DOCSIS 3.1 Cable Modem and Upstream Channel Simulation in MATLAB

A Thesis Submitted
to the College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Electrical and Computer Engineering
University of Saskatchewan

by

Ben Fortosky

Saskatoon, Saskatchewan, Canada

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, it is agreed that the Libraries of this University may make it freely available for inspection. Permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professors who supervised this thesis work or, in their absence, by the Head of the Department of Electrical and Computer Engineering or the Dean of the College of Graduate Studies and Research at the University of Saskatchewan. Any copying, publication, or use of this thesis, or parts thereof, for financial gain without the written permission of the author is strictly prohibited. Proper recognition shall be given to the author and to the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

Request for permission to copy or to make any other use of material in this thesis in whole or in part should be addressed to:

Head of the Department of Electrical and Computer Engineering
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5A9

Acknowledgments

I would like to thank my faculty supervisors, Professor J. Eric Salt, Professor Ha H. Nguyen, and my industry supervisor, Dr. Brian Berscheid for their encouragement and patience throughout the duration of the Industry Oriented Master's (IOM) program. The courses they taught and the information they provided served as excellent resources throughout the development of my project.

I would also like to thank Vecima Networks Inc., who provided the funding for my research in conjunction with the Natural Sciences and Engineering Research Council of Canada (NSERC). Furthermore, the mentorship provided by Dr. Berscheid during my work term at Vecima was very helpful, and something for which I am incredibly grateful.

A very special thanks goes to my fellow IOM cohorts, Tung Nguyen, Yayi Xiao, and Chad Holst. Their ideas and work helped guide my research, and their friendship served as a constant source of motivation and inspiration over the years.

Finally, I must thank my fiancée, Lee-Anne Gilecki, and my parents, Owen and Deborah Fortosky, for the constant love and support they have provided me with over the years. I could not have done it without you.

Abstract

The cable television (CATV) industry has grown significantly since its inception in the late 1940's. Originally, a CATV network was comprised of several homes that were connected to community antennae via a network of coaxial cables. The only signal processing done was by an analogue amplifier, and transmission only occurred in one direction (i.e. from the antennae/head-end to the subscribers). However, as CATV grew in popularity, demand for services such as pay-per-view television increased, which lead to supporting transmission in the upstream direction (i.e. from subscriber to the head-end). This greatly increased the signal processing to include frequency diplexers.

CATV service providers began to expand the bandwidth of their networks in the late 90's by switching from analogue to digital technology. In an effort to regulate the manufacturing of new digital equipment and ensure interoperability of products from different manufacturers, several cable service providers formed a non-for-profit consortium to develop a data-over-cable service interface specification (DOCSIS). The consortium, which is named CableLabs, released the first DOCSIS standard in 1997.

The DOCSIS standard has been upgraded over the years to keep up with increased consumer demand for large bandwidths and faster transmission speeds, particularly in the upstream direction. The latest version of the DOCSIS standard, DOCSIS 3.1, utilizes orthogonal frequency-division multiple access (OFDMA) technology to provide upstream transmission speeds of up to 1 Gbps. As cable service providers begin the process of upgrading their upstream receivers to comply with the new DOCSIS 3.1 standard, they require a means of testing the various functions that an upstream receiver may employ. It is convenient for service providers to employ cable modem (CM) plus channel emulator to perform these tests in-house during the product development stage. Constructing the emulator in digital technology is an attractive option for testing.

This thesis approaches digital emulation by developing a digital model of the CMs and upstream channel in a DOCSIS 3.1 network. The first step in building the emulator is to simulate its operations in MATLAB, specifically upstream transmission over the network.

The MATLAB model is capable of simulating transmission from multiple CMs, each of which transmits using a specific "transmission mode." The three transmission modes described in the DOCSIS 3.1 standard are included in the model. These modes are "traffic mode," which is used during regular data transmission; "fine ranging mode," which is used to perform fine timing and power offset corrections; and "probing" mode, which is presumably used for estimating the frequency response of the channel, but also is used to further correct the timing and power offsets.

The MATLAB model is also capable of simulating the channel impairments a signal may encounter when traversing the upstream channel. Impairments that are specific to individual CMs include integer and fractional timing offsets, micro-reflections, carrier phase offset (CPO), fractional carrier frequency offset (CFO), and network gain/attenuation. Impairments common to all CMs include carrier hum modulation, AM/FM ingress noise, and additive white Gaussian noise (AWGN).

It is the hope that the MATLAB scripts that make up the simulation be translated to Verilog HDL to implement the emulator on a field-programmable gate array (FPGA) in the near future. In the event that an FPGA implementation is pursued, research was conducted into designing efficient fractional delay filters (FDFs), which are essential in the simulation of micro-reflections. After performing an FPGA implementation cost analysis between various FDF designs, it was determined that a Kaiser-windowed sinc function FDF with roll-off parameter $\beta = 3.88$ was the most cost-efficient choice, requiring at total of 24 multipliers and 32,562 bits of memory when implemented using an optimized structure.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AWGN - Additive White Gaussian Noise

BPSK - Binary Phase-Shift Keying

CATV - Community Antenna Television, also Cable Television

CFO - Carrier Frequency Offset

CM - Cable Modem

CMTS - Cable Modem Termination System

CNR - Carrier to Noise Ratio

CPO - Carrier Phase Offset

dB - DeciBels

dBc - DeciBels with respect to Carrier

DFT - Discrete Fourier Transform

DOCSIS - Data Over Cable Service Interface Specification

DSL - Digital Subscriber Line

FDF - Fractional Delay Filter

FDMA - Frequency Division Multiple Access

FEC - Forward Error Correction

FFT - Fast Fourier Transform

FIR - Finite Impulse Response

FPGA - Field Programmable Gate Array

IFFT - Inverse Fast Fourier Transform

ICI - Inter-Carrier Interference

IIR - Infinite Impulse Response

IPv6 - Internet Protocol version 6

ISI - Inter-Symbol Interference

LDPC - Low Density Parity Check

LFSR - Linear Feedback Shift-Register

LUT - Look-Up Table

MAC - Media Access Control

MATLAB - MATrix LABoratory

MSE - Mean Squared Error

OFDM - Orthogonal Frequency Division Multiplexing

OFDMA - Orthogonal Frequency Division Multiple Access

PAPR - Peak to Average Power Ratio

QAM - Quadrature Amplitude Modulation

QoS - Quality of Service

QPSK - Quadrature Phase-Shift Keying

RAM - Random Access Memory

RC - Raised Cosine

RF - Radio Frequency

S-CDMA - Synchronous Code Division Multiple Access

SNR - Signal to Noise Ratio

TDMA - Time Division Multiple Access

1. Introduction

1.1 Brief History of Cable Television in North America

The origins of cable television in North America date back to the late 1940's in the rural United States, which was a time when many rural communities had poor antennae reception due to geographic factors. To remedy this, innovators such as L. E. "Ed" Parsons of Astoria, Oregon [1] and John Walson of Mahoney City, Pennsylvania [2] began independently creating community antenna/access television (CATV) systems. An antenna would be placed on an elevated surface in the community in order to capture television signals broadcast from nearby major urban centres. Coaxial cables were strung from the community antennae to the households in the community, with amplifiers placed appropriately along the path to compensate for the attenuation of the cable. This technology continued to spread across rural America until the late 1960's, by which point the market had reached saturation [3].

Several technological innovations introduced over the next decade would cause CATV to experience rapid growth, allowing it to expand beyond its traditional rural market. One such innovation was the addition of an upstream return path to allow two-way communication over a cable network. Arpanet founder and RAND Corporation analyst Paul Baran contributed significantly to development of the concept in the late 1960's [4]. Although Baran initially considered the technology for military defensive purposes [4], as the civilian applications become more apparent companies such RAND, MITRE, Coaxial Scientific, and Theta-Com SRS began developing early prototypes of two-way communication systems for consumers in urban centres [5]. Ultimately, this research would result in some of the first cable modems.

Summarized by E. K. Smith in 1975 [5], four main categories of services were explored

during the upstream prototyping phase: "narrow-band subscriber response services (e.g. opinion polling, sensor monitoring, pay-TV [ordering]); shared two-way channels (e.g. ... remote medical diagnosis, ...); subscriber-initiated services (e.g. ... reservation and banking services, catalogue shopping); and point-to-point services (e.g. high-speed data-exchange, teleconferencing, and fascimile)." The majority of these services, however, would not be offered until the introduction of digital cable years later. The primary reason for this was content providers such as Home Box-Office (HBO) had already discovered high demand for pay-television [6]. Thus, the focus of most CATV service providers and manufacturers shifted towards developing effective pay-television systems.

Focusing on pay-television would ultimately pay off thanks to the introduction of satellite technology. Following the launch of SATCOM I in 1975, many television stations began making the transition from terrestrial to satellite broadcasting [6]. This resulted in a plethora of additional channels being made available to subscribers, as well as an expansion of the broadcast frequency spectrum in order to accommodate them. With additional channels and pay-television services now accessible to the general populous, cable became much more than a means of improving signal quality. It evolved to a highly desired source of entertainment. As a result, cable television experienced a surge in popularity, allowing it to break through into the previously untapped large urban-centre market [1].

The next major change for the cable industry would occur during the late 1980's and early 1990's with the advent of the Internet. As customer demand for faster, higher quality services arose, the industry began transitioning from analogue to digital signal distribution. An integral part of this transition involved subdividing their existing distribution systems in order to implement hybrid-fibre coaxial (HFC) networks. The HFC networks allowed for faster and more efficient data transmission amongst multiple users, which rivalled that of digital subscriber line (DSL) technologies offered by many telephone companies [7]. Knowing this, many cable companies began competing directly with telephone companies by upgrading their cable modems to provide both high-speed data and voice services in addition to the traditional television and pay-per-view services.

Unfortunately, at the time, there were no standards in place to govern the manufacturing

of cable equipment. As a result, CATV service providers purchased proprietary equipment from a single manufacturer. To ensure fair competition, the service providers collaborated to form CableLabs, a not-for-profit consortium whose purpose was to generate open standards that ensured interoperability among different manufacturers of the same equipment. The first standard developed by CableLabs was the data-over-cable service interface specification, DOCSIS.

1.2 Legacy DOCSIS

CableLabs released the original North American data-over-cable standard, DOCSIS 1.0, in 1997. In this first version of the standard, a DOCSIS network was defined as multiple cable modems (CMs) connected over a hybrid fibre-coaxial (HFC) network to a cable modem termination system (CMTS) [8], as shown in Figure 1.1. When transmitting upstream (direction from the subscriber to the service provider), the CMs are viewed as transmitters, and the CMTS as the receiver. The opposite is true when transmitting downstream (direction from the service provider to subscriber).

Although the network structure itself has remained relatively unchanged over the years, the DOCSIS standard has continued to grow and adapt to rising user demands for larger bandwidth and faster transmission speeds. The demand has been particularly high in the

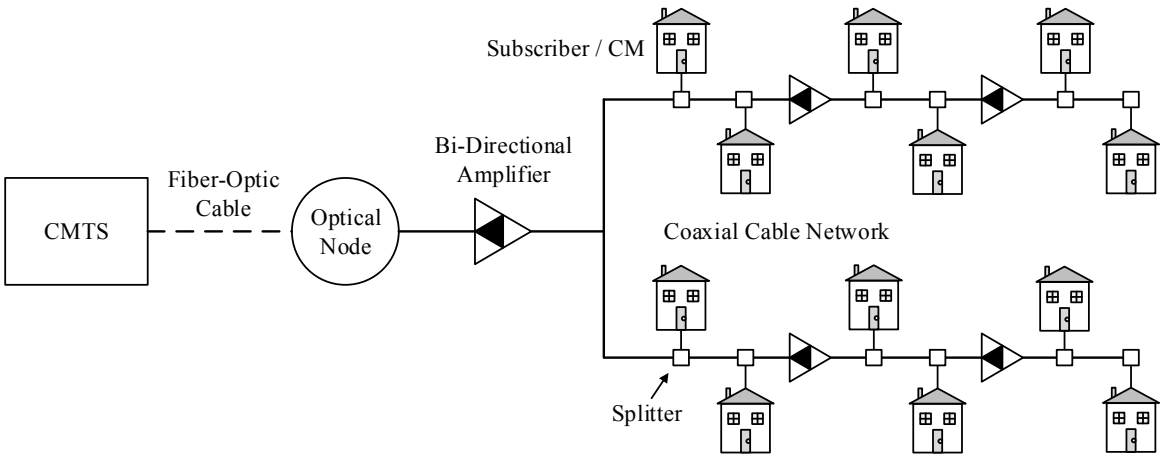


Figure 1.1: DOCSIS Network

upstream direction, which is the direction from the subscriber to the network. This has resulted in three major upgrades to the DOSCIS standard. These upgrades are referred to as version 2.0, released December 31, 2001; version 3.0, released August 4, 2006 and 3.1, released October 29, 2013. It should be noted that DOCSIS 1.1 was released prior to DOCSIS 2.0; however, the changes made between DOCSIS 1.0 and DOCSIS 1.1 were focused primarily on quality of service (QoS) standardization. As such, DOCSIS 1.0 and DOCSIS 1.1 are now often collectively referred to as DOCSIS 1.x. Significant upgrades between DOCSIS 1.x and 2.0 included the addition of a synchronous code-division multiple access (S-CDMA) format, as well as an increase in the maximum upstream quadrature amplitude modulation (QAM) constellation size from 16-QAM to 64-QAM (128-QAM using S-CDMA with trellis-coded modulation (TCM) encoding). DOCSIS 3.0 expanded upon this further by adding channel bonding, as well as support for Internet protocol version 6 (IPv6). A comparison of several aspects of legacy DOCSIS systems, including their maximum data rates, can be found in Table 1.1.

1.3 DOCSIS 3.1

As evident in Table 1.1, many significant changes have been introduced in the latest version of the standard, DOCSIS 3.1. The bandwidths in both the upstream and downstream directions have increased considerably. The downstream bandwidth could span from 54 MHz to 1.794 GHz. The upstream could span from from 5 MHz to 204 MHz and beyond, which is over six times the original 30 MHz limit used in the 1970's [5]. Different channels in both DOCSIS 3.0 and 3.1 can be bonded into a single service flow. Perhaps the most important addition to DOCSIS 3.1 is orthogonal frequency-division multiplexing (OFDM) in the downstream, and orthogonal frequency division multiple access (OFDMA) in the upstream.

In an OFDM system, the available frequency spectrum is subdivided into multiple orthogonal subcarriers. The input data stream is converted from serial to parallel and distributed amongst the subcarriers. Data from each active subcarrier is then mapped to a QAM constellation. Following mapping, the modulation is accomplished by taking an inverse fast

Table 1.1: Evolution of DOCSIS in the Upstream

Channel Configuration	**DOCSIS Version** [8] [9] [10] [11] [12]			
	1.x	**2.0**	**3.0**	**3.1**
Lower Operating Range (MHz)	5	5	5	5
Upper Operating Range (MHz)	42	42	42 (85)	42 (65, 85, 117, 204)
Max QAM Constellation Size	16	64 (128 w/ S-CDMA TCM)	64 (128 w/ S-CDMA TCM)	4096
Multiplexing Technique	FDMA, TDMA	FDMA, TDMA, S-CDMA	FDMA, TDMA, S-CDMA	OFDMA, TDMA, S-CDMA
Max Raw Data Rate	10.24 Mbps (9) Mbps	30.72 Mbps (27) Mbps	$n \times 30.72$ Mbps ($n \times$ 27 Mbps)[i]	1 Gbps

[i] Here, n refers to the number of bonded channels in a DOCSIS 3.0 system.

Fourier transform (IFFT) of the complex scalers for the subcarriers in the bandwidth. The IFFT output is then lengthened by prefixing copies of the samples at the end of the IFFT. Because the prefix is a copy of the IFFT, it is referred to as a "cyclic prefix". This cyclic prefix can effectively eliminate inter-symbol interference (ISI), provided it is longer than the channel delay spread [13].

OFDMA is used for upstream transmission in DOCSIS 3.1. The main difference between the OFDM and OFDMA is that OFDM has a single transmitter while OFDMA allows several devices to transmit simultaneously. The latter allows different subcarriers to be allocated to different users on a frame-by-frame basis. OFDMA has several significant advantages over time-division multiple access (TDMA) and code-division multiple access (CDMA), which are legacy DOCSIS transmission formats. First and foremost, it is scalable in bandwidth. As

such, the IFFT size can be matched to the channel bandwidth while fixing the subcarrier spacing. Specifically, an upstream DOCSIS 3.1 OFDMA system has a maximum bandwidth of 95 MHz and utilizes an IFFT size of either 2048 or 4096, with respective subcarriers spacing of 50 kHz and 25 kHz. Other upstream transmission methods such as TDMA and CDMA are not easily scalable to such large bandwidths [13].

The flexible power distribution in OFDMA is also an asset. Unlike TDMA, which distributes power evenly across the entire bandwidth, the spectrum of OFDMA can be shaped by scaling the power in the subcarriers. This is particularly useful when providing service to the more "noisy" users, which are typically located far away, as their assigned subcarriers can be given more power to elevate the signal-to-noise ratio (SNR) and improve the signal quality [14]. Additionally, if ingress interference is present and dominates a particular subcarrier, then that subcarrier can be turned off or excluded. This is not possible with TDMA or CDMA [13].

While the advantages of OFDMA are significant, the technique itself is not without drawbacks. Since each sample of the IFFT output is the sum of N independent variables, the resulting time domain sequence has an asymptotically Gaussian amplitude distribution. This means that there will be a high peak-to-average power ratio (PAPR) [14]. Also, depending on the subcarrier spacing, OFDMA can be very sensitive to downconversion frequency offsets and phase noise, making frequency synchronization critical [13].

Ultimately, through the use of OFDMA, the goal of DOCSIS 3.1 in the upstream is to increase data speeds up to 1 Gbps. The standard has also implemented the use of low-density parity-check (LDPC) encoding in conjunction with OFDMA, which has allowed the maximum upstream QAM constellation sizes to be increased to 4096. Theoretically, this will result in the ability to transmit up to 50% more data over existing HFC networks, provided the appropriate channel conditions are met.

1.4 Problem Statement

Cable companies are already positioning themselves to adopt DOCSIS 3.1 technology. In order to ensure the new DOCSIS 3.1 equipment meets specifications, they must be able to simulate data transmission over a DOCSIS 3.1 network. The simulation of data transmission in the upstream direction can be achieved by modelling both a cable modem (CM) and an upstream cable channel in a digital environment. The main objective of this research is the design of a MATLAB model of the upstream path in the network for the purpose of testing a DOCSIS 3.1 OFDMA receiver. The transmission model must include the generation of DOCSIS 3.1 OFDMA frames, as well as the possible channel impairments that a transmitted signal may encounter when travelling upstream over an HFC network.

In order to model a DOCSIS 3.1 CM, its internal functionality must first be understood. A DOCSIS 3.1 CM is capable of converting data from an external device into a transmittable OFDMA frame. Although the format of the data may vary depending on the type of external device connected, it is assumed for the purposes of this research that the data to be transmitted is presented as a serial stream of binary numbers. With this assumption, the CM's functionality begins at the input to the serial to parallel converter, as depicted in Figure 1.2. Following conversion from serial to parallel, the data must be allocated to specific subcarriers, mapped to an appropriate QAM constellation, pre-equalized, and then modulated using an IFFT. The CM must then add both a cyclic prefix and a roll-off period to the time-domain signal and shape the combined sequence using a Tukey raised-cosine (RC) window. Finally, the CM sends the windowed time-domain signal up the channel. There is no need for upconversion in a digital model as the radio-frequency (RF) signal and all RF processing can be done in baseband. Baseband equivalent operation will be explored in more detail in Chapter 2.

As the signal travels over an upstream cable channel, it can encounter impairments that can be categorized as one of two types. The first category includes impairments that can be considered unique to each individual CM, such as fractional and integer timing offsets, micro-reflections, carrier phase offset (CPO), carrier frequency offset (CFO), and network gain/attenuation. The second category includes common impairments that affect the re-

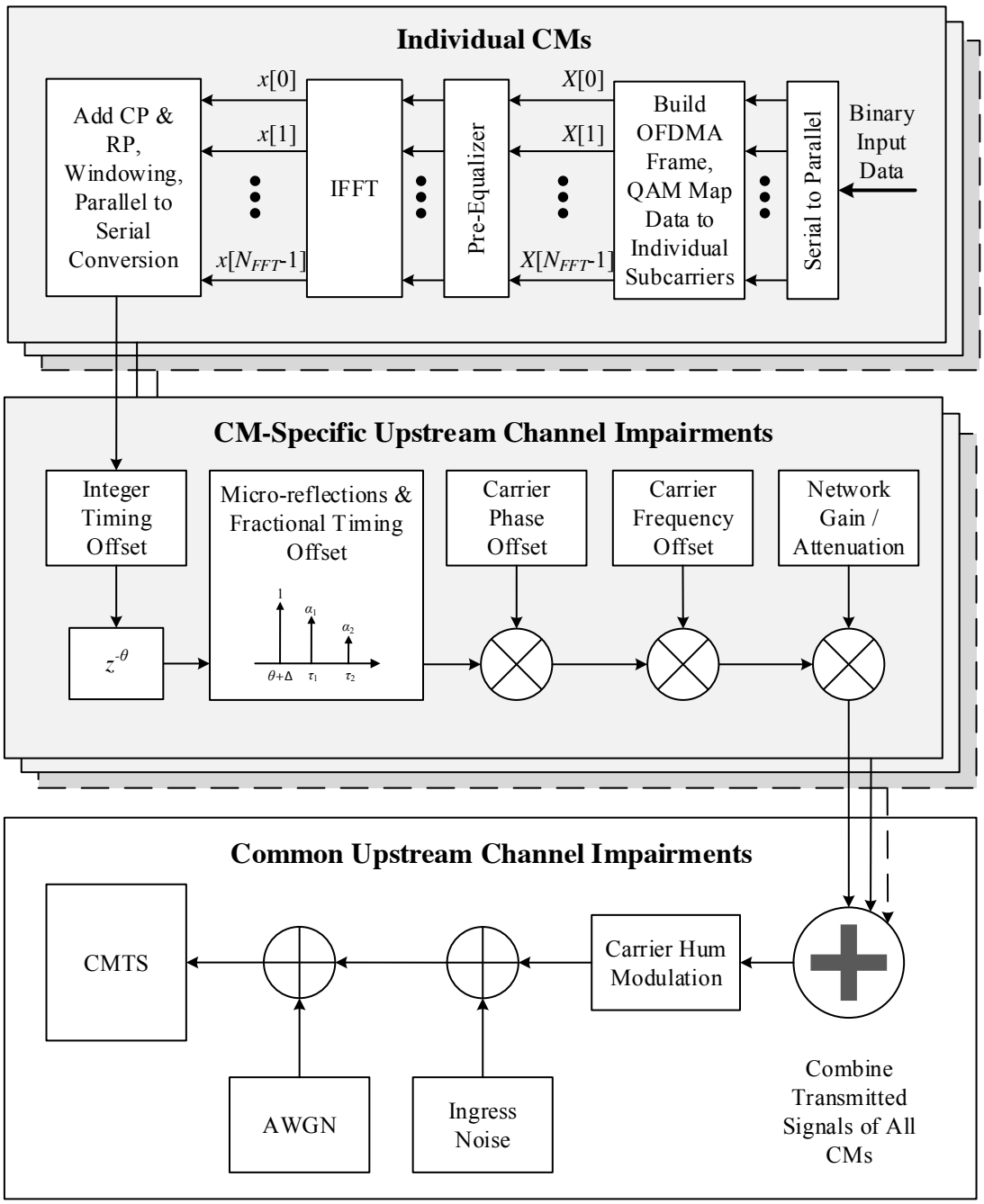


Figure 1.2: Transmitter and Channel System Block Diagram

ceived signal as a whole, such as carrier hum modulation, AM and FM ingress noise, and thermal noise, or additive white Gaussian noise (AWGN). All of the aforementioned impairments are depicted in Figure 1.2.

In the future, it may be desirable to migrate the MATLAB simulation of the CM and

upstream channel to a field-programmable gate array (FPGA). The FPGA implementation becomes a low-cost alternative to a commercial RF emulator. A secondary objective of this research is to provide a preliminary cost analysis in terms of the number of multipliers and the bits of memory required to implement some of the main components of the MATLAB model on an FPGA.

1.5 Thesis Outline

The remainder of the thesis is organized as follows. Chapter 2 discusses the theoretical aspects of upstream transmission over a DOCSIS 3.1 network. The first half of the chapter describes the functionality of DOCSIS 3.1 CMs and the methods that they employ to generate OFDMA frames. The chapter then segues into a discussion surrounding the theory of an upstream cable channel and how it can be modelled in digital baseband. As the chapter progresses, a discrete-time representation of the channel and its impairments is developed. The final product of the chapter is a discrete-time equation that models all of the CM-specific and common channel impairments listed in the problem statement.

Having established a firm theoretical background, Chapter 3 moves on to describe the implementation and verification techniques that were used to simulate the aforementioned DOCSIS 3.1 CM functionality and upstream channel impairments in MATLAB. Particular attention is given to micro-reflections, which are simulated using fractional delay filters.

Since it may be desirable to implement the MATLAB model on an FPGA in the future, Chapter 4 explores several methods for designing cost-effective fractional delay filters. Three filter designs are ultimately selected and evaluated based on their passband mean squared error (MSE) performance, as well as their cost in terms of multipliers and bits of memory.

The thesis concludes with Chapter 5, which summarizes the contributions made by this research.

2. Background

The purpose of this chapter is to provide the necessary background required to understand the various cable modem (CM) functions and channel impairments included in the MATLAB model. The chapter is separated into two parts. The first part covers the operations of the CM. It begins with a discussion of the CM's transmission modes and their respective orthogonal frequency-division multiple access (OFDMA) frame structures. Pre-equalization is then explored, and is followed by a discussion of the unique equations used by the data-over-cable service interface specification (DOCSIS) 3.1 standard to perform an inverse fast Fourier transform (IFFT). Finally, the cyclic-prefix, roll-off, and windowing methods used by the standard are examined.

The second portion of the chapter explores the theory behind the two categories of channel impairments described in the problem statement. It begins with a discussion concerning some critical assumptions made about the channel model, followed by the definition of a complex baseband equivalent signal. Next, CM-specific impairments are discussed, including integer and fractional timing offsets, micro-reflections, fractional carrier frequency offset and network gain/attenuation. Lastly, common impairments are discussed, including carrier hum modulation, AM/FM ingress noise, and additive white Gaussian noise (AWGN). The end result is an equation that combines both categories of impairments, thus demonstrating how the channel itself can be digitally modelled in baseband.

2.1 CM Operations

2.1.1 General OFDMA Frame Structure Parameters

In order for a CM to meet the DOCSIS 3.1 transmission criteria, it must be able to transform and restructure incoming data so that it conforms to one of the many OFDMA frame structures outlined in the DOCSIS 3.1 standard. The frame structures themselves are determined by the transmission mode of the CM. There are two main transmission modes: "traffic' mode," in which the CM transmits user data, and "ranging' mode," where the CM transmits frames specifically designed to obtain information about the channel. Ranging can be further be subdivided into three separate modes: initial ranging, fine ranging, and probing, each of which is used to perform a different channel estimation task.

The media access controller (MAC), whose function is to control the physical layer, controls switching between these modes. The switching is based on channel information that the cable modem termination system (CMTS) obtains periodically via ranging. This model, however, is designed to give the user full control over both the transmission modes and frame structures outlined in the standard. As such, the intricacies of the MAC are beyond the scope of this research.

Before delving into the specifics of OFDMA frame structures, it is important to establish the definition of an OFDMA frame. The DOCSIS 3.1 standard defines an OFDMA frame as a contiguous sequence of OFDMA symbols aligned in the time domain, as depicted in Figure 2.1. The number of OFDMA symbols in a frame is variable and denoted as K. Each OFDMA symbol is defined as a sequence of quadrature amplitude modulation (QAM)-mapped symbols carried by N_{FFT} subcarriers, N_{FFT} being the size of the IFFT.

Thus, the first parameter that is specified when setting up an OFDMA frame structure is the IFFT size. DOCSIS 3.1 allows an IFFT to modulate either 2048 subcarriers with 50 kHz spacing or 4096 subcarriers with 25 kHz spacing, which are referred to in this model as "2k" and "4k" modes, respectively. Following the IFFT size selection, a specific number of subcarriers are assigned to each CM. Any subcarriers that are not assigned to a CM are labelled as either unused or excluded, the later of which require that no data be transmitted

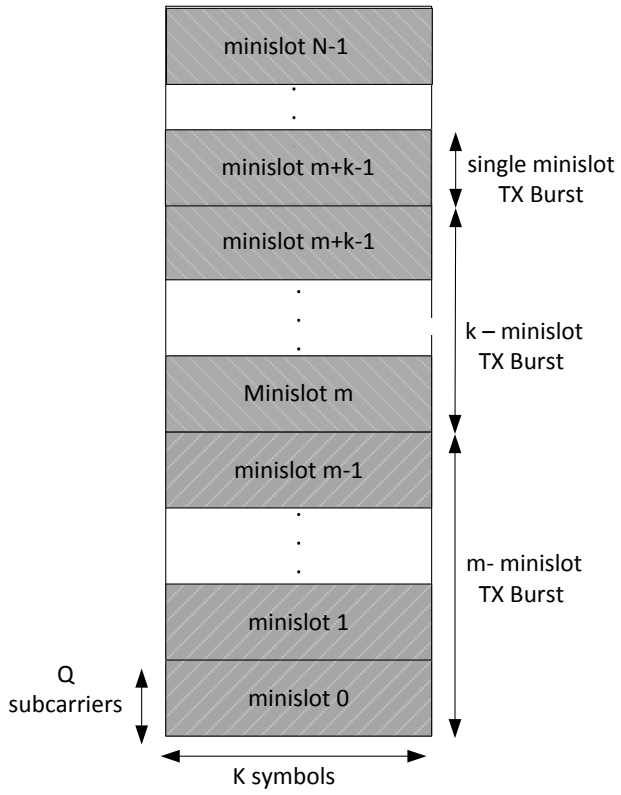


Figure 2.1: General OFDMA Frame Structure with Minislots [12]

over them. The first and last 74 subcarriers in 2k mode, as well as the first and last 148 subcarriers in 4k mode, are always excluded in order to provide guard bands at the beginning and end of the spectrum.

Subcarriers that are assigned to a specific CM are grouped into minislots. A minislot is the data carried by Q contiguous subcarriers distributed across K OFDMA symbols in a single frame. The value of Q is either 8 subcarriers in 2k mode or 16 subcarriers in 4k mode. The value of K can range from 6 to 36 OFDMA symbols in 2k mode, or from 6 to 18 OFDMA symbols in 4k mode. Refer to Figure 2.1 for a visual representation.

Each OFDMA symbol can contain a mixture of QAM-mapped data symbols, pilot symbols and/or complementary pilots depending on the transmission mode selected. It should be noted that for the purposes of this model, only binary phase-shift keying (BPSK) and square QAM constellations up to 4096-QAM are considered (BPSK being used only for pilots

and complementary pilots).

DOCSIS 3.1 pilot symbols are generated using a linear feedback shift-register (LFSR) with generator polynomial $X^{12} + X^9 + X^8 + X^5 + 1$ and a seed value of 12'd3071. This sequence is periodic with a period of $2^{12} - 1$. These symbols are then mapped using BPSK.

Complementary pilots are slightly different than pilots. They are modulated by data using a reduced order, specifically $2^{P\text{-}4}$ QAM where P is the constellation order of the data carrying subcarriers. For example, a complementary pilot symbol for a minislot utilizing 4096-QAM would be mapped using 256-QAM. Complementary pilots for any QAM constellation with an order less than or equal to 16-QAM uses BPSK.

Forward error correction (FEC) encoding is beyond the scope of this research. However, it should be noted that low-density parity check (LDPC) FEC encoding is typically performed on all data symbols in a frame following the assembly of its structure (except in the case of probing).

With a general understanding of the parameters used to generate OFDMA frames now established, the next few subsections will proceed to discuss specific frame structures used in the CM's transmission modes.

2.1.2 Traffic Mode

Once the system has been initialized, traffic mode is used to transfer data from the CM to the CMTS. The CMTS controls the upstream transmission by allocating a specific number of minislots to each transmitting CM. The data, which is QAM-mapped to data-carrying subcarriers, the pilots and the complementary pilots are distributed throughout the minislot (i.e. throughout the matrix of Q subcarriers by K OFDMA symbols) according to one of 14 different pilot pattern mappings. The pilot pattern is specified by the CMTS as part of the grant. Patterns 1-7 apply to 2k mode, while patterns 8-14 are used in 4k mode. The patterns also vary depending on whether the minislot is an "edge" or "body" minislot. A minislot is defined by the DOCSIS 3.1 standard as an edge minislot if it is:

(a) The first minislot in a transmission burst,

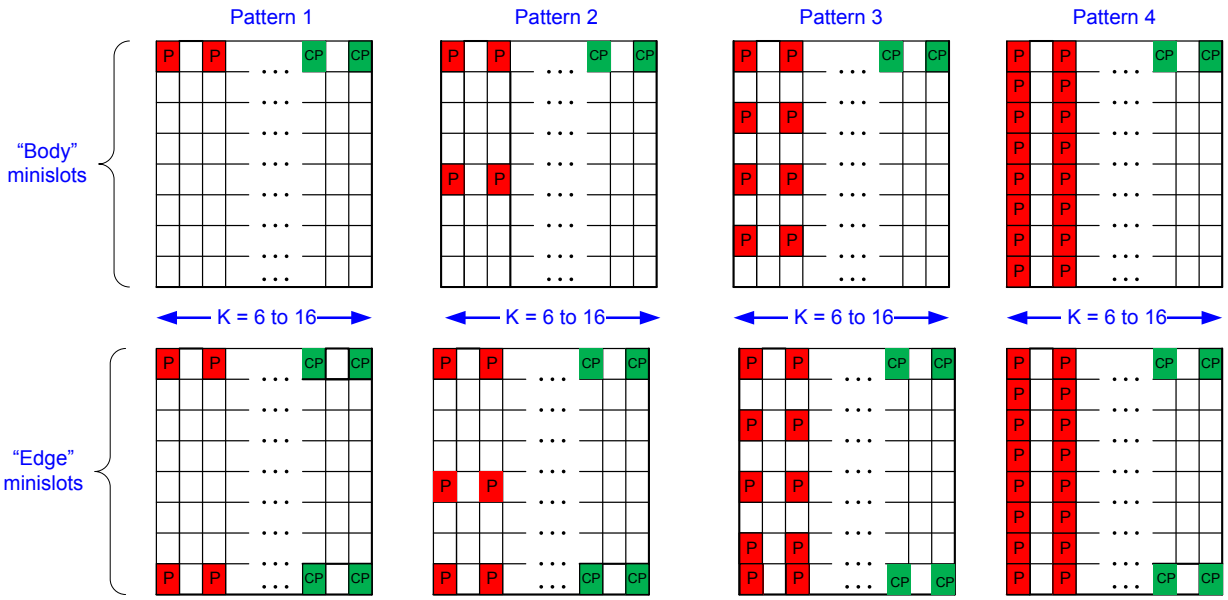


Figure 2.2: Pilot Patterns 1-4, 2k Mode [12]

(b) The first minislot of an OFDMA frame that is not zero-valued, or

(c) The first minislot after an exclusion band or after one or more contiguous skipped subcarriers or after a zero valued minislot.

All other minislots assigned to CMs are defined as body minislots. A sample pilot pattern is illustrated in Figure 2.2, with the white, red, and green squares representing data, pilots, and complementary pilots, respectively.

Interleaving is typically performed in tandem with symbol mapping. However, as with FEC encoding, interleaving is outside the scope of this research.

2.1.3 Ranging Mode

To correct for timing and power offsets introduced by the transport delay and attenuation throughout the channel, a CM is "ranged" in three stages: initial ranging, admission, and steady state. The progress is illustrated in Figure 2.3.

After a CM is installed or even just reset it must take action to join the system. A CM

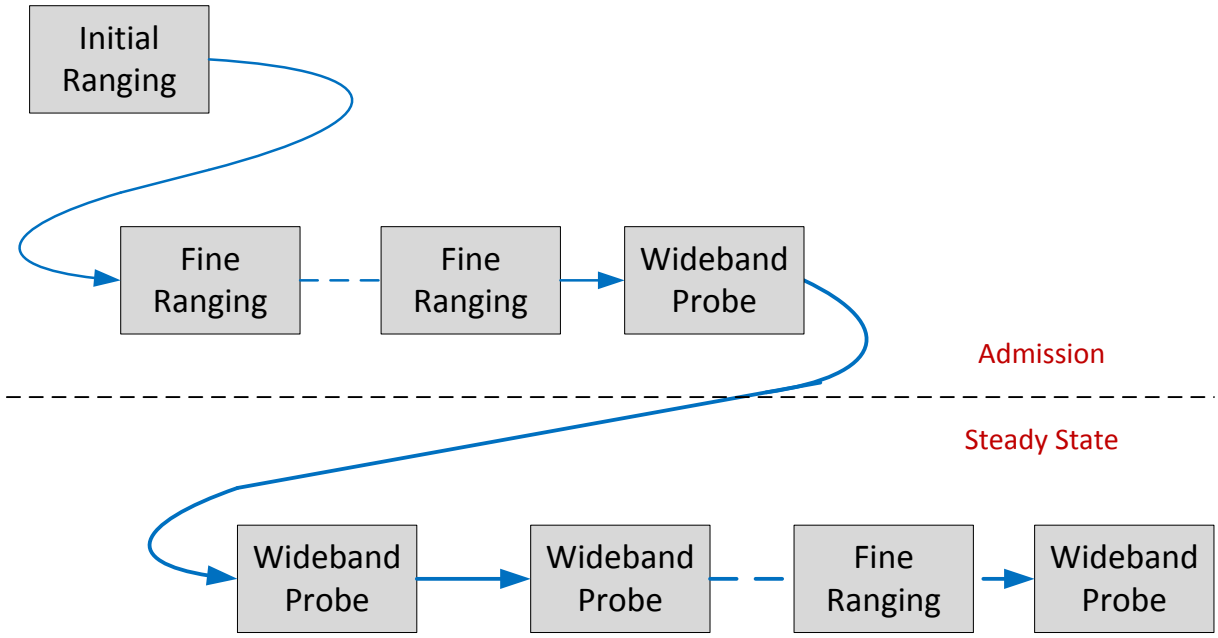


Figure 2.3: Ranging Stages of a DOCSIS 3.1 System [12]

seeking to join the system must first undergo initial ranging. During this stage, coarse timing estimation is performed and the transmit signal power is set. Upon the completion of initial ranging, the CMs are eligible to be admitted to the system. The admission process begins with fine ranging, which is used to fine tune the timing and power. The last step of admission is probing, which is used to further refine the signal power and time. Finally, having been admitted to the system, the CMs enter steady state; however, fine ranging and probing are performed periodically to track changes in power and transport delay that happen over time.

For the purposes of this research, it is assumed that all transmitting CMs have already been admitted to the system, the reasoning for which will be discussed in Section 2.2. Therefore, only the fine ranging and probing steps are included in the model of the transmitter.

2.1.3.1 Fine Ranging

A fine ranging signal consists of M minislots containing N_{fr} active subcarriers and N_{gb} null subcarriers. The null subcarriers form a guard band from a BPSK-mapped preamble sequence known to the receiver, which is distributed across the N_{fr} active subcarriers and

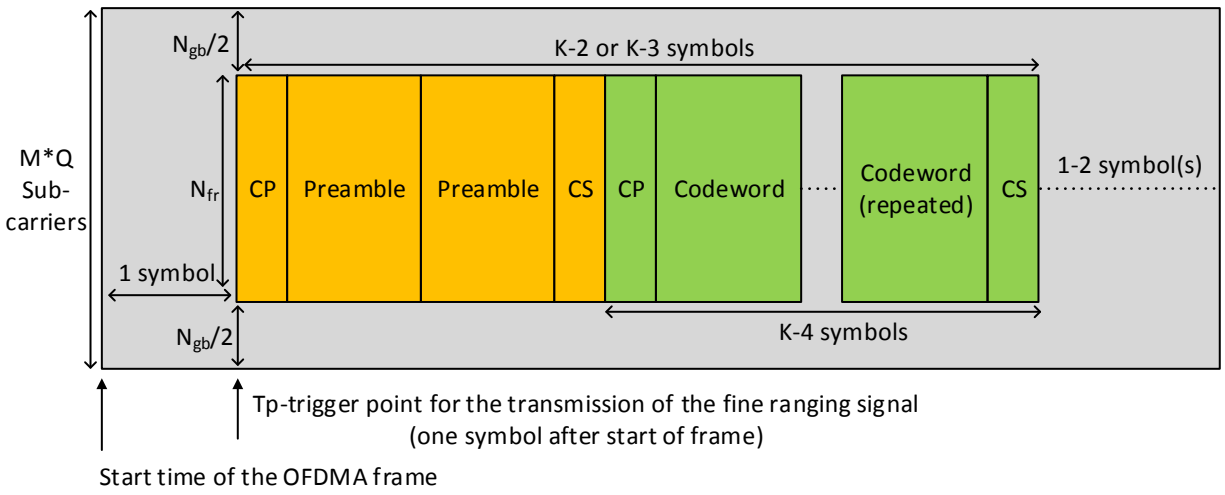


Figure 2.4: Fine Ranging OFDMA Frame Structure [12]

duplicated in a second OFDMA symbol. This preamble is then followed by a quadrature phase-shift keying (QPSK)-mapped data sequence, also duplicated in a second symbol as depicted in Figure 2.4. The beginning of each fine ranging signal is padded with one empty OFDMA symbol. The end is also padded with an empty symbol, either one or two depending on whether K is an even or odd number, respectively.

It is important to note that the fine ranging signal occupies one or more minislots (contiguous in frequency) in one frame. However, the cyclic prefix and roll-off period parameters for the OFDMA symbols in a frame are different for a fine-ranging signal. Instead of following the standard cyclic prefix / roll-off period / windowing method, which will be described in Section 2.1.6, a cyclic prefix is added to the beginning of the first symbol. The second symbol is then placed immediately after, followed by a cyclic suffix of length $N_{RCP} = N_{RP} + N_{CP}$. The structured is illustrated in Figure 2.5, which is taken from the DOCSIS 3.1 Physical Layer Specification document.

Upon receiving a fine ranging signal, the CMTS estimates the timing and power offsets, which it then sends downstream back to the CM in another specialized frame[1].

[1]For the purposes of this research, the downstream transmission is assumed to have zero delay, the ramifications of which will be discussed in Chapter 3.

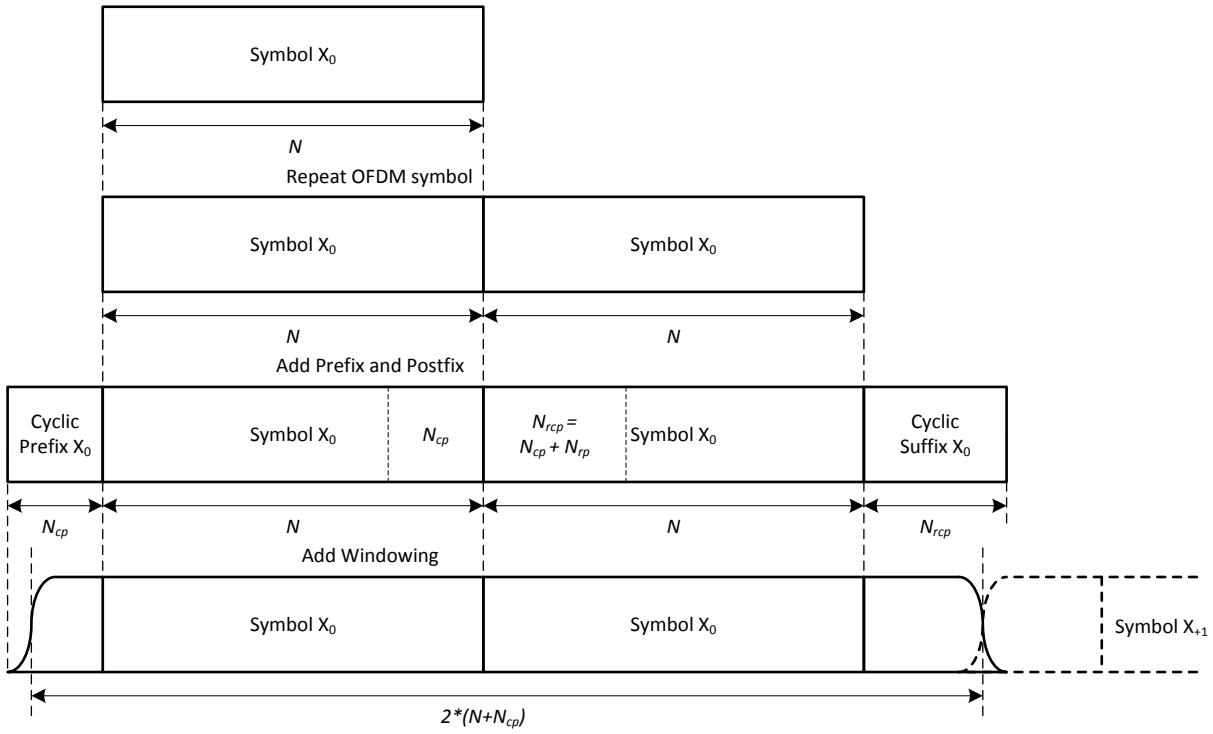
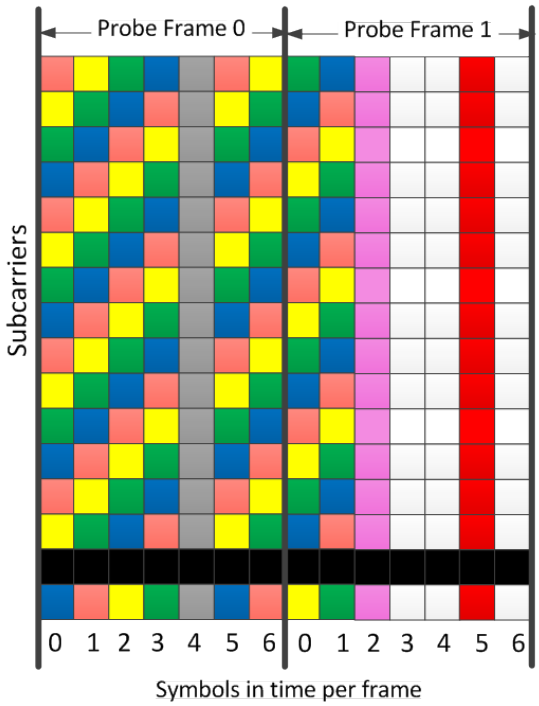


Figure 2.5: Fine Ranging OFDMA Symbol Structure [12]

2.1.3.2 Probing

Probing is accomplished through the use of dedicated "probing frames." These frames have the same length as regular frames, but are not organized into minislots. Also the OFDMA symbols contained in probing frames consist entirely of pilot subcarriers. One probing fame can be used to probe multiple CMs. Each CM is assigned a different "Start Subcarrier" and "Subcarrier Skipping" value between 0 and 7. The Subcarrier Skipping value determines how many subcarriers are skipped, i.e. not used by the CM, beginning with the Start Subcarrier. For example, if the Start Subcarrier value for one of the CMs being probed was 0, and the Subcarrier Skipping value was 4, the CM would use every 5th subcarrier in either 2k or 4k mode beginning with subcarrier 0. Using the largest possible skipping factor, which is 7, allows for 8 CMs to be probed with a single OFDMA symbol.

Each CM is also designated as either staggering or non-staggering. A non-staggering CM probes the channel in only one OFDMA symbol using the method described



SID	RSVD	Stagger	Probe Frame	Symbol In Frame	Start Subc	Subc Skip
Blue	0	1	0	0	0	3
Green	0	1	0	0	1	3
Yellow	0	1	0	0	2	3
Salmon	0	1	0	0	3	3
Med gray	0	0	0	4	0	0
Blue	0	1	0	5	0	3
Green	0	1	0	5	1	3
Yellow	0	1	0	5	2	3
Salmon	0	1	0	5	3	3
Purple	0	0	1	2	0	0
Red	0	0	1	5	0	0

Figure 2.6: DOCSIS 3.1 Probing Frame Structure Example [15]

in the previous paragraph. Should a CM be designated as staggering, it will use Subcarrier Skipping + 1 OFDMA symbols to probe the channel. The Start Subcarrier value is increased by 1 for each of the symbols used in the probe. Of course the same Subcarrier Skipping factor is used in all the Subcarrier Skipping + 1 OFDMA symbols as shown in Figure 2.6.

Note that Figure 2.6 illustrates staggering across multiple OFDMA frames. DOCSIS 3.1 specifies this as an optional probing feature; thus, only staggering within a single OFDMA frame is considered for the purposes of this research.

2.1.4 Pre-Equalization

DOCSIS 3.1 CMs have a pre-equalizer that compensates for the frequency response of the channel. The pre-equalizer is a set of multipliers that multiplies the inputs to the IFFT by a unique complex coefficient. Each coefficient is the additive inverse of the channel frequency response at the frequency of the corresponding subcarrier. This additive inverse is calculated by the equalizer in the CMTS upon receiving a probing frame. Essentially, the

pre-equalization coefficients form the frequency response of an inverted channel that negates the effects of the fractional timing offset, micro-reflections, and network gain/attenuation.

Naturally, the reverse operation can be also performed in order to simulate channel impairments. That is to say, a set of coefficients can be calculated and pre-loaded into the pre-equalizer in order to simulate the fractional timing offset and micro-reflections. The methodology behind this will be discussed in Sections 2.2.2.1 and 2.2.2.2, respectively.

2.1.5 Inverse Fast Fourier Transform

Following pre-equalization, each subcarrier in the OFDMA symbol is modulated using an inverse discrete Fourier transform (IDFT). It is well known that the inverse fast Fourier transform (IFFT) algorithm is an efficient way to compute the IDFT, provided the size of the IDFT is a power of 2. Since the size of the IDFT in DOCSIS 3.1 is a power of 2, the remainder of this thesis will refer to the IDFT as an IFFT.

A standard IFFT is defined by:

$$x[n] = \text{IFFT}\left\{X[k]\right\} = \frac{1}{N_{FFT}} \sum_{k=0}^{N_{FFT}-1} X[k] e^{j\frac{2\pi nk}{N_{FFT}}}, \quad 0 \leq n \leq N_{FFT} - 1 \tag{2.1}$$

where N_{FFT} is the size of the IFFT. DOCSIS 3.1, however, uses a modified version of the IFFT, which is:

$$x[n] = \frac{1}{\sqrt{N_{FFT}}} \sum_{k=0}^{N_{FFT}-1} X[k] e^{j\frac{2\pi n(k-N_{FFT}/2)}{N_{FFT}}}, \quad 0 \leq n \leq N_{FFT} - 1 \tag{2.2}$$

Clearly, the two main differences between the standard and modified IFFT equations are the phase shift of $-N_{FFT}/2$ cycles/sample and the scaling factor of $\sqrt{N_{FFT}}$.

Note that DOCSIS 3.1 also uses a modified version of the standard DFT/FFT formula; however, the FFT is part of the CMTS functionality, so its discussion is omitted.

When implementing a modified IFFT in a DOCSIS 3.1 system, it may be convenient to use an existing standard IFFT module and apply the phase shift and scaling factor separately. There are two methods that can be used to implement the phase shift. The first method is very straightforward, and involves multiplying the IFFT output by $e^{-j\pi n}$, as depicted in

Equation 2.3:

$$x[n] = \sqrt{N_{FFT}} \cdot \text{IFFT}\left\{X[k]\right\} \cdot e^{-j\pi n}, \quad 0 \leq n \leq N_{FFT} - 1 \tag{2.3}$$

where IFFT$\{X[k]\}$ denotes a pre-existing IFFT module. This means that every odd output will have its sign reversed, as $e^{-j\pi} = -1$ for n odd.

The second method involves changing the order in which the data from each subcarrier is passed to the IFFT function. The standard practice is to pass the data from subcarrier 0 first, and data from subcarrier $N_{FFT} - 1$ last. However, if the data from subcarriers $N_{FFT}/2$ to $N_{FFT} - 1$ is passed in first, and is followed by the data from subcarriers 0 to $N_{FFT}/2 - 1$, the resulting output sequence will be shifted in phase by the appropriate amount. Ordering the inputs in such a fashion is sometimes referred to as constructing a "DC-centred" sequence [16].

Accounting for the $\sqrt{N}$ scaling factor is another straightforward technique, requiring only a single multiplication. Technically, from a mathematical standpoint the scaling could be performed at either the input or the output with the same result; however, from a hardware perspective it is best to scale at the output to help reduce rounding or truncation errors in the IFFT module.

2.1.6 Cyclic Prefix, Roll-off Period, and Windowing

Once the IFFT output has been appropriately scaled, three actions are performed on the resulting time-domain sequence to eliminate inter-symbol interference (ISI) and curtail out-of-band emissions. The first is the addition of a cyclic prefix, which consists of N_{CP} samples copied from the end of the OFDMA symbol. The second action is the addition of a roll-off period suffix, which consists of N_{RP} samples copied from the beginning of the OFDMA symbol (prior to inserting the cyclic prefix). The values for N_{RP} and N_{CP} can be found in Table A.1 in Appendix A. DOCSIS 3.1 requires that the value of N_{RP} be less than N_{CP} in all instances.

The third and final action performed prior to transmission is windowing each OFDMA symbol. DOCSIS 3.1 specifies a Tukey raised-cosine (RC) window, which it applies over the

first and last N_{RP} data symbols (including the cyclic prefix and roll-off). The application of the window used in the time-domain greatly curtails the spectral leakage into adjacent frequency bands. The OFDMA symbol is then transmitted, the first N_{RP} symbols being overlapped with the last N_{RP} symbols of the previous frame. Following this final action, the signal would normally be passed to an upconverter to transmit the signal in the radio-frequency (RF) band; however, for reasons discussed in the next section, the signal can be left at baseband and this step can be bypassed. Thus, the baseband signal is passed directly to the digital baseband channel that models the RF channel.

2.2 Channel Modelling

2.2.1 Methodology

In addition to modelling a CM transmitter, this research creates a digital baseband model of a DOCSIS 3.1 cable network. Before delving into specifics, an important assumption about the channel must be discussed. In any cable network, the conditions of the channel can change due to a wide variety of environmental and mechanical factors. However, the changes experienced by the system are usually gradual, and are thus relatively constant over short periods of time. For example, a cable will likely perform differently during mid-day than it does during midnight due to temperature differences. Since the channel changes very slowly relative to the duration of an OFDMA frame, it is assumed to be static (i.e. time-invariant).

This first assumption must be clarified. Section 2.1.3 mentioned that any CMs transmitting over the channel are assumed to be in steady state, where timing and power offsets would have been corrected. However, CMs are in steady state long enough for gradual environmental change, such as a change in temperature, to cause the channel conditions to change a little. In other words, it must be assumed that the channel has developed some offsets over a fixed time period, but these offsets will not change during the course of the simulation. Ultimately, these two assumptions allow the model's simulation to operate on a frame-by-frame basis, a topic which is discussed in greater detail in Chapter 3.

Another important item to note before moving forward is that analogue signals sent upstream over a physical DOCSIS 3.1 network are transmitted in a passband of bandwidth B and centre frequency f_c. The signal bounds $[f_c - B/2, f_c + B/2]$ must be set so that the total signal bandwidth (maximum 102.4 MHz) lies within the upstream bounds of 5 MHz and 204 MHz. The CMs and CMTS, however, perform all of their signal processing digitally using complex baseband signals. Thus, for a CM to transmit a signal over a physical network, it requires an upconverter to shift the signal from baseband to passband. Similarly, the CMTS requires a downconverter to shift the passband signal back to baseband. Although both of these converters could be implemented digitally and the channel can be modelled in passband, a more practical approach is to consider a complex baseband equivalent of the channel and eliminate the up/down conversions.

To understand how the channel can be modelled using a complex baseband equivalent, the relationship between a passband signal and its complex baseband equivalent must first be established. According to [17], this relationship is given as follows:

$$s(t) = \mathrm{Re}\left\{ s_b(t) e^{j2\pi f_c t} \right\} = \frac{1}{2} s_b(t) e^{j2\pi f_c t} + \frac{1}{2} s_b^*(t) e^{-j2\pi f_c t} \tag{2.4}$$

where $s(t)$ denotes the passband signal and $s_b(t)$ denotes the baseband signal. Thus, a passband signal can be represented as the sum of two baseband signals that are complex conjugates of each other, one of which modulates a carrier frequency of f_c, the other of which modulates a carrier of $-f_c$. With this information in mind, the problem of creating a complex baseband equivalent channel model can now be addressed on an individual impairment basis.

2.2.2 CM-Specific Impairments

2.2.2.1 Timing Offset

The first impairment that will be modelled is the timing offset at the CMTS, which is necessary for testing timing recovery algorithms used in the ranging modes. Before going into detail about the timing offset itself, the concepts of a time delayed signal and a discrete-time model are addressed.

Signals transmitted from a DOCSIS 3.1 CM over an HFC network are subject to transport

delay that depend both on the distance between the CM and CMTS and the physical plant. Since an HFC time-invariant system is considered, the delay is a fixed value, denoted τ, which has units of seconds. Thus, a lossless echo-free channel is described by:

$$y_1(t) = \hat{h}_1(t) * x(t) = x(t - \tau) \tag{2.5}$$

where $x(t)$ is the band-limited transmitted RF signal, $y_1(t)$ is the received RF signal, and the '$*$' operator denotes convolution between $x(t)$ and the RF channel impulse response (CIR), $\hat{h}_1(t)$, which is in turn given by:

$$\hat{h}_1(t) = \delta(t - \tau) \tag{2.6}$$

To represent the channel in baseband, let $x_b(t)$ and $y_b(t)$ be the complex baseband equivalents of $x(t)$ and $y_1(t)$, respectively. Then $x(t) = \mathrm{Re}\left\{x_b(t)e^{j2\pi f_c t}\right\}$ and $y_1(t) = \mathrm{Re}\left\{y_{1b}(t)e^{j2\pi f_c t}\right\}$, where f_c is the frequency of the RF channel. Substituting these into Equation 2.5 yields:

$$\mathrm{Re}\left\{y_{1b}(t)e^{j2\pi f_c t}\right\} = \hat{h}_1(t) * \mathrm{Re}\left\{x_b(t)e^{j2\pi f_c t}\right\} \tag{2.7a}$$

$$= \mathrm{Re}\left\{x_b(t - \tau)e^{j2\pi f_c (t-\tau)}\right\} \tag{2.7b}$$

$$= \frac{1}{2}x_b(t - \tau)e^{-j2\pi f_c \tau}e^{j2\pi f_c t} + \frac{1}{2}x_b^*(t - \tau)e^{j2\pi f_c \tau}e^{-j2\pi f_c t} \tag{2.7c}$$

Equation 2.7b shows that the carrier frequency of the received signal is phase shifted by an amount proportional to the transport delay. Thus, the complex baseband equivalent of a single path (i.e. echo free) lossless channel with a fixed transport delay is given by:

$$y_{1b}(t) = \hat{h}_1(t) * x_b(t) = e^{-j2\pi f_c \tau}x_b(t - \tau) \tag{2.8}$$

Discrete-Time Model

The focus of this research is on constructing a digital model for the channel, which means a discrete-time description is needed. Converting the continuous-time channel of Equation 2.8 to discrete-time requires the application of sampling theory.

Let the digital output of the CM be the complex digital data sequence $x[m]$. Hypothetically speaking, each element of this discrete sequence corresponds to a specific voltage level.

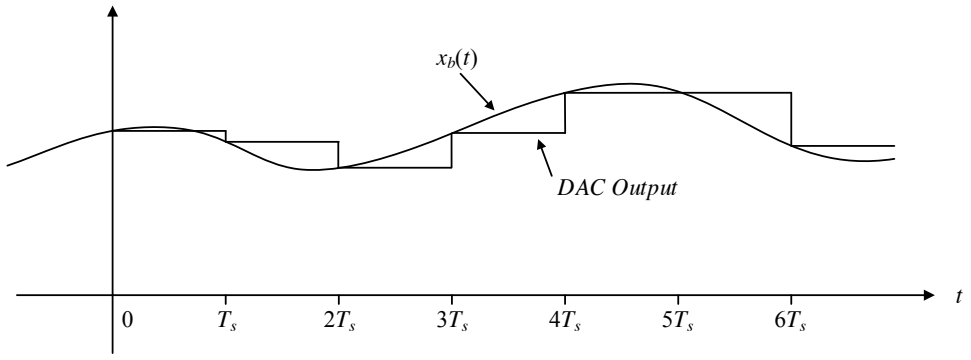


Figure 2.7: Sample and Hold DAC Output vs. Reconstructed Signal

To obtain the analogue signal $x_b(t)$, the samples of $x[m]$ must be uniformly spaced apart at intervals of time T_s such that $T_s \geq 1/B$, where B is the bandwidth of the signal (i.e. $x_b(mT_s) \equiv x[m]$). Setting T_s to this value ensures that the Nyquist criteria is met, thus eliminating aliasing.

Normally at this point, the signal $x_b(mT_s)$ is sent to a digital-to-analogue converter (DAC). The DAC "holds" the current value of $x_b(mT_s)$ until the next sample arrives, producing an output similar to the one shown in Figure 2.7. As a consequence of this "sample-and-hold" process, the frequencies of the resulting DAC output signal are replicated periodically about integer multiples of 2π, thus producing an infinite number of "mirror images".

To "remove" (i.e. sufficiently suppress) the mirror images and band-limit $x_b(t)$, a low-pass "reconstruction filter" must be applied to the DAC output. The reconstruction filter interpolates between the samples of the DAC output, resulting in a smoother analogue signal. Theoretically, the optimal reconstruction filter is a "brick-wall" filter, which has the following frequency response:

$$H_c(e^{j\omega}) = \begin{cases} 1 & |\omega| < B/2 \\ 0 & |\omega| > B/2 \end{cases} \tag{2.9}$$

Taking the inverse Fourier transform of the brick-wall filter's frequency response produces its time-domain equivalent, which is a sinc function:

$$\text{sinc}(t) = \frac{\sin(\pi t)}{\pi t} \tag{2.10}$$

Thus, the analogue baseband signal $x_b(t)$, band-limited to $B/2$, can be represented as:

$$x_b(t) = \sum_m x[m] \operatorname{sinc}\left(\frac{t - mT_s}{T_s}\right) \tag{2.11}$$

Note that the ideal *sinc* function has an infinite impulse response, which makes it unsuitable for implementation in a practical setting. However, its use will suffice for the purpose of a theoretical explanation.

Substituting the discrete-time representation of $x_b(t)$ into Equation 2.8 yields:

$$y_{1b}(t) = \sum_m x[m] e^{-j\frac{2\pi F_c \tau}{F_s}} \operatorname{sinc}\left(\frac{t - (\tau + m)T_s}{T_s}\right) \tag{2.12}$$

where the transport delay τ is now in units of samples, F_c is the carrier frequency in Hz, and F_s is the sampling frequency in cycles/sample.

To obtain a complete discrete-time description of the channel, $y_{1b}(t)$ must be sampled by an analogue-to-digital converter (ADC). Before this sampling occurs, $y_{1b}(t)$ is typically passed through a low-pass "anti-aliasing filter," which, for the purposes of this research, is assumed to be identical to the reconstruction filter described earlier (i.e. a sinc function). The anti-aliasing filter eliminates any high-frequency components introduced by noise in the channel, thus band-limiting $y_{1b}(t)$ to $B/2$. The ADC is then used to sample the band-limited signal such that $y_{1b}(nT_s) \equiv y_1[n]$, where $y_1[n]$ is the discrete-time signal processed by the receiver. Here the variable n is used instead of m in order to differentiate between the samples of the CMTS and the CM. Equation 2.12 therefore becomes:

$$y_1[n] = \sum_m x[m] e^{-j\frac{2\pi F_c \tau}{F_s}} \operatorname{sinc}(n - m - \tau) \tag{2.13}$$

The above equation can be simplified using the method described in [17]. First, let $k = n - m$, such that:

$$y_1[n] = \sum_k x[n - k] e^{-j\frac{2\pi F_c \tau}{F_s}} \operatorname{sinc}(k - \tau) \tag{2.14}$$

Next, define the channel impulse response to be:

$$\hat{h}_1[k] = e^{-j\frac{2\pi F_c \tau}{F_s}} \operatorname{sinc}(k - \tau) \tag{2.15}$$

Substituting Equation 2.15 into Equation 2.14 then yields:

$$y_1[n] = \sum_k \hat{h}_1[k]x[n-k] \tag{2.16}$$

In this simplified equation, $\hat{h}_1[k]$ represents the k^{th} complex channel filter tap at time n. Each tap can be thought of as samples of the low-pass filtered baseband channel impulse response $\hat{h}_1(t)$ convolved with the function $\text{sinc}(t/T_s)$ [17].

Now that the fundamentals of a time-delayed channel have been addressed and a discrete-time representation has been established, the focus can be shifted towards the timing offset. The next two subsections provide a detailed description of the integer and fractional components of the timing offset, as well as the techniques used to model both components.

Integer Timing Offset

The CMTS recovers OFDMA symbols by applying an FFT "window" of size N_{FFT} to the received signal. This process is depicted in Figure 2.8. In a perfectly time-synchronized system, the incoming frame would be sampled at exactly the same points as the samples at the output of the IFFT in the CM. However, when a CM first joins the network, the transport delay of the channel is unknown to the CMTS. As a result, the sampling reference is lost. This causes two problems:

1. the sampling points in the CMTS do not line up with the samples at the output of the IFFT in the CM, and

2. the CMTS does not know where to place the FFT window.

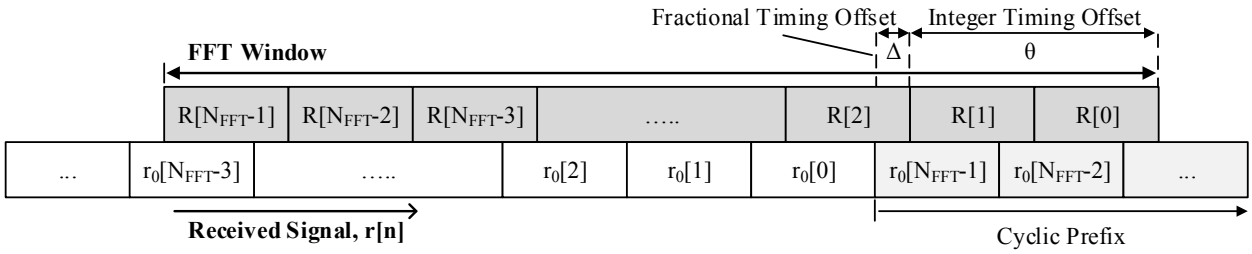


Figure 2.8: Timing Offset of a Received OFDMA Frame

The integer timing offset is defined as the integer portion of time in units of samples between the first sample extracted for the FFT, i.e. the first sample in the FFT window, and the first sample of the OFDMA symbol.

As mentioned in Section 2.1.3, the integer timing offset can be estimated using ranging signals. A standard estimation technique involves running the received samples of the ranging signal through a correlation algorithm. Recall from Section 2.1.3.1 that the ranging signal contains a preamble sequence that is known to the CMTS. The correlation algorithm compares each sample of the received ranging signal against this preamble sequence until the samples line up. The integer timing offset is then calculated based on the difference between the alignment point and the first sample received. Note that if the offset lies within the interval of $-N_{CP} + N_{RP} \leq \theta \leq 0$ samples relative to the first sample of the OFDMA frame, no ISI will be introduced to the system. Any other value of θ will result in ISI [18] [19].

From the processes described above, it is clear that the transport delay and the timing offset are not the same. While the transport delay is a fixed value, the integer timing offset will grow smaller and eventually reach zero as it is iteratively corrected in a sequence of ranging frames. As such, the integer portion of the transport delay of the received signal, τ (samples), in Equation 2.15 can be replaced with the integer timing offset, henceforth denoted as θ, also in units of samples. It is important to note, however, that this substitution does not encompass the phase shift of the carrier frequency introduced by converting the signal to baseband. This shift will always be dependent upon the "cable" distance between the CM and the CMTS. Thus, Equation 2.15 can be re-written as:

$$\hat{h}_2[k] = e^{-j\frac{2\pi F_c \tau}{F_s}} \operatorname{sinc}(k - \theta - \Delta) \tag{2.17}$$

where Δ represents the remaining fractional component of the transport delay in units of samples, which will be discussed in the following subsection.

From Equation 2.17, it is clear that the integer timing offset can be simulated by simply inserting an integer number of delays into the received signal.

Fractional Timing Offset

Although the CMTS is able to estimate the integer portion of the timing offset via ranging, there will almost always exist a fractional part of the timing offset that ranging is unable to correct. As such, the fractional timing offset, henceforth referred to as Δ in units of samples, will remain constant no matter how many iterations of ranging are performed (assuming a static channel with perfect frequency synchronization). Its value will therefore be equal to the fractional portion of the transport delay, which can vary between $-0.5 \leq \Delta \leq 0.5$ samples.

Two methods for simulating the fractional timing offset were explored during this research. The first method involved placing a fractional delay filter along the main channel path. Designs for this filter will be discussed in great detail in Chapter 4. The second method involved the use of pre-equalization coefficients, as discussed in Section 2.1.4. The coefficients for a single CM can be calculated by assuming that the fractional timing offset is the only impairment present in the channel, which would cause it to have the following frequency response:

$$\hat{H}_2(e^{jw}) = e^{-j\frac{2\pi\Delta n}{N_{FFT}}}, \quad -\frac{N_{FFT}}{2} \leq n \leq \frac{N_{FFT}}{2} - 1 \tag{2.18}$$

Ultimately, pre-equalization coefficients proved to be the most effective method for simulating the fractional timing offset. The reasoning behind this choice will be explored in Chapter 4.

With a firm understanding of transport delay effects and the timing offset now established, the theory behind multipath propagation and micro-reflections can be explored.

2.2.2.2 Micro-reflections

An HFC network is comprised of a wide variety of components, including numerous splitters and bidirectional amplifiers, all of which are used to connect a large number of CMs to a CMTS. If the connector of an amplifier or any other network component along the main path becomes corroded or is improperly terminated, an impedance mismatch occurs [20].

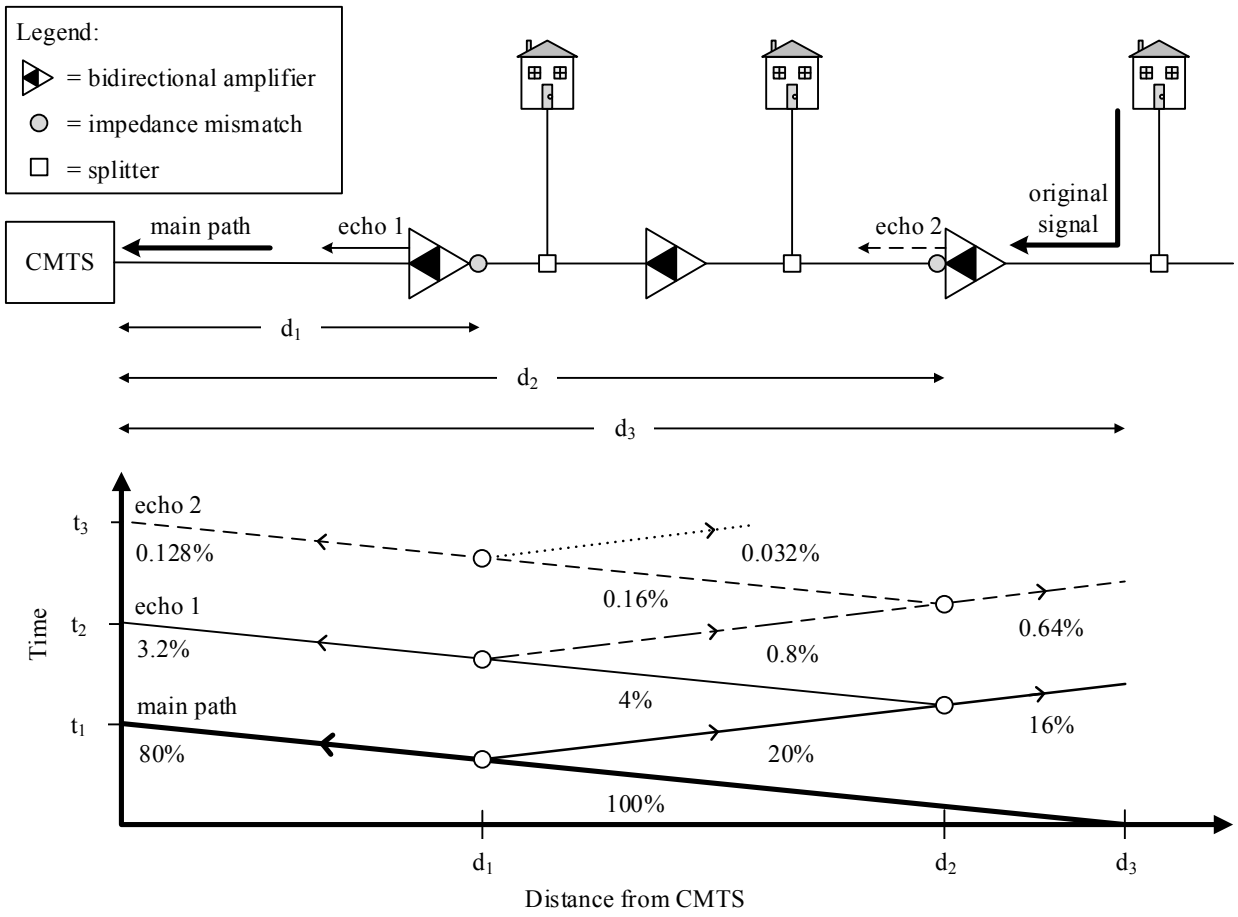


Figure 2.9: Example of Micro-reflections in an HFC Network

When a signal travelling upstream encounters this impedance mismatch, part of its energy is reflected back towards the CM. If there is another impedance mismatch in the network, the reflected signal will be re-reflected. This produces echoes within the system that are attenuated and offset in phase. Eventually, these echoes will reach the CMTS where they will add to the main signal and cause unwanted amplitude and phase distortion [21].

For a visual representation of this effect, consider the example provided in Figure 2.9. Here, two impedance mismatches are present: one at the input of an amplifier, and one at the output of a different amplifier. It is assumed that the signal is attenuated by 20% each time it is reflected by an impedance mismatch. The figure illustrates the first two echoes that occur in the system. Echo 1 has a total travel time equal to t_2 and an amplitude that is 3.2% relative to the original signal. Echo 2 has a total travel time equal to t_3 and an

Table 2.1: DOCSIS 3.1 Micro-reflection Categories [12]

Attenuation (dBc)	**Echo Time (µs)**	**Echo Time (samples)**
-16	≤ 0.5	≤ 51.2
-22	≤ 1.0	≤ 102.4
-29	≤ 1.5	≤ 153.6
-35	> 2.0	> 204.8
-42	> 3.0	> 307.2
-51	> 4.5	> 460.8

amplitude that is 0.128% relative to the original signal.

DOCSIS refers to echoes as "micro-reflections," and groups them according to both their attenuation and "echo time" (i.e. travel time/transport delay). A stronger echo corresponds to a shorter echo time, as the cable has less of an opportunity to attenuate the echo over shorter distances. Previous versions of the standard, namely DOCSIS 3.0, accounted for up to three bounded micro-reflections in separate categories. DOCSIS 3.1 specifies a better maintained plant that has at most a single bounded micro-reflection. The categories provided for this single echo are listed in Table 2.1, where dBc is deciBels relative to the carrier signal, which is assumed to have a gain of unity.

Taking micro-reflections into account, Equation 2.17 can be expanded as follows:

$$\hat{h}_3[k] = e^{-j\frac{2\pi F_c \tau}{F_s}} \left(\text{sinc}(k - \theta - \Delta) + \sum_{p=1}^{P} 10^{\frac{A_p}{20}} e^{-j\frac{2\pi F_c \tau_p}{F_s}} \text{sinc}(k - \tau_p) \right) \tag{2.19}$$

where P is the number of micro-reflections, A_p is the echo attenuation of each micro-reflection in dBc, and τ_p is the echo time of each micro-reflection in samples. To help simplify this equation, the complex attenuation experienced by each micro-reflection can be represented as a single variable α_p such that:

$$\alpha_p = 10^{\frac{A_p}{20}} e^{-j\frac{2\pi F_c \tau_p}{F_s}} \tag{2.20}$$

Equation 2.19 then becomes:

$$\hat{h}_4[k] = e^{-j\frac{2\pi F_c \tau}{F_s}} \left(\text{sinc}(k - \theta - \Delta) + \sum_{p=1}^{P} \alpha_p \, \text{sinc}(k - \tau_p) \right) \tag{2.21}$$

Thus, for the purposes of this research, micro-reflections can be modelled by adding additional paths to the channel that delay, attenuate and shift the phase of the original signal. Note that only one additional path is required for DOCSIS 3.1 modelling (i.e. $P = 1$ in Equation 2.21); however, should a user wish to model multiple micro-reflections, more paths may be added.

To simulate the echo time of micro-reflections, the same techniques discussed in Section 2.2.2.1 can be used. This involves delaying the signal by an integer to simulate the integer portion of the echo time, and using either pre-equalization coefficients or a fractional delay filter along the echo path to simulate the fractional portion. It is important to note, however, that fractional delay filters are capable of simulating the effects of micro-reflections more accurately than pre-equalization coefficients. To understand why, consider the equation used to generate micro-reflection pre-equalization coefficients:

$$\hat{H}_4(e^{jw}) = 1 + \sum_{p=1}^{P} \alpha_p e^{-j\frac{2\pi n \tau_p}{N_{FFT}}}, \quad -\frac{N_{FFT}}{2} \leq n \leq \frac{N_{FFT}}{2} - 1 \tag{2.22}$$

Similar to Equation 2.18, this equation represents the frequency response of the channel; however, this time only micro-reflections are assumed to be present. The pre-equalization coefficients once again apply a type of "phase ramp" across the spectrum, allowing them to apply their effects to each individual subcarrier as desired. The problem arises when the cyclic prefix and roll-off are added to the signal following modulation by the IFFT.

In reality, a micro-reflection contains a delayed version of the entire signal that is transmitted by a CM. However, if the micro-reflection is simulated prior to the IFFT, the samples selected for the cyclic prefix and roll-off will have already been affected by the micro-reflection at a different time. This results in a discontinuous signal, as illustrated in Figure 2.10.

Since the cyclic prefix and roll-off are removed when the CMTS begins extracting the received OFDMA symbol, the discontinuities that occur when simulating micro-reflections

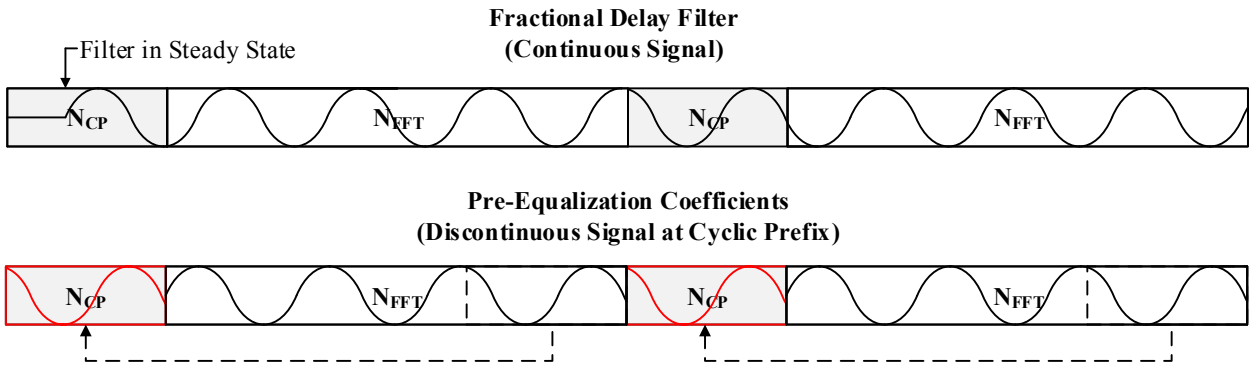


Figure 2.10: Trade-off Between Simulating Micro-reflections Using Fractional Delay Filter vs. Pre-Equalization Coefficients

using pre-equalization coefficients would technically not affect the results of the simulation. This is, however, assuming that no timing offset exists, which is not the case in this model. Therefore, the best way to simulate micro-reflections is to use add an integer delay to the main path signal followed by a fractional delay filter. This will ensure that all parts of the transmitted signal are delayed equally with no discontinuities.

2.2.2.3 Carrier Phase Offset and Carrier Frequency Offset

In a typical upstream communications system, different crystal oscillators control the upconverter and downconverter. DOCSIS 3.1 intends that both of these oscillators be phase-locked to the CMTS reference oscillator, which runs at a frequency of 10.24 MHz. However, DOCSIS 3.1 allows for oscillators to be frequency-locked but not phase-locked, which means they will not be synchronized. Thus, a carrier phase offset (CPO) and a carrier frequency offset (CFO) could be introduced. To account for these offsets, Equation 2.21 is modified as follows:

$$\hat{h}_5[k] = e^{-j\frac{2\pi n\nu}{N_{FFT}}+\phi} e^{-j\frac{2\pi F_c \tau}{F_s}} \left(\text{sinc}(k-\theta-\Delta) + \sum_{p=1}^{P} \alpha_p \,\text{sinc}(k-\tau_p) \right) \tag{2.23}$$

where ϕ is the CPO in radians and ν is the CFO in terms of subcarrier spacings. Depending on the severity of the discrepancies between oscillators, the CFO may be more than 1 subcarrier spacing; thus, it can be modelled as having integer and fractional components η and ϵ, respectively, such that $\nu = \eta + \epsilon$. The integer portion of the CFO, η, shifts the data onto other subcarriers, while the fractional portion, ϵ, produces inter-carrier interference (ICI)

between subcarriers [22].

In a DOCSIS 3.1 network, the CFO is initially estimated and corrected using downstream tracking. Specifically, the CM must lock the upstream subcarrier clock (25 kHz or 50 kHz) to the master downstream clock rate of the CMTS with an accuracy of 0.4 ppm. The individual subcarrier frequencies themselves must be accurate within ±30 Hz of the master clock rate. Should the channel conditions change over time, the CMTS can also make use of the pilots and complementary pilots sent by a CM to track changes in the CFO, thus ensuring that the CFO stays within the bounds defined above. The same pilots and complementary pilots are also used to initially estimate and correct the CPO, and can similarly be used to track its changes over time.

Given the tight frequency restrictions imposed upon the CFO, it is clear that only its fractional portion, ϵ, need be modelled in this research; thus, its integer portion, η, can be omitted, and ν becomes ϵ in equation Equation 2.23. The equation can be further simplified by absorbing the timing-offset-induced carrier phase shift term of Equation 2.23 into the CPO term, ϕ, as the net effect CPO is a rotation of the received QAM constellation points between $-\pi$ and π radians. The simplified equation is as follows:

$$\hat{h}_6[k] = e^{-j\frac{2\pi n\epsilon}{N_{FFT}}+\phi} e^{-j\frac{2\pi F_c \tau}{F_s}} \left(\text{sinc}(k-\theta-\Delta) + \sum_{p=1}^{P} \alpha_p \,\text{sinc}(k-\tau_p) \right) \tag{2.24}$$

2.2.2.4 Network Gain

The degree of attenuation experienced by a signal travelling through a DOCSIS 3.1 network is dependent upon the type of coaxial cables and amplifiers that are used. Cable manufacturers typically specify the attenuation in a coaxial cable in dB/ft or dB/m. Properly distributed amplifiers will compensate more or less for the cable attenuation; however, the amplifiers cannot track attenuation/gain changes due to environmental changes The gain/attenuation between the CM and the CMTS, which is referred to as the "network gain/attenuation," is modelled as a scaling factor, A, in dB. This yields a channel impulse

response:

$$\hat{h}_7[k] = 10^{\frac{A}{20}} e^{-j\frac{2\pi n\epsilon}{N_{FFT}}+\phi} \left(\text{sinc}(k-\theta-\Delta) + \sum_{p=1}^{P} \alpha_p \,\text{sinc}(k-\tau_p) \right) \tag{2.25}$$

Note the impulse response assumes the environmental changes are essentially constant for the duration of a frame so the attenuation/gain A is a constant. A network gain/attenuation between -9 dB and +3 dB is selected for this model, which represents the maximum range of the CMTS's input power settings outlined in the DOCSIS 3.1 standard [12].

In the digital channel model described in Figure 1.2, the network gain/attenuation could theoretically be applied anywhere between the transmitter and the combination of transmitted signals. However, the input design parameters for any required fractional delay filters can be relaxed substantially if the network gain/attenuation is applied post-filtering. This, in turn, may help reduce the quantization noise generated by truncating products in the filter, depending on the filter implementation. Thus, the network gain/attenuation in a digital model should be applied immediately prior to the combination of transmitted signals, as depicted in Figure 1.2.

Correcting for the network gain/attenuation is accomplished by adjusting the power at the output of the CM (i.e. before the signal enters the channel).

2.2.3 Common Channel Impairments

In the digital model, channel impairments common to all CMs are implemented after the transmitted signals from each CM have been combined. Of course the CM-specific impairments are implemented prior to combining the signals. However, the equations derived in Section 2.2.2 only apply to a single CM. Thus, an equation that accounts for multiple CMs must be created. This is given by:

$$y[n] = \sum_{i=1}^{N_{CM}} \sum_{k} 10^{\frac{A_i}{20}} e^{-j\frac{2\pi n\epsilon_i}{N_{FFT}}+\phi} \times ... \left(\text{sinc}(k-\theta_i-\Delta_i) + \sum_{p=1}^{P} \alpha_{p,i} \,\text{sinc}(k-\tau_{p,i}) \right) x[n-k] \tag{2.26}$$

where N_{CM} is the number of CMs connected to the system and i is the index of each CM. By defining:

$$h_i[k] = 10^{\frac{A_i}{20}} e^{-j\frac{2\pi n\epsilon_i}{N_{FFT}}+\phi_i} \left(\text{sinc}(k - \theta_i - \Delta_i) + \sum_{p=1}^{P} \alpha_{p,i} \,\text{sinc}(k - \tau_{p,i}) \right) \tag{2.27}$$

the channel equation can be simplified to:

$$y[n] = \sum_{i=1}^{N_{CM}} \sum_{k} h_i[k] x_i[n-k] \tag{2.28}$$

The signal $y[n]$ from Equation 2.28 will be used throughout the remainder of the chapter to denote the received signal before any common channel impairments have been applied. The remainder of this chapter will focus on constructing the signal $r[n]$ (given in Equation 2.40), which represents the final received signal with all impairments applied.

2.2.3.1 Carrier Hum Modulation

In North America, power is typically supplied to an HFC system by a periodic quasi-square waveform with frequency 60 Hz. This waveform is often transmitted along the same wire as the transmitted signal. Carrier hum modulation occurs when the transmitted signal is amplitude modulated by the power supply waveform. [23]. Carrier hum modulation has two main sources, the most obvious source being a voltage ripple at the output of an amplifier. A secondary source can be attributed to parametric modulation of magnetic component properties (i.e. inductors and/or transformers) [24]. Both sources are due to either faulty or degraded internal amplifier components

The power supply waveform itself is typically modelled as a trapezoidal waveform [23] [24] [25], with rise and fall times of approximately 250 µs [25]. Since the rise and fall times are so small compared to the period of the trapezoidal waveform, the waveform itself can also be approximated by clipping a 60 Hz sinusoid at the correct intervals.

To represent the carrier hum waveform mathematically, consider the following sinusoid:

$$x_{\text{hum}}(t) = \sin\left(2\pi F_{\text{hum}} t + \phi_{\text{hum}}\right) \tag{2.29}$$

where F_{hum} is the carrier hum frequency and ϕ_{hum} is a phase shift in radians. By setting ϕ_{hum} and evaluating this sinusoid at the transition time, t_{trans}, its maximum clipped amplitude

can be determined. Scaling the waveform by its maximum clipped amplitude and applying basic sampling theory yields a discrete-time description of the carrier hum waveform:

$$m_{\text{hum}}[n] = \begin{cases} 10^{\frac{-A_{\text{hum}}}{20}}, & x_{\text{hum}}[n] > 1 \\ -10^{\frac{-A_{\text{hum}}}{20}}, & x_{\text{hum}}[n] < -1 \\ 10^{\frac{-A_{\text{hum}}}{20}} x_{\text{hum}}[n], & otherwise \end{cases} \tag{2.30}$$

where $x_{\text{hum}}[n] \equiv x_{\text{hum}}(nT_s)$. The received signal is then amplitude modulated onto the discrete-time carrier hum waveform:

$$r_1[n] = y[n](1 + m_{\text{hum}}[n]) \tag{2.31}$$

DOCSIS 3.1 specifies the carrier hum modulation in the upstream be limited to less than -26 dBc, or 5% of the total signal power; thus, the clipped sinusoid should be scaled by a maximum factor of $A_{\text{hum}} = 26$ dBc.

2.2.3.2 Ingress Noise

Ingress noise is the result of external narrowband AM and/or FM signals entering the HFC network. This typically occurs if a cable line is breached or a connector becomes corroded [26]. If the ingress signal is strong enough, it can cause a "spur" in the upstream spectrum, potentially rendering any subcarriers at or near its frequency useless.

To digitally model ingress noise, a baseband AM and/or FM signal must be generated and added to the received signal. The first step in generating either type of baseband signal is to determine its bandwidth, which is regulated by the International Telecommunication Union (ITU) and is easily available. Once this has been established, a baseband message signal whose frequencies lie within the bandwidth is generated. For the purposes of this research, both AM and FM message signals are defined as the following sum of sinusoids:

$$m[n] = \sum_{q=1}^{Q} A_q^m \cos\left(2\pi \frac{F_q^m}{F_s} n + \phi_q^m\right) \tag{2.32}$$

In Equation 2.32, Q represents the number of sinusoids in a message signal, F_s represents the sampling frequency, and A_q^m, F_q^m and ϕ_q^m represent the amplitude, frequency and phase

of each sinusoid, respectively. Note in this case the superscript is used like a subscript. It does not indicate "raised to the power of m."

After the message signal has been generated, it amplitude modulates (AM) or frequency modulates (FM) a carrier signal, the frequency of which is again regulated by the ITU. This carrier signal is defined relative to the baseband spectrum of the OFDMA signal as follows:

$$c_{AM}[n] = \frac{C}{2} e^{j\left(2\pi \frac{F_c}{F_s} n + \phi_c\right)}, \quad -51.2 \text{ MHz} < F_c < 51.2 \text{ MHz} \tag{2.33}$$

where C is the carrier amplitude and ϕ_c is the carrier phase.

AM ingress noise is thus represented as:

$$I_{AM}[n] = c[n](1 + m[n]) \tag{2.34}$$

To create a discrete-time representation of the FM noise, however, its continuous-time equation must first be examined, which is:

$$I_{FM}(t) = c(t) e^{j2\pi f_\Delta \int_0^t m(\tau) d\tau} \tag{2.35}$$

where f_Δ in Equation 2.35 is the peak frequency deviation of the FM signal [27]. Note that the message signal is integrated over time, with τ, in this case, serving as the placeholder variable of integration, which is not its usual role of representing transport delay. In discrete-time, this becomes:

$$I_{FM}[n] = c[n] e^{j2\pi f_\Delta m_{\text{int}}[n]} \tag{2.36}$$

where the integral is replaced by:

$$m_{\text{int}}[n] = \sum_{n'=n_0}^{n} m[n'] T_s \tag{2.37}$$

which is approximately the Riemann sum of the message signal, where n_0 represents the starting index and $T_s = 1/F_s$ is the sampling time.

With discrete-time representations of both ingress noise types established, their effect on the received signal can be modelled by adding Equations 2.34 and 2.36 to Equation 2.31, which yields:

$$r_2[n] = y[n](1 + m_{\text{hum}}[n]) + \sum_{u=1}^{U} I_{AM\_u}[n] + \sum_{v=1}^{V} I_{FM\_v}[n] \tag{2.38}$$

where U is the total number of AM ingress signals and V is the total number of FM ingress signals.

2.2.3.3 Additive White Gaussian Noise

The final impairment considered in this model is thermal noise, which is generated by the random movement of electrons [28]. Thermal noise can be found in all communication systems. It is considered to be approximately "white", meaning that it has a power spectral density (PSD) that is essentially flat over all frequencies. Thermal noise within a finite bandwidth is modelled as having a zero-mean Gaussian amplitude distribution; thus, another common term for thermal noise is additive white Gaussian noise, or AWGN as it is henceforth referred to.

To simulate AWGN, random noise must be generated and scaled by its variance. For the purposes of this model, the AWGN variance, σ_w^2, is defined as [18]:

$$\sigma_w^2 = E\left\{|X[n]|^2\right\} 10^{-\frac{\text{CNR}}{10}} \tag{2.39}$$

where $E\{|X[n]|^2\}$ is the average energy in each employed QAM constellation and the CNR is the carrier-to-noise ratio in dBc. Since DOCSIS 3.1 requires that each QAM constellation be scaled to ensure their average energy is unity, $E\{|X[n]|^2\} = 1$. The CNR is defined by the DOCSIS 3.1 standard as the ratio of average signal power in the occupied bandwidth to the average noise power in the occupied bandwidth given by the noise power spectral density integrated over the same occupied bandwidth. Normally, the CNR is varied to test the channel under different noise conditions; however, it is important to note that the standard outlines minimum CNR values that must be met for each individual QAM constellation, all of which are provided in Table 2.2.

Adding AWGN to the received signal yields the final equation of this chapter:

$$r[n] = y[n](1 + m_{\text{hum}}[n]) + \sum_{u=0}^{U-1} I_{AM\_u}[n] + \sum_{v=0}^{V-1} I_{FM\_v}[n] + w[n] \tag{2.40}$$

where $w[n]$ is the low-pass filtered noise sampled at time T_s (i.e. $w[n] = w(nT_s)$. This final equation accounts for the effects of all CM-specific ($y[n]$) and common channel impairments contained in this model.

Table 2.2: DOCSIS 3.1 CNR Values [12]

Constellation	**CNR (dB)**
QPSK	11.0
8-QAM	14.0
16-QAM	17.0
32-QAM	20.0
64-QAM	23.0
128-QAM	26.0
256-QAM	29.0
512-QAM	32.5
1024-QAM	35.5
2048-QAM	39.0
4096-QAM	43.0

3. MATLAB Model

3.1 Overview

This chapter provides a detailed description of the MATLAB model used to simulate the functionality of the cable modems (CMs) and upstream channel as specified in the data-over-cable service interface specification (DOCSIS) 3.1 standard. In general, the model is capable of simulating traffic, fine ranging, and probing transmissions, along with all of the channel impairments discussed in the previous chapter. All simulations are performed on a frame-by-frame basis, which allows any corrections based on fine ranging or probing information to be made between frames.

3.1.1 Program Flow

The MATLAB simulation is run from a script named `main.m`. Prior to running `main.m`, the user must first set up the simulation by specifying the number of frames to be sent, the structure of each orthogonal frequency-division multiple access OFDMA frame, and the channel settings. This is accomplished by declaring the appropriate variables in the MATLAB workspace.

Following the initial setup, the program calls the function `Channel_Setup`, which generates pseudo-random values for the channel offsets. The program then enters the main loop, which repeatedly calls the `Transmitter` and `Channel` functions until all OFDMA frames have been generated and passed through the channel. The result is a single Rx vector containing the combined "received" signals of each CM.

Since modelling a DOCSIS 3.1 receiver is beyond the scope of this project, no receiver

function is included in the MATLAB simulation. Should the user desire to test a receiver function, it should be placed in the main loop immediately following the `Channel` function. This will allow the receiver to decode each frame's Rx signal vector individually.

A comprehensive program flow diagram is provided in Figure 3.1. The inner workings of the functions depicted in Figure 3.1, along with the program setup process, are described in detail throughout the remaining subsections of this chapter. However, before any specific details are discussed, it is important to understand how several key components of the MATLAB model are structured.

3.1.2 Variable Structuring

One of the most useful features of MATLAB is its visual command interface. A user is able to view and interact with the contents of an array in the "Variables" window simply by double-clicking on its variable name in the "Workspace" window. This can be highly advantageous when debugging or running tests using the model. In order to allow the user to make full use of this feature, several guidelines were followed when creating and storing variables.

First, all important output vectors are stored on a CM-by-CM basis using cell arrays. These cell arrays, in turn, may be stored in additional cell arrays on a frame-by-frame basis in order to preserve data throughout a multi-frame simulation. Several variables which use this technique are given in Table A.8 in Appendix A. Should the user desire to store more variables in such a fashion, they must construct their own cell arrays in the `main.m` script.

Second, in the case of Tx (transmitted) and Rx (received) signal vectors, the first element of the vector is the first item transmitted and the first item received, respectively.

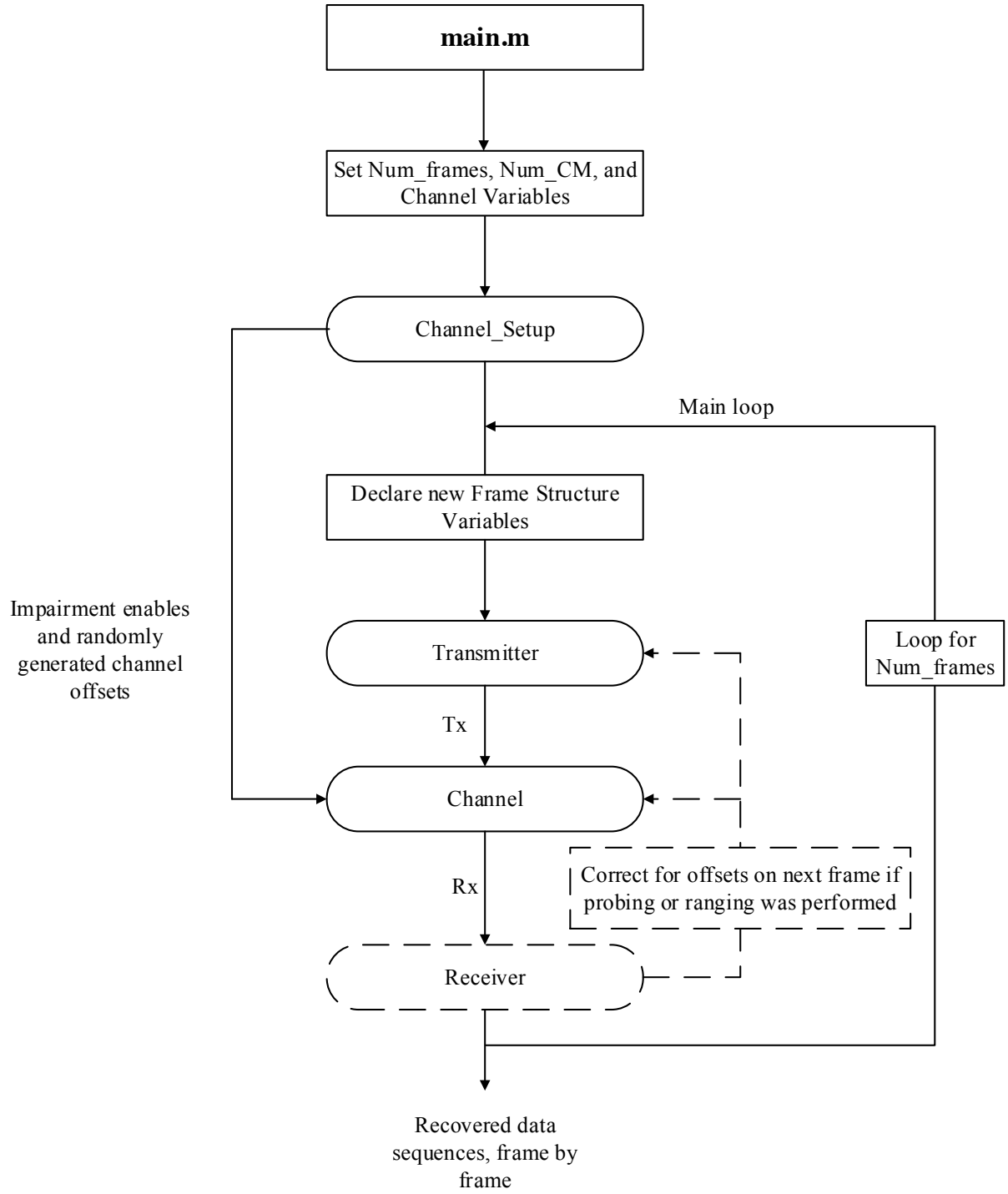


Figure 3.1: Program Flow Diagram

Finally, while the first element of any array is assigned a MATLAB index of "1", this element is still referred to in the documentation by its theoretical index of "0" when dealing with sub-carriers and OFDMA symbol numbers. CMs, frames, and channel impairment signals, however, are indexed according to MATLAB's system, as described by the equations derived in Chapter 2 (i.e. CM #1 is the first CM, frame #1 is the first frame, Ingress #1 is the first ingress, etc.).

3.2 Initial Setup

The MATLAB model was designed to give users the maximum amount of control possible over both the transmitter and channel impairment settings in order to provide a wide range of test scenarios. Consequently, a large number of input variables must be manually specified by the user before a simulation can be run. The input variables can be divided into two main categories: frame structure variables and channel variables.

3.2.1 Frame Structure Variables

The OFDMA frame structure variables can be further subdivided into three separate subcategories: general, fine ranging, and probing. General variables must be specified for any given frame, while fine ranging and probing variables must only be specified if a frame contains data from CMs that are in either mode, respectively. Descriptions and parameters for all variables in these three subcategories are provided in Tables A.1, A.2, and A.3, respectively, in Appendix A.

In order to produce these variables, it is recommended that the user list them individually in a customized MATLAB script. It is important to note that if the values of any of the variables are changed from frame to frame, the variables must be updated prior to calling the function `Transmitter`.

3.2.2 Channel Variables

The channel settings can be specified by manually altering the variables provided underneath the commented section in `main.m` labelled "Channel". Most importantly, the channel

variables must be specified prior to calling the function `Channel_Setup`. Descriptions and parameters for these variables are provided in Table A.5 in Appendix A.

In general, for most channel impairments to be applied, their respective enable variables must be set to "1". Unlike the frame structure variables, the channel variables need only be declared once, and are valid for the entire run of the simulation.

3.3 Offset Generation

Once the initial setup has been completed, `main.m` can be run. The first function called by the `main.m` is `Channel_Setup`. `Channel_Setup` performs two primary tasks. The first task is checking the channel variables for errors. If all of the channel variable values are valid, `Channel_Setup` will then move onto its second task, which involves generating pseudo-random offsets for the channel impairments that were selected for simulation. A description of these offsets is provided in Table A.4.

Note that the main-path fractional timing offset, while generated in terms of samples, is applied through the use of pre-equalization coefficients.

3.4 Transmitter

Having checked and generated all channel variables, the program then enters the main loop. The first function that is called in the main loop is `Transmitter`, which is used to generate OFDMA frames and convert them into Tx signal vectors. A flow diagram for the `Transmitter` function is provided in Figure 3.2.

Upon instantiation, `Transmitter` calls the sub-function `PRBS_LFSR`, which generates the probing symbol for each individual sub-carrier according to the DOCSIS 3.1 physical layer specification. `Transmitter` then converts the probing symbols to binary phase-shift keying (BPSK), and performs error checking on the general frame structure variables.

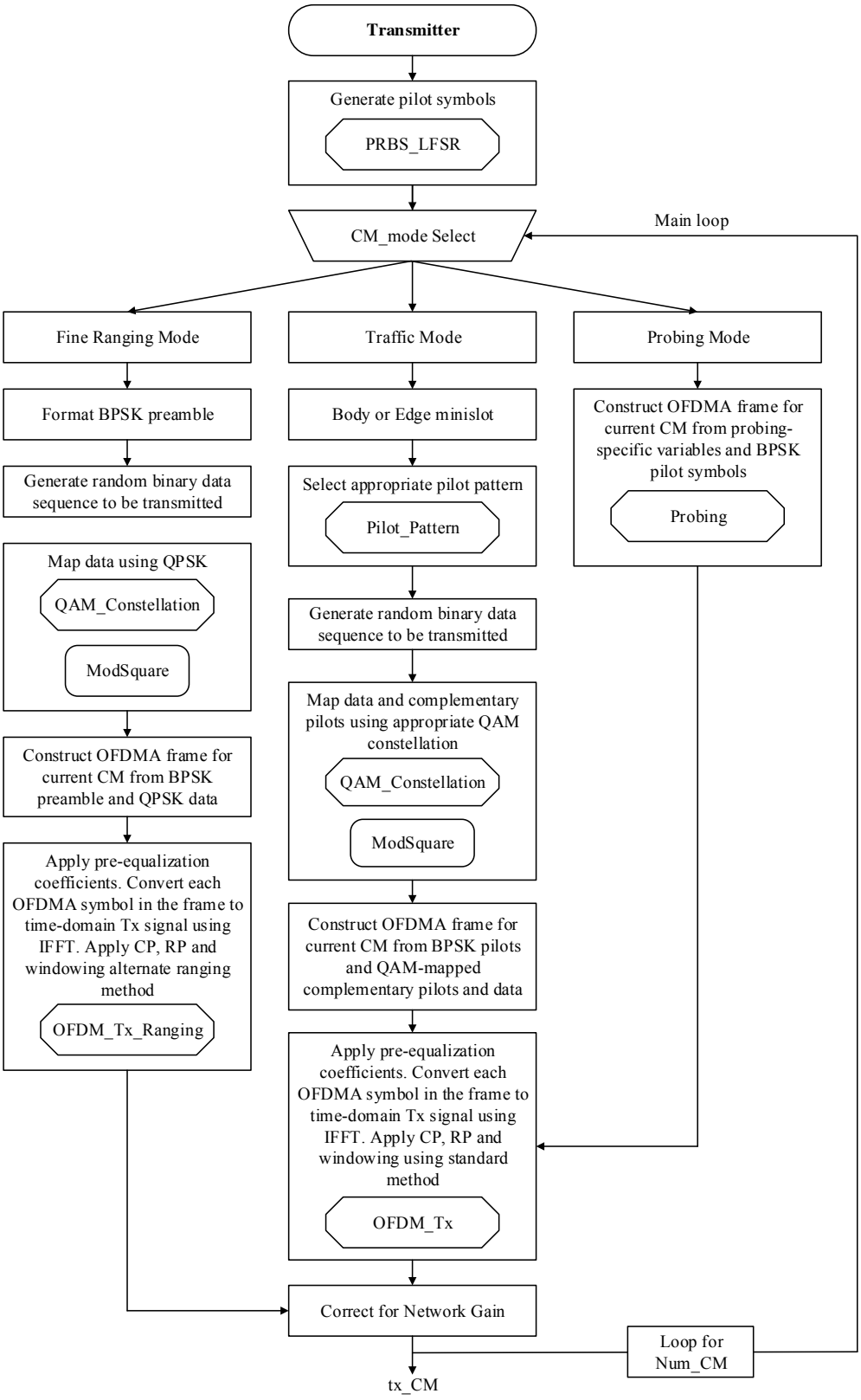


Figure 3.2: Transmitter Function Flow Diagram

Following basic error checking, `Transmitter` enters an OFDMA "frame-component" generation loop that operates on a CM-by-CM basis. Here the term "frame-component" refers to a portion of the total OFDMA frame (N_{FFT} subcarrier by K OFDMA symbols) that has been allocated to a specific CM, which translates to a unique Tx signal vector when its IFFT is taken.

Upon entering this loop, `Transmitter` scans the frame structure variables for the transmission mode of the current CM. Recall from Section 2.1 that there are three possible transmission modes: traffic mode, fine ranging mode, or probing mode.

3.4.1 Traffic Mode

If the current CM is in traffic mode, `Transmitter` will begin constructing the CM's corresponding OFDMA frame-component on a minislot-by-minislot basis. First, the sub-carrier array (`SC2k/SC4k`) is scanned in order to determine whether a Body or Edge minislot is present. The pilot array (`Pilot2k/Pilot4k`) is then scanned to determine the appropriate pilot pattern for the minislot. Both pieces of information are fed to the sub-function `Pilot_Pattern`, which checks for errors and generates a 2-D array of the current minislot with markers indicating the locations of pilot and complementary pilots. Following this, the orders of the quadrature amplitude modulation (QAM) constellation for both regular data and the complementary pilots in the minislot are obtained from the QAM-mapping array (`Mod2k/Mod4k`).

The next stage in the traffic mode process involves obtaining the data sequence that will be transmitted. Referring back to the problem statement in Section 1.4, it is assumed that this data has been formatted into a serial stream of binary numbers. To simulate this, `Transmitter` calculates the exact number of bits that can be sent in a single minislot based on the pilot pattern and modulation orders for both regular data and complementary pilots. MATLAB's `rand` function is then used to generate a pseudo-random binary bit sequence. Due to the high degree of complexity associated with generating minislot-specific bit sequences, the user is not allowed to control the pseudo-random bit stream itself; however, it can be viewed on a CM-by-CM basis by accessing the variable `b_orig_CM`.

Once the binary data sequence has been generated, `Transmitter` enters into a loop that maps the sequence to the appropriate elements of the current 2-D minislot array. If the loop index corresponds to a pilot marker, a BPSK-pilot symbol is placed accordingly. Otherwise, the sub-function `QAM_Constellation` is called to map the data and place either a regular QAM symbol or a complementary pilot appropriately. `QAM_Constellation` itself relies on a recursive sub-function named `ModSquare` in order to map the data to the correct symbol in a given QAM constellation. As was mentioned in Section 2.1.1, only square constellations are considered for the purposes of this research; hence, `ModSquare` is only capable of mapping BPSK, quadrature phase-shift keying (QPSK), 16-QAM, 64-QAM, 256-QAM, 1024-QAM, and 4096-QAM. The mapping process is repeated until each element of each 2-D minislot array in the OFDMA frame-component has been filled.

After all of the data has been mapped to each minislot, the completed frame-component of the specified CM is passed to the sub-function `OFDM_Tx`, which uses a loop environment to convert each OFDMA symbol of the frame-component into a Tx signal vector. `OFDM_Tx` begins the conversion process by applying pre-equalization coefficients to each sub-carrier (row) of the frame-component. Next, `OFDM_Tx` uses MATLAB's default `ifft` function (modified in accordance with Equation 2.3) to obtain the IDFT of each OFDMA symbol. A cyclic prefix and roll-off period are then added to each OFDMA symbol's IDFT sequence. Following this, `OFDM_Tx` windows each OFDMA symbol using a Tukey raised-cosine (RC) window. The result is the time domain signal for the specified CM, which is stored according to its CM index in the cell array `tx_CM`. Finally, if the attenuation is being corrected for, `Transmitter` scales the Tx signal by the designated correctional factor.

3.4.2 Fine Ranging Mode

If the current CM is in fine ranging mode, `Transmitter` checks for errors in the setup of the fine ranging and the guard band sub-carriers. Provided that the setup is valid, `Transmitter` then proceeds to read a user-defined preamble sequence, which must be a column vector of binary numbers. Should the length of the preamble not match the number of fine ranging sub-carriers, N_{fr}, it will be either padded with zeros or truncated to length

N_{fr}. The preamble is then mapped using BPSK.

As was mentioned in Section 2.1.3.1, the preamble sequence of a fine ranging signal is typically followed by several OFDMA symbols that contain QPSK-mapped data. Realistically, this data sequence would serve as a unique identifier for each individual CM; however, since this model uses its own variables to identify CMs, it does not serve any functional purpose. Nevertheless, it is unlikely that a preamble will ever be sent by itself; therefore, in an effort to make the scenario as realistic as possible, an array of pseudo-random binary numbers is generated using MATLAB's `rand` function to be sent along with the preamble.

Once the preamble has been formatted and the data sequence has been generated, `Transmitter` begins constructing the specified CM's OFDMA frame-component. This is done using a loop, in which both the BPSK-mapped preamble symbols and the QPSK-mapped data sequence symbols are mapped to a 2-D minislot array. The completed frame-component is then passed to `OFDM_Tx_Ranging`, which performs essentially the same tasks as `OFDM_Tx`. The difference between the two sub functions resides in the way that the time-domain sequence is structured, which was covered in detail in Section 2.1.3.1. The resulting time domain sequence is then stored and attenuated using the exact same method described for traffic mode.

It is important to note that the DOCSIS 3.1 standard is somewhat ambiguous in regards to how it structures the QPSK-mapped data sequence in the time domain; thus, for simplicity's sake, `OFDM_Tx_Ranging` was programmed so that every 2nd OFDMA symbol was a copy of the symbol preceding it, and a new data sequence was generated every two symbols. Similar to traffic mode, the length of the binary data sequence is exactly proportional to the number of sub-carriers and symbols used; therefore, the user is not allowed to control it. However, the sequence can be viewed by accessing the variable `b_orig_CM`.

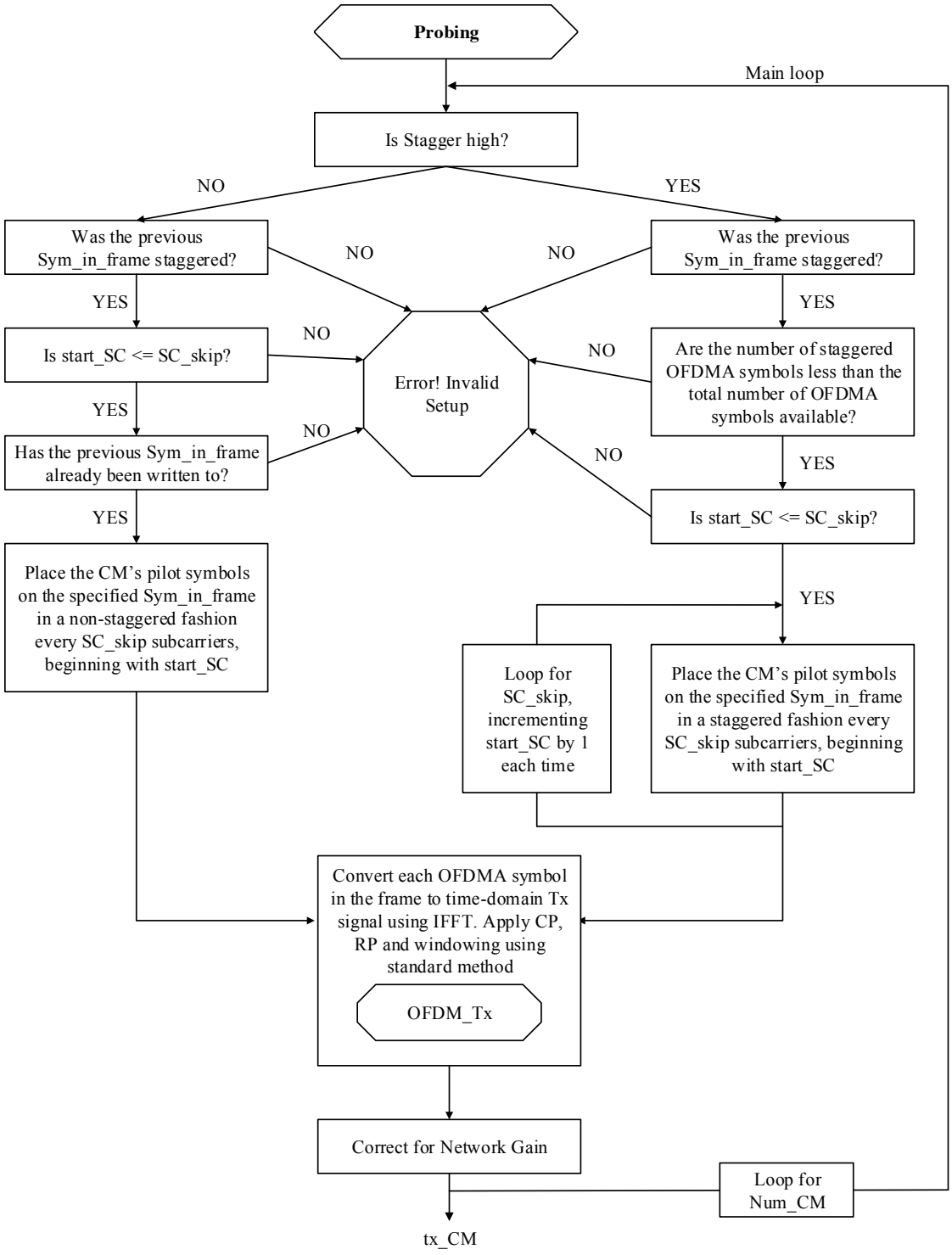


Figure 3.3: Probing Sub-Function Flow Diagram

3.4.3 Probing Mode

If any CM in the current frame is in probing mode, then the entire frame is considered to be a probing frame. This means that an invalid setup message will be produced if any other CMs in the frame are in traffic or fine ranging mode. When a probing frame is detected, `Transmitter` calls the function `Probing` to begin setting up the frame.

As with the other two transmission modes, the first process performed by `Probing` is basic error checking, which encompasses all probing-specific variables listed in Table A.3 in Appendix A. Once this has been completed, `Probing` enters the frame setup loop. Within the loop are a sequence of nested "if" statements that are used to determine the appropriate frame-component structure for a given CM and its probing-specific variables. The flow of these "if" statements is depicted in Figure 3.3.

In each case, after a CM's complete frame-component has been generated it is passed to `OFDMA_Tx`. Like the other two transmission modes, it is then converted to the time domain, stored and attenuated.

3.5 Channel

Having used `Transmitter` to generate each CM's OFDMA frame components and convert them to Tx signals, the main loop then invokes the `Channel` function. By this point, a large majority of the error checking has already been performed; thus, `Channel` is able to go straight into its main loop and begin applying CM-specific offsets. The flow of the `Channel` function is illustrated in Figure 3.4.

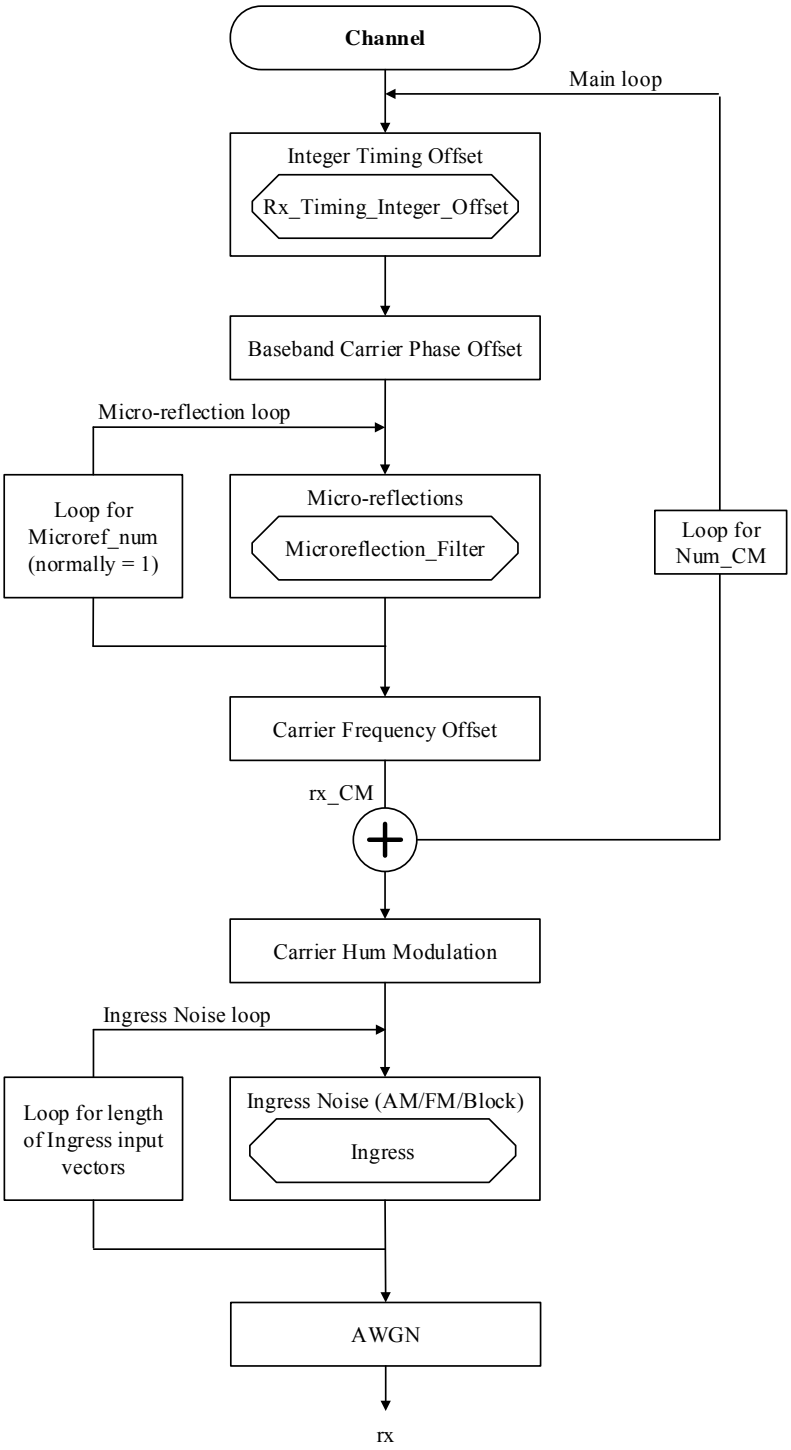


Figure 3.4: Channel Function Flow Diagram

3.5.1 Integer Timing Offset

The first offset applied by the channel is the integer timing offset, which is accomplished using sub-function `Rx_Integer_Timing_Offset`. This sub-function works by concatenating the last N_{RP} samples of the previous frame's individual CM-specific Tx signals with the first N_{RP} samples of the current frame's individual CM-specific Tx signals. The sub-function then delays each of the concatenated Tx signals by an integer amount according to its CM-specific integer timing offset, the values of which were generated previously using `Channel_Setup`. In MATLAB, delaying a specific CM's Tx signal by the integer timing offset is the equivalent of inserting an additional θ samples of the previous frame's Tx signal. The resulting offset Tx signals output by the function are referred to as Rx signals, as they have begun passing through the channel.

Although this process is simple to implement in theory, the fact that the vectors are different lengths makes it difficult to process them in a loop environment. `Rx_Integer_Timing_Offset` was therefore designed so that the size of all Rx output vectors are relative to the vector with the largest timing offset. This is accomplished by appending all smaller vectors with extra elements until all vectors are of uniform length. For example, if the largest integer timing offset is $\theta_1 = 100$, and a smaller offset is $\theta_2 = 90$, 10 additional zero-valued elements are appended to θ_2.

Note that when simulating the first frame in any transmission burst in MATLAB, there is no "previous frame" to concatenate it with. This means that the first $\theta + N_{RP}$ samples of the first frame are inherently unknown, which presents a challenge when trying to model a system that is transmitting in steady state. To address this, the variable `Frame_tag_mode` was added to the model. `Frame_tag_mode` allows the user to choose one of two types of data sequences that may precede a CM's Tx signal in the first frame of a transmission burst:

- `Frame_tag_mode` $= 0$: Fill the additional θ vector elements at the beginning of the current frame with zeros.
- `Frame_tag_mode` $= 1$: Concatenate the beginning of the first frame with a duplicate

of the last $\theta + N_{RP}$ samples of the first frame.

It is recommended that the user choose the second option, as filling the additional elements with values is more representative of a steady-state burst.

3.5.2 Micro-reflections

Once the signal has been time-delayed by an integer amount, `Channel` begins to generate micro-reflections for each CM's Rx signal vector. Although the DOCSIS 3.1 standard specifies only a single micro-reflection (as was discussed in Section 2.2.2.2), a design choice was made to give the simulation the ability to generate multiple micro-reflections (up to a maximum of 6, one for each category in Table 2.1), should the user desire to explore their effects. This is accomplished through the use of the sub-function `Microreflection_Filter`.

`Microreflection_Filter` works by first separating the total echo time (transport delay) of each micro-reflection into integer and fractional components. Next, it enters a loop where it generates a unique set of coefficients for a finite impulse response (FIR) fractional delay filter (FDF) based on the fractional delay value obtained from the previous step. The design of this filter is the subject of Chapter 4.

Once the coefficients have been generated, the Rx signal vector is filtered using MATLAB's built-in `filter` function. Following filtering, the fractionally delayed output sequence is truncated to match the original signal length. A complex attenuation corresponding to the current micro-reflection is then obtained from `Channel_Setup` and applied to the truncated signal.

Recall from Equation 2.20 that the complex attenuation, denoted as α_p, contains both a scaling factor and a phase shift. The scaling factor, A_p (in dBc), is obtained from Table 2.1, which is stored as a look-up-table (LUT) in `Channel_Setup`. The phase shift is modelled as a pseudo-random variable between $-\pi$ and π radians.

After the complex attenuation has been applied, the micro-reflection vector is shifted by the integer delay component and added back to the original Rx signal vector, thus simulating a micro-reflection. Since shifting the micro-reflection vector by an integer amount extends it

beyond the length of the frame, the extra samples at the end of the frame are truncated and stored in a new vector called `rx_CM_microref_tail_prev`. This vector will be added to the beginning of the next frame in order to simulate continuous transmission.

There is a slight drawback associated with simulating the micro-reflections using the method described above. When a signal is first passed through a digital filter, the filter must undergo a transition period before it able to produce a steady-state output. For an FIR filter, the length of this transition period is equal to the filter order, $N - 1$. The outputs produced during this period are referred to as transient samples, or "transients". If the input signal is continuous, the first $N - 1$ samples will be transients; however, since the MATLAB simulation operates on a frame-by-frame basis, there are inherently a fixed number of samples available for the filter to process. Hence, the filter will undergo another transition period at the end of the frame, resulting in another $N - 1$ transient samples.

In order to eliminate the transients, the last $N - 1$ samples of the previous frame and first $N - 1$ samples of the following frame (omitting the integer timing offset and roll-off samples) should be appended to the beginning and end of the current frame, respectively, prior to filtering. These additional samples could then be truncated, thus producing a fractionally delayed frame whose length matches that of the input frame. However, as mentioned previously, the simulation cannot generate data from "future" frames; only data from the current and previous frame may be used. Therefore, no matter what is done, the first $N - 1$ samples of any filtered frame will always be transient samples.

At this point, it is important to note that within the transient samples themselves, the first $(N - 1)/2$ samples output by the filter are not considered to be valid data, while the subsequent $(N - 1)/2$ samples are considered to be valid, but distorted. Similarly, the last $(N - 1)/2$ output samples are invalid, while the $(N - 1)/2$ samples that precede them are valid, but distorted. In other words, the filter has a delay of $(N - 1)/2$ samples before valid data can be output.

With this information in mind, a choice was made to approximate an ideal micro-reflection by truncating the first and last $(N - 1)/2$ invalid samples of the FIR FDF's output. The

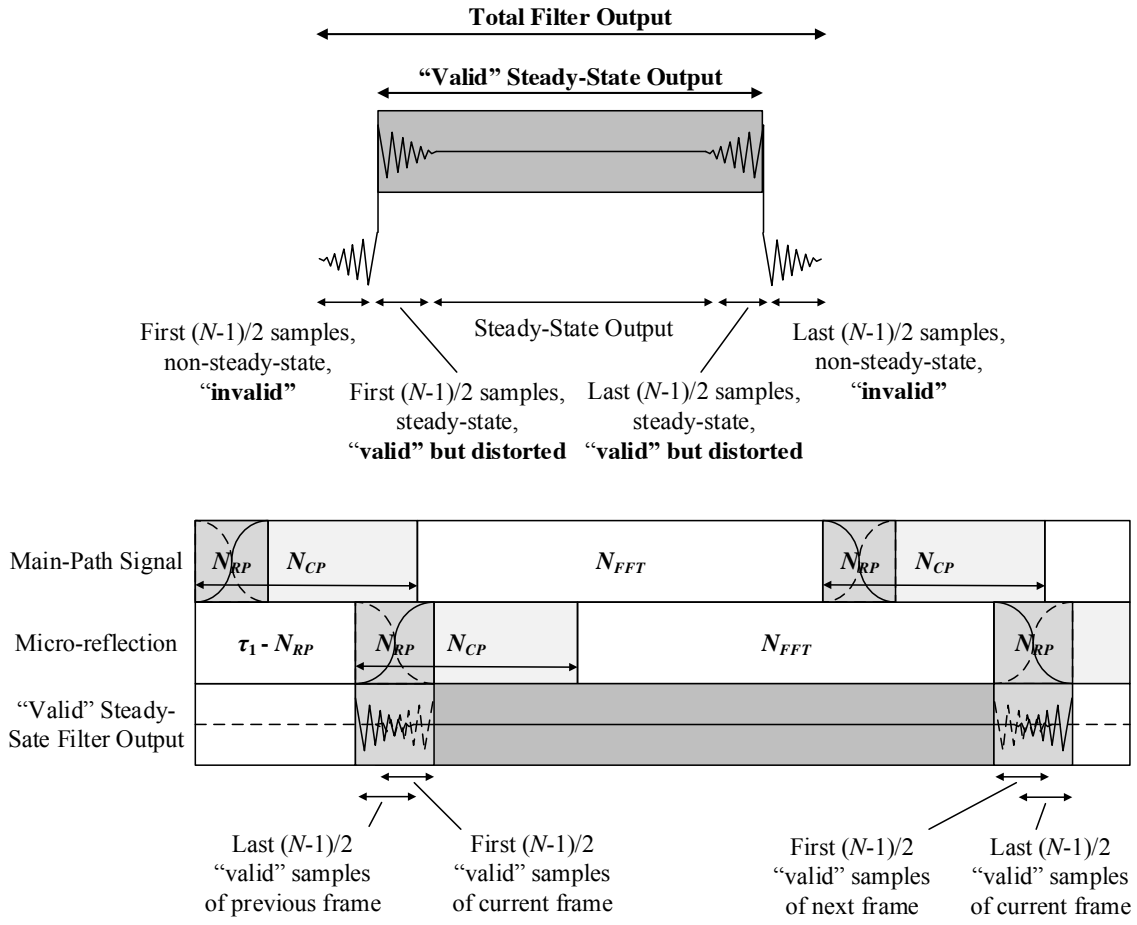


Figure 3.5: Distorted Filter Output Caused by Using Frames of a Finite Length to Simulate Micro-reflections

first and last $(N-1)/2$ of the resulting output frame are therefore distorted, yet valid, as Figure 3.5 illustrates.

The drawback associated with this approximation occurs after `Microreflection_Filter` has finished generating and adding the current CM's micro-reflection vector(s) to its Rx signal vector. Once this procedure has been completed, `Channel` enters into another loop where the tail-end of each of the previous frame's micro-reflection vector(s) is added to the beginning of the current frame. Thus, at any given time, there will be $N-1$ distorted samples present in the micro-reflections, as depicted in Figure 3.5[1]. If

[1]Note that Figure 3.5 depicts only a single OFDMA symbol in the given frame, while in reality the frame would contain K OFDMA symbols.

these distorted samples lie within the original frame's cyclic prefix, then their effect on the received signal is negligible, as the cable modem termination system (CMTS) will remove the cyclic prefix samples. However, if the combination of τ_1, N_{RP}, and N_{CP} causes some of these samples to occur outside the cyclic prefix, as Figure 3.5 illustrates, the distorted samples will affect the received signal in an minor, yet undesirable, fashion.

3.5.3 Carrier Phase and Frequency Offsets and Network Gain

The final three CM-specific impairments that must be applied are the carrier phase offset (CPO), the carrier frequency offset (CFO), and the network gain/attenuation, the values of which are pseudo-randomly generated in `Channel_Setup` and stored respectively in the variables
`Phase_offset`, `Freq_offset`, and `Pwr_offset`. The value of `Phase_offset` can range anywhere from $-\pi$ to π radians. Typically, the value of `Freq_offset` would be from -0.5 to 0.5 samples, which translates to between $\pm$25 kHz based on the minimum subcarrier spacing of 50 kHz in 2k transmission mode, or $\pm$12.5 kHz based on the minimum subcarrier spacing of 25 kHz in 4k mode. However, due to the strict frequency synchronization requirements imposed by DOCSIS 3.1, it was decided that a range of ϵ should be reduced to $-0.01 \leq \epsilon \leq 0.01$ samples with a resolution of 10^{-3} samples. As mentioned in Section 2.2.2.4, the value of `Pwr_offset` was selected to range from -9 dB to +3 dB.

To apply the CPO and CFO, `Phase_offset` and `Freq_offset` are first substituted for ϵ and ϕ, respectively, in the complex exponential of Equation 2.27. This complex exponential is then implemented directly in MATLAB as a vector. Note that while the CPO remains the same throughout the vector, the CFO is dependent upon the vector index and will change as the index increases. Finally, each element of the complex exponential vector is multiplied by the element of the Rx signal vector that corresponds to its index (i.e. " .* " multiplication in MATLAB).

The network gain/attenuation is applied simply by converting `Pwr_value` to an amplitude multiplying the Rx signal vector by the result.

Following the application of these three impairments, the current CM's Rx signal is added

to the combined total of all Rx signals. `Channel` then loops back and applies new CM-specific impairments to the remaining CMs until the Rx signals of all transmitting CMs have been combined.

3.5.4 Carrier Hum Modulation

Hum modulation is a common impairment. The MATLAB function `Channel` implements carrier hum modulation by first determining a starting and ending sample indices for the carrier hum waveform. The starting index is calculated by subtracting the current value of the integer timing offset from the ending index of the previous frame. The ending index is the starting index plus the current frame length. Keeping track of the indices in such a fashion allows the signal to retain continuity across multiple frames, thus accurately modelling a burst transmission scenario.

Once the indices have been determined, the carrier hum waveform is generated. As discussed in Section 2.2.3.1, this waveform has a trapezoidal shape, which can be approximated using a sinusoid with a clipped amplitude. In MATLAB, the waveform is created by first determining the maximum amplitude that the sinusoid can take on after a user-specified transition period (recommended 250 ms). Next, the full sinusoid is generated according to a user-specified frequency (recommended 60 Hz) and phase shift. Any values exceeding the previously established amplitude threshold are then clipped, and the final waveform is scaled according the user-specified carrier hum attenuation factor in dBc. Finally, the carrier hum waveform is amplitude modulated onto the combined Rx signal, as depicted in Equation 2.31. A single cycle of the scaled waveform is provided in Figure 3.6 for verification purposes.

3.5.5 Ingress Noise

Ingress noise is another common channel impairment. Similar to carrier hum modulation, the first step involved in generating an ingress noise signal is determining its sample indices. Following this step, `Channel` enters a loop that repeatedly calls on the sub-function `Ingress` to generate any number of ingress noise signals.

Each time `Ingress` is called, it can generate one of three different types of ingress noise

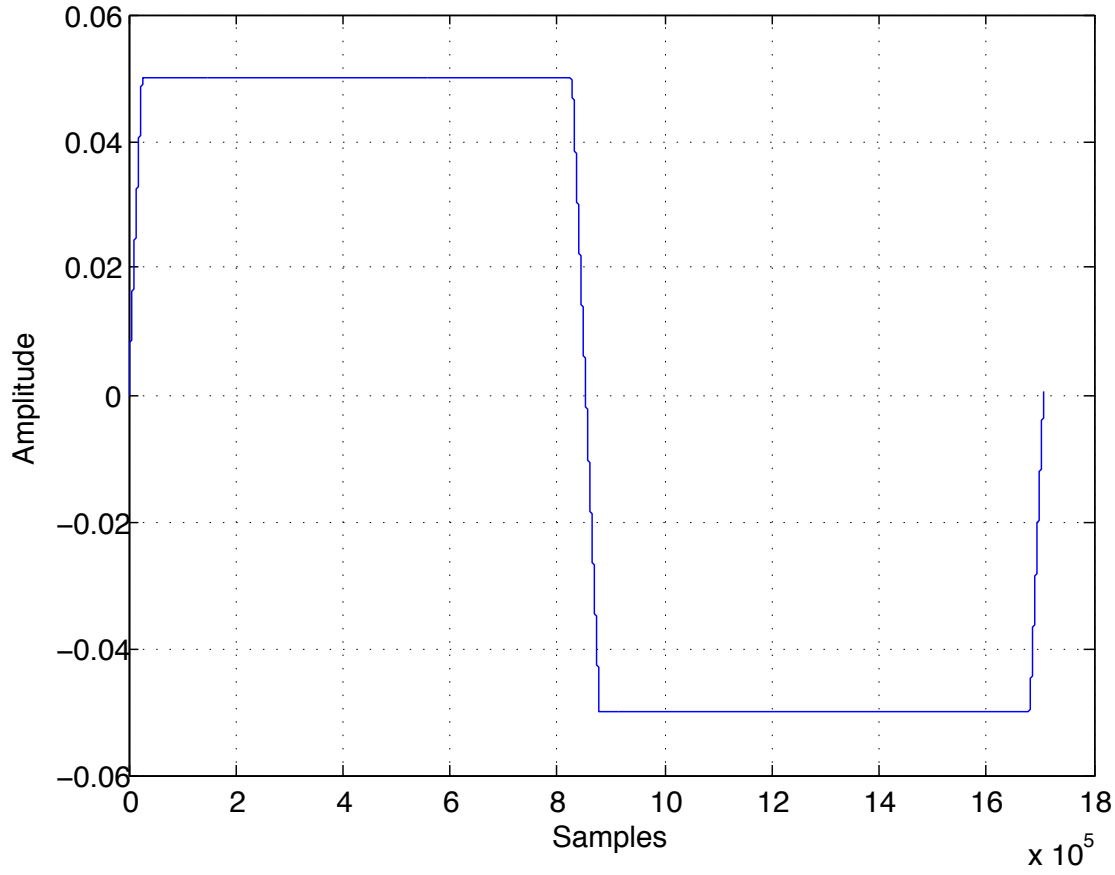


Figure 3.6: 60 Hz Carrier Hum Waveform, $F_s = 102.4$ MHz

signals: AM, FM, or "block." The "block" ingress noise option is a feature that was carried over from an early version of the `Ingress` function. When "block" is selected, `Ingress` generates blocks of ingress noise at frequencies on either side of a centre carrier frequency that are equally spaced across a specified bandwidth. The phases of the frequencies are pseudo-randomly generated by `Channel_Setup` and stored in the first row of the variable `Ingress_sin_setup`, while their uniform amplitude is set by the user (in dBc, relative to the main carrier frequency, assumed to be of unity gain). The block ingress can be useful in certain instances where the user wants to test the effects of ingress noise over a narrow, precise bandwidth.

`Ingress` generates AM or FM ingress noise signals by first generating a message signal. The phases, frequencies and amplitudes of each sinusoid in a given message signal are all pseudo-randomly generated by `Channel_Setup` and stored respectively along the rows

of the array `Ingress_sin_setup`, which is then passed to `Ingress`. `Ingress` then generates each sinusoid in the message signal and adds it to the previous sinusoid in a loop environment.

Once the message signal has been generated, `Ingress` generates a carrier, the frequency (Hz), amplitude (dBc), and phase (rads) of which are controlled by the user. Next, `Ingress` amplitude or frequency modulates the user-specified carrier frequency using either Equation 2.34 or Equations 2.36, respectively. Note that in the case of an FM ingress noise signal, the integration is completed by taking the cumulative sum of the message signal, as depicted in Equation 2.37. The MATLAB function `cumsum` is used to accomplished this.

The output of `Ingress` is a time-domain vector of ingress noise that is the same length as the combined Rx signal vector. An estimate of the ingress power is also provided by taking the variance of the output signal. After all of the specified ingress noise signals have been generated, they are added to the combined Rx signal. Similarly, the estimated power values are added together, and an approximate total ingress power value is provided in dBc.

3.5.6 Additive White Gaussian Noise

The final common impairment imposed upon the system is additive white Gaussian noise (AWGN), the real and imaginary components of which are generated using MATLAB's pseudo-random Gaussian number generator function, `randi`. Once a noise vector that matches the length of the combined Rx signal has been generated, it is then normalized using the variance calculated from Equation 2.39. The user is able to set the carrier-to-noise ratio (CNR) in the aforementioned equation using the variable `CNIR`.

After the noise vector has been added to the combined Rx signal, `Channel` performs one last calculation, which is estimating the carrier-to-noise-and-interference ratio, plus ingress. This is done using the following equation:

$$\text{CNIR\_plus\_ingress\_dB} = 10 \log_{10} \frac{P_C}{P_N + P_I} \tag{3.1}$$

where P_C is the carrier power (which is assumed to be unity), P_N is the power in the noise vector (calculated by taking its variance), and P_I is the ingress noise power (previously

obtained from the `Ingress` function).

Should the value of this combined ratio fall below 25 dB, a warning message will be displayed, as the DOCSIS 3.1 standard specifies a CNIR, plus ingress, value of no less than 25 dB. If this occurs, it is recommended that the user adjust either the CNR or the ingress noise generation parameters accordingly.

Finally, the combined Rx signal with all of its impairments is sent back to the main function where it can be passed to a receiver function, which is not implemented in this study, in order to be decoded.

3.6 Verification

The verification processes followed throughout the design of the MATLAB model can be broken down into two main categories: transmitter and channel verification. As their names imply, these verification categories encompass the functions `Transmitter` and `Channel`, as well as their respective associated sub-functions.

3.6.1 Transmitter Verification

To perform transmitter verification, two additional functions were created. The first was a pseudo-random test case generator named `Test_Random_Setup_Full`, which was capable of assigning any combination of frame generation variables to a frame in traffic, fine ranging or probing mode. This function was written collaboratively with University of Saskatchewan master's student Chad Holst. The second function was a receiver function, appropriately titled `Receiver`, which is capable of decoding any transmitted sequence obtained using `Test_Random_Setup_Full`. This was written by Chad Holst and Yayi Xiao, another master's student at the University of Saskatchewan. Both functions were developed in tandem with `Transmitter` so that verification could be performed progressively as more features and sub-functions were added.

The use of MATLAB's visual interface greatly assisted in the verification process for all functions designed for use in the MATLAB model. In most cases, visual comparisons of

arrays and variables present in the MATLAB workspace were enough to validate certain features. To assist with this visual verification process, several intermediate debugging variables were included in `Transmitter`, which can be found in Table A.8 in Appendix A.

The ultimate verification test for the transmitter and its associated functions was determining the bit-error-ratio (BER) between the transmitted and received bit sequences of each CM with no channel impairments present. The received bit sequence of each CM was decoded using `Receiver`. The BER of each CM was calculated as follows:

$$\texttt{BER\_CM} = \frac{1}{N_b} \sum_{i=1}^{N_b} |\texttt{b\_orig\_CM}[i] - \texttt{b\_rec\_CM}[i]| \tag{3.2}$$

where N_b is the total number of bits transmitted, `b_orig_CM` is the original transmitted binary bit sequence, and `b_rec_CM` is the received bit sequence. A BER of zero for all CMs in any test-case scenario indicated that the transmitter was functioning correctly.

3.6.2 Channel Verification

In terms of CM-specific channel impairments, the integer timing offset was verified visually in the MATLAB workspace by checking to see if the correct number of array elements had been inserted in the appropriate place in a CM's Tx vector.

The fractional timing offset was verified by comparing the magnitudes of two identical, randomly generated Rx signal vectors that were each fractionally delayed by the same randomly generated amount using a different method. The first Rx signal vector was obtained through the use of pre-equalization coefficients (generated using Equation 2.18) in the transmitter. The second Rx signal vector was obtained through the use of a FDF in the channel. Specifically, main-path FDF Design #2, which is discussed in Chapter 4, was used; however, the quality of the filter was not relevant to this particular verification process. The first 200 samples of each Rx signal's magnitude were plotted in Figure 3.7. Overlapping signals indicate a correct implementation, ignoring the error introduced by the FDF itself.

Micro-reflections were verified using a similar technique to the one described above. A micro-reflection with a random echo time and corresponding complex attenuation was generated using two different methods. The first method involved the use of pre-equalization

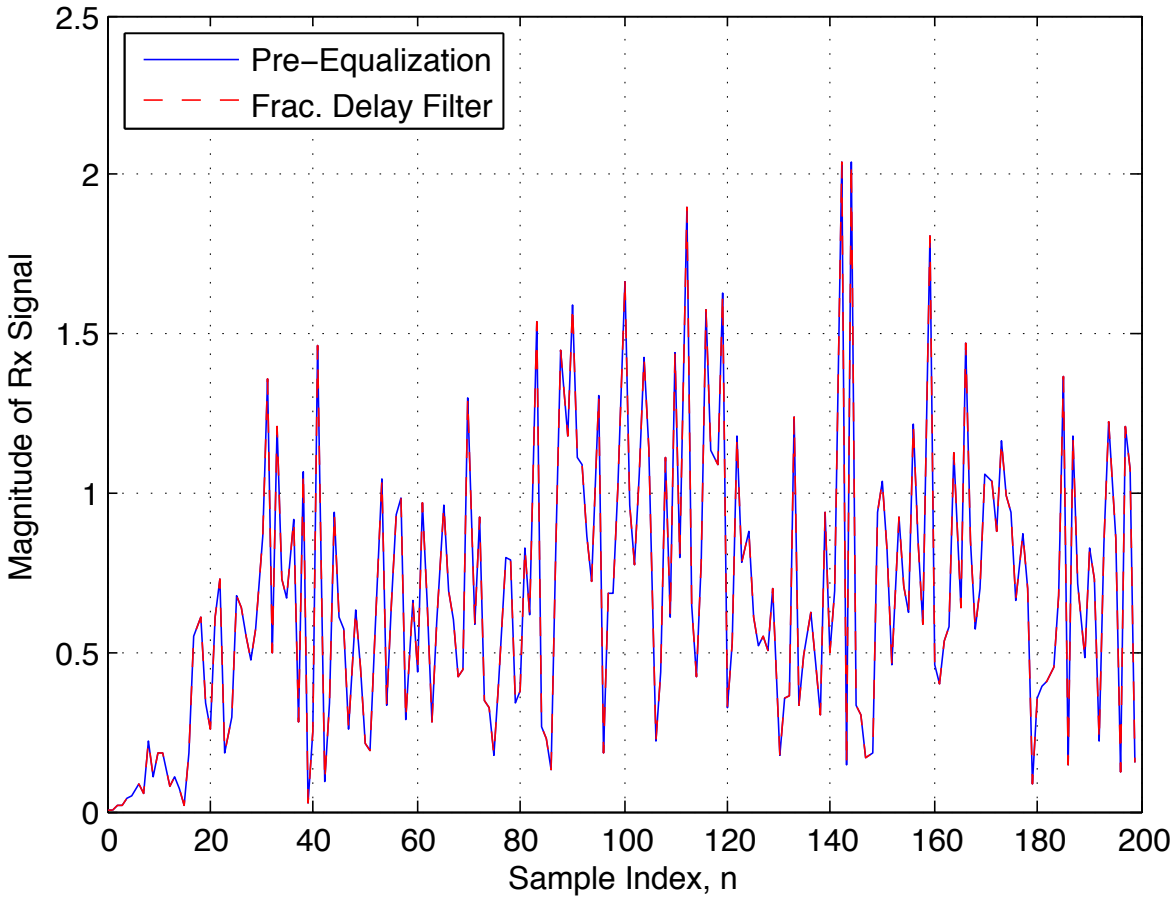


Figure 3.7: Fractional Timing Offset Verification: Pre-Equalization Coefficient Simulation Method vs. Fractional Delay Filter Simulation Method, $\Delta = 0.5$ samples

coefficients in the transmitter, which were calculated using Equation 2.22. The second method involved the use of a secondary channel path and its corresponding integer offset, FDF (secondary-path Design #2 in Chapter 4), and attenuation, all of which were encompassed in the function `Microreflection_Filter`. The micro-reflections obtained using each method were added to two identical, randomly generated Rx vectors, respectively. The magnitudes of the first 200 samples of these vectors were then compared, as illustrated in Figure 3.8. Once again, overlapping signals indicate a correct implementation, ignoring the error introduced by the FDF itself.

Note that in the scenario illustrated Figure 3.8, the length of the cyclic prefix is $N_{CP} = 96$ samples. Although it may be difficult to discern given the resolution of Figure 3.8, there is

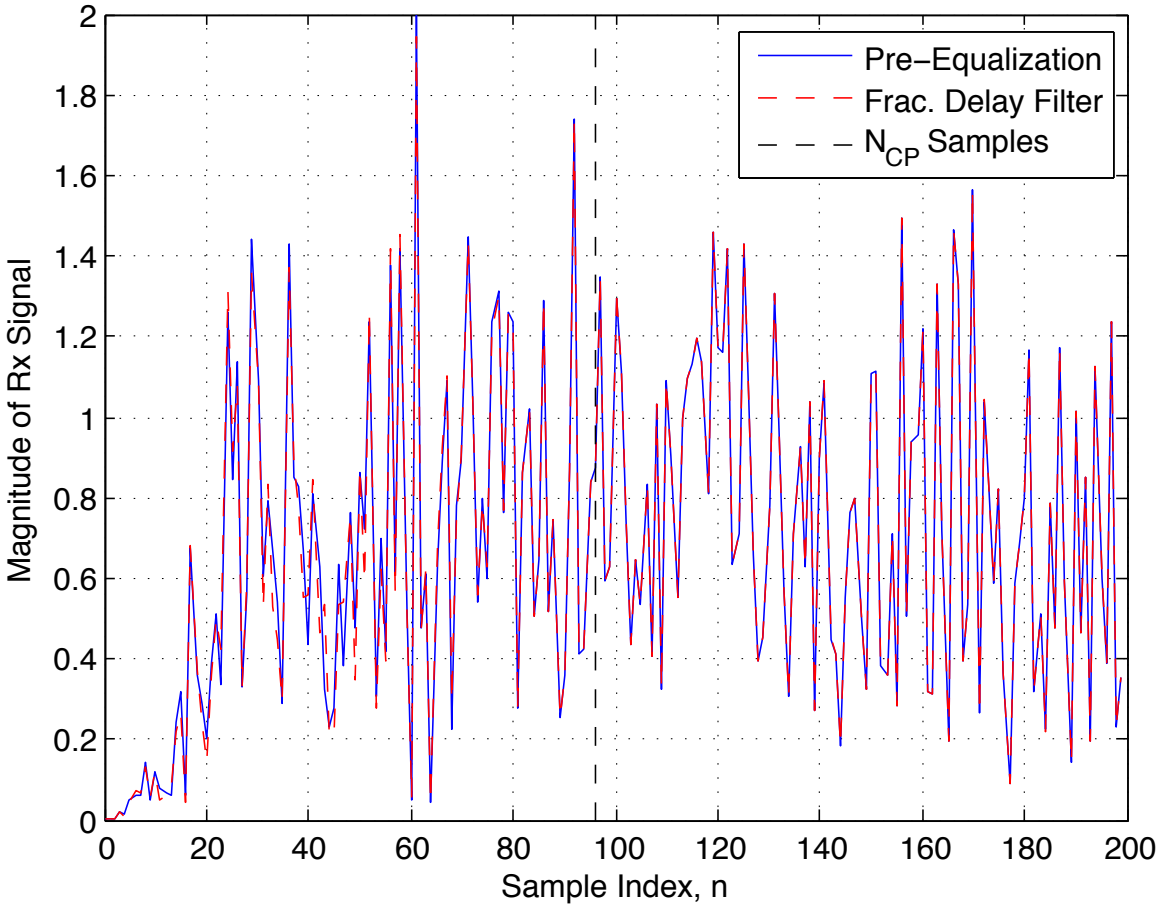


Figure 3.8: Micro-Reflection Verification: Pre-equalization Coefficient Simulation Method vs. Fractional Delay Filter Simulation Method

more error present in the first N_{CP} samples of the Rx signal magnitude compared to the rest. Recall from Section 2.2.2.2 that this discrepancy is expected, as simulating micro-reflections using pre-equalization coefficients results in signal discontinuities in the cyclic prefix.

The MATLAB implementations of the CPO, the CFO, and network gain/attenuation were carefully built to match their mathematical representations exactly; thus, no further verification was deemed necessary.

In terms of common channel impairments, Figure 3.6 and 3.9 demonstrates that the carrier hum waveform behaves as expected.

To verify the functionality of `Ingress`, a test-case scenario involving all three ingress

noise types was created. The input parameters used to generate each ingress noise type are provided in Table 3.1. AWGN with an arbitrarily chosen CNR of 60 dB was also added to the test-case.

An estimated power spectral density (PSD) plot of the test-case's spectrum (with AWGN present) is provided in Figure 3.9. The plot was obtained using MATLAB's `dspdata.psd`. Note that by default MATLAB plots two-sided power spectra from 0 to F_s instead of from $-F_s/2$ to $F_s/2$. Thus, ingress noise signals that occur in the range $-F_s/2$ to 0 are wrapped so that they appear in the range $F_s/2$ to F_s. From Figure 3.9, it can be seen that each ingress noise type appears on the PSD at its specified carrier frequency given in Table 3.1.

Finally, AWGN functionality was verified by implementing Equation 3.1 directly in MATLAB and setting $P_C = 1$, $P_I = 0$, and P_N equal to the variance of a pseudo-randomly generated noise vector with an arbitrarily chosen CNR. The resulting CNR of the equation matched the arbitrarily chosen CNR in all cases.

Table 3.1: Properties of Ingress Noise Types Depicted in Figure 3.9

Variable	Description	Value		
		Block	**AM**	**FM**
Ingress_carrier_freq	Carrier frequency	-44 MHz	-20 MHz	-30 MHz
Ingress_carrier_phase	Carrier phase	π/4	π/4	π/4
Ingress_BW	Bandwidth	200 kHz	10 kHz	200 kHz
Ingress_freq_spacing	Frequency spacing (Block Only)	10 kHz	N/A	N/A
Ingress_FM_freqdev	Peak frequency deviation (FM only)	N/A	N/A	75 kHz
Ingress_num_message_sig	Number of sinusoids in a message signal (AM & FM only)	N/A	100	100
Ingress_atten_dBc	Carrier attenuation in dBc (relative to "unity gain" OFDMA carrier)	-40	-40	-40

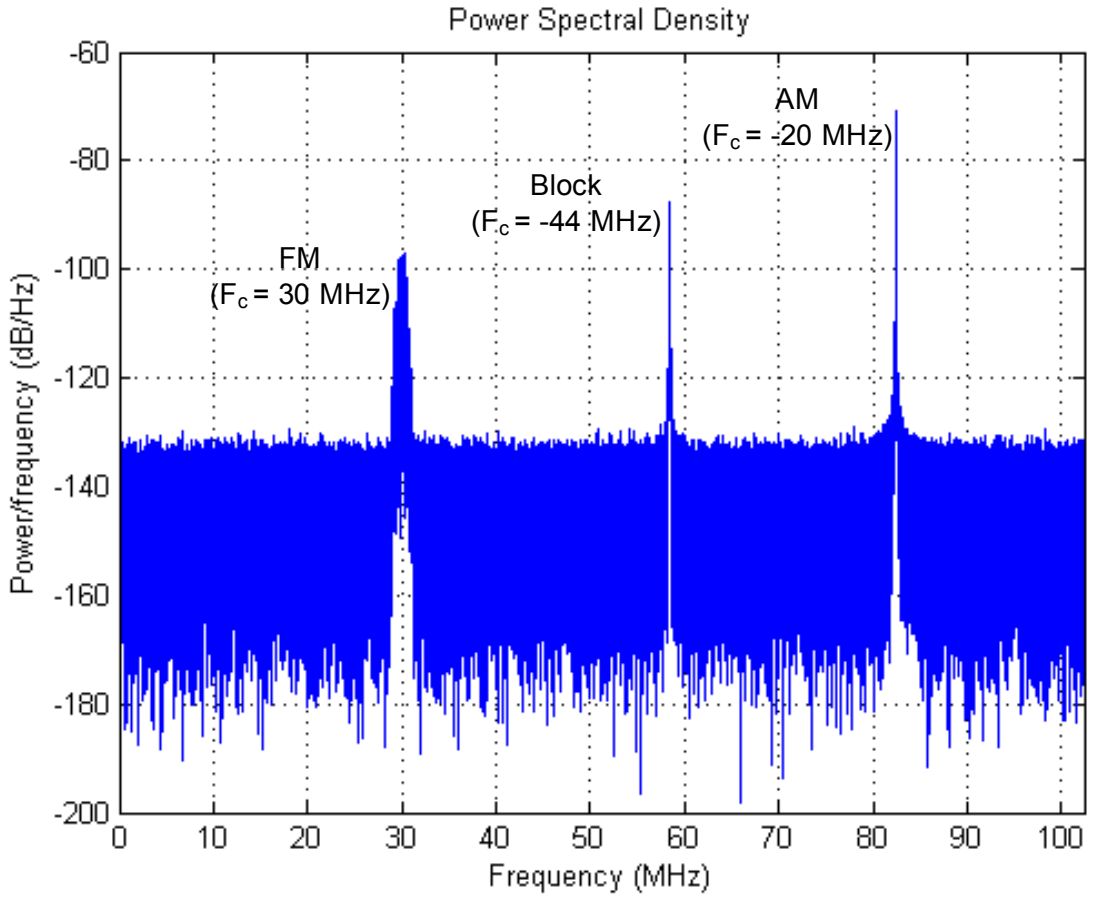


Figure 3.9: Ingress Noise Power Spectral Density Example

4. Fractional Delay Filter Design

This chapter advances a topic broached in Chapter 2. Chapter 2 laid the theoretical framework for fractional timing offsets and micro-reflections. It was mentioned that either pre-equalization coefficients or fractional delay filters (FDFs) could be used to simulate both impairments. It was stated without reason that pre-equalization coefficients were the method best suited for simulating the main-path fractional timing offset. It was also stated that a secondary path with an FDF was the better choice for simulating micro-reflections.

The purpose of this chapter is to explain these decisions by exploring the FDF design process that was used during this research. The chapter begins with a discussion of general FDF theory. Following this, two FDF design methods are presented, and a metric for comparing their performances is established. Next, techniques for implementing filter structures efficiently on a field programmable gate array (FPGA) are described in detail. The FDF designs themselves and their performance metrics are then evaluated. Finally, a cost analysis in an FPGA implementation is done. The cost metrics are number of multipliers and bytes of memory.

4.1 General Fractional Delay Filter Theory

To begin the discussion surrounding the FDF design process, the basic theory of FDFs is first explained. A fractional delay filter (FDF) is a digital filter whose purpose is to delay a signal by a fraction of a sampling period. In this research, the fractional delay is considered to be the fractional portion of the transport delay in units samples of either the main or secondary channel paths. An ideal FDF would therefore have an impulse response identical

to that of Equation 2.6, and thus a frequency response of:

$$H_{ideal}(e^{j\omega}) = e^{-j\Delta\omega} \tag{4.1}$$

where Δ is the fractional delay which can range between $-0.5 \leq \Delta \leq 0.5$ samples.

Since the function of the filter is to introduce a pure delay, the ideal magnitude response is:

$$|H_{ideal}(e^{j\omega})| = 1, \quad -\pi \leq \omega \leq \pi \tag{4.2}$$

and the ideal phase response is:

$$\angle H_{ideal}(e^{j\omega}) = -\Delta\omega, \quad -\pi \leq \omega \leq \pi \tag{4.3}$$

In other words, an ideal FDF will have an all-pass magnitude response, as well as a linear phase response with a slope equal to the negative of the fractional delay [29].

The impulse response of the ideal filter is obtained by taking the inverse discrete-time Fourier transform of the frequency response, which results in the discrete-time sinc function:

$$h_{ideal}[n, \Delta] = \frac{\sin(\pi(n-\Delta))}{\pi(n-\Delta)} = \text{sinc}(n-\Delta), \quad -\infty < n < \infty \tag{4.4}$$

In general, the impulse response has infinite length and is non-causal. Visual representations of the impulse response, which is also the coefficients of finite impulse response (FIR) filter implementations, are provided in Figure 4.1. The top and bottom plots of the figure show the impulse responses for fractional delays of $\Delta = 0$ and $\Delta = 0.25$ samples, respectively.

It is clear from Figure 4.1 that delaying the signal by a fraction of a sample results in an infinite number of non-zero coefficients, some of which are at $n < 0$. This means that the ideal filter is non-causal, and thus non-realizable (i.e. unable to operate in real-time). A realizable FDF must have a finite-length, causal impulse response. One approach to obtaining such a response is to approximate the ideal response. The approximation should be made using a method that in some way minimizes the following error function:

$$E(e^{j\omega}) = H_{FD}(e^{j\omega}) - H_{ideal}(e^{j\omega})e^{-j\omega\frac{N-1}{2}} \tag{4.5}$$

where $H_{FD}(e^{j\omega})$ is the frequency response of the finite-length, causal FDF; N is the length of the FDF; and $e^{-j\omega\frac{N-1}{2}}$ accounts for a delay that places the majority of the energy in the

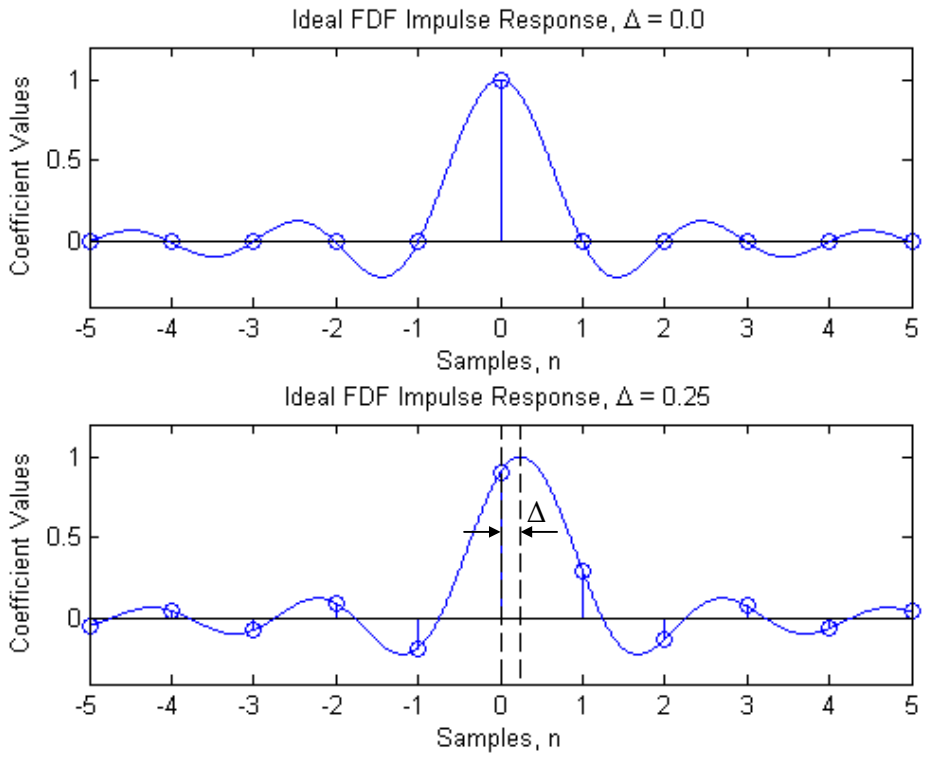


Figure 4.1: Ideal FDF Impulse Response

sinc function in the interval $0 \leq n \leq N-1$ [29] [30]. The best approximation is one that minimizes Equation 4.5 using the fewest resources possible. The most important resources in an FPGA are multipliers and bytes of memory.

According to Laakso et. al. [31], there are numerous techniques that can be used to accomplish this objective, each of which minimizes the error function in a different way. The techniques can be divided into two main categories: finite impulse response (FIR), infinite impulse response (IIR). However, since the impulse response of an IIR filter may potentially extend beyond the cyclic prefix and interfere with the data of an orthogonal frequency-division multiple access (OFDMA) frame, only FIR filter designs are considered in this research.

Two FIR design methods were selected for evaluation: the windowing method, and the maximally-flat method, both of which are non-iterative and have closed form solutions [31]. Derivations of both techniques are provided in the following section.

4.2 Design Methods

4.2.1 Windowing Method

The windowing method of designing an FIR FDF is straightforward. To understand the basic concepts of windowing, a least squared error approach is often taken [29] [30] [31], which requires modifying Equation 4.5 to become:

$$E_{LS} = \frac{1}{\pi}\int_0^{\pi} |H_{FD}(e^{j\omega}) - H_{ideal}(e^{j\omega})e^{-j\omega\frac{N-1}{2}}|^2 d\omega \tag{4.6}$$

The simplest way to minimize the least squared error is by truncating the delayed sinc function such that:

$$h_{FD}[n,\Delta] = \begin{cases} \text{sinc}(n - \frac{(N-1)}{2} - \Delta) & n = 0, 1, ..., N-1 \\ 0 & otherwise \end{cases} \tag{4.7}$$

The sinc function has been delayed by $(N-1)/2$ so that most of the energy in the truncated sinc function lies within the interval $0 \leq n \leq N-1$.

A side effect of the truncation process is the introduction of passband ripple in the frequency domain [32]. Depending on the filter's performance metric, it may be desirable to mitigate the passband ripple. This can be accomplished by applying discrete-time window function, $w[n,\Delta]$, to the truncated sinc function such that:

$$h_{FD}[n,\Delta] = \begin{cases} \text{sinc}(n - \frac{(N-1)}{2} - \Delta)w[n,\Delta] & n = 0, 1, ..., N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

This windowing process effectively tapers the truncated sinc function in the time domain, which translates to a smoother passband and a wider transition band in the frequency domain. Consequently, the least squared error will also be increased; however, this is not necessarily a drawback depending on the portion of the passband over which the performance is measured.

The degree to which the truncated sinc function is tapered depends on the type of window used. Five types of windows are explored in this research. The first four are the Rectangular,

Hanning, Hamming, and Blackman windows. Their respective equations are provided in [32] and listed below:

Rectangular

$$w[n,\Delta] = \begin{cases} 1 & n = 0, 1, ..., N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

Hanning

$$w[n,\Delta] = \begin{cases} 0.5\left(1 - \cos\left(\frac{2\pi(n - \frac{(N-1)}{2} - \Delta)}{N-1}\right)\right) & n = 0, 1, ..., N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

Hamming

$$w[n,\Delta] = \begin{cases} 0.54 + 0.46\cos\left(\frac{2\pi(n - \frac{(N-1)}{2} - \Delta)}{N-1}\right) & n = 0, 1, ..., N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

Blackman

$$w[n,\Delta] = \begin{cases} 0.42 - 0.5\cos\left(\frac{2\pi(n - \frac{(N-1)}{2} - \Delta)}{N-1}\right) + ... \\ \qquad 0.08\cos\left(\frac{4\pi(n - \frac{(N-1)}{2} - \Delta)}{N-1}\right) & n = 0, 1, ..., N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

It is important to note that applying a Rectangular window to the sinc function is the same as truncating it, which, as previously stated, results in more passband ripple than the other windows. Thus, using a Rectangular window is generally not considered acceptable in practice [29] [31], and its inclusion in this thesis is purely for reference.

Aside from the fractional delay parameter Δ, the Rectangular, Hanning, Hamming, and Blackman windows have only a single adjustable parameter, which is their length, N. This makes them somewhat limited from a design standpoint; however, they are simple to compute, making them easily implementable in a practical setting.

The fifth type of window explored is the Kaiser window, also given by [32] as:

Kaiser

$$w[n,\Delta] = \begin{cases} \frac{I_0\left(\beta\sqrt{1-(\frac{2(n-\frac{(N-1)}{2}-\Delta)}{N-1}-1)^2}\right)}{I_0(\beta)} & n = 0,1,...,N-1 \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

where $I_0(\cdot)$ refers to a Bessel function of the first kind.

The Kaiser window is more versatile than the other four windows as it has an additional adjustable parameter, β, which controls the shape of the window. In fact, by selecting an appropriate β value, the Kaiser window can approximately model all four of the previously mentioned window types (with $\beta = 0$ corresponding to a Rectangular window). The trade-off is computational complexity of the coefficients due to their dependency on the Bessel function.

4.2.2 Maximally-Flat Method (Lagrange Interpolation)

Another way of minimizing the error function of Equation 4.5 is to make it "maximally-flat" at frequency $\omega_0 = 0$. For an N-tap FIR filter, this can be achieved by ensuring that the first $N-1$ derivatives of $E(e^{j\omega})$ are equal to zero at $\omega = 0$ [29] [31] [33]. That is to say:

$$\frac{\delta^n E(e^{j\omega})}{\delta\omega^n}\Big|_{\omega=0} = 0, \quad n = 0,1,2,...,N-1 \tag{4.14}$$

Performing N differentiations on the above equation produces a series of N linear equations. According to [29], [31], and [33], these linear equations can be solved in terms of the filter's impulse response as follows:

$$h_{FD}[n] = \prod_{\substack{k=0 \\ k\neq n}}^{N-1} \frac{\frac{N-1}{2}+\Delta-k}{n-k}, \quad n = 0,1,2,...,N-1 \tag{4.15}$$

which is equivalent to the classical Lagrange interpolation method given in Equation 4.16, where $y(x_n)$ is the value of a sample at index x_n. This means that each filter coefficient

$h_{FD}[n]$ is a Lagrange-basis polynomial $p_n(x)$ evaluated at $x = (N-1)/2 + \Delta$.

$$\begin{aligned} P(x) &= \sum_{n=0}^{N-1} p_n(x) y_(x_n) \\ p_n(x) &= \prod_{\substack{k=0 \\ k \neq n}}^{N-1} \frac{x - x_k}{x_n - x_k} \end{aligned} \tag{4.16}$$

Thus, filtering a set of uniformly spaced samples using a FIR FDF of length N that is maximally-flat about $\omega = 0$ is the same as interpolating the set of samples using a fitted polynomial $P(x)$ in the Lagrange form of order $N-1$.

Similar to the windowed sinc filters mentioned above (excluding the Kaiser window), the only adjustable parameter of maximally-flat filters is their length N. Thus, while being limited from a design standpoint, their coefficients are also simple to compute, making them easily implementable in practice.

4.3 Comparison Metric

Since both the windowing and maximally-flat methods minimize the error function of Equation 4.5 in different ways, their standard performance criteria cannot be compared against one another. Thus, to compare both methods, a universal performance metric must be used instead. In order to determine which performance metric is best suited to this research, several aspects of the data-over-cable service interface specification (DOCSIS) 3.1 standard are revisited.

First of all, the bandwidth of a DOCSIS 3.1 OFDMA signal at radio-frequency (RF) is 102.4 MHz. The first and last 3.7 MHz on either side of the channel are designated as guard bands, meaning that the sub-carriers that lie within this range are excluded. This leaves the signal with a usable bandwidth (passband) of 95 MHz, making it a wideband signal. At baseband, this usable bandwidth is halved. Thus, from a filter design perspective, the FDF should have a worst-case passband corner located at $f_p = 95/102.4/2 = 0.463867187$ cycles/sample, and a worst-case transition band that spans from f_p to 0.5.

Second, the minimum carrier-to-noise ratio (CNR) specified in the DOCSIS 3.1 standard

is 43 dB, which is required in order for a cable modem termination system (CMTS) to decode 4096-QAM [12]. To ensure that this CNR is achievable, the filters should be designed in such a way that any "noise" they introduce to the system does not have a significant effect on the overall CNR. Thus, the metric for determining a filter's performance is considered to be "filter noise".

For the purposes of this research, "filter noise" is defined as the worst-case mean squared error (MSE) in a filter's passband, which occurs when the fractional delay $\Delta = 0.5$. Mathematically, the worst-case MSE is given by:

$$\text{MSE}_{\text{worst-case}} = \frac{1}{2\pi}\int_{-\omega_p}^{\omega_p} |S_{xx}(e^{j\omega})||H_{FD}(e^{j\omega},\Delta) - H_{ideal}(e^{j\omega},\Delta)e^{-j\omega\frac{N-1}{2}}|^2 d\omega \quad , \quad \Delta = 0.5 \tag{4.17}$$

where $S_{xx}(e^{j\omega})$ is the power spectral density (PSD) of the input signal, $H_{FD}(e^{j\omega},\Delta)$ and $H_{ideal}(e^{j\omega},\Delta)$ are the frequency responses of the respective designed and ideal filters, and $\omega_p = 2\pi(0.463867187)$ rads/sample is the passband corner frequency.

Strictly speaking, the power spectrum of the OFDMA signal in the analogue domain is not representative of the actual signal due to the presence of the cyclic prefix. The power spectrum of the actual signal is best represented as excerpts of this analogue signal that ignore the cyclic prefix. The excerpts are interpreted as periodic signals consisting of pure sinusoids, each of which is a subcarrier. Thus, the power spectrum of the actual signal is given by:

$$|S_{xx}(e^{j\omega})| = \sum_{n=m}^{N_{FFT}-m-1} 2\pi\delta\left(\omega - \frac{2\pi n}{N_{FFT}}\right) \tag{4.18}$$

where n represents the subcarrier index, N_{FFT} is the total number of subcarriers in the signal (2048 for 2k-mode, or 4096 for 4k-mode), and m represents the index of the first usable subcarrier (74 for 2k-mode, or 148 for 4k-mode due to the predefined DOCSIS 3.1 spectral guardband).

Equation 4.17 can therefore be simplified to:

$$\text{MSE}_{\text{worst-case}} = \frac{1}{M}\sum_{n=m}^{N_{FFT}-m-1} |H_{FD}(e^{j\frac{2\pi n}{N_{FFT}}},\Delta) - H_{ideal}(e^{j\frac{2\pi n}{N_{FFT}}},\Delta)e^{-j\omega\frac{N-1}{2}}|^2 \quad ,\Delta = 0.5 \tag{4.19}$$

where M is the total number of subcarriers in the passband (1900 for 2k-mode, or 3800 for 4k-mode, again due to the guardband).

Based on the above definition, it was decided that the total worst-case MSE of any combination of filters should not exceed 0.001 (i.e. -60 dB) in order to ensure that any "filter noise" introduced into the system lies well below the minimum CNR of 43 dB. This means that if multiple filters are used, each individual filter must be designed so their worst-case MSEs are less than -60 dB. For example, if both a main-path and a secondary-path FDF are used, they might be designed so that their individual worse-case MSEs are approximately less than -63.1 dB, as $10\log_{10}(2 \times 10^{\frac{-63.1}{10}}) = -60.1 \text{ dB} \leq -60 \text{ dB}$.

Note that micro-reflections are attenuated. Taking this attenuation into account allows the MSE parameters to be relaxed when designing FDFs that model micro-reflections. Specifically, the worse-case micro-reflection attenuation given in Table 2.1 is -16 dBc; thus, the worst-case MSE of a secondary-path FDF must be less than -44 dB to ensure a total worst-case MSE of -60 dB at its output. Also note that it is more practical to apply the attenuation after the output of the secondary-path FDF so that any unnecessary headroom can be removed at the filter input.

4.4 Single-Sampling-Rate Structures

One way of reducing the length of an FIR FDF is to "narrow" the bandwidth of the signal prior to implementing the fractional delay and then "expand" it back its original bandwidth after the signal has been fractionally delayed. This is a multirate signal processing technique. Since the error introduced by an FIR FDF designed using any of the aforementioned techniques is largest near the end of its passband, reducing the size of the passband would consequently reduce the passband MSE. Thus, a shorter-length filter that less-accurately approximates an ideal FDF in the original passband could be used to meet the original MSE criteria in the reduced passband. It will ultimately be shown that one of the most efficient ways to implement this filter is to incorporate it into a multirate structure, which can then be converted to a "single-sampling-rate structure" using "polyphase decomposition."

4.4.1 Multirate Theory

The basic concept of a multirate FDF system is illustrated in Figure 4.2. It is convenient to think of the system as having three stages, the first of which is interpolation. The purpose of interpolation is to increase the sampling rate of the system and consequently "compress" the spectrum of the input signal, thus allowing the FDF to operate with a smaller passband.

Interpolation is performed in two steps: upsampling, followed by low-pass filtering. The upsampling is accomplished by "zero stuffing," which inserts $L-1$ zeros between samples, with L being an integer value referred to as the "upsampling factor." This is process is represented by the first block of the interpolation circuit shown in Figure 4.2. Upsampling a discrete-time signal $x[n]$ through zero stuffing causes its spectral components to be replicated as follows:

$$X_u^{<z>}(e^{j\omega}) = \sum_{r=0}^{L-1} \frac{1}{L} X_u\left(e^{j(\omega - \frac{2\pi r}{L})}\right) \tag{4.20}$$

where $X_u(e^{j\omega})$ is the frequency response of the original upsampled signal with no images. From this equation and its corresponding visual representation in Figure 4.3, it is clear that these spectral images must be removed; hence, the signal is filtered by a low-pass filter, which is denoted as $H_{LPU}(z^L)$ in Figure 4.2.

In the second stage, the low-pass filtered, upsampled signal $x_u[k]$ is passed through the FDF $H_{FD}(z^L)$, where its samples are fractionally delayed by a factor of Δ.

The third and final stage of the multirate FDF system is decimation. Typically, decimation involves filtering the upsampled, fractionally delayed signal with another low-pass filter and then downsampling the result. However, if the low-pass filter used in the first stage is

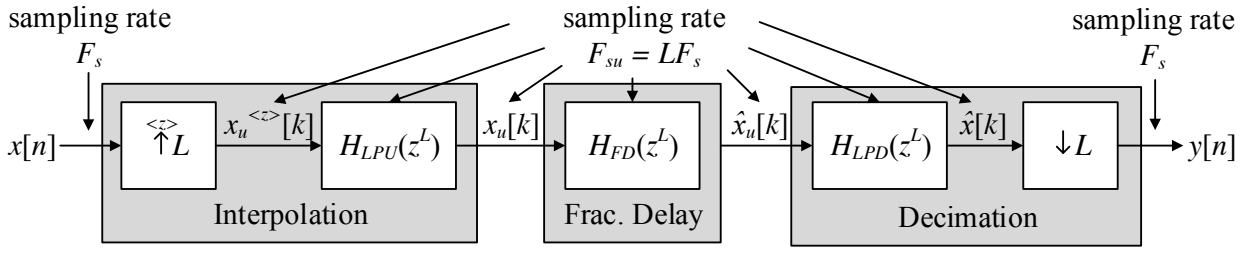


Figure 4.2: Multirate System

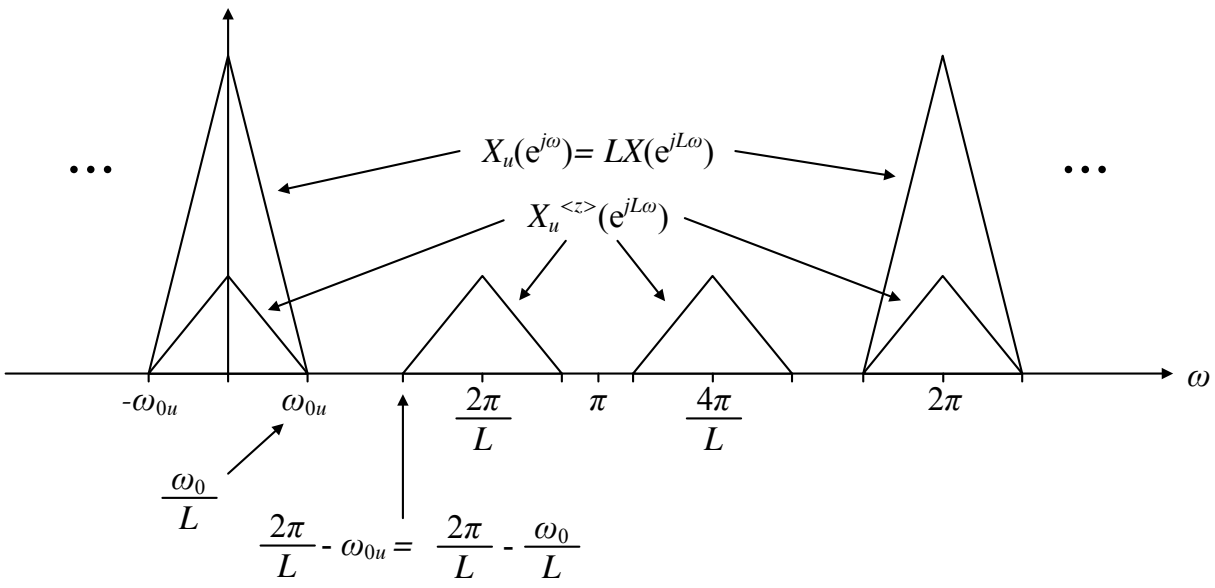


Figure 4.3: Upsampled Spectrum

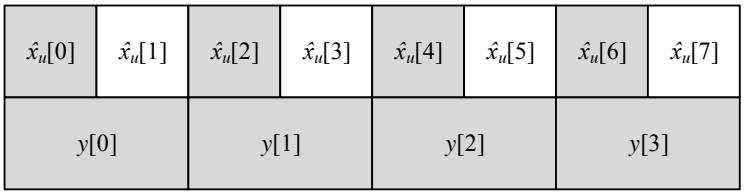


Figure 4.4: Downsampling

designed well enough, no spectral images will be present. This makes $H_{LPD}(z^L)$ redundant and unnecessary so it can be eliminated; thus, downsampling is the only operation that must be performed during decimation.

Unfortunately, implementing a system in such a fashion has several disadvantages. First of all, $H_{LPU}(z^L)$ and $H_{FD}(z^L)$ run at a rate of $F_{su} = LF_s$, which means they perform L-times more multiplications per second than a filter that operates at the original input rate of F_s. However, when the signal is downsampled, only every L^{th} sample is kept, as depicted in Figure 4.4; the remaining samples are simply discarded, which makes the system extremely inefficient where multiplication is concerned.

Additionally, it was mentioned in the problem statement that it may be desirable to

implement the MATLAB simulation on an FPGA in the near future. As it stands, the MATLAB simulation does not operate in real-time. Assuming that the FPGA implementation also does not operate in real-time, the rate at which the simulation runs on an FPGA (i.e. the FPGA's clock rate, F_{clk}) can be selected somewhat arbitrarily. F_{clk} must be set such that the highest sampling rate used by any component of the simulation is less than the FPGA's maximum clock rate, denoted as F_{max}. Since the stages of a multirate filter system operate at multiples of the sampling frequency, implementing a multirate structure on an FPGA can potentially be a limiting factor where processing speed is concerned.

To avoid multiplication inefficiencies and ensure that processing speed is not a limiting factor, a multirate structure can sometimes be converted into a single-sampling-rate structure. This typically involves breaking the filters of the multirate structure into smaller "sub-filters" through polyphase decomposition, the basic concepts of which will be discussed in the following subsection. Ultimately, it will be shown that by rearranging the upsamplers and downsamplers appropriately, each sub-filter can be made to run at a single sampling rate defined at the structure's input.

4.4.2 Polyphase Decomposition

Polyphase decomposition can be used to implement multirate filter structures more efficiently by reducing the number of multiplications per output sample of a filter. Consider first the interpolation stage of the multirate FDF system shown in Figure 4.2. The first step involved in polyphase decomposition is breaking the low-pass filter down into a number of smaller "sub-filters," the exact number of which corresponds to the upsampling factor, L. For FIR filters, each sub-filter processes the data associated with a "phase" of the output. The filter for phase "P" is obtained by using a starting index which must be between 0 and $L-1$ inclusive and then selecting every L^{th} coefficient from the original filter. Typically, the filters are then placed in parallel with one another. An integer delay is then inserted between each filter, thus forming the polyphase structure shown in Figure 4.5a. According

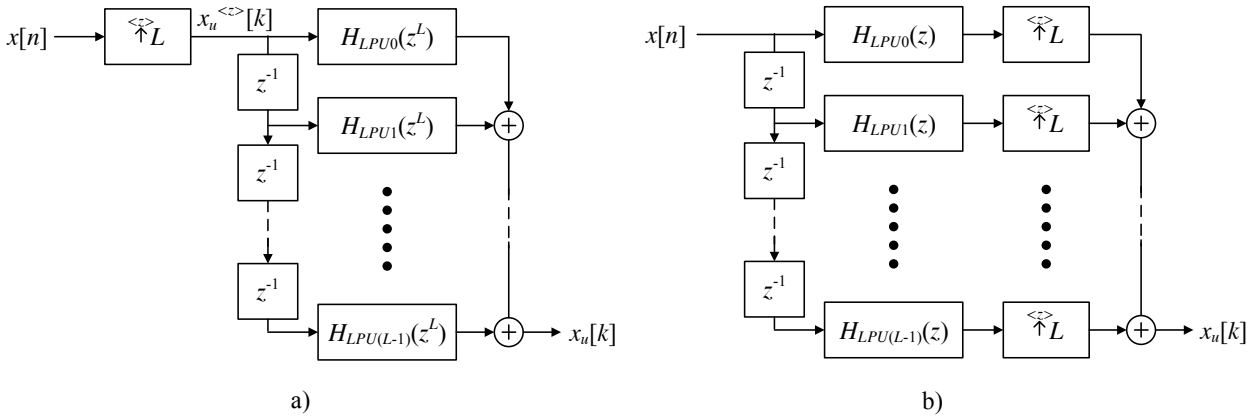


Figure 4.5: Inefficient (a) vs. Efficient (b) Structures for Performing Interpolation

to [34], this FIR polyphase structure can be expressed mathematically as:

$$H(z) = \sum_{l=0}^{L-1} z^{-l} H_l(z^L) = \sum_{l=0}^{L-1} \sum_{n=-\infty}^{\infty} h[Ln + l] z^{-(Ln+l)} \tag{4.21}$$

where l is the index of each sub-filter.

Note that all sub-filters must have the same delay in order to ensure that their samples arrive at the output at the correct time. Thus, the outputs of any sub-filters that are shorter than the longest polyphase filter must be padded with integer delays until their total delay matches that of the longest sub-filter.

Each sub-filter runs at a rate of $F_{su} = LF_s$. Since upsampling with zero stuffing occurs before filtering, it is inevitable that some filters will perform wasteful multiplication on the zero-valued samples. However, using a noble identity [35] which states that an upsampler followed by a system function that is a function of z^L, i.e. $H(z^L)$, is equivalent to system function $H(z)$ followed by an upsampler, the upsampler can be shifted to the other side of the sub-filters as shown in Figure 4.5b. The filters can now run at the original input rate of F_s. This reduces the number of multiplies required to produce each output sample by a factor of L, thus making the system L-times more efficient.

In a similar fashion, polyphase decomposition can be performed on the fractional delay and decimation stages of Figure 4.2. Here, the polyphase filter has "M" phases corresponding to the downsampling factor M, which in this case is equal to the upsampling factor L. Recall

that the low-pass filter $H_{LPD}(z^L)$ is redundant and can be eliminated. Breaking $H_{FD}(z^L)$ into L sub-filters and inserting delay z^P between the filter input and the input of phase P produces the polyphase structure shown in Figure 4.6a.

Once again, the filters operate at a rate of $F_{su} = LF_s$. Since only every L^{th} sample is retained by the downsampler, each filter is performing $L - 1$ wasteful multiplications. Shifting the downsampler to the other side of the sub-filters as shown in Figure 4.6b allows them to operate at the desired output rate of F_s, which again improves the multiplication efficiency of the structure by a factor of L. The end result is the fractionally delayed output signal, $y[n]$.

At this point, two key structures have been established: the interpolation structure of Figure 4.5b and the fractional delay/decimation structure of Figure 4.6b. By conceptually placing these structures side-by-side, it is clear that the upsamplers and downsamplers along each path will effectively cancel each other out. Thus, the single-sampling-rate structure of Figure 4.7 can be obtained.

The challenge now is to design a FIR low-pass filter $H_{LPU}(z^L)$ and a FIR FDF $H_{FD}(z^L)$ that, when implemented in the aforementioned single-sampling-rate structure, will use less computational resources than standard FDF. Additionally, the structure must be designed so that it is capable of providing a full range of fractional delay values (i.e. $-0.5 \leq \Delta \leq 0.5$

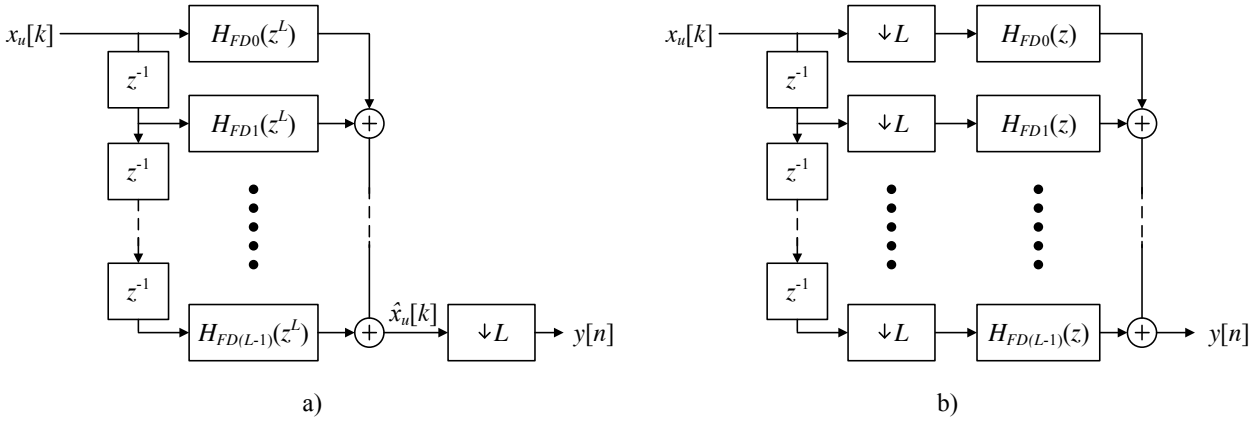


Figure 4.6: Inefficient (a) vs. Efficient (b) Structures for Inducing a Fractional Delay and Performing Decimation

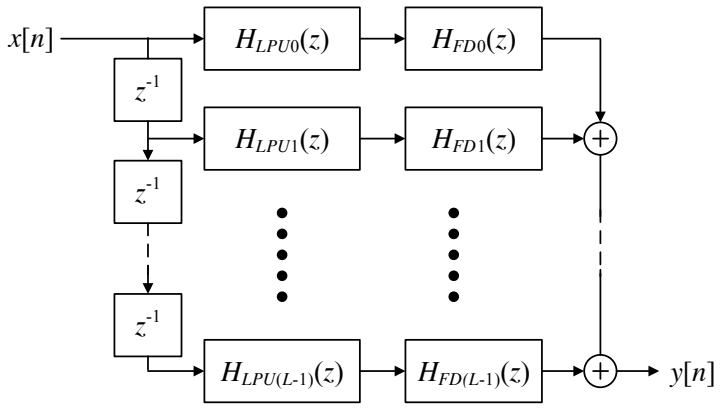


Figure 4.7: Single-Sampling-Rate Structure

samples).

4.4.3 Interpolating with Halfband Filters

The interpolation stage of the multirate structure of Figure 4.2 can be implemented using a low-pass FIR halfband filter [36] [37] [38], as depicted in Figure 4.8a. When it is implemented in this fashion, the low-pass FIR halfband filter is able to upsample a signal by a factor of $L = 2$ while simultaneously suppressing any spectral images.

To achieve this implementation, the FIR halfband filter must be decomposed into its polyphase filter components. The FIR halfband filter is unique in that every second coefficient as measured from both sides of the centre tap is a zero, the centre tap having a value of 0.5. Thus, its polyphase filter components consist of a symmetric low-pass filter and an integer delay path, which are respectively denoted as $H_0(z^2)$ and $z^{-N_{HB}}$ in Figure 4.8a. Both filters can be made to run at the input sampling rate by rearranging the structure so that upsampling is performed at the filter outputs, which is illustrated in Figure 4.8b. The rearranged structure can be made more efficient by replacing the upsamplers and the adder with a single commutator at the filter's output that runs at twice the input rate, as shown in Figure 4.8c [38]. Note that there would normally be a gain at the filter output to compensate for any filter attenuation; however, for the purposes of this research, it is assumed that all filter coefficients have been normalized with respect to the center tap.

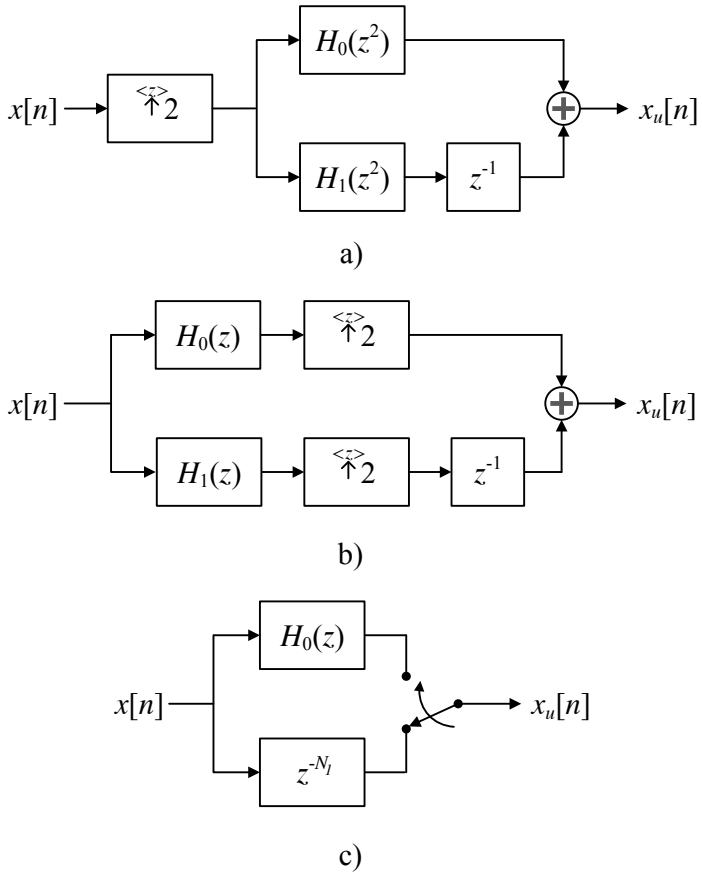


Figure 4.8: Multirate Halfband Upsampler

Implementing a FIR halfband filter in a single-sampling-rate structure requires that the commutator be removed from the output. This results in two streams of data at the filter's output, as depicted in Figure 4.9. The first stream, denoted as $u_0[n]$, is the output of filter $H_0(z)$, which is of length is $(N_{HB} + 1)/2$, where N_{HB} is the length of the original FIR halfband filter. The second stream, denoted as $u_1[n]$, is the original input signal $x[n]$ delayed by an a factor of $N_1 = ((N_{HB} - 1)/2 - 1)/2$ samples. Since FIR halfband filters are always of odd length, $H_0(z)$ will always be of even length. Thus, $u_0[n]$ will be delayed by a factor of 0.5 samples relative to u_1. Both streams of data can then be connected to an FDF that has been decomposed into $L = 2$ phases.

Another advantage of the FIR halfband filter is that it can be cascaded with other FIR halfband filters to increase the upsampling rate by powers of 2 (i.e. $L = 4, 8, 16, ...$) [38].

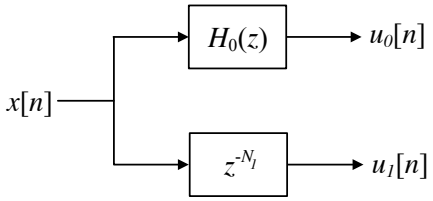


Figure 4.9: Single-Sampling-Rate Halfband Upsampler

With each subsequent cascade stage, the width of the transition band also increases by a factor of L; thus, each subsequent filter requires fewer coefficients to implement, up to a minimum of filter length of $N_{HB} = 5$.

Note that implementing a cascade of FIR halfband filters in a single-sampling-rate structure becomes significantly more challenging with each additional stage in the cascade, as the structure must divide the signal into L streams at each stage. An example showing how a FIR halfband cascade can be implemented as a single-sampling-rate structure to upsample a signal by a factor of $L = 4$ is provided in Appendix B.

4.4.4 Polyphase FDF Structure and Downsampling

As discussed in Section 4.4.2, fractional delay and decimation can be performed by implementing the FDF in the same fashion as the polyphase structure of Figure 4.7. Up until this point, it has been assumed that the structure is "fixed" in the sense that the input to each sub-filter in the FDF polyphase filter structure is tied to the output of a specific sub-filter of the interpolator. This results in an output signal $y[n]$ that is delayed by some fixed fractional amount of samples. The purpose of the FDF, however, is to be able to fractionally delay $y[n]$ by any value in the range of $-0.5 \leq \Delta \leq 0.5$ samples. Achieving this full range of Δ values in a single-sampling-rate structure requires that the FDF input streams be reconnected in specific ways; thus, switching logic must be added to the structure.

To understand why this is the case, consider a generalized single-sampling-rate structure that utilizes an upsampling/downsampling factor of L. The output of the interpolator is partitioned into L "polyphase streams," which, following the notation used in Appendix B

are denoted respectively as $\alpha_{-b/L}[n]$, $0 \leq b \leq L-1$. In this case, the total value of the subscript (i.e. $-b/L$) represents the amount by which each stream is delayed in samples relative to the original sample rate of F_s. When combined using the method described below, these streams form the interpolated signal $\hat{x}_u[k]$.

$$\begin{aligned}
\hat{x}_u[0] &= \alpha_0[0] \\
\hat{x}_u[-1] &= \alpha_{-1/L}[0] \\
\hat{x}_u[-2] &= \alpha_{-2/L}[0] \\
\vdots &= \vdots \\
\hat{x}_u[-L+1] &= \alpha_{-(L-1)/L}[0]
\end{aligned}$$

Assume for the moment that there is no FDF present in the system. Recall that downsampling in a multirate structure is performed by selecting every L^{th} sample of $\hat{x}_u[k]$, such that $y[n] = \hat{x}_u[nL]$. If the desired output is a set of interpolated $x[n]$ values, $\hat{x}_u[k]$ need only be delayed by some integer b so that $y[(n+b)/L] = \hat{x}_u[(n+b)L]$. This can alternatively be thought of as setting the output $y[n]$ equal to one of the interpolator output/FDF input streams $\alpha_{-k/L}[n]$ in a single-sampling-rate structure.

Now consider a system with an FDF that has been decomposed into L sub-filters. Each of the interpolator's output streams is connected to the input of one of the sub-filters. The coefficients of each sub-filter are calculated based on a given fractional delay value, Δ, in units samples. The sub-filter outputs are then added together to produce a fractionally delayed, "downsampled" (decimated) output signal, $y[n]$. This method of decimation is more efficient than running the FDF at a higher sampling rate and then downsampling by L because it reduces the number of multiplications per second while simultaneously making use of each sample (i.e. no samples are "discarded").

However, as a consequence of breaking the FDF up into L smaller filters, the actual range of Δ values that the FDF is capable of applying to a given signal is reduced by a factor of L. In other words, the FDF can only apply a delay of $-1/L \leq \Delta \leq 1/L$ to any given output stream. Producing a full range of Δ values therefore requires that the filter be capable of switching between different output streams. Note that the inherent delay of the

single-sampling-rate structure (i.e. the total delay excluding Δ) will affect where the chosen output stream appears relative to an output stream with no fractional delay.

For example, if $L = 2$, there will be two streams: $\alpha_0[n]$ and $\alpha_{-1/L}[n]$. If the inherent delay of the system is such that stream $\alpha_0[n]$ is not offset by any default fractional amount, the fractional delay range over which $\alpha_0[n]$ can operate will be from $-0.25 \leq \Delta \leq 0.25$ samples. The range of δ for $\alpha_{-1/L}[n]$, on the other hand, will be from $-0.75 \leq \Delta \leq -0.25$. To operate in the remaining range of $0.25 \leq \Delta \leq 0.5$ samples, each output stream need only be delayed by a single integer amount.

The last item that must be discussed is how a specific stream $\alpha_{-k/L}[n]$ can be selected at the output. To select a specific stream, it must be connected to filter $H_0(z)$. All other streams must be connected to the remaining filters in a way that ensures their samples are arranged "forward" in time. That is to say, if stream $\alpha_0[n]$ is the input to filter $H_0(z)$, then the input to $H_1(z)$ must be $\alpha_{-(L-1)/L}[n]$ delayed by one sample, the input to $H_2(z)$ must be $\alpha_{-(L-2)/L}[n]$ delayed by one sample, and so on and so forth, with the final filter $H_{L-1}(z)$ having the input $\alpha_{-1/L}[n]$ delayed by one sample. Conversely, if $\alpha_{-(L-1)}[n]$ is the input to $H_0(z)$, then the input to $H_1(z)$ must be $\alpha_{-(L-2)/L}[n]$, the input to $H_2(z)$ must be $\alpha_{-(L-3)/L}[n]$, and so on and so forth, with the final filter $H_{L-1}(z)$ having input $\alpha_0[n]$. Both scenarios are depicted in Figures 4.10a and 4.10b, respectively. Thus, there are a total of L possible ways to connect the interpolator output streams to the sub-filter inputs of the FDF polyphase filter structure.

In summary, an FIR FDF can be implemented as a single-sampling-rate structure through the use of polyphase decomposition. Additional logic that controls switching between sub-filter input streams must be included in order to obtain a full fractional delay range of $-0.5 \leq \Delta \leq 0.5$ samples. The logic must also be able to change the filter coefficients in accordance with the desired fractional delay value. The final structure will look similar to the one depicted in Figure 4.11.

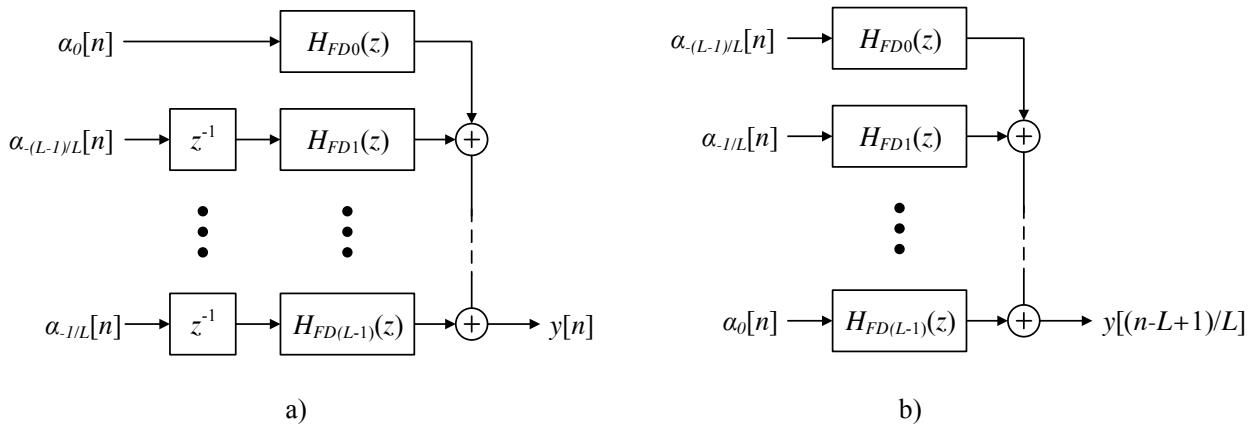


Figure 4.10: Connection Sequences Required to Obtain Streams a) $\alpha_0[n]$ and b) $\alpha_{-(L-1)}[n]$ at the Structure Output

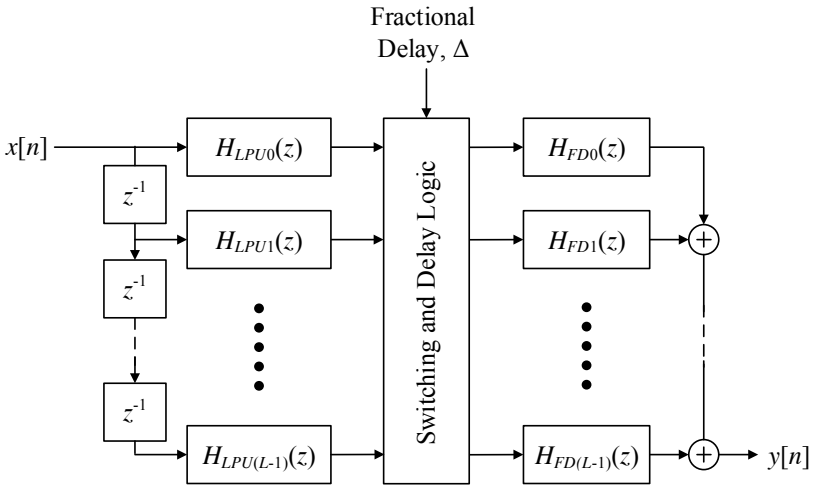


Figure 4.11: Single-Sampling-Rate Structure with Delay and Switching Logic

4.5 Filter Design Process

Filters are designed to minimize some performance measure or another. In this thesis, the filters are designed to minimize the MSE with "equal" weighting across the passband. The error in the transition and stop bands is ignored. Referring back to Section 4.3, this means that if two filters were designed to achieve a combined MSE of -60 dB, their individual MSEs were set to approximately half of the combined MSE (i.e. -63.1 dB each). Similarly, if three filters were used, their individual MSEs were each set to approximately -64.8 dB.

Note that in the case of a halfband filter, the filter's MSE is considered to be the sum of its passband and stopband MSEs. This is because the halfband filter is used to upsample a signal while simultaneously suppressing any spectral images. When the signal is downsampled, the suppressed spectral images will recombine with the original signal, thus producing additional error.

4.5.1 Halfband Filter Design

The FIR halfband filters used in this research were all designed using MATLAB's `fdesign.halfband` function. The function designs the filter using either an equiripple or Kaiser-windowed method, and optimizes it with respect to to one of three parameters: stopband attenuation, transition width, or filter length. Obviously, two parameters must be specified in order to optimize for the third.

An equiripple design method was selected so that the MSE in the stopband was identical to that in the passband, which made it easier to control the total MSE. For each filter, the `fdesign.halfband` function was given a transition bandwidth and stopband attenuation (the additive inverse of the desired passband MSE in dB) as inputs in order to optimize the filter length required to meet the previously specified passband MSE criteria.

4.5.2 Fractional Delay Filter Design

In order to determine the shortest-length filter capable of meeting the specified MSE criteria, a MATLAB script was used. The script compared the MSEs of filters designed using the Rectangular, Hanning, Hamming, Blackman, and Kaiser windowing methods, as well as the maximally-flat filter with coefficients obtained from the Lagrange interpolation formula, against increasing filter length, N, for a specified fractional delay, Δ, and upsampling rate, L. In the case of the Kaiser window, the shape parameter, β, was selected through trial and error until the optimal filter length was determined. Only odd-length filters were considered so that their fixed delay was an integer value (i.e. $D = (N-1)/2$ is an integer). The fractional delay was selected to be $\Delta = 0.5$ samples, which, as previously mentioned, produces the worst-case MSE scenario for all odd-length filters. Upsampling rates of $L = 1, 2$ and 4

were chosen ($L = 1$ corresponding to no upsampling). Plots of the MSE versus the filter length for all methods are provided in Figures 4.12 through 4.17. The MSE thresholds of the plots that utilize upsampling were set in accordance with the "equal weighting" parameter. Specifically, $L = 2$ requires one halfband filter and one FDF, thus the MSE threshold is $\approx$ -63.1 dB (-47.1 dB for the secondary path). $L = 4$ requires two halfband filters and one FDF, thus the MSE threshold is $\approx$ -64.8 dB (-48.8 dB for the secondary path).

4.6 Performance Evaluation and Cost Analysis

It is clear from the plots in Figures 4.12 and 4.15 that an increase of filter length N corresponds to a decrease in passband MSE. The rate at which this decrease occurs is filter-specific. For the windowed sinc functions, this decrease is proportional to the amount that each window tapers the sinc function in the passband. For the maximally-flat filter whose coefficients were obtained from the Lagrange interpolation formula, the decrease is proportional to the interpolation error.

Note that since the maximally-flat filter is maximally-flat about 0 cycles/sample, its frequency response tapers off significantly at the passband limit of 0.463867187 cycles/sample; hence, the MSE is very high for the maximally-flat filter in Figures 4.12 and 4.15. Figures 4.13, 4.14, 4.16, and 4.17, however, demonstrate that upsampling a signal prior to applying a fractional allows FDFs to be designed so that they meet the MSE criteria in a reduced passband using fewer coefficients. Upsampling the signal by larger factors is therefore particularly advantageous for minimizing the length of maximally-flat filters, as Figures 4.14 and 4.17 demonstrate.

Based on the results obtained from Figures 4.12 through 4.17, three possible design candidates for both the main and secondary path FDFs were selected:

1. Kaiser-Windowed Sinc Function, No Upsampling
2. Kaiser-Windowed Sinc Function, Single-Sampling-Rate Polyphase Structure, $L = 2$, Equiripple Halfband

3. Maximally-Flat (Lagrange Interpolation), Single-Sampling-Rate Polyphase Structure, $L = 4$, Equiripple Halfband

Tables 4.1 and 4.2 provide the MSE performance, cost in terms of multipliers, and estimated FPGA memory usage for the main and secondary path filter designs, respectively.

The multiplier cost analysis assumes that the coefficients are pre-calculated and stored in FPGA memory; thus, the total number of multipliers required in any design is assumed to be equal to the combined length of all filters in the design. Coefficient symmetry is accounted for when applicable, which significantly reduces the number of multipliers required to implement halfband filters. All multipliers are assumed to be of size 18 x 18 bits.

The memory cost analysis provides an estimate of the number of memory blocks that may be used to store coefficients. In this analysis, one memory block is considered to be an M10K block on an ALTERA FPGA, which contains 10240 bits of memory. It is assumed that each coefficient will use 18-bits of memory on an FPGA, plus 2 parity bits, which allows for a total of 512 coefficients to be stored on each M10K blocks. Since the DOCSIS 3.1 time stamp is only accurate up to a value of 2^{-8} samples [15], the maximum fractional delay resolution implementable on an FPGA is considered to be 2^{-8} samples. Each filter is therefore considered to have a maximum of 2^8 total possible coefficient implementations, which reduces to $2^7 - 1$ unique coefficient implementations when FIR symmetry is accounted for. That is to say, the impulse responses of FIR filters that correspond to a delay between -0.5 to 0 samples will simply be mirror images of those that correspond to a delay between 0 to 0.5 samples; thus, their coefficients are redundant and need not be stored.

4.7 Results

Ultimately, the most cost-effective main-path FDF proved to be Design #2, which was the Kaiser-windowed sinc function with roll-off parameter $\beta = 4.970$ implemented as a single-sampling-rate structure. This design involves upsampling the input by a factor of $L = 2$ via a FIR halfband filter. The total passband MSE of the design was -60.7 dB. A total of 40 multipliers and 6 M10K memory blocks are required for implementation.

For the secondary-path FDF, the most cost effective design was also Design #2, which was the Kaiser-windowed sinc function with roll-off parameter $\beta = 3.056$ implemented as a single-sampling-rate structure. Again, this design involves upsampling the input by a factor of $L = 2$. The total passband MSE of the design was -45.5 dB, which translates to -61.5 dB with a worst-case micro-reflection with power -16 dBc. A total of 29 multipliers and 15 M10K memory blocks are required for implementation.

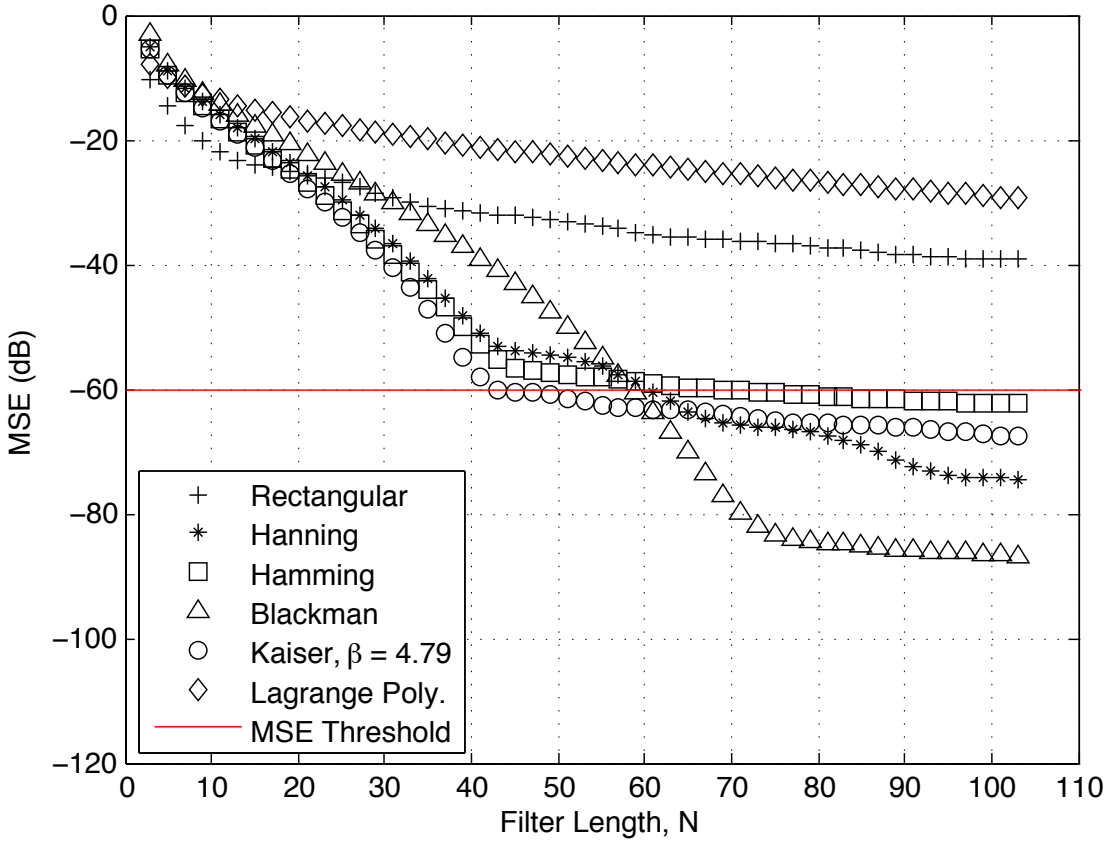


Figure 4.12: Main Path: Worst Case MSE ($\Delta = 0.5$ samples) vs. Odd Filter Length, No Upsampling

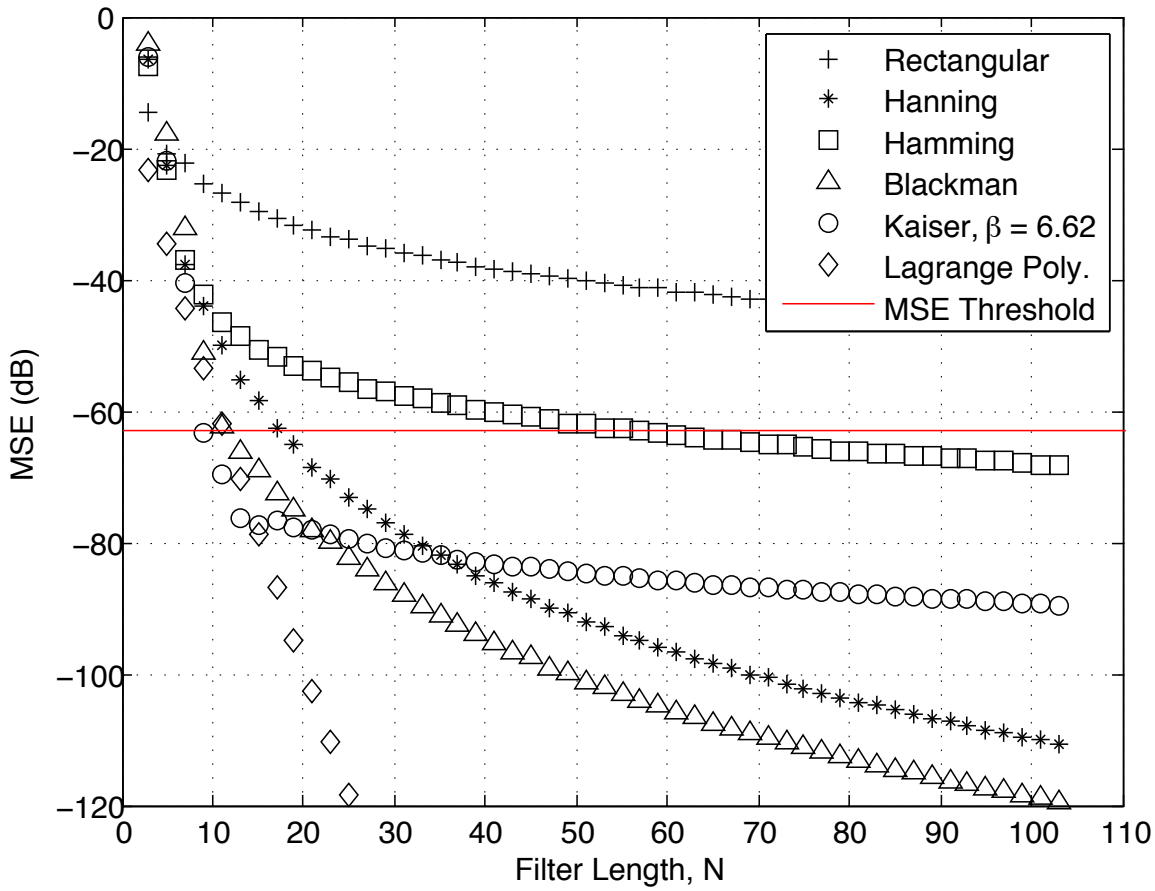


Figure 4.13: Main Path: Worst Case MSE ($\Delta = 0.5$ samples) vs. Odd Filter Length, Signal Upsampled by $L = 2$

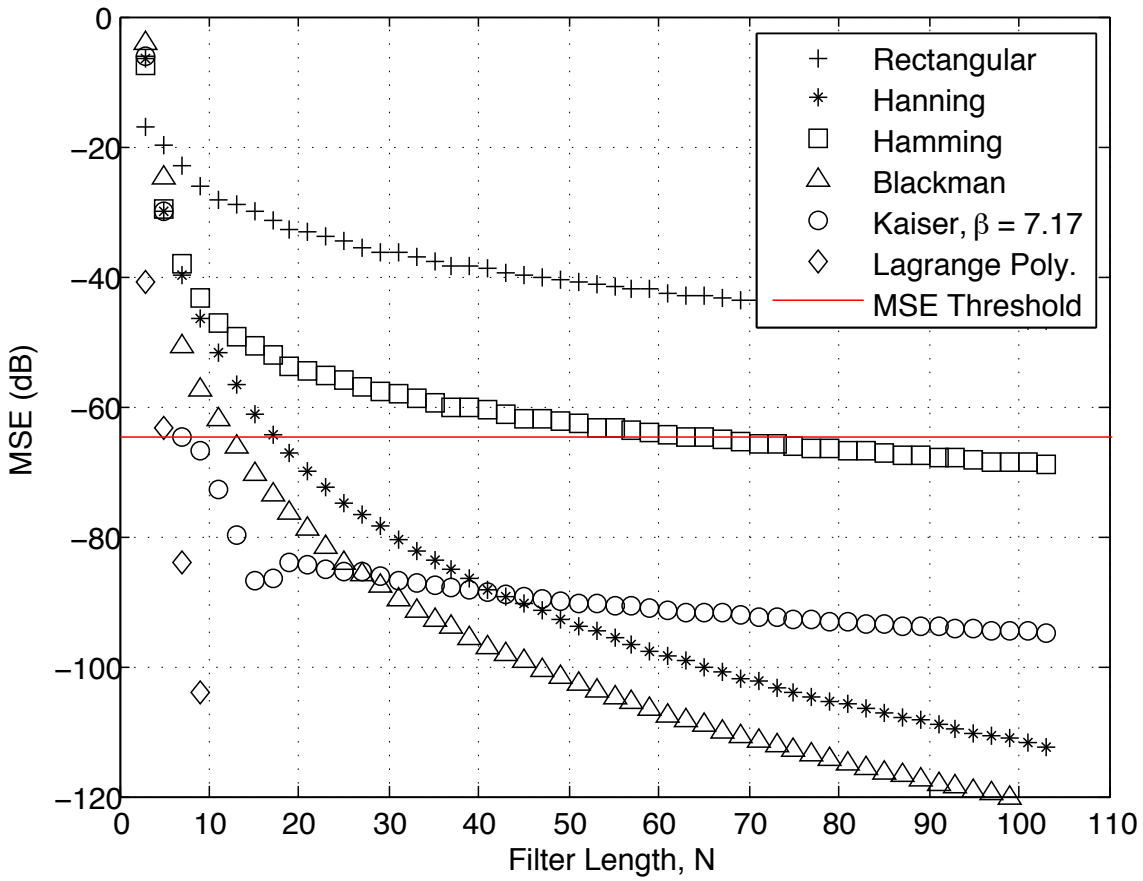


Figure 4.14: Main Path: Worst Case MSE ($\Delta = 0.5$ samples) vs. Odd Filter Length, Signal Upsampled by $L = 4$

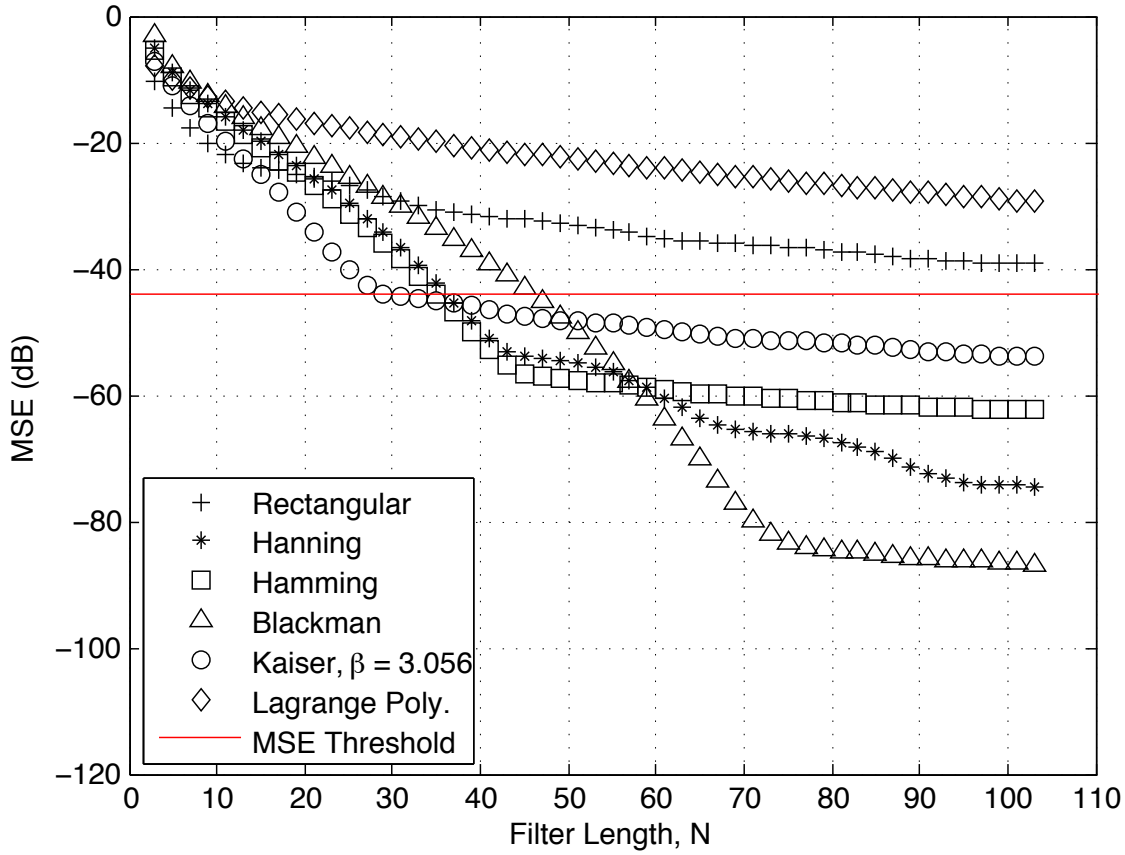


Figure 4.15: Secondary Path: Worst Case MSE ($\Delta = 0.5$ samples) vs. Odd Filter Length, No Upsampling

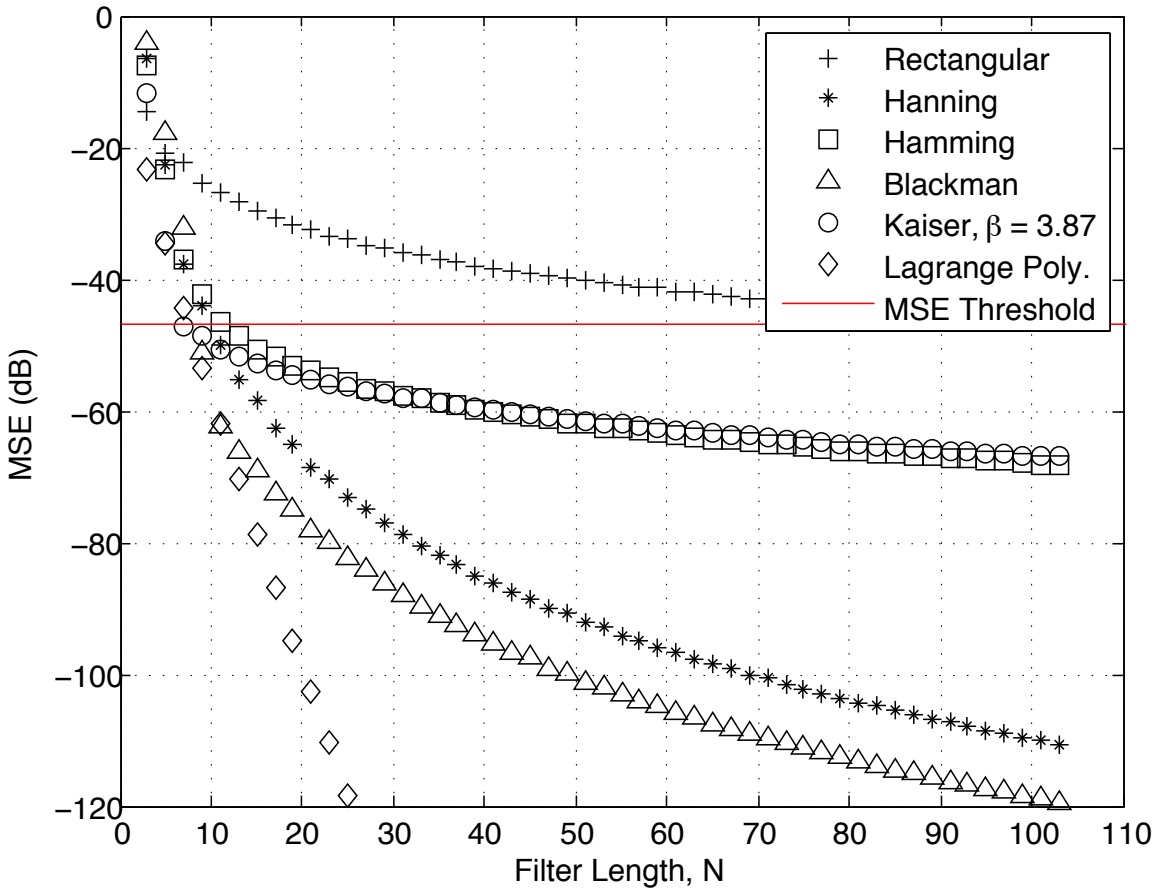


Figure 4.16: Secondary Path: Worst Case MSE ($\Delta = 0.5$ samples) vs. Odd Filter Length, Signal Upsampled by $L = 2$

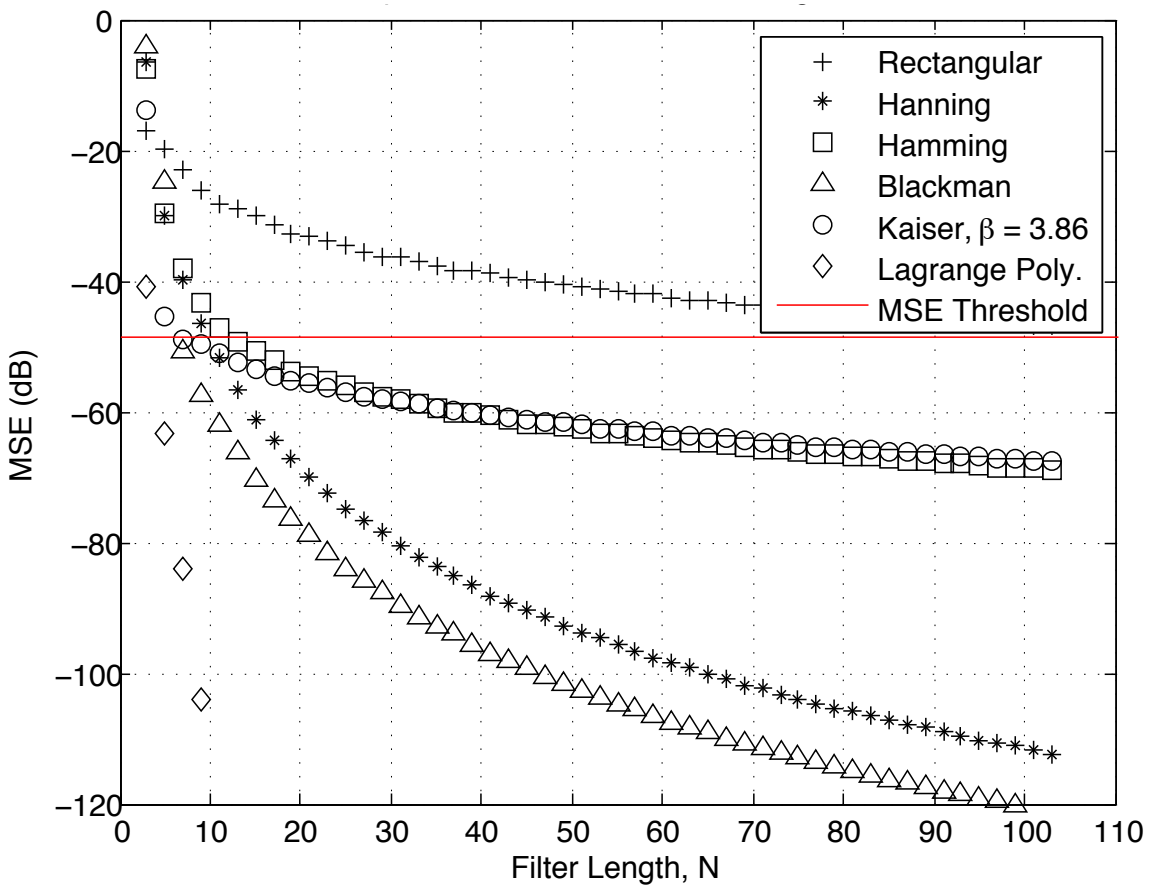


Figure 4.17: Secondary Path: Worst Case MSE ($\Delta = 0.5$ samples) vs. Odd Filter Length, Signal Upsampled by $L = 4$

Table 4.1: Main Path Filter Designs: Performance and Cost Analysis

Design #1 (No Upsampling)				
Filter	**MSE (dB)**	**Multipliers**	**Memory (bits)**	**Memory (M10K Blocks)**
Kaiser Windowed Sinc FD, β = 4.790	-60.0	43	198144	22
Totals	**-60.0**	**43**	**198144**	**22**

Design #2 (Upsample by *L*=2)				
Filter	**MSE (dB)**	**Multipliers**	**Memory (bits)**	**Memory (M10K Blocks)**
Equiripple Halfband	-64.5	25	450	5
Kaiser Windowed Sinc FD, β = 6.60	-63.1	9	41472	
Totals	**-60.7**	**34**	**41922**	**5**

Design #3 (Upsample by *L*=4)				
Filter	**MSE (dB)**	**Multipliers**	**Memory (bits)**	**Memory (M10K Blocks)**
Equiripple Halfband #1	-68.7	26	468	4
Equiripple Halfband #2	-78.2	4	72	
Lagrange Polynomial FD	-84.1	7	32256	
Totals	**-68.1**	**37**	**32796**	**4**

Table 4.2: Secondary Path Filter Designs: Performance and Cost Analysis

Design #1 (No Upsampling)				
Filter	**MSE (dB)**	**Multipliers**	**Memory (bits)**	**Memory (M10K Blocks)**
Kaiser Windowed Sinc FD, β = 3.056	-44.0	29	133632	15
Totals	**-44.0**	**29**	**133632**	**15**

Design #2 (Upsample by *L*=2)				
Filter	**MSE (dB)**	**Multipliers**	**Memory (bits)**	**Memory (M10K Blocks)**
Equiripple Halfband	-47.1	17	306	4
Kaiser Windowed Sinc FD, β = 3.88	-50.5	7	32256	
Totals	**-45.5**	**24**	**32562**	**4**

Design #3 (Upsample by *L*=4)				
Filter	**MSE (dB)**	**Multipliers**	**Memory (bits)**	**Memory (M10K Blocks)**
Equiripple Halfband #1	-53.5	18	324	3
Equiripple Halfband #2	-53.5	3	54	
Lagrange Polynomial FD	-63.3	5	23040	
Totals	**-50.2**	**26**	**23418**	**3**

5. Conclusion and Future Work

5.1 Contributions and Results

The main contribution of this work is a theoretical model for a DOCSIS 3.1 cable modem combined with its upstream channel that is suitable for implementation in an field programmable gate array (FPGA) as an emulator. While the emulator itself was not constructed, the model was verified through simulation with MATLAB scripts. The MATLAB scripts simulate steady-state transmission in the upstream direction over a data-over-cable service interface specification (DOCSIS) 3.1 network. Since the channel and cable modem (CM) are modelled at baseband, upconversion and downconversion are eliminated. This allows the emulator to be placed on the same FPGA as the receiver while debugging the receiver.

The transmitter portion of the model generates pseudo-random data sequences for a user-defined number of CMs. It is capable of formatting the data sequences into orthogonal frequency-division multiple access (OFDMA) frames according to one of three transmission modes outlined in the DOCSIS 3.1 standard: traffic mode, fine ranging mode, or probing mode. The resulting OFDMA frames are converted into time-domain sequences for use with the upstream channel model.

The upstream channel model is capable of simulating impairments that are specific to each CM, as well as impairments that are common to all CMs on a DOCSIS 3.1 network. CM-specific impairments include integer and fractional timing offsets, micro-reflections, carrier phase offset (CPO), fractional carrier frequency offset (CFO), and network gain/attenuation. Common channel impairments include carrier hum modulation, AM/FM ingress noise, and

additive white Gaussian noise (AWGN).

It was found that micro-reflections could be emulated with a fractional delay filter (FDF). With the aim that the emulator, i.e. the MATLAB model, be implemented on an field programmable gate array (FPGA), efficient FDF structures were researched and compared from an FPGA implementation cost perspective. The filter structure of Design #2 from Table 4.1 uses the least number of multipliers relative to any other design candidate, making it the most promising from an FPGA implementation perspective.

Furthermore, the MATLAB scripts developed in this thesis can be used in industry at the stage where the receiver is simulated in MATLAB. It provides a means for developers to test DOCSIS 3.1 receiver algorithms, such as those used for timing recovery and channel estimation.

5.2 Future Work

Due to the large scope of the DOCSIS 3.1 standard, it was decided that only the most critical features required to simulate upstream transmission should be modelled for the purposes of this thesis. It is suggested that the "cross-shaped" quadrature amplitude modulation (QAM) constellations (i.e. 8-QAM, 32-QAM, 128-QAM, 512-QAM, 2048-QAM) as well as forward error correction (FEC) via low-density parity check (LDPC) encoding be incorporated into the MATLAB model in the future.

To test FEC/LDPC encoding would require both burst and impulse noise be added to the channel model. Burst noise is, as its names suggests, a short wideband burst of noise. It is typically caused by some electrical disturbance in the home, for example turning on a blender. Impulse noise is shorter and stronger. It could be caused by lightning striking the ground near the coaxial cable. Normally, both of these noise types severely corrupt any data sent during the time that they are present, rendering it unrecoverable at the receiver. However, if FEC/LDPC encoding is implemented, the chances of recovering the corrupted data are significantly higher.

One other channel impairment that may be worth implementing in the future is phase

noise. Phase noise is produced by the oscillators that are used in the CMs and the cable modem termination system (CMTS) to upconvert and downconvert the baseband signal to and from radio frequency, respectively [39]. Since OFDMA uses multiple subcarriers that are spaced closely together, it is likely that phase noise will have an interesting effect on the received OFDMA signal.

In Chapter 4, infinite impulse response (IIR) filters were excluded from the research as possible FDF design candidates. This was due to the fact that their impulse response will extend beyond the cyclic prefix, which could affect the data of an OFDMA frame in an undesirable way. However, if the impulse response becomes small enough by the time it crosses over into the OFDMA frame, its affects may be negligible. If this is the case, then IIR filters may also be a viable means of approximating an ideal FDF. In fact, according to [31], IIR filters generally require fewer multiplies to implement compared to finite impulse response (FIR) filters, which can potentially make them more cost-effective when considering an FPGA implementation. It may be worth investigating IIR filter designs and performing a cost analysis on them.

A. MATLAB Variable Descriptions

This appendix contains the tables of MATLAB variables referenced in Chapter 3. With the exception of Table A.8, each table is formatted as follows:

Name	Description	Format	Values
Variable name as it appears in the MATLAB simulation.	A description of the variable.	Format in which the variable is stored (i.e. integer, array, etc.)	The value or range of values pertaining to the variable. Instructions on how to declare the variable and/or recommended settings may be included here as well.

In the case of Table A.8, which lists the "Output and Debug Variables," any relevant information pertaining to the values of the variables is provided in the "Description" of the second column. The fourth column instead shows which variables are required outputs or optional for debugging.

Note that should the user desire to correct for the integer timing, frequency, or network gain (attenuation) offsets between frames, their respective offset correction variables provided in Table A.4 (`Timing_int`, `Freq_adj`, `Pwr_offset`) must be declared in the MATLAB workspace by the user prior to calling the transmitter function.

Table A.1: General OFDMA Frame Setup Variables

Name	Description	Format	Values
Num_frames	The number of frames to simulate	Integer	0, 1, 2, …
Num_CM	The number of CMs connected to the system, regardless of whether or not they are transmitting	Integer	0, 1, 2, …
TranMode	IFFT size of the frame	Integer	0 = 2k Mode (IFFT size 2048) 1 = 4k Mode (IFFT size 4096)
CM_mode	Transmission mode of each CM	Array of size Num_CM	0 = Do Not Transmit 1 = Traffic 2 = Fine Ranging 3 = Probing
SC2k / SC4k	Subcarrier mappings for the frame	Array of size 2048 or 4096	0 = Excluded Subcarrier -1 = Unused Subcarrier 1, 2, 3, … = Index of Transmitting CM
Mod2k / Mod4k	QAM constellation orders for the frame	Array of size 2048 or 4096	1 - 12 = BPSK to 4096-QAM (2^order = Mod Type) Note: Only 1, 2, 4, 6, 8, 10, 12 available in the current simulation
Pilot2k / Pilot4k	Pilot mappings for the frame	Array of size 2048 or 4096	1 - 7 = Pilot Patterns 1 - 7 (2k Mode) 8 - 14 = Pilot Patterns 8 - 14 (4k Mode)
K	Number of OFDMA frames	Integer	6 - 36 (2k Mode) 6 - 18 (4k Mode)
N_CP	Cyclic Prefix	Integer	96, 128, 160, 192, 224, 256, 288, 320, 384, 512, 640
N_RP	Roll-off Period	Integer	0, 32, 64, 96, 128, 160, 192, 224

Table A.2: Fine Ranging Variables

Name	Description	Format	Values
N_fr	Number of Fine Ranging subcarriers allocated to each CM in Fine Ranging Mode	Array of size Num_CM	<= 512 for 2k Mode <= 256 for 4k Mode -Must span <= 512 subcarriers including excluded subcarriers in either 2k or 4k Mode
N_gb	Number of Guard Band subcarriers allocated to each CM in Fine Ranging Mode	Array of size Num_CM	-Must be an even number -Must be distributed equally amongst both sides of the Fine Ranging subcarriers (N_gb/2 on either side) N_gb = M*Q - N_fr, where M is the number of minislots allocated to Fine Ranging and Q is the number of subcarriers per minislot (8 or 16, 2k or 4k Mode)
Preamble	Preamble sequence for Fine Ranging	Array of size N_fr	-Must span less than N_fr subcarriers

Table A.3: Probing Frame Variables

Name	Description	Format	Values
Sym_in_frame	Index of the OFDMA symbol on which probing starts for a specific CM	Array of size Num_CM	0 - K-1
Stagger	Determines whether or not a CM is probed using a staggered	Array of size Num_CM	0 = No staggering 1 = Staggering
start_SC	Subcarrier index on which probing begins	Array of size	0 - 7
SC_skip	Number of subcarriers each CM will skip in between probing	Array of size Num_CM	0 - 7
PW_flag	Specifies whether or not to use alternate power settings during probing based on the value of start_SC NOTE: This setting has not been rigorously tested in the current version of the MATLAB model, and should therefore be set = 0 in all test cases	Array of size Num_CM	0 = Transmit using regular power settings 1 = Use alternate power settings specified by the values of start_SC listed below: 0 = reduce power per subcarrier by 2 dB 1 = reduce power per subcarrier by 3 dB 2 = reduce power per subcarrier by 4 dB 3 = reduce power per subcarrier by 5 dB 4 = reduce power per subcarrier by 6 dB 5 = reduce power per subcarrier by 7 dB 6 = reduce power per subcarrier by 8 dB 7 = reduce power per subcarrier by 9 dB
EQ_flag	Determines whether or not to use transmit equalization	Array of size Num_CM	0 = Equalizer enabled 1 = Equalizer disabled Note: May conflict with Fractional Timing Offset, which uses pre-equalization coefficients. Use caution when disabling.

Table A.4: Offset and Offset Correction Variables

Name	Description	Format	Values
Timing_int_offset	Randombly generated integer timing offsets for each CM (samples)	Array of size Num_CM	0 to 2048-N_RP (2k Mode) 0 to 4096-N_RP (4k Mode)
Timing_int	Correctional factors for integer timing offsets of each CM	Array of size Num_CM	0 to 2048-N_RP (2k Mode) 0 to 4096-N_RP (4k Mode)
Timing_frac_offset	Randombly generated fractional timing offsets for each CM	Array of size Num_CM	-0.5 to 0.5 2^-8 resolution
Phase_offset	Randomly generated carrier phase offset	Array of size	-π to π
Freq_offset	Randomly generated carrier frequency offsets for each CM (cycles/samples)	Array of size Num_CM	-0.5 to 0.5 10^-3 resolution
Freq_adj	Correctional factors for carrier frequency offsets of each CM (cycles/samples)	Array of size Num_CM	-0.01 to 0.01 10^-3 resolution
Pwr_offset	Randombly generated network gain for each CM (dB)	Array of size Num_CM	-9 to 3 10^-2 resolution
Pwr_adj	Correctional factors for network gain of each CM (dB)	Array of size Num_CM	-9 to 3 10^-2 resolution
Eq_set_2k / Eq_set_4k	Pre-equalization coefficients for each CM (each row contains a specific CM's coefficients)	Array of size Num_CM x 2048 or Num_CM x 4096	Complex coefficients Note: If fractional timing offset is enabled, these coefficients will be used to induce it. Thus, to correct for the fractional timing offset, any receiver function must adjust coefficients appropriately.

Table A.5: Channel Setup Variables (1/3)

Name	Description	Format	Values
Microref_en	Enable micro-reflections	Integer	0 = Disabled 1 = Enabled
Microref_num	Number of micro-reflections per CM	Integer	0 to 6 Note: DOCSIS 3.1 standard accounts for only one micro-reflection
Microref_time_limit	Maximum echo time of a micro-reflection (microseconds)	Double	<= 800 Recommended value = 6
Timing_int_offset_en	Enable integer timing offset	Integer	0 = Disabled 1 = Enabled
Timing_frac_offset_en	Enable fractional timing offset	Integer	0 = Disabled 1 = Enabled
Phase_offset_en	Enable carrier phase offset	Integer	0 = Disabled 1 = Enabled
Frame_tag_mode	Determines whether or not the beginning of the first frame in a transmission burst should be concatenated with a seequence of zeros or a duplicate of the first frame	Integer	0 = Fill the additional array elements at the start and end of the current frame with zeros. 1 = Concatenate the beginning of the first frame with the end of a copy of the first frame
Freq_offset_en	Enable carrier frequency offset	Integer	0 = Disabled 1 = Enabled
Hum_en	Enable AM carrier hum modulation	Integer	0 = Disabled 1 = Enabled
Hum_freq	Carrier hum waveform frequency (Hz)	Double	Recommended value = 60
Hum_trans_time	Transition time of carrier hum waveform (seconds)	Double	Recommended value = 0.25*10^-3
Hum_phase	Phase of carrier hum waveform (rads)	Double	-π to π

Table A.6: Channel Setup Variables (2/3)

Hum_atten_dBc	Attenuation of carrier hum waveform (dBc)	Double	Recommended value <= -26
Ingress_en	Enable ingress noise	Integer	0 = Disabled 1 = Enabled Note: When ingress noise is enabled, the user must specify the charateristics of each individual ingress noise signal separately in their respective row vectors. For example, if there are three ingress noise signals present, each of the following vectors will take on the form: [x, y, z] where x, y, and z each represent their own signal.
Ingress_type	Vector specifying the type of each individual ingress noise signal	Row vector of integers	0 = "Block" of frequencies 1 = AM 2 = FM
Ingress_carrier_freq	Vector specifying the carrier frequency of each individual ingress noise signal (Hz)	Row vector of doubles	-51.2*10^6 to 51.2*10^6 Note 1: The user should ensure that the bandwidth of the ingress signal does not exceed these boundaries. Note 2: The carrier frequency of a "Block" ingress is the center frequency of the block itself.
Ingress_carrier_phase	Vector specifying the carrier frequency	Row vector of doubles	-π to π
Ingress_BW	Vector specifying the bandwidth of each individual ingress noise signal (Hz)	Row vector of doubles	Recommended value = 200*10^3 ("Block", AM or FM)
Ingress_freq_spacing	Vector specifying the spacing between frequencies in "Block" ingress noise signals (Hz)	Row vector of doubles	Must be smaller than half the bandwidth. Only applicable to "Block" ingress. If no "Block" ingress is present, all elements must be zeros.
Ingress_FM_freq	Vector specifying the frequency gain constant in V/Hz, used to calculate peak frequency deviation in FM message signals	Row vector of doubles	Recommended value = 75*10^3

Table A.7: Channel Setup Variables (3/3)

Ingress_carrier_phase	Vector specifying the carrier frequency	Row vector of doubles	-π to π
Ingress_BW	Vector specifying the bandwidth of each individual ingress noise signal (Hz)	Row vector of doubles	Recommended value = 200*10^3 ("Block", AM or FM)
Ingress_freq_spacing	Vector specifying the spacing between frequencies in "Block" ingress noise signals (Hz)	Row vector of doubles	Must be smaller than half the bandwidth. Only applicable to "Block" ingress. If no "Block" ingress is present, all elements must be zeros.
Ingress_FM_freq	Vector specifying the frequency gain constant in V/Hz, used to calculate peak frequency deviation in FM message signals	Row vector of doubles	Recommended value = 75*10^3
Ingress_num_message_sig	Vector specifying the number of sinusoids an AM or FM message signal	Row vector of integers	0, 1, 2, … Only applicable to AM or FM ingress. If no AM or FM ingresses are present, all elements must be zeros.
Ingress_atten_dBc	Vector specifying the carrier attenuation in each ingress noise	Row vector of doubles	< 0 Example = -40
Noise_en	Enable AWGN	Integer	0 = Disabled 1 = Enabled
CNIR	Carrier-to-noise ratio (dB)	Double	> 0 Example = 40

Table A.8: Output and Debug Variables

Name	Description	Format	Type
b_orig_CM_frame	Cell array containing Num_CM cell arrays, which in turn contain the bit sequences transmitted by each CM in traffice mode over a single frame	Cell array of size Num_frames	Required Output
Complete_Frame	The number of CMs connected to the system, regardless of whether or not they are transmitting	Cell array of size Num_frames	Optional Debug Output
Complete_Frame_Probing _CM_index	Cell array containing all 2048 x K or 4096 x K OFDMA frame structures for probing frames, with CM indices replacing the pilot symbols	Cell array of size Num_frames	Optional Debug Output
tx_CM_frame	Cell array containing Num_CM column vectors, which in turn contain the Tx signals of each CM in each frame	Cell array of size Num_frames	Required Output
tx_frame	Cell array containing the combined Tx signals of all CMs in each frame	Cell array of size Num_frames	Optional Debug Output
rx_CM_frame	Cell array containing Num_CM column vectors, which in turn contain the Rx signals of each CM in each frame	Cell array of size Num_frames	Required Output
rx_frame	Cell array containing the combined Rx signals of all CMs in each frame	Cell array of size Num_frames	Required Output

B. Upsample by $L = 4$ Single-Sampling-Rate Structure Example

This appendix contains the example that is referenced in Section 4.4.3, which involves two finite impulse response (FIR) halfband filters that are cascaded in a single-sampling-rate structure to upsample a signal by a factor of $L = 4$. A block diagram of the cascade is provided in Figure B.1. The numbers used in this example come from main-path filter Design #3, which was described in Section 4.6. Referring to Figure B.1, $H_{HB1}(z)$ has a length of $N_{HB1} = 103$ coefficients and a transition width of 0.0361328125 cycles/samples. Filter $H_{HB2}(z)$ has a length of $N_{HB2} = 15$ coefficients and a transition width of 0.268066406 cycles/samples. Both filters have a stop-band attenuation of 74 dB, which translates to a worst-case combined MSE of -68.2 dB.

$H_{HB1}(z)$ is decomposed into polyphase components $H_0(z)$ and $H_1(z)$. $H_0(z)$ is a filter of length $N_0 = (N_{HB1} + 1)/2 = 52$ coefficients and delay $D_0 = (N_0 - 1)/2 = 25.5$ coefficients, while $H_1(z)$ is a filter of length $N_1 = (N_1 - 1)/2 = 51$ which actually translates to a pure integer delay of $D_1 = (N_1 - 1)/2 = 25$ samples. The output of $H_0(z)$ is stream $u_0[n]$, which is delayed by a factor of 0.5 samples relative to stream $u_1[n]$, the output of filter $H_1(z)$.

$H_{HB2}(z)$ is decomposed into polyphase components $H_2(z)$ and $H_3(z)$. $H_2(z)$ is a sub-filter of length $N_2 = (N_{HB2} + 1)/2 = 8$ coefficients and delay $D_2 = (N_2 - 1)/2 = 3.5$ coefficients, while $H_1(z)$ is a sub-filter of length $N_3 = (N_{HB2} - 1)/2 = 7$ coefficients, which actually translates to a pure integer delay of $D_3 = (N_3 - 1)/2 = 3$ samples.

In order to produce $L = 4$ streams at the output that are fractionally delayed by a factor of $1/L$ relative to each other, sub-filters $H_2(z)$ and $H_3(z)$ must be further decomposed. $H_2(z)$

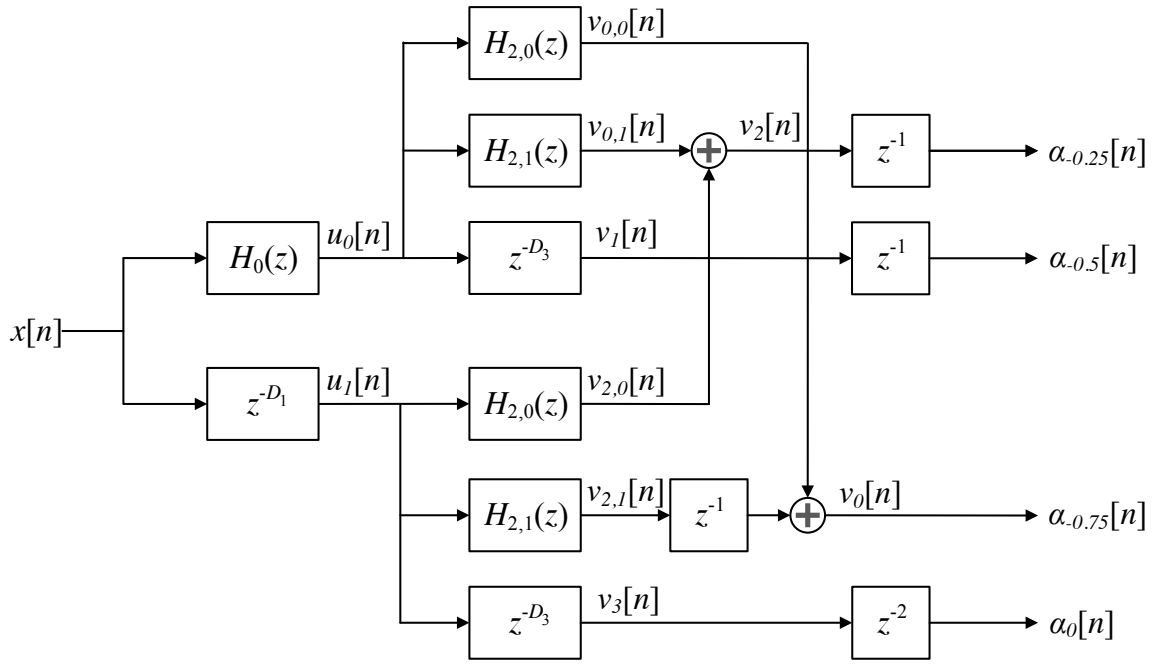


Figure B.1: Single-Sampling-Rate FIR Halfband Cascade, $L = 4$

thus becomes sub-filters $H_{2,0}(z)$ and $H_{2,1}(z)$, both of which are of length $N_{2,X} = N_2/2 = 4$ coefficients and delay $D_{2,X} = (N_{2,X} - 1)/2 = 1.5$ samples. Since $H_{3,X}(z)$ is actually a pure integer delay path, its decomposition is merely a reduction in delay from 3 samples to 1.5 samples. However, since it is not possible to implement a pure half-sample of delay, the delay is rounded up to 2 samples.

Following further decomposition, streams $u_0[n]$ and $u_1[n]$ are each connected to their own chain of filters consisting of $H_{2,0}(z)$, $H_{2,1}(z)$ and the delay path $H_{3,X}(z)$. This results in a total of 6 output streams. The output of sub-filters $H_{2,0}(z)$, $H_{2,1}(z)$ and the delay path $H_{3,X}(z)$ to which stream $u_0[n]$ is connected are denoted as streams $v_{0,0}[n]$, $v_{0,1}[n]$, and $v_1[n]$, respectively. The output of sub-filters $H_{2,0}(z)$, $H_{2,1}(z)$ and the delay path $H_{3,X}(z)$ to which stream $u_1[n]$ is connected are denoted as streams $v_{2,0}[n]$, $v_{2,1}[n]$, and $v_3[n]$, respectively.

Although 6 output streams are present, the desired output of the structure is $L = 4$ streams that are fractionally delayed by a factor of $1/L$ relative to each other. While not mandatory, it is convenient to define these streams as having 0, -0.25, -0.5, and -0.75 samples, respectively, relative to a given integer delay. At this point, the total delay along each path

is follows:

$$D_{0,0}^{tot}[n] = 25.5 + 1.5 = 27 \text{ samples}$$
$$D_{0,1}^{tot}[n] = 25.5 + 1.5 = 27 \text{ samples}$$
$$D_{1}^{tot}[n] = 25.5 + 2 = 27.5 \text{ samples}$$
$$D_{2,0}^{tot}[n] = 25 + 1.5 = 26.5 \text{ samples}$$
$$D_{2,1}^{tot}[n] = 25 + 1.5 = 26.5 \text{ samples}$$
$$D_{3}^{tot}[n] = 25 + 2 = 27 \text{ samples}$$

Upon examining Figure B.1, it is clear that streams $v_1[n]$ and $v_3[n]$ are streams with 0.5 and 0 samples of fractional delay, respectively. Adding stream $v_{0,1}[n]$ to stream $v_{2,0}[n]$ produces stream $v_2[n]$, which is fractionally delayed by a factor of 0.75 samples. Inserting an integer delay along the path of stream $v_{2,1}[n]$ and adding it to stream $v_{0,0}[n]$ produces stream $v_0[n]$, which is fractionally delayed by a factor of 0.25 samples.

Accounting for the delays introduced by the adders, the total delay along each path is now:

$$D_{0}^{tot}[n] = 28.25 \text{ samples}$$
$$D_{1}^{tot}[n] = 27.5 \text{ samples}$$
$$D_{2}^{tot}[n] = 27.75 \text{ samples}$$
$$D_{3}^{tot}[n] = 27 \text{ samples}$$

Finally, by strategically inserting integer delays along select paths as shown in Figure B.1, the output streams can be reorganized such that they are fractionally delayed by 0, -0.25, -0.5, and -0.75 samples relative to an integer delay of 33 samples. The paths are relabelled to match this new scheme using the notation α_X, where X is the relative fractional delay.

$$\alpha_0 = v_3[n]z^{-2}$$
$$\alpha_{-0.25} = v_2[n]z^{-1}$$
$$\alpha_{-0.5} = v_1[n]z^{-1}$$
$$\alpha_{-0.75} = v_0$$

Thus, the final delay along each path is:

$$D_0^{tot}[n] = 29 \text{ samples}$$

$$D_{-0.25}^{tot}[n] = 28.75 \text{ samples}$$

$$D_{-0.5}^{tot}[n] = 28.5 \text{ samples}$$

$$D_{-0.75}^{tot}[n] = 28.25 \text{ samples}$$

References

[1] E. S. Smith, "The emergence of cable TV: A look at the evolution of a revolution," in *Proc. IEEE*, vol. 58, pp. 967–982, July 1970.

[2] E. Champy, "Broadband wireless connectivity," in *Proc. Wescon'97*, (Nashua, NH, USA), pp. 200–205, Nov. 1997.

[3] W. S. Ciciora, "An introduction to cable television in the United States," *IEEE LCS Magazine*, vol. 1, pp. 19–25, Feb. 1990.

[4] J. S. Light, "Before the internet, there was cable," *IEEE Annals of the History of Computing*, vol. 25, pp. 94–96, Apr. 2003.

[5] E. K. Smith, "Pilot two-way CATV systems," *IEEE Transactions on Communications*, vol. 23, pp. 111–120, Jan. 1975.

[6] V. Brugliera, "History of compatibility between cable systems and receivers," in *Proc. IEEE International Conference on Consumer Electronics*, (Rosemont, IL, USA), pp. 186–187, Nov. 1994.

[7] J. M. Cioffi, "Lighting up copper," *IEEE Communications Magazine*, vol. 49, pp. 30–43, May 1990.

[8] CableLabs, "DOCSIS 1.0 Radio Frequency Interface Specification, SP-RFI-C01-011119," Nov. 2001.

[9] CableLabs, "DOCSIS 1.1 Radio Frequency Interface Specification, CM-SP-RFIv1.1-C01-050907," Sept. 2005.

[10] CableLabs, "DOCSIS 2.0 + IPv6 Cable Modem Specification, CM-SP-DOCSIS2.0-IPv6-I07-130404," Apr. 2013.

[11] CableLabs, "DOCSIS 3.0 Physical Layer Specification, CM-SP-PHYv3.0-I11-130808," Dec. 2008.

[12] CableLabs, "DOCSIS 3.1 Physical Layer Specification, CM-SP-PHYv3.1-I04-141218," Dec. 2014.

[13] H. Yin and S. Alamouti, "OFDMA: A broadband wireless access technology," in *Sarnoff Symposium, IEEE*, (Princeton, NJ, USA), pp. 1–4, Nov. 2006.

[14] C. Ciochina and H. Sari, "A review of OFDMA and single-carrier FDMA and some recent results," *IEEE Advances in Electronics and Telecommunications*, vol. 1, pp. 35–40, Apr. 2010.

[15] CableLabs, "DOCSIS 3.1 MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.1-I04-141218," Dec. 2014.

[16] Altera, "FFT MegaCore Function User Guide," Dec. 2014.

[17] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, ch. 2, pp. 22–24. Cambridge University Press, 2005.

[18] M. Morelli, C. C. J. Kuo, and M.-O. Pun, "Synchronization techniques for orthogonal frequency division multiple access (OFDMA): a tutoral review," in *Proc. IEEE*, vol. 95, pp. 1394–1427, July 2007.

[19] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for ofdm," *IEEE Trans. Commun.*, vol. 45, pp. 1613–1621, Dec. 1997.

[20] D. Large and J. Farmer, *Broadband Cable Acess Networks: The HFC Plant*, ch. 7, pp. 200–201. Morgan Kaufmann, 1999.

[21] B. Berscheid, *FPGA-Based DOCSIS Upstream Demodulation.* PhD thesis, University of Saskatchewan, Saskatoon, SK, Canada, July 2011.

[22] X. Dai, "Carrier frequency offset estimation and correction for OFDMA uplink," *IET Communications*, vol. 1, pp. 273–281, Apr. 2007.

[23] K. K. Parhi and T. Nishitami, eds., *Digital Signal Processing for Multimedia Systems.* New York, NY, USA: Marcel Dekker Inc., 1999.

[24] D. Large and J. Farmer, *Broadband Cable Acess Networks: The HFC Plant*, ch. 2, p. 37. Morgan Kaufmann, 1999.

[25] S. Ovadia, *Broadband Cable TV Acess Networks: From Technologies to Applications*. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[26] R. Osso, *Handbook of Emerging Communications Technologies: The Next Decade*, ch. 5, p. 94. CRC Press, 2000.

[27] F. Wang, G. Yang, W. W. Fang, and J. Liu, "Statistical analysis of the effective bandwidth of fm hd radio," in *Control Engineering and Information Systems* (Z. Liu, ed.), pp. 453–457, CRC Press, 2015.

[28] H. H. Nguyen and E. Shwedyk, *A First Course in Digital Communications*, ch. 3, pp. 104–106. Cambridge University Press, 2009.

[29] J. Diaz-Carmona and G. J. Dolecek, "Fractional delay filters," in *Applications of MATLAB in Science and Engineering* (T. Michaowski, ed.), ch. 12, InTech, 2011.

[30] V. Valimaki and T. T. Laakso, "Principles of fractional delay filters," in *Proc. ICASSP'00*, (Istanbul, Turkey), pp. 1–4, June 2000.

[31] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay: Tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, pp. 30–60, Jan. 1996.

[32] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, ch. 7, pp. 465–486. Upper Saddle River, NJ, USA: Prentice Hall, second ed., 1999.

[33] E. Hermanowicz, "Explicit formulas for weighting coefficients of maximally flat tunable FIR delayers," *Electronic Letters*, vol. 28, pp. 1936–1937, Sept. 1992.

[34] F. H. H. Johansson, "Polyphase decomposition of digital fractional-delay filters," *Signal Processing Letters, IEEE*, vol. 22, pp. 1021–1025, Aug. 2015.

[35] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial," in *Proc. IEEE*, pp. 56–93, Jan. 1990.

[36] G. Ramirez-Conejo, J. D. Carmona, A. Ramirez-Agundis, A. Padilla-Medina, and J. Delgado-Frias, "FPGA implementation of adjustable wideband fractional delay FIR filters," in *International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, (Quintana Roo, Mexico), pp. 406–411, Dec. 2010.

[37] H. Johansson, O. Gustafsson, K. Johansson, and L. Wanhammar, "Adjustable fractional-delay FIR filters using the Farrow structure and multirate techniques," in *APCCAS'06*, (Singapore), pp. 1055–1058, Dec. 2006.

[38] F. J. Harris, *Multirate Signal Processing for Communications Systems*, ch. 8. Upper Saddle River, NJ, USA: Prentice Hall, 2004.

[39] M. S. El-Tanay and Y. Wu, "Impact of phase noise on the performance of OFDM systems over frequency selective channels," in *Proc. Wescon'97*, vol. 3, (Phoenix, NH, USA), pp. 1802–1806, May 1997.