\frac{j2\pi in}{M}}) \quad (3.8)$$

Thus the value of transmitted signal is equal to

$$x[n] = \frac{1}{\sqrt{M}}\sum_{i=-\infty}^{\infty} X_i e^{\frac{j2\pi ni}{M}} = IDFT_n(X) \quad (3.9)$$

Thus the transmitted signal can be implemented using the following block diagram shown in Figure 3.12.

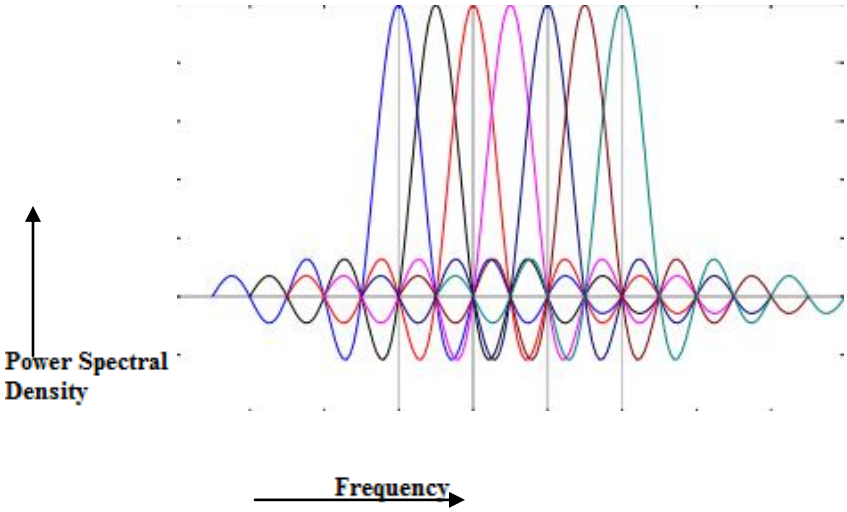

Figure 3.11: OFDM Spectrum

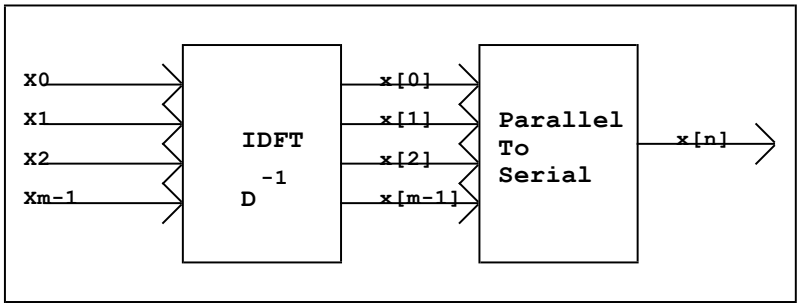

Figure 3.12: Transmitter Structure

The receiver structure can be similarly implemented and is shown in the Figure 3.13.

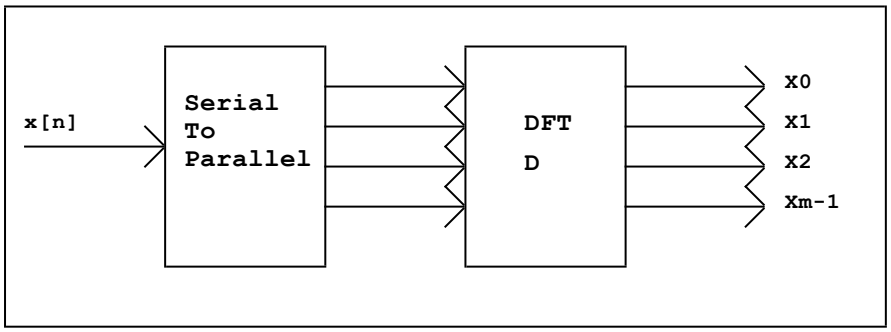

Figure 3.13: Receiver Structure

The DFT and IDFT blocks are reinstated with FFT and IFFT blocks for their digital implementation on the chip. These are shown in Figure 3.14 and Figure 3.15 respectively.

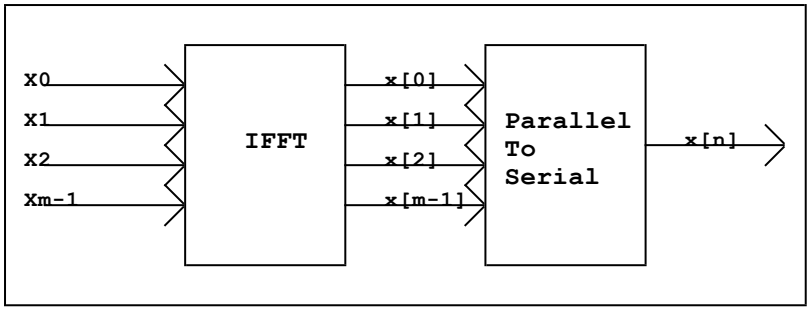

Figure 3.14: Transmitter Structure using IFFT block

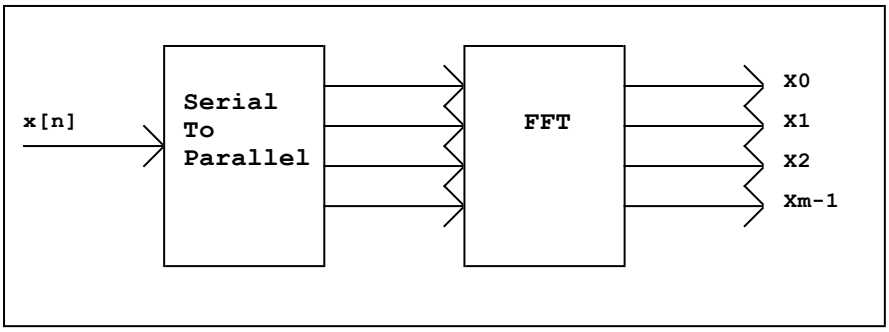

Figure 3.15: Receiver Structure using FFT block

3.2.2 Multi Path Scenario in OFDM

When the transmitted signal is obtained on the receiver end in an OFDM modulation the signal is comprised of not only direct path signal which can be line-of sight signals but also reflected signals which can be from reflection from stationary objects (such as trees, buildings). This can be termed multi path scenario in which the same signal reaches the receiver at different times. Figure 3.16 illustrates this phenomenon. From the figure it is observed the following phenomenon occurs:

- The same symbols are added up from the different paths and delays (denoted by T_1, T_2 and T_3). This is known as **Self Symbol Interference** (SSI) which is depicted in the blue regions where either symbol 1 or symbol 2 is added with the shifted versions of the same signal.

- The other observable fact is the addition of the two different symbols of the shifted version of the same signal which comes from different delays denoted by T_1, T_2 and T_3. This is known as **Inter Symbol Interference** (ISI) which is shown in red regions.

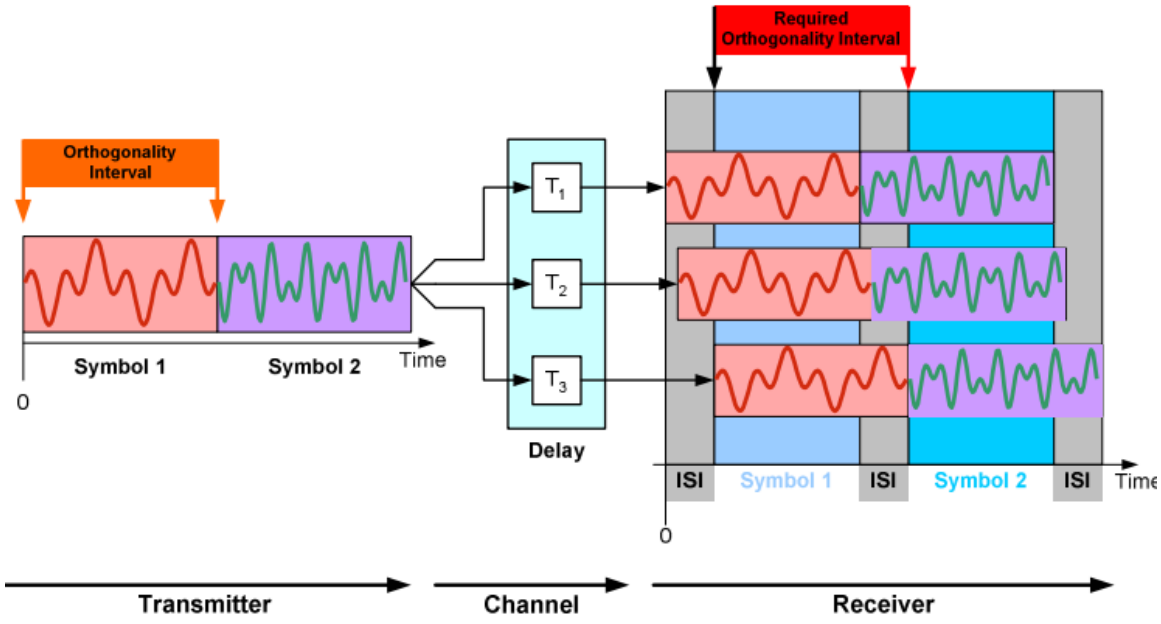

Figure 3.16: Multi Path Scenario in OFDM [9]

The **Self Symbol Interference (SSI)** is beneficial to the system as the symbols of the same type add up and provide a higher level of amplitude which is used for FFT carriers. This advantage is valid when the orthogonality of the carriers is maintained and the sampling frequency is as given by the following equation.

$$F_s = \Delta f. N_{FFT} \tag{3.10}$$

The **Inter Symbol Interference** (ISI) is not advantageous and cannot be corrected at the receiver's side since the receiver does not have the next symbol which it has to receive. So, in order to correct this error Guard period is appended as shown in the Figure 3.17.

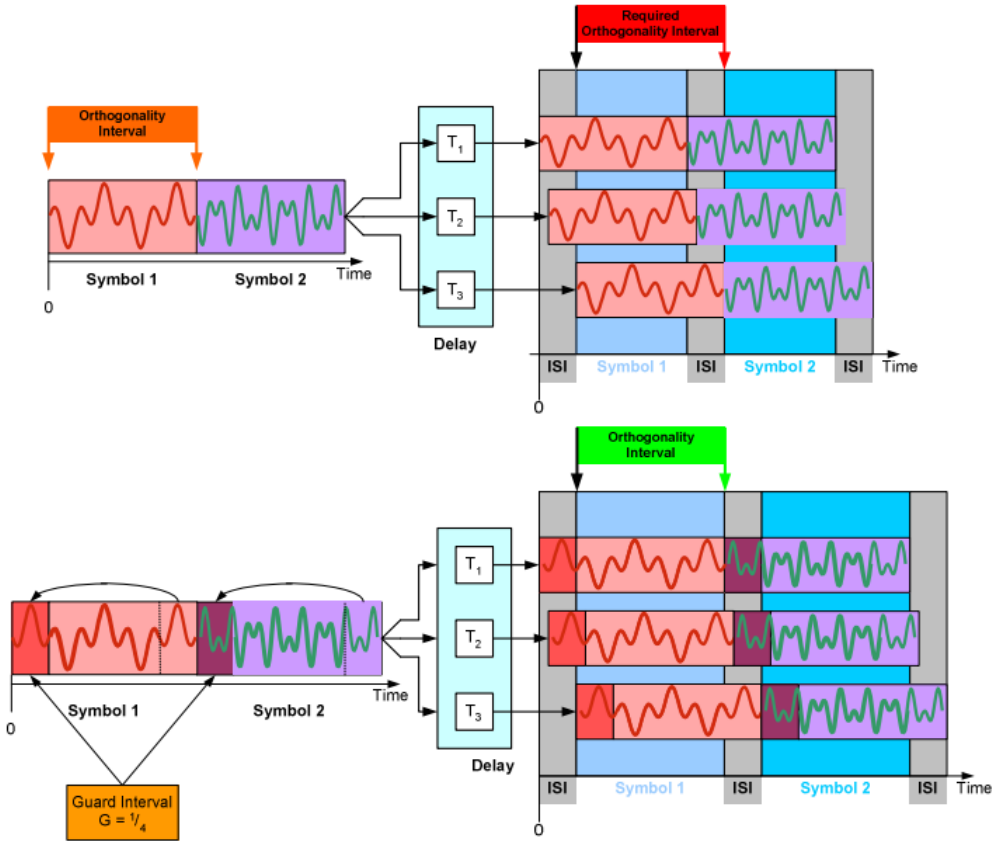

Figure 3.17: OFDM Transmission with and without Guard Interval [9]

The last part of the OFDM symbol to be transmitted is added to the front due to which the receiver is able to receive the same signal for a longer time and thus the interference is prevented in this case.

The length of the guard interval is dependent on the operating conditions and longer paths require larger guard interval. A tradeoff has to be done with increasing the guard interval and data throughput time since the same symbol will require more time to transmit thereby decreasing the number of symbols that can be transmitted in particular time unit.

3.2.3 OFDM Terminologies

- **Channel Bandwidth** (Hz) - It is defined as the bandwidth dedicated to the OFDM system provided by the governing body.

- **Used Bandwidth** (Hz) - The bandwidth which is engaged physically by the WiMAX signal. This is normally lesser than the channel bandwidth.

- **Sampling Frequency** F_s **–** The frequency at which the new samples are obtained from the A/D converter is known as sampling frequency.

- **Sampling Factor** n – The sampling factor is defined as the ratio of the sampling frequency to the bandwidth. This is given by

$$n = \frac{F_s}{BW} \tag{3.11}$$

- **FFT Size** N_{FFT} – This size specifies the number of samples that are used for the FFT block used to generate orthogonal signals. This is always taken in power of 2.

- **Sub-Carrier Spacing** Δf – This is indicated as the interval between two adjacent physical OFDM carriers. The sub carrier spacing is calculated by the following equation.

$$\Delta f = \frac{F_s}{N_{FFT}} \tag{3.12}$$

- **Useful Symbol Time** T_b – It is defined as the time for which the symbol remains valid or undisturbed. This is given by

$$T_b = \frac{1}{\Delta f} \tag{3.13}$$

- **Guard Period Interval (Cyclic Prefix Time)** T_g – The ratio of the useful symbol added to the OFDM symbol while taking into account multi path scenario is known as the guard period interval. The absolute time which is calculated from the above ratio is known as the cyclic prefix.

$$T_g = G.T_b \tag{3.14}$$

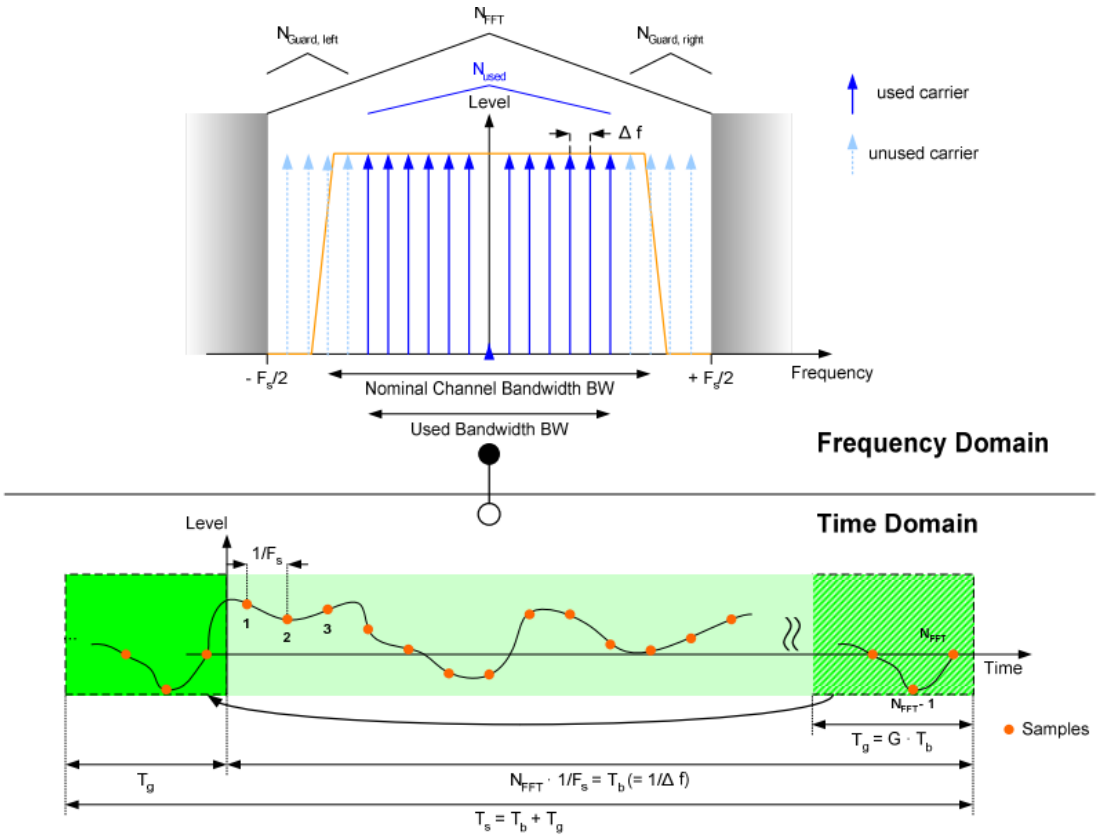

Figure 3.18: OFDM Terms Definition [9]

3.2.4 OFDM Calculation

This section shows the calculation of the main factors used for OFDM WiMAX system signal calculation

- FFT size N_{FFT} = 256 carriers
- User carriers N_{used} = 200
- Pilot carriers N_{Pilot} = 8
- System bandwidth BW = 7 MHz
- Sampling factor n $= \frac{8}{7}$

The parameters can be calculated as

Sampling Frequency = $n.BW = \frac{8}{7}.7 =$ 8 MHz

Carrier Spacing Δf = $\frac{F_s}{N_{FFT}} = \frac{8\,MHz}{256} = 31.25\,KHz$

Useful Symbol Time $T_b = \frac{1}{\Delta f} = \frac{1}{31.25KHz} = 32\,\mu s$

The calculation of the guard band interval is calculated as follows assuming that the guard period interval ratio **G** is equal to 1/16.

$$T_g = G.T_b = \frac{1}{16}.32\mu s = 2\mu s$$

3.3 OFDMA

The OFDM technique introduced in the previous section used a single signal for transmission. Now in order to transmit multiple user signals the OFDM technique is linked to multi user access technique which can be TDMA (Time Division Multiple Access) or FDMA (Frequency Division Multiple Access). This is illustrated in the following Figure 3.19.

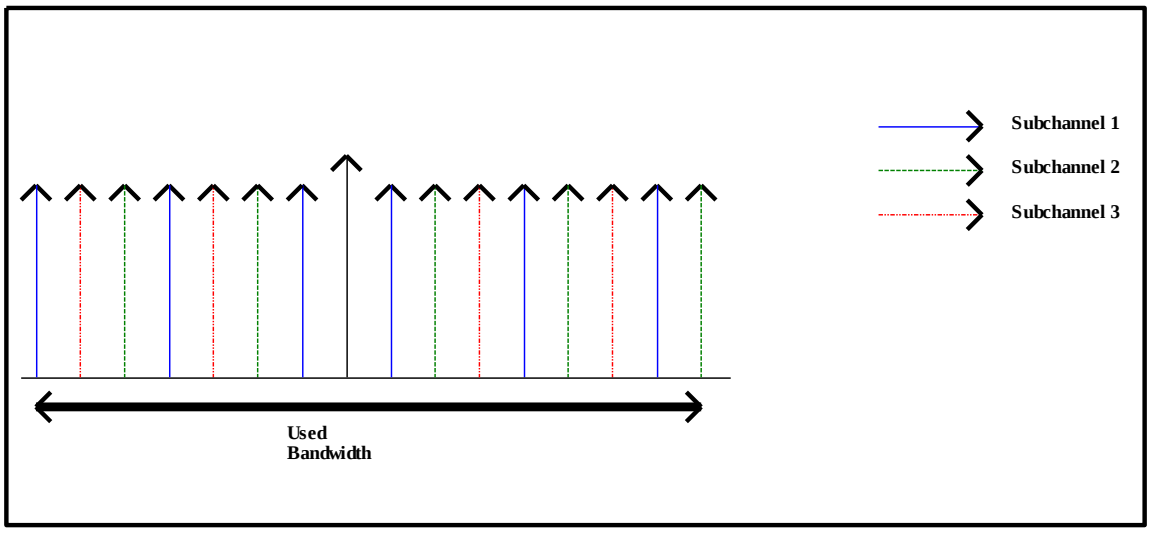

Figure 3.19: OFDMA

These advantages of OFDM are further extended for multiple access schemes by handing over a subset of subcarriers or tones of OFDM to individual users. This multiple access technique is termed as OFDMA. The allocation of subsets of tones to various users allows for concurrent transmission of data from multiple users, allowing for sharing the physical medium. Although this technique looks similar to FDMA, the large guard bands required in FDMA are not needed in OFDMA.

3.3.1 Subchannelization

In OFDMA, the active subcarriers are divided into subsets of subcarriers. Each subset represents a subchannel, as shown in the Figure 3.14. These sub-carriers that form a single subchannel need not be contiguous. Thus, an OFDM symbol is subdivided into several subchannels by grouping the subcarriers. In the downlink, a single subchannel may be intended for different receivers whereas, in the uplink, a transmitter may be assigned one or more subchannels, and all transmitters may transmit simultaneously.

3.3.2 Advantages of OFDMA in WiMAX

In OFDMA, subchannelization defines subchannels that can be assigned to subcarrier stations depending on their channel conditions and data requirements. Many Subscriber Stations can transmit in the same time slot over several subchannels. On the basis of the channel conditions and data requirements modulation and coding is set individually for each subscriber. The transmitter power can be adapted according to the requirements, which optimizes the use of network resources. Due to subchannelization OFDMA signals are more complex than OFDM ones but offer better performance and scalability. This feature is very useful for WiMAX BSs. By using subchannelization, within the same time slot, the BS is able to allocate more transmitter power to those SSs with lower SNR and less power to the ones with higher SNR. This feature also enables the BS to allocate higher power to subchannels assigned to indoor SSs, which results in better in-building coverage. Subchannelization in the uplink saves the power of the user device by concentrating power to the selected subchannels allocated to it. This power saving feature is indeed very useful for battery powered SSs.

3.4 Channel Coding in WiMAX

The OFDM modulation and transmission is the major building block for the WiMAX physical layer. The radio link which is used in wireless transmission systems have a problem of interference associated with them. In order to correctly transmit and receive data in wireless transmission systems the channel coding techniques are used. The primary function of the channel coding is to prevent the errors and also correct them in a

wireless communication system. This is an essential requirement for a high data rate system.

The channel coding consists of the following blocks in the order listed below:

- Randomizer
- Forward Error Correction
- Interleaving

These blocks are applied in this order in the transmission side. A reverse order is used on receiver link. The channel coding chain is illustrated in the Figure 3.20.

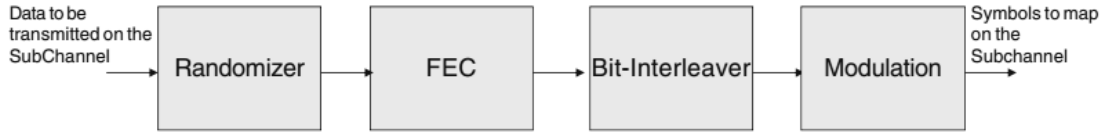

Figure 3.20: Channel Coding in WiMAX for OFDM-PHY [10]

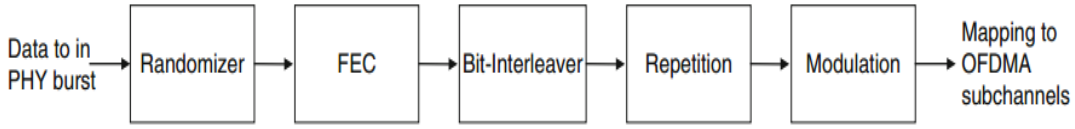

Figure 3.21: Channel Coding in WiMAX for OFDMA-PHY [10]

3.4.1 Randomization

Data randomization is performed on both the uplink as well as the downlink in order to avoid the long sequences of consecutive ones or zeros. The data stream is padded with 0XFF or a sequence of ones only if the amount of data to be transmitted does not match the allocated data transmission length. The uncertainty or the randomness of data being transmitted is achieved with the help of Pseudo Random Binary Sequence Generator (PRBS) as shown in the figure.

The polynomial for PRBS generator is given as

$$f(X) = 1 + X^{14} + X^{15} \quad (3.15)$$

Here the data bytes consist of the information bytes which are to be transmitted from the wireless system and these exclude the preamble which is used with the information bytes.

Data byte which is to be sent enters in a sequential manner into the PRBS Generator. The most significant byte enters first followed by the sequence of significant lower bytes.

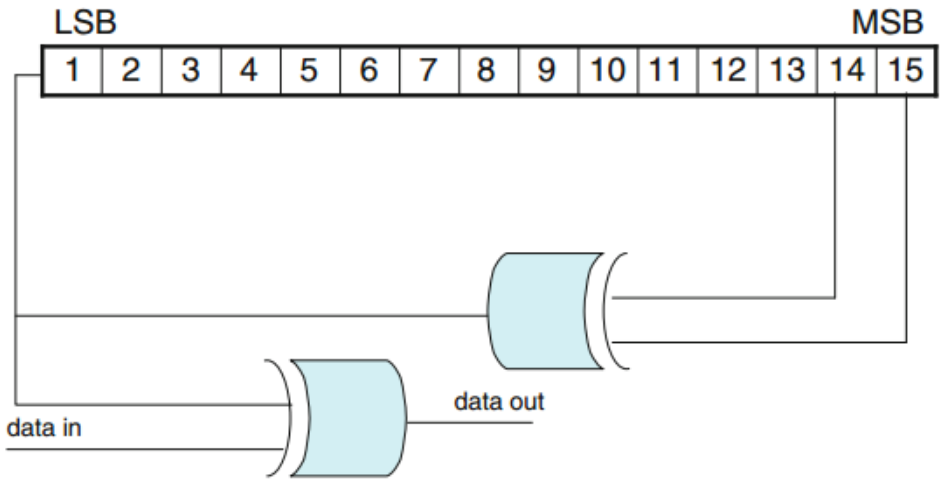

Figure 3.22: PRBS Generator for Randomization [3]

For each new burst the shift registers are reinitialized. In the case of OFDM PHY the downlink randomizer is initialized with a sequence 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0. The randomizer does not reset after the first burst. After the second burst the initialization vector is depicted as per the following figure.3.23.

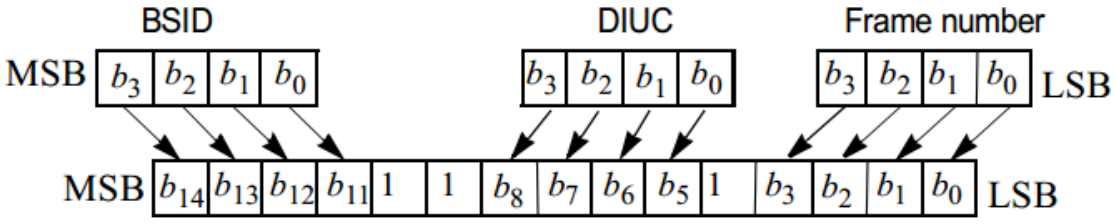

Figure 3.23: OFDM Randomizer Downlink Initialization Vector for Burst 2…N [3]

Similarly for the uplink the initialization vector is illustrated in the Figure 3.24.

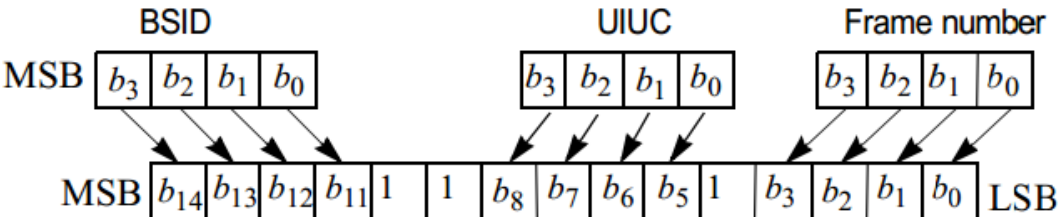

Figure 3.24: OFDM Randomizer Uplink Initialization Vector for Burst 2…N [3]

3.4.2 Forward Error Correction Coding (FEC)

The forward error correction is applied in order to provide redundancy to the data before it is being transmitted. The channel coding in OFDM PHY can be done by three methods:

- Reed Solomon concatenated with convolutional coding(RS-CC)
- Block Turbo Codes(BTC)
- Convolutional Turbo Codes(CTC)

For FEC the RS-CC block is mandatory whereas the other two coding schemes are optional.

For the OFDMA-PHY the following coding block is needed:

- Convolutional Code(CC)- This coding is mandatory
- Block Turbo Codes(BTC)
- Low Density Parity Check Codes(LDPC)

3.4.2.1 Reed Solomon concatenated with Convolutional Coding (RS-CC)

The RS-CC coding is done for the OFDM-PHY by passing the data in the block format from the randomizer to a Reed Solomon Encoder (RS), which serves as outer coding block and then passing the same data through a convolutional coder(CC).

The RS encoder is employed in many communication applications. The RS error correction adds redundant bits to the data sequence. First the polynomial is constructed using uncoded data and then its oversampling is done. The oversampling of the uncoded

data is done in order to make sure the receiver can recover the original polynomial in presence of errors.

The RS encoding is given by RS(N,K) with T-bit symbols where

N - Overall bytes after encoding,

K- The number of data bytes before encoding

T- Number of data bytes which can be corrected

The number of T-bit symbols in an encoded block is given by

$$N = 2^T - 1 \tag{3.16}$$

So for 8 bit symbol

$N = 2^8 - - = 255$ symbols per coded block.

The number K where K<N, is a design parameter where the number of parity symbols added is equal to N-K of (T-bits each).The decoder can therefore correct $\frac{N-K}{2}$ symbols that contain error within the encoded block.

In OFDM PHY the RS encoder is given by (N,K) =(255,239) code which is able to correct 8 symbol errors per block. The Reed Solomon Encoding uses a Galois Field operator GF(2^8) where Code generator polynomial is given by

$$g(x) = (x + \lambda^0)(x + \lambda^1)(x + \lambda^2) \ldots . (x + \lambda^{2T-1}),\ \lambda = 02(HEX) \tag{3.17}$$

Field generator polynomial is given by

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{3.18}$$

The coding rate of OFDM PHY RS encoder is equal to 239/255. This is shortened and punctured in order to enable the variable block size and also variable error-correction capabilities.

The RS encoded block is then fed to a convolutional coder with coding rate of ½. It is a zero-terminating convolutional encoder that is a single 0X00 tail byte is added to end of each burst. This is needed for decoding algorithm. The convolutional coder with a coding rate of ½ is shown in the Figure 3.25.The RS-CC block is shown in Figure 3.26.

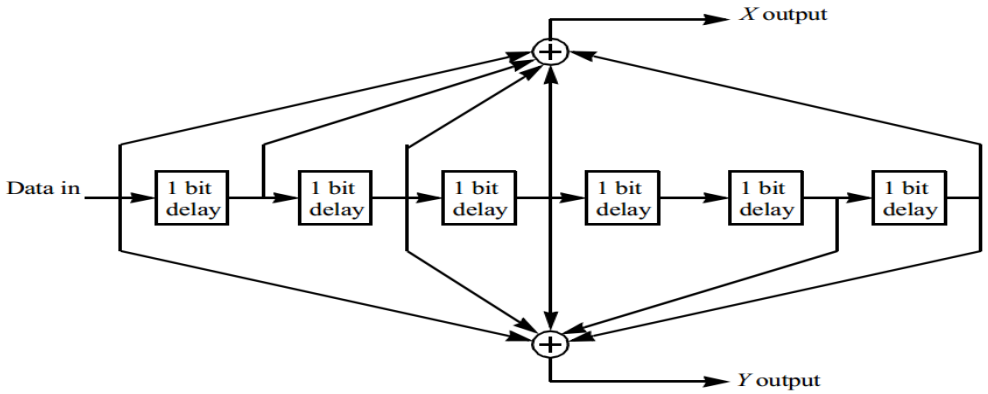

Figure 3.25: Convolutional Encoder with Coding Rate ½ [3]

Modulation	Uncoded block size	Coded block size (bytes)	Overall Coding Rate	RS Code	CC Code Rate
BPSK	12	24	½	(12,12,0)	½
QPSK	24	48	½	(32,24,4)	2/3
QPSK	36	48	¾	(40,36,2)	5/6
16-QAM	48	96	½	(64,48,8)	2/3
16-QAM	72	96	3/4	(80,72,4)	5/6
64-QAM	96	144	2/3	(108,96,6)	¾
64-QAM	108	144	¾	(120,108,6)	5/6

Table 3.1: Mandatory Channel Coding per Modulation

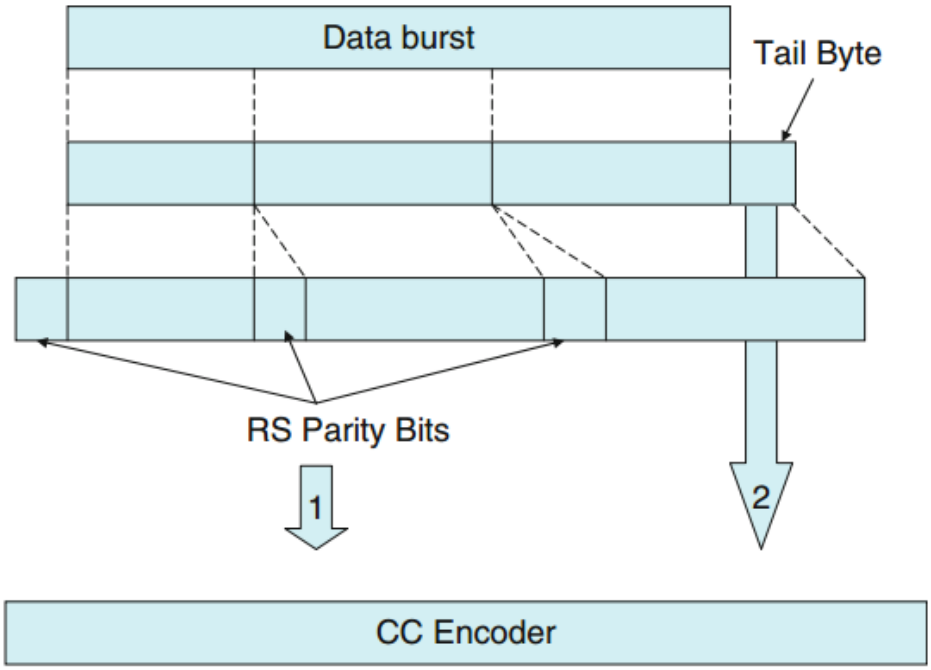

Figure 3.26: RS-CC Encoder [10]

3.4.3 Interleaving

Interleaving is a method in which sequential data words or packets are spread through several transmitted bursts. This is done in order to ensure the transmission against long sequences of adjacent errors, since they are difficult to correct on the receiver side. These errors can cause burst losses if the errors are found in the adjacent bits. All encoded data bits are interleaved using a block interleaver with a block size equal to the number of coded bits per the subchannels allocated per OFDM symbol, denoted by N_{cbps}.

The interleaver is characterized by a two step permutation. The first permutation is done so that the adjacent coded bits are map onto nonadjacent subcarriers. The second permutation is done for the adjacent coded bits to be mapped alternately to the less or more significant bits of the constellation, thereby preventing long runs of lowly reliable bits. After bit interleaving, the data bits are entered serially to the constellation mapper. BPSK, QPSK, 16-QAM, and 64-QAM are supported, whereas the support of 64-QAM is optional for license-exempt bands. Pilot subcarriers are inserted into each data burst in

order to constitute the Symbol and they are modulated by their carrier location within the OFDM symbol.

Let,

N_{cpc} - number of coded bits per subcarrier which is equal to 1, 2, 4 or 6 for BPSK, QPSK, 16-QAM or 64-QAM respectively

k - index of the coded bit before the first permutation,

m_k - index of the coded bit after the first and before the second permutation

j_k - index after the second permutation, just before modulation mapping

Let $s = ceil(\frac{N_{cpc}}{2})$

The first permutation is given by

$$m_k = \frac{N_{cbps}}{12}.k_{mod(12)} + floor\left(\frac{k}{12}\right),\ k = 0,1,\dots N_{cbps} - 1 \quad (3.19)$$

The second permutation is given by

$$j_k = s.floor\left(\frac{m_k}{s}\right) + (m_k + N_{cbps} - floor(12.\frac{m_k}{N_{cbps}}))_{mod(s)},\ j = 0,1,\dots N_{cbps} - 1 \quad (3.20)$$

The deinterleaver carries out inverse operation by two step permutation. When a receiver has block of Ncbps bits

Let,

j - Index of a received bit before the first permutation

m_j - index of that bit after the first and before the second permutation;

k_j - index of that bit after the second permutation, just before delivering the block to the decoder

$$m_j = s.floor\left(\frac{j}{s}\right) + (j + floor\left(12.\frac{j}{N_{cbps}}\right))_{mod(s)},\ j = 0,1,\dots N_{cbps} - 1 \quad (3.21)$$

The second permutation step is given by the following equation

$$k_j = 12.m_j - \left(N_{cbps} - 1\right).floor(12.\frac{m_j}{N_{cbps}}) \quad (3.22)$$

3.4.4 Repetition

This block is added to the existing channel coding chain for OFDMA–PHY to increase the signal margin. For the uplink transmission the number of allocated slots N_s will be multiple of repetition factor R which can be 2, 4 or 6. In case of downlink the allocated slots will be in the range $R \times K, R \times K + (R - 1)$where K is the number of required slots before application of the repetition scheme. So after the FEC and bit-interleaving the data is segmented into slots and these slots are then repeated R times to form R contiguous slots. The slots are placed to the normal slot ordering which is used for data mapping. This is only applied to QPSK modulation scheme and can use all coding schemes except Hybrid Automatic Request with CTC.

3.4.5 Block Turbo Codes

Block Turbo Codes (BTC) is optional FEC for OFDM and OFDMA PHY. In BTC the product of two simple component codes is calculated, which can be binary extended Hamming codes or parity check codes.

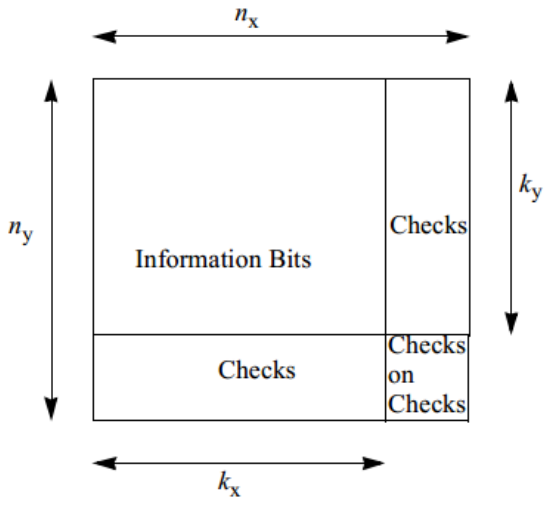

Figure 3.27: Block Turbo Code [3]

The data bit ordering for the composite BTC matrix is done in such a manner that the first bit in the first row is the LSB (Least Significant Byte) followed by last data bit in the last row to be the MSB. The overall block size of product code is $n = n_x . n_y$ and total information bits is $k = k_x . k_y$ and code rate is equal to$R = R_x . R_y$, where $R_i = \frac{k_i}{n_i}$, where$i = \text{x, y}$. This is depicted in Figure 3.27.

3.4.6 Convolutional Turbo Codes

The convolutional code uses a double binary Circular Recursive Systematic Convolutional code. In this the bits of data to be encoded are alternatively fed to A and B in Figure 3.24 with the MSB being fed to A. The encoder is then fed with k bits (N couples where k=2N bits). In this for all frame sizes k is a multiple of 8 and N is multiple of 4 where 8≤N/4≤1024. The size of the encoding block is dependent on the number of subchannels allocated and the type of modulation used for the transmission. In order to make larger blocks of coding concatenation of subchannels can be done. The CTC packet generation can be illustrated from the following Figure 3.24. The CTC encoded code-word is first passed through coding rate of 1/3 and interleaving and puncturing is done.

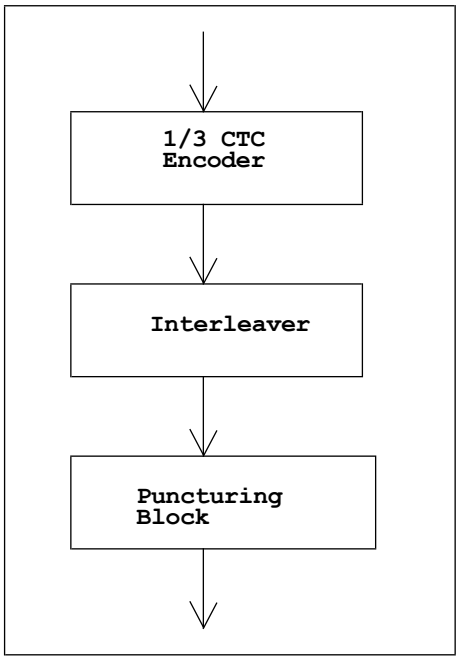

Figure 3.28: Subpacket Generation Block Diagram [3]

CHAPTER 4

INTERLEAVER DESIGN

The scheme of interleaving is to rearrange the original sequence of the transmitted data to a new one so that the burst of errors which occur in the interleaved sequence during the transmission phase can be spread into different locations in the deinterleaved sequence.

In order to rearrange the data stream to be transmitted two approaches are used as depicted from the last chapter. These approaches are:

- Block Interleaving
- Convolutional Interleaving

In the Block Interleaving approach the sequence of data is first broken into a separate number of blocks. The order of the data in these blocks is permutated according to some equation or a predetermined function.

In the Convolutional Interleaving scheme the sequence is shuffled using a sliding window technique.

The type of interleaving is matched with the type of error control code being adopted.

4.1 Block Interleaving

The block diagram of either interleaver or deinterleaver can be seen from Figure. 4.1. This consists of two memory blocks and a permutation function lookup table. These two memory blocks function as complementary double buffer blocks, each block can store up to an entire block of data. The input data which is to be kept in the same block will be stored in the empty memory block. After storing this data block, the data will be obtained from the memory by the order of permutation specified by the block interleaver function or equation. The size of the memory block will depend on the system specification. The basic function of these complementary memory blocks is to store the input data of incoming data block into two different block. This is done to ensure that the input data

block which is being written into a particular memory block does not overwrite the data which has not been output by the system.

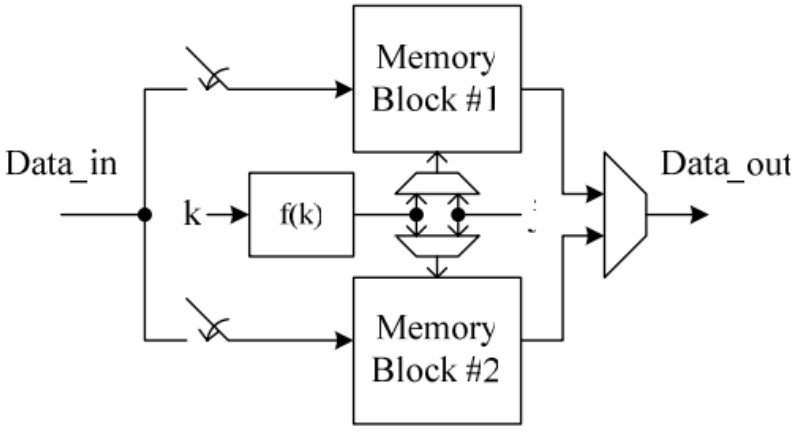

Figure 4.1: General Block Diagram of Block Interleaver [13]

The interleaving is as per IEEE standards is given by two permutation steps. Let the number of coded bits per block be represented by N_{cbps} From the

$$m = \left(\frac{N_{cbps}}{16}\right).k_{mod(16)} + \left\lfloor\frac{k}{16}\right\rfloor \tag{4.1}$$

The block of data is organized as $16 \times (N_{cbps}/16)$ matrix. The second step of permutation is defined by

$$j = s.floor\left(\frac{m}{s}\right) + (m + N_{cbps} - floor(16.\frac{m_k}{N_{cbps}}))_{mod(s)} \tag{4.2}$$

The reordering of the data is done in the transposed matrix for different values of ***s***, except those which are divisible by ***s***.

In the deinterleaver design the reverse operation of the two permutation steps in interleaver is carried out i.e. equation (4.2) followed by equation (4.1).

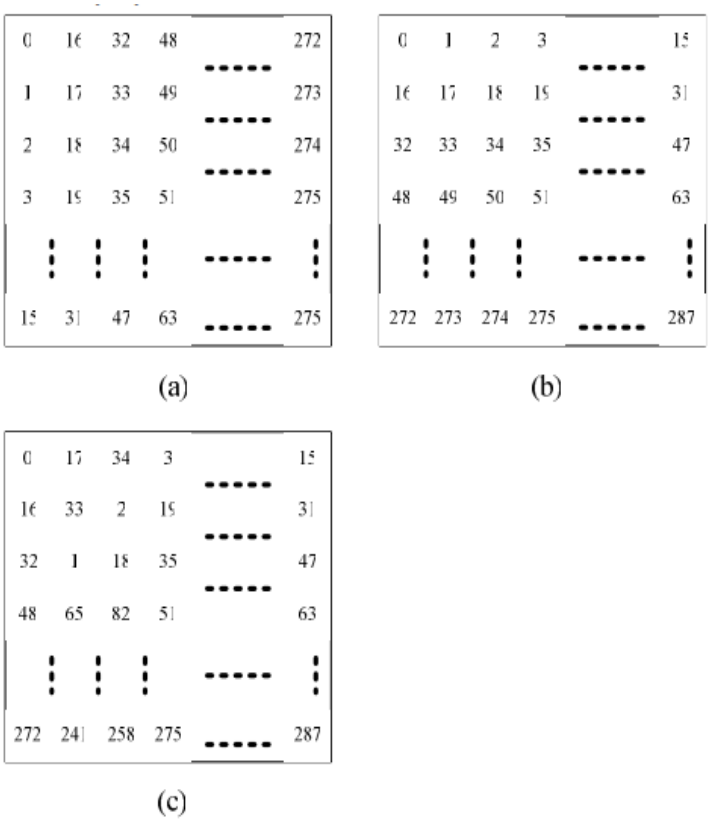

Figure 4.2: Data interleaving for 1 subchannel 64-QAM a) Original Data Matrix b) Data Matrix after First Permutation c) Final Matrix [13]

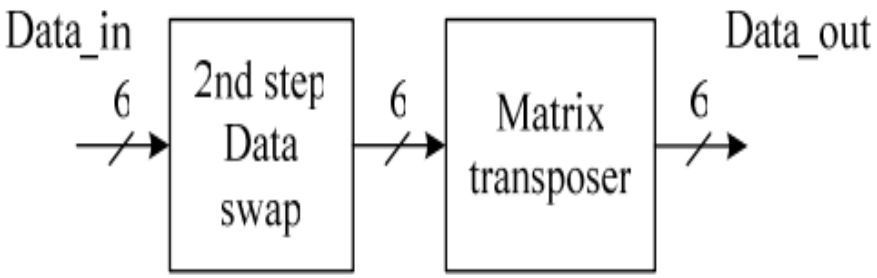

Figure 4.3: General Deinterleaver Block Diagram [13]

The general block diagram of the deinterleaver block is shown in Figure 4.3. This consists of data swap module which performs the second deinterleaving permutation that is transforming data matrix shown in Figure 4.2(c) to 4.2(b). After the swapping is done, the data is passed through a matrix transposer module which performs the operation of getting the original data matrix as depicted from Figure 4.2(b) to 4.2(a).

In the IEEE 802.16e standard, the number of input coded bits for different modulation schemes can be up to 6. So at every cycle 6 bits of deinterleaved data are formed. This involves up to six memory-read operations since all the six deinterleaved data which are produced at the output belong to the same output cycle enter the module at different cycles in such a manner that they may be stored in different locations. Therefore, the memory block utilized in the matrix transposer may require six read-write ports which can have a significant area overhead. Although faster memory operation clock can also maintain large memory bandwidth with smaller number of ports, the problem of multi-clock, critical path and power dissipation may arise in the design.

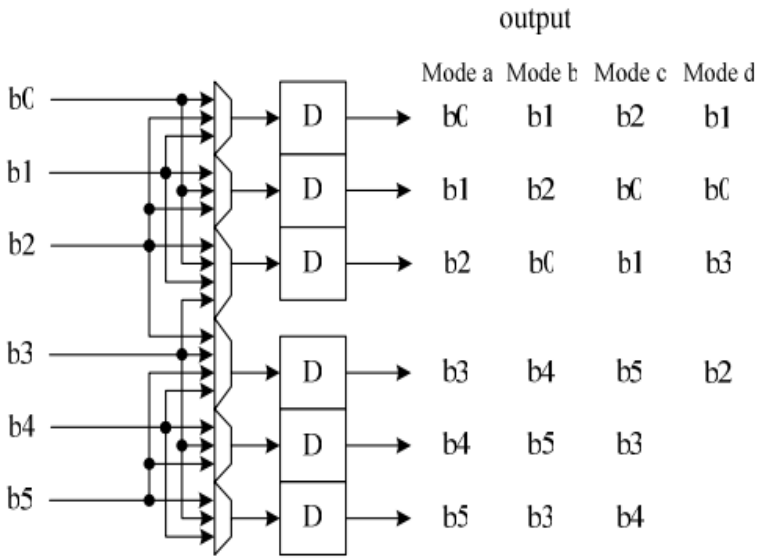

Figure 4.4: Data Swap Module [13]

In this work an efficient transposer matrix $M \times N$ structure is reported [13] based on multiple memory banks. If the number of input data at each cycle is equal to ***D***, and the number of memory banks is equal to ***L*** and if the memory bank is to be realized using less memory ports the data has to be allocated uniformly into multiple memory banks and the minimum number of ports will be equal to $\lceil L/D \rceil$. If the data is not distributed uniformly then ***D*** ports will be required for each memory bank.

For the transpose of $M \times N$ matrix a data allocation table is built by filling each entry with triplet symbol viz. *ind, bank, addr*. The first element *ind* represents the input

sequence order index of the corresponding data in the matrix. The second element *bank* denotes the bank number and the third element *addr* represents the address of the memory location where data will be stored. The bank number and address values are calculated as

$$bank = (ind + \left\lfloor \frac{c}{N_B} \right\rfloor)_{mod(D)} \tag{4.3}$$

$$addr = \left\lfloor \frac{ind}{D} \right\rfloor \tag{4.4}$$

$$N_B = \frac{D}{\gcd(M,D)} \tag{4.5}$$

The data allocation for 6X4 matrix is shown in the Table 4.1. In this allocation the value of ***D*** is equal to 4, so for each cycle 4 consecutive data are input and divided into the separate memory banks. Since the data of each row are divided into different banks they can be retrieved in parallel and the number of ports for each memory bank will be equal to 1.

(0,0,0)	(6,2,1)	(12,1,3)	(18,3,4)
(1,1,0)	(7,3,1)	(13,2,3)	(19,0,4)
(2,2,0)	(8,0,2)	(14,3,3)	(20,1,5)
(3,3,0)	(9,1,2)	(15,0,3)	(21,2,5)
(4,0,1)	(10,2,2)	(16,1,4)	(22,3,5)
(5,1,1)	(11,3,2)	(17,2,4)	(23,0,5)

Table 4.1: Allocation Table for 6X4 Matrix

The architecture of this transposer matrix is as shown in Figure 4.5. It consists of two memory blocks each containing four - 6 entry banks. The storage sequence of the input data in each memory bank is sequential such that the memory-write address can be procured by a simple incrementer module. Due to $\left\lfloor \frac{c}{N_B} \right\rfloor$ term in equation (4.3), the destination bank of each input data may vary such that the input has to be permuted at some time. For the 6x4 transposition matrix, the permutation pattern will change every two columns. A permutation block is provided in order to get the requisite permutation pattern. After the block of input data has been stored in one of the memory blocks, the

transposed output can then be taken through the output while the next block of inputs will be saved into the other block concurrently.

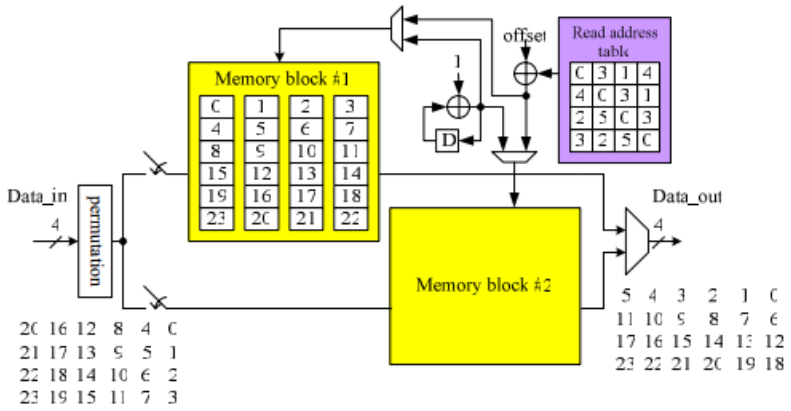

Figure 4.5: Matrix Transpose Architecture [13]

The generation of the output transposed data sequence is a complex task as the location of each output data on the memory is not regular. So an online generation circuit is required and it is highly complex. So an on-chip lookup table is used to store the address order of the output data sequence.

Another technique which is used to reduce the memory banks without increasing the port number is to merge multiple data originally saved in separate banks into the same memory location. The merging pattern is shown in Table 4.2. This will cause the number of memory banks to be reduced by half but this a tradeoff of putting an extra output buffer to reshuffle the data to designated output cycles. The additional output buffer is a small transposer that requires a temporary data storage space of size N.

The architecture is illustrated in Figure 4.6.

4.2 2-D Realization of WiMAX Channel Interleaver

In order to optimize the hardware of the interleaver block the alternate method is to take the two steps of permutation as a single step and find the correlation between input and output. So the realization of 1-dimensional equation into 2-dimensional equation is done in order to make the hardware more efficient and the same set of equations can be used by just swapping the order of read and write of data into the memory.

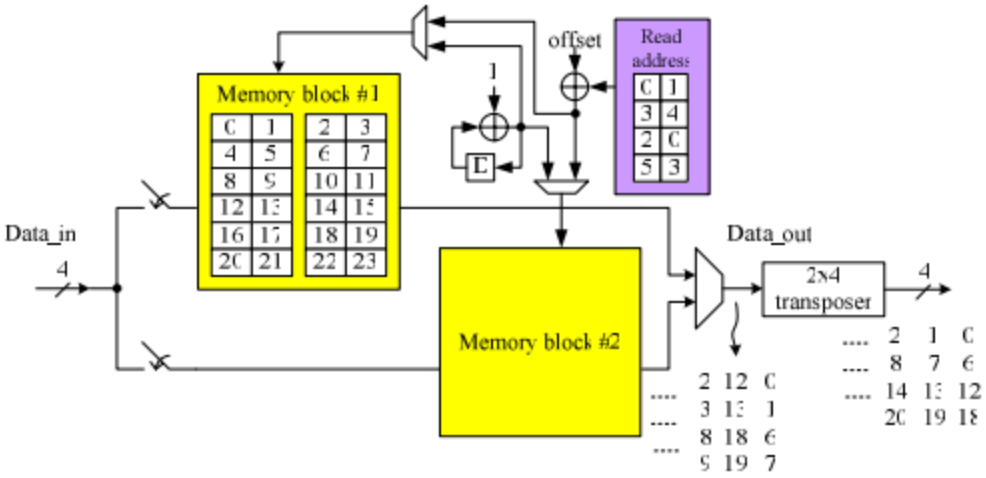

Figure 4.6: 6X4 Output Buffered Matrix Transpose Architecture [13]

For BPSK/QPSK this equations can be reduced to

$$k_n = d.\beta_n + \gamma_n \tag{4.6}$$

where β_n and γ_n is given by

$$\beta_n = n - \frac{N}{d}.\left\lfloor \frac{d.n}{N} \right\rfloor \tag{4.7}$$

$$\gamma_n = \left\lfloor \frac{d.n}{N} \right\rfloor \tag{4.8}$$

and $N = N_{cbps}$

If this can be rewritten in two dimensions ***i*** and ***j*** which is a 2-dimensional array and ***j*** increments when value of ***i*** expires the range of ***i*** and ***j*** can be defined as follows:

$i = 0,1 \ldots \left(\frac{N}{d} - 1\right)$, which satisfies with n when $i = (n \,\%\, \frac{N}{d})$ (4.9)

$j = 0,1 \ldots (d-1)$, which satisfies with n when $j = \left\lfloor \frac{d.n}{N} \right\rfloor$ (4.10)

Therefore $k_n = k_{(i,j)} = d.i + j$ (4.11)

Similarly for the 16-QAM case, the 2-D interleaver equation can be written as

$$k_n = k_{(i,j)} = d.r_{(i,j)} + j \tag{4.12}$$

Where

$$\beta_n = r_{(i,j)} = [1 - (j\%2)].i + [(j\%2)] \times \{(i+1)[1 - (i\%2)] + (i-1)(i\%2))\} \tag{4.13}$$

Where the parameter $\boldsymbol{r_{(i,j)}}$ gives an intra-row permutation pattern series for selective columns, in a manner that the permutation is applied to all the alternate columns $\boldsymbol{(2y+1)^{th}}$ and no permutation is applied to $\boldsymbol{2y^{th}}$ columns where $\boldsymbol{y = 0,1,\ldots.d/2}$. Similarly for the $\boldsymbol{R}$ rows, the inter row permutation is applied as $\boldsymbol{i+1}$ to $\boldsymbol{2i^{th}}$ and $\boldsymbol{i-1}$ to $\boldsymbol{2(i+1)^{th}}$ row respectively. This is modeled into hardware as shown in the Figure 4.7.

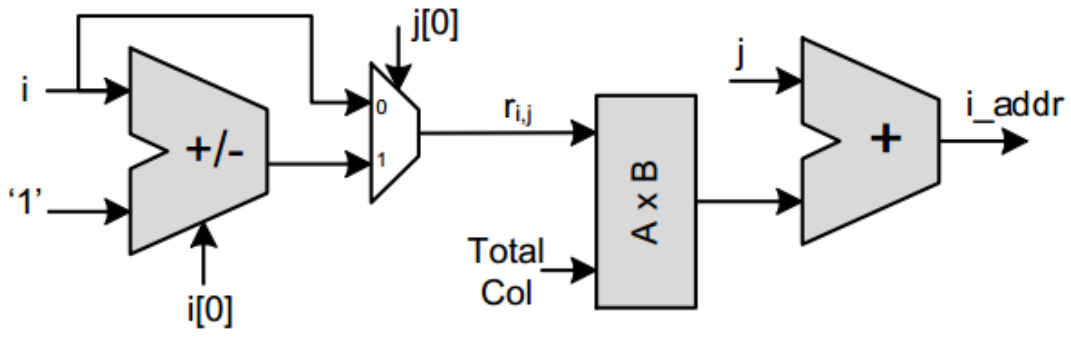

Figure 4.7: Hardware Realization of 16-QAM in WiMAX [14]

In case of 64-QAM the restructuring for the expression of $\boldsymbol{\beta_n}$ is done and a complex pattern is obtained given by equation 4.14. Although equation (4.15) is very long and complicated, but eventually, a hardware efficient solution is obtained as shown in figure 4.8.

$$\beta_n = r_{(i,j)} = \left[(1-j') + \frac{j'(j'-1)}{2}\right].i + \left\{[j' - (j'-1)]\left\{(i-2)\left[(1-i') + \frac{i'(i'-1)}{2}\right] + (i+1)\left[i' - \frac{i'1-1}{2}\right]\right\}\right\} + \left\{\frac{j'(j'-1)}{2}\{(i+2)[i' - i'(i'-1)] + (i-1)[(1-i') + i'(i'-1)]\}\right\} \tag{4.15}$$

Where $i' = (i + 1)\%3$ and $j' = j\%3$ (4.16)

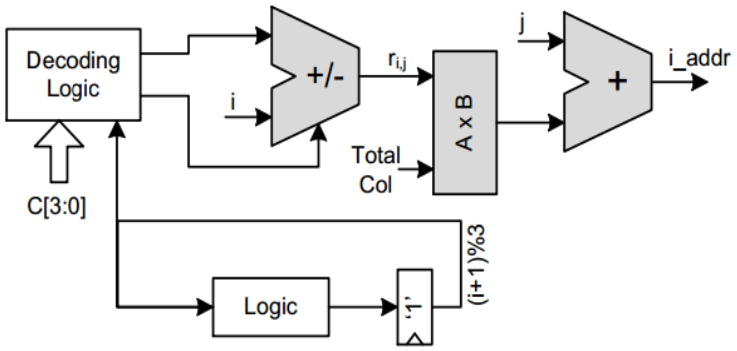

Figure 4.8: Hardware Realization of 64-QAM [14]

The generic interleaver structure is depicted in Figure 4.9. In addition to hardware necessary for address generation a small control logic is also neededfor row and column counter and synchronization with external world. The use of multiplier can be evaded but is kept for the generality of design. The multiplier provides flexibility to design of the interleaver for any number of columns.

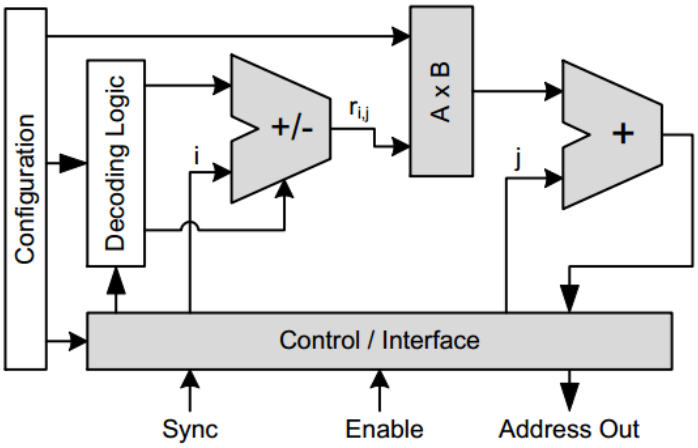

Figure 4.9: Hardware for Address Generation [14]

4.3 Address Generation for Interleaver Using Finite State Machine

The first and second level of permutation is given by

$$m_k = \left(\frac{N_{cbps}}{d}\right)(k\%d) + \left\lfloor\frac{k}{d}\right\rfloor \tag{4.17}$$

$$j_k = s.\left\lfloor\frac{m_k}{s}\right\rfloor + \left(m_k + N_{cbps} - \left\lfloor\frac{d.m_k}{N_{cbps}}\right\rfloor\right)\%s \tag{4.18}$$

The values of j_k are evaluated from the equations (4.17) and (4.18) and the first 32-samples are enlisted in Table 4.2. From the table it is found that the subsequent values of j_k are not uniformly spaced in all cases. It follows a fixed type of pattern irrespective of coding rate for different modulation scheme. This is evaluated in Table 4.3. The architecture for the address generator is illustrated in Figure 4.8. The first level MUXs is used to implement the unequal increments required in 16-QAM and 64-QAM. The four-interleaver depths of 16-QAM are implemented by first four MUXs from the top. The select lines of these four MUXs are then strapped together and are connected to a T-flip-flop QAM16_SEL. The last four MUXs are used for 64-QAM modulation. The select lines are driven by a MOD-3 counter - QAM64_SEL. The second level MUXs then pick up one input depending on the values of ID. The topmost MUX in level 2 is used to implement the eight interleaver depths for the QPSK modulation scheme. The second and third MUXs in level 2 are used for 16-QAM and 64-QAM respectively. The outputs derived from level 2 MUXs are then connected to the subsequent section by level 3 MUX depending on MOD_TYP value. The 7-bit output which is output in level-3 MUX is used as one input for the 10-bit adder circuit when the zero padding is done. The other input to the adder circuit is the accumulator which is used to store the previous address. The preset logic is a finite state machine logic whose purpose is to generate the correct beginning addresses for all subsequent iterations.

N_{cbps} =96 bits, ½ code rate, QPSK	0	6	12	18	24	30	36	42
	48	54	60	66	72	78	84	90
	1	7	13	19	25	31	37	43
	49	55	61	67	73	79	85	91
N_{cbps} =288 bits, ¾ code rate, 16-QAM	0	19	36	55	72	91	108	127
	144	163	180	199	216	235	252	271
	1	18	37	54	73	90	109	126
	145	162	181	198	217	234	253	270
N_{cbps} =384 bits, code rate, 64-QAM	0	26	49	72	98	121	144	170
	193	216	242	265	288	314	337	360
	1	24	50	73	96	122	145	168
	194	217	240	266	289	312	338	361

Table 4.2: Permutation Address of 3-code rate and modulation schemes

4.3.1 Preset Logic

The preset logic is used to keep a track of the address generated depending on the MOD_TYPE. The CLR = 1 is used to drive the FSM to its initial state and depending on the input it transitions into three possible states (S_{MT0}, S_{MT1}, S_{MT2}). The FSM then moves to next level states (S_{ID0} to S_{ID7} from the state S_{MT0}, S_{ID0} to S_{ID3} from the state S_{MT1}or S_{MT2}). The various states in this level depict one of the interleaver depths. It makes a transition to next level of states depending on the value in the accumulator. When the terminal value is reached, it makes transition where the accumulator is loaded with an initial value (e.g. Preset=1) of the next iteration. This continues till all the addresses are generated for given ID and MOD_TYP.

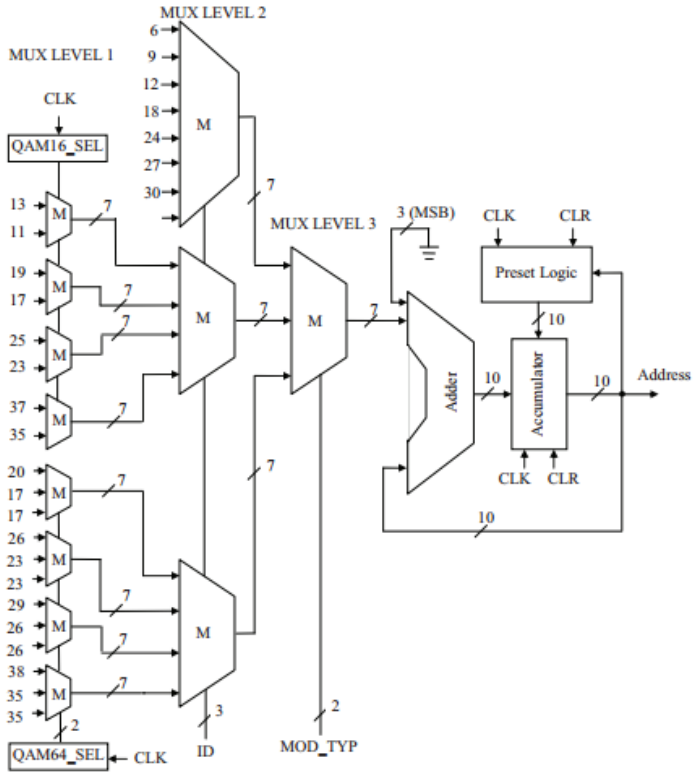

Figure 4.10: Address Generation Scheme

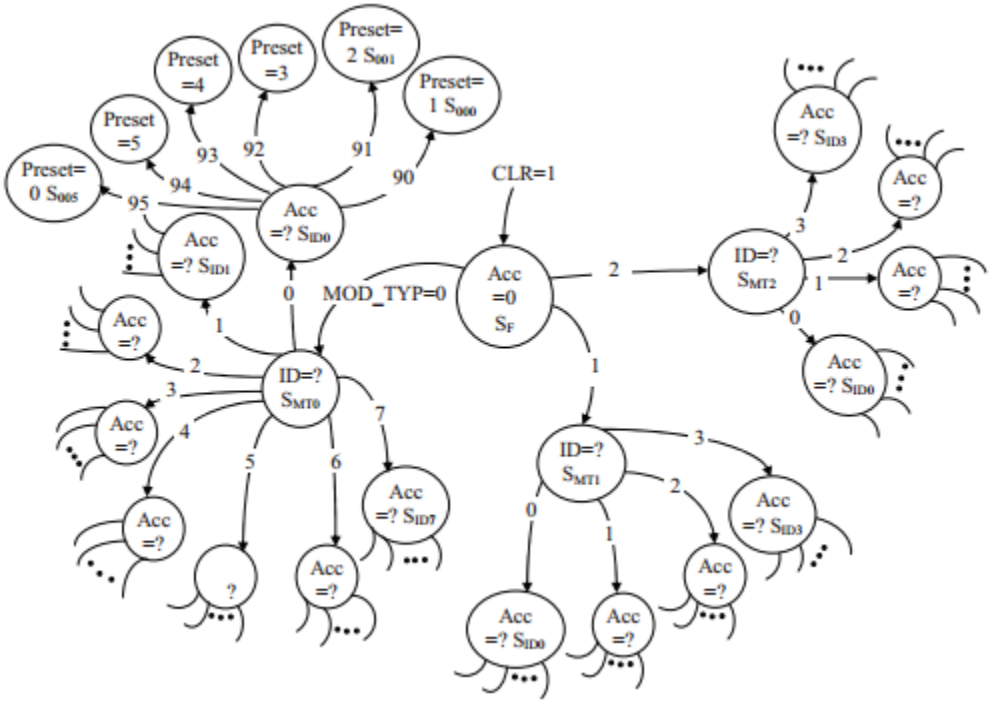

Figure 4.11: FSM for Address Generation

4.4 Address Generator for WiMAX Deinterleaver

The deinterleaver performs the operation of retrieving the coded bits which are mapped to non adjacent subcarriers by performing the opposite function of the interleaving structure. The operation of a deinterleaver is defined by the following equations.

$$m_j = s.\left\lfloor \frac{j}{s} \right\rfloor + \left(j + \left\lfloor \frac{d.m_k}{N_{cbps}} \right\rfloor \right) \% s \tag{4.19}$$

$$k_j = d.m_j - (N_{cbps} - 1)\left\lfloor \frac{d.m_j}{N_{cbps}} \right\rfloor \tag{4.20}$$

The different interleaver depths for the WiMAX IEEE 802.16e standard are shown in Table 4.3. From this table it is surmised that the two equations are to be evaluated for given interleaver depths and the different code rates associated with them.

Mod.Scheme	QPSK (s=1)		16-QAM(s=2)		64-QAM(s=3)		
Code Rate	½	¾	½	¾	½	2/3	¾
Interleaver Depth N_{cbps} in bits	96	144	192	288	288	384	432
	192	288	384	576	576	-	-
	288	432	576	-	-	-	-
	384	576	-	-	-	-	-
	480	-	-	-	-	-	-
	576	-	-	-	-	-	-

Table 4.3: Interleaver Depths for different modulation schemes for IEEE 802.16e

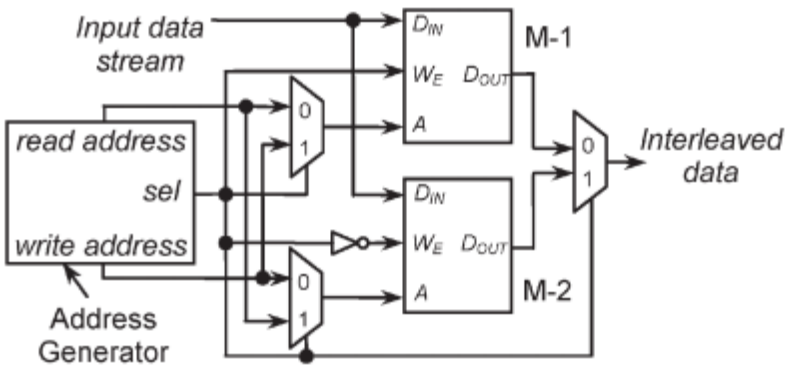

Figure 4.12: Interleaver/Deinterleaver Structure [16]

Due to the presence of the floor and modulo functions the direct implementation of these equations on the hardware is not possible. So at first the equations are evaluated and the deinterleaver addresses generated for the first four rows and five columns are tabulated and shown in Table 4.4.

Here the value of ***d*** is chosen to be 16.So the number of rows is fixed and equal to ***d***.

The number of columns is given by$\frac{N_{cbps}}{d}$.

From the table it is found out that a correlation can be established between the address generated and the values of deinterleaver addresses can be calculated by a simple multiplication and an addition operation if the values of N_{cbps} and d are known to the processor. The correlation between deinterleaver addresses generated and the value of d is shown in Table 4.5.

N_{cbps},Code Rate,Modulation Type	Deinterleaver Address				
N_{cbps} = 96-bits,1/2 code rate,QPSK	0	16	32	48	64
	1	17	33	49	65
	2	18	34	50	66
	3	19	35	51	67
N_{cbps} = 192-bits,1/2 code rate,16-QAM	0	16	32	48	64
	17	1	49	33	81
	2	18	34	50	66
	19	3	51	35	83
N_{cbps} = 96-bits,1/2 code rate,64-QAM	0	16	32	48	64
	17	33	1	65	81
	34	2	18	82	50
	3	19	35	51	67

Table 4.4: Deinterleaver Addresses for different modulation schemes

4.4.1 Algorithm for QPSK

From the tabulated values shown in Table 4.4 and the correlation established from Table 4.5, it is clear that the deinterleaver addresses for the QPSK modulation scheme can be described by the following equation.

$$k_{n,QPSK} = d \times i + j \; for \; \forall j \; and \; \forall i \tag{4.21}$$

This can be illustrated by flow graph shown in Figure 4.13.

N_{cbps},Code Rate,Modulation Type	Deinterleaver Address				
N_{cbps} = 96-bits,1/2 code rate,QPSK	*d*.0+0 =0	*d*.1+0 = 16	*d*.2+0 = 32	*d*.3+0 =48	*d*.4+0 =64
	d.0+1 = 1	*d*.1+1 = 17	*d*.2+1 =33	*d*.3+1 =49	*d*.4+1 =65
	d.0+2 = 2	*d*.1+2 = 18	*d*.2+2 =34	*d*.3+2 =50	*d*.4+2 =66
	d.0+3 = 3	*d*.1+2 = 19	*d*.2+3 =35	*d*.3+2 =51	*d*.4+2 =67
N_{cbps} = 192-bits,1/2 code rate,16-QAM	*d*.0+0 =0	*d*.1+0 =16	*d*.2+0 =32	*d*.3+0 =48	*d*.4+0 =64
	d.1+1 =17	*d*.0+1 =1	*d*.3+1 =49	*d*.2+1 =33	*d*.5+1 =81
	d.0+2 =2	*d*.1+2 =18	*d*.2+2 =34	*d*.3+2 =50	*d*.4+2 =66
	d.1+3 =19	*d*.0+3 =3	*d*.3+3 =51	*d*.2+3 =35	*d*.5+3 =83
N_{cbps} = 96-bits,1/2 code rate,64-QAM	*d*.0+0 =0	*d*.1+0 =16	*d*.2+0 =32	*d*.3+0 =48	*d*.4+0 =64
	d.1+1 =17	*d*.2+1 =33	*d*.0+1 =1	*d*.4+1 =65	*d*.5+1 =81
	d.2+2 =34	*d*.0+2 =2	*d*.1+2 =18	*d*.5+2 =82	*d*.3+2 =50
	d.0+3 =3	*d*.1+3 =19	*d*.2+3 =35	*d*.3+3 =51	*d*.4+3 =67

Table 4.5: Correlation between Deinterleaver Addresses

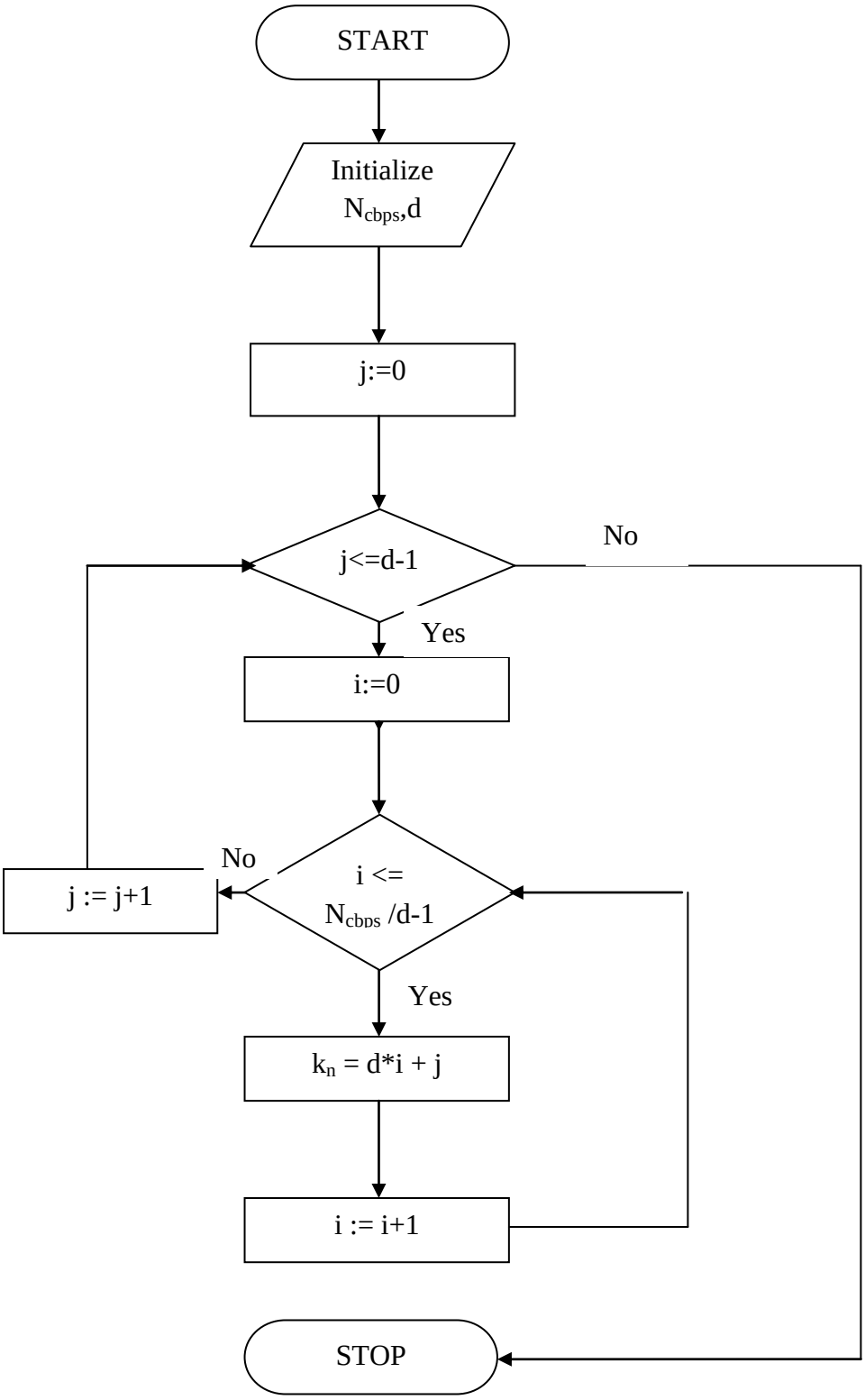

Figure 4.13: Flow Graph for QPSK

4.4.2 Algorithm for 16-QAM

From the tabulated values shown in Table 4.4 and the correlation established from Table 4.5, it is clear that the deinterleaver addresses for the 16-QAM modulation scheme can be described by the following equation.

$$k_{n,QPSK} = d \times i + j \; for \; j\%2 = 0 \; and \; for \; \forall i \tag{4.22}$$

$$k_{n,QPSK} = d \times (i+1) + j \; for \; j\%2 = 1 \; and \; for \; i\%2 = 0 \tag{4.23}$$

$$k_{n,QPSK} = d \times (i-1) + j \; for \; j\%2 = 1 \; and \; for \; i\%2 = 1 \tag{4.24}$$

This can be illustrated by flow graph shown in Figure 4.14.

4.4.3 Algorithm for 64-QAM

From the tabulated values shown in Table 4.4 and the correlation established from Table 4.5, it is clear that the deinterleaver addresses for the 64-QAM modulation scheme can be described by the following equation.

$$k_{n,QPSK} = d \times i + j \; for \; j\%3 = 0 \; and \; for \; \forall i \tag{4.25}$$

$$k_{n,QPSK} = d \times (i-2) + j \; for \; j\%3 = 1 \; and \; for \; i\%3 = 2 \tag{4.26}$$

$$k_{n,QPSK} = d \times (i+1) + j \; for \; j\%3 = 1 \; and \; for \; i\%3 \neq 2 \tag{4.27}$$

$$k_{n,QPSK} = d \times (i+2) + j \; for \; j\%3 = 2 \; and \; for \; i\%3 = 0 \tag{4.28}$$

$$k_{n,QPSK} = d \times (i-1) + j \; for \; j\%3 = 2 \; and \; for \; i\%3 \neq 0 \tag{4.29}$$

This can be illustrated by flow graph shown in Figure 4.15.

From the above equations it can be deduced that the floor and modulo function can be rewritten in multiplication and addition with modulo-2 and 3 operations.

These functions are first tested in MATLAB (Appendix A) and results are compared with the previous MATLAB algorithm formed from the two deinterleaver equations.

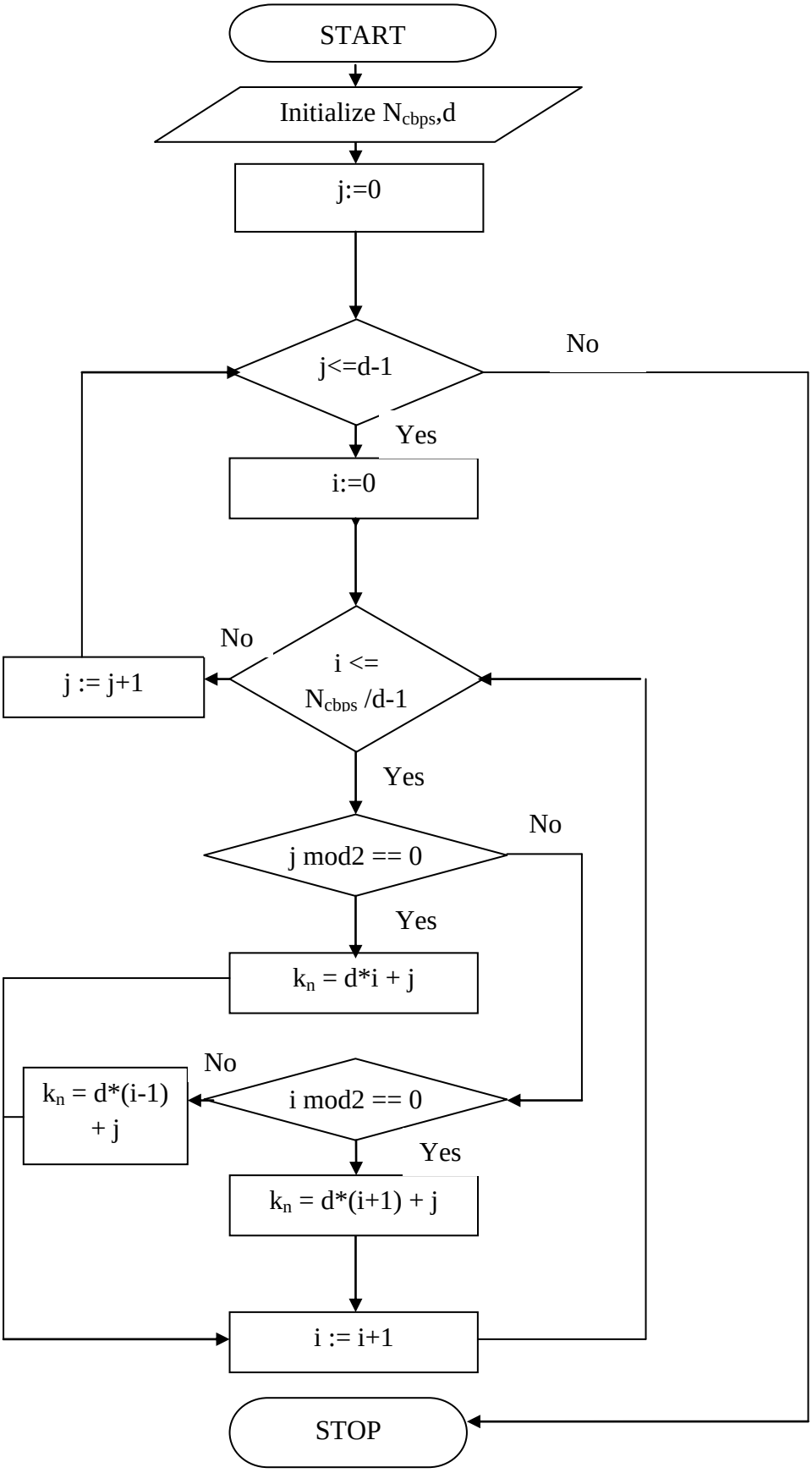

Figure 4.14: Flow Graph for 16-QAM

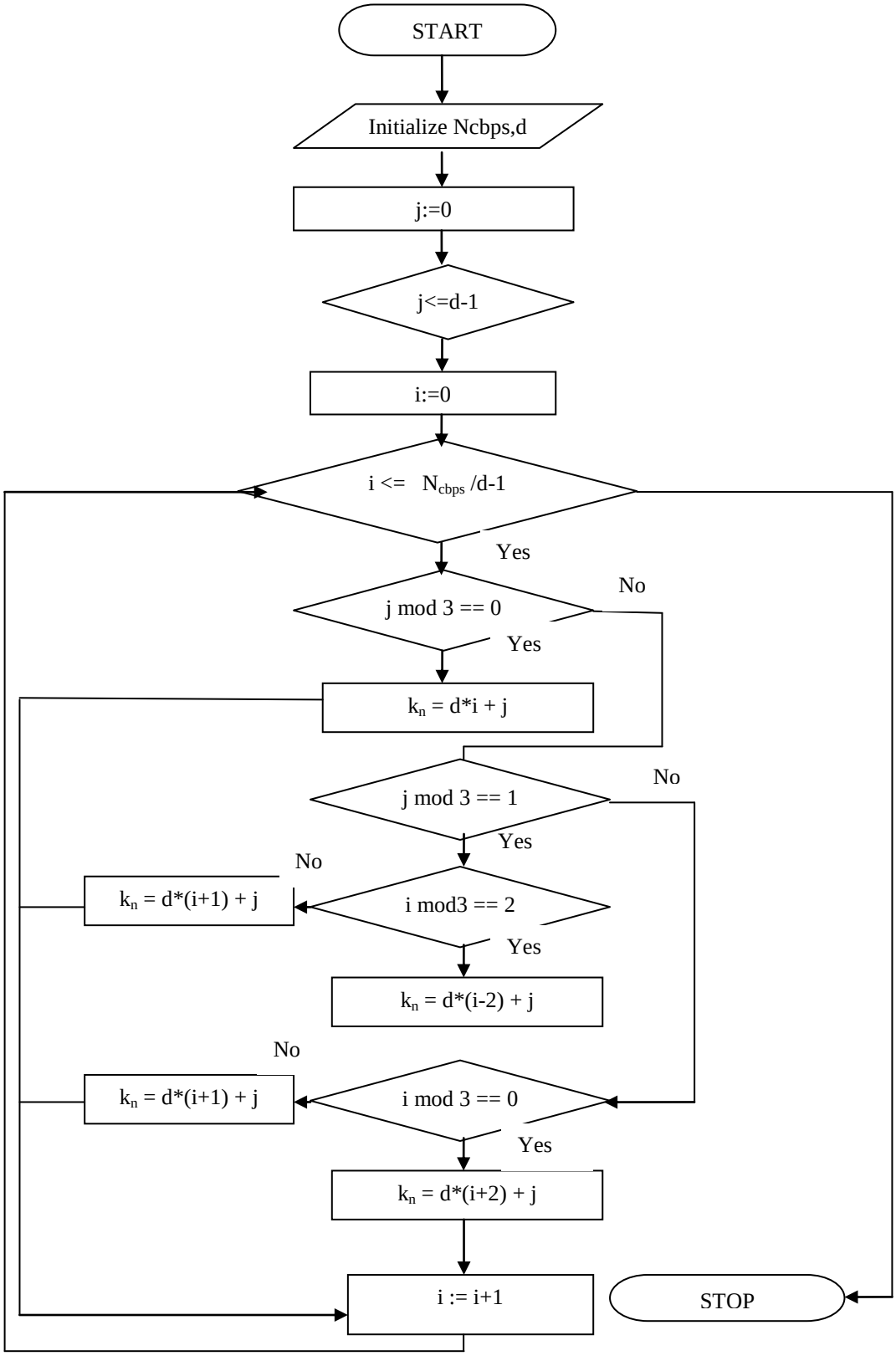

Figure 4.15: Flow Graph for 64-QAM

CHAPTER 5

VERILOG DESIGN AND SIMULATION RESULTS

The algorithms explained in the earlier chapter are modelled into hardware using Xilinx Spartan 3 Series (Device XC3S400a-4ftg256) [19].

5.1 Design of Address Generator for QPSK

In order to implement address generation for QPSK modulation scheme, two counters are used which are used for incrementing the rows and the columns in the circuit. The input to multiplexer is M0 which is used for comparator circuit in the column counter. The column counter C0 consists of a comparator and a counter where the comparator checks the counter value being generated with the reference value from M0 and generates a reset when both the values are equal. The output from the column counter is fed to multiplier M1 and simultaneously to row counter R0.

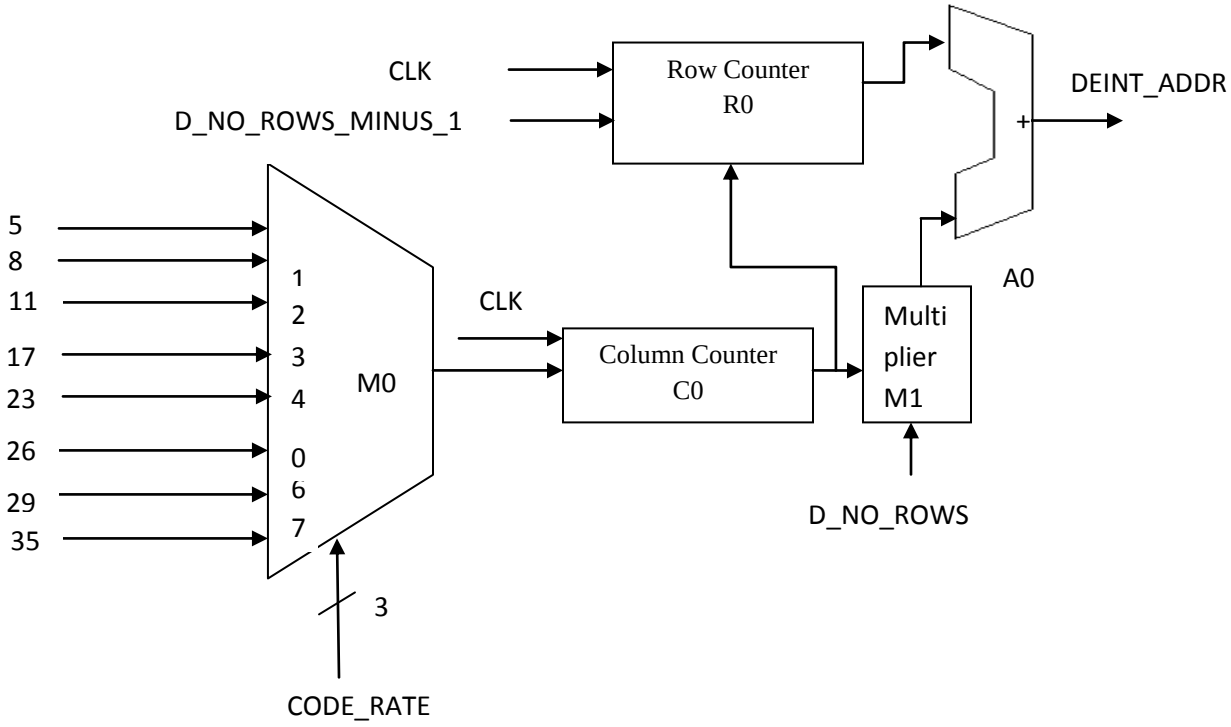

Figure 5.1: Hardware Design of QPSK Block

The row counter consists of a counter with a comparator value from the output of column counter. The value of row counter is incremented whenever the column counter is reset.

The output of C0 is given to an multiplier unit (M1) and is then added up with the row counter (A1) to give the deinterleaver address for QPSK. This is shown in Figure 5.1.

5.2 Simulation Results of Address Generator for QPSK

The synthesis of the above model is done using Verilog HDL (Appendix A.2) and the RTL schematic shown in Figure 5.2 and Figure 5.4. The description of the input and output is given below.

CODE_RATE [2:0] - 3 bit code rate input for ***Ncbps***

D_NO_ROWS [9:0] – 10 bit input for number of rows (***d***)

D_NO_ROWS_MINUS_1 [9:0] - 10 bit input for ***d-1***

CLK- Input Clock

G_RESET- Global Reset (Active High)

DEINT_ADDR- 12 bit Output Address (***kn***)

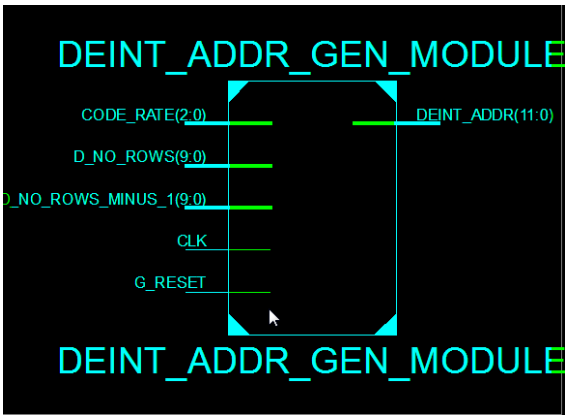

Figure 5.2: RTL Schematic of QPSK Block Address Generator

Minimum period: 9.641ns (Maximum Frequency: 103.724MHz)

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	78	3584	2%
Number of Slice Flip Flops	20	7168	0%
Number of 4 input LUTs	62	7168	0%
Number of bonded IOBs	27	195	13%
Number of GCLKs	1	24	4%

Figure 5.3: Device Utilization Summary for QPSK Block Address Generator

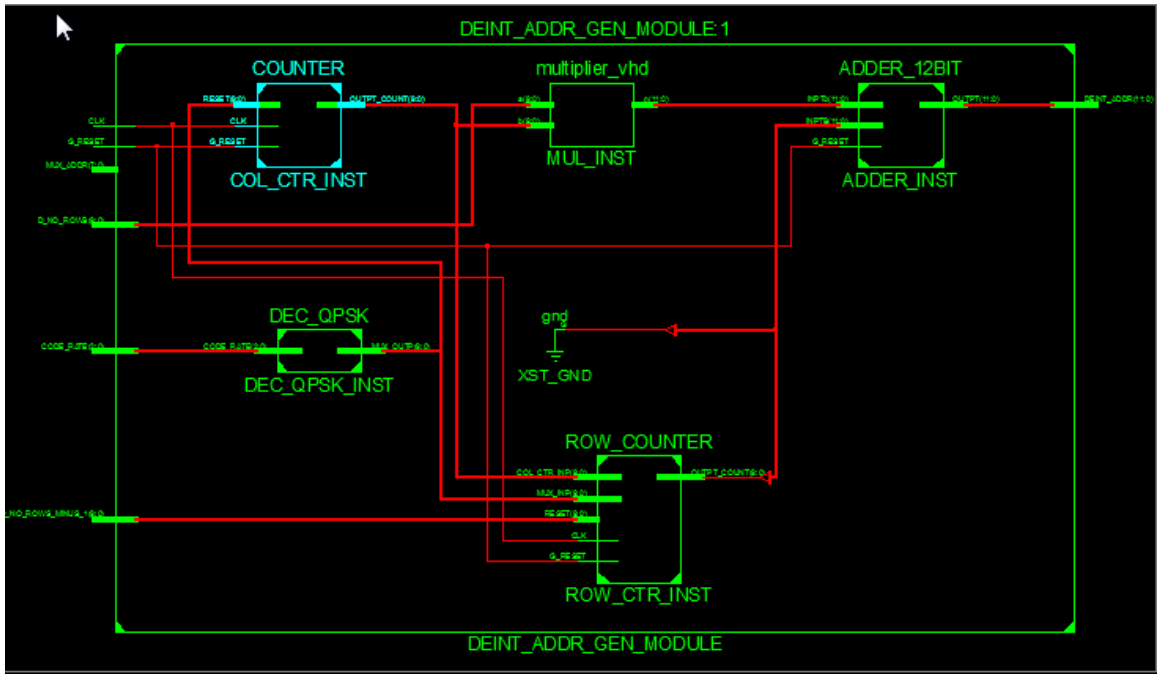

Figure 5.4: Detailed RTL Schematic of QPSK Block Address Generator

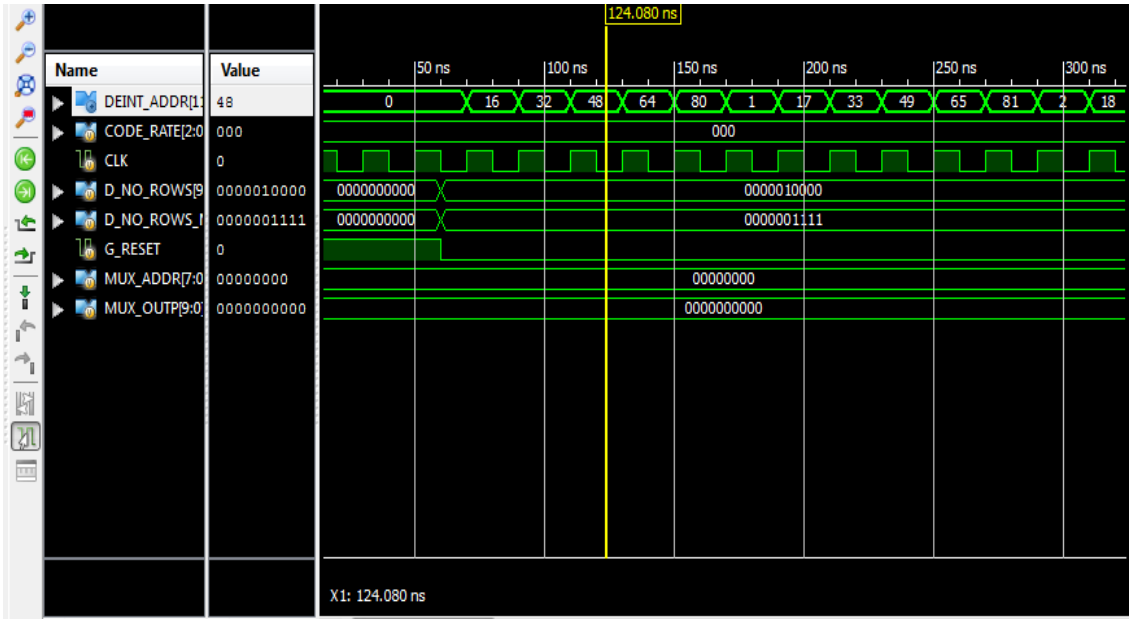

Figure 5.5: Simulation of QPSK Block Address Generator

5.3 Design of Address Generator for 16-QAM

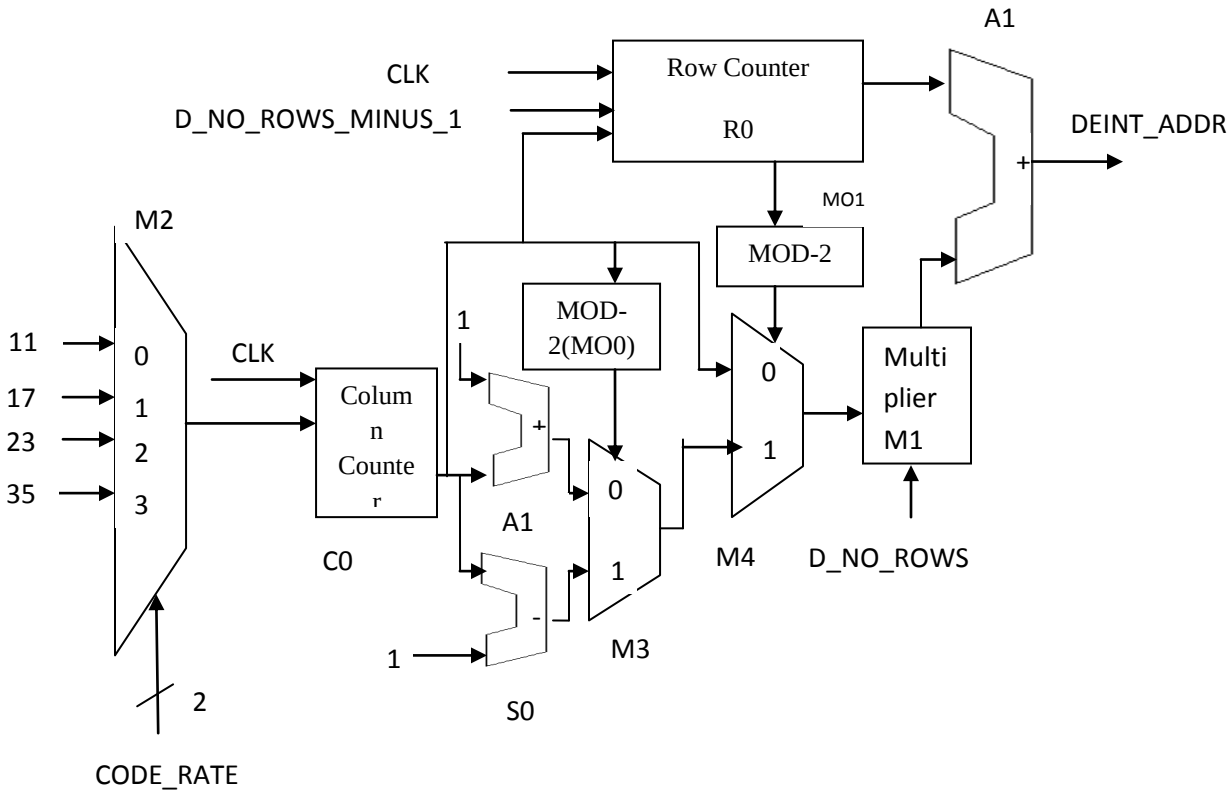

Figure 5.6: Hardware Design of 16-QAM Block Address Generator

For the hardware implementation for 16-QAM, modulus 2 block is needed. For the implementation of modulus 2 block in Verilog, the least significant bit of the input is considered and if it is equal to 1, the remainder is 1 else the remainder is 0.

The hardware modeling is done using M2 as multiplexer with CODE_RATE as input. The input is then fed to C0 which gives the output to adder A1 and subtractor S0. The output of the two blocks is then used by a multiplexer M3 which gives an output depending on the validity of equation (4.23) or (4.24). A multiplexer M4 is used to select between the output of M3 and output of C0 (if equation 4.22 is valid). The output is then fed to multiplier M1 and subsequently to adder A1.

5.4 Simulation Results of Address Generator for 16-QAM

The synthesis of the above model is done using Verilog HDL (Appendix A.2) and the RTL schematic shown in Figure 5.7 and Figure 5.8. The device utilization summary is given in Figure 5.9.

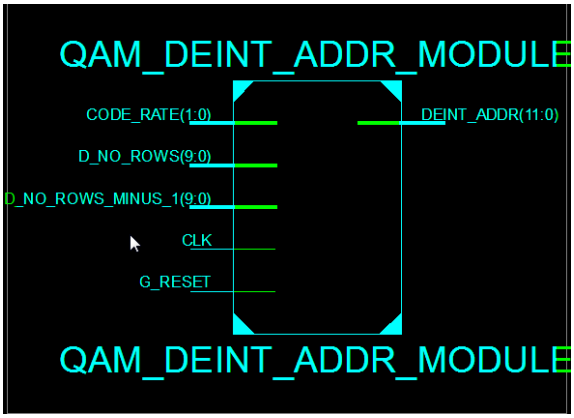

Figure 5.7: RTL Schematic of 16-QAM Block Address Generator

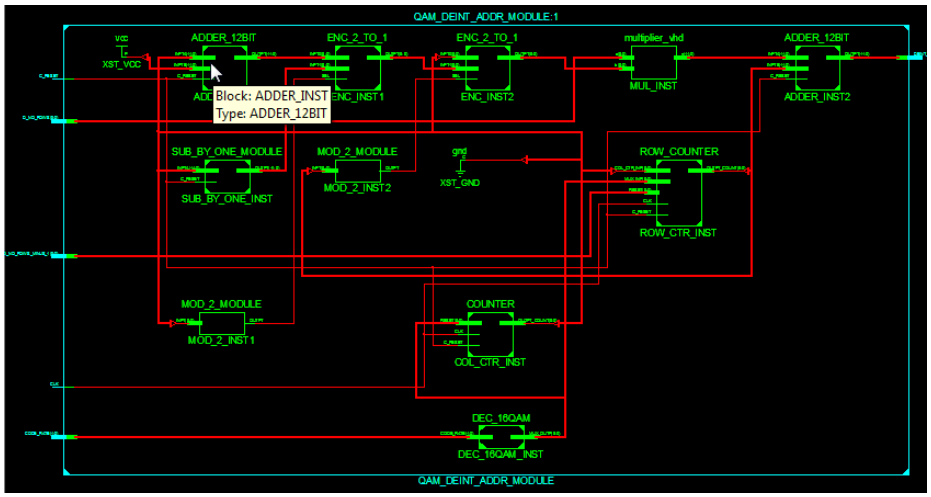

Figure 5.8: Detailed RTL Schematic of 16-QAM Block Address Generator

Minimum period: 6.936ns (Maximum Frequency: 144.175MHz)

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	95	3584	2%
Number of Slice Flip Flops	20	7168	0%
Number of 4 input LUTs	83	7168	1%
Number of bonded IOBs	26	195	13%
Number of GCLKs	1	24	4%

Figure 5.9: Device Utilization of 16-QAM Block Address Generator

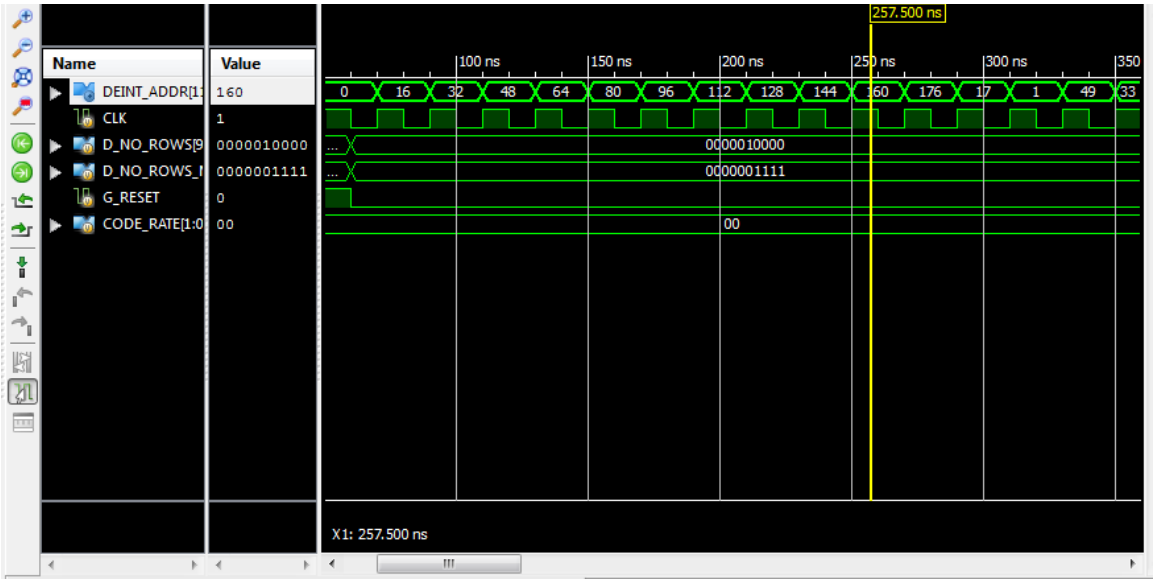

Figure 5.10: Simulation Result of 16-QAM Block Address Generator

5.5 Proposed Design of Address Generator for 64-QAM

For the hardware realization of address generator for 64-QAM it is clear from equation 4.25 to 4.29 that a modulus 3 block is required for selecting the output from the given set of equations (4.25-4.29). A modulus -3 operation cannot be directly realized in Verilog so generally a lookup table is used which is generally a ROM circuitry[16]. In order to decrease the access time required for lookup circuitry a modulus -3 technique using a set of adders is proposed in the subsequent section.

5.5.1 Modulus -3 Operation

Consider a 4-bit binary number which is represented as $b_3 b_2 b_1 b_0$ where $b_3\ and\ b_0$ are MSB and LSB respectively. When the number is converted to decimal form it is given by

$$d_3 d_2 d_1 d_0 = b_3.2^3 + b_2.2^2 + b_1.2^1 + b_0.2^0$$

Where the decimal number is represented as $d_3 d_2 d_1 d_0$ where $d_3\ and\ d_0$ are MSB and LSB respectively.

For the modulus operation in decimal representation the individual digits are added and the sum is divided by 3 and the number is divisible if the sum is divisible by 3.So applying the same operation to the set of binary numbers$b_3 b_2 b_1 b_0$, where bits are divided into group of 2 starting from LSB and then added using 2-bit full adder as shown in the figure 5.11. The sum bits $c_0 s_1 s_0$ is fed to 1-bit full adder and the result is compared with lookup table given in Table 5.1. Similarly for n- digits the modulus 3 can be calculated by considering similar cascade structure.

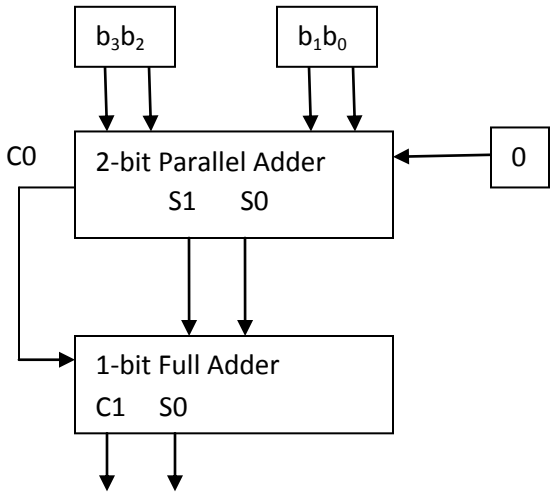

Figure 5.11: Proposed Modulus 3 Structure using Binary Adders

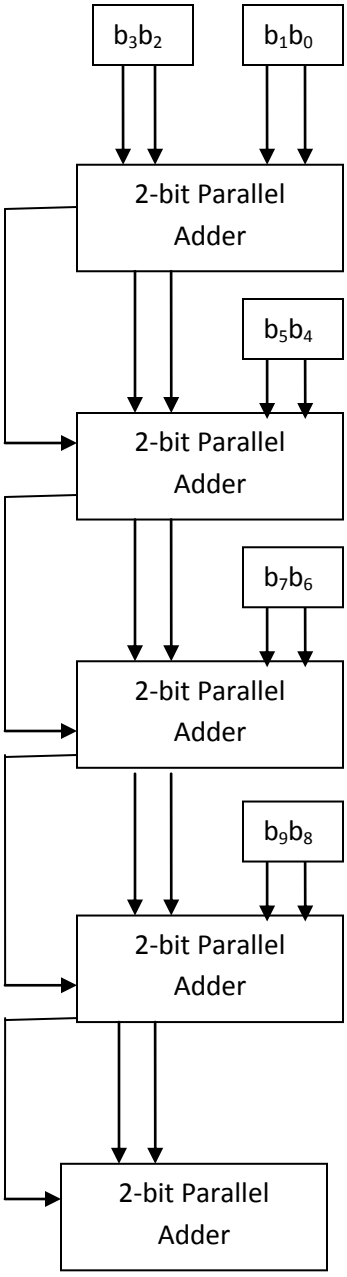

Figure 5.12: Proposed Modulus 3 Structure For 64-QAM

Thus, by using a modulus 3 module as shown in Figure 5.12 a hardware realization of 64-QAM module is obtained. This is illustrated in Figure 5.13.

Sum	Remainder
00	0
01	1
10	2
11	0

Table 5.1: Modulus 3 Lookup Table

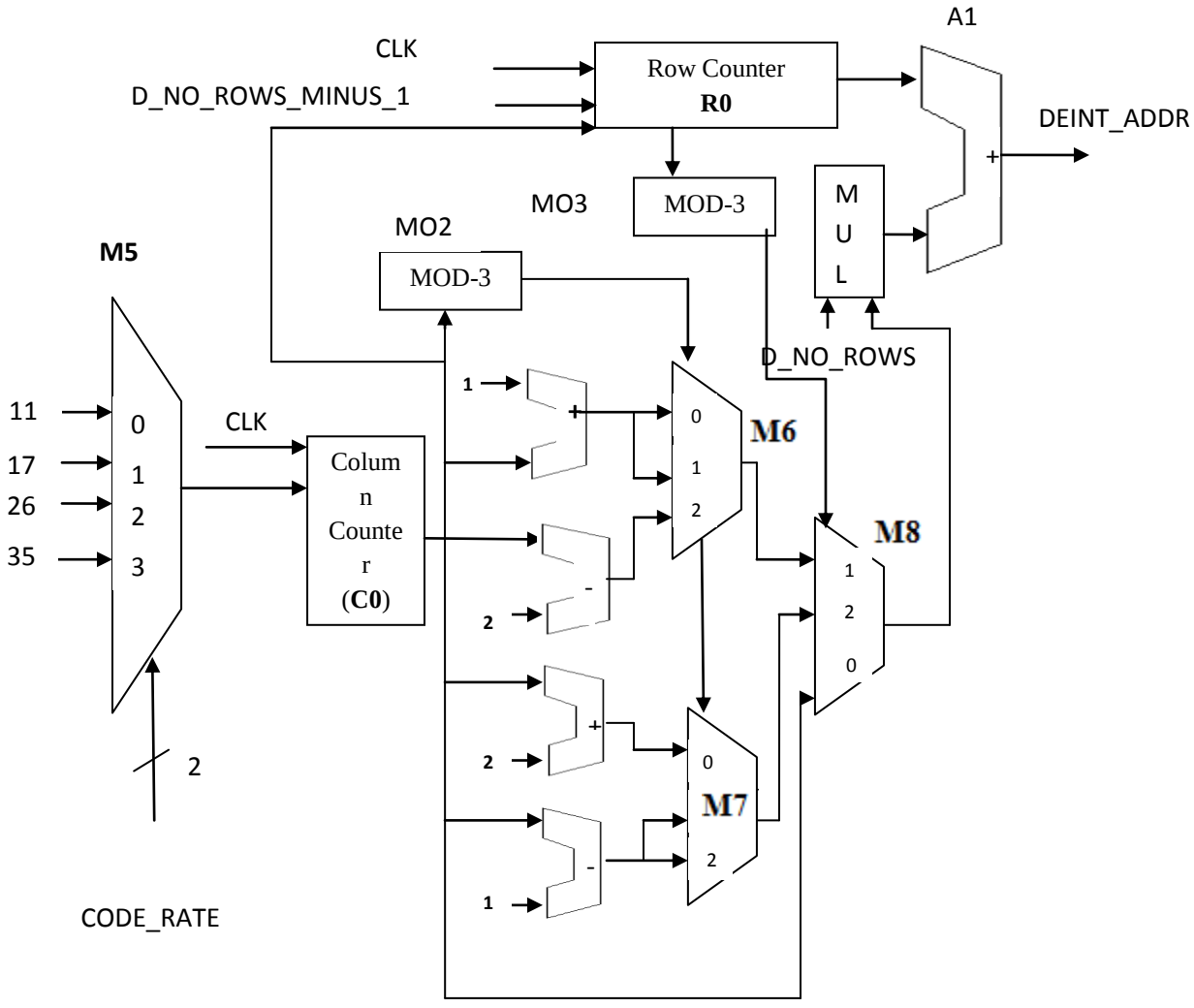

Figure 5.13: Design of 64-QAM Block Address Generator

5.6 Simulation Results of Address Generator for 64-QAM

The synthesis of the above model is done using Verilog HDL (Appendix A.2) and the RTL schematic shown in Figure 5.14 and Figure 5.15. The device utilization summary is given in Figure 5.16.

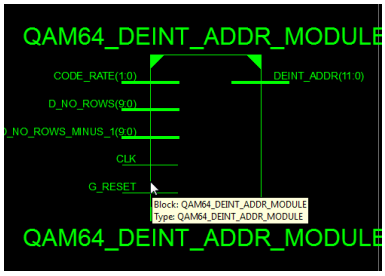

Figure 5.14: RTL Schematic of 64-QAM Block Address Generator

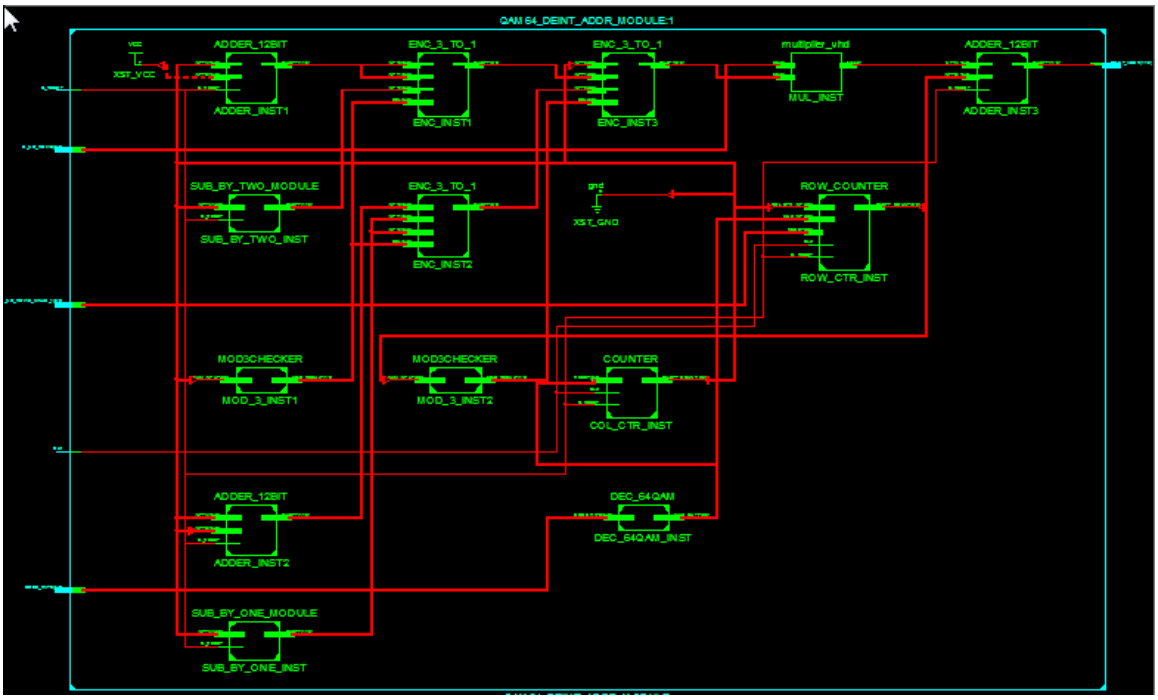

Figure 5.15: Detailed RTL Schematic of 64-QAM Block Address Generator

Minimum period: 7.087ns (Maximum Frequency: 141.103MHz)

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	133	3584	3%
Number of Slice Flip Flops	20	7168	0%
Number of 4 input LUTs	157	7168	2%
Number of bonded IOBs	26	195	13%
Number of GCLKs	1	24	4%

Figure 5.16: Device Utilization Summary of 64-QAM Block Address Generator

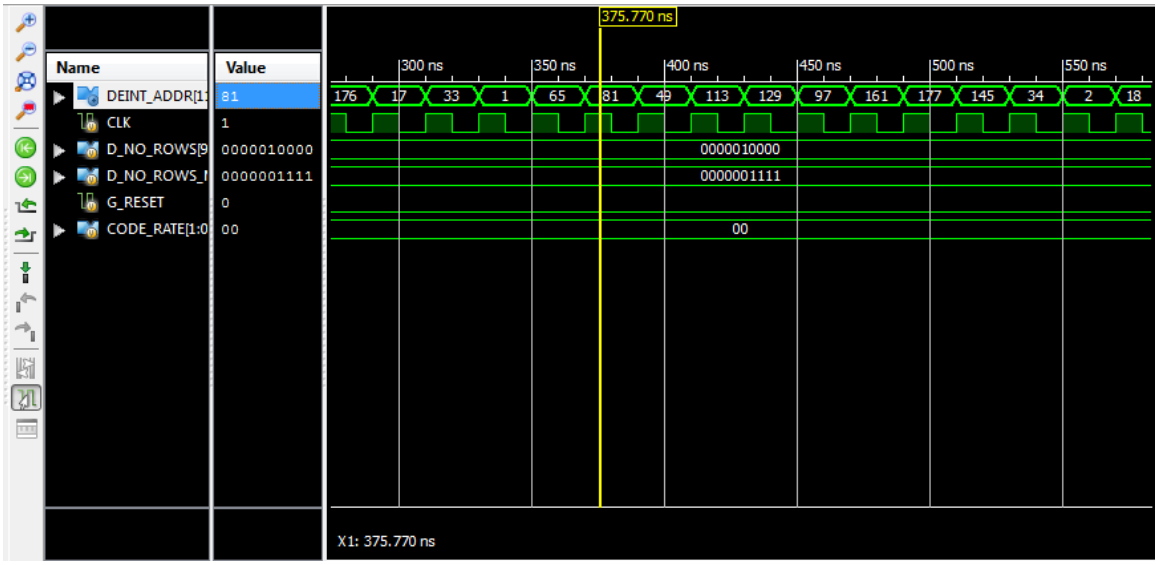

Figure 5.17: Simulation Results of 64-QAM Block Address Generator

5.7 Hardware Implementation of Address Generator for WiMAX Deinterleaver

In order to realize the complete hardware for address generator for WiMAX Deinterleaver the common blocks which are the multiplier unit (M1), the adder unit for generation of deinterleaver addresses (A1) and the row and column counter (R0 and C0) are shared resources for all the other blocks as shown in the Figure 5.18. The multiplexer for selecting the code rate are combined to form a common block (M0, M2, M5) which is selected by the type of modulation scheme. The address generation hardware for different

modulation techniques are kept as separate entities. Finally a multiplexer is used to select the deinterleaver address on basis of type of modulation scheme selected.

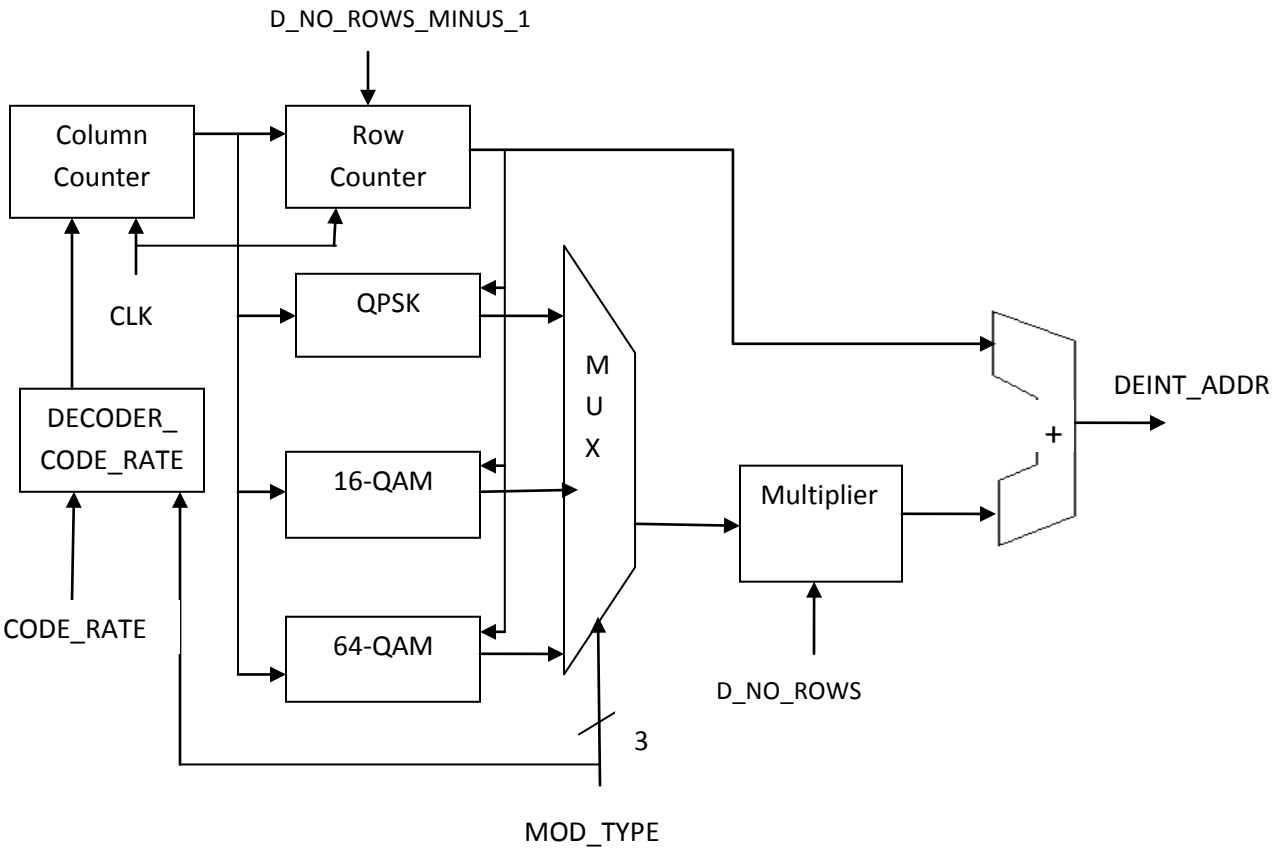

Figure 5.18: Hardware Design of Address Generator for WiMAX Deinterleaver

5.8 Simulation Results of Address Generator for WiMAX Deinterleaver

The synthesis of the above model is done using Verilog HDL (Appendix A.2) and the RTL schematic shown in Figure 5.19 and Figure 5.20. The description of the input and output is given below.

CODE_RATE [2:0] - 3 bit code rate input for ***Ncbps***

MOD_TYPE[1:0]- 2-bit selection line for modulation type

D_NO_ROWS [9:0] – 10 bit input for number of rows (***d***)

D_NO_ROWS_MINUS_1 [9:0] - 10 bit input for ***d-1***

CLK- Input Clock

G_RESET- Global Reset (Active High)

DEINT_ADDR- 12 bit Output Address (***kn***)

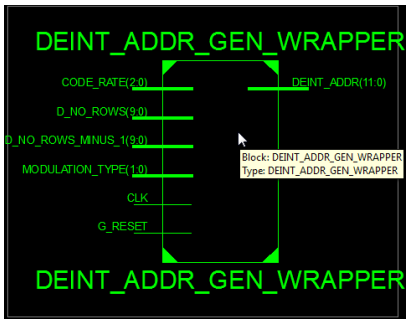

Figure 5.19: RTL Schematic of Address Generator for WiMAX Deinterleaver

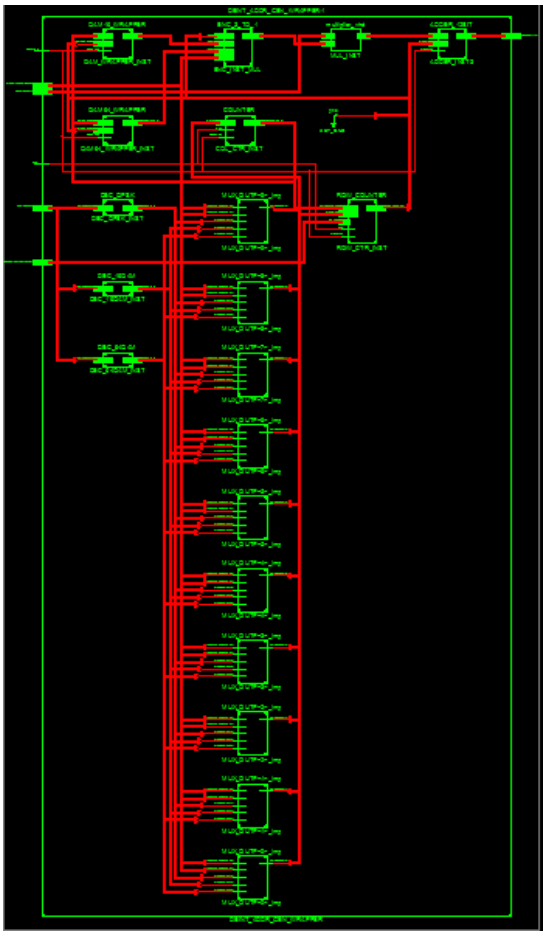

Figure 5.20: Detailed RTL Schematic of Address Generator for WiMAX Deinterleaver

Minimum period: 6.926ns (Maximum Frequency: 144.383MHz)

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	167	3584	4%
Number of Slice Flip Flops	21	7168	0%
Number of 4 input LUTs	216	7168	3%
Number of bonded IOBs	29	195	14%
Number of GCLKs	1	24	4%

Figure 5.21: Device Utilization Summary of Address Generator for WiMAX Deinterleaver

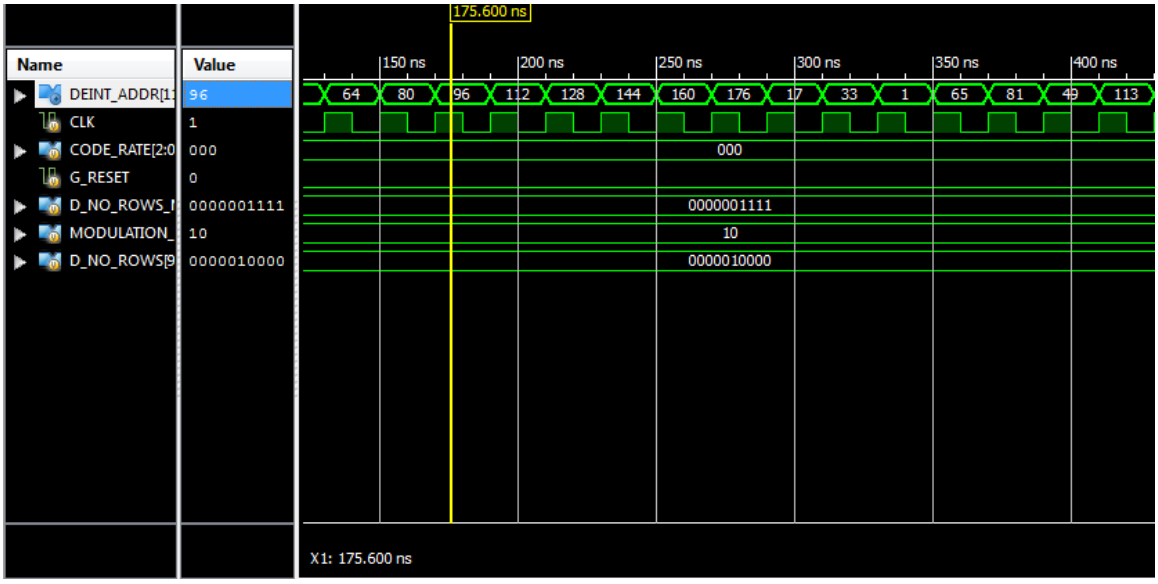

Figure 5.22: Simulation Results of Address Generator for WiMAX Deinterleaver

5.9 Comparative Analysis of Implementation Results

The implementation of the above technique is compared with the earlier implementation done for the design of the address generator block for WiMAX deinterleaver using Spartan 3 FPGA. It is found that a significant reduction in number of flip flops is obtained as compared to the work done by B.K.Upadhyaya et.al [16]. The operating frequency is also increased to 144.383 MHz. and there is reduction in the utilization of 4-input LUTs. This shows that the technique utilizes lesser lookup table as compared to the previous work. There is 14.6% increase in the number of slices which are being used.

As compared to the LUT Based technique there is significant increase in the FPGA Performance Parameters.

FPGA Parameters	Performance of Proposed Techniqu ue	Performance of B.K.Upadhyaya et.al.[16]	Performance of LUT Based Techniqu e[16]	%Reduction/Improvement with B.K.Upadhyaya et.al.[16]	% Reduction/Improvement with LUT Based Technique[16]
Slices	4%	3.49%	17.66%	14.6%	-77.34%
Flip flops	0.29%	0.50%	0.78%	-42%	-62.82%
4 input LUTs	3%	3.35%	17.15%	-10.44%	-82.50%
Operating Frequency	144.383 MHz	121.82 MHz	62.51MHz	15.62%	56.70%

Table 5.2: Performance Comparison

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusion

The IEEE 802.16 standard or the WiMAX is a solution widely in demand in Wireless Communication which has been developed with the rising demand in internet and multimedia based applications. It provides scalable network architecture with high bandwidth over a long-data transmission and a high data rate as per the user requirements for multimedia based applications.

In this work the physical layer of the WiMAX standard has been studied and the physical layer model implementation is analyzed. In the physical layer model the interleaver is an important block since it rearranges the original sequence of the transmitted data to a new one so that the burst of errors in the adjacent block is reduced. The deinterleaver performs the reverse operation of retrieving the coded bits which are now mapped to non-adjacent subcarriers.

The address generation for the deinterleaver is translated to digital design using Verilog HDL in this work by reducing the floor and modulo function into simpler algorithm with mathematical formulation essential for all code rates and modulation schemes as per the IEEE 802.16e standard. In the address generation scheme a modulus 3 block has been proposed using set of adders with the mathematical formulations. Thus a better operating frequency is achieved with significant improvement in resource utilization.

6.2 Future Work

The WiMAX transceiver structure can be developed on hardware using the optimized address generator for deinterleaver block. This can be used for Traffic Monitoring Network where the data is to be sent over large distances so as to regulate and monitor the traffic[17].

REFERENCES

[1] B. Li, Y. Qin, C. P. Low, and C. L. Gwee, "A survey on mobile WiMAX," IEEE Communication Magazine, vol. 45, no. 12, pp. 70–75, Dec. 2007.

[2] L.Nuaymi," WiMAX Technology for Broadband Wireless Access", Wiley Publications, 2007.

[3] Local and Metropolitan Networks—Part 16: Air Interface for Fixed Broadband Wireless Access Systems, IEEE Std. 802.16-2004, 2004.

[4] K. Etemad, "Overview of Mobile WiMAX Technology and Evolution," in IEEE Commun. Mag., vol. 46, no. 10, pp. 31–40, Oct. 2008.

[5] Wu, Zhongshan, "MIMO-OFDM Communication Systems: Channel Estimation and Wireless Location", PhD Thesis, Dept. of Electrical & Computer Engineering, Louisiana State University, USA, May 2006

[6] M.A.Hasan," Performance Evaluation of WiMAX/IEEE 802.16 OFDM Physical Layer",M.S Thesis, Department of Electrical and Communication Engineering, Helsinki University Of Technology, Espo, June 2007.

[7] J.W. Cooley and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Computation, vol. 19, pp. 297 - 301, 1965.

[8] http://nptel.ac.in/courses/Webcourse-contents/IIT%20Kharagpur/Digi%20Comm/pdf-m-5/m5l30.pdf

[9] Rohde & Shwarz. "WiMAX - General Information about the Standard 802.16 – ApplicationNotes." pp.34 2010.

[10] R.Prasad,F.J.Velez "Chapter 2 OFDMA WiMAX Physical Layer", WiMAX Networks, Springer 2010.

[11] M.Abramovici, M.A.Breur, A.D.Friedman,"Digital Systems Testing and Testable Design", IEEE Press, 1990.

[12] http://cmclab.rice.edu/433/slides/2-Modulation.pdf

[13] Y. N. Chang, "A low-cost dual mode de-interleaver design," in IEEE Transactions on Consumer Electronics, Vol. 54, No. 2, May 2008. pp326-332.

[14] R. Asghar and D. Liu, "2D realization of WiMAX channel interleaver for efficient hardware implementation," in World Academy of Science,Engineering and Technology, Hong Kong, 2009, vol. 3, pp. 25–29.

[15] B. K. Upadhyaya, I. S. Misra, and S. K. Sanyal, "Novel design of address generator for WiMAX multimode interleaver using FPGA based finite state machine," in Proceedings of 13th International Conference on Computing and Information Technology, Dhaka, Bangladesh, 2010, pp. 153–158.

[16] B. K. Upadhyaya, S. K. Sanyal, "Efficient Implementation of Address Generator for WiMAX Deinterleaver", IEEE Transactions on Circuits and Systems-II Express Briefs, August 2013,vol.60, pp.492-496.

[17] A.Khalatkar, S.Indu, R.Agarwal, V.Maurya "Implementation of Camera Based Traffic Monitoring System" WIT Transaction on Information and Communication Technologies, Vol.58,WIT Press,2014.pp.441-448.

[18] A. A. Khater, M. M. Khairy, and S. E.-D. Habib, "Efficient FPGA implementation for the IEEE 802.16e interleaver," in Proceedings of International Conference in Microelectronics, Marrakech, Morocco, 2009, pp. 181–184.

[19] Xilinx Spartan-3 FPGA Family: Complete Data Sheet, Xilinx, Inc., San Jose, CA, USA, 2012

Appendix A

A.1 MATLAB Codes for the different modulation schemes and code rates

```
function [ kn ] = deinterleaverWimax( Ncbps,d,modType )
%Address Generator For WIMAX deinterleaver
kn = zeros(d,Ncbps/d);
if(modType == 0)
    % Code for QPSK Address Generation
    for j = 0:1:d-1
        for i = 0:1:(Ncbps/d)-1
            kn((j+1),(i+1)) = d*i+j;
        end
    end
elseif(modType == 1)
    % Code for 16-QAM Address Generation
    for j = 0:1:d-1
        for i = 0:1:(Ncbps/d)-1
            tempMod2j = mod(j,double(2));
            tempMod2i = mod(i,double(2));
            if(tempMod2j == 0)
                kn((j+1),(i+1))=d*i + j;
            else
                if(tempMod2i == 0)
                    kn((j+1),(i+1)) = d*(i+1) + j;
                else
                    kn((j+1),(i+1))= d*(i-1) + j;
                end
            end

        end
    end
else
    % Code for 64-QAM Address Generation
    for j = 0:1:d-1
        for i = 0:1:(Ncbps/d)-1
            tempModVarj = (mod(j,double(3)));
            tempModVari = (mod(i,double ((3))));
            if(tempModVarj == 0)
               kn((j+1),(i+1))=d*i + j;
            elseif(tempModVarj == 1)
                if(tempModVari == 2)
                    kn((j+1),(i+1))=d*(i-2) + j;
                else
                    kn((j+1),(i+1))=d*(i+1) + j;
                end
            else
                if(tempModVari == 0)
                    kn((j+1),(i+1))=d*(i+2) + j;
                else
                    kn((j+1),(i+1))=d*(i-1) + j;
                end
            end
       end
```

```
    end
  end end
```

A.2 Verilog Codes

```
module DEINT_ADDR_GEN_WRAPPER(CLK, CODE_RATE, G_RESET, D_NO_ROWS,
D_NO_ROWS_MINUS_1,MODULATION_TYPE, DEINT_ADDR  );
/*************************************Port Declaration*********************/
        output [11:0]DEINT_ADDR;
        input CLK;
        input [9:0]D_NO_ROWS;
        input [9:0]D_NO_ROWS_MINUS_1;
        input G_RESET;
        input [2:0]CODE_RATE;
        input [1:0]MODULATION_TYPE;
/*************************************Intermediate Signal Declaration ********/
//Column Counter and Comparator output
        wire [9:0]COL_CTR_OP;
//Row Counter and Comparator output
        wire [9:0]ROW_CTR_OP;
//Outputs to Encoder
        wire [9:0]QAM16_OP_ENC,QAM64_OP_ENC;
//Output to Multiplier
        wire [9:0]ENC_OP_MUL;
//Multiplier Output
        wire [11:0]MULT_OUTP;
        wire [9:0]MUX_OUTP;
        wire [9:0]MUX_OUTP1,MUX_OUTP2,MUX_OUTP3;
/*************************************Instance Connections ***************************/
//Column Counter Block
COUNTER
COL_CTR_INST(.CLK(CLK),.RESET(MUX_OUTP),.G_RESET(G_RESET),.OUTPT_COUNT(COL_CTR_OP));//
Row Counter
ROW_COUNTER
ROW_CTR_INST(.CLK(CLK),.RESET(D_NO_ROWS_MINUS_1),.G_RESET(G_RESET),.MUX_INP(MUX_OUT
P),.COL_CTR_INP(COL_CTR_OP),.OUTPT_COUNT(ROW_CTR_OP));
//QAM16 Wrapper
QAM16_WRAPPER
QAM_WRAPPER_INST(.COL_CTR_IP(COL_CTR_OP),.ROW_CTR_IP(ROW_CTR_OP),.G_RESET(G_RESET),.
ENC_OP_MUL(QAM16_OP_ENC));
//QAM64 Wrapper
QAM64_WRAPPER
QAM64_WRAPPER_INST(.COL_CTR_IP(COL_CTR_OP),.ROW_CTR_IP(ROW_CTR_OP),.G_RESET(G_RESET
),.ENC_OP_MUL(QAM64_OP_ENC));
//Encoder Module
ENC_3_TO_1
ENC_INST_MUL(.OUTPT(ENC_OP_MUL),.INPT0(ROW_CTR_OP),.INPT1(QAM16_OP_ENC),.INPT2(QAM64_
OP_ENC),.SEL(MODULATION_TYPE));
//Multiplier for module_multiplier
multiplier_vhd MUL_INST(.p(MULT_OUTP),.a(D_NO_ROWS),.b(ENC_OP_MUL));
//Adder 12 Bit
ADDER_12BIT
ADDER_INST3(.INPTA(MULT_OUTP),.INPTB({2'b00,ROW_CTR_OP}),.OUTPT(DEINT_ADDR),.G_RESET(G_
RESET));
//Decoder
DEC_QPSK DEC_QPSK_INST(.MUX_OUTP(MUX_OUTP1),.CODE_RATE(CODE_RATE));
DEC_16QAM DEC_16QAM_INST(.MUX_OUTP(MUX_OUTP2),.CODE_RATE(CODE_RATE) );
DEC_64QAM DEC_64QAM_INST(.MUX_OUTP(MUX_OUTP3),.CODE_RATE(CODE_RATE));
assign MUX_OUTP = (MODULATION_TYPE == 2'b00) ? MUX_OUTP1:(MODULATION_TYPE == 2'b01) ?
MUX_OUTP2:MUX_OUTP3;
endmodule
```

```
module COUNTER(CLK, RESET, G_RESET, OUTPT_COUNT);
input CLK;
//Reset Count
  input [9:0]RESET;
input G_RESET;
output [9:0] OUTPT_COUNT;
reg [9:0] OUTPT_COUNT;
always @(posedge CLK or posedge G_RESET)
        begin
                if(G_RESET == 1'b1 )
                        begin
                                OUTPT_COUNT <= 10'b0000000000;
                                //ROW_CTR_INC <= 1'b0;
                        end
                else if (OUTPT_COUNT == RESET)
                        begin
                                OUTPT_COUNT <= 10'b0000000000;
                                //ROW_CTR_INC <= 1'b1;
                        end
                else
                        begin
                                OUTPT_COUNT <= OUTPT_COUNT + 1;
                                //ROW_CTR_INC <= 1'b0;
                        end
        end
endmodule

module ROW_COUNTER(CLK, RESET, G_RESET, MUX_INP, COL_CTR_INP, OUTPT_COUNT);
 input CLK;
 input [9:0]RESET;
 input G_RESET;
//Used To monitor Output of COlumn Counter
input [9:0] MUX_INP;
input [9:0] COL_CTR_INP;
output [9:0] OUTPT_COUNT;
reg [9:0] OUTPT_COUNT;
//Internal Reset to Store When G_RESET is pressed
reg INT_G_RESET;
always @(posedge CLK or posedge G_RESET)
        begin
                if(G_RESET == 1'b1 )
                        begin
                                OUTPT_COUNT <= 10'b0000000000;
                        end
                else if(COL_CTR_INP == MUX_INP)
                        begin
                                                        OUTPT_COUNT <= OUTPT_COUNT + 1;
                        end
end
endmodule
module QAM16_WRAPPER(COL_CTR_IP,ROW_CTR_IP, G_RESET,ENC_OP_MUL );
/*************************************Port Declaration*********************************/
output [9:0] ENC_OP_MUL;
input [9:0] COL_CTR_IP;
input [9:0] ROW_CTR_IP;
input G_RESET;
/*************************************Intermediate Signal Declaration ******************/
//Adder 12-bit output to be input to encoder
wire [9:0]ADDER_OP,SUBTR_OP;
//MOD-2 Counter Output
  wire MOD_2_COL_OP,MOD_2_ROW_OP;
//Encoder OUtput
```

```
  wire [9:0]ENC_OP;
/****************************************Instance Connections ************************/
//Adder 12 bit
ADDER_12BIT
ADDER_INST(.INPTA({2'b00,COL_CTR_IP}),.INPTB({12'b000000000001}),.OUTPT(ADDER_OP),.G_RESET(G
_RESET));
//Subtractor 12 bit
SUB_BY_ONE_MODULE
SUB_BY_ONE_INST(.INPTA({2'b00,COL_CTR_IP}),.OUTPT(SUBTR_OP),.G_RESET(G_RESET));
//MOD-2 Module
MOD_2_MODULE MOD_2_INST1(.OUTPT(MOD_2_COL_OP),.INPT(COL_CTR_IP));
//Encoder Module
ENC_2_TO_1
ENC_INST1(.OUTPT(ENC_OP),.INPT0(ADDER_OP),.INPT1(SUBTR_OP),.SEL(MOD_2_COL_OP));
//MOD-2 Module
MOD_2_MODULE MOD_2_INST2(.OUTPT(MOD_2_ROW_OP),   .INPT(ROW_CTR_IP));
//Encoder Module
ENC_2_TO_1
ENC_INST2(.OUTPT(ENC_OP_MUL),.INPT0(COL_CTR_IP),.INPT1(ENC_OP),.SEL(MOD_2_ROW_OP));
Endmodule
module QAM64_DEINT_ADDR_MODULE(  DEINT_ADDR, CLK, D_NO_ROWS, D_NO_ROWS_MINUS_1,
G_RESET, CODE_RATE);
output [11:0] DEINT_ADDR;
input CLK;
input [9:0]D_NO_ROWS;
input [9:0]D_NO_ROWS_MINUS_1;
input G_RESET;
input [1:0] CODE_RATE;
wire [9:0]MUX_OUTP;
//Column Counter and Comparator output
wire [9:0]COL_CTR_OP;
//Row Counter and Comparator output
  wire [9:0]ROW_CTR_OP;
//Adder 12-bit output to be input to encoder
wire [9:0]ADDER_OP_1,ADDER_OP_2;
//Subtractor 12-bit output to be input to encoder
wire [9:0]SUBTR_OP_1,SUBTR_OP_2;
//MOD-3 Counter Output
 wire [1:0]MOD_3_COL_OP,MOD_3_ROW_OP;
//Encoder OUtput
  wire [9:0]ENC_OP_1,ENC_OP_2,ENC_OP_3;
//Multiplier Output
wire [11:0] MULT_OUTP;
//Column Counter
COUNTER
COL_CTR_INST(.CLK(CLK),.RESET(MUX_OUTP),.G_RESET(G_RESET),.OUTPT_COUNT(COL_CTR_OP));
//Adder 12 bit
ADDER_12BIT ADDER_INST1(.INPTA({2'b00,COL_CTR_OP}), .INPTB({12'b000000000001}),
.OUTPT(ADDER_OP_1),  .G_RESET(G_RESET)  );
//Substractor by 2 Module
SUB_BY_TWO_MODULE SUB_BY_TWO_INST(.INPTA({2'b00,COL_CTR_OP}),
.OUTPT(SUBTR_OP_2),.G_RESET(G_RESET)   );
//Adder 12 bit
ADDER_12BIT ADDER_INST2(.INPTA({2'b00,COL_CTR_OP}),  .INPTB({12'b000000000010}),
.OUTPT(ADDER_OP_2),  .G_RESET(G_RESET)  );
//Subtractor by 1 Module
SUB_BY_ONE_MODULE
SUB_BY_ONE_INST(.INPTA({2'b00,COL_CTR_OP}),.OUTPT(SUBTR_OP_1),.G_RESET(G_RESET) );
//MOD-3 Module
MOD3CHECKER MOD_3_INST1(.MOD_RESULT(MOD_3_COL_OP), .NUM_INPUT(COL_CTR_OP));
//Encoder Module
```

```
ENC_3_TO_1 ENC_INST1(.OUTPT(ENC_OP_1),  .INPT0(ADDER_OP_1),  .INPT1(ADDER_OP_1),
.INPT2(SUBTR_OP_2),  .SEL(MOD_3_COL_OP));
//Encoder Module
ENC_3_TO_1 ENC_INST2(.OUTPT(ENC_OP_2), .INPT0(ADDER_OP_2), .INPT1(SUBTR_OP_1),
.INPT2(SUBTR_OP_1),  .SEL(MOD_3_COL_OP));
//Row Counter
ROW_COUNTER ROW_CTR_INST(.CLK(CLK),.RESET(D_NO_ROWS_MINUS_1),.G_RESET(G_RESET),
.MUX_INP(MUX_OUTP),.COL_CTR_INP(COL_CTR_OP),.OUTPT_COUNT(ROW_CTR_OP));
//MOD-3 Module
MOD3CHECKER MOD_3_INST2(.MOD_RESULT(MOD_3_ROW_OP), .NUM_INPUT(ROW_CTR_OP));
//Encoder Module
ENC_3_TO_1 ENC_INST3(.OUTPT(ENC_OP_3),  .INPT0(COL_CTR_OP),.INPT1(ENC_OP_1),
.INPT2(ENC_OP_2),.SEL(MOD_3_ROW_OP));
//Multiplier for module_multiplier
multiplier_vhd MUL_INST(.p(MULT_OUTP),.a(D_NO_ROWS),.b(ENC_OP_3));
//Adder 12 Bit
ADDER_12BIT
ADDER_INST3(.INPTA(MULT_OUTP),.INPTB({2'b00,ROW_CTR_OP}),.OUTPT(DEINT_ADDR),.G_RESET(G_
RESET)  );
DEC_64QAM DEC_64QAM_INST(.MUX_OUTP(MUX_OUTP),.CODE_RATE(CODE_RATE));
endmodule
module MOD3CHECKER(NUM_INPUT,MOD_RESULT);
input [9:0]NUM_INPUT;
output [1:0] MOD_RESULT;
//For Intermediate Sum
wire [1:0]INTRMD_SUM1;
wire [1:0]INTRMD_SUM2;
wire [1:0]INTRMD_SUM3;
wire [1:0]INTRMD_SUM4;
wire [1:0]FINAL_SUM;
//For Intermediate Carry
wire INTRMD_CARRY1;
wire INTRMD_CARRY2;
wire INTRMD_CARRY3;
wire INTRMD_CARRY4;
//Added For Testing
FULLADDER_2BIT FA2BITINST1(.A(NUM_INPUT[1:0]),.B(NUM_INPUT[3:2]),.CIN(1'b0),
.SUM(INTRMD_SUM1),.COUT(INTRMD_CARRY1));
FULLADDER_2BITFA2BITINST2(.A(NUM_INPUT[5:4]),.B(INTRMD_SUM1),.CIN(INTRMD_CARRY1),
.SUM(INTRMD_SUM2), .COUT(INTRMD_CARRY2) );
FULLADDER_2BIT FA2BITINST3(.A(NUM_INPUT[7:6]),.B(INTRMD_SUM2),.CIN(INTRMD_CARRY2),
.SUM(INTRMD_SUM3),.COUT(INTRMD_CARRY3));
FULLADDER_2BIT FA2BITINST4(.A(NUM_INPUT[9:8]),.B(INTRMD_SUM3),.CIN(INTRMD_CARRY3),
.SUM(INTRMD_SUM4),.COUT(INTRMD_CARRY4));
FULLADDER_2BIT FA2BITINST5(.A(INTRMD_SUM4),.B({1'b0,INTRMD_CARRY4}),.CIN(1'b0),
.SUM(FINAL_SUM));
 assign MOD_RESULT = (FINAL_SUM == 2'b00) ? 2'b00  :(FINAL_SUM == 2'b01) ? 2'b01 :(FINAL_SUM ==
2'b10) ? 2'b10 : 2'b00;
endmodule
```